Qucs

Report Book
Technical reports and descriptions

Mike Brinson
Stefan Jahn
Hélène Parruitte

Contents

6

1 Verilog-AMS interface

1.1 Introduction

Verilog-AMS is a hardware description language. It can be used to specify the analogue behaviour of compact device models. Usually these are C or C++ implementations in analogue simulators. The effort to implement modern compact models in C/C++ is quite high compared to the description in Verilog-AMS.

1.2 ADMS

The software ADMS (see `http://mot-adms.sourceforge.net`) allows Verilog-AMS descriptions to be translated into any other programming language. It generates a structured XML tree representing the compact device model description.
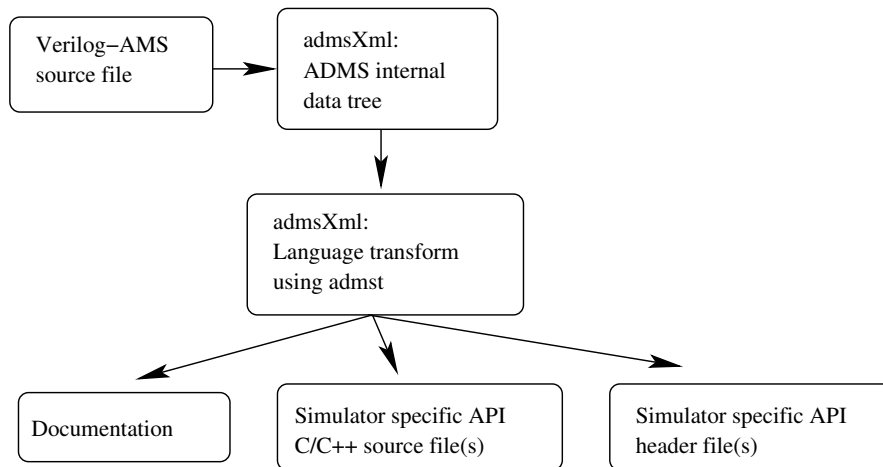
Figure 1.1: ADMS data flow

The internal XML tree is used to generate ready-to-compile C or C++ code which is specific to the simulators API. The code generator is able to produce

- evaluation of device equations (current and charge) including their derivatives,

- glue code for the simulator API,

- documentation and

- any other data described by the original Verilog-AMS input file.

1.3 XML admst scripts

The language transformation uses a language named **admst**. It is itself a XML description. The command line in order to run a transformation is

```
$ admsXml <device.va> -e <interface-1.xml> -e <interface-2.xml>
```

From version 0.0.11 Qucs comes with the following Verilog-AMS transformers

- `qucsMODULEcore.xml`
  creating the actual analogue simulator implementation

- `qucsMODULEdefs.xml`
  creating the parameter descriptions for the analogue simulator

- `qucsMODULEgui.xml`
  creating the implementation for the GUI integration

- `qucsVersion.xml`
  basic admst library

- `analogfunction.xml`
  creating analogue function code

In order to create **admst** scripts for a simulator it is necessary to understand both, the simulator specific API and how the ADMS data tree items – which are based on the Verilog-AMS source file describing the model – relate to the API.

The command lines for transforming a Verilog-AMS source file into the appropiate Qucs C++ source files are

```
$ admsXml <device.va> -e qucsVersion.xml -e qucsMODULEcore.xml
$ admsXml <device.va> -e qucsVersion.xml -e qucsMODULEgui.xml
$ admsXml <device.va> -e qucsVersion.xml -e qucsMODULEdefs.xml
$ admsXml <device.va> -e analogfunction.xml
```

each creating an appropriate `*.cpp` and `*.h` file.

The **admst** language is used to traverse the internal tree. The tree's root is defined by the Verilog-AMS module definition.

```
module device (node1, node2, ...)
  // module definitions and code
endmodule
```

1.3.1 Short introduction into the admst language syntax

Usually the **admst** language instructions are basically formed as

```
<admst:instruction argument=...>
  ...
</admst:instruction>
```

or

```
<admst:instruction argument=.../>
```

Any other text outside these instruction is output to the console or into an appropriate file. Some of the most important language construct are listed below.

- Traversing a list: The construct allows to traverse all children of the selected branch.

  ```
  <admst:for-each select="/module">
    ...
  </admst:for-each>
  ```

- Defining and using variables: Values from the data tree can be put into named and typed variables which are then accessible using the $ operator.

  ```
  <admst:value-of select="name">
  <admst:variable name="module" select="%s">
  ```

- Opening a file: Printed text (see below) is output into the given file.

  ```
  <admst:open file="$module.cpp">
    ...
  </admst:open>
  ```

- Output text: Special characters must be encoded (e.g. `"` → `"` `<` → `<` `>` → `>` and `\n` → `\\n`).

  ```
  <admst:text format="This is a text."/>
  ```

- Definition of a function: Functions in **admst** are called templates.

```
<admst:template match="name_of_function">
   . . .
</admst:template>
```

- Running a function: Templates are applied to parts of the internal data tree.

```
<admst:apply-templates select="date_tree_root_for_function"
                        match="name_of_function"/>
```

- Comments:

```
<!-- this is a comment -->
```

1.3.2 Analogue simulator script

The analogue simulator implementation consists of several parts. For each type of simulation appropriate functions must be implemented.

- DC simulation
  `module::initDC (void)`,
  `module::restartDC (void)` and
  `module::calcDC (void)`

- AC simulation
  `module::initAC (void)` and
  `module::calcAC (nr_double_t)`

- S-parameter simulation
  `module::initSP (void)` and
  `module::calcSP (nr_double_t)`

- Transient simulation
  `module::initTR (void)` and
  `module::calcTR (nr_double_t)`

- AC noise simulation
  `module::initNoiseAC (void)` and
  `module::calcNoiseAC (nr_double_t)`

- S-parameter noise simulation
  `module::initNoiseSP (void)` and
  `module::calcNoiseSP (nr_double_t)`

- Harmonic simulation
  `module::initHB (int)` and
  `module::calcHB (int)`

These functions go into the `module.core.cpp` file. An appropriate `module.core.h` header file is also created.

In case the Verilog-AMS source file contains analog functions in the module definition, e.g.

```
analog function real name;
   input x;
   real x;
   begin
      name = x * x;
   end
endfunction
```

the `analogfunction.xml` script produces the appropriate C/C++ and header files

- `module.analogfunction.cpp`

- `module.analogfunction.h`.

The ADMS language transformer is aware of how analogue simlators solve a network of linear and non-linear devices. The general Newton-Raphson algorithm for a DC simulation used in SPICE-like simulators as well as in Qucs can be expressed as

$$
\begin{aligned}
J^{(m)} \cdot V^{(m+1)} &= J^{(m)} \cdot V^{(m)} - f\left(V^{(m)}\right) \\
&= I_{lin}^{(m)} + I_{nl}^{(m)} - J_{nl}^{(m)} \cdot V^{(m)}
\end{aligned}
\tag{1.1}
$$

whereas J denotes the Jacobian

$$
J^{(m)} = \left.\frac{\partial f\left(V\right)}{\partial V}\right|_{V^{(m)}}
\tag{1.2}
$$

There are basically two types of contributions supported by ADMS: currents and charges. The appropiate Jacobian matrices are

$$
J^{(m)} = J_I^{(m)} + J_Q^{(m)} = \underbrace{\left.\frac{\partial I\left(V\right)}{\partial V}\right|_{V^{(m)}}}_{\text{"static"}} + \underbrace{\left.\frac{\partial Q\left(V\right)}{\partial V}\right|_{V^{(m)}}}_{\text{"dynamic"}} = G + C
\tag{1.3}
$$

consisting of two real valued matrices G (conductance/transconductance matrix) and C (capacitance/transcapacitance matrix).

In the Verilog-AMS desciptions only the current and charge contributions are mentioned. The appropriate derivatives are meant to be automatically formed by the language translator ADMS.

```
I(ci, ei) <+ it;        // current contribution
I(si, ci) <+ ddt(Qjs); // charge contribution
```

The right-hand side of eq. (1.1) consists of the linear and non-linear currents of the network as well as the Jacobian matrix multiplied by the voltage vector. Since devices are usually uncorrelated both parts can be computed directly on a per device basis.

1.3.3 Current limitations of ADMS

The following section contains some simple examples demonstrating which Verilog-AMS statements can be successfully handled by ADMS and which not.

Example 1

There was a bug in the **admst** scripts. They failed to place the function calls of analogue functions correctly. This has been fixed.

```
module diode(a, c);
  inout a, c;
  electrical a, c;

  analog function real current;
    input is, v;
    begin
      current = is * (exp (v / 26e-3) - 1);
    end
  endfunction

  real Vd;
  real Id;

  analog begin
    Vd = V(a, c);
    Id = current (1e-15, Vd);
    I(a, c) <+ Id;
  end

endmodule
```

Example 2

It is not allowed to mix current and charge contributions in assignments. Mixing both emits wrong C/C++ code. It accounts the current to be a charge in this case.

```
module diode(a, c);
  inout a, c;
  electrical a, c;

  real Vd, Id, Is, Cp;

  analog begin
    Is = 1e-15;
    Cp = 1e-12;
    Vd = V(a, c);
    Id = Is * (exp (Vd / 26e-3) - 1);

    // not allowed in ADMS
    I(a, c) <+ Id + ddt(Cp*V(a,c));

    // allowed in ADMS
    I(a, c) <+ Id;
    I(a, c) <+ ddt(Cp*V(a,c));
  end

endmodule
```

Example 3

The following example is rejected by the **admst** scripts with this error message.

```
[info] admsXml-2.2.4 Oct 18 2006 19:50:46
[fatal] qucsVersion.xml:1497:admst:if[lhs]: missing node source/insource
```

An immediate potential on the right hand side is not allowed embedded in a function.

```
module diode(a, c);
  inout a, c;
  electrical a, c;

  real Id, Is;
```

```
analog begin
  Is = 1e−15;
  Id = Is * (exp (V(a, c) / 26e−3) − 1);

  // not allowed in ADMS
  I(a, c) <+ Is * (exp (V(a, c) / 26e−3) − 1);

  // allowed in ADMS
  I(a, c) <+ Id;
end

endmodule
```

Example 4

Potentials on the left-hand side of contribution assignments are not allowed. This
kind of assignment emits wrong C/C++ code.

```
module diode(a, c);
  inout a, c;
  electrical a, c;

  real Id, Is;
  real Rp;

  analog begin
    Rp = 1e9;
    Is = 1e−15;
    Id = Is * (exp (V(a, c) / 26e−3) − 1);
    I(a, c) <+ Id;

    // allowed in ADMS
    I(a, c) <+ V(a, c) / Rp;

    // not allowed in ADMS
    V(a, c) <+ I(a, c) * Rp;
  end

endmodule
```

1.3.4 Code for the GUI integration

The **admst** script for the GUI creates files according to its API. Basically it produces the list of parameters as well their descriptions (if available) in a property list. Additionally the symbol drawing code is emitted which should be adapted to the devices requirements.
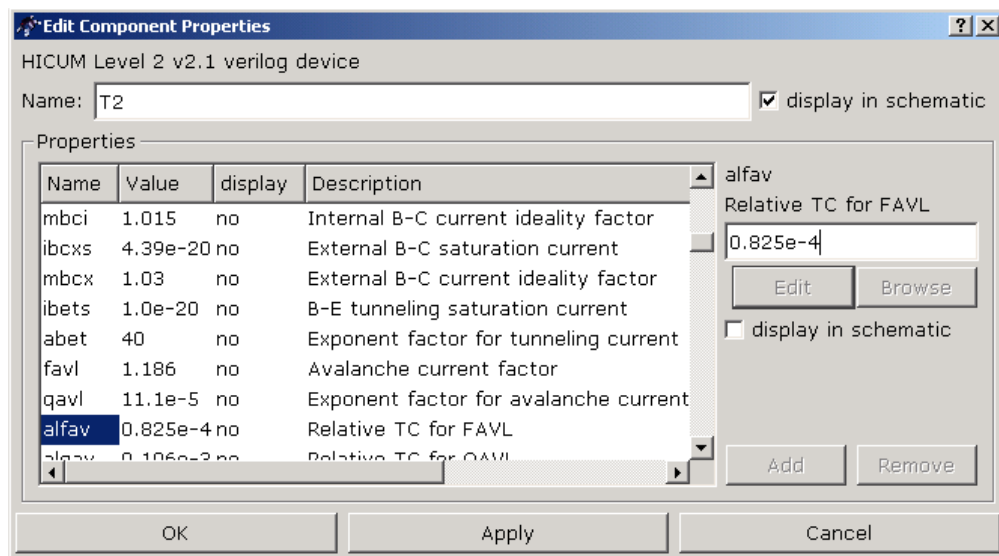
Figure 1.2: component property dialog in the GUI

In fig. 1.2 the component property dialog of an implemented device is shown. In the schematic in fig. 1.3 the symbol for the HICUM model is surrounded with a red circle. The GUI code is also responsible for the icons in the listview on the left hand side of the figure.
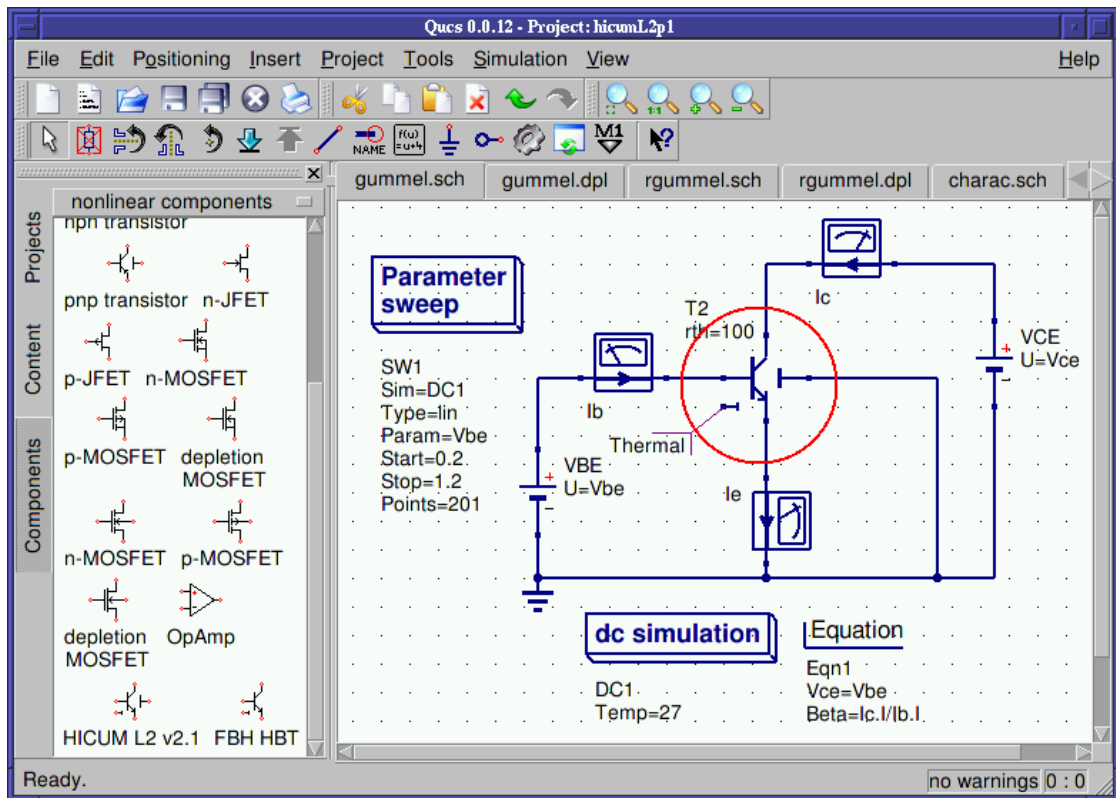
Figure 1.3: schematic with the HICUM transistor symbol

1.4 Adding Verilog-AMS devices to Qucs

The section gives an overview how to add a Verilog-AMS model to Qucs in a step-by-step manner. Be aware that the description is meant for advanced users partly familiar with the usual GNU/Linux build system and C/C++ programming.

1.4.1 The Verilog-AMS source file

The starting point of a new Verilog-AMS model in Qucs is the Verilog-AMS source file which must be aware of the discussed ADMS limitations. The example we are going to discuss is a simple diode model shown in the following listing contained in the **diode.va** file.

```
`include "disciplines.vams"
`include "constants.vams"

// ADMS specific definitions
```

```verilog
`define attr(txt) (*txt*)

module diode(a, c);
  // device terminals
  inout a, c;
  electrical a, c;

  // internal node
  electrical ci;

  // model parameters
  parameter real Is   = 1E-15 from [0:1]
    `attr(info="saturation_current");
  parameter real Cp   = 1E-12 from [0:1]
    `attr(info="parallel_capacitance");
  parameter real Rs   = 1.0   from (0:inf)
    `attr(info="series_resistance");
  parameter real Temp = 300   from (0:inf)
   `attr(info="temperature");

  real Vd, Id, fourkt, twoq, Qp;

  analog begin
    Vd = V(a, ci);
    Id = Is * (exp (Vd / 26e-3) - 1);
    Qp = Cp * Vd;

    I(a, ci) <+ Id;
    I(a, ci) <+ ddt(Qp);
    I(ci, c) <+ V(ci, c) / Rs;

  begin : noise
    fourkt = 4.0 * `P_K * Temp;
    twoq   = 2.0 * `P_Q;
    I(ci, c) <+ white_noise(fourkt/Rs, "thermal");
    I(a, ci) <+ white_noise(twoq*Id, "shot");
  end // noise

  end // analog
```

endmodule

1.4.2 Integrating the model into the analogue simulator

The qucs-core tree of Qucs must be configured using the –enable-maintainer-mode option.

```
$ ./configure --enable-maintainer-mode --prefix=/tmp
```

The source trees of qucs (GUI) as well as qucs-core (simulator) are within the release tarballs available at `http://qucs.sourceforge.net/download.html`. During development of new Verilog-A models it is recommended to use CVS. Latest CVS trees of qucs and qucs-core can be obtained using the following command lines.

```
$ cvs -z3 -d:pserver:anonymous@qucs.cvs.sourceforge.net:/cvsroot/qucs \
  co qucs-core
$ cvs -z3 -d:pserver:anonymous@qucs.cvs.sourceforge.net:/cvsroot/qucs \
  co qucs
```

Also the software **adms** must be installed. It is available at `http://sourceforge.net/projects/mot-adms`.

The file **diode.va** must be copied into the source tree.

```
$ cp diode.va qucs-core/src/components/verilog/
```

In this directory (qucs-core/src/components/verilog/) the Verilog model can be checked if it can be translated successfully using the following command lines.

```
$ admsXml diode.va -e qucsVersion.xml -e qucsMODULEcore.xml
[info] admsXml-2.2.4 Oct 18 2006 19:50:46
[info] diode.core.cpp and diode.core.h: files created
[info] elapsed time: 0.0339946
[info] admst iterations: 4146 (4146 freed)
$ admsXml diode.va -e analogfunction.xml
[info] admsXml-2.2.4 Oct 18 2006 19:50:46
[info] diode.analogfunction.h created
[info] diode.analogfunction.cpp created
[info] elapsed time: 0.0262961
[info] admst iterations: 3919 (3919 freed)
```

These command lines create the files for the model evaluation code. The file names (diode.*) are due to the name of the module contained in the Verilog-AMS source file.

Additionally the source code must be changed in some more locations.

- `src/components/component.h`
  In this file it is necessary to add the line

  ```
  #include "verilog/diode.core.h"
  ```

- `src/components/component_id.h`
  The file contains unique component identifiers. It is necessary to add the Verilog modules name.

  ```
  enum circuit_type {
    CIR_UNKNOWN = -1,
    ...
    // verilog devices
    CIR_diode,
    ...
  };
  ```

- `src/input.cpp`
  In order to be able to instantiate the new model the file must be modified as follows.

  ```
  // The function creates components specified by the type of component.
  circuit * input::createCircuit (char * type) {
    if (!strcmp (type, "Pac"))
      return new pac ();
    ...
    else if (!strcmp (type, "diode"))
      return new diode ();

    logprint (LOG_ERROR, "no such circuit type '%s'\n", type);
    return NULL;
  }
  ```

- `src/qucsdefs.h`
  Finally the properties including their range definitions must be added to this file. The appropiate file can be obtained using the following command line.

  ```
  $ admsXml diode.va -e qucsVersion.xml -e qucsMODULEdefs.xml
  [info] admsXml-2.2.4 Oct 18 2006 19:50:46
  [info] diode.defs.h: file created
  [info] elapsed time: 0.0335337
  [info] admst iterations: 4294 (4294 freed)
  ```

  The emitted file **diode.defs.h** looks like

  ```
  /* diode verilog device */
  { "diode", 2, PROP_COMPONENT, PROP_NO_SUBSTRATE, PROP_NONLINEAR,
  ```

```
    {
      { "Is", PROP_REAL, { 1E-15, PROP_NO_STR }, { '[', 0, 1, ']' } },
      { "Cp", PROP_REAL, { 1E-12, PROP_NO_STR }, { '[', 0, 1, ']' } },
      { "Rs", PROP_REAL, { 1.0, PROP_NO_STR }, { ']', 0, 0, '.' } },
      { "Temp", PROP_REAL, { 300, PROP_NO_STR }, { ']', 0, 0, '.' } },
      PROP_NO_PROP },
    { PROP_NO_PROP }
  },
```

representing the parameters defined in the original Verilog-AMS file. This excerpt must be included in the **src/qucsdefs.h** file into this structure:

```
// List of available components.
struct define_t qucs_definition_available[] =
{
  ...
  /* end of list */
  { NULL, 0, 0, 0, 0, { PROP_NO_PROP }, { PROP_NO_PROP } }
};
```

Finally the **Makefile.am** in the **src/components/verilog/** directory must be adjusted to make the build-system aware of the new component. There must be added

```
libverilog_a_SOURCES = ... \
        diode.analogfuncction.cpp diode.core.cpp

noinst_HEADERS = ... \
        diode.analogfuncction.h diode.core.h

VERILOG_FILES = ... diode.va

if MAINTAINER_MODE
...
diode.analogfunction.cpp: analogfunction.xml
diode.analogfunction.cpp: diode.va
        $(ADMSXML) $< -e analogfunction.xml
diode.core.cpp: qucsVersion.xml qucsMODULEcore.xml
diode.core.cpp: diode.va
        $(ADMSXML) $< -e qucsVersion.xml -e qucsMODULEcore.xml
diode.defs.h: qucsVersion.xml qucsMODULEdefs.xml
diode.defs.h: diode.va
        $(ADMSXML) $< -e qucsVersion.xml -e qucsMODULEdefs.xml
diode.gui.cpp: qucsVersion.xml qucsMODULEgui.xml
diode.gui.cpp: diode.va
```

```
        $(ADMSXML) $< -e qucsVersion.xml -e qucsMODULEgui.xml
  ...
  else
```

in order to create the correct build rules.

If everything worked fine then the new Verilog-AMS model is now completely
integrated into the analogue simulator.

1.4.3 Integrating the model into the GUI

Very likely as the qucs-core tree in the previous section also the qucs source tree
must be configured using the –enable-maintainer-mode option.

```
  $ ./configure --enable-maintainer-mode --prefix=/tmp
```

Still in the **qucs-core/src/components/verilog** directory it is now necessary to
create the C++ code for the GUI using the following command line.

```
  $ admsXml diode.va -e qucsVersion.xml -e qucsMODULEgui.xml
  [info] admsXml-2.2.4 Oct 18 2006 19:50:46
  [info] diode.gui.cpp and diode.gui.h: files created
  [info] elapsed time: 0.0345217
  [info] admst iterations: 4146 (4146 freed)
```

Both the created files **diode.gui.cpp** and **diode.gui.h** should be copied to the
qucs/qucs/components directory.

Depending on the type of device several changes must be applied to these files.
The constructor will basically contain the properties of the device as they are going
to occur in the component property dialog as depicted in fig. 1.2. Check these if
they are complete or if some can be left.

```
diode :: diode ()
{
  Description = QObject :: tr ("diode verilog device");

  Props.append (new Property ("Is", "1E−15", false,
    QObject :: tr ("saturation current")));
  Props.append (new Property ("Cp", "1E−12", false,
    QObject :: tr ("parallel capacitance")));
  Props.append (new Property ("Rs", "1.0", false,
    QObject :: tr ("series resistance")));
  Props.append (new Property ("Temp", "300", false,
```

```
        QObject::tr ("temperature")));
   ...
}
```

The **diode::info** function should be adapted to meet the devices requirements.
The **BitmapFile** should be changed to represent a correct PNG file in the **qucs/bitmaps** directory. Both the **Name** and the bitmap are going to appear in the
left-hand component tab as shown in fig. 1.3.

```
Element * diode::info(QString& Name, char * &BitmapFile,
                      bool getNewOne)
{
  Name = QObject::tr("Diode");
  BitmapFile = "diode";

  if(getNewOne) return new diode();
  return 0;
}
```

The **diode::info_pnp()** method can be completely deleted. It can be necessary
for bipolar transistors where two types of devices could be displayed (npn and pnp
type). The method must also be deleted from the header file. Also, in this case,
the new diode class inherits **Component** instead of **MultiViewComponent**.

The **diode::createSymbol()** method must be adapted to represent a valid schematic
symbol. Anyway, the default code can be used as a template.

```
void diode::createSymbol()
{
  // normal bipolar
  Lines.append(new Line(-10,-15,-10, 15,QPen(QPen::darkBlue,3)));
  Lines.append(new Line(-30,  0,-10,  0,QPen(QPen::darkBlue,2)));
  Lines.append(new Line(-10, -5,  0,-15,QPen(QPen::darkBlue,2)));
  Lines.append(new Line(  0,-15,  0,-30,QPen(QPen::darkBlue,2)));
  Lines.append(new Line(-10,  5,  0, 15,QPen(QPen::darkBlue,2)));
  Lines.append(new Line(  0, 15,  0, 30,QPen(QPen::darkBlue,2)));

  // substrate node
  Lines.append(new Line(  9,  0, 30,  0,QPen(QPen::darkBlue,2)));
  Lines.append(new Line(  9, -7,  9,  7,QPen(QPen::darkBlue,3)));

  // thermal node
  Lines.append(new Line(-30, 20,-20, 20,QPen(QPen::darkBlue,2)));
  Lines.append(new Line(-20, 17,-20, 23,QPen(QPen::darkBlue,2)));
```

22

```
  // arrow
  if(Props.getFirst()−>Value == "npn") {
    Lines.append(new Line( −6, 15,   0, 15,QPen(QPen::darkBlue,2)));
    Lines.append(new Line(  0,  9,   0, 15,QPen(QPen::darkBlue,2)));
  } else {
    Lines.append(new Line( −5, 10, −5, 16,QPen(QPen::darkBlue,2)));
    Lines.append(new Line( −5, 10,  1, 10,QPen(QPen::darkBlue,2)));
  }

  // terminal definitions
  Ports.append(new Port(  0,−30)); // collector
  Ports.append(new Port(−30,  0)); // base
  Ports.append(new Port(  0, 30)); // emitter
  Ports.append(new Port( 30,  0)); // substrate
  Ports.append(new Port(−30, 20)); // thermal node

  // relative boundings
  x1 = −30; y1 = −30;
  x2 =  30; y2 =  30;
}
```

In order to test the component in the GUI it is necessary to modify some more source code files.

- `qucs/qucs.cpp`
  In this file the new Verilog device will be made available to the component tab. If a new non-linear device is added then place it here:

  ```
  pInfoFunc nonlinearComps[] =
    { ... &diode::info, 0};
  ```

- `qucs/components/component.cpp`
  The new verilog device must be loadable in the GUI. So in the function **getComponentFromName()** add this:

  ```
  Component* getComponentFromName(QString& Line)
  {
    ...
    case 'd' : if(cstr == "iode") c = new diode();
          break;
    ...
    return c;
  }
  ```

- `qucs/components/components.h`
  In the component header file it is necessary to add:

```
        #include "diode.h"
```

In order to make the build-sytem aware of the new Verilog model the file **qucs/-components/Makefile.am** must be modified in the following way.

```
libcomponents_a_SOURCES = ... \
  diode.gui.cpp

noinst_HEADERS = ... \
  diode.gui.h
```

With these steps the component is now fully integrated into the GUI.

1.5 Implemented devices

The following sections presents the models already added in Qucs including some test schematics and simulation results.

1.5.1 HICUM/L2 v2.11 model

The name HICUM was derived from HIgh-CUrrent Model, indicating that HICUM initially was developed with special emphasis on modelling the operating region at high current densities which is very important for certain high-speed applications.

The Verilog-AMS source code can be obtained from `http://www.iee.et.tu-dresden.de/iee/eb/hic_new/hic_source.html`. The model used in Qucs was a bit modified due to some ADMS limitations.

Some schematics have been setup to verify that the model emits basically the correct output values and the source code translations worked properly.

DC simulation

The setup for the forward Gummel plot is depicted in fig. 1.4. The base-emitter voltage is swept ($V_{BE} = 0.2 \ldots 1.2$) with a constant voltage across the second diode $V_{BC} = 0$.
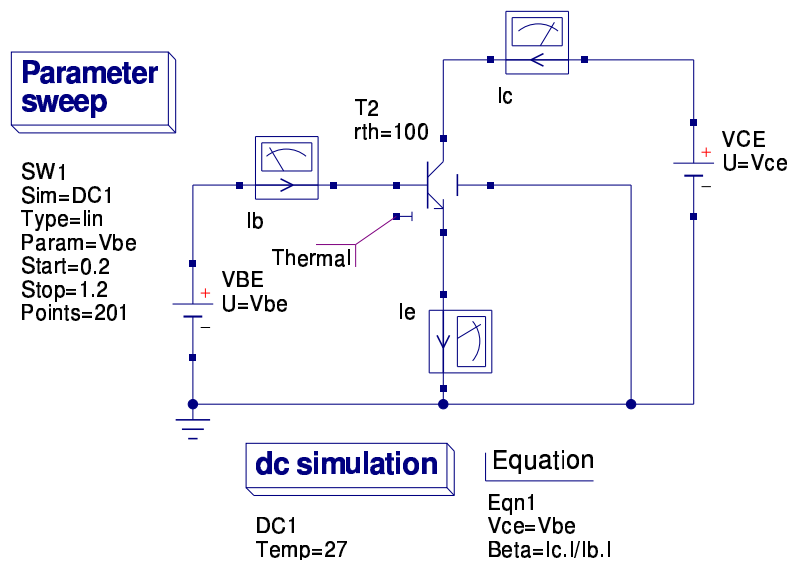
Figure 1.4: forward Gummel plot schematic for HICUM/L2 v2.11 model

In fig. 1.5 the logarithmic Gummel plot is shown including the ratio between the base current and collector current on the secondary axis.
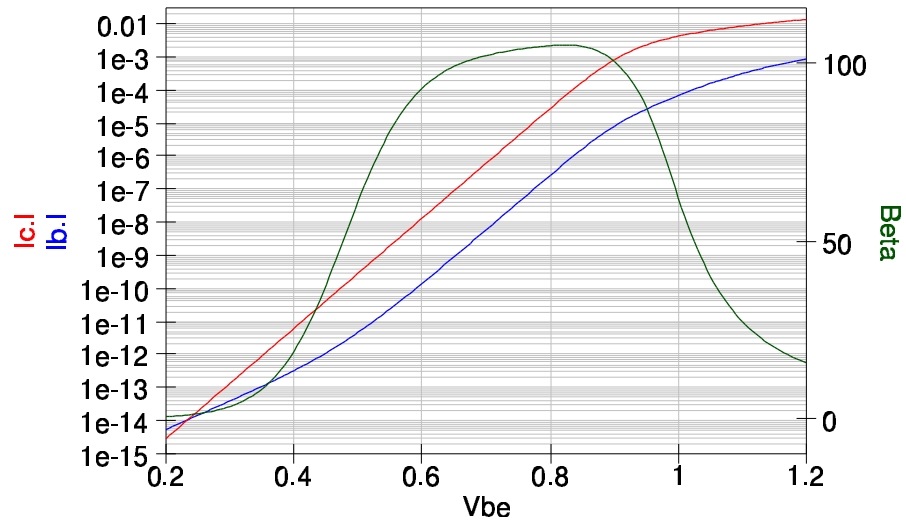
Figure 1.5: forward Gummel plot for HICUM/L2 v2.11 model

DC simulation

In fig. 1.6 the schematic for the output characteristics of the HICUM model is shown. The 2-dimensional sweep describes the function

$$I_C = f\left(V_{CE}, V_{BE}\right) \tag{1.4}$$

dc simulation

DC1

Parameter sweep

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Vce
Start=0
Stop=3
Points=101

SW2
Sim=SW1
Type=lin
Param=Vbe
Start=0.8
Stop=0.9
Points=6

Ic

VCE
U=Vce

Ib

VBE
U=Vbe

T1
rth=100
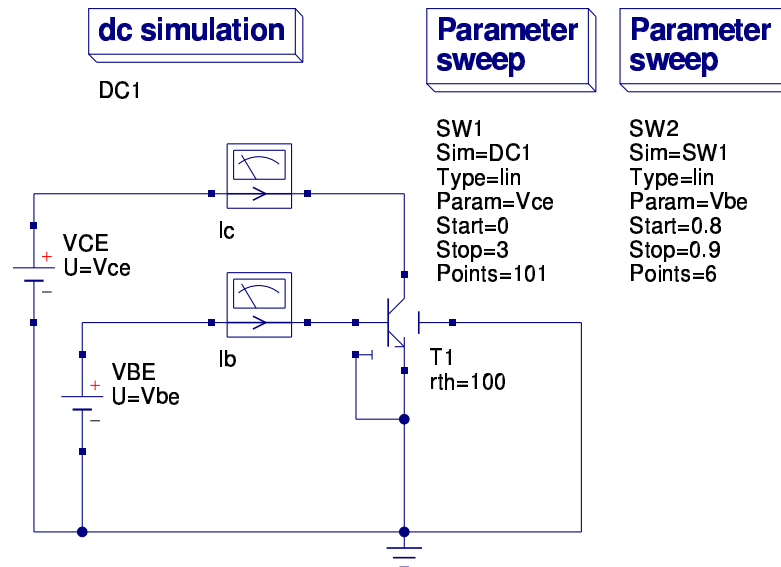
Figure 1.6: output characteristics schematic for HICUM/L2 v2.11 model

Figure 1.7 shows the results of the DC simulations for the output characteristics of the bipolar transistor.
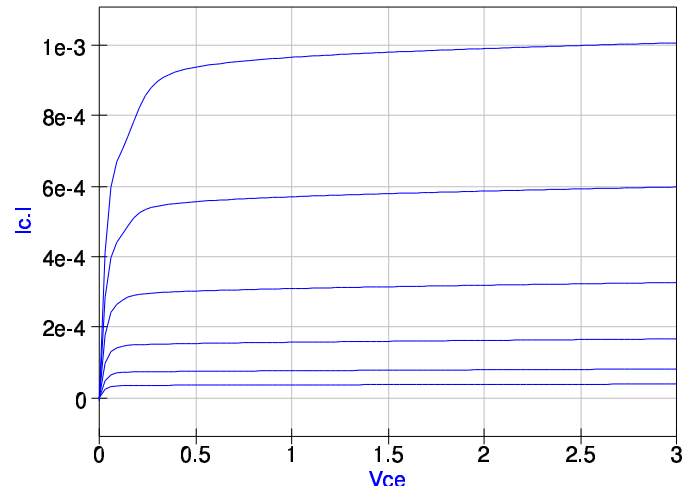
Figure 1.7: output characteristics plot for HICUM/L2 v2.11 model

AC simulation

Figures 1.8 and 1.9 depict the schematic and diagrams for an AC simulation in a given bias point. The current gains magnitude and phase are shown as well as the characteristics of the small signal base and collector current in the complex plane (polar plot).
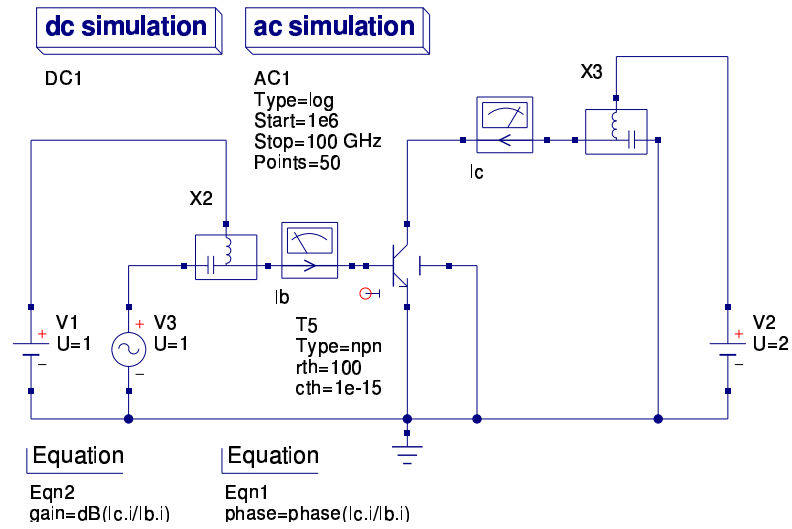
Figure 1.8: AC simulation schematic for HICUM/L2 v2.11 model
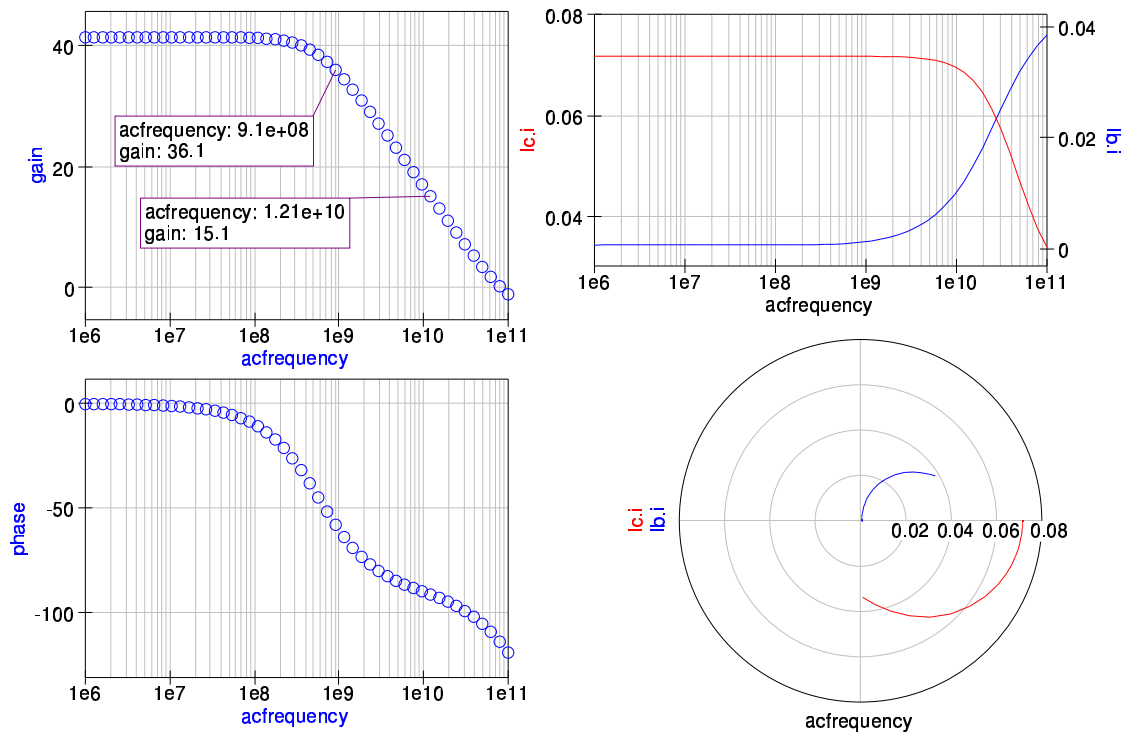
Figure 1.9: AC simulation plot for HICUM/L2 v2.11 model

S-parameter simulation

In the figures 1.10 and 1.11 a two-port S-parameter simulation schematic (including noise) as well as the results are shown. The four S-parameters are displayed in two Polar-Smith combi diagrams. The noise figure as well as the minimal noise figure are displayed in a logarithmic Cartesian diagram.
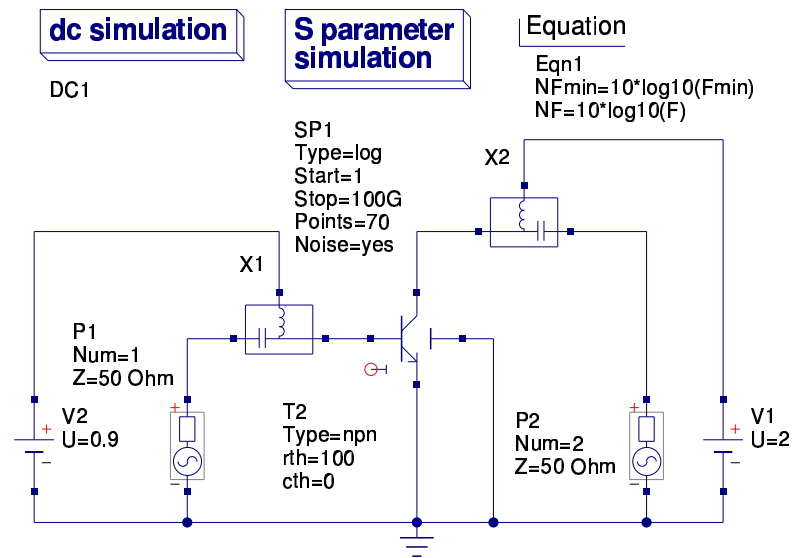
Figure 1.10: S-parameter simulation schematic for HICUM/L2 v2.11 model
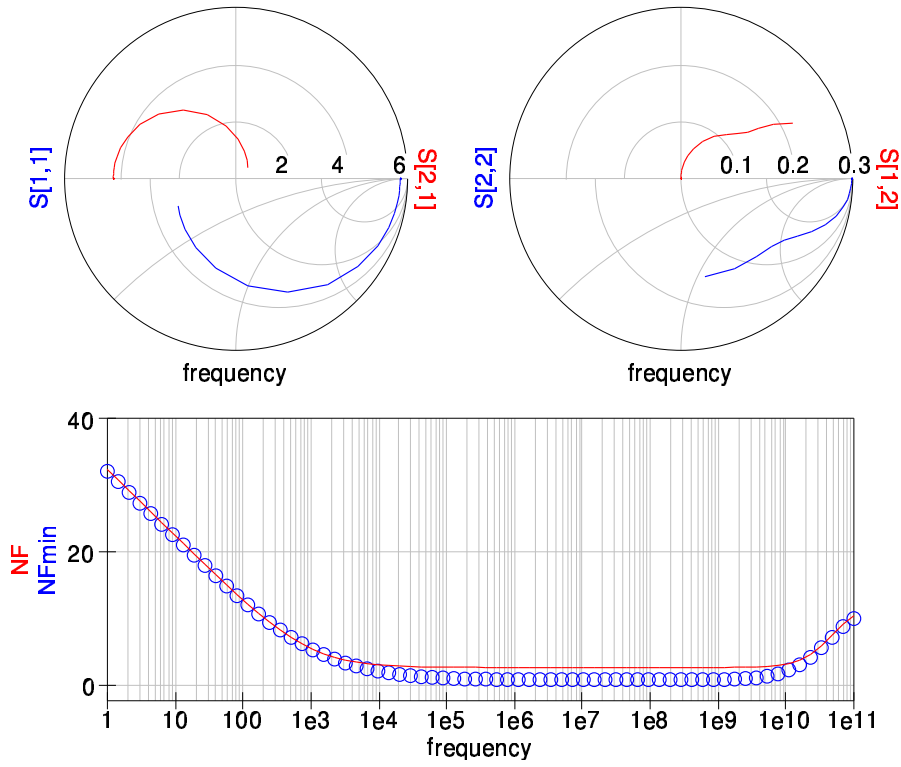
Figure 1.11: S-parameter simulation plot for HICUM/L2 v2.11 model

Transient simulation

In the schematic in fig. 1.12 a current pulse is fed into the base of the transistor. Varying the input capacitance changes the response in the collector current as shown in fig. 1.13.

base

T1
cjei0=CJ
rth=1000

VCE
U=3

I1
I1=0
I2=1m
T1=1n
T2=5n
Tr=1 n
Tf=1 n

Ib

transient simulation

Parameter sweep

Equation

Eqn1
Ic=-VCE.It

TR1
Type=lin
Start=0
Stop=15n
Points=501

SW1
Sim=TR1
Type=lin
Param=CJ
Start=1e-15
Stop=200e-15
Points=5
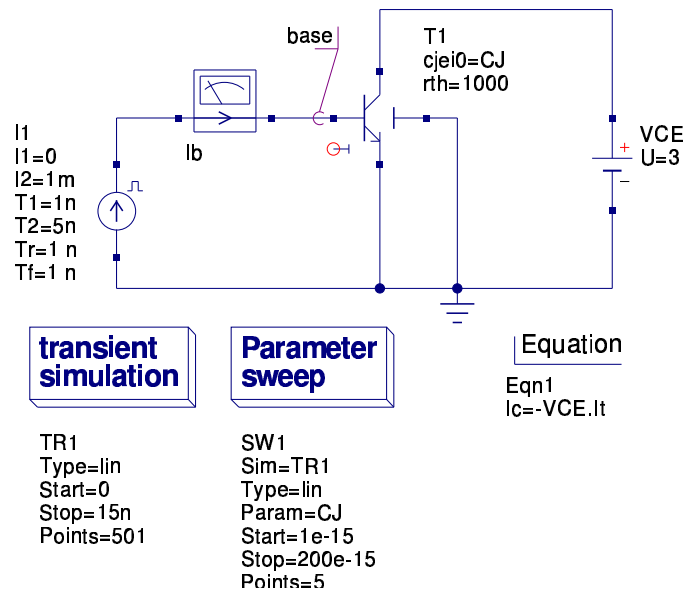
Figure 1.12: Transient simulation schematic for HICUM/L2 v2.11 model
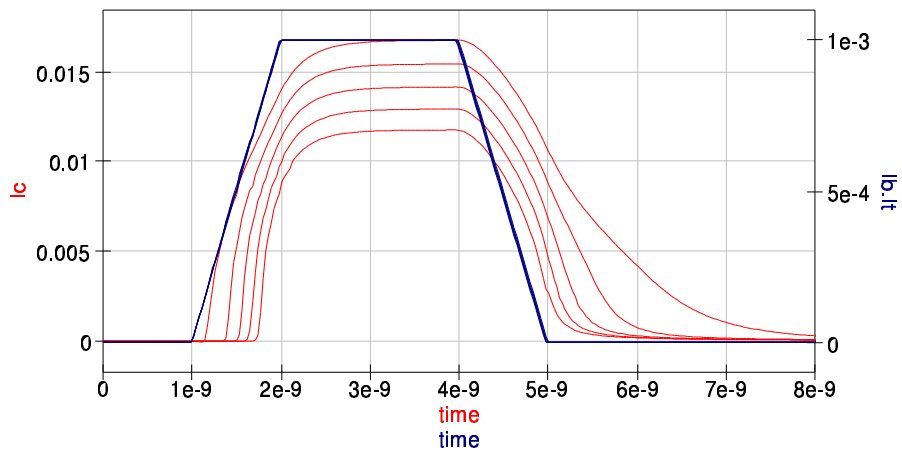
Figure 1.13: Transient simulation plot for HICUM/L2 v2.11 model

1.5.2 FBH-HBT model version 2.1

The HBT (Hetero Bipolar Transistor) model developed by Matthias Rudolph at the FBH (Ferdinand-Braun-Institut für Hochfrequenztechnik) is available in the Internet at http://www.designers-guide.org/VerilogAMS.

DC simulation

Forward Gummel plot.
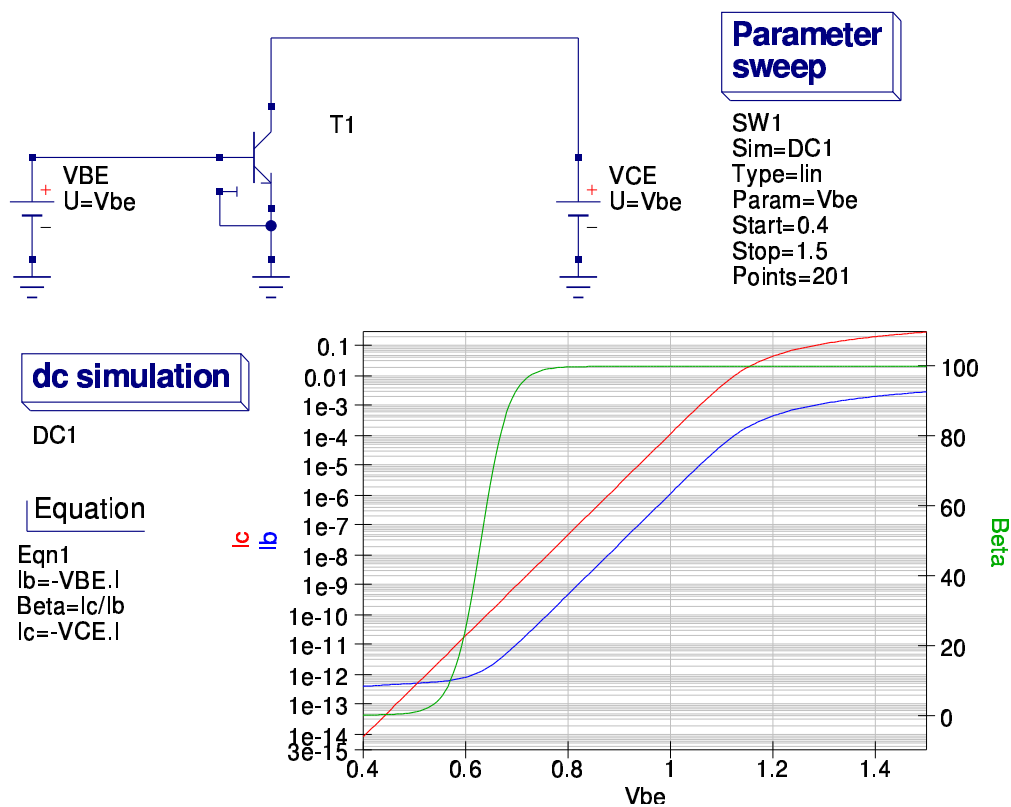
Figure 1.14: forward Gummel plot schematic for HBT model

DC simulation

Output characteristics.
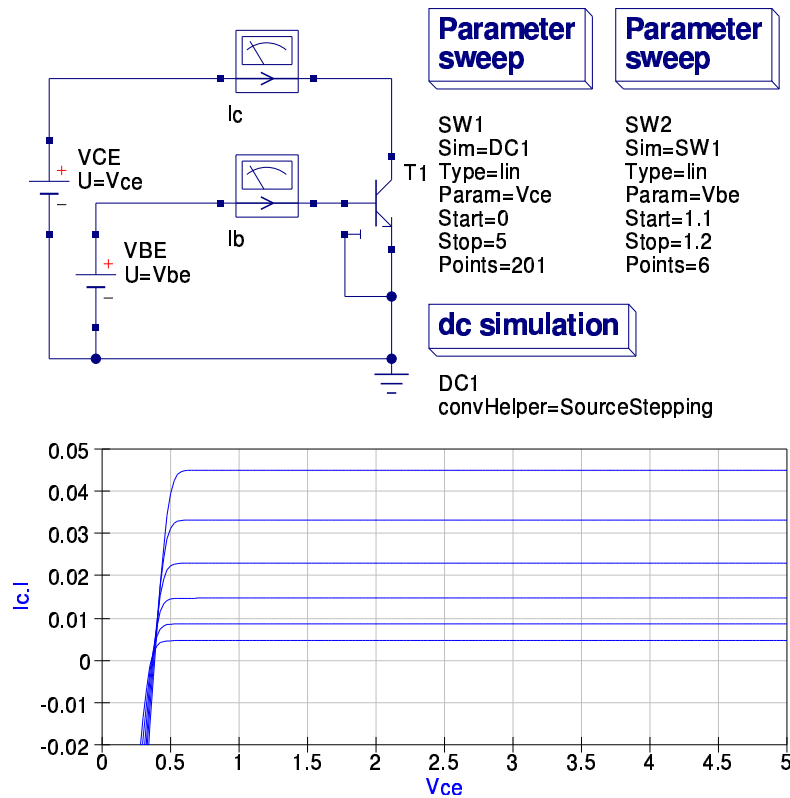
Figure 1.15: output characteristics schematic for HBT model

1.6 End Note

The preferred way to add a new Verilog model to Qucs is certainly to prepare a model, check if it is accepted by ADMS and finally ask the maintainers to integrate it. Until there is no possibility to load device modules dynamically (which is on the TODO list) too many hand-made changes have to be done to automate this process. The dynamic modules require changes in the simulator API as well in the GUI code.

Furthermore work is going to be continued on ADMS itself as well as on the **admst** scripts.

The authors would like to thank Hélène Parruitte for the initial implementation of the Verilog-AMS interface in Qucs. Also thanks go to the company Xmod Technologies (see `http://www.xmodtech.com`) allowing her to work on such an interface and finally to share the outcome of her internship under the GPL.

2 Verilog-A Modular Macromodel for Operational Amplifiers

2.1 Introduction

Since the release of Qucs version 0.0.11 the Qucs team has spent a lot of time and effort improving and debugging the Qucs device and circuit modelling facilities. Initially, the primary device and circuit modelling technique available to users was based on a subcircuit approach that allowed new models to be formed from the built in components distributed with each release of the package[1]. Releases 0.0.11 and 0.0.12 added model construction centred around the nonlinear equation defined device (EDD) and ADMS compact device models. For these modelling techniques preliminary documentation and application information has been posted on the Qucs Web site[2], outlining the fundamental basis of these procedures. Unfortunately, both the EDD and ADMS modelling techniques do require a level of specialised knowledge which many Qucs users may be unfamiliar with. These notes have been written in the hope that the information they provide will help add to the body of information published on Qucs modelling and also be useful to other Qucs users who would like to try constructing, and experimenting with, new device and circuit models using the ADMS software. Today Qucs has moved on from simply a GPL circuit simulator with user friendly schematic capture, and has started to evolve into an advanced circuit and device modelling tool which offers features that were not commonly available in previous generations of GPL circuit simulators. This report introduces an ADMS synthesised compact macromodel for the modular operational amplifier previously described

---

[1]It is also possible to construct models directly in C++ and compile and link them to the main body of the Qucs code. This is the approach taken by the package developers. However, for the average Qucs user who is primarily interested in using the package, rather than taking part in it's development, this route to model development is not really an option.

[2]Mike Brinson, "Qucs Tutorial: Component, compact device and circuit modelling using symbolic equations", `http://qucs.sourceforge.net/docs.html`, and Stefan Jahn, Hélène Parruitte, "Qucs Description: Verilog-AMS interface", `http://qucs.sourceforge.net/docs.html`.

in the Qucs tutorial on operational amplifiers.[3]. This macromodel is characterised by a similar symbol to the primitive operational amplifier gain block included with all Qucs releases. However, it provides Qucs with a more realistic general purpose operational amplifier model that can be used with confidence when designing many practical circuits which are dependent on amplifier characteristics for correct operation. The process of compiling and linking ADMS generated C++ models with the Qucs C++ code has been a learning vehicle on my part, allowing some of the complexities of the ADMS Verilog-A compiler to be decoded and understood. The ADMS synthesised operational amplifier macromodel also marks a departure from the previously described Qucs compact device models in that it represents a circuit function rather than a semiconductor device characteristic. The source code for the modular macromodel is included in this report and the ADMS generated C++ code can be found in the Qucs release source tarballs archive[4]. The procedure used to synthesize and link the model to Qucs followed the route suggested by Stefan and Hélène in their report on the Verilog-AMS/Qucs interface.

2.2 A Modular Macromodel for an Operational Amplifier

An introduction to the technical specification and structure of a modular macromodel for a general purpose operational amplifer is given on page 7 of the Qucs operational amplifier tutorial. The characteristics of this operational amplifier macromodel are:

- Input stage: Offset voltage and current, bias current, differential resistance and capacitance.

- Gain stages: Two pole differential response and single zero common-mode response.

- Large signal response: Slew rate limiting.

- Output stage: Resistance, output voltage and current limiting.

2.3 Parameters

---

[3]Mike Brinson, "Qucs Tutorial: Modelling Operational Amplifiers", `http://qucs.sourceforge.net/docs.html`.

[4]Available from `http://qucs.sourceforge.net/download.html`

Name	Symbol	Description	Unit	Default*
GBP	GBP	Gain bandwidth product	Hz	1e6
AOLDC	$AOLDC$	Open-loop differential gain at DC	dB	106.0
FP2	$FP2$	Second pole frequency	Hz	3e6
RO	RO	Output resistance	Ω	75
CD	CD	Differential input capacitance	F	1e-12
RD	RD	Differential input resistance	Ω	2e6
IOFF	$IOFF$	Input offset current	A	20e-9
IB	IB	Input bias current	A	80e-9
VOFF	VOFF	Input offset voltage	V	7e-4
CMRRDC	$CMRRDC$	Common-mode rejection ratio at DC	dB	90.0
FCM	FCM	Common-mode zero corner frequency	Hz	200.0
PSRT	$PSRT$	Positive slew rate	V/s	5e5
NSRT	$NSRT$	Negative slew rate	V/s	5e5
VLIMP	$VLIMP$	Positive output voltage limit	V	14
VLIMN	$VLIMPN$	Negative output voltage limit	V	-14
ILMAX	$ILMAX$	Maximum output current at DC	A	35e-3
CSCALE	$CSCALE$	Current limit scale factor		50

* The default parameters are for a typical UA741 Operational Amplifier.

2.4 Verilog-A model code

```
//    Qucs modular OP AMP model:
//    Default parameters are for a typical UA741.
//
// This is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation; either version 2, or (at your option)
// any later version.
//
// Copyright (C), Mike Brinson, mbrin72043@yahoo.co.uk, November 2007.
//
`include "disciplines.vams"
`include "constants.vams"
//
 module mod_amp (in_p, in_n, out_p);
 inout in_p, in_n, out_p;
 electrical in_p, in_n, out_p;
 electrical n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12;
//
 `define attr(txt) (*txt*)
//
 parameter real GBP =1e6 from [1 : inf]
  `attr(info="Gain_bandwidth_product_(Hz)");
 parameter real AOLDC = 106.0 from [0.01 : inf]
  `attr(info="Open_loop_DC_gain_(dB)");
 parameter real FP2 = 3e6 from [0.01 : inf]
  `attr(info="Second_pole_frequency_(Hz)");
```

```verilog
  parameter real RO = 75 from [0.01 : inf]
    `attr(info="Output_resistance_(Ohm)");
  parameter real CD = 1e-12 from [1e-20 : inf]
    `attr(info="Differential_input_capacitance_(F)");
  parameter real RD = 2e6 from [0.01 : inf]
    `attr(info="Differential_input_resistance_(Ohm)");
  parameter real IOFF = 20e-9 from [1e-20 : inf]
    `attr(info="Input_offset_current_(A)");
  parameter real IB = 80e-9 from [1e-20 : inf]
    `attr(info="Input_bias_current_(A)");
  parameter real VOFF = 7e-4 from [0 : inf]
    `attr(info="Input_voltage_offset_(A)");
  parameter real CMRRDC = 90.0 from [1 : inf]
    `attr(info="Common_mode_rejection_ratio_at_DC_(dB)");
  parameter real FCM = 200 from [0.01 : inf]
    `attr(info="Common_mode_zero_corner_frequency_(Hz)");
  parameter real PSRT = 5e5 from [1 : inf]
    `attr(info="Positive_slew_rate_(V/s)");
  parameter real NSRT = 5e5 from [1 : inf]
    `attr(info="Negative_slew_rate_(V/s)");
  parameter real VLIMP = 14 from [0.01 : inf]
    `attr(info="Positve_output_voltage_limit_(V)");
  parameter real VLIMN = -14 from [-inf : 0]
    `attr(info="Negative_output_voltage_limit_(V)");
  parameter real ILMAX = 35e-3 from [1e-9 : inf]
    `attr(info="Maximum_DC_output_current_(A)");
  parameter real CSCALE = 50 from [0 : inf]
    `attr(info="Current_limit_scaling_factor");
//
  real RP1, CP1, RP2, CP2;
  real Rdiff, Voffset;
  real CMRR0, CMgain, CCM;
  real Slewratepositive, Slewratenegative;
  real MTWOPI;
//
  analog begin
//
// Constants
//
    MTWOPI =  6.28318530717958647693;
//
// Design equations
//
    Voffset = VOFF*5;
    Rdiff = RD/2;
    CMRR0 = pow(10, CMRRDC/20);
    CMgain = 1e6/CMRR0;
    CCM = 1.0/(MTWOPI*1e6*FCM);
    RP1 = pow(10, AOLDC/20);
    CP1 = 1/(MTWOPI*GBP);
    RP2 = 1;
    CP2 = 1/(MTWOPI*FP2);
    Slewratepositive = PSRT/(MTWOPI*GBP);
    Slewratenegative = NSRT/(MTWOPI*GBP);
//
// Input voltage offset
//
    I(in_p, n7) <+ V(in_p, n7);
    I(in_p, n7) <+ Voffset;
//
    I(in_n, n9) <+ V(in_n, n9);
    I(in_n, n9) <+ -Voffset;
```

```verilog
//
// Input bias currents
//
    I(n7) <+ IB;
    I(n9) <+ IB;
//
// Input current offset
//
    I(n7,n9) <+ IOFF/2;
//
// Input differential resistance and capacitance
//
  I(n7, n8) <+ V(n7, n8)/Rdiff;
  I(n9, n8) <+ V(n9, n8)/Rdiff;
  I(n7, n9) <+ ddt(CD*V(n7, n9));
//
// Common mode stage
//
  I(n6) <+ -CMgain*V(n8);
  I(n6) <+ V(n6);
  I(n6, n10) <+ V(n6, n10)/1e6;
  I(n6, n10) <+ ddt(CCM*V(n6, n10));
  I(n10) <+ V(n10);
//
// Differential mode and common mode signal adder stage
//
    I(n11) <+ -V(n10);
    I(n11) <+ -V(n7, n9);
    I(n11) <+  V(n11);
//
// Slew rate limiting stage
//
    if (V(n11) > Slewratepositive)
                        I(n12) <+ -Slewratepositive;
    else if (V(n11) < -Slewratenegative)
                        I(n12) <+  Slewratenegative;
    else I(n12) <+ -V(n11);

    I(n12) <+ V(n12);
//
// First pole
//
    I(n3) <+ -V(n12);
    I(n3) <+ V(n3)/RP1;
    I(n3) <+ ddt(CP1*V(n3));
//
// Second pole
//
    I(n5) <+ -V(n3);
    I(n5) <+ V(n5)/RP2;
    I(n5) <+ ddt(CP2*V(n5));
//
// Current limiter stage
//
    if (V(n2, out_p) >= ILMAX)
        begin
          I(n4) <+  -V(n5);
          I(n4) <+ CSCALE*V(n5)*(V(n2, out_p) - ILMAX);
          I(n4) <+ V(n4);
        end
    else if (V(n2, out_p) <= -ILMAX)
        begin
```

```
            I ( n4 )  <+   −V( n5 ) ;
            I ( n4 )  <+ −CSCALE∗V( n5 )∗(V( n2 ,  out_p )  + ILMAX) ;
            I ( n4 )  <+ V( n4 ) ;
         end
     else
         begin
            I ( n4 )  <+ −V( n5 ) ;
            I ( n4 )  <+   V( n4 ) ;
         end
//
//  Output  resistance
//
     I ( n4 ,  n2 )  <+ V( n4 ,  n2 )/(RO−1) ;
     I ( n2 ,  out_p )  <+ V( n2 ,  out_p ) ;
//
//
//  Voltage  limiter  stage
//
     if  (V( out_p )  > VLIMP)
         begin
            I ( out_p )  <+   −10.0∗VLIMP;
            I ( out_p )  <+ 10.0∗V( out_p ) ;
         end
     else  if  (V( out_p )  < VLIMN)
         begin
            I ( out_p )  <+  −10.0∗VLIMN;
            I ( out_p )  <+ 10.0∗V( out_p ) ;
         end
     end
     endmodule
```

The ADMS syntax is a subset of Verilog-A. Allowed language structures are
outlined in a SYNTAX-SUPPORTED file which can be downloaed from `http:`
`//mot-adms.sourceforge.net`.

2.5 Model test examples

In the following sections a series of test results are presented. These illustrate how
the modular macromodel performs in comparison to the transistor level[5] model for
the UA741 operational amplifier. Typical parameters for the UA741 operational
amplifier are listed in the introduction to this report. These values have been
extracted from UA741 device data sheets provided by manufacturers. In the test
simulations the modular UA741 parameters have been adjusted to give similar sim-
ulation results obtained from the transistor level model. A number of interesting
differences between the simulation results obtained with the modular macromodel,
plus default parameters, and the transistor level model are observed, for example
the transistor level simulation results yield a value for IOFF of approximately zero
amperes. With a real device this is unlikely to occur due to mismatches in the
input transistor properties. In the transistor level model the input transistors are
identical implying perfect matching. This is, of course, unlikely to be the case

---

[5]The UA741 transistor level model can be found in the Qucs OpAmps component library.

with a real device. Once again the results demonstrate one of the most important
rules in circuit simulation, namely that the accuracy of the results from a specific
simulation does largely depend on how well a model represents a physical device
or circuit. Further comments about the Verilog-A code and the accuracy of the
simulation results are given with each set of test results.

2.5.1 Input voltage offset

Input voltage offset is represented by the Verilog-A code listed in this section.
This models the input voltage offset by two batteries of value VOFF/2. These
are formed by current generators in parallel with one Ohm resistors. Notice that
the direction of the current generator current flow determines the polarity of the
batteries.

```
// Input  voltage  offset
//
    I ( in_p ,  n7 ) <+ V ( in_p ,  n7 );
    I ( in_p ,  n7 ) <+ Voffset ;
//
    I ( in_n ,  n9 ) <+ V ( in_n ,  n9 );
    I ( in_n ,  n9 ) <+ -Voffset ;
```
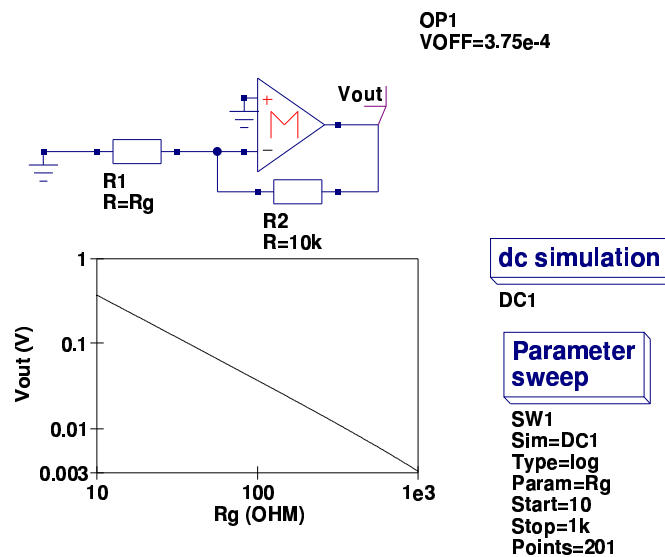

Figure 2.1: UA741 modular macromodel input voltage offset test circuit and results
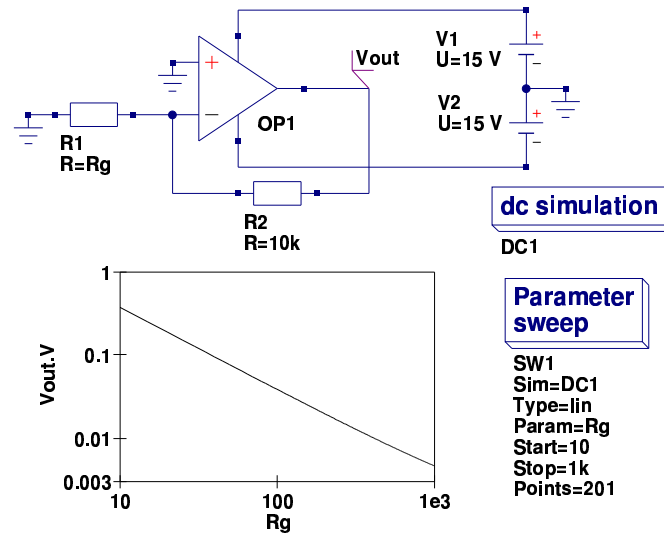
Figure 2.2: UA741 transistor level input voltage offset test circuit and simulation results

2.6 Input bias and offset currents

Input bias and offset currents are represented by the Verilog-A code listed in this section. Simple current generators are employed to model the input current effects. Values for IB and IOFF can be extracted from the results given in Figs. 2.3 and 2.4, using equations (2.1) and (2.2).

$$Vp = -\left(IB + \frac{IOFF}{2}\right) \cdot 1e3 \tag{2.1}$$

$$Vn - Vs = \left(\frac{IOFF}{2} - IB\right) \cdot 1e3 \tag{2.2}$$

The remainder of the input stage Verilog-A code adds differential input resistance and capacitance to the input stage of the operational amplifier macromodel.

```
// Input bias currents
//
    I(n7) <+ IB;
    I(n9) <+ IB;
//
// Input current offset
//
    I(n7,n9) <+ IOFF/2;
//
// Input differential resistance and capacitance
//
  I(n7, n8) <+ V(n7, n8)/Rdiff;
  I(n9, n8) <+ V(n9, n8)/Rdiff;
  I(n7, n9) <+ ddt(CD*V(n7, n9));
```

42
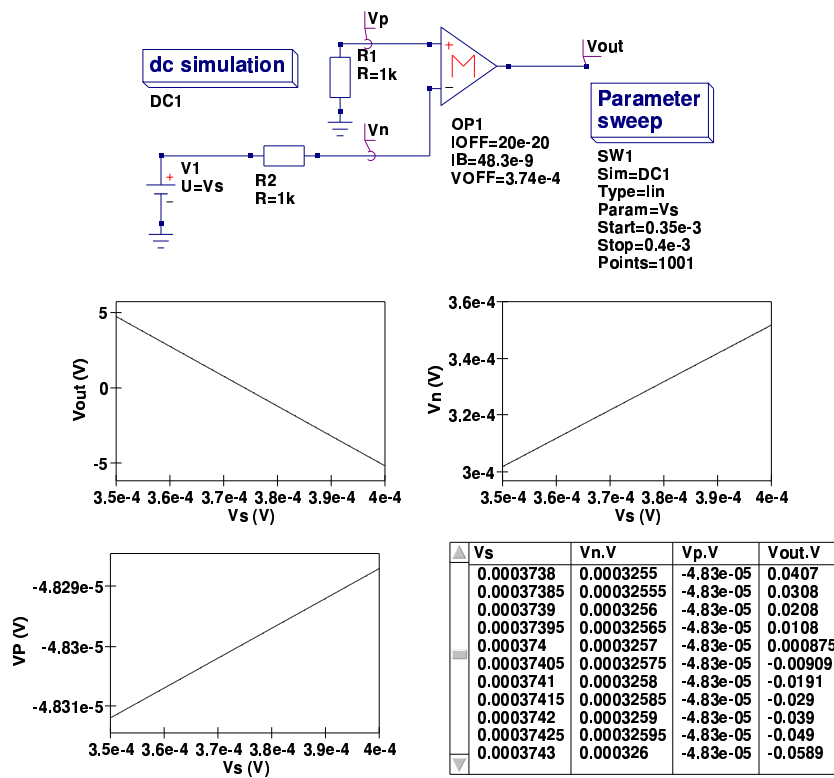
Figure 2.3: UA741 modular macromodel input bias and offset current test circuit and simulation results

Circuit labels:

dc simulation
DC1

Vp
R1
R=1k

Vn

V1
U=Vs

R2
R=1k

Vout

OP1
IOFF=20e-20
IB=48.3e-9
VOFF=3.74e-4

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Vs
Start=0.35e-3
Stop=0.4e-3
Points=1001

Vs	Vn.V	Vp.V	Vout.V
0.0003738	0.0003255	-4.83e-05	0.0407
0.00037385	0.00032555	-4.83e-05	0.0308
0.0003739	0.0003256	-4.83e-05	0.0208
0.00037395	0.00032565	-4.83e-05	0.0108
0.000374	0.0003257	-4.83e-05	0.000875
0.00037405	0.00032575	-4.83e-05	-0.00909
0.0003741	0.0003258	-4.83e-05	-0.0191
0.00037415	0.00032585	-4.83e-05	-0.029
0.0003742	0.0003259	-4.83e-05	-0.039
0.00037425	0.00032595	-4.83e-05	-0.049
0.0003743	0.000326	-4.83e-05	-0.0589

Figure 2.4: UA741 transistor level input bias and offset current test circuit and simulation results

Vs	Vn.V	Vp.V	Vout.V
0.000377665	0.000329166	-4.82e-05	0.0019
0.00037767	0.000329171	-4.82e-05	0.00138
0.000377675	0.000329176	-4.82e-05	0.000857
0.00037768	0.000329181	-4.82e-05	0.000338
0.000377685	0.000329186	-4.82e-05	-0.000181
0.00037769	0.000329191	-4.82e-05	-0.0007
0.000377695	0.000329196	-4.82e-05	-0.00122
0.0003777	0.000329201	-4.82e-05	-0.00174
0.000377705	0.000329206	-4.82e-05	-0.00226
0.00037771	0.000329211	-4.82e-05	-0.00278
0.000377715	0.000329216	-4.82e-05	-0.0033

2.7 Open loop differential voltage gain

Differential voltage gain is represented by the Verilog-A code listed in this section. Differential gain is modelled with three distinct quantities. These are 1. resistor RP1 which is set equal to the open loop differential gain at DC, 2. the primary pole in the voltage gain frequency response which is set by capacitor CP1, and 3. a high frequency pole set by CP2 and resistor RP2. Each pole stage is driven by a voltage controlled current generator. Note the current generator negative sign. This is required to maintain the correct signal phase. The derivation, and a more detailed discussion, of the model open-loop differential voltage gain properties can be found in pages 13 to 19 of the Qucs operational amplifier tutorial. Simulated results for the open-loop differential gain response are shown in Figs. 2.5 and 2.6 The AOLDC and CD parameters given in Fig. 2.5 have been adjusted to give the same response as the Qucs transistor level model. Again due to perfect transistor matching a value of CD roughly zero is required to produce similar high frequency responses above the second pole frequency.

```
//
// First pole
//
   I(n3) <+ -V(n12);
   I(n3) <+ V(n3)/RP1;
   I(n3) <+ ddt(CP1*V(n3));
//
// Second pole
//
   I(n5) <+ -V(n3);
   I(n5) <+ V(n5)/RP2;
   I(n5) <+ ddt(CP2*V(n5));
//
```

Figure 2.5: UA741 modular macromodel open-loop differential voltage gain test circuit and simulation results
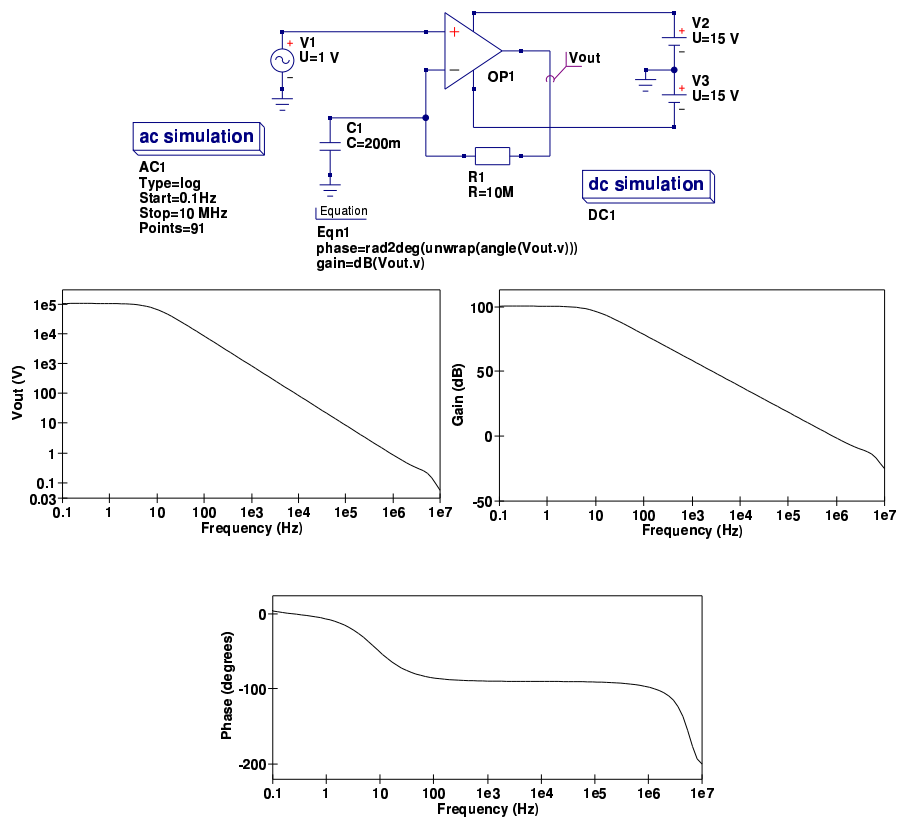
Figure 2.6: UA741 transistor level open-loop differential voltage gain test circuit and simulation results

2.8 Common mode effects

Operational amplifier common-mode effects are represented by the Verilog-A code listed in this section. Common-mode effects are simulated using an identical network to that outlined in page 20 the Qucs operational amplifier tutorial. The simulated results for a unity gain CMMR test circuit are illustrated in Figs. 2.7 and 2.8. The macromodel parameters CMRR and FCM have been set at their default values to demonstrate the difference when compared to the transistor level model. Common-mode simulation results observed with the transistor level model are significantly better than those obtained from measurements on real devices, mainly because the simulation model is constructed from perfectly matched transistors. Associated with the differential amplifier stages and the common-mode section of the macromodel is a signal adder which combines the differential and common-mode signals. The Verilog code for this adder stage is also repeated with common-mode code. An adder can be formed by a one Ohm resistor driven by differential and common-mode currents. The resulting volt drop across the one Ohm resistor then becomes the sum of the two signal voltages.

```
//
// Common mode stage
//
   I(n6) <+ -CMgain*V(n8);
   I(n6) <+ V(n6);
   I(n6, n10) <+ V(n6, n10)/1e6;
   I(n6, n10) <+ ddt(CCM*V(n6, n10));
   I(n10) <+ V(n10);
//
// Differential mode and common mode signal adder stage
//
   I(n11) <+ -V(n10);
   I(n11) <+ -V(n7, n9);
   I(n11) <+  V(n11);
//
```

OP1
GBP=1e6
AOLDC=100.0
CD=1e-20
IOFF=20e-20
IB=48.3e-9
VOFF=3.2e-4
CMRRDC=90
FCM=200

R2
R=10k

Vout

dc simulation
DC1

ac simulation
AC1
Type=log
Start=0.01Hz
Stop=10 kHz
Points=81

R1
R=10k

V1
U=1 V

R3
R=10k

R4
R=10k

Equation
Eqn1
CMRR=dB(1/Vout.v)
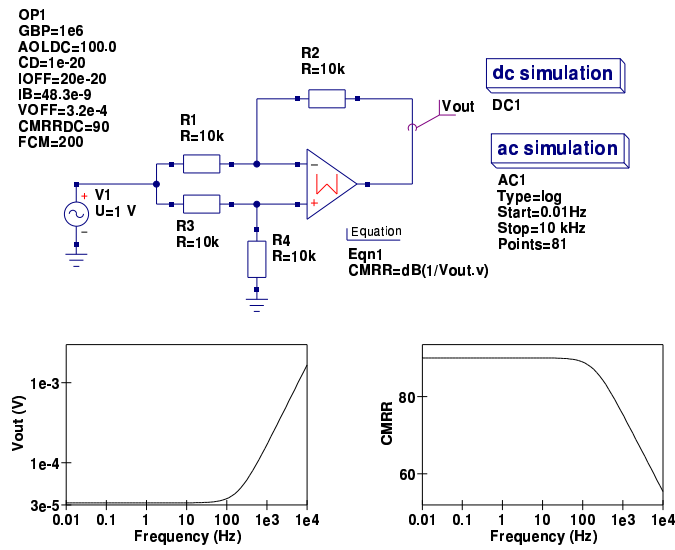
Figure 2.7: UA741 modular macromodel CMRR test circuit and simulation results

R2
R=10k

Vout

dc simulation
DC1

ac simulation
AC1
Type=log
Start=0.01Hz
Stop=10 kHz
Points=81

Equation
Eqn1
CMRR=dB(1/Vout.v)

R1
R=10k

OP1

V3
U=15 V

V1
U=1 V

V2
U=15 V

R3
R=10k

R4
R=10k
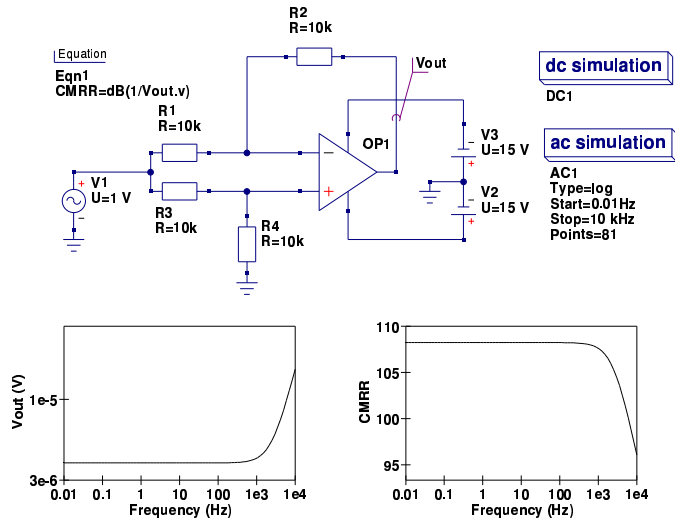
Figure 2.8: UA741 transistor level CMRR test circuit and simulation results

2.9 Slew rate limiting

Operational amplifier slew rate limiting effects are represented by the Verilog-A code listed in this section. Slew rate limiting can be modelled in Verilog-A using the if-else language statement. This allows the maximum signal current at node n12 to be limited to a value set by variables Slewratepositve or Slewratenegative which in turn are functions of parameters PSRT and NSRT (see page 25 of the Qucs operational amplifier tutorial). Again please note the sign of I(n12). The simulated results for a UA741 slewrate limiting test circuit are illustrated in Figs. 2.9 and 2.10. In general the results are very similar for both models. However, there is one point worth commenting on; in the case of the transistor level model there is a marked difference in the level of slewing for negative and positive signals (see waveform Vout3 in Fig. 2.10). This effect is probably due to the fact that the UA741 operational amplifier circuitry is very different near the power supply rails, resulting in significant differences in signal shape at high signal swings. This effect is not modelled by the modular macromodel.

```
//
// Slew  rate  limiting  stage
//
    if  (V(n11) > Slewratepositive)
                        I(n12) <+ −Slewratepositive;
    else  if  (V(n11) < −Slewratenegative)
                        I(n12) <+   Slewratenegative;
    else  I(n12) <+ −V(n11);
    I(n12) <+ V(n12);
```
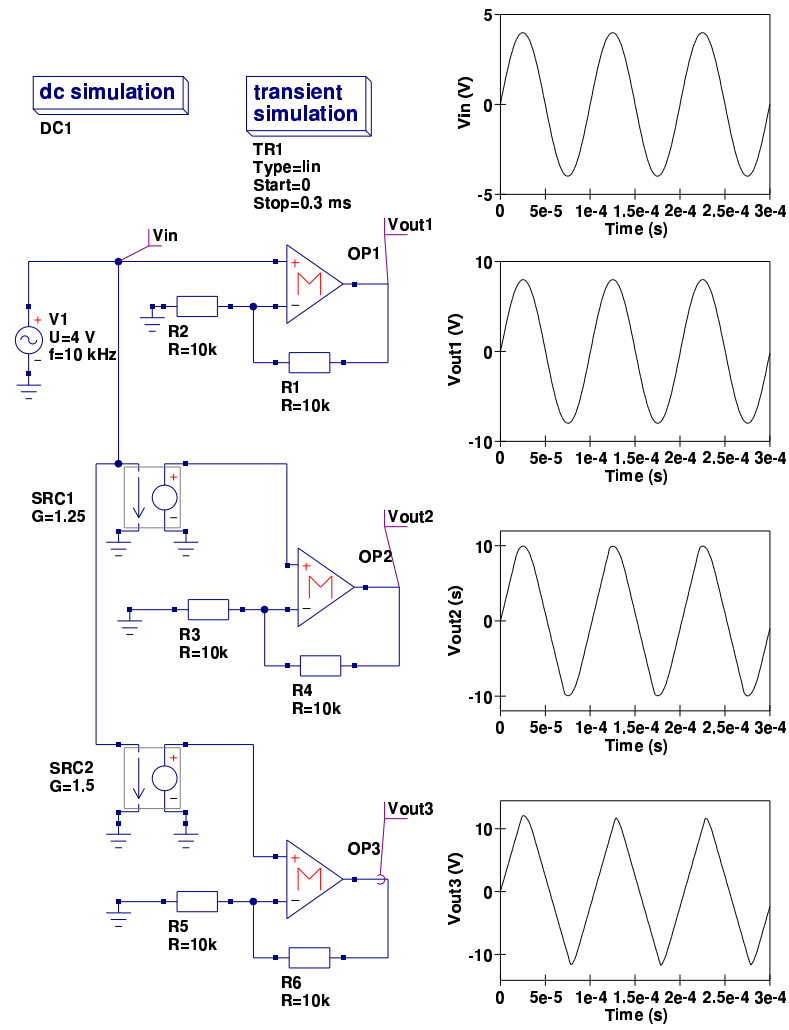
Figure 2.9: UA741 modular macromodel slewrate limiting test circuit and simulation results
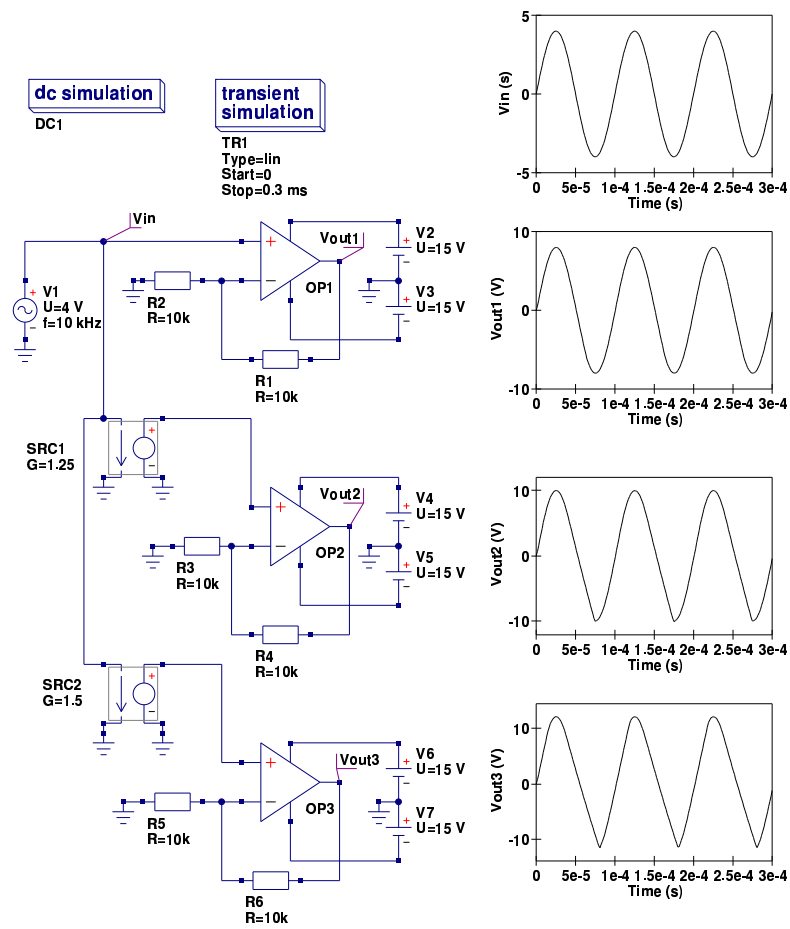
Figure 2.10: UA741 transistor level slewrate limiting test circuit and simulation results

2.10 Output voltage limiting

Operational amplifier output voltage limiting effects are represented by the Verilog-A code listed in this section. A Verilog-A if-else statement is used to model voltage limiting. This language construction also demonstrates the use of the Verilog-A block construction formed with begin-end code words. Effectively the if-else statement swaps circuit elements as the voltage signal polarity changes. The simulated results for a UA741 voltage limiting test circuit are illustrated in Figs. 2.11 and 2.12. Again the results are very similar for both models. However, in the case of the macromodel there is no attempt to reduce the differential gain when output voltage limiting occurs and as a result some waveform distortion takes place. In practice if, as in the case of pure AC simulation, output voltage limiting is not required then simply set VLIMP and VLIMN well outside the range of required operating voltage and voltage limiting is never triggered.

```
//
// Voltage limiter stage
//
   if (V(out_p) > VLIMP)
      begin
        I(out_p) <+  −10.0∗VLIMP;
        I(out_p) <+ 10.0∗V(out_p);
      end
  else if (V(out_p) < VLIMN)
      begin
        I(out_p) <+ −10.0∗VLIMN;
        I(out_p) <+ 10.0∗V(out_p);
      end
   end
```
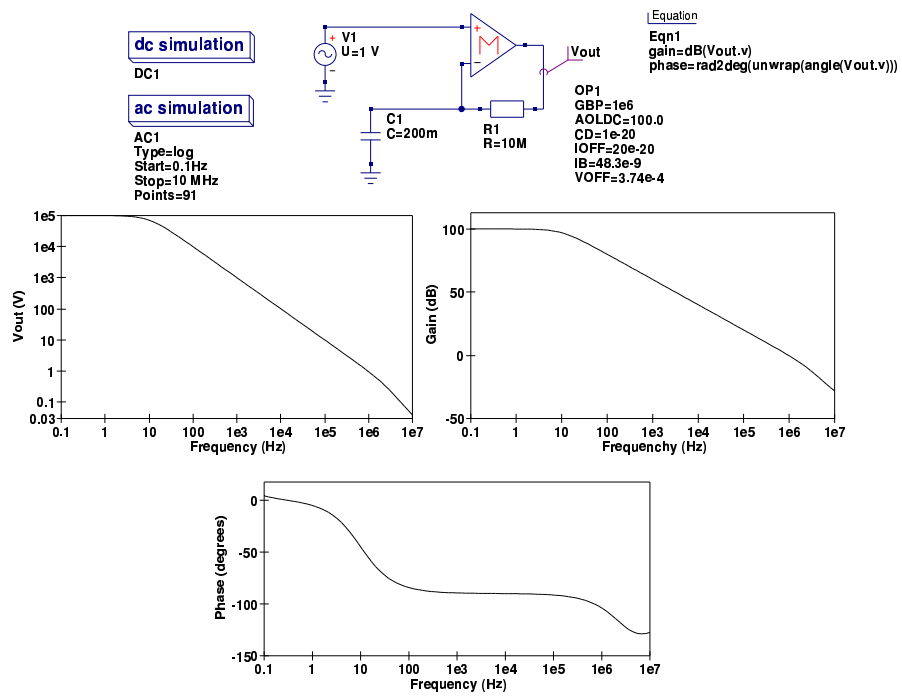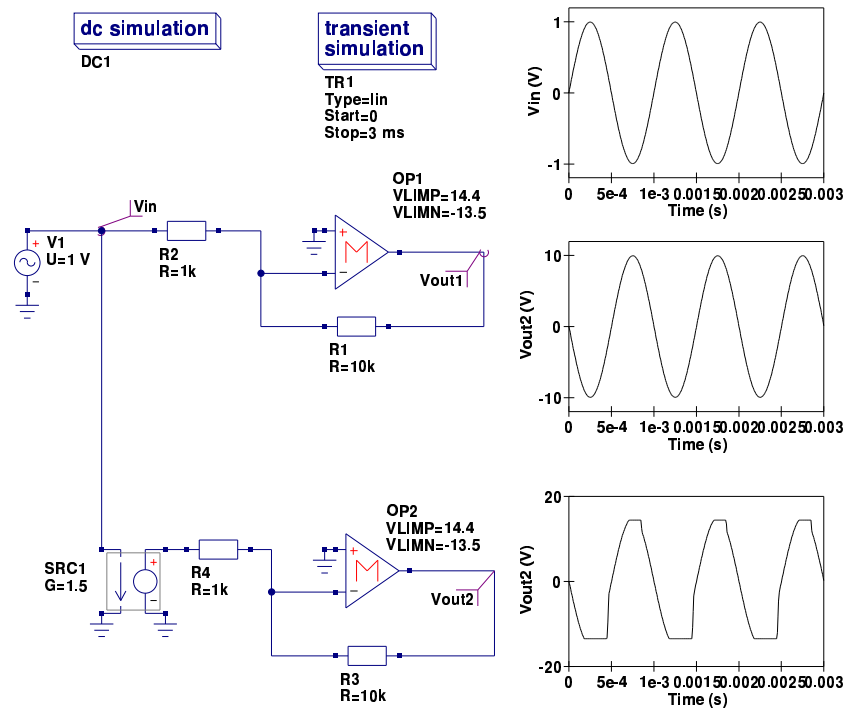
Figure 2.11: UA741 modular macromodel output voltage limiting test circuit and simulation results
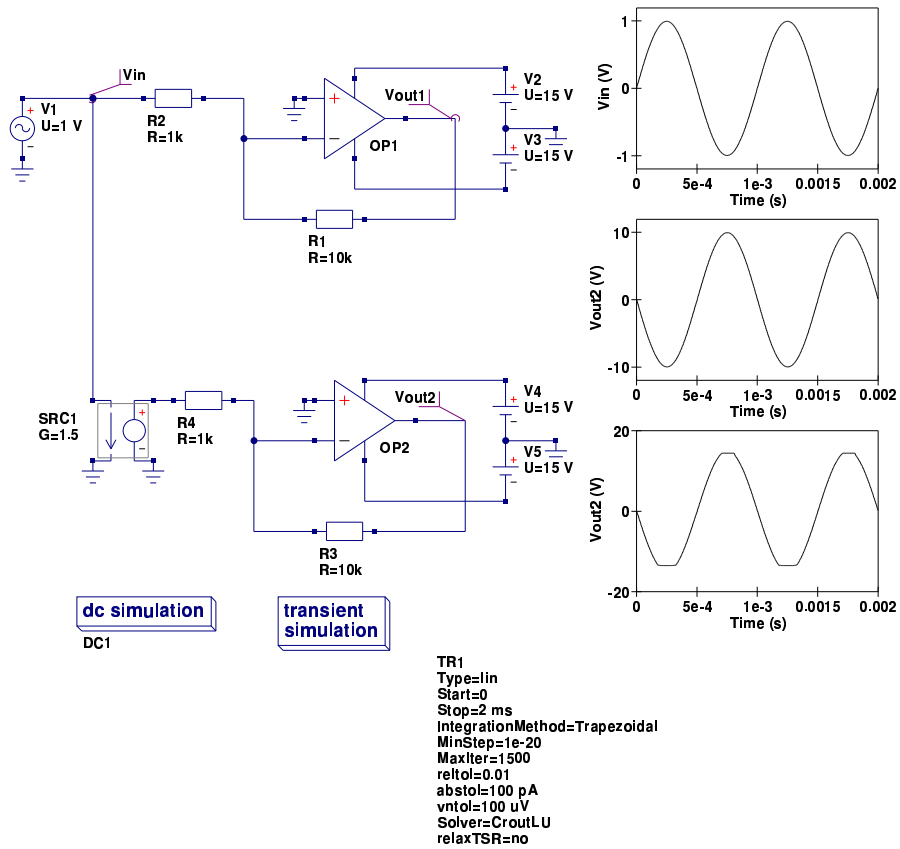
Figure 2.12: UA741 transistor level output voltage limiting test circuit and simulation results

2.11 Output current limiting

Operational amplifier output current limiting effects are represented by the Verilog-A code listed in this section. A Verilog-A if-else statement is used to model current limiting. The effect is modelled by a feedback mechanism which reduces the magnitude of current I(n4) which is proportional to the difference of the current flowing in the output path and ILMAX. A scale factor CSCALE is used to adjust maximum clamped output current. The simulated results for a UA741 current limiting test circuit are illustrated in Figs. 2.13 and 2.14. Current clamping induces distortion in the output waveform. Both the macromodel and the transistor level model give roughly the same clamped output currents but the wave shapes, and hence the distortion, are somewhat different. This is not surprising as the macromodel does not include a mechanism to control clamping distortion. Setting CSCALE to zero removes the current limiting process from the operational amplifier macromodel.

```
//
// Current limiter stage
//
    if (V(n2, out_p) >= ILMAX)
       begin
         I(n4) <+  -V(n5);
         I(n4) <+ CSCALE*V(n5)*(V(n2, out_p) - ILMAX);
         I(n4) <+ V(n4);
       end
    else if (V(n2, out_p) <= -ILMAX)
         begin
          I(n4) <+  -V(n5);
          I(n4) <+ -CSCALE*V(n5)*(V(n2, out_p) + ILMAX);
          I(n4) <+ V(n4);
         end
    else
         begin
             I(n4) <+ -V(n5);
             I(n4) <+  V(n4);
         end
//
```
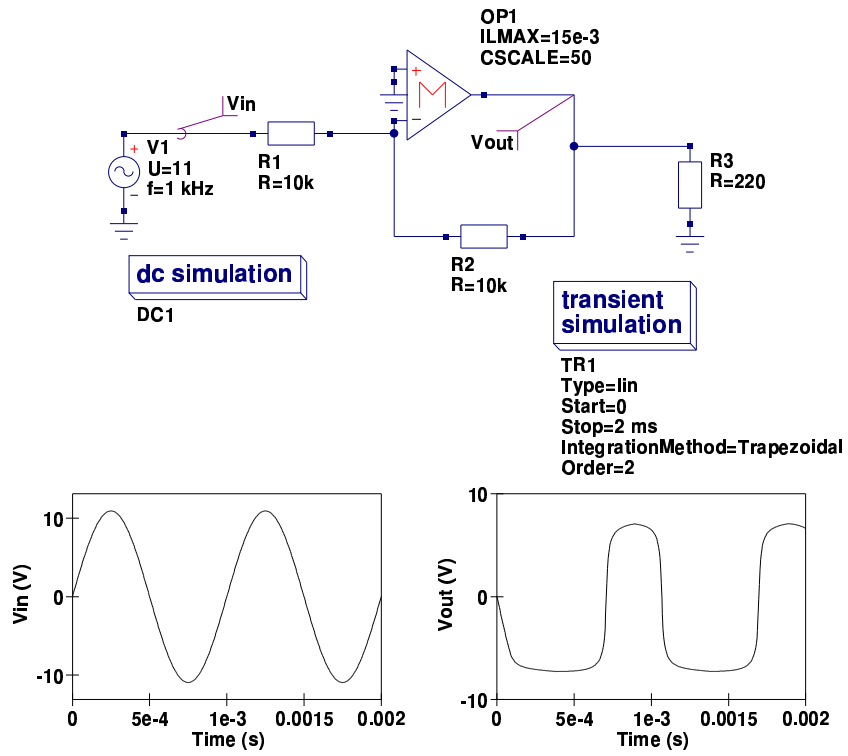
Figure 2.13: UA741 modular macromodel output current limiting test circuit and simulation results
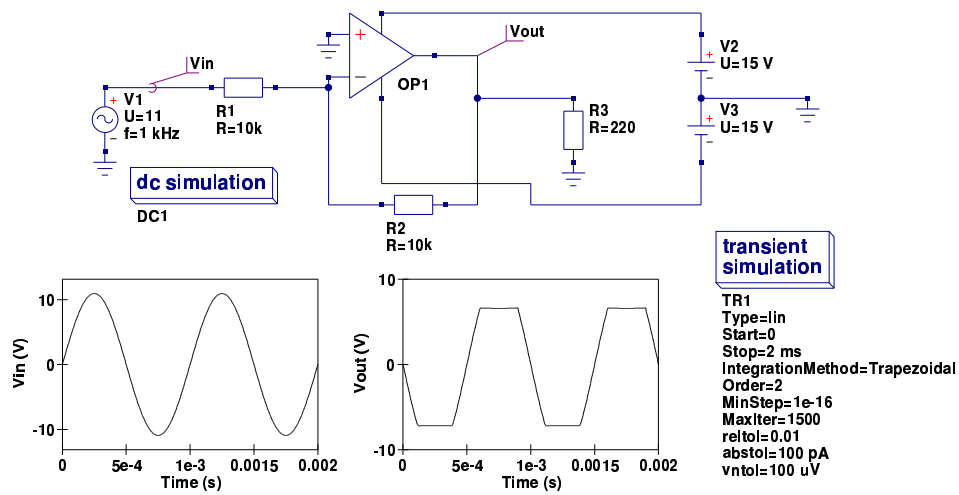
Figure 2.14: UA741 transistor level output current limiting test circuit and simulation results

2.12 End note

These notes summarise a number of techniques for modelling operational amplifier functions using Verilog-A. Verilog-A is primarily intended for modelling compact semiconductor devices. However, as this report demonstrates it can be equally employed for macromodelling of integrated circuits and circuits in general. The Qucs C++ code for the modular operational amplifier model, generated by ADMS, can be found in the Qucs release source tarballs at `http://qucs.sourceforge.net`. The procedure followed to convert the Verilog-A code into C++ for compilation and linking with Qucs closely followed that described by Sefan Jahn and Hélène Parruitte. Although it takes more work to construct models using ADMS the effort is worthwile because the finished models are very efficient in terms of memory usage and have significantly reduced run times. One interesting observation which is worth recording here is the fact that Qucs equation defined device (EDD) models and ADMS Verilog-A models have a very similar structure, implying that EDD models can be used as prototypes prior to compilation and linking via the compact device modelling route using ADMS. Once again a special thanks to Stefan Jahn for all his help and encouragement over the period that I have been developing the Verilog-A version of the operational amplifier macromodel, writing this report and testing the examples it includes.

3 Verilog-A Logarithmic Amplifier Macromodel

3.1 Introduction

The development of simulation component and device models in many ways reflects how circuit simulator functionality has improved with increasing personal computer computational power. Early circuit simulators were often restricted to the DC, AC and transient analysis domains. Similarly, in the early days of circuit simulation, the available types of component and device models were extremely limited, often being confined to the fundamental passive components, simple voltage and current sources and the basic semiconductor devices. Adding new models to a circuit simulator was, in most instances, a complex task. Today this picture is changing. Modern circuit simulators, like Qucs, are equipped with an ever widening selection of simulation tools plus increasingly sophisticated modelling tools. The later allows easy construction of subcircuit models, linear and non-linear macromodels, equation defined device models, and Verliog-A compact device models. This report describes how a compact macromodel of a logarithmic amplifier can be constructed from a model description written in the Verilog-A hardware description language, compiled to C++ code using the ADMS compiler plus Qucs XML interface, and linked to the main body of Qucs C++ code[1]. The logarithmic amplifier model demonstrates an approach to Qucs modelling which allows high-level functional models of integrated circuits to be added to the existing range of Qucs component and device models.

---

[1]A detailed description of the procedure for the compilation of Verilog-A models with ADMS and linking of the resulting C++ code to Qucs can be found in the Qucs publication : "Qucs Description: Verilog-AMS interface", by Stefan Jahn, Hélène Parruitte, located at http://qucs.sourceforge.net/docs.html.

3.2 The ideal logarithmic amplifier

The operation of an ideal logarithmic amplifier[2] is defined by the function given in equation (3.1).

$$V_{out}(ideal) = K_v \cdot log\left(\frac{I_i}{I_r}\right) \qquad (3.1)$$

The device accepts two input currents (I_i and I_r) and outputs voltage $V_{out}(ideal)$ which is proportional to the base ten logarithm of the current ratio I_i/I_r, where the constant of proportionality is K_v volts per decade. In equation (1) current I_r is called the reference current and current I_i represents the amplifier input signal. The primary purpose of a logarithmic amplifier is not to amplify input signals but to compress a wide dynamic range input signal to give as output it's logarithmic equivalent. In some respects the name logarithmic amplifier is misleading and it would be better to consider the amplifier as a measurement device rather than an amplifier. For the ideal logarithmic amplifier when $I_i = I_r$, the output voltage $V_{out}(ideal) \Rightarrow 0$. In a practical logarithmic amplifier the current input terminals are effectively at virtual ground potential which allows the amplifier input signals to be derived from input voltages divided by scaling resistors of identical value. In this case $I_r = V_r/R$ and $I_i = V_i/R$, where V_r and V_i are the input signal voltages respectively, and R is the scaling resistance. Transforming from current inputs to voltage inputs gives equation (3.2).

$$V_{out}(ideal) = K_v \cdot log\left(\frac{V_i}{V_r}\right) \qquad (3.2)$$

To prevent the argument of the logarithmic amplifier from becoming negative the input signal I_i must always have the same polarity as reference signal I_r. The amplitude of the signal at terminal I must also be the same or greater than the signal at terminal R. This implies that logarithmic amplifiers are unipolar input devices. Similarly, to prevent the logarithmic amplifier output voltage from becoming excessively large and causing the output voltage to saturate[3] the ratio of the two currents must be limited to a maximum value. In this context the dynamic range of a logarithmic amplifier is defined by equation (3.3).

$$\text{dynamic range} \simeq log\left(\frac{I_i}{I_r}\right) \simeq log\left(\frac{V_i}{V_r}\right) \qquad (3.3)$$

---

[2]A very good introduction to the theory of logarithmc amplifiers is given in Sergio Franco, Design with Operational amplifiers and Analog Integrated Circuits, McGraw-Hill Book Company, ISBN 0-07-021799-8.

[3]Power supply voltages are often limited to $\pm$ 15 volts.

In a real logarithmic amplifier I_r can be as low as 10 nA with the maximum allowed value of I_i around 10 mA, yielding a dynamic range of six decades. Integrated logarithmic amplifiers are available with dynamic ranges of five or six decades.

3.3 The practical logarithmic amplifier

The voltage output from a practical logarithmic amplifier[4] differs from that given in equation (3.1) and can be written as

$$V_{out} = V_{out}(ideal) \pm TE \tag{3.4}$$

Where TE is called the total error. This can be represented by a function formed from the combination of errors in gain scale factor, input offset current, input bias current, output offset voltage and transfer function non-linearity. On adding the major contributions due to these errors equation (3.4) becomes equation (3.5).

$$V_{out} = K_v \cdot (1 \pm \Delta K_v) \cdot log \left(\frac{I_i - I_{b1}}{I_r - I_{br}} \right) \pm 2 \cdot K_v \cdot N \cdot m \pm V_{osout} \tag{3.5}$$

In terms of voltage inputs equation (3.5) can be written as

$$V_{out} = K_v \cdot (1 \pm \Delta K_v) \cdot log \left(\frac{\frac{V_i}{R} - I_{b1}}{\frac{V_r}{R} - I_{br}} \right) \pm 2 \cdot K_v \cdot N \cdot m \pm V_{osout} \tag{3.6}$$

Where ΔK_v is the scale error factor in percent, I_{b1} is the bias current at input I in amperes, I_{br} is the bias current at the reference input R in amperes, N is the log conformity error in percent, m is the number of decades over which N is specified, and V_{osout} is the output offset voltage. The log conformity error of a logarithmic amplifier is defined as the peak deviation from the best-fit straight line of V_{out} versus the $log(I_i/I_r)$ curve expressed as a percentage of peak-to-peak full-scale. The other error parameters have their usual meaning. In general error parameters may be plus or minus in sign, leading to the $\pm$ signs in the previous equations.

3.4 Logarithmic amplifier temperature effects

The error parameters introduced in the last section of this report are normally specified in manufacturers device data sheets as functions of temperature. As a

---

[4]See the manufacturers data sheets for (a) Burr Brown (from Texas Instruments) LOG100 and LOG101 amplifiers, and (b) Maxim MAX4206 amplifier.

first approximation parameter temperature dependence is usually limited to the simple linear functions of temperature given in equation (3.7).

$$\Delta K_v \left(Temp\right) = \Delta K_v \cdot + \Delta K_v tc \cdot \left(Temp - Tnom\right)$$
$$I_{b1} \left(Temp\right) = I_{b1} + I_{b1}tc \cdot \left(Temp - Tnom\right)$$
$$I_{br} \left(Temp\right) = I_{br} + I_{br}tc \cdot \left(Temp - Tnom\right)$$
$$N \left(Temp\right) = N + Ntc \cdot \left(Temp - Tnom\right)$$
$$V_{osout} \left(Temp\right) = V_{osout} + V_{osout}tc \cdot \left(Temp - Tnom\right) \tag{3.7}$$

Where $Temp$ is the circuit simulation temperature in Celsius, $Tnom$ is the device parameter measurement temperature in Celsius and $\Delta K_v tc$, $I_{b1}tc$, $I_{br}tc$, Ntc and $V_{osout}tc$ are first order linear temperature coefficients in percentage per degree Celsius or units per degree Celsius.

3.5 A compact macromodel for a logarithmic amplifier

3.5.1 Parameters

Name	Symbol	Description	Unit	Default*
Kv	K_v	Gain scale factor	V/decade	1.0
Dk	ΔK_v	Gain scale factor error	%	0.3[+]
Ib1	I_{b1}	Bias current at input I	A	5e-12
Ibr	I_{br}	Bias current at reference input R	A	5e-12
M	M	Number of decades over which N is specified		5
N	N	Log conformity error	%	0.1[+]
Vosout	V_{osout}	Output offset voltage	V	3e-3[+]
Rinp	R_{inp}	Amplifier input resistance	Ω	1e6
Fc	Fc	Amplifier voltage gain 2dB frequency	Hz	1e3
Ro	Ro	Amplifier output resistance	Ω	1e-3
Ntc	Ntc	Log conformity error temperature coefficient	%/Celsius	0.002
Vosouttc	$V_{osout}tc$	Output offset voltage temperature coefficient	V/Celsius	80e-6
Dktc	$\Delta K_v tc$	Gain scale factor temperature coefficient	%/Celsius	0.03
Ib1tc	$I_{b1}tc$	Input I bias current temperature coefficient	A/Celsius	0.5e-12[++]
Irtc	$I_r tc$	Input R bias current temperature coefficient	A/Celsius	0.5e-12 [++]
Tnom	$Tnom$	Parameter measurement temperature	Celsius	26.85

* The default parameters are for a typical integrated circuit logarithmic amplifier.
+ These parameters may be negative.

++ Typical bias current doubles for every eight to ten degrees temperature increase.

3.5.2 Verilog-A model code

```
//    Qucs generic logarithmic amplifier model:
//    This model can be used to construct working models for
//    a range of different manufacturer's logarithmic amplifier ICs −
//    for example the LOG100 and the MAX4206.
//    All required parameters can be extracted directly from manufacturers data sheets.
//    The structure and theoretical background to the logarithmic amplifier
//    Verilog−a model are presented in the Qucs log_amp report.
//
//    This is free software; you can redistribute it and/or modify
//    it under the terms of the GNU General Public License as published by
//    the Free Software Foundation; either version 2, or (at your option)
//    any later version.
//
//    Copyright (C), Mike Brinson, mbrin72043@yahoo.co.uk, January 2008.
//
`include "disciplines.vams"
`include "constants.vams"
//
 module log_amp (P_I1, P_Ir, P_Vout);
 inout P_I1, P_Ir;
 inout P_Vout;
//
 electrical P_I1, P_Ir, P_Vout;
//
// Internal nodes
//
 electrical n3, n4;
//
 `define attr(txt) (*txt*)
//
 parameter real Kv = 1.0 from [−inf : inf]
  `attr(info="scale_factor" unit = "V/decade");
 parameter real Dk = 0.3 from [−100 : 100]
  `attr(info="scale_factor_error" unit = "%");
 parameter real Ib1 = 5e−12 from [−inf : inf]
  `attr(info="input_I1_bias_current" unit = "A");
 parameter real Ibr = 5e−12 from [−inf : inf]
  `attr(info="input_reference_bias_current" unit = "A");
 parameter real M = 5 from [1 : inf]
  `attr(info="number_of_decades");
 parameter real N = 0.1 from [0 : 100]
  `attr(info="conformity_error" unit = "%");
 parameter real Vosout = 3e−3 from [−inf : inf]
  `attr(info="output_offset_error" unit = "V");
 parameter real Rinp = 1e6 from [1 : inf]
  `attr(info="amplifier_input_resistance" unit = "Ohm");
 parameter real Fc = 1e3 from [1 : inf]
  `attr(info="amplifier_3dB_frequency" unit = "Hz");
 parameter real Ro = 1e−3 from [1e−3 : inf]
  `attr(info="amplifier_output_resistance" unit = "Ohm");
parameter real Ntc = 0.002 from [−100 : 100]
  `attr(info="conformity_error_temperature_coefficient" unit = "%/Celsius");
parameter real Vosouttc = 80e−6 from [−inf : inf]
  `attr(info="offset_temperature_coefficient" unit = "V/Celsius");
```

```verilog
parameter real Dktc = 0.03 from [−100 : 100]
  `attr(info="scale_factor_error_temperature_coefficient" unit = "%/Celsius");
parameter real Ib1tc = 0.5e−12 from [−inf : inf]
  `attr(info="input_I1_bias_current_temperature_coefficient" unit = "A/Celsius");
parameter real Ibrtc = 0.5e−12 from [−inf : inf]
  `attr(info="input_reference_bias_current_temperature_coefficient" unit = "A/Celsius");
parameter real Tnom = 26.85 from [−273 : inf]
  `attr(info="parameter_measurement_temperature" unit = "Celsius");
//
real R, Ix;
real V1, V2;
real Cc, PI;
real TempK, TnomK, Tdiff, NTemp, VosoutTemp, DkTemp, Ib1Temp, IbrTemp;
//
analog begin
//
// Constants
//
PI=3.14159265358979323846;
//
// Model equations
//
V1=V(P_I1);
V2=V(P_Ir)+1e−20;
R=Rinp+1e−6;
Cc=1/(2*PI*Fc);
//
TempK=$temperature;
TnomK=Tnom+273.15;
Tdiff=TempK−TnomK;
NTemp=N+Ntc*Tdiff;
VosoutTemp=Vosout+Vosouttc*Tdiff;
DkTemp=Dk+Dktc*Tdiff;
Ib1Temp=Ib1+Ib1tc*Tdiff;
IbrTemp=Ibr+Ibrtc*Tdiff;
//
if (V1 >= V2 ) Ix = Kv*(1+DkTemp/100)*log(((V1/R)−Ib1Temp)/((V2/R)−IbrTemp))+
                     (Kv*2*(NTemp/100)*M)+VosoutTemp ;
else Ix = 0.0;
//
// Circuit stages
//
// Input stage
//
I(P_I1) <+ V(P_I1)/R;
I(P_Ir) <+ V(P_Ir)/R;
//
// Log function stage
//
I(n3) <+ −Ix;
I(n3) <+ V(n3);
//
// Frequency compensation
//
I(n4) <+ −V(n3);
I(n4) <+ V(n4);
I(n4) <+ ddt(Cc*V(n4));
//
// Output stage
I(P_Vout) <+ −V(n4)/Ro;
I(P_Vout) <+ V(P_Vout)/Ro;
end
```

endmodule

The ADMS syntax is a subset of Verilog-A. Allowed language structures are outlined in a SYNTAX-SUPPORTED file which can be downloaded from `http://mot-adms.sourceforge.net`.

3.6 Basic logarithmic amplifier operation

The circuit shown in Fig. 3.1 demonstrates the operation of the logarithmic amplifier macromodel with the input signals set as voltages. Input scaling resistance Rinp is 10kΩ, the reference voltage is 100 uV DC (which is equivalent to $Ir = 10$ nA), and the input signal Vs is swept between 1 uV and 10 V DC. For input signals ($V2$) less than $V1$ the macromodel restricts the output voltage to zero volts, preventing signal ratios from being less than one. Figure 3.2 illustrates the logarithmic amplifier operating in current input mode. In Fig. 3.2 the input scaling resistors are set to 1Ω, causing the amplifier inputs to become effectively virtual earth points. Both Fig. 3.1 and Fig. 3.2 illustrate the performance of a typical general purpose logarithmic amplifier over five signal decades, clearly showing the signal compression properties of the device.
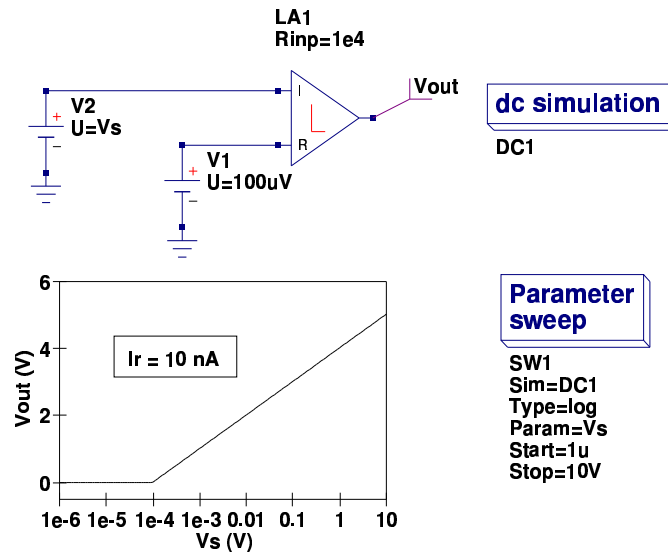
Figure 3.1: Qucs schematic for a basic voltage driven logarithmic amplifier and simulated DC transfer function
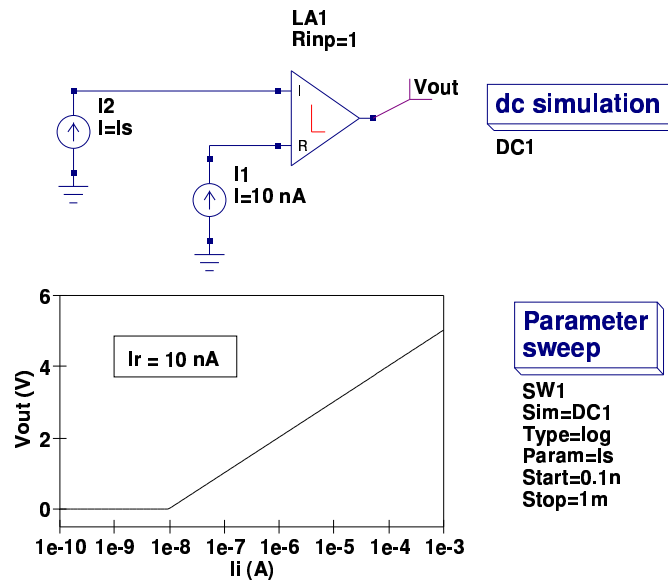
Figure 3.2: Qucs schematic for a basic current driven logarithmic amplifier and simulated DC transfer function

3.7 Functional description of the Verilog-A logarithmic amplifier macromodel

The macromodel of a general purpose logarithmic amplifier presented in previous sections simulates the following device characteristics:

- Input stage : Scaling resistances and DC bias currents.

- Gain stage : Logarithmic base ten transfer function over N decades with a single pole frequency response. Errors; gain scale and log conformity.

- Output stage : Resistance and DC offset voltage.

- Properties with temperature variation : gain scale, log conformity, bias currents and output offset voltage.

3.7.1 Input stage

The input stage is represented by the Verilog-A code listed in this subsection of the report. The voltages at input terminals P_{I1} and P_{Ir} are divided by scaling resistors R to give the required input currents. When R is small the input stage operates in current mode.

```
//
// Input stage
//
I(P_I1) <+ V(P_I1)/R;
I(P_Ir) <+ V(P_Ir)/R;
```

3.7.2 Gain stage

The gain stage is represented by the Verilog-A code listed in this subsection of the report. This part of the model determines the primary logarithmic amplifier transfer function, including logarithmic characteristics, frequency response, errors and temperature effects. An if-else statement is used to test if the log function argument is greater than one, setting the transfer function output to zero when the test fails. Both error terms and temperature factors have been included in the amplifier transfer function equation. Logarithmic amplifier transfer function frequency response is normally a complex function of input currents and an internal frequency compensation capacitance. Manufacturer's data sheets usually provide curves of frequency response for typical values of input current and compensation capacitance. Frequency effects are represented in the macromodel by a single pole response. The 3dB corner frequency being set by device parameter Fc. The default value of 3 kHz should be changed to suit the circuit operating conditions.

Figure 3.3 illustrates a small signal test circuit which allows amplifier transfer function frequency response to be simulated.

```
//
// Constants
//
PI=3.14159265358979323846;
//
// Model equations
//
V1=V(P_I1);
V2=V(P_Ir)+1e-20;
R=Rinp+1e-6;
Cc=1/(2*PI*Fc);
//
TempK=$temperature;
TnomK=Tnom+273.15;
Tdiff=TempK-TnomK;
NTemp=N+Ntc*Tdiff;
VosoutTemp=Vosout+Vosouttc*Tdiff;
DkTemp=Dk+Dktc*Tdiff;
Ib1Temp=Ib1+Ib1tc*Tdiff;
IbrTemp=Ibr+Ibrtc*Tdiff;
//
if (V1 >= V2 ) Ix = Kv*(1+DkTemp/100)*log(((V1/R)-Ib1Temp)/((V2/R)-IbrTemp))+
                      (Kv*2*(NTemp/100)*M)+VosoutTemp ;
else Ix = 0.0;
..
..
..
//
// Log function stage
//
I(n3) <+ -Ix;
I(n3) <+ V(n3);
//
// Frequency compensation
//
I(n4) <+ -V(n3);
I(n4) <+ V(n4);
I(n4) <+ ddt(Cc*V(n4));
```

3.7.3 Output stage

The output stage is represented by the Verilog-A code listed in this subsection of the report. This final section of the model introduces an output resistance $R0$. In a practical logarithmic amplifier this is often low in value.

```
//
// Output stage
I(P_Vout) <+ -V(n4)/Ro;
I(P_Vout) <+ V(P_Vout)/Ro;
```
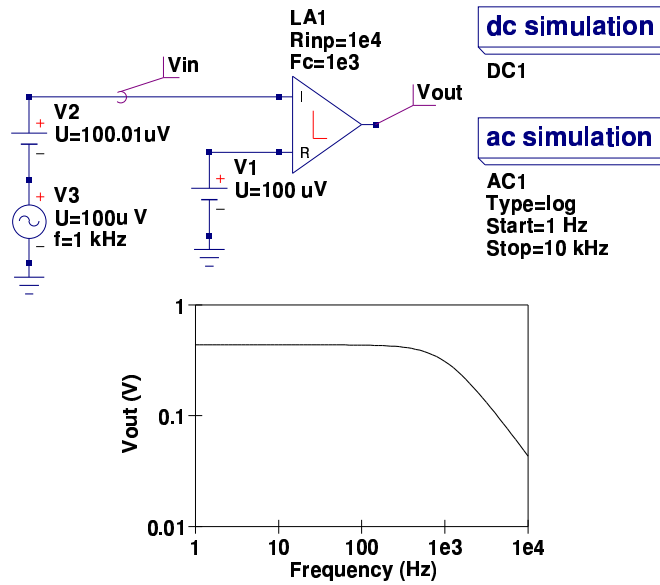
Figure 3.3: Qucs small signal AC test circuit and simulated transfer function

3.8 Logarithmic amplifier large signal AC response

The circuit and waveforms presented in Fig. 3.4 illustrate how the large signal AC response of a logarithmic amplifier can be tested and displayed. In this circuit a 100 Hz, 1.9 V peak sinewave signal, in series with a 2V battery, is applied to amplifier input I. Amplifier reference R has a 0.1V battery connected as the reference signal. As the logarithmic amplifier output signal is proportional to the log of the ratio of the input signals I/R, the shape of resulting output signal differs considerably from that of the excitation sinewave. When the peak of the input sinewave is 1.9 V the input ratio is 29 and when the negative peak reaches -1.9 V, the ratio is 1, yielding an output waveform with signal values between log(29) and 0. This example test circuit once again demonstrates the compression properties of logarithmic amplifiers.
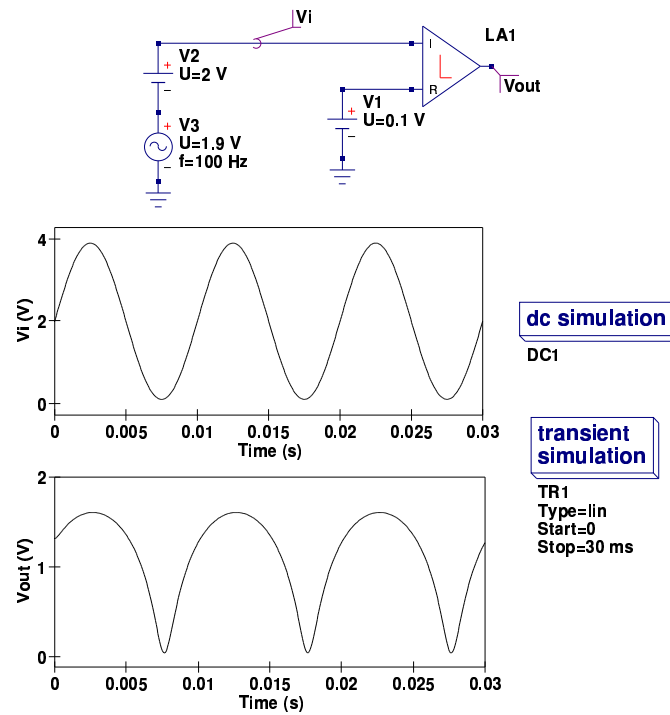
Figure 3.4: Qucs large signal logarithmic amplifier AC test circuit and simulated transient response

3.9 Logarithmic amplifier transfer function temperature variation

The test circuit shown in Fig. 3.5 introduces a double sweep which changes the amplifier circuit temperature from -110 to 100 Celsius, while simultaneously at each temperature, simulates, records and displays the amplifier transfer function. In this very basic example of amplifier response to temperature changes all circuit dependent parameters are varied at the same time and no attempt is made to identify individual parameter contributions to the overall temperature dependency. From the results of this simple test the simulation waveforms indicate that the typical temperature coefficients only have minimal effect on circuit performance. Modern integrated logarithmic amplifies are, in general, well temperature compensated, minimising the effects of temperature changes on circuit performance.
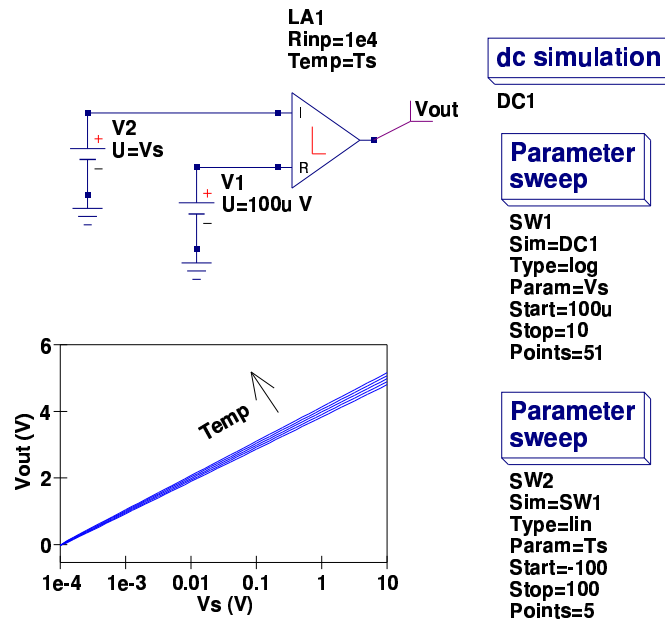
Figure 3.5: Qucs logarithmic amplifier transfer function test circuit and simulated temperature response

3.10 Logarithmic amplifier applications

In this section of this brief report two applications of the logarithmic amplifier are introduced. These examples are also used to demonsrate how Verilog-A modelled devices can be merged with other Qucs built-in components, subcircuits and equation defined devices. The application examples have been chosen to illustrate the power of Qucs modelling and simulation.

3.10.1 A simple signal multiplier

One of the most basic, and earliest, applications of the logarithmic amplifier was signal multiplication. Figure 3.6 shows a test circuit which includes two logarithmic amplifiers, two voltage controlled voltages sources (these act as a voltage summer) and an antilog amplifier.

The circuit illustrated in Fig. 3.6 multiplies the ratios of the two sets of inputs. This is done by taking the log of each input ratio then adding the results and finally finding the antilog of the sum. The antilog of a voltage can be found using the circuit shown in Fig. 3.7. Parameters Ir and K are used to scale the output voltage. This must be within the power supply range of a practical device. In the example shown in Fig. 3.6 the second input ratio is constant at 4 and the first has
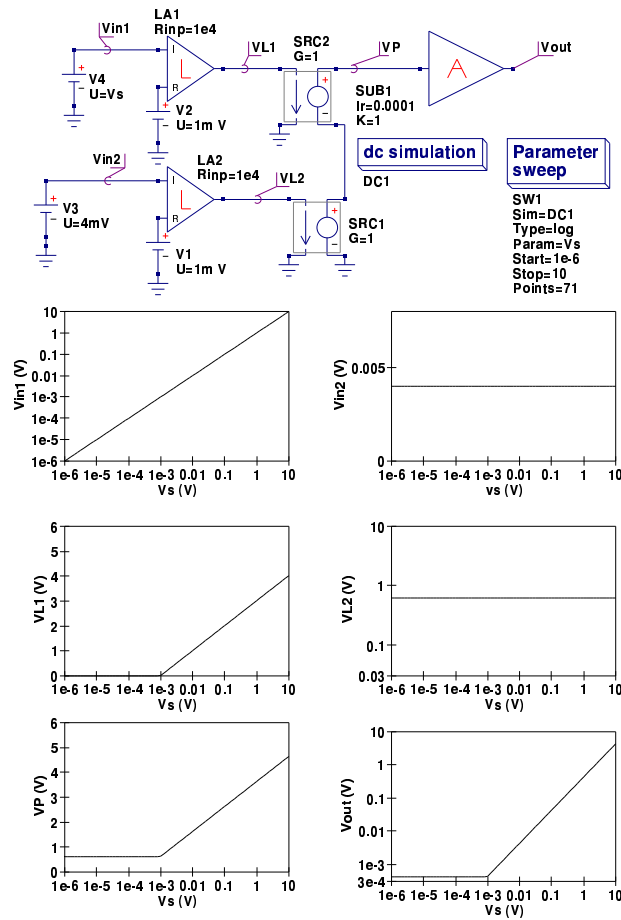
Figure 3.6: A simple electronic multiplier circuit and test waveforms

a maximum value of 10/1e-3 = 1e4, causing the output result to be 4e4 which is way beyond the output voltage of a practical circuit. By setting k=1 and Ir to 1e-4 the output voltage is scaled so that the maximum output becomes roughly 4 volts.

3.10.2 Light absorption measurements using photodiodes and a log amplifier

A number of years ago a colleague in Germany made the profound remark that the 20th century was the century of the electron and that the 21st century was likely to be the century of the photon. The current state of Qucs model development does to some extent reflect this thinking. At present the major modelling tasks involve extending the Qucs component/model range and the improvement of their indi-
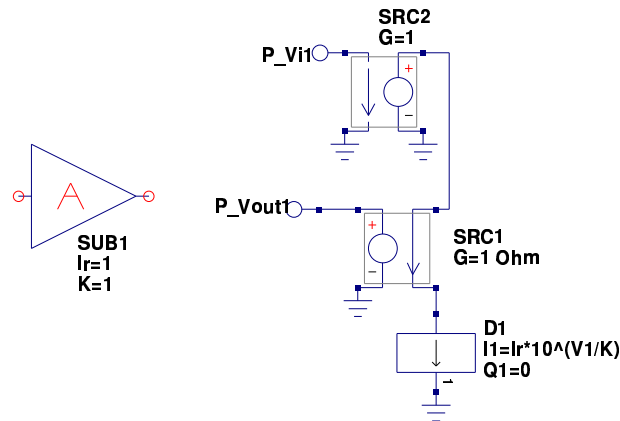
Figure 3.7: A simple antilog amplifier based on a single equation defined device: Controlled source SRC2 acts as a buffer and source SCR1 converts EDD current I1 to voltage

vidual performances. Looking through the current device/model list immediately confirms that it is dominated by electrical components and other domain devices, such as transducers, actuators and optical components, are non-existant. The logarithmic amplifier application reported here attempts to address this imbalance by providing some preliminary information on an area of modelling where significant work is likely to be done in the coming cycles of Qucs development, namely optical components. Optoelectronic devices, such as LEDs and photodiodes, are well established in the current electronics scene. Unfortunately, to my knowledge, Qucs lacks the models for such devices. The test circuit shown in Fig. 3.8 presents a basic arrangement for measuring the light absorption properties of liquids. This is a classical application of the logarithmic amplifier, working as a device that measures the ratio of two currents. These currents are generated by photodiodes.

In Fig. 3.8 a light source shines on two photodiodes: directly on one and indirectly on the other. In the second case the light travels through a vessel containing a liquid which attenuates the light. The level of attenuation being dependent on the light absorption properties of the liquid. When little or no light is absorbed the logarithmic amplifier output is small. However, at high light absorption levels the amplifier output is high. The circuit given in Fig. 3.8 works over six decades of light transmission coefficient. Moreover, it does not require the absolute values of the light intensities to be measured. The circuit output voltage is compressed to a range between 0 and 5 volts which is ideal for interfacing with a unipolar analogue-to-digital converter. Qucs cannot represent light signals directly. However, it is possible to use a voltage to represent the intensity of a light. Once one realises that the numerical value of light intensity can be represented by a voltage of the

same numerical value then it also becomes possible to represent light signals paths by nets with specific voltage values. In Fig. 3.8 the light paths are represented by wires connecting the light source, the liquid vessel and the photodiodes. Models for the light absorbing liquid and the photodiodes are shown in Fig. 3.9.
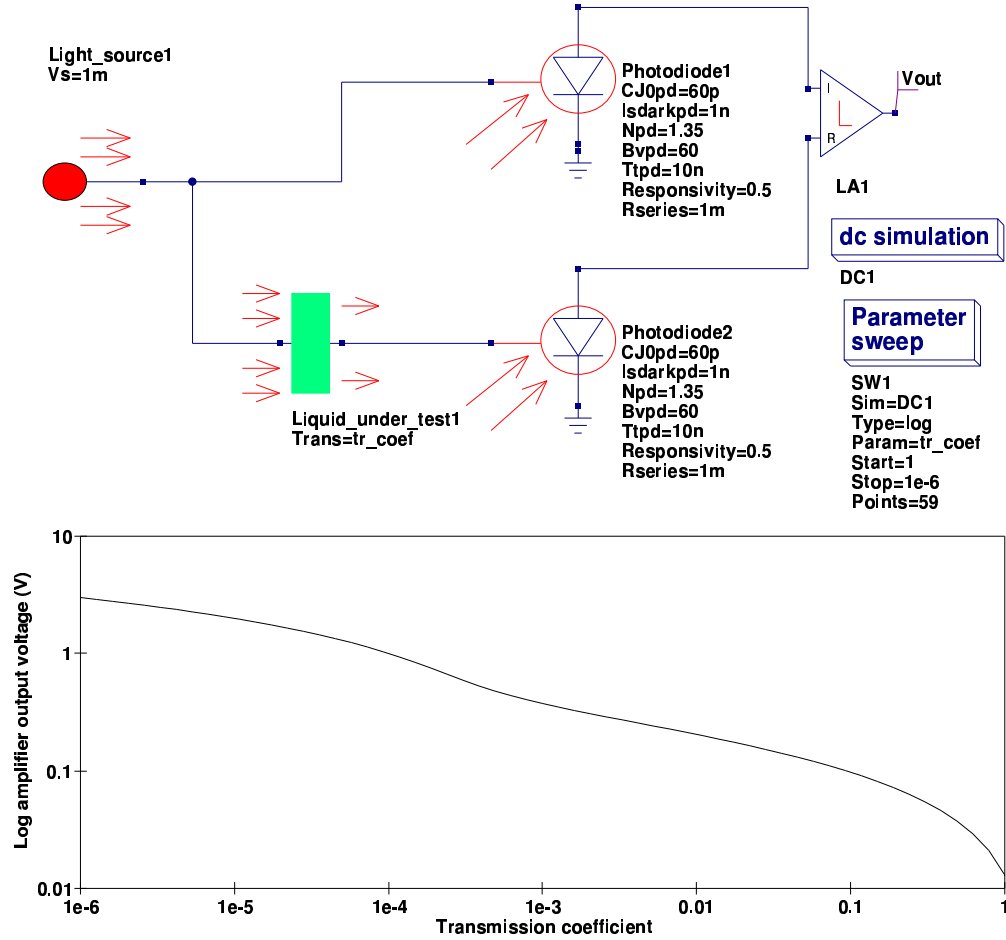
Figure 3.8: Light absorption measurements using photodiodes and a logarithmic amplifier

In Fig. 3.9 the light absorbing liquid is represented by a voltage controlled voltage source where the source gain models the transmission coefficient: a gain of one implies 100% transmission and a gain of zero no light transmission at all. The photodiode model is much more complex. A voltage controlled current source in parallel with a diode forms the core of the model. The gain of the controlled source represents the responsivity of the diode. Responsivity is expressed in amperes per watt or as the photodiode current for a given input light power per unit area. As the current generated is also expressed as the current per unit area, effectively
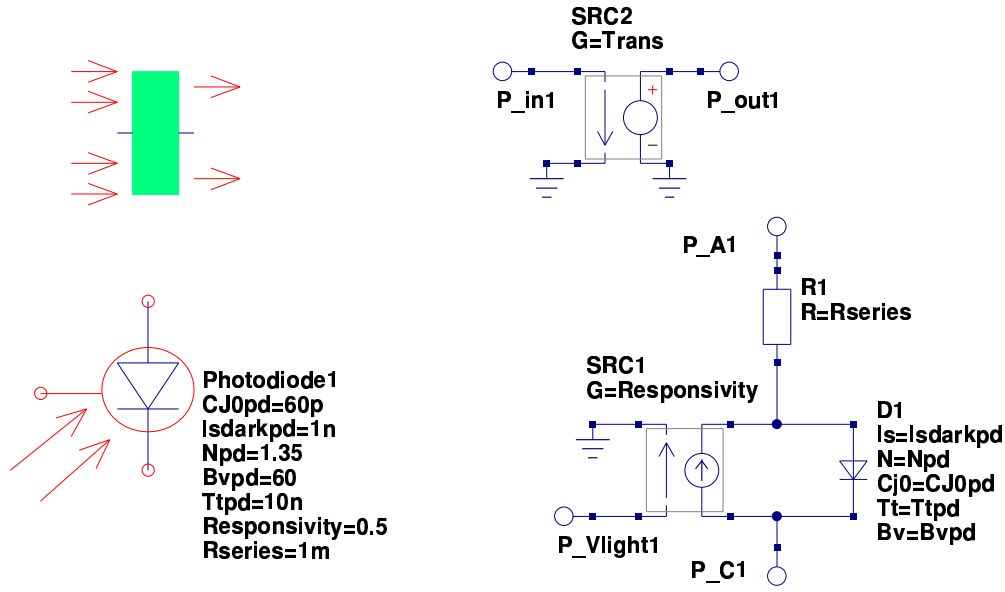
Figure 3.9: Experimental Qucs models for a light absorbing liquid and a photodiode

the area factor is eliminated. Equation (3.8) relates the photodiode current to the voltage at the light input terminal:

$$Id = Responsivity \cdot V(PVlight) \tag{3.8}$$

Responsivity is a function of the light wavelength and is around 0.5 to 0.6 for wavelengths of 900nm. It is roughly 0.3 for blue wavelengths and drops to 0.2 in the ultraviolet region of the spectrum. With no illumination falling on a photodiode the resulting diode current is small, being called the device dark current. It ranges from tens of pA for ultraviolet detecting photodiodes to several μA for low cost silicon diodes. Typical values for Rseries are in the range 1m to 20mΩ. The other parameters are similar to those representing standard semiconductor diodes but with values specifically chosen to represent the properties of photodiodes.

3.11 End note

These notes summarise a basic technique for modelling logarithmic amplifiers using Verilog-A. Verilog-A is primarily intended for modelling compact semiconductor devices. However, as this report demonstrates it can be equally employed for macromodelling of integrated circuits and circuits in general. The Qucs C++ code for the modular operational amplifier model, generated by ADMS, can be

found in the Qucs release archives at `http://qucs.sourceforge.net/download.html`. The procedure undertaken to convert the Verilog-A code into C++ for compilation and linking with Qucs closely followed that described by Stefan Jahn and Hélène Parruitte. Although it takes more work to construct models using ADMS the effort is worthwile because the finished models are very efficient in terms of memory usage and have significantly reduced run times. It is worth noting that Verilog-A based models can be combined with other forms of Qucs model, forming a powerful combination that extends Qucs simulation capabilities. Once again a special thanks to Stefan Jahn for all his help and encouragement over the period that I have been developing the Verilog-A logarithmic amplifier macromodel, writing this report and testing the examples it includes.

4 Verilog-A Macromodel for Resistive Potentiometers

4.1 Introduction

The resistive potentiometer is a common component in electronic systems. Unfortunately, it is often very poorly modelled by circuit simulators. A common approach is to simply treat the device as two series resistances with values set by linear or logarithmic algebraic equations. This is fine as a rough and ready approach to modelling the basic potentiometer function. However, it fails to acknowledge the fact that potentiometers are much more complex devices that involve not only electrical characteristics but also mechanical rotational features. Furthermore, potentiometers are subject to a number of errors such as taper conformity and linearity. This report presents a more realistic model of a resistive potentiometer and shows how Qucs can be used to develop practical models of this fundamental component that give an order of magnitude improvement over the commonly used two resistor models. Finally, the notes show how a Qucs subcircuit of the potentiometer can be coded as a compact Verilog-A macromodel.

4.2 The two resistor potentiometer model

The schematic for a two resister model of a linear resistive potentiometer is shown in Fig. 4.1. Sweep parameter *Position* is used to control the movement of the potentiometer central output terminal. A range of zero to one covers the movement of the output terminal from one end of the potentiometer to the other. This model acts more like a sliding arm potentiometer rather than the more common rotary type. Notice the use of non-integer values 1.00001 and 0.00001 to ensure that R1 or R2 do not become zero in value when the mid-terminal is slid to either end of the potentiometer. Changing equations R1eqn and R2eqn allows different laws to be specified that relate the values of resistors R1 and R2 to the position of the mid-terminal.
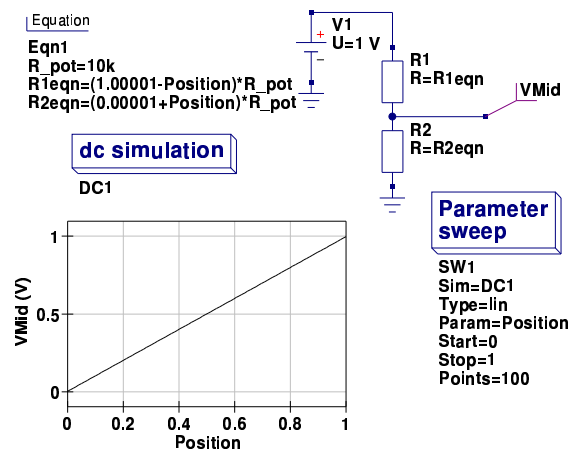
Figure 4.1: Basic two resistor model of a linear resistive potentiometer

4.3 Potentiometer types

The basic types of resistive potentiometer are the rotary and slider forms[1]. These are constructed from wire, cemet or conductive plastic resistive strips attached to a suitable non-conducting holder with fixed contacts at the ends and a moveable wiper contact. The moveable contact either traverses in a rotary direction or in a straight line. The resistive strip has either a constant width or is shaped depending on the type of potentiometer, giving either constant resistance or changing resistance per unit length along the length of a potentiometer. Probably the most common of the non-linear types of potentiometer is the logarithmic potentiometer. Figure 4.2 illustrates how linear, logarithmic and inverse logarithmic device laws vary as a function of wiper contact angle. Logarithmic potentiometers are sometimes called audio potentiometers due to their use as volume control devices in audio amplifiers. Potentiometers with logarithmic resistive laws are necessary because the human ear responds to the logarithm of loudness, making control of amplifier output level smoother with a logarithmic potentiometer rather than the simpler linear device. The shape of the potentiometer response law is often called the potentiometer taper. Today linear potentiometers are the most common type made by different manufacturers. One reason for this is the cost of manufacturing a true logarithmic law device. In many cases so called logarithmic potentiometers are in fact simpler devices constructed from two different sections of resistive material where the low angle part of the potentiometer track has a smaller resistivity than the high angle section. Logarithmic potentiometers are normally specified as

---

[1]R.G. Keen, The secret life of pots, 1999, `http://www.geofex.com/Article_Folders/potsecrets/potscret.htm`

20% or 10% types, being defined as the percentage of total potentiometer resistance between the bottom fixed cantact and the wiper contact when the wiper is moved to the middle of it's full range. Inverse logarithmic potentiometers have a similar characteristic to the logarithmic potentiometer except that potentiometer taper is reversed. This has the effect of causing large changes in output at small wiper angles and the reverse at high angles. Inverse logarithmic potentiometers have in recent years become unusual items and are no longer widely manufactured.
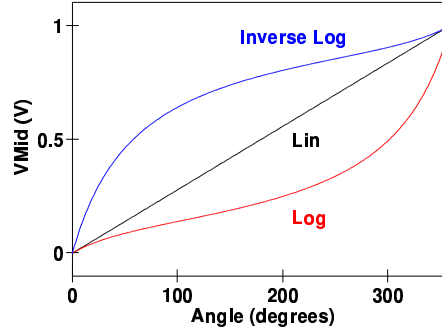
Figure 4.2: Resistive potentiometer characteristics: Vin = 1V DC

4.4 Modelling potentiometer taper laws

Due to their lower cost, and availability, linear potentiometers are often chosen as a starting point when designing a circuit that requires a potentiometer with a specific taper law. The same applies to modelling potentiometers with a known resistive taper. By adding a fixed resistance, external to a potentiometer, a linear potentiometer can be converted to one with a designer specified taper. Consider the three resistor diagram shown in Fig. 4.3.

The transfer function for the schematic illustrated in Fig. 4.3 is given in equation (4.1).

$$\frac{Vout}{Vin} = \frac{(R_2 \parallel R_T)}{R_1 + (R_2 \parallel R_T)} = \frac{1}{1 + \dfrac{R_1}{R_T} + \dfrac{R_1}{R_T}} \tag{4.1}$$

Writing $Rpot = R_1 + R_2$ and setting $a = R_2/Rpot$ and $TaperCoeff = R_T/Rpot$ yields equation (4.2).
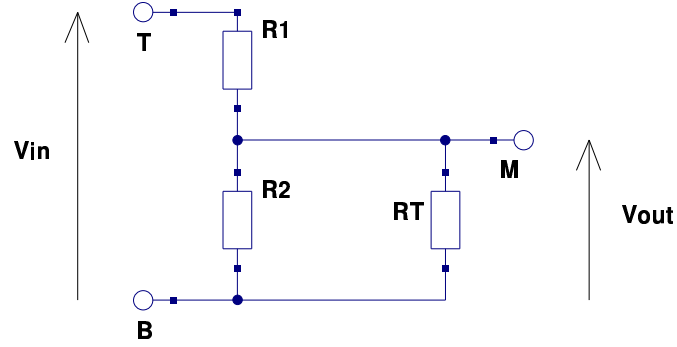
Figure 4.3: Basic two resister potentiometer with taper resister RT connect between the bottom fixed contact (B) and the wiper arm contact (M)

$$\frac{Vout}{Vin} = \frac{1}{1 + \dfrac{1-a}{a} + \dfrac{1-a}{TaperCoeff}} = \frac{a \cdot TaperCoeff}{a - a^2 + TaperCoeff} \qquad (4.2)$$

Where $TaperCoeff$ is the potentiometer taper coefficient (a fixed value greater than zero) and parameter a is a measure of the wiper arm position (being in the range 0 to 1). As the taper coefficient approaches a large value (much greater than one) the potentiometer law approaches that of a simple linear potentiometer, effectively this implies that $R_T >> Rpot$. For rotary potentiometers, parameter a can also be expressed as $a = Angle/MaxAngle$. In the case of a single turn potentiometer $0 <= Angle <= 360$ degrees and $MaxAngle = 360$ degrees. Figure 4.4 illustrates the effect of different $TaperCoeff$ values on the potentiometer response curves. Logarithmic potentiometer response is approximated when the $TaperCoeff$ is in the range 0.2 to 0.25.
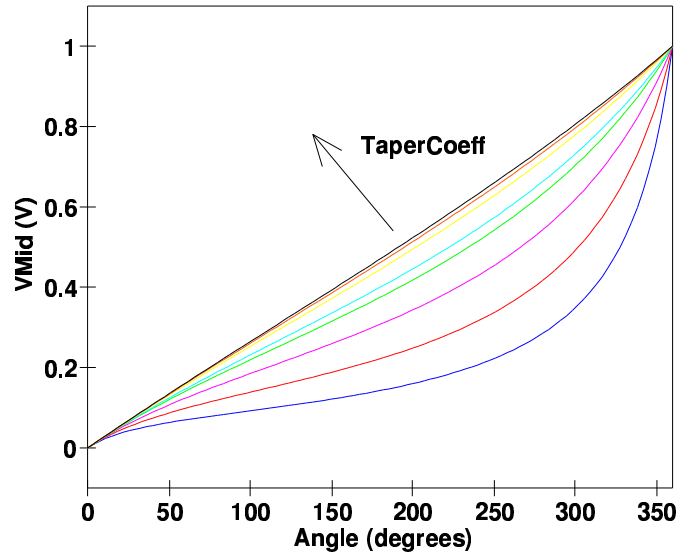
Figure 4.4: Potentiometer response curves for different $TaperCoeff$ values: curves blue to black; $TaperCoeff = 0.1, 0.2, 0.75, 1, 2, 3, 4, 5$

4.5 Modelling inverse logarithmic potentiometers

By placing the fixed taper resistance in parallel with resistor R1 the potentiometer response changes to the inverse logarithmic form shown in Fig. 4.2. Consider the three resister model shown in Fig. 4.5.
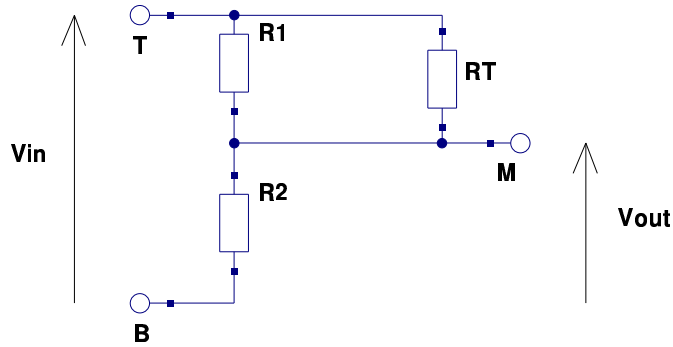
Figure 4.5: Basic two resister potentiometer with taper resister RT connect between the top fixed contact (T) and the wiper arm contact (M)

The transfer function for the schematic illustrated in Fig. 4.5 is given in equation (4.3).

$$\frac{Vout}{Vin} = \frac{R_2}{(R_1 \parallel R_T) + R_2} = \frac{R_2}{\left(\dfrac{R_1 \cdot R_T}{R_1 + R_T}\right) + R_2} = \frac{R_1 \cdot R_2 + R_2 \cdot R_T}{R_1 \cdot R_2 + R_1 \cdot R_T + R_2 \cdot R_T}$$

$$(4.3)$$

Writing $Rpot = R_1 + R_2$ and setting $a = R_2/Rpot$ and $TaperCoeff = R_T/Rpot$ yields equation (4.4).

$$\frac{Vout}{Vin} = \frac{a - a^2 + a \cdot TaperCoeff}{a - a^2 + TaperCoeff} \qquad (4.4)$$

Figure 4.6 illustrates the effect of different $TaperCoeff$ values on the inverse potentiometer response curves. Inverse logarithmic potentiometer response is approximated when the $TaperCoeff$ is in the range 0.2 to 0.25.
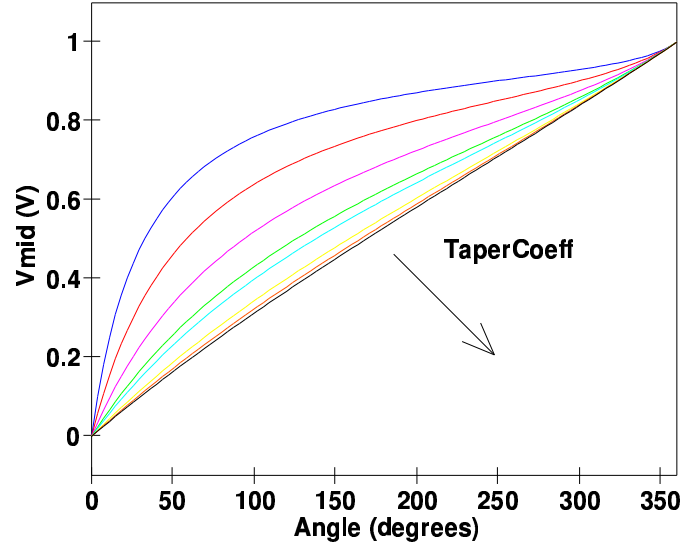
Figure 4.6: Inverse potentiometer response curves for different $TaperCoeff$ values: curves blue to black; $TaperCoeff = 0.1, 0.2, 0.75, 1, 2, 3, 4, 5$

4.6 A Qucs subcircuit model of a resistive potentiometer

The schematic for a resistive potentiometer modelled as a Qucs subcircuit is shown in Fig. 4.7. This model is based on the background ideas and theory presented in

the previous sections of this report. The subcircuit schematic represents a practical resistive potentiometer, with electrical and positional characteristics set by a resistive taper coefficient, a conformity error parameter, a linearity parameter, wiper arm contact resistance and resistance temperature coefficient. The subcircuit models both single and multi-turn potentiometers of type linear, logarithmic or inverse logarithmic and indeed any other laws set by the value of parameter Taper_Coeff. The following list outlines the function and units of the model parameters.

4.6.1 Parameters

Name	Symbol	Description	Unit	Default*
Rpot	$Rpot$	Nominal device resistance	Ω	1e4
Rotation	$Rotation$	Shaft/wiper arm rotation	degrees	120
Taper_Coeff	$TaperCoeff$	Resistive law taper coefficient		0
LEVEL	$LEVEL$	Device type selector		1[+]
Temp	$Temp$	Circuit temperature	Celsius	26.85
Max_Rotation	$MaxRotation$	Maximum shaft/wiper rotation	degrees	240[++]
Conformity	$Conformity$	Conformity error	%	0.2
Linearity	$Linearity$	Linearity error	%	0.2
Tnom	$Tnom$	Parameter measurement temperature	Celsius	26.85
Contact_Res	$ContactRes$	Wiper arm contact resistance resistance	Ω	1
Temp_Coeff	$TempCoeff$	Resistance temperature coefficient	PPM/Celsius	100[+++]

* The default parameters are for typical 10k cemet linear potentiometer.
+ Parameter LEVEL selects the potentiometer type: LEVEL=1; linear, LEVEL=2; logarithmic and LEVEL=3; inverse logarithmic.
++ For a single turn potentiometer 240 <= MaxRotation <= 360 degrees [2]. For multi-turn potentiometers add 360 degrees per complete shaft rotation.
+++ Potentiometer temperature coefficients are normally quoted in parts per million (PPM) per degree Celsius by manufacturers. Typical values are: cemet ± 100 PPM/Celsius, wire ± 50 PPM/Celsius and conductive plastic ± 1000 PPM/Celsius.

Potentiometer functional errors are introduced in the model illustrated in Fig. 4.7 by adding conformity and linearity factors to the basic taper coefficient via equation (4.5).

---

[2]The value of parameter MaxRotation is normally set by the position of an end stop at the end of the resistive track.

$$Tpcoeff = TaperCoeff + \frac{Conformity + Linearity \cdot \sin{(RadAngle)}}{100} \qquad (4.5)$$

Where $RadAngle = Rotation \cdot \pi/180$ is the shaft rotation angle in radians. Absolute conformity error is defined as the maximum deviation of the taper function characteristic from an ideal theoretical characteristic. Conformity error is normally specified in percentage and can have plus or minus values. Absolute linearity has a similar definition. However, it often shows sinusoidal characteristics[3] due to shaft rotation, particularly with multi-turn potentiometers. Contact resistance has been introduced into the model to take account of any resistance that exists from the wiper terminal to the contact on the potentiometer resistive track. With sprung loaded contacts this resistance should be small and can often be neglected. Effects of temperature on the model performance are included via a simple linear temperature coefficient which controls the change in device resistance as temperature varies. A potentiometer test circuit and the response of a multi-turn linear potentiometer with zero conformity and linearity errors are given in Fig. 4.8. Multi-turn potentiometers are modelled by setting the shaft rotation parameter to multiplies of 360 degrees plus slightly less angle for the final rotation. Figure 4.9 illustrates a test circuit for comparing the performace of log and inverse log single-turn potentiometers. The response curves show both simulation results and those obtained by fitting logarithmic curves to the simulated output data. With the Taper_Coeff parameter set to 0.2, remarkably good agreement is observed when the simulation data is compared to the ideal logarithmic performance. The test circuit and simulation curves illustrated in Fig. 4.10 demonstrate the effects that conformity and linearity parameters have on a multi-turn potentiometer. Normally both of these parameters are much less than one percent, making their effects difficult to observe. In Fig. 4.10 they have both been set at ten percent, greatly exaggerating their effect on potentiometer performance.

---

[3]See section 3.2: Novotechnik Potentiometer Definitions, `http://www.novotechnik.com/novotechnic_tech_ref.html`

P_bot1

R4
R=Rbot

R1
R=RTB

R5
R=Contact_Res

R3
R=Rtop

R2
R=RTT

P_top1

P_mid1
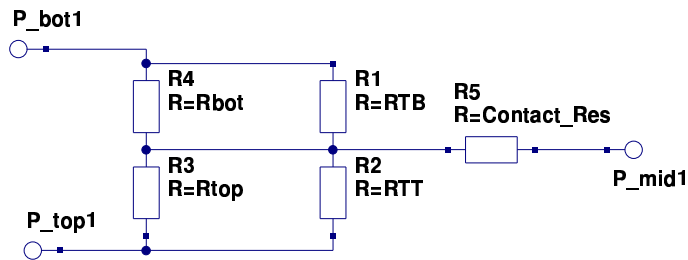
| Equation

Eqn1
Rtop=(1.000001-(Rotation/(Max_Rotation+1e-20)))*R_pot_Temp
Tpcoeff=Taper_Coeff+(Conformity+Linearity*sin(Rad_Angle))/100
Rad_Angle=Rotation*pi/180
R_pot_Temp=R_pot*(1+Temp_Coeff*(Temp-Tnom)/1e6)
Rbot=(0.000001+(Rotation/(Max_Rotation+1e-20)))*R_pot_Temp
RTB=(LEVEL==2 ? (Taper_Coeff != 0.0) ? R_pot_Temp*Tpcoeff : 1e15 : 1e15)
RTT=(LEVEL==3 ? (Taper_Coeff != 0.0) ? R_pot_Temp*Tpcoeff : 1e15 : 1e15)

B   T
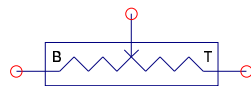
POT1
R_pot=10k
Rotation=120
Taper_Coeff=0
LEVEL=1
Temp=26.85
Max_Rotation=240
Conformity=0.2
Linearity=0.2
Tnom=26.85
Temp_Coeff=100
Contact_Res=1

Figure 4.7: Qucs subcircuit of a resistive potentiometer: schematic, equations and symbol

dc simulation

DC1

VMid

B   T

V1
U=1 V

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Angle
Start=0
Stop=1670
Points=168

POT1
R_pot=10k
Rotation=Angle
Taper_Coeff=0
LEVEL=1
Temp=26.85
Max_Rotation=1670
Conformity=0
Linearity=0
Tnom=26.85
Temp_Coeff=100
Contact_Res=1
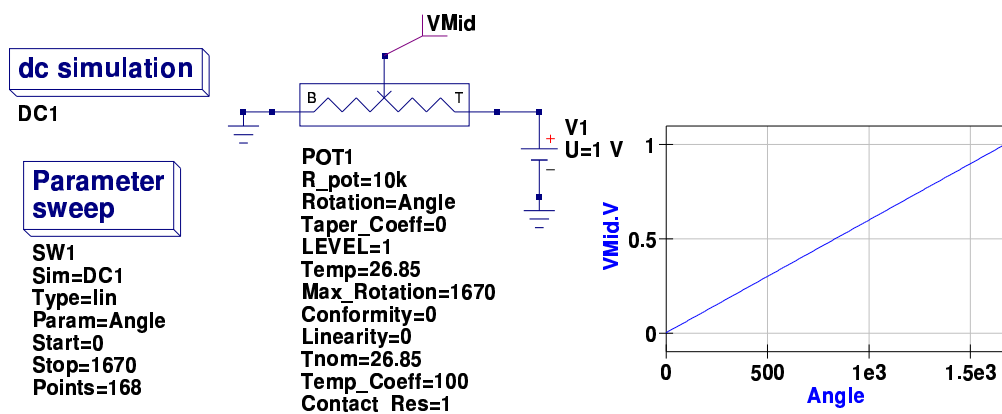
Figure 4.8: Multi-turn linear potentiometer test circuit and performance curve

dc simulation

DC1

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Angle
Start=0
Stop=280
Points=281

VMPot1

POT1
R_pot=10k
Rotation=Angle
Taper_Coeff=0.2
LEVEL=2
Temp=26.85
Max_Rotation=280
Conformity=0
Linearity=0
Tnom=26.85
Temp_Coeff=100
Contact_Res=1

VMPot2

POT2
R_pot=10k
Rotation=Angle
Taper_Coeff=0.2
LEVEL=3
Temp=26.85
Max_Rotation=280
Conformity=0
Linearity=0
Tnom=26.85
Temp_Coeff=100
Contact_Res=1

V1
U=1 V

Equation
Eqn1
TFPot1=1-k*log10((Max+Min-Angle)/Min)
Min=10
k=0.66
Max=280

Equation
Eqn2
k1=0.66
TFPot2=k1*log10((Angle+Min)/Min)

Figure 4.9: Single-turn log and inverse log potentiometer test circuit and performance curves: (1) simulated data solid curves, (2) fitted data dotted curves

VMPot1

B   T

POT1
R_pot=10k
Rotation=Angle
Taper_Coeff=0.2
LEVEL=2
Temp=26.85
Max_Rotation=1720
Conformity=0
Linearity=0
Tnom=26.85
Temp_Coeff=100
Contact_Res=1

VMPot2

B   T

POT2
R_pot=10k
Rotation=Angle
Taper_Coeff=0.2
LEVEL=2
Temp=26.85
Max_Rotation=1720
Conformity=10
Linearity=0
Tnom=26.85
Temp_Coeff=100
Contact_Res=1

VMPot3

B   T

POT3
R_pot=10k
Rotation=Angle
Taper_Coeff=0.2
LEVEL=2
Temp=26.85
Max_Rotation=1720
Conformity=10
Linearity=10
Tnom=26.85
Temp_Coeff=100
Contact_Res=1

V1
U=1 V

dc simulation
DC1

Parameter sweep
SW1
Sim=DC1
Type=lin
Param=Angle
Start=0
Stop=1720
Points=173

Figure 4.10: Multi-turn log potentiometer test circuit and response curves, illustrating conformity and linearity errors: both conformity and linearity parameters are set at ten percent to clearly indicate their effects; blue curve Pot1, red curve Pot2 and black curve Pot3

4.7 A Verilog-A resistive potentiometer model

One of the advantages of using Qucs subcircuits, and indeed EDD models, is that they allow fast interactive prototyping of new models. Once a new model is tested and functioning satisfactorily it can often be easily translated into Verilog-A code for compiling into C++ code, using the ADMS compiler, and finally compiled and linked with the main Qucs software. When prototyping new models it is advisable to use, whenever possible, circuit elements/structures that can be easily transposed to Verilog-A statements. The Verilog-A code listed in the next section is based directly on the Qucs subcircuit model of the potentiometer. In general there is a one-to-one correspondence between the different sections of the subcircuit model and the Verilog-A code. However, for completeness the Verilog-A code also includes white noise statements to account for resistor noise[4]. Overall the subcircuit model and the Verilog-A code give the same performance except that the Verilog-A code appears to simulate much faster. This is not surprising considering it comprises compiled and linked machine code rather than being a netlist which is interpreted during circuit simulation.

4.8 Verilog-A model code

```
//      Qucs  resistive  potentiometer  model:
//      This model can be used to construct working linear , logarithmic and inverse logarithmic
//      potentiometers , or other devices with designer specified resistive taper functions.
//      All required parameters can be extracted directly from
//      manufacturers data sheets.
//
//      The structure and theoretical background to the potentiometer
//      Verilog−a model are presented in the Qucs potentiometer report.
//
//      This is free software; you can redistribute it and/or modify
//      it under the terms of the GNU General Public License as published by
//      the Free Software Foundation; either version 2, or (at your option)
//      any later version.
//
//      Copyright (C), Mike Brinson , mbrin72043@yahoo.co.uk , February 2008.
//
`include ”disciplines.vams”
`include ”constants.vams”
//
 module potentiometer (B, M, T);
 inout B, M, T;
//
 electrical B, M, T;
//
// Internal nodes
//
 electrical n1;
```

---

[4]The resistors comprising the potentiometer subcircuit have thermal noise built-in as a standard component feature.

```
//
'define attr(txt) (*txt*)
//
 parameter real R_pot = 1e4 from [1e−6 : inf]
  'attr(info="nominal_device_resistance" unit = "Ohm");
 parameter real Rotation = 120 from [0 : inf]
  'attr(info="shaft/wiper_arm_rotation" unit = "degrees");
 parameter real Taper_Coeff = 0 from [0 : inf]
  'attr(info="resistive_law_taper_coefficient" );
 parameter integer LEVEL = 1 from [1 : 3]
  'attr(info="device_type_selector" );
 parameter real Max_Rotation = 240.0 from [0 : inf]
  'attr(info="maximum_shaft/wiper_rotation" unit = "degrees");
 parameter real Conformity = 0.2 from [−inf : inf]
  'attr(info="conformity_error" unit = "%");
 parameter real Linearity = 0.2 from [−inf : inf]
  'attr(info="linearity_error" unit = "%");
 parameter real Contact_Res = 1 from [1e−6 : inf]
  'attr(info="wiper_arm_contact_resistance" unit = "Ohm");
 parameter real Temp_Coeff = 100 from [0 : inf]
  'attr(info="resistance_temperature_coefficient" unit = "PPM/Celsius");
 parameter real Tnom = 26.85 from [−273 : inf]
  'attr(info="parameter_measurement_temperature" unit = "Celsius");
//
real Rad_Angle, R_pot_Temp, Rtop, Rbot, Tpcoeff, Rcontact;
real RTB, RTT, error_term;
real fourkt;
//
analog begin
//
// Model equations
//
Rcontact=Contact_Res+1e−6;
Rad_Angle=Rotation*'M_PI/180;
R_pot_Temp=(R_pot+1e−6)*(1+Temp_Coeff*($temperature−Tnom)/1e6);
Tpcoeff=Taper_Coeff+(Conformity+Linearity*sin(Rad_Angle))/100;
error_term=(1+(Conformity+Linearity*sin(Rad_Angle))/100);
//
case (LEVEL)
    2: begin
         RTB=R_pot_Temp*Tpcoeff;
         RTT=1e15;
         Rtop=(1.000001−(Rotation/(Max_Rotation+1e−20)))*R_pot_Temp;
         Rbot=(0.000001+(Rotation/(Max_Rotation+1e−20)))*R_pot_Temp;
       end
    3: begin
         RTB=1e15;
         RTT=R_pot_Temp*Tpcoeff;
         Rtop=(1.000001−(Rotation/(Max_Rotation+1e−20)))*R_pot_Temp;
         Rbot=(0.000001+(Rotation/(Max_Rotation+1e−20)))*R_pot_Temp;

       end
  default: begin
         RTB=1e15;
         RTT=1e15;
         Rtop=(1.000001−(Rotation/(Max_Rotation+1e−20)))*R_pot_Temp*error_term;
         Rbot=(0.000001+(Rotation/(Max_Rotation+1e−20)))*R_pot_Temp*error_term;

       end
endcase
//
if (Taper_Coeff == 0.0)
```

```
    begin
      RTB=1e15;
      RTT=1e15;
      Rtop=(1.000001−(Rotation/(Max_Rotation+1e−20)))*R_pot_Temp*error_term;
      Rbot=(0.000001+(Rotation/(Max_Rotation+1e−20)))*R_pot_Temp*error_term;
    end
  //
  // Macromodel
  //
  I(T, n1) <+ V(T, n1)/Rtop;
  I(T, n1) <+ V(T, n1)/RTT;
  I(B, n1) <+ V(B, n1)/Rbot;
  I(B, n1) <+ V(B, n1)/RTB;
  I(M, n1) <+ V(M, n1)/Rcontact;
  //
  // Noise contributions
  //
  fourkt=4.0*'P_K*$temperature;
  I(T, n1) <+ white_noise(fourkt/Rtop, "thermal");
  I(T, n1) <+ white_noise(fourkt/RTT, "thermal");
  I(B, n1) <+ white_noise(fourkt/Rbot, "thermal");
  I(B, n1) <+ white_noise(fourkt/RTB, "thermal");
  I(M, n1) <+ white_noise(fourkt/Rcontact, "thermal");

  //
  end
  endmodule
```

The ADMS syntax is a subset of Verilog-A. Allowed language structures are outlined in a SYNTAX-SUPPORTED file which can be downloaded from `http://mot-adms.sourceforge.net`. Recent news concerning the ADMS Verilog-A compiler can also be found on the ADMS Website Wiki.

4.9 End note

Resistive potentiometers are fundamental components in electronic system design. They deserve due attention when forming part of a circuit simulation package. This report introduces a number of basic properties of potentiometric devices and outlines how they can be accurately and efficiently simulated by Qucs. While writing this report I have tried to demonstrate how practical resistive potentiometer models can be constructed from a set of basic concepts. Qucs subcircuits and component defining equations form the fundamental tools for modelling a potentiometer. Both encourage interactive model development and the subsequent conversion of their properties into Verilog-A code. My thanks to Stefan Jahn for all his help and encouragement during the period I have been working on the potentiometer model and writing this report.

5 Verilog-A compact device models for GaAs MESFETs

5.1 Introduction

A previous Qucs Report[1] described a MESFET model based on an equation defined device (EDD) representation of the level 1 Curtice model. This model evolved as a test example during the initial Qucs EDD development phase. Today the EDD model is popular amongst Qucs users as either a powerful non-linear component in it's own right or as the basis of a component prototyping system for constructing compact Verilog-A device models, translated with ADMS to C++ code, compiled to object code and finally linked to the main body of the Qucs program code. Over the last year the Qucs development team has invested a significant amount of time improving both EDD prototyping and Verilog-A compact device/circuit model development, making the development process more transparent to anyone interested in trying their hand at model construction. One branch of the current Qucs modelling activities is concentrating on adding new models which fill in some of the gaps in the Qucs released model lists. One such model in this category is the GaAs MESFET. This report outlines the background and mathematical basis for a number of MESFET models. These have been coded in Verilog-A and tested using recent Qucs CVS code. They will be included in the next full release of Qucs.

5.2 The GaAs MESFET

The metal-semiconductor FET (MESFET) is a Schottky-barrier gate FET which is normally made from Gallium Arsenide. It is a popular device for high frequency applications because of it's high electron mobility and usable gain at microwave frequencies. An early simulation model for the MESFET device was developed by Walter R. Curtice[2] in 1980 at the RCA Laboratory in Princeton, New Jersey, USA. Since Curtice published his original MEFET model a number of authors

---

[1] M. Brinson and S. Jahn, Qucs: Compact device- circuit macromodel specification; A Curtice level 1 MESFET model, http://qucs.sourceforge.net/docs.html

[2] W.R. Curtice, 1980, A MESFET model for use in the design of GaAs integrated circuits, IEEE Transactions on Microwave Theory and Techniques, MTT-28, pp. 448-456.

have contributed improvements to the basic model, including for example Statz *et. al.* (Raytheon)[3] and TriQuint Semiconductor Inc.[4]. These models form the basis of the Qucs MESFET model described in this report.

5.3 The Qucs MESFET model

Parameters

Name	Symbol	Description	Unit	Default
LEVEL		model selector		1
Vto	V_{to}	gate threshold voltage	V	-1.8
Beta	β	transconductance parameter	A/V^2	3m
Alpha	α	coefficient of Vds in tanh function	1/V	2.25
Gamma	γ	dc drain pull coefficient		0.015
Lambda	λ	channel length modulation parameter	1/V	0.05
B	B	doping profile parameter	1/V	0
Qp	Qp	power law exponent parameter		2.1
Delta	δ	power feedback parameter	1/W	0.1
Vmax	$Vmax$	maximum junction voltage before cap. limiting	V	0.5
Vdelta1	$Vdelta1$	capacitance saturation transition voltage	V	0.3
Vdelta2	$Vdelta2$	capacitance threshold transition voltage	V	0.2
Nsc	Nsc	subthreshold conductance parameter		1
Is	I_S	diode saturation current	A	10f
N	N	diode emission coefficient		1
Vbi	Vbi	built-in gate potential	V	1.0
Bv	Bv	diode breakdown voltage	V	60
XTI	X_{TI}	diode saturation current temperature coefficient		0
TAU	τ	internal time delay from drain to source	s	10p
Rin	Rin	series resistance to Cgs	Ω	1m
Fc	Fc	forward-bias depletion capacitance coefficient		0.5
Area	$Area$	area factor		1

---

[3] H. Statz, P. Newman, I.W. Smith, R.A. Pucel, and H.A. Haus, gaAs FET Device and Circuit Simulation in SPICE, IEEE Transactions on Electron Devices, Vol. 34, pp. 160-169, Feb. 1987.

[4] For example, D.H. Smith, TOM-2: An improved Model for GaAs MESFETs, TriQuint Report, TriQuint Semiconductor, Inc Fe. 27, 1995 (11 pages).

Name	Symbol	Description	Unit	Default
Eg	Eg	bandgap voltage	V	1.11
M	M	grading coefficient		0.5
Cgs	Cgs	zero-bias gate-source capacitance	F	0.2p
Cgd	Cgd	zero-bias gate-drain capacitance	F	1p
Cds	Cds	zero-bias drain-source capacitance	F	1p
Betatc	$Betatc$	Beta temperature coefficient	%/C	0
Alphatc	$Alphatc$	Alpha temperature coefficient	%/C	0
Gammatc	$Gammatc$	Gamma temperature coefficient	%/C	0
Ng	Ng	subthreshold slope gate parameter		2.65
Nd	Nd	subthreshold drain pull parameter		-0.19
ILEVELS	$ILEVELS$	gate-source current equation selector		3
ILEVELD	$ILEVELD$	drain-source current equation selector		3
QLEVELS	$QLEVELS$	gate-source charge equation selector		2
QLEVELD	$QLEVELS$	gate-source charge equation selector		2
QLEVELDS	$QLEVELDS$	drain-source charge equation selector		2
Vtotc	$Vtotc$	Vto temperature coefficient	V/C	0
Rg	Rg	gate series resistance	Ω	5.1
Rd	Rd	drain series resistance	Ω	1.3
Rs	Rs	source series resistance	Ω	1.3
Rgtc	$Rgtc$	gate series resistance temperature coefficient	1/C	0
Rdtc	$Rdtc$	drain series resistance temperature coefficient	1/C	0
Rstc	$Rstc$	source series resistance temperature coefficient	1/C	0
Ibv	Ibv	gate reverse breakdown current	A	1m
Rf	Rf	forward bias slope resistance	Ω	10
R1	$R1$	breakdown slope resistance	Ω	10
Af	Af	Flicker noise exponent		1.0
Kf	Kf	flicker noise coefficient		0.0
Gdsnoi	$Gdnsnoi$	shot noise coefficient		1.0
Tnom	$Tnom$	device parameter measurement temperature	°C	26.85
Temp	$Temp$	device circuit temperature	°C	26.85

Where parameter LEVEL selects a MESFET model listed in Table 5.2.

MESFET gate current equations can be selected by setting parameters ILEVELS and ILEVELD. Table 5.3 lists the available options.

MESFET charge equations can be selected by setting parameters QLEVELS, QLEVELD and QLEVELDS. Table 5.4 lists the available options. Although it is possible to mix the five basic MESFET models with different gate current and

LEVEL	MESFET model type
1	Quadratic Curtice - basic form
2	Quadratic Curtice - basic plus subthreshold properties
3	Statz et. al. (Raytheon) - same as SPICE 3f5
4	TriQuint - TOM 1 model
5	TriQuint - TOM 2 model

Table 5.2: Qucs MESFET model types

ILEVELS - ILEVELD	Gate-source current	Gate-drain current
0	Igs=0	Igd=0
1	Linear no reverse breakdown	Linear no reverse breakdown
2	Linear with reverse breakdown	Linear with reverse breakdown
3	Diode no reverse breakdown	Diode no reverse breakdown
4	Diode with reverse breakdown	Diode with reverse breakdown

Table 5.3: Qucs MESFET gate current model types

charge equation models the common default models are the ones listed in Table 5.5.

QLEVELS		QLEVELD		QLEVELDS	
0	Qgs=0	0	Qgd=0	0	Qds=0
1	Constant cap.	1	Constant cap.	1	Constant cap.
2	Diode	2	Diode	2	Constant cap.+ tran
3	Statz	3	Statz		

Table 5.4: Qucs MESFET charge equation types

Model	LEVEL	ILEVELS	ILEVELD	QLEVELS	QLEVELD	QLEVELDS
Curtice L1	1	0 to 4	0 to 4	0 to 2	0 to 2	0 to 2
Curtice (Adv.)	2	0 to 4	0 to 4	0 to 2	0 to 2	0 to 2
Statz-Raytheon	3	4	4	3	3	2
TOM 1	4	4	4	3	3	2
TOM 2	5	4	4	3	3	2

Table 5.5: Qucs MESET default selection parameters

5.4 The Qucs MESFET simulation model

The large signal equivalent circuit for the Qucs MESFET model is illustrated in Fig. 5.1. The currents flowing in each of the circuit branches are given by the Verilog-A code fragment shown in Fig. 5.1. The Verilog-A HDL code for the entire Qucs MESFET model is available from the Qucs CVS archive[5]. In order to simulate the operation of an MESFET, equations based on the physical operation of the device are required for all the current contribution components in Fig. 5.1. These equations are presented in the remaining sections of this report. Examples are also introduced to demonstrate the simulation performance of each model.

5.5 MESFET gate current equations

- ILEVELS = 0: $Igs = 0$ A

- ILEVELS = 1: if $(V(b1) > Vbi)$

$$Igs = \frac{V(b1) - Vbi}{Rf} \qquad (5.1)$$

  else $\qquad Igs = -Area \cdot Is + GMIN \cdot V(b1)$

- ILEVELS = 2: if $(V(b1) > Vbi)$

$$Igs1 = \frac{V(b1) - Vbi}{Rf} \qquad (5.2)$$

  else $\qquad Igs1 = -Area \cdot Is + GMIN \cdot V(b1)$

  if $V(b1) < -Bv)$

$$Igs2 = \frac{V(b1) - Vbi}{R1} \qquad (5.3)$$

$$Igs = Igs1 + Igs2 \qquad (5.4)$$

---

[5]http://qucs.sourceforge.net/

- ILEVELS = 3: if $(V(b1) > Vbi)$

$$Igs = Is\_T2 \cdot \left\{ limexp\left(\frac{V(b1)}{N \cdot Vt\_T2} \right) - 1.0 \right\} + GMIN \cdot V(b1) \qquad (5.5)$$

else $\qquad Igs = -Is\_T2 + GMIN \cdot V(b1)$

- ILEVELS = 4: if $(V(b1) > -5 \cdot N \cdot Vt\_T2)$

$$Igs1 = Area \cdot Is\_T2 \cdot \left\{ limexp\left(\frac{V(b1)}{N \cdot Vt\_T2} \right) - 1.0 \right\} + GMIN \cdot V(b1)$$
$$(5.6)$$

else $\quad Igs1 = 0$

if $((-Bv < V(b1))$ and $(V(b1) < -5 \cdot N \cdot Vt\_T2))$

$$Igs2 = -Area \cdot Is\_T2 + GMIN \cdot V(b1) \qquad (5.7)$$

else $\qquad Igs2 = 0$

if $(V(b1) == -Bv)$

$$Igs3 = -Ibv \qquad (5.8)$$

else $\qquad Igs3 = 0$

if $(V(b1) < -Bv)$

$$Igs4 = -Area \cdot Is\_T2 \cdot \left\{ limexp\left(\frac{-(Bv + V(b1))}{Vt\_T2} \right) - 1.0 + \frac{Bv}{Vt\_T2} \right\}$$
$$(5.9)$$

else $\quad Igs4 = 0$

$$Igs = Igs1 + Igs2 + Igs3 + Igs4 \qquad (5.10)$$

Where xx_T2 indicates the values of temperature dependent parameters at circuit temperature T2. See later sections of this report for more details. The gate to drain current equations are identical except Igs is replaced by Igd, Igsx by Igdx, and V(b1) by V(b2). More details can be found in the Verilog-A listing given in the Qucs CVS code held at the Qucs Sourceforge site.
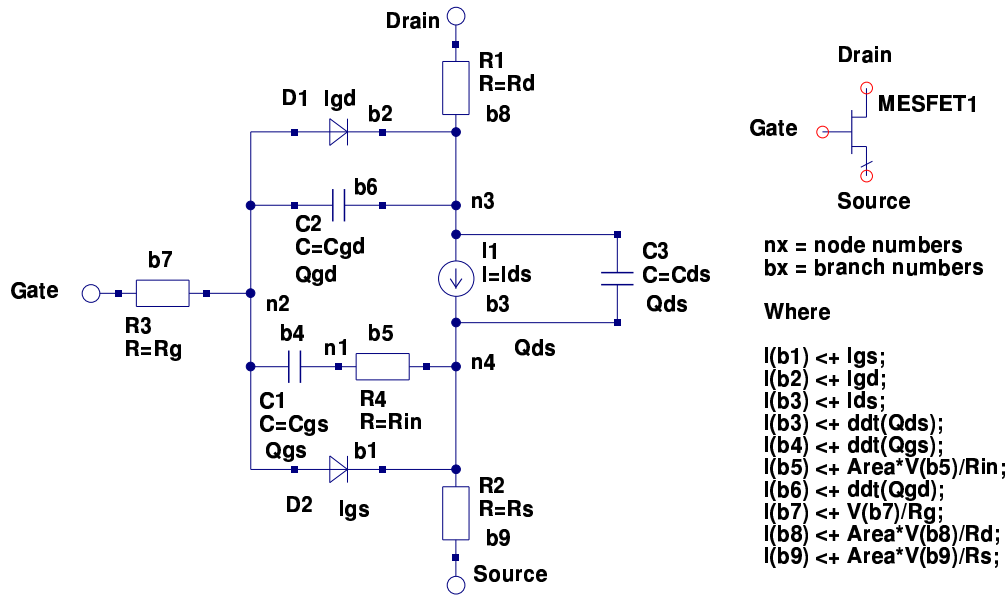
Figure 5.1: Qucs MESFET symbol and large signal equivalent circuit

5.6 MESFET charge equations QLEVELS 0 to 2

- QLEVELS = 0: [NO charge]:

$$Qgs = 0 \tag{5.11}$$

- QLEVELS = 1: [Fixed capacitor charge]

$$Qgs = Area \cdot Cgs \cdot V(b4) \tag{5.12}$$

- QLEVELS = 2: [Diode charge]
  if $(V(b4) < (Fc \cdot Vbi))$

$$Qgs1 = \frac{Cgs\_T2 \cdot Vbi\_T2}{(1-M)} \cdot \left\{ 1 - \left(1 - \frac{V(b4)}{Vbi\_T2} \right)^{1-M} \right\} \tag{5.13}$$

  if $(V(b4) >= (Fc \cdot Vbi))$

$$H1 = \frac{M}{2 \cdot Vbi\_T2} \cdot (V(b4) \cdot V(b4) - (Fc \cdot Fc \cdot Vbi\_T2 \cdot Vbi\_T2)) \tag{5.14}$$

$$Qgs2 = Cgs\_T2 \cdot \left[F1 + \frac{1}{F2} \cdot \{ F3 \cdot (V(b4) - Fc \cdot Vbi\_T2) + H1 \} \right] \tag{5.15}$$

Where,

$$F1 = \frac{Vbi\_T2}{1 - M} \cdot \left\{ 1 - (1 - Fc)^{1-M} \right\}, \tag{5.16}$$

$$F2 = (1 - Fc)^{1+M}, \tag{5.17}$$

and

$$F3 = 1 - Fc \cdot (1 + M). \tag{5.18}$$

Again xx_T2 indicates the values of temperature dependent parameters at circuit temperature T2. See a later section of this report for more details. The gate to drain charge equations (types 0 to 2) are identical except Qgs is replaced by Qgd, Qgsx by Qgdx, and V(b4) by V(b6). More details can be found in the Qucs CVS code held at the Qucs Sourceforge site.

5.7 MESFET charge equations QLEVELDS 0 to 2

- QLEVELDS = 0: [NO charge]:

$$Qds = 0 \tag{5.19}$$

- QLEVELDS = 1: [Fixed capacitor charge]

$$Qds = Area \cdot Cds \cdot V(b3) \tag{5.20}$$

- QLEVELS = 2: [Fixed capacitor plus transit charge]

$$Qds = Area \cdot Cds \cdot V(b3) + Tau \cdot Ids \tag{5.21}$$

5.8 Curtice hyperbolic tangent model: LEVEL = 1

if $(V(b1) - Vto\_T2) > 0$

$$Ids = Beta\_T2 \cdot (V(b1) - Vto\_T2)^2 \cdot \left\{ 1 + Lambda \cdot V(b3) \right\} \cdot tanh(Alpha \cdot V(b3)) \tag{5.22}$$

else $Ids = 0$.
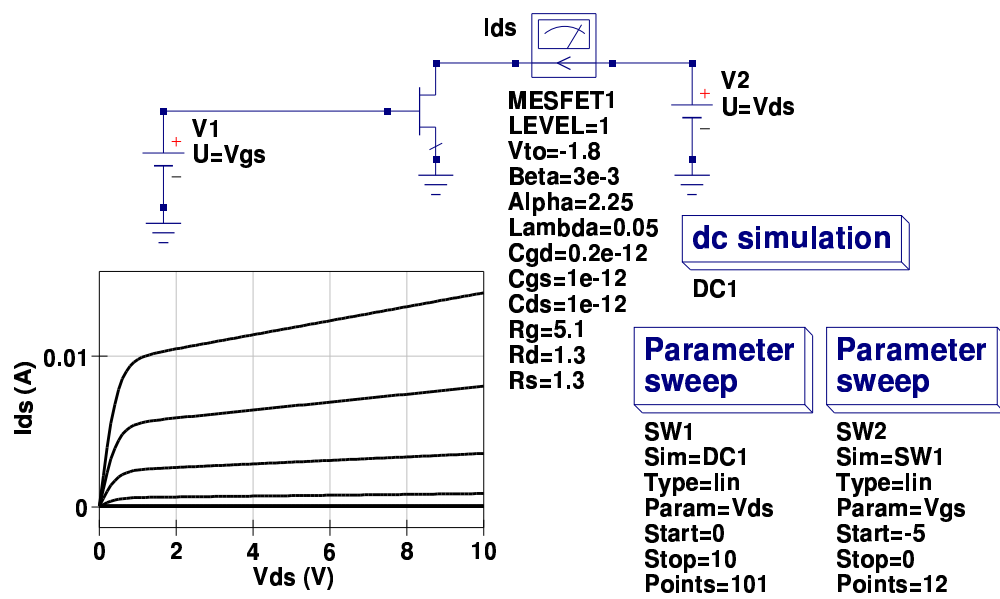
Figure 5.2: Curtice LEVEL 1 DC test circuit and Ids-Vds characteristics
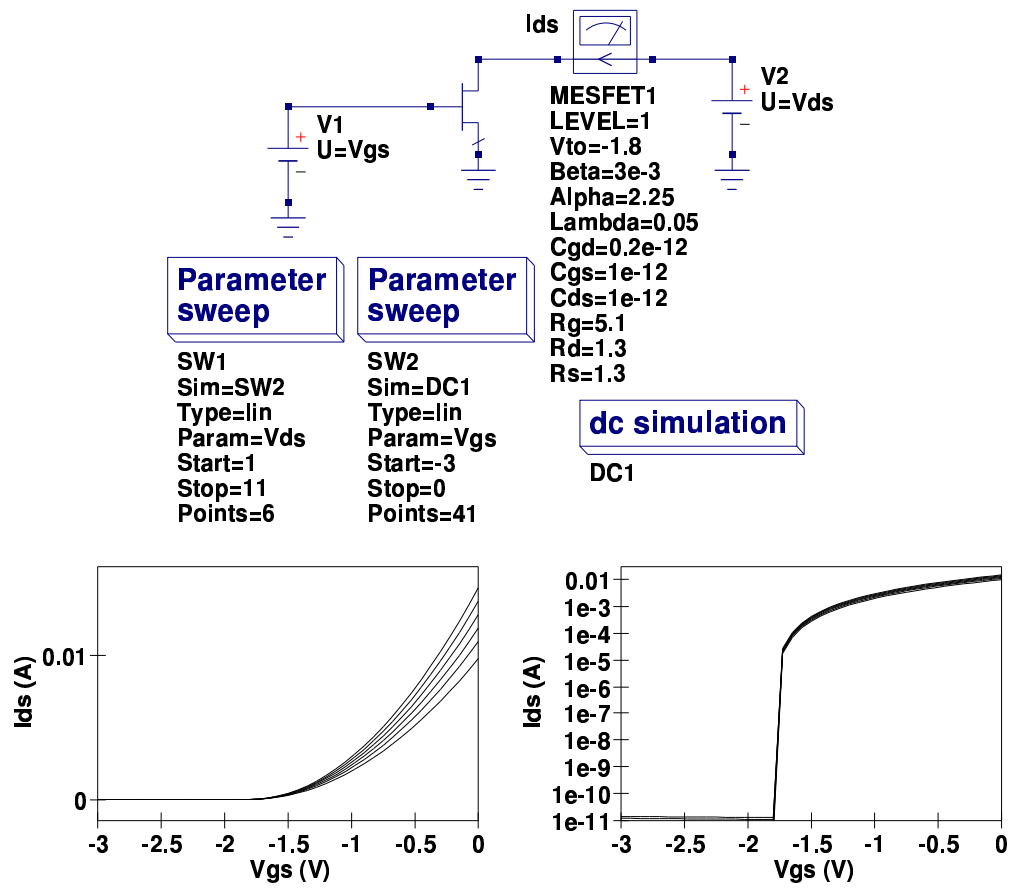
Figure 5.3: Curtice LEVEL 1 DC test circuit and Ids-Vgs characteristics

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Vgs
Start=0.6
Stop=1
Points=21

Parameter sweep

SW2
Sim=SW1
Type=lin
Param=Temp
Start=-100
Stop=100
Points=6

MESFET1
LEVEL=1
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
ILEVELS=3
ILEVELD=3
Rg=5.1
Rd=1.3
Rs=1.3
Temp=Temp

dc simulation

DC1
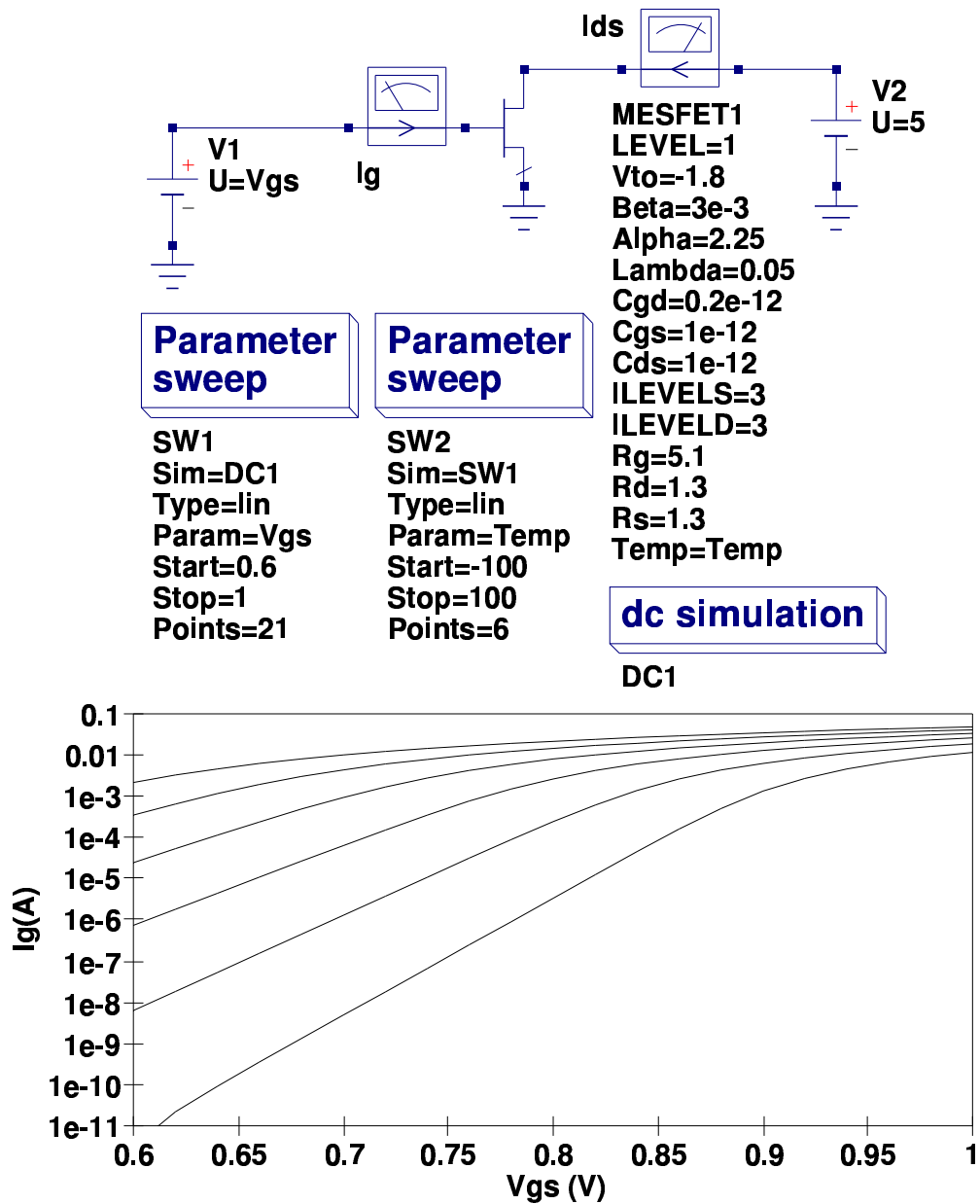
Figure 5.4: Curtice LEVEL 1 DC test circuit and Ig-Vgs characteristics

P1
Num=1
Z=50 Ohm

X1 **V1**
U=0

dc simulation

DC1

Parameter sweep

SW1
Sim=SP1
Type=lin
Param=Vds
Start=5
Stop=20
Points=6

MESFET1
LEVEL=1
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
ILEVELS=3
ILEVELD=3
QLEVELS=2
QLEVELD=2
QLEVELDS=2
Rg=5.1
Rd=1.3
Rs=1.3

X2 **V2**
U=Vds

P2
Num=2
Z=50 Ohm

S parameter simulation

SP1
Type=log
Start=1 kHz
Stop=10 GHz
Points=351
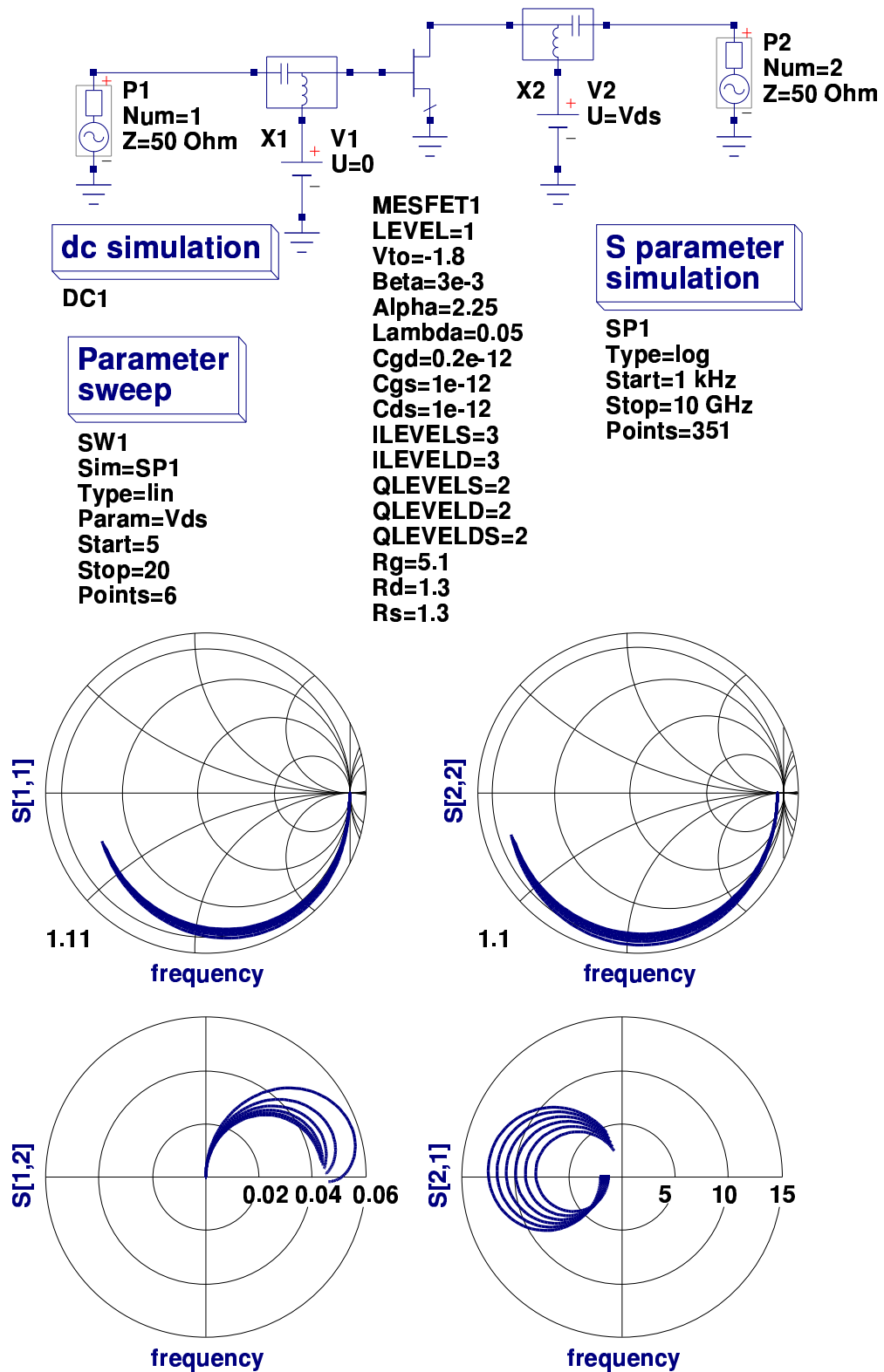
Figure 5.5: Curtice LEVEL 1 S parameter test circuit and characteristics

5.9 Curtice hyperbolic tangent model with subthreshold modification: LEVEL = 2

$$Ids = Beta\_T2 \cdot Vf^2 \cdot \{1 + Lambda \cdot V(b3)\} \cdot tanh(Alpha \cdot V(b3)) \qquad (5.23)$$

Where

$$Vf = \frac{1}{Ah} \cdot ln\left\{1 + exp\left(Ah \cdot (V(b1) - Vto\_T2)\right)\right\} \qquad (5.24)$$

and

$$Ah = \frac{1}{2 \cdot Nsc \cdot Vt\_T2} \qquad (5.25)$$

When $V(b2) > Vto\_T2, Vf \implies V(b2) - Vto\_T2$. Otherwise, Vf approaches zero asymptotically. This modification to the basic Curtice model provides an improved match to channel gradual pinch-off and MESFET subthreshold conduction.
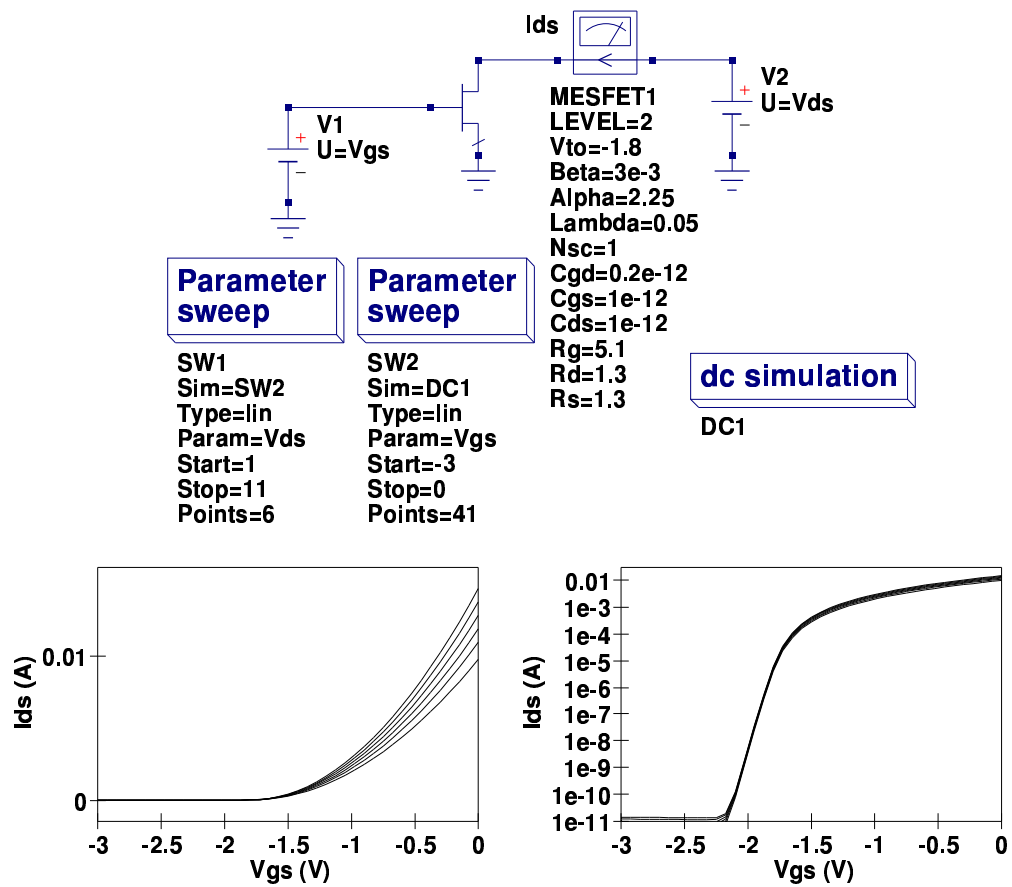
Figure 5.6: Curtice LEVEL 2 DC test circuit and Ids-Vgs characteristics illustrating subthreshold conduction modification

5.10 Statz et. al. (Raytheon) model: LEVEL = 3

if $(V(b1) - Vto\_T2) > 0$

  if $(0 < V(b3))$ and $(V(b3) < \dfrac{3}{Alpha})$

   begin

$$H1 = \frac{1 - \left\{ 1 - \dfrac{Alpha \cdot V(b3)}{3} \right\}^3}{1 + B \cdot (V(b1) - Vto\_T2)} \tag{5.26}$$

$$Ids = Beta\_T2 \cdot \{1 + Lambda \cdot V(b3)\} \cdot (V(b1) - Vto\_T2)^2 \cdot H1 \tag{5.27}$$

   end

  if $(V(b3) > \dfrac{3}{Alpha})$

$$Ids = \frac{Beta\_T2 \cdot \{1 + Lambda \cdot V(b3)\} \cdot (V(b1) - Vto\_T2)^2}{1 + B \cdot (V(b1) - Vto\_T2)} \tag{5.28}$$

else $Ids = 0$.

5.10.1 MESFET charge equations QLEVELS = 3 and QLEVELD = 3

QLEVELS = 3 : Statz et. al. charge equations

$$Vmax = min(Fc \cdot Vbi, Vmax) \tag{5.29}$$

$$Veff1 = 0.5 \cdot \left\{ V(b4) + V(b6) + \sqrt{(V(b6) - V(b4))^2 + Vdelta1^2} \right\} \tag{5.30}$$

$$Vnew = 0.5 \cdot \left\{ Veff1 + Vto\_T2 + \sqrt{(Veff1 - Vto\_T2)^2 + Vdelta2^2} \right\} \tag{5.31}$$

if $(Vnew > Vmax)$

$$Qgs = Cgs\_T2 \cdot \left\{ 2 \cdot Vbi\_T2 \left(1 - \sqrt{1 - \frac{Vmax}{Vbi\_T2}} \right) + \frac{Vnew - Vmax}{\sqrt{1 - \dfrac{Vmax}{Vbi\_T2}}} \right\} \tag{5.32}$$

if $(Vnew <= Vmax)$

$$Qgs = Cgs\_T2 \cdot 2 \cdot Vbi\_T2 \cdot \left\{ 1 - \sqrt{1 - \frac{Vnew}{Vbi\_T2}} \right\} \tag{5.33}$$
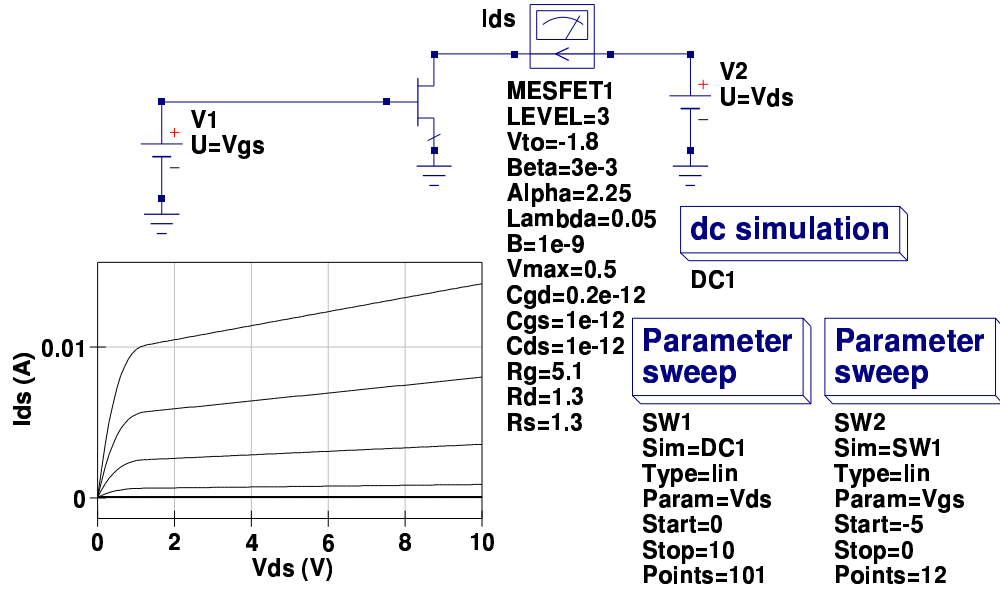
Figure 5.7: Statz *et. al.* LEVEL 3 DC test circuit and Ids-Vds characteristics

QLEVELD = 3 : Statz et. al. charge equations

$$Veff2 = 0.5 \cdot \left\{ V(b4) + V(b6) - \sqrt{(V(b4) - V(b6))^2 + Vdelta1^2} \right\} \qquad (5.34)$$

$$Qds = Cgd\_T2 \cdot Veff2 \qquad (5.35)$$

During simulation gate charge must be partitioned between gate-source and gate-drain branches. The Qucs implementation of the Statz et. al. MESFET model uses the procedure adopted by Divehar [6].

---

[6]D. Divehar, Comments on GaAs FET device and circuit simulation in SPICE,IEEE Transactions on Electronic Devices, Vol. ED-34, pp 2564-2565, Dec. 1987

Ids

V1
U=Vgs

MESFET1
LEVEL=3
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
B=1e-9
Vmax=0.5
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
Rg=5.1
Rd=1.3
Rs=1.3

V2
U=Vds

Parameter sweep

SW1
Sim=SW2
Type=lin
Param=Vds
Start=1
Stop=11
Points=6

Parameter sweep

SW2
Sim=DC1
Type=lin
Param=Vgs
Start=-3
Stop=0
Points=41

dc simulation

DC1
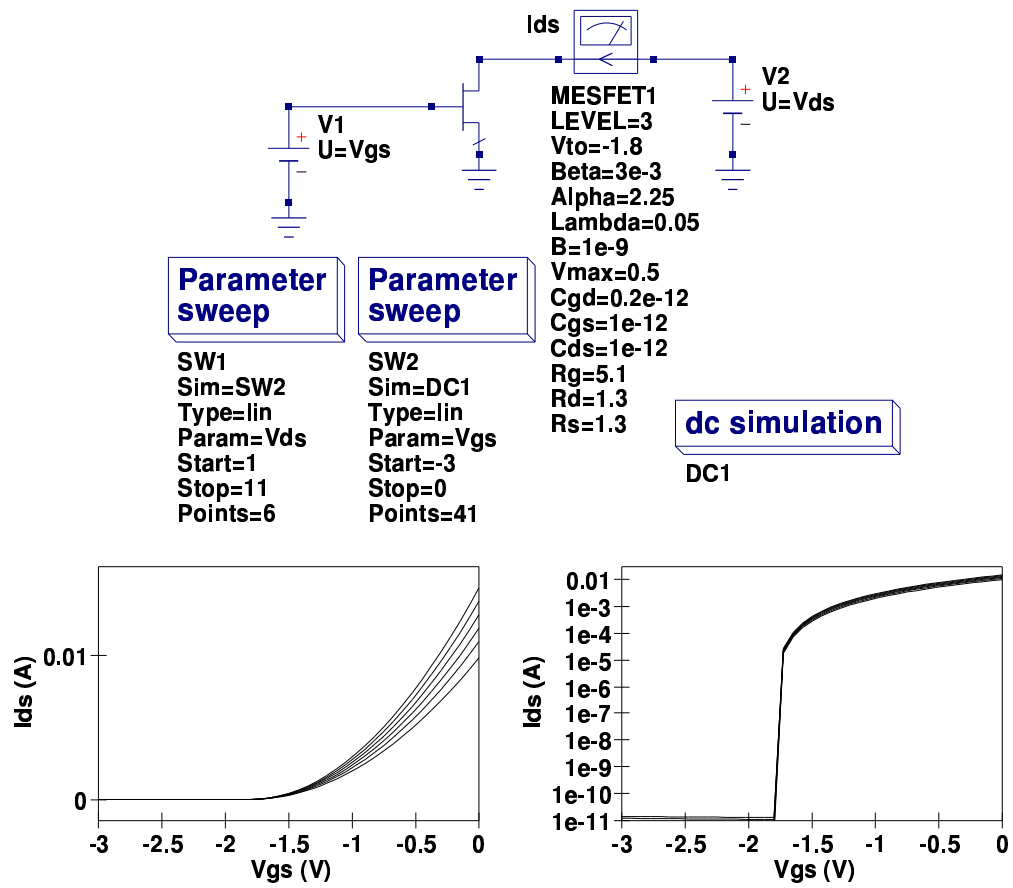
Figure 5.8: Statz *et. al.* LEVEL 3 DC test circuit and Ids-Vgs characteristics
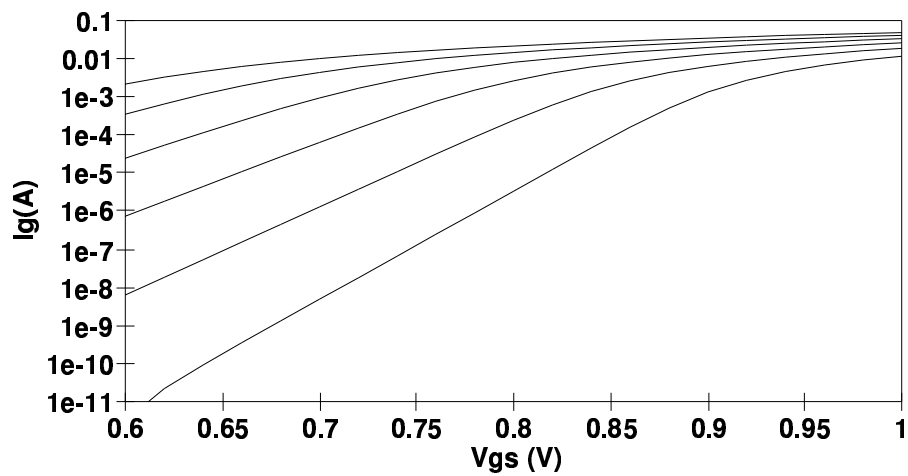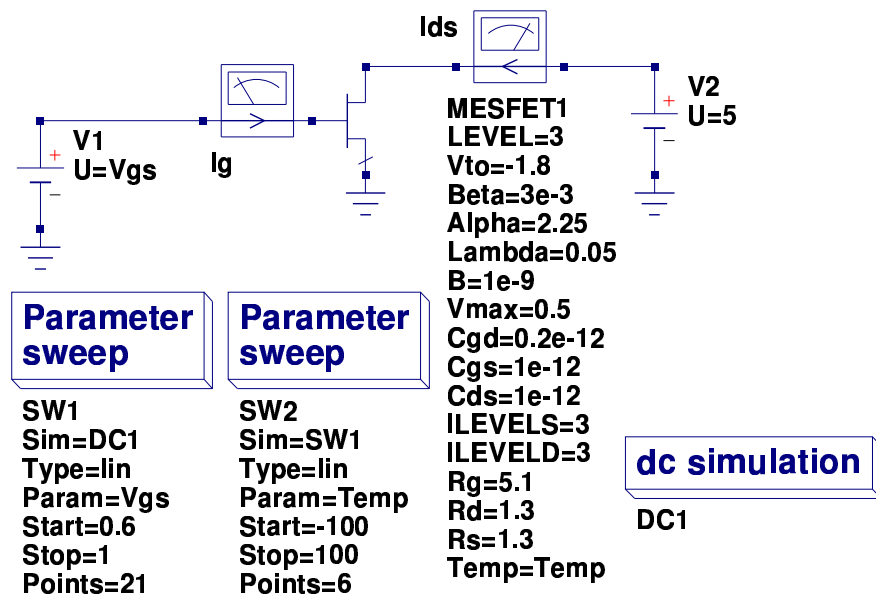
Figure 5.9: Statz *et. al.* LEVEL 3 DC test circuit and Ig-Vgs characteristics

dc simulation

DC1

Parameter sweep

SW1
Sim=SP1
Type=lin
Param=Vds
Start=5
Stop=20
Points=6

MESFET1
LEVEL=3
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
Vdelta1=0.3
Vdelta2=0.2
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
ILEVELS=3
ILEVELD=3
QLEVELS=3
QLEVELD=3
QLEVELDS=2
Rg=5.1
Rd=1.3
Rs=1.3

S parameter simulation

SP1
Type=log
Start=1 kHz
Stop=10 GHz
Points=351
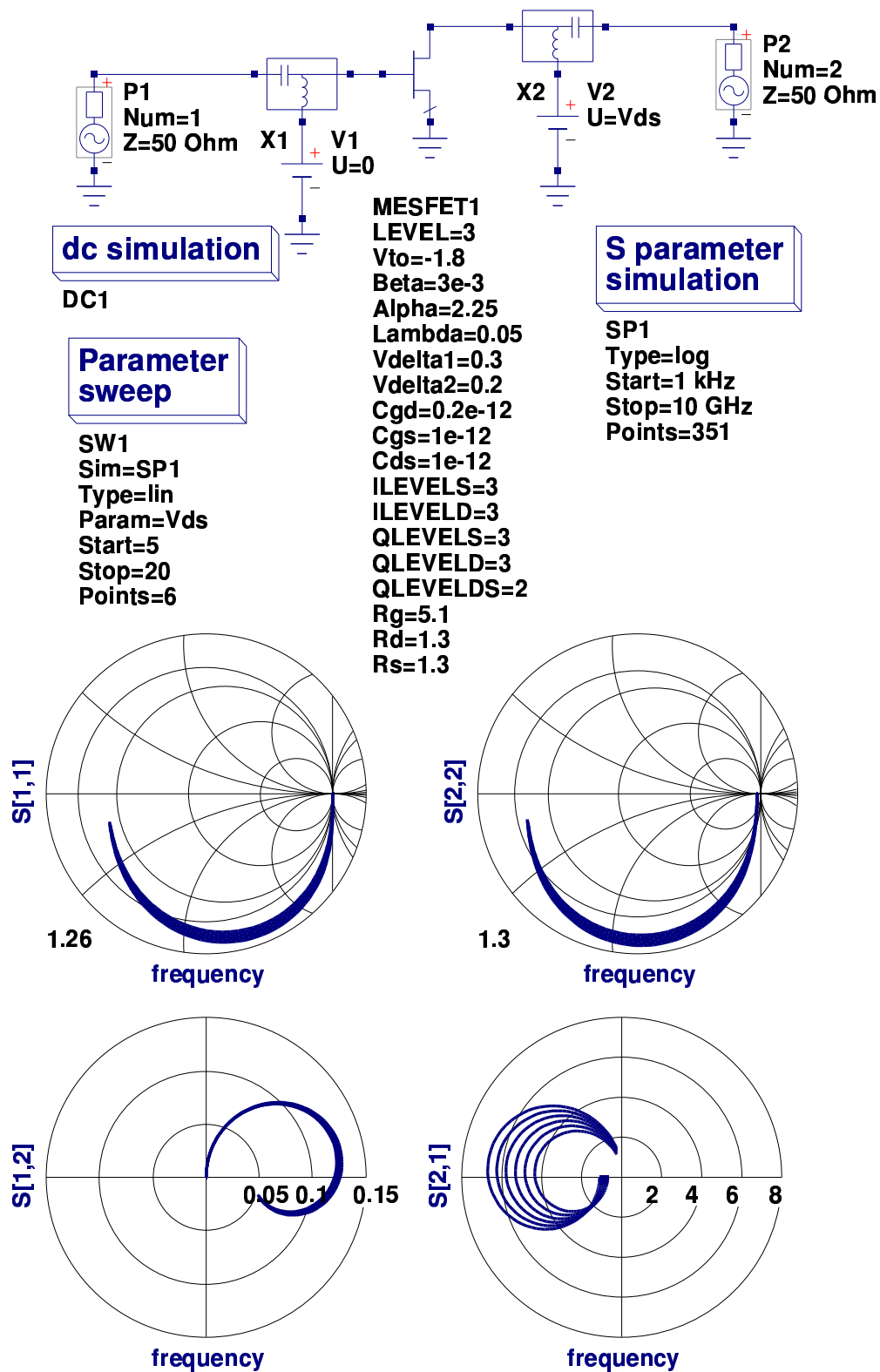
Figure 5.10: Statz *et. al.* LEVEL 3 S parameter test circuit and characteristics

5.11 TriQuint Semiconductor TOM 1 model: LEVEL = 4

if $(V(b1) - Vto\_T2) > 0$

    if $(0 < V(b3))$ and $(V(b3) < \dfrac{3}{Alpha})$

      begin

$$Ids1 = \left\{ Beta\_T2 \cdot (V(b1) - Vto\_T2)^{Qp} \right\} \cdot \left\{ 1 - \left\{ 1 - \frac{Alpha \cdot V(b3)}{3} \right\}^3 \right\}$$

$$(5.36)$$

$$Ids = \frac{Ids1 \cdot \{1 + Lambda \cdot V(b3)\}}{1 + Delta \cdot V(b3) \cdot Ids1} \tag{5.37}$$

      end

    if $(V(b3) > \dfrac{3}{Alpha})$

$$Ids1 = Beta\_T2 \cdot (V(b1) - Vto\_T2)^{Qp} \tag{5.38}$$

$$Ids = \frac{Ids1 \cdot \{1 + Lambda \cdot V(b3)\}}{1 + Delta \cdot (V(b3) \cdot Ids1} \tag{5.39}$$

else $Ids = 0$.

Ids

MESFET1
LEVEL=4
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
B=1e-9
Qp=2.0
Delta=0.1
Vmax=0.5
Vdelta1=0.3
Vdelta2=0.2
Gamma=0.015
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
QLEVELS=3
QLEVELD=3
Rg=5.1
Rd=1.3
Rs=1.3

V1
U=Vgs

V2
U=Vds

dc simulation
DC1

Parameter sweep
SW1
Sim=DC1
Type=lin
Param=Vds
Start=0
Stop=10
Points=101

Parameter sweep
SW2
Sim=SW1
Type=lin
Param=Vgs
Start=-5
Stop=0
Points=12

Ids (A)
0.01
0

0 2 4 6 8 10
Vds (V)

Figure 5.11: TOM1 LEVEL 4 DC test circuit and Ids-Vds characteristics

dc simulation
DC1

Ids

V1
U=Vgs

MESFET1
LEVEL=4
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
B=1e-9
Qp=2.0
Delta=0.1
Vmax=0.5
Vdelta1=0.3
Vdelta2=0.2
Gamma=0.015
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
QLEVELS=3
QLEVELD=3
Rg=5.1
Rd=1.3
Rs=1.3

V2
U=Vds

Parameter sweep
SW1
Sim=SW2
Type=lin
Param=Vds
Start=1
Stop=11
Points=6

Parameter sweep
SW2
Sim=DC1
Type=lin
Param=Vgs
Start=-3
Stop=0
Points=41

Ids (A)
0.01
0

-3 -2.5 -2 -1.5 -1 -0.5 0
Vgs (V)

Ids (A)
0.01
1e-3
1e-4
1e-5
1e-6
1e-7
1e-8
1e-9
1e-10
1e-11

-3 -2.5 -2 -1.5 -1 -0.5 0
Vgs (V)

Figure 5.12: TOM1 LEVEL 4 DC test circuit and Ids-Vgs characteristics

MESFET1
LEVEL=4
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
B=1e-9
Qp=2.0
Delta=0.1
Vmax=0.5
Vdelta1=0.3
Vdelta2=0.2
Gamma=0.015
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
ILEVELS=3
ILEVELD=3
QLEVELS=2
QLEVELD=3
Rg=5.1
Rd=1.3
Rs=1.3
Temp=Temp

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Vgs
Start=0.6
Stop=1
Points=21

Parameter sweep

SW2
Sim=SW1
Type=lin
Param=Temp
Start=-100
Stop=100
Points=6

dc simulation

DC1
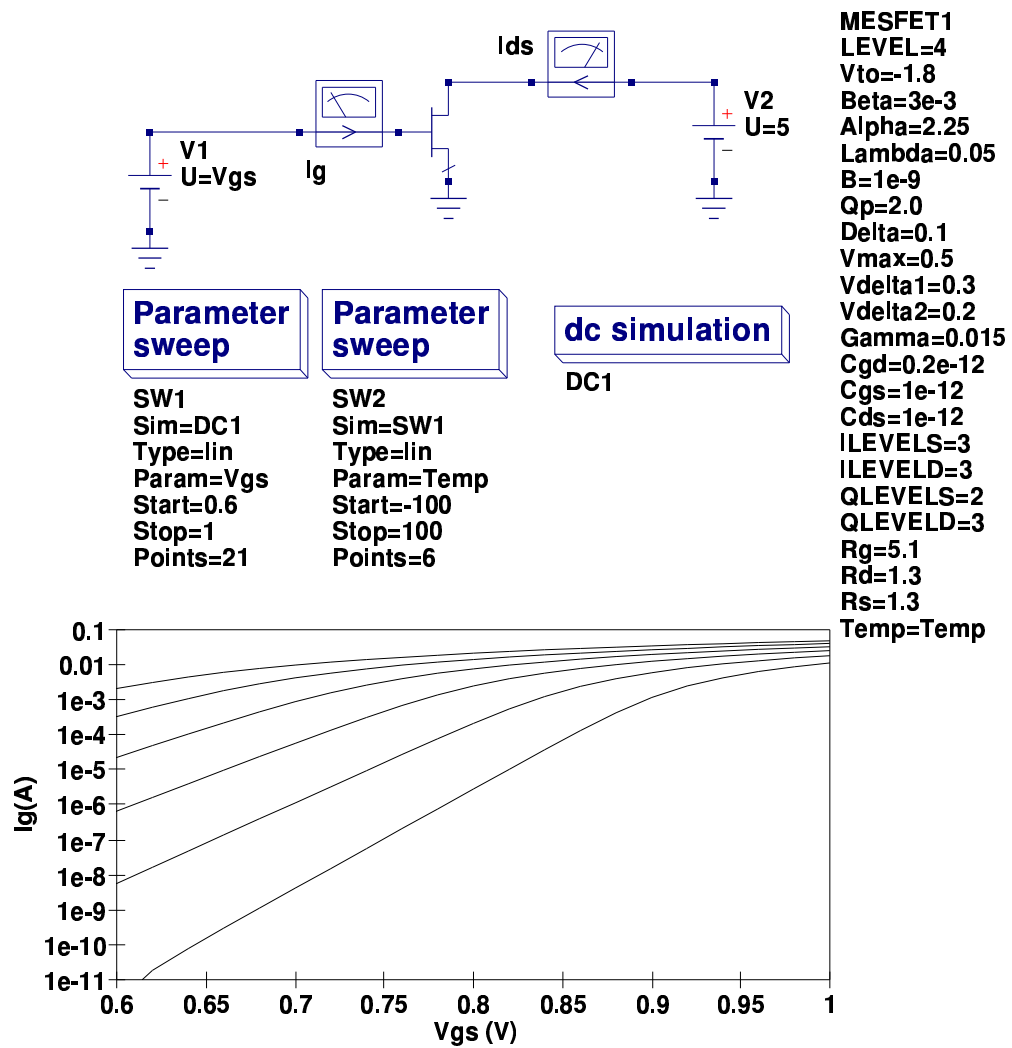
Figure 5.13: TOM1 LEVEL 4 DC test circuit and Ig-Vgs characteristics

P1
Num=1
Z=50 Ohm

X1

V1
U=0

P2
Num=2
Z=50 Ohm

X2   V2
U=Vds

dc simulation

DC1

**S parameter
simulation**

SP1
Type=log
Start=1 kHz
Stop=10 GHz
Points=351

**Parameter
sweep**

SW1
Sim=SP1
Type=lin
Param=Vds
Start=5
Stop=20
Points=6

MESFET1
LEVEL=4
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
Qp=2.1
Delta=0.1
Vmax=0.5
Vdelta1=0.3
Vdelta2=0.2
Gamma=0.015
Nsc=1
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
Ng=2.65
Nd=0
ILEVELS=3
ILEVELD=3
QLEVELS=3
QLEVELD=3
QLEVELDS=2
Rg=5.1
Rd=1.3
Rs=1.3

Figure 5.14: TOM1 LEVEL 4 S parameter test circuit and characteristics

5.12 TriQuint Semiconductor TOM 2 model: LEVEL = 5

if $(V(b1) - Vto\_T2) > 0$
begin

$$Nst = Ng + Nd \cdot V(b3) \tag{5.40}$$

if $(Nst < 1.0)Nst = 1.0$

$$Vst = Nst \cdot Vt\_T2 \tag{5.41}$$

$$Vg = Qp \cdot Vst \cdot ln\left(exp\left\{\frac{V(b1) - Vto\_T2 + Gamma\_T2 \cdot V(b3)}{Qp \cdot Vst}\right\} + 1\right) \tag{5.42}$$

$$Al = Alpha\_T2 \cdot V(b3) \tag{5.43}$$

$$Fd = \frac{Al}{\sqrt{1 + Al \cdot Al}} \tag{5.44}$$

$$Ids1 = Beta\_T2 \cdot Vg^{Qp} \cdot Fd \tag{5.45}$$

$$Ids = Ids1 \cdot \frac{1 + Lambda \cdot V(b3)}{1 + Delta \cdot V(b3) \cdot Ids1} \tag{5.46}$$

end
else $Ids = 0$

Ids

MESFET1
LEVEL=5
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
B=1e-9
Qp=2.0
Delta=0.1
Vmax=0.5
Vdelta1=0.3
Vdelta2=0.2
Gamma=0.015
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
Ng=2.65
Nd=-1.9e-1
QLEVELS=3
QLEVELD=3
Rg=5.1
Rd=1.3
Rs=1.3

V1
U=Vgs

V2
U=Vds

dc simulation
DC1

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Vds
Start=0
Stop=10
Points=101

Parameter sweep

SW2
Sim=SW1
Type=lin
Param=Vgs
Start=-5
Stop=0
Points=12
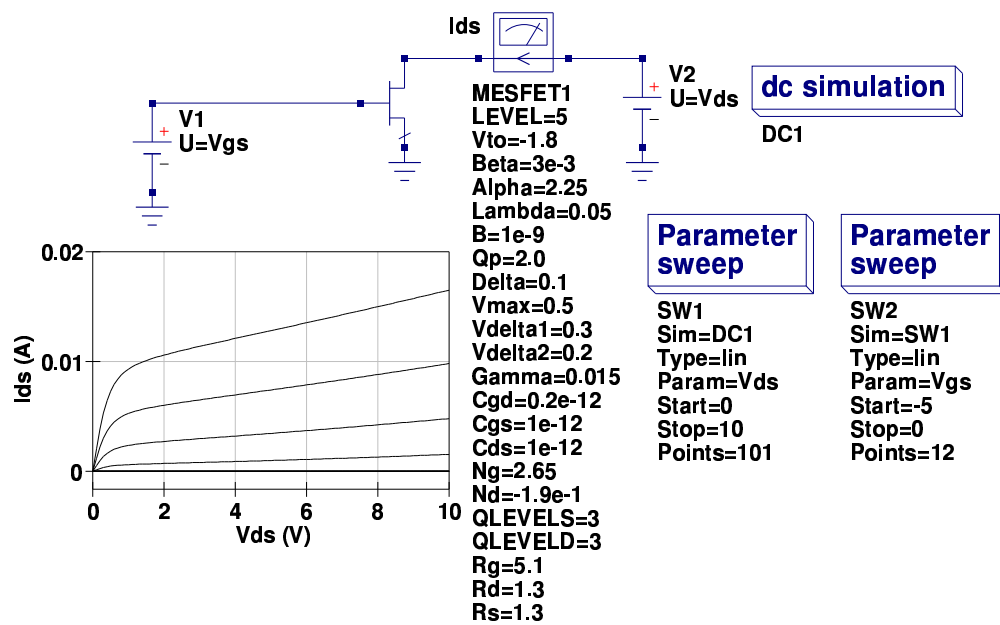
Figure 5.15: TOM2 LEVEL 5 DC test circuit and Ids-Vds characteristics

dc simulation

DC1

V1
U=Vgs

Ids

V2
U=Vds

Parameter sweep

SW1
Sim=SW2
Type=lin
Param=Vds
Start=1
Stop=11
Points=6

Parameter sweep

SW2
Sim=DC1
Type=lin
Param=Vgs
Start=-3
Stop=0
Points=41

MESFET1
LEVEL=5
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
B=1e-9
Qp=2.0
Delta=0.1
Vmax=0.5
Vdelta1=0.3
Vdelta2=0.2
Gamma=0.015
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
Ng=1
Nd=-0.19
QLEVELS=3
QLEVELD=3
Rg=5.1
Rd=1.3
Rs=1.3
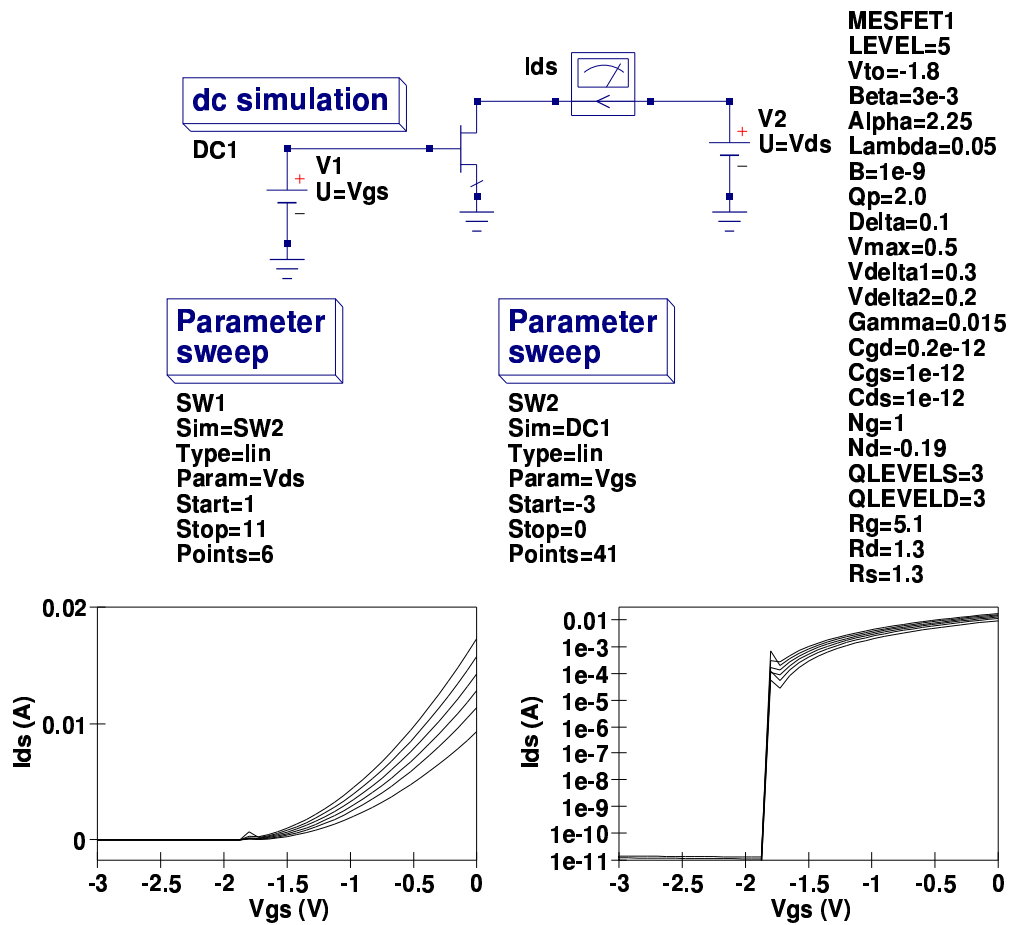
Figure 5.16: TOM2 LEVEL 5 DC test circuit and Ids-Vgs characteristics

116

MESFET1
LEVEL=5
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
B=1e-9
Qp=2.1
Delta=0.1
Vmax=0.5
Vdelta1=0.3
Vdelta2=0.2
Gamma=0.015
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
Ng=2.65
Nd=-0.19
ILEVELS=3
ILEVELD=3
QLEVELS=2
QLEVELD=3
Rg=5.1
Rd=1.3
Rs=1.3
Temp=Temp

SW1
Sim=DC1
Type=lin
Param=Vgs
Start=0.6
Stop=1
Points=21

SW2
Sim=SW1
Type=lin
Param=Temp
Start=-100
Stop=100
Points=6

DC1
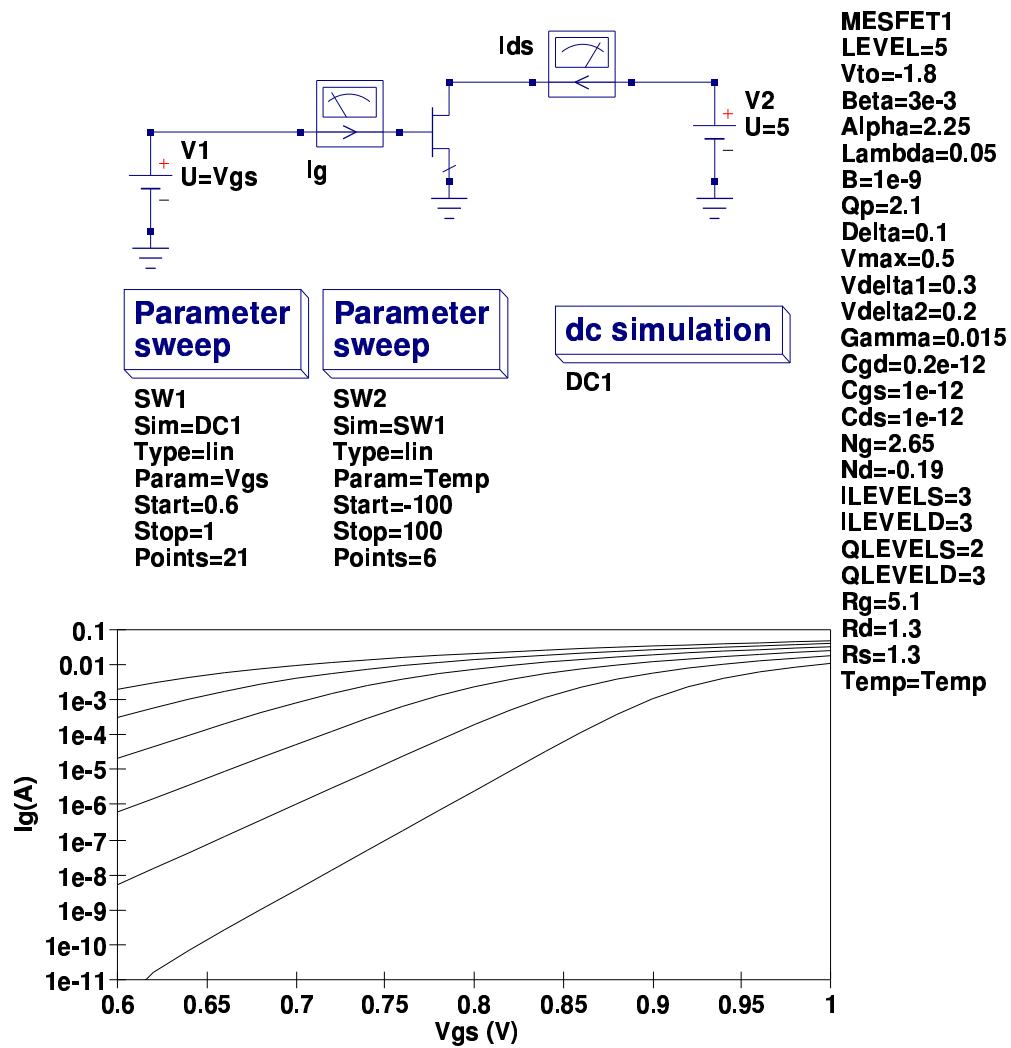
Figure 5.17: TOM2 LEVEL 5 DC test circuit and Ig-Vgs characteristics

117

P1
Num=1
Z=50 Ohm

X1 **V1**
U=0

dc simulation
DC1

Parameter sweep
SW1
Sim=SP1
Type=lin
Param=Vds
Start=5
Stop=20
Points=6

X2 **V2**
U=Vds

P2
Num=2
Z=50 Ohm

S parameter simulation
SP1
Type=log
Start=1 kHz
Stop=10 GHz
Points=351

MESFET1
LEVEL=5
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
Qp=2.0
Delta=0.1
Vmax=0.5
Vdelta1=0.3
Vdelta2=0.2
Gamma=0.015
Nsc=1
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
Ng=2.65
Nd=0
ILEVELS=3
ILEVELD=3
QLEVELS=3
QLEVELD=3
QLEVELDS=2
Rg=5.1
Rd=1.3
Rs=1.3

Figure 5.18: TOM2 LEVEL 5 S parameter test circuit and characteristics

5.13 Temperature scaling relations

```
T1=Tnom+273.15;
T2=Temp+273.15;
Tr=T2/T1;
con1=pow(Tr, 1.5);
Rg_T2=Rg*(1+Rgtc*(T2-T1));
Rd_T2=Rd*(1+Rdtc*(T2-T1));
Rs_T2=Rs*(1+Rstc*(T2-T1));
Beta_T2=Area*Beta*pow(1.01, Betatc*(T2-T1));
Vt_T2=$vt;
Eg_T1=Eg-7.02e-4*T1*T1/(1108+T1);
Eg_T2=Eg-7.02e-4*T2*T2/(1108+T2);
Vbi_T2=(Tr*Vbi)-(2*Vt_T2*ln(con1)) - ( Tr*Eg_T1-Eg_T2);
Is_T2=Area*Is*pow( Tr, (Xti/N))*limexp(-(`P_Q*Eg_T1)*(1-Tr)/(`P_K*T2));
Cgs_T2=Area*Cgs*(1+M*(400e-6*(T2-T1)-(Vbi_T2-Vbi)/Vbi));
Cgd_T2=Area*Cgd*(1+M*(400e-6*(T2-T1)-(Vbi_T2-Vbi)/Vbi));
Vto_T2=Vto+Vtotc*(T2-T1);
Gamma_T2=Gamma*(1+Gammatc*(T2-T1));
Alpha_T2=Alpha*( pow( 1.01, Alphatc*(T2-T1)));
```

5.14 MESFET noise

5.14.1 Main components

- Thermal noise: generated by resistors Rg, Rd and Rs.

- Channel noise: 1. Linear region: essentially thermal noise; 2. Saturation region: diffusion noise.

- Gate noise: Mainly channel noise induced in the gate (via the channel to gate capacitance) The resulting noise is amplified by the MESFET. The capacitive coupling causes the gate noise to have a power spectral density proportional to frequency.

- Flicker noise: Due to random carrier generation-recombination in the lattice imperfections or contaminating impurities. Flicker noise power has a $\dfrac{1}{f^n}$ behavior, with $n \approx 1$.

A typical plot of GaAs MESFET Ids noise current is shown in Fig. 5.19, where
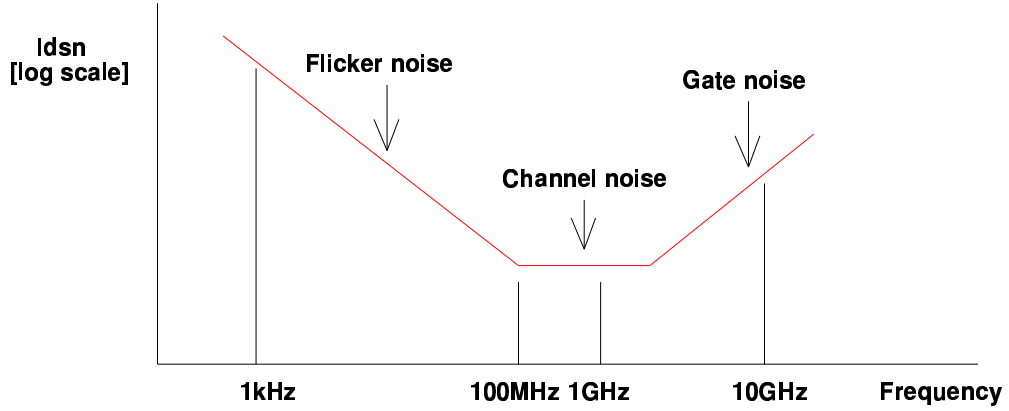
Figure 5.19: Typical GaAS MESFET Ids noise characteristic

the device drain to source noise current is given by

$$Idsn = \text{channel-thermal-noise-current} + \text{flicker-noise-current} \qquad (5.47)$$

To a first approximation:

- Channel-thermal-noise-current[7]$= \sqrt{\dfrac{8 \cdot K \cdot T}{3} \cdot gm \cdot \left\{ \dfrac{1 + \alpha + \alpha^2}{1 + \alpha} \right\} \cdot Gdsnoi}$

  Where $gm = \dfrac{\partial Ids}{\partial Vgs}$,

  and $\alpha = 1 - \dfrac{Vds}{Vgs - Vto}$, when $Vds < \dfrac{3}{Alpha}$ – Linear region of operation

  Or $\alpha = 0$, when $Vds >= \dfrac{3}{Alpha}$ – Saturation region of operation

- flicker-noise-current $= \sqrt{\dfrac{Kf \cdot Ids^{Af}}{f}}$

- Resister thermal noise equations $IRgn = \sqrt{\dfrac{4 \cdot K \cdot T}{Rg}}$, $IRdn = \sqrt{\dfrac{4 \cdot K \cdot T}{Rd}}$,
  and $IRsn = \sqrt{\dfrac{4 \cdot K \cdot T}{Rs}}$

120

Drain

R1
R=Rd

I2
i=IRd_n

b8

Drain

MESFET1

Gate

Source

D1 Igd b2

I5
i=IRg_n

b6

n3

b7

C2
C=Cgd
Qgd

I1
i=Ids

I3
i=Ids_n

C3
C=Cds

nx = node numbers
bx = branch numbers

Gate

n2

b3

Qds

Where

R3
R=Rg

b4 n1 b5

n4 Qds

I(b1) <+ Igs;
I(b2) <+ Igd;
I(b3) <+ Ids;
I(b3) <+ ddt(Qds);
I(b4) <+ ddt(Qgs);
I(b5) <+ Area*V(b5)/Rin;
I(b6) <+ ddt(Qgd);
I(b7) <+ V(b7)/Rg;
I(b8) <+ Area*V(b8)/Rd;
I(b9) <+ Area*V(b9)/Rs;
fourkt = 4.0*'P_K*T2
Ids_n = I(b3) <+ white_noise(thermal_pwr, "thermal")+
        flicker_noise(flicker_pwr, 1.0, "flicker")
IRg_n = I(b7) <+ white_noise(Area*fourkt/Rg_T2, "thermal")
IRd_n = I(b8) <+ white_noise(Area*fourkt/Rd_T2, "thermal")
IRs_n = I(b9) <+ white_noise(Area*fourkt/RS_T2, "thermal")

Values for thermal_pwr and flicker_pwr are given in the text.

C1
C=Cgs
Qgs

R4
R=Rin

D2 Igs b1

R2
R=Rs

I4
i=IRs_n

b9

Source

Figure 5.20: Typical GaAS MESFET equivalent circuit illustrating noise current components

5.14.2 MESFET equivalent circuit with noise current components

5.14.3 Curtice hyperbolic tangent model: LEVEL = 1 or 2: Noise equations

1. Verilog-A equations

```
fourkt=4.0*'P_K*T2;
gm=2*Beta_T2*(V(b1)-Vto_T2)*(1+Lambda*V(b3))*tanh(Alpha_T2*V(b3));
    if ( V(b3) < 3/Alpha ) begin  An=1-V(b3)/(V(b1)-Vto_T2);
     thermal_pwr= (8*'P_K*T2*gm/3)*((1+An+An*An)/(1+An))*Gdsnoi;
    end
    else
     thermal_pwr=(8*'P_K*T2*gm/3)*Gdsnoi;
     I(b3)<+white_noise(thermal_pwr,"thermal"); flicker_pwr = Kf*pow(Ids,Af);
     I(b3)<+flicker_noise(flicker_pwr,1.0,"flicker");
    end
I(b7) <+ white_noise(Area*fourkt/Rg_T2, "thermal");
```

---

[7]Yannis Tsividis, Operation and modeling of the MOS transistor, McGraw-HIll 1987, p340

```
I(b8) <+ white_noise(Area*fourkt/Rd_T2, "thermal");
I(b9) <+ white_noise(Area*fourkt/Rs_T2, "thermal");
```

2. Typical noise simulation results
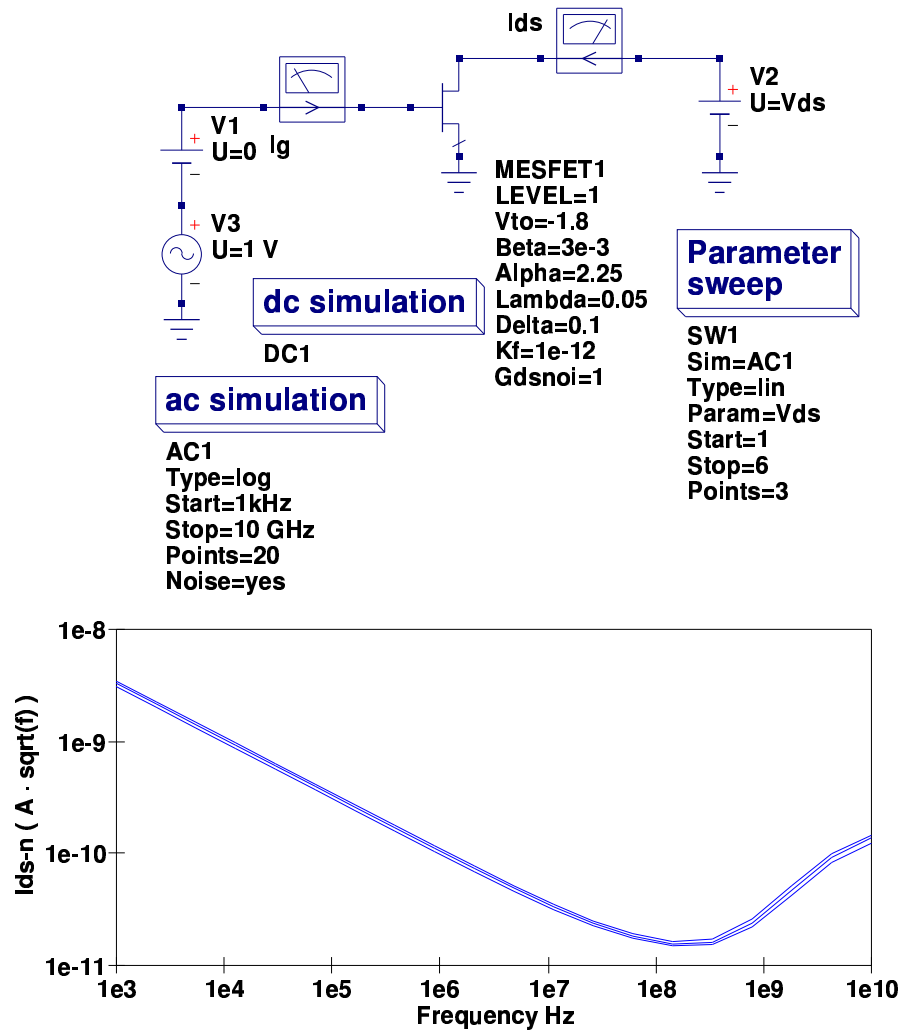
Figure 5.21: Typical LEVEL 1 (or 2) GaAS MESFET Ids noise characteristic

5.14.4 Statz et. al. (Raytheon) model: LEVEL = 3: Noise equations

1. Verilog-A equations

```
if ( V(b3) < 3/Alpha )begin
H1=(1-(1-(Alpha*V(b3))/3))/(1+B*(V(b1)-Vto_T2));
gm=2*Beta_T2*(V(b1)-Vto_T2)*(1+Lambda*V(b3))*H1+(Beta_T2*
    (1+Lambda*V(b3))*pow((V(b1)-Vto_T2),2))*B*H1/(1+B*(V(b1)-Vto_T2));
An=1-V(b3)/(V(b1)-Vto_T2);
thermal_pwr= (8*`P_K*T2*gm/3)*((1+An+An*An)/(1+An))*Gdsnoi;
end
else begin
gm=2*Beta_T2*(V(b1)-Vto_T2)*(1+Lambda*V(b3))/(1+B*(V(b1)-Vto_T2))+
(Beta_T2*(1+Lambda*V(b3))*pow((V(b1)-Vto_T2),2))
*B/pow( (1+B*(V(b1)-Vto_T2)),2);
thermal_pwr=(8*`P_K*T2*gm/3)*Gdsnoi;
end
I(b3) <+ white_noise(thermal_pwr, "thermal");
flicker_pwr = Kf*pow(Ids,Af);
I(b3) <+ flicker_noise(flicker_pwr,1.0, "flicker");
I(b7) <+ white_noise(Area*fourkt/Rg_T2, "thermal");
I(b8) <+ white_noise(Area*fourkt/Rd_T2, "thermal");
I(b9) <+ white_noise(Area*fourkt/Rs_T2, "thermal");
```

2. Typical noise simulation results

5.14.5 TriQuint Semiconductor TOM 1 model: LEVEL = 4: Noise equations

1. Verilog-A equations

```
if ( V(b3) < 3/Alpha )begin
Ids1=(Beta_T2*pow( (V(b1)-Vto_T2), Qp) )*(1-pow( (1-Alpha*V(b3)/3), 3));
gm1=Qp*Beta_T2*pow( V(b1)-Vto_T2, Qp-1)*(1-(1-pow(Alpha*V(b3)/3, 3)));
gm=(gm1*(1+Lambda*V(b3))/(1+Delta*V(b1)*Ids1))*(1+(Delta*V(b3)*Ids1)/
    (1+Delta*V(b3)*Ids1));
        An=1-V(b3)/(V(b1)-Vto_T2);
thermal_pwr= (8*`P_K*T2*gm/3)*((1+An+An*An)/(1+An))*Gdsnoi;
end
else begin
Ids1=(Beta_T2*pow( (V(b1)-Vto_T2), Qp) );
gm1=Qp*Beta_T2*pow( V(b1)-Vto_T2, Qp-1);
gm=(gm1*(1+Lambda*V(b3))/(1+Delta*V(b1)*Ids1))*(1+(Delta*V(b3)*Ids1)/
    (1+Delta*V(b3)*Ids1));
        thermal_pwr=(8*`P_K*T2*gm/3)*Gdsnoi;
```
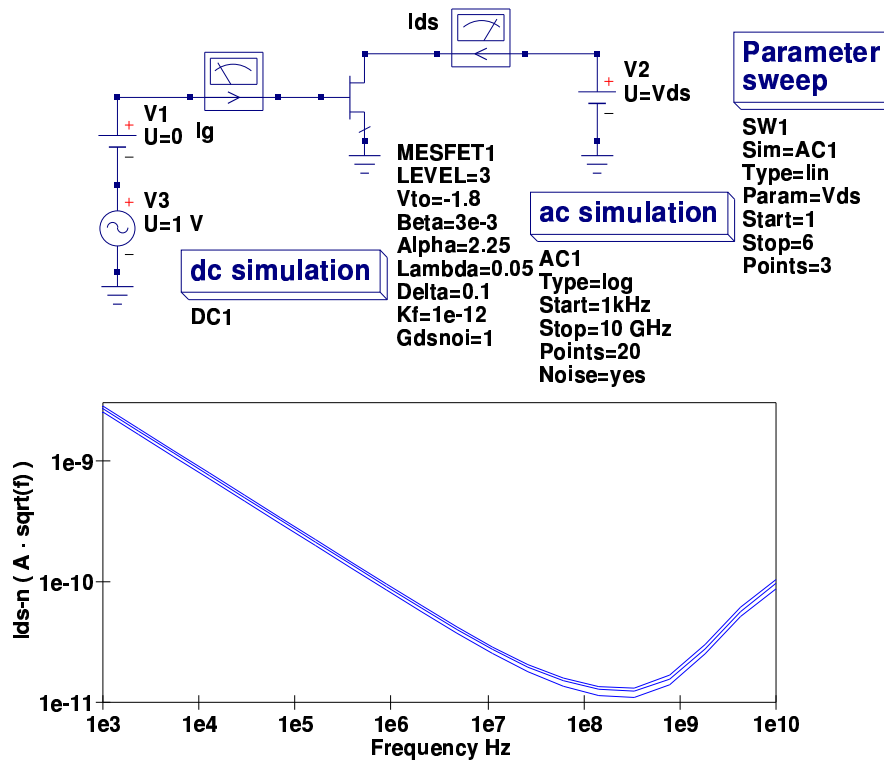
Figure 5.22: Typical LEVEL 3 GaAS MESFET Ids noise characteristic

```
end
I(b3) <+ white_noise(thermal_pwr, "thermal");
flicker_pwr = Kf*pow(Ids,Af);
I(b3) <+ flicker_noise(flicker_pwr,1.0, "flicker");
I(b7) <+ white_noise(Area*fourkt/Rg_T2, "thermal");
I(b8) <+ white_noise(Area*fourkt/Rd_T2, "thermal");
I(b9) <+ white_noise(Area*fourkt/Rs_T2, "thermal");
```

2. Typical noise simulation results

5.14.6 TriQuint Semiconductor TOM 2 model: LEVEL = 5

1. Verilog-A equations

```
if ( V(b3) < 3/Alpha )begin
Nst=Ng+Nd*V(b3);
if ( Nst < 1.0) Nst=1.0;
 Vst=Nst*Vt_T2;
```
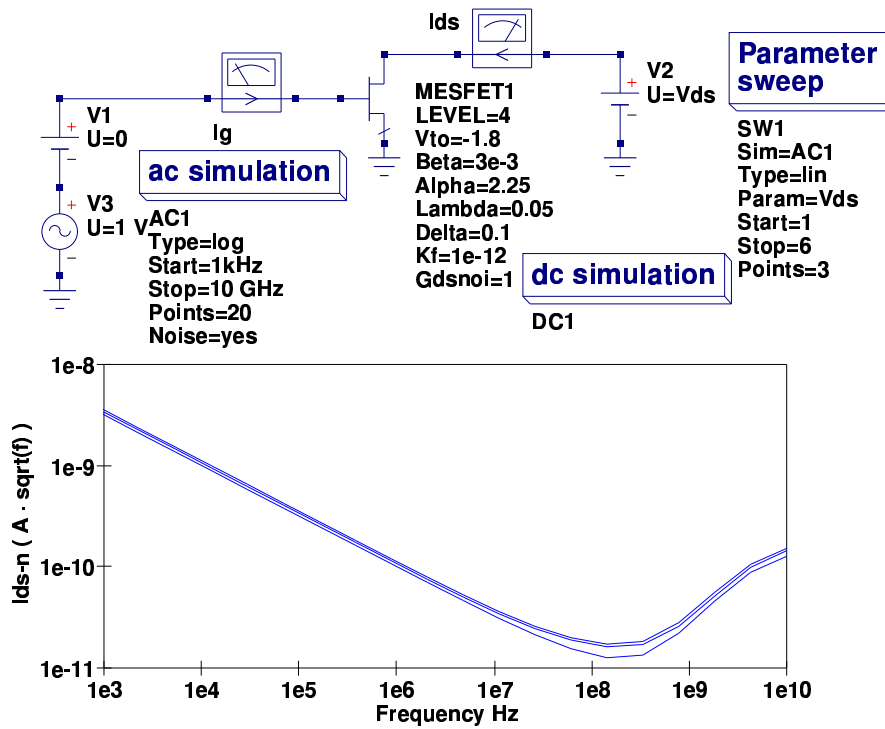
Figure 5.23: Typical LEVEL 4 GaAS MESFET Ids noise characteristic

```
Vg=Qp*Vst*ln( exp( (V(b1)-Vto_T2+Gamma_T2*V(b3)) / (Qp*Vst) ) + 1);
Al= Alpha_T2*V(b3); Fd=Al/sqrt( 1.0+(Al*Al) );
Ids1=Beta_T2*pow( Vg, Qp)*Fd;
gm1=(Ids1/Vg)*Qp/(exp(-((V(b1)-Vto_T2+Delta*V(b3))/(Qp*Vst)))+1);
gm=gm1/pow( (1+Delta*V(b3)*Ids1),2);
         An=1-V(b3)/(V(b1)-Vto_T2);
   thermal_pwr= (8*'P_K*T2*gm/3)*((1+An+An*An)/(1+An))*Gdsnoi;
end
else begin
 Nst=Ng+Nd*V(b3); if ( Nst < 1.0) Nst=1.0;
 Vst=Nst*Vt_T2;
 Vg=Qp*Vst*ln( exp( (V(b1)-Vto_T2+Gamma_T2*V(b3)) / (Qp*Vst) ) + 1);
 Al= Alpha_T2*V(b3); Fd=Al/sqrt( 1.0+(Al*Al) );
 Ids1=Beta_T2*pow( Vg, Qp)*Fd;
 gm1=(Ids1/Vg)*Qp/(exp(-((V(b1)-Vto_T2+Delta*V(b3))/(Qp*Vst)))+1);
 gm=gm1/pow( (1+Delta*V(b3)*Ids1),2);
 thermal_pwr=(8*'P_K*T2*gm/3)*Gdsnoi;
end
```

```
        I(b3) <+ white_noise(thermal_pwr, "thermal");
        flicker_pwr = Kf*pow(Ids,Af);
        I(b3) <+ flicker_noise(flicker_pwr,1.0, "flicker");
        I(b7) <+ white_noise(Area*fourkt/Rg_T2, "thermal");
        I(b8) <+ white_noise(Area*fourkt/Rd_T2, "thermal");
        I(b9) <+ white_noise(Area*fourkt/Rs_T2, "thermal");
```

2. Typical noise simulation results
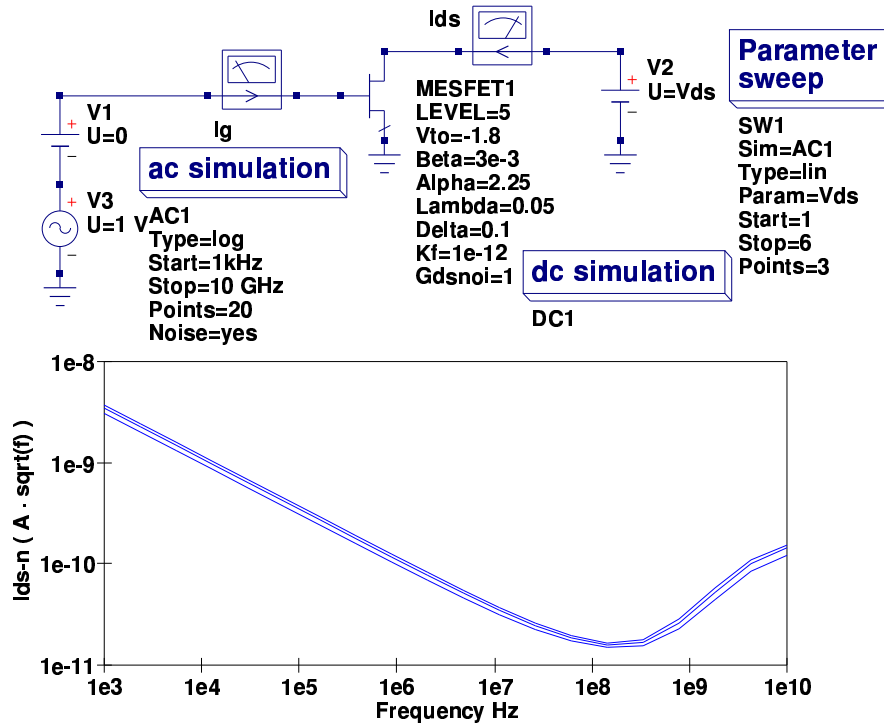
Figure 5.24: Typical LEVEL 5 GaAS MESFET Ids noise characteristic

5.15 Adding external passive components to the MESFET models

The Curtice model outlined in the first Qucs report on MESFETs included lead
inductance in each of the device signal paths. These inductances were not included
in the Verilog-A models described in this report, mainly to simplify the model code.
If required they can be added as external components. The test circuit shown in
Fig. 5.25 indicates how this can be done and illustrates the effect such components
have on the Curtice S parameter characteristics.

P1
Num=1
Z=50 Ohm

X1 V1
     U=0

dc simulation

DC1

Lg
L=10 nH

MESFET1
LEVEL=3
Vto=-1.8
Beta=3e-3
Alpha=2.25
Lambda=0.05
Vdelta1=0.3
Vdelta2=0.2
Cgd=0.2e-12
Cgs=1e-12
Cds=1e-12
ILEVELS=3
ILEVELD=3
QLEVELS=3
QLEVELD=3
QLEVELDS=2
Rg=5.1
Rd=1.3
Rs=1.3

Ls
L=10 nH

Ld
L=Ld

X2 V2
     U=10

P2
Num=2
Z=50 Ohm

S parameter simulation

SP1
Type=log
Start=1 kHz
Stop=10 GHz
Points=351

Parameter sweep

SW1
Sim=SP1
Type=lin
Param=Ld
Start=0
Stop=10n
Points=11

Figure 5.25: S parameter simulated characteristics for test circuit shown in Fig. 5.5 that has external inductance added

5.16 End note

MESFETs are important high frequency devices which have been missing from the range of active component models supplied with Qucs. While developing the models described in this report I have attempted to make them as flexible as possible so as to allow users the opportunity to select which model, or indeed the make-up of the components of a model, they would like to try for a specific simulation. The work described in this report is very much work in progress, mainly because there are a number of other published MESFET models that have not been included. My intention has simply been to provide a number of practical models which were not previously available to Qucs users. Also knowing that many Qucs users have an interest in high frequency circuit design and simulation, the work would be of direct relevance to making Qucs more "universal". The procedures employed for model development are another example of the work being undertaken by the Qucs team in response to Qucs being adopted by the wider modelling community as part of the Verilog-A compact device standardization project. Overall the simulation results from the models described here show a high degree of consistency from DC to the high frequency S parameter domain. The noise results are particularly interesting as they are based on mix of available theories and extensions introduced especially for Qucs. Some readers will probably have spotted one area where there appears to be differences in the simulation results from the different models; look at the S[1,2] and S[2,1] characterstics for each model. Here there are noticeable difference which are possibly due to the lack of symmetry in some of the model charge equations? MESFET modelling is a complex subject, suggesting that there are likely to be errors /bugs in the models. If you find an error/bug please inform the Qucs development team so that we can correct problems as they are found. In the future, particularly if the response to this group of models is positive, I will attempt to add more MESFET models to Qucs. Once again a special thanks to Stefan Jahn for all his help and encouragement over the period that I have been developing the Qucs MESFET models and writing the report which outlines their physical and mathematical fundamentals.

6 Verilog-A implementation of the EKV v2.6 long and short channel MOSFET models

6.1 Introduction

This report presents the background to the Qucs implementation of the EKV 2.6 long and short channel MOSFET models. During 2007 the Qucs development team employed the EKV v2.6 MOSFET model as a test case while developing the Qucs non-linear equation defined devices (EDD)[1]. More recently complete implementations of the long and short channel EKV v2.6 models have been developed using the Qucs Verilog-A compact device modelling route. This work forms part of the Verilog-A compact device modelling standardisation initiative[2]. The EKV v2.6 MOSFET model is a physics based model which has been placed in the public domain by its developers. It is ideal for analogue circuit simulation of submicron CMOS circuits. Since the models introduction and development between 1997 and 1999 it has been widely used in industry and by academic circuit design groups. Today the EKV v2.6 model is available with most of the major commercial simulators and a growing number of GPL simulators. The Verilog-A code for the Qucs ADMS[3] compiled version of the EKV v2.6 model is given in an appendix to this report.

6.2 Effects modelled

The EKV v2.6 MOSFET model includes the following effects:

---

[1]An example EDD macromodel of the short channel EKV 2.6 model can be found at `http://qucs.sourceforge.net/`.

[2]Stefan Jahn, Mike Brinson, Michael Margraf, Hélène Parruitte, Bertrand Ardouin, Paolo Nenzi and Laurent Lemaitre, GNU Simulators Supporting Verilog-A Compact Model Standardization, MOS-AK Meeting, Premstaetten, 2007, `http://www.mos-ak.org/premstaetten/papers/MOS-AK_QUCS_ngspice_ADMS.pdf`

[3]Lemaitre L. and GU B., ADMS - a fully customizable Verilog-AMS compiler approach, MOS-AK Meeting, Montreux. Available from `http://www.mos-ak.org/montreux/posters/17_Lemaitre_MOS-AK06.pdf`

- Basic geometrical and process related features dependent on oxide thickness, junction depth, effective channel length and width

- Effects of doping profile

- Modelling of weak, moderate and strong inversion behaviour

- Modelling of mobility effects due to vertical and lateral fields, velocity saturation

- Short channel effects including channel-length modulation, source and drain charge-sharing and reverse channel effect

- Modelling of substrate current due to impact ionization

- Thermal and flicker noise

- First order non-quasistatic model for the transconductances

- Short-distance geometry and bias dependent device matching

The Qucs implementation of the short channel EKV v2.6 model includes nearly all the features listed above[4]. A simpler long channel version of the model is also available for those simulations that do not require short channel effects. Both nMOS and pMOS devices have been implemented. No attempt is made in this report to describe the physics of the EKV v2.6 model. Readers who are interested in learning more about the background to the model, its physics and function should consult the following references:

- Matthias Bucher *et. al.*, The EPFL-EKV MOSFET Model Equations for Simulation, Electronics Laboratories, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, Model Version 2.6, Revision II, July 1998.

- Władysław Grabiński *et. al.* Advanced compact modelling of the deep sub-micron technologies, Journal of Telecommunications and Information Technology, 3-4/2000, pp. 31-42.

- Matthias Bucher *et. al.*, A MOS transistor model for mixed analog-digital circuit design and simulation, pp. 49-96, Design of systems on a chip - Devices and Components, KLUWER Academic Publishers, 2004.

---

[4]This first release of the Qucs implementation of the EKV v2.6 MOSFET model does not include the first-order non-quasistatic model for transconductances.

- Trond Ytterdal *et. al.*, Chapter 7: The EKV model, pp. 209-220, Device Modeling for Analog and RF CMOS Circuit Design, John Wiley & Sons, Ltd, 2003.

- Patrick Mawet, Low-power circuits and beyond: a designer's perspective on the EKV model and its usage, MOS-AK meeting, Montreux, 2006, `http://www.mos-ak.org/montreux/posters/09_Mawet_MOS-AK06.pdf`

- Christian C. Enz and Eric A. Vittoz, Charge-based MOS transistor Modeling - The EKV model for low-power and RF IC design, John Wiley & Sons, Ltd, 2006.

6.3 The Qucs long channel EKV v2.6 model

A basic DC model for the long channel nMOS EKV v2.6 model is given at the EKV Compact MOSFET model website[5]. Unfortunately, this model is only of limited practical use due to its restricted modelling features[6]. It does however, provide a very good introduction to compact device modelling using the Verilog-A hardware description language. Readers who are unfamiliar with the Verilog-A hardware description language should consult the following references:

- Accellera, Verilog-AMS Language Reference Manual, Version 2.2, 2004, Available from `http://www.accellera.org`.

- Kenneth S. Kundert and Olaf Zinke, The Designer's Guide to Verilog-AMS, Kluwer Academic Publishers, 2004.

- Dan Fitzpatrick and Ira Miller, Analog Behavioral Modeling with the Verilog-A Language, Kluwer Academic Publishers, 1998.

- Coram G. J., How to (and how not to) write a compact model in Verilog-A, 2004, IEEE International Behavioural modeling and Simulation Conference (BMAS2004), pp. 97-106.

The equivalent circuit of the Qucs EKV long channel n type MOSFET model is shown in Fig. 6.1. In this model the inner section, enclosed with the red dotted box, represents the fundamental intrinsic EKV v2.6 elements. The remaining components model extrinsic elements which represent the physical components connecting the intrinsic MOSFET model to its external signal pins. In the Qucs implementation of the EKV v2.6 long channel MOSFET model the drain to source

---

[5]See `http://legwww.epfl.ch/ekv/verilog-a/` for the Verilog-A code.
[6]No dynamic, noise or temperature effects.
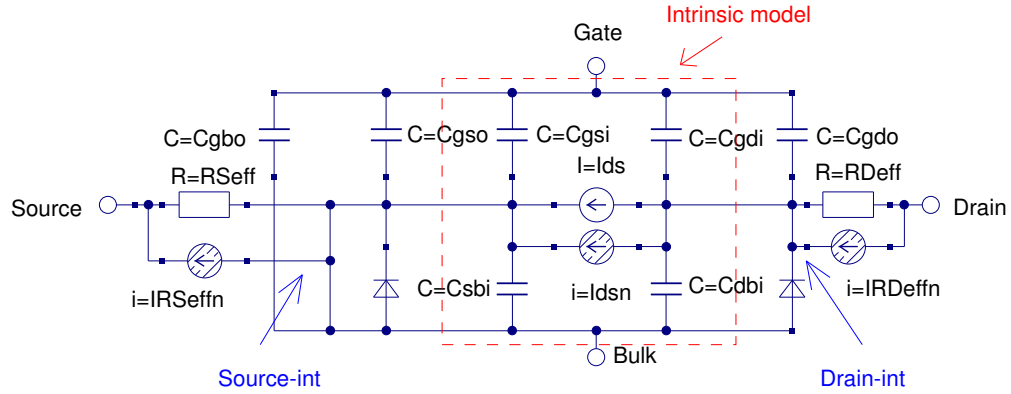
Figure 6.1: Equivalent circuit for the Qucs EKV v2.6 long channel nMOS model

DC current Ids is represented by the equations listed in a later section of the report, capacitors *Cgdi*, *Cgsi*, *Cdbi* and *Csbi* are intrinsic components derived from the charge-based EKV equations, capacitors *Cgdo*, *Cgso* and *Cgbo* represent external overlap elements, the two diodes model the drain to channel and source to channel junctions (including diode capacitance) and resistors *RDeff* and *RSeff* model series connection resistors in the drain and source signal paths respectively.

6.3.1 Long channel model parameters (LEVEL = 1)

Name	Symbol	Description	Unit	Default nMOS	Default pMOS
LEVEL		Model selector		1	1
L	L	length parameter	m	$10e-6$	$10e-6$
W	W	width parameter	m	$10e-6$	$10e-6$
Np	Np	parallel multiple device number		1	1
Ns	Ns	series multiple device number		1	1
Cox	Cox	gate oxide capacitance per unit area	F/m^2	$3.4e-3$	$3.4e-3$
Xj	Xj	metallurgical junction length	m	$0.15e-6$	$0.15e-6$
Dw	Dw	channel width correction	m	$-0.02e-6$	$-0.02e-6$
Dl	Dl	channel length correction	m	$-0.05e-6$	$-0.05e-6$
Vto	Vto	long channel threshold voltage	V	0.5	-0.55
Gamma	$Gamma$	body effect parameter	$\sqrt{V}$	0.7	0.69
Phi	Phi	bulk Fermi potential	V	0.5	0.87
Kp	Kp	transconductance parameter	A/V^2	$50e-6$	$20e-6$
Theta	$Theta$	mobility reduction coefficient	$1/V$	$50e-3$	$50e-3$
Tcv	Tcv	threshold voltage temperature coefficient	V/K	$1.5e-3$	$-1.4e-3$
Hdif	$Hdif$	heavily doped diffusion length	m	$0.9e-6$	$0.9e-6$
Rsh	Rsh	drain-source diffusion sheet resistance	$\Omega/square$	510	510
Rsc	Rsc	source contact resistance	Ω	0.0	0.0
Rdc	Rdc	drain contact resistance	Ω	0.0	0.0
Cgso	$Cgso$	gate to source overlay capacitance	F	$1.5e-10$	$1.5e-10$
Cgdo	$Cgdo$	gate to drain overlay capacitance	F	$1.5e-10$	$1.5e-10$
Cgbo	$Cgbo$	gate to bulk overlay capacitance	F	$4e-10$	$4e-10$
N	N	diode emission coefficient		1.0	1.0
Is	Is	leakage current	A	$1e-1$	$1e-14$
Bv	Bv	reverse breakdown voltage	V	100	100
Ibv	Ibv	current at Bv	A	$1e-3$	$1e-3$

Name	Symbol	Description	Unit	Default nMOS	Default pMOS
Vj	Vj	junction potential	V	1.0	1.0
Cj0	$Cj0$	zero bias depletion capacitance	F	$1e-12$	$1e-12$
M	M	grading coefficient		0.5	0.5
Area	$Area$	relative area		1.0	1.0
Fc	Fc	forward-bias depletion capcitance coefficient		0.5	0.5
Tt	Tt	transit time	s	$0.1e-9$	$0.1e-9$
Xti	Xti	saturation current temperature exponent		3.0	3.0
Kf	KF	flicker noise coefficient		$1e-27$	$1e-28$
Af	Af	flicker noise exponent		1.0	1.0
Tnom	$Tnom$	parameter measurement temperature	$°C$	26.85	26.85
Temp	$Temp$	device temperature	$°C$	26.85	26.85

6.3.2 Fundamental long channel DC model equations (LEVEL = 1)

<22> $Vg = V(Gate) - V(Bulk)$

<23> $Vs = V(Source) - V(Bulk)$

<24> $Vd = V(Drain) - V(Bulk)$

<33> $VGprime = Vg - Vto + Phi + Gamma \cdot \sqrt{Phi}$

<34> $Vp = VGprime - Phi - Gamma \cdot \left(\sqrt{VGprime + \left[\dfrac{Gamma}{2} \right]^2} - \dfrac{Gamma}{2} \right)$

<39> $n = 1 + \dfrac{Gamma}{2 \cdot \sqrt{Vp + Phi + 4 \cdot Vt}}$

<58, 64> $\beta = Kp \cdot \dfrac{W}{L} \cdot \dfrac{1}{1 + Theta \cdot Vp}$

<44> $X1 = \dfrac{Vp - Vs}{Vt} \qquad If = \left[ln \left\{ 1 + limexp \left(\dfrac{X1}{2} \right) \right\} \right]^2$

<57> $X2 = \dfrac{Vp - Vd}{Vt} \qquad Ir = \left[ln \left\{ 1 + limexp \left(\dfrac{X2}{2} \right) \right\} \right]^2$

<65> $Ispecific = 2 \cdot n \cdot \beta \cdot Vt^2$

<66> $Ids = Ispecific \cdot (If - Ir)$

Where *VGprime* is the effective gate voltage, *Vp* is the pinch-off voltage, *n* is the slope factor, β is a transconductance parameter, *Ispecific* is the specific current, *If* is the forward current, *Ir* is the reverse current, *Vt* is the thermal voltage at the device temperature, and *Ids* is the drain to source current. EKV v2.6 equation numbers are given in "< >" brackets at the left-hand side of each equation. Typical plots of Ids against Vds for both the nMOS and pMOS long channel devices are given in Figure 6.2.

EKV26nMOS1
LEVEL=2
L=10e-6
W=10e-6
Vto=0.6
Gamma=0.71
Phi=0.97
Kp=50e-6
Theta=50e-3

dc simulation

DC1

Idsn

V2
U=Vdsn

V1
U=Vgsn

Idsp

V3
U=Vdsp

V4
U=Vgsp

EKV26pMOS1
LEVEL=2
L=10e-6
W=25e-6
Vto=-0.55
Gamma=0.69
Phi=0.87
Kp=20e-6
Theta=50e-3

Equation

Eqn1
Ids_pMOS_3D=PlotVs(Idsp.I, Vgsp, Vdsp)
Ids_pMOS=PlotVs(Idsp.I, Vgsp, Vdsp)
Ids_nMOS=PlotVs(Idsn.I, Vgsn, Vdsn)
Vdsp=-Vdsn
Vgsp=-Vgsn
Ids_nMOS_3D=PlotVs(Idsn.I, Vgsn, Vdsn)

Parameter sweep

SW2
Sim=DC1
Type=lin
Param=Vdsn
Start=0
Stop=3
Points=61

Parameter sweep

SW1
Sim=SW2
Type=lin
Param=Vgsn
Start=0
Stop=3
Points=17

Figure 6.2: Ids versus Vds plots for the Qucs EKV v2.6 long channel nMOS and pMOS models

6.4 Testing model performance

Implementing advanced component models like the EKV v2.6 MOSFET model is a complex process, involving the translation of a set of equations into the Verilog-A hardware design language, conversion of the Verilog-A code into C++ code via the ADMS compiler, and finally compiling and linking the model code with the main body of Qucs code. At all stages in the process accuracy becomes an important issue. This section of the Qucs EKV v2.6 report introduces a number of test simulations which were used during the model development cycle to check the performance of the Qucs EKV v2.6 implementation. The tests also demonstrate how a circuit simulator can be used to extract model parameters. The values of which help to confirm correct model operation.

6.4.1 Extraction of Ispec

When a MOS transistor is operating in the saturation region, reverse current Ir approaches zero and the drain to source current is approximated by

$$Ids = Ispecific \cdot If = Ispecific \cdot \left[ln\{1 + limexp\left(\frac{Vp - Vs}{2 \cdot Vt}\right)\}\right]^2 \qquad (6.1)$$

In saturation $limexp\left(\frac{Vp-Vs}{2 \cdot Vt}\right) >> 1$, yielding

$$\sqrt{Ids} = \sqrt{\frac{Ispecific}{2 \cdot Vt^2}} \cdot (Vp - Vs) \qquad (6.2)$$

Hence

$$\frac{\partial(\sqrt{Ids})}{\partial Vs} = -\sqrt{\frac{Ispecific}{2 \cdot Vt^2}} = -slope \qquad (6.3)$$

Or

$$Ispecific = 2 \cdot slope^2 \cdot Vt^2 \qquad (6.4)$$

Figure 6.3 shows a typical test circuit configuration for measuring and simulating Ids with varying Vs. Qucs post-simulation functions in equation block Eqn1 are used to calculate the value for $Ispecific$. The value of $Ispecific$ for the nMOS transistor with the parameters given in Fig. 6.3 is 3.95e-8 A. Figure 6.4 illustrates a test circuit for measuring Vp with the transistor in saturation. In this circuit $Is=Ispecific$ and the threshold voltage corresponds to Vg when $Vp = 0$V. Notice also that $n = \partial Vg/\partial Vp$. In Fig. 6.4 Qucs post-simulation processing functions are also used to generate data for Vp, $VGprime$ and n. The value of the threshold voltage for the device shown in Fig. 6.4 is 0.6V. At this voltage n = 1.37. The two test configurations illustrated in Figs. 6.3 and 6.4 go some way to confirming

that the Qucs implementation of the EKV v2.6 long channel model is functioning correctly.

6.4.2 Extraction of model intrinsic capacitance

The Qucs implementation of the EKV v2.6 MOSFET model uses the charge-based model for transcapacitances. This model ensures charge-conservation during transient analysis. Both the long channel and short channel versions employ the quasi-static charge-based model. The EKV v2.6 charge equations for the long channel intrinsic device are:

$$nq = 1 + \frac{Gamma}{2 \cdot \sqrt{Vp + Phi + 1e - 6}} \tag{6.5}$$

$$Xf = \sqrt{\frac{1}{4} + If} \tag{6.6}$$

$$Xr = \sqrt{\frac{1}{4} + Ir} \tag{6.7}$$

$$qD = -nq \cdot \left\{ \frac{4}{15} \cdot \frac{3 \cdot Xr^3 + 6 \cdot Xr^2 \cdot Xf + 4 \cdot Xr \cdot Xf^2 + 2 \cdot Xf^3}{(Xf + Xr)^2} - \frac{1}{2} \right\} \tag{6.8}$$

$$qS = -nq \cdot \left\{ \frac{4}{15} \cdot \frac{3 \cdot Xf^3 + 6 \cdot Xf^2 \cdot Xr + 4 \cdot Xf \cdot Xr^2 + 2 \cdot Xr^3}{(Xf + Xr)^2} - \frac{1}{2} \right\} \tag{6.9}$$

$$qI = qS + qD = -nq \cdot \left\{ \frac{4}{3} \cdot \frac{3 \cdot Xf^3 + Xr \cdot Xf + Xr^2}{Xf + Xr} - 1 \right\} \tag{6.10}$$

$$qB = -Gamma \cdot \sqrt{Vp + Phi + 1e - 6} \cdot \frac{1}{Vt} - \left(\frac{nq - 1}{nq} \right) \cdot qI \quad \forall (VGprime > 0) \tag{6.11}$$

$$qB = -VGprime \cdot \frac{1}{Vt} \quad \forall (VGprime <= 0) \tag{6.12}$$

$$qG = -qI - qB \tag{6.13}$$

$$COX = Cox \cdot Np \cdot Weff \cdot Ns \cdot Leff \tag{6.14}$$

EKV26nMOS1
L=10e-6
W=10e-6
Cox=3.45e-3
Vto=0.6
Gamma=0.71
Phi=0.97
Kp=50e-6
Theta=50e-3

dc simulation

DC1

+ V3
− U=2 V

+ V1
− U=Vs

Ids

Parameter sweep

+ V2
− U=3 V

SW1
Sim=DC1
Type=lin
Param=Vs
Start=0
Stop=2
Points=21

Equation

Eqn1
root_Id=sqrt(Ids.I)
Ispecific=(slope^2)*2*vt(300)*vt(300)
slope=-max(abs(diff_root_Id))
diff_root_Id=diff(root_Id, Vs)

number	slope	Ispecific
1	-0.00544	3.95e-08

Vs	root_Id	diff_root_Id
0	0.00588	-0.00544
0.1	0.00534	-0.00544
0.2	0.00479	-0.00544
0.3	0.00425	-0.00544
0.4	0.00371	-0.00544
0.5	0.00316	-0.00543

Figure 6.3: Ispecific extraction test circuit and post simulation data processing results

EKV26nMOS1
LEVEL=1
L=10e-6
W=10e-6
Cox=3.45e-3
Vto=0.6
Gamma=0.71
Phi=0.97
Kp=50e-6

V1
U=Vg

Vs

I1
I=3.95e-8

dc simulation

DC1

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Vg
Start=0
Stop=1
Points=101

Equation

Eqn1
VGprime=Vg-Vto+Phi+Gamma*sqrt(Phi)
Gamma=0.71
Vto=0.6
Phi=0.97
Vp=VGprime-Phi-Gamma*(sqrt(VGprime+(Gamma/2)*(Gamma/2)) -Gamma/2)
n=diff(Vg, Vs.V)

Vg	n	Vs.V	Vp	VGprime
0.57	1.37297	-0.0187	-0.022	1.64
0.58	1.37151	-0.0114	-0.0147	1.65
0.59	1.37006	-0.00408	-0.00735	1.66
0.6	1.36862	0.00322	-1.11e-16	1.67
0.61	1.3672	0.0105	0.00735	1.68
0.62	1.3658	0.0179	0.0147	1.69

Figure 6.4: Vp extraction test circuit and post simulation data processing results

$$Q(I, B, D, S, G) = COX \cdot Vt \cdot q(I, B, D, S, G) \tag{6.15}$$

The first release of the Qucs EKV v2.6 MOSFET model assumes that the gate and bulk charge is partitioned between the drain and source in equal ratio[7]. Fifty percent charge portioning yields the following *Ids* current contributions:

$$I(Gate, Source\_int) < +0.5 \cdot p\_n\_MOS \cdot ddt(QG) \tag{6.16}$$

$$I(Gate, Drain\_int) < +0.5 \cdot p\_n\_MOS \cdot ddt(QG) \tag{6.17}$$

$$I(Source\_int, Bulk) < +0.5 \cdot p\_n\_MOS \cdot ddt(QB) \tag{6.18}$$

$$I(drain\_int, Bulk) < +0.5 \cdot p\_n\_MOS \cdot ddt(QB) \tag{6.19}$$

Where p_n_MOS = 1 for nMOS devices or -1 for pMOS devices. Charge associated with the extrinsic overlap capacitors, Cgs0, Cgd0 and Cgb0, is represented in the Qucs EKV v2.6 implementation by the following equations:

$$Qgs0 = Cgs0 \cdot Weff \cdot Np \cdot (VG - VS) \tag{6.20}$$

$$Qgd0 = Cgd0 \cdot Weff \cdot Np \cdot (VG - VD) \tag{6.21}$$

$$Qgb0 = Cgb0 \cdot Leff \cdot Np \cdot VB \tag{6.22}$$

The drain to bulk and source to bulk diodes also introduce additional components in the extrinsic capacitance model. The default value of CJ0 being set at 300fF. Analysis of the y-parameters[8] for the EKV v2.6 equivalent circuit shown in the test circuit illustrated in Fig. 6.5 yields

$$y_{11} = \frac{j \cdot \omega \cdot Cg}{1 + \omega^2 \cdot (Rgn \cdot Cg)^2} \tag{6.23}$$

---

[7]For an example of this type of charge partitioning see F. Pregaldiny et. al., An analytic quantum model for the surface potential of deep-submicron MOSFETS, 10th International Conference, MIXDES 2003, Lodz, Poland, 26-28 June 2003.

[8]A more detailed anlysis of the EKV v2.6 y-parameters can be found in F. Krummenacher *et. al.*, HF MOSET MODEL parameter extraction, European Project No. 25710, Deliverable D2.3, July 28, 2000.

Or

$$y_{11} \cong \omega^2 \cdot Rg \cdot Cg^2 + j \cdot \omega \cdot Cg, \quad \text{when} \quad \omega \cdot Rg \cdot Cg << 1. \qquad (6.24)$$

Hence, $Cg = imag(y_{11}/\omega)$ and $Rgn = real(y_{11}/(\omega^2 \cdot Cg^2))$, where $\omega = 2 \cdot \pi \cdot f$, and f is the frequency of y-parameter measurement, Rg is a series extrinsic gate resistance and $Cg \cong Cgs + Cgd + Cgb$. With equal partitioning of the intrinsic gate charge Cgb approximates to zero and $Cg \cong Cgs + Cgd$. The data illustrated in Figures 6.5 and 6.6 shows two features which are worth commenting on; firstly the values of Cg are very much in line with simple hand calculations (for example in the case of the nMOS device $Cg(max) = W \cdot L \cdot Cox = 10e-6*10e-6*3.45e-3 = 3.45e - 13F$) and secondly both sets of simulation data indicate the correct values for the nMOS and pMOS threshold voltages (for example -0.55 V for the pMOS device and 0.6 V for the nMOS device), reinforcing confidence in the EKV v2.6 model implementation.

6.4.3 Extraction of extrinsic diode capacitance and drain resistance

The extrinsic section of the EKV v2.6 model includes diodes which in turn are modelled by conventional DC characteristics and parallel capacitance. This capacitance is represented by depletion layer capacitance in the diode reverse bias region of operation. In the diode forward bias section of the I-V characteristic diffusion capacitance dominates. Figure 6.7 illustrates a test circuit that allows the diode capacitance to be extracted as a function of Vds. In Fig. 6.7 the nMOS device is turned off and the drain to bulk diode reverse biased. Simple analysis indicates that

$$y_{11} \cong \omega^2 \cdot RDeff \cdot Cd^2 + j \cdot \omega \cdot Cd, \quad \text{when} \quad \omega \cdot RDeff \cdot Cd << 1. \qquad (6.25)$$

Hence, $Cd = imag(y_{11}/\omega)$ and $Rdeff = real(y_{11}/(\omega^2 \cdot Cd^2))$, where $\omega = 2 \cdot \pi \cdot f$, f is the frequency of y-parameter measurement, and Cd is the diode capacitance. The data shown in Fig. 6.7 indicate good agreement with the expected values for Cd and $RDeff$; which are expected to be $Cd = 300$fF at Vds=0V, and $Rdeff = 46\Omega$.

6.4.4 Simulating EKV v2.6 MOSFET noise

The EKV v2.6 intrinsic device noise is modelled by a noise current source connected between the internal drain and source terminals. The noise current source $Idsn$, see Fig. 6.1, is composed of a thermal noise component and a flicker noise component. The Power Spectral Density (S_{PSD}) of these components are given by:

dc simulation

DC1

P1
Num=1
Z=50 Ohm

Rgn
R=1

X1 + V1
    − U=Vgs

Ids_n

+ V2
− U=Vds

EKV26nMOS1
LEVEL=1
L=10e-6
W=10e-6
Cox=3.45e-3
Vto=0.6
Gamma=0.71
Phi=0.97
Kp=50e-6
Theta=50e-3
Cgso=1.5e-10
Cgdo=1.5e-10
Cgbo=4.0e-10
Cj0=300e-15

Parameter sweep

SW1
Sim=SW2
Type=lin
Param=Vds
Start=0
Stop=3
Points=31

Parameter sweep

SW2
Sim=SP1
Type=lin
Param=Vgs
Start=-2
Stop=2
Points=201

Equation
Eqn1
y=stoy(S)
L=10e-6
W=10e-6
Cox=3.45e-3
Cg=imag(y[1,1])/Omega
Cgpl=PlotVs(Cg/(Cox*W*L), Vds, Vgs)
Rg=PlotVs((real(y[1,1])/(Omega*Omega*Cg*Cg)), Vds, Vgs)
Cg_2D=PlotVs(Cg, Vgs)
Omega=2*pi*frequency

S parameter simulation

SP1
Type=const
Values=[1 MHz]

Figure 6.5: y_{11} test circuit and values of Cg for the long channel EKV v2.6 nMOS model

dc simulation

DC1

P1
Num=1
Z=50 Ohm

X1 + V1
− U=Vgs

Rgn
R=1

Ids_n

+ V2
− U=Vds

EKV26pMOS1
L=10e-6
W=20e-6
Cox=3.45e-3
Vto=-0.55
Gamma=0.69
Phi=0.87
Kp=20e-6
Theta=50e-3
Cgso=1.5e-10
Cgdo=1.5e-10
Cgbo=4.0e-10
Cj0=300e-15

Parameter sweep

SW1
Sim=SW2
Type=lin
Param=Vds
Start=0
Stop=-3
Points=31

Parameter sweep

SW2
Sim=SP1
Type=lin
Param=Vgs
Start=2
Stop=-2
Points=201

Equation
Eqn1
y=stoy(S)
L=10e-6
W=10e-6
Cox=3.45e-3
Cg=imag(y[1,1])/Omega
Cgpl=PlotVs(Cg/(Cox*W*L), Vds, Vgs)
Rg=PlotVs((real(y[1,1])/(Omega*Omega*Cg*Cg)), Vds, Vgs)
Cg_2D=PlotVs(Cg, Vgs)
Omega=2*pi*frequency

S parameter simulation

SP1
Type=const
Values=[1 MHz]

Figure 6.6: y_{11} test circuit and values of Cg for the long channel EKV v2.6 pMOS model

142

EKV26nMOS1
LEVEL=1
L=10e-6
W=10e-6
Cox=3.45e-3
Vto=0.6
Gamma=0.71
Phi=0.97
Kp=50e-6
Theta=50e-3

Ids

dc simulation

DC1

V1
U=0

S parameter simulation

SP1
Type=const
Values=[1 MHz]

Equation
Eqn2
y=stoy(S)
Cd=PlotVs(imag(y[1,1])/Omega, Vgs, Vds)
Cd_2D=PlotVs(imag(y[1,1])/Omega, Vds)
RDeff=real(y[1,1])/(Omega^2*Cd^2)
PL_RDeff=PlotVs(real(y[1,1])/(Omega^2*Cd^2),Vgs,Vds)
Omega=2*pi*frequency

X1   V2
U=Vds

P1
Num=1
Z=50 Ohm

Parameter sweep

SW1
Sim=SW2
Type=lin
Param=Vgs
Start=0
Stop=3
Points=31

Parameter sweep

SW2
Sim=SP1
Type=lin
Param=Vds
Start=0
Stop=3
Points=61

Vgs	Vds	Ids.I
0	0	0
	0.05	1.47e-14
	0.1	1.68e-14
	0.15	1.72e-14
	0.2	1.72e-14

Figure 6.7: Test circuit for extracting EKV v2.6 extrinsic diode capacitance

143

$$S_{PSD} = S_{thermal} + S_{flicker} \tag{6.26}$$

Where

- Thermal noise

$$S_{thermal} = 4 \cdot k \cdot T \cdot \beta \cdot \mid qI \mid \tag{6.27}$$

- Flicker noise

$$S_{flicker} = \frac{KF \cdot g_{mg}^2}{Np \cdot Weff \cdot Ns \cdot Leff \cdot Cox \cdot f^{Af}}, \tag{6.28}$$

$$g_{mg} = \frac{\partial Ids}{\partial Vgs} = \beta \cdot Vt \cdot \left(\sqrt{\frac{4 \cdot If}{Ispecific} + 1} - \sqrt{\frac{4 \cdot Ir}{Ispecific} + 1} \right) \tag{6.29}$$

Where β is a transconductance factor, $qI = qD + qS$, and the other symbols are defined in the EKV v2.6 long channel parameter list or have their usual meaning. Noise has been implemented in both the Qucs long channel and short channel EKV v2.6 models. In addition to the intrinsic device noise the Qucs EKV v2.6 model includes the thermal noise components for both extrinsic resistors *RDeff* and *RSeff*. Figure 6.8 presents a typical noise test circuit and simulated noise currents. In Figure 6.8 four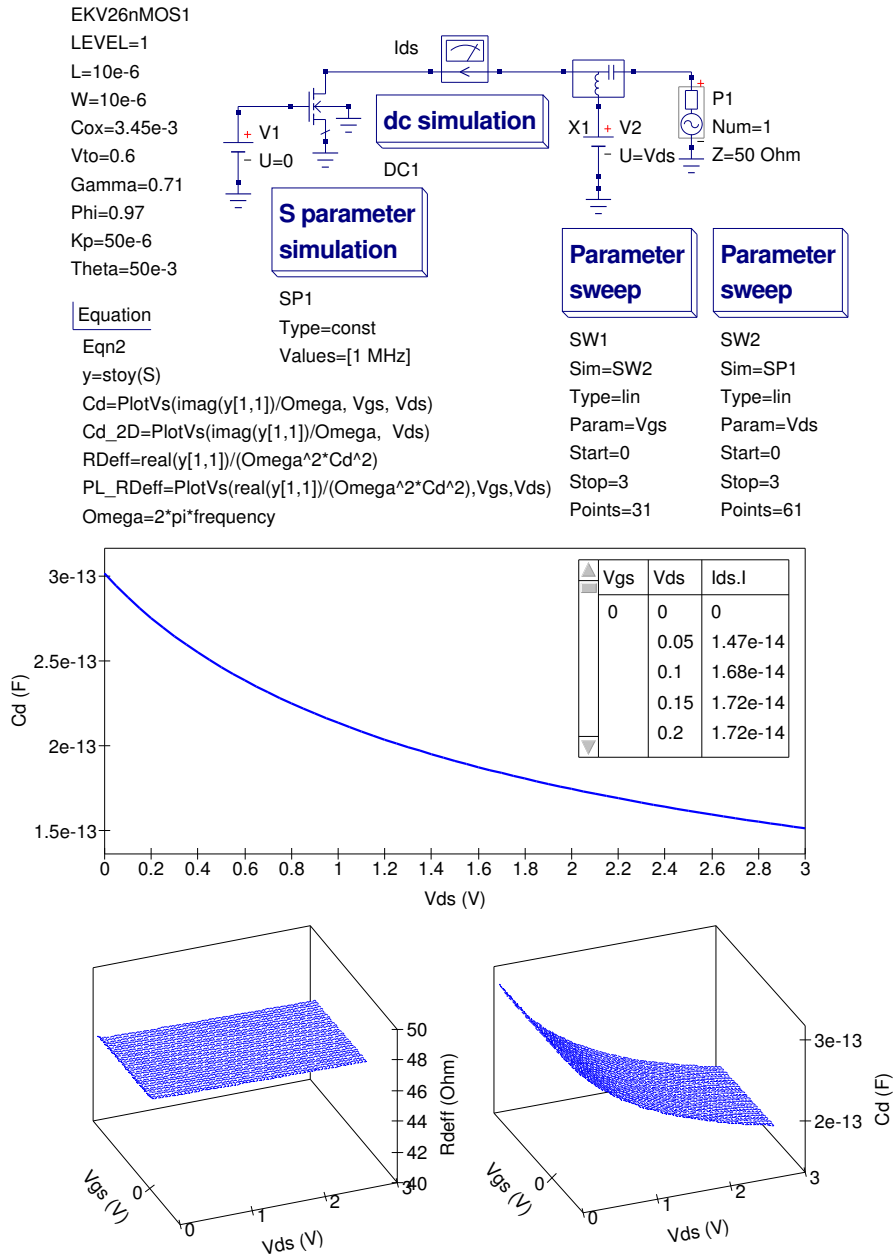 nMOS devices are biased under different DC conditions and their noise current simulated for a range of W values between 1e-6 m and 100e-6 m. The first three devices include both thermal and flicker noise components ($KF =$ 1e-27) while the fourth device has it's flicker component set to zero. The resulting current noise curves clearly demonstrate the effect of summing intrinsic thermal and flicker components on the overall performance of the EKV v2.6 noise model.
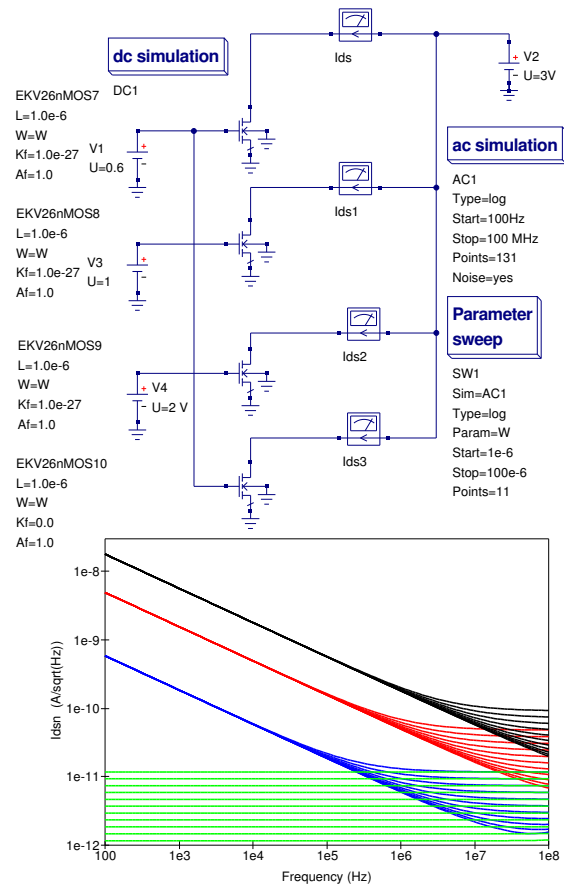
Figure 6.8: Test circuit for simulating EKV v2.6 noise: Ids.in blue curve, Ids1.in red curve, Ids2.in black curve and Ids3.in green curve

6.5 The Qucs short channel EKV v2.6 model

The Qucs implementation of the short short channel EKV v2.6 MOSFET model contains all the features implemented in the long channel version of the model plus a number of characteristics specific to short channel operation. However, the short channel version of the model does not use parameter *Theta*. Parameter LEVEL set to 2 selects the short channel model. Both pMOS and nMOS versions of the model are available for both long and short channel implementations. The entire short channel EKV v2.6 MOSFET model is described by roughly 94 equations. Readers who are interested in the mathematics of the model should consult "The EPFL-EKV MOSFET Model Equations for Simulation" publication cited in previous text. Appendix A lists the complete Verilog-A code for the first release of the Qucs EKV v2.6 MOSFET models. Additional Verilog-A code has been added to the model equation code to (1) allow interchange of the drain and source terminals, and (2) select nMOS or pMOS devices.

6.5.1 Short channel model parameters (LEVEL = 2)

Name	Symbol	Description	Unit	Default nMOS	Default pMOS
LEVEL		Model selector		2	2
L	L	length parameter	m	$10e-6$	$10e-6$
W	W	width parameter	m	$10e-6$	$10e-6$
Np	Np	parallel multiple device number		1	1
Ns	Ns	series multiple device number		1	1
Cox	Cox	gate oxide capacitance per unit area	F/m^2	$3.4e-3$	$3.4e-3$
Xj	Xj	metallurgical junction length	m	$0.15e-6$	$0.15e-6$
Dw	Dw	channel width correction	m	$-0.02e-6$	$-0.02e-6$
Dl	Dl	channel length correction	m	$-0.05e-6$	$-0.05e-6$
Vto	Vto	long channel threshold voltage	V	0.5	-0.55
Gamma	$Gamma$	body effect parameter	$\sqrt{V}$	0.7	0.69
Phi	Phi	bulk Fermi potential	V	0.5	0.87
Kp	Kp	transconductance parameter	A/V^2	$50e-6$	$20e-6$
EO	EO	mobility reduction coefficient	V/m	$88e-6$	$51e-6$
Ucrit	$Ucrit$	longitudinal critical field	V/m	$4.5e-6$	$18e-6$
Lambda	$Lambda$	depletion length coefficient		0.23	1.1
Weta	$Weta$	narrow channel effect coefficient		0.05	0.0
Leta	eta	short channel effect coefficient		0.28	0.45
Q0	$Q0$	reverse short channel effect peak charge density		$280e-6$	$200e-6$
Lk	Lk	reverse short channel effect characteristic length	m	$0.5e-6$	$0.6e-6$
Tcv	Tcv	threshold voltage temperature coefficient	V/K	$1.5e-3$	$-1.4e-3$
Bex	Bex	mobility temperature coefficient		-1.5	-1.4
Ucex	$Ucex$	longitudinal critical field temperature exponent		1.7	2.0
Ibbt	$Ibbt$	temperature coefficient for Ibb	$1/K$	0.0	0.0
Hdif	$Hdif$	heavily doped diffusion length	m	$0.9e-6$	$0.9e-6$
Rsh	Rsh	drain-source diffusion sheet resistance	$\Omega/square$	510	510
Rsc	Rsc	source contact resistance	Ω	0.0	0.0
Rdc	Rdc	drain contact resistance	Ω	0.0	0.0
Cgso	$Cgso$	gate to source overlay capacitance	F	$1.5e-10$	$1.5e-10$
Cgdo	$Cgdo$	gate to drain overlay capacitance	F	$1.5e-10$	$1.5e-10$
Cgbo	$Cgbo$	gate to bulk overlay capacitance	F	$4e-10$	$4e-10$
N	N	diode emission coefficient		1.0	1.0
Is	Is	leakage current	A	$1e-1$	$1e-14$

Name	Symbol	Description	Unit	Default nMOS	Default pMOS
Bv	Bv	reverse breakdown voltage	V	100	100
Ibv	Ibv	current at Bv	A	$1e-3$	$1e-3$
Vj	Vj	junction potential	V	1.0	1.0
Cj0	$Cj0$	zero bias depletion capacitance	F	$1e-12$	$1e-12$
M	M	grading coefficient		0.5	0.5
Area	$Area$	relative area		1.0	1.0
Fc	Fc	forward-bias depletion capcitance coefficient		0.5	0.5
Tt	Tt	transit time	s	$0.1e-9$	$0.1e-9$
Xti	Xti	saturation current temperature exponent		3.0	3.0
Kf	KF	flicker noise coefficient		$1e-27$	$1e-28$
Af	Af	flicker noise exponent		1.0	1.0
Avto	$Avto$	area related threshold mismatch parameter		0	0
Akp	Akp	area related gain mismatch parameter		0	0
Agamma	$Agamma$	area related body effect mismatch parameter		0	0
Iba	Iba	first impact ionization coefficient	$1/m$	$2e8$	0.0
Ibb	Ibb	second impact ionization coefficient	V/m	$3.5e8$	$3.5e8$
Ibn	Ibn	saturation voltage factor for impact ionization		1.0	1.0
Tnom	$Tnom$	parameter measurement temperature	$^{\circ}C$	26.85	26.85
Temp	$Temp$	device temperature	$^{\circ}C$	26.85	26.85

6.5.2 Simulating short channel charge sharing effects

A simple test circuit for demonstrating the effects of charge sharing is given in Figure 6.9. With charge sharing disabled, by setting Weta and Leta to zero, the magnitude and slope of the *Ids* vs. *Vds* curve shows a marked difference to that where charge sharing is enabled. One point to note with this test: charge sharing in short channel devices significantly reduces the device output resistance which could have, of course, important consequences on circuit performance.
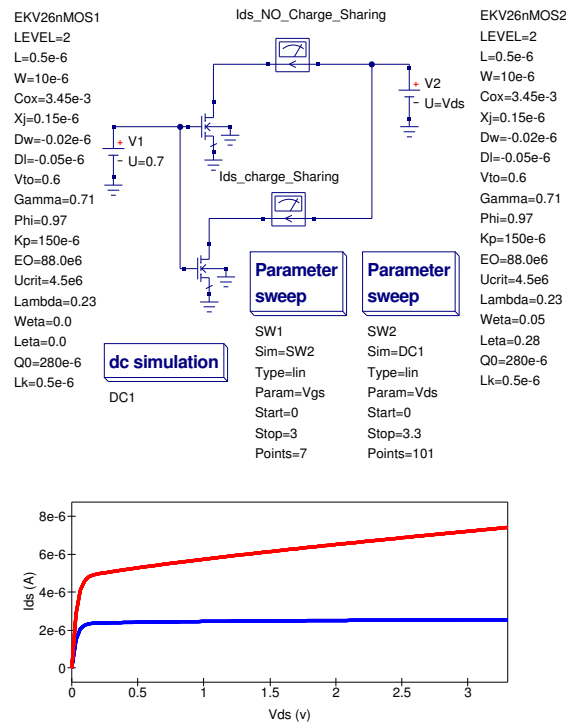
Figure 6.9: Test circuit for simulating EKV v2.6 charge sharing effects in short channel devices: *Ids* blue curve; NO charge sharing (*Weta* = 0.0, *Leta* = 0.0), *Ids* red curve; charge sharing (*Weta* = 0.05, *Leta* = 0.28)

6.6 End note

This report outlines some of the background to the Qucs implementation of the EKV v2.6 MOSFET model. A series of test results demonstrate a range of results that have been achieved with this new Qucs compact device model. Although the test results give data similar to what is expected in all cases it must be stressed that this is the first release of this MOSFET model and as such it will probably contain bugs. A great deal of work has gone into providing this new Qucs model. However, all the effort has been worthwhile because for the first time Qucs now has a submicron MOSFET model. Please use the model and report bugs to the Qucs development team. Much work still remains to be done in the development of MOSFET models for Qucs. In future releases both bug fixes and new models are likely to feature strongly. Once again I would like to thank Stefan Jahn and Władysław Grabiński (of MOS-AK) for their encouragement and support during the period I have been working on developing the Qucs implementation of the EKV v2.6 model and writing this report.

6.7 Qucs Verilog-A code for the EKV v2.6 MOSFET model

6.7.1 nMOS: EKV equation numbers are given on the right-hand side of code lines

```
//      Qucs EPFL–EKV 2.6 nMOS model:
//
//      The structure and theoretical background to the EKV 2.6
//      Verilog−a model is presented in the Qucs EPFL–EKV 2.6 report.
//      Typical parameters are for 0.5um CMOS (C) EPLFL–LEG 1999.
//      Geometry range: Short channel W >= 0.8um, L >= 0.5um
//                      Long channel  W >= 2um,   L >= 2um
//      Voltage range:  |Vgb| < 3.3V, |Vdb| < 3.3V, |Vsb| < 2V
//
//      This is free software; you can redistribute it and/or modify
//      it under the terms of the GNU General Public License as published by
//      the Free Software Foundation; either version 2, or (at your option)
//      any later version.
//
//      Copyright (C), Mike Brinson, mbrin72043@yahoo.co.uk, May 2008.
//
`include "disciplines.vams"
`include "constants.vams"

//
 module EKV26nMOS (Drain, Gate, Source, Bulk);
 inout Drain, Gate, Source, Bulk;
electrical Drain, Gate, Source, Bulk;
// Internal nodes
electrical Drain_int, Source_int;
```

```verilog
`define attr(txt) (*txt*)
// Device dimension parameters
 parameter real LEVEL = 1 from [1 : 2]
        `attr(info="long_=_1,_short_=_2");
 parameter real L = 0.5e-6 from [0.0 : inf]
        `attr(info="length_parameter" unit = "m" );
 parameter real W = 10e-6 from [0.0 : inf]
        `attr(info="Width_parameter" unit = "m");
 parameter real Np = 1.0 from [1.0 : inf]
        `attr(info="parallel_multiple_device_number");
 parameter real Ns = 1.0 from [1.0 : inf]
        `attr(info="series_multiple_device_number");
// Process parameters
 parameter real Cox = 3.45e-3 from [0 : inf]
        `attr(info="gate_oxide_capacitance_per_unit_area" unit = "F/m**2" );
 parameter real Xj = 0.15e-6 from [0.01e-6 : 1.0e-6]
        `attr(info="metallurgical_junction_depth" unit="m");
 parameter real Dw = -0.02e-6 from [-inf : 0.0]
        `attr(info="channel_width_correction" unit="m");
 parameter real Dl = -0.05e-6 from [-inf : 0.0]
        `attr(info="channel_length_correction" unit="m");
// Basic intrinsic model parameters
 parameter real Vto = 0.6 from [1e-6 : 2.0]
        `attr(info="long_channel_threshold_voltage" unit="V" );
 parameter real Gamma = 0.71 from [0.0 : 2.0]
        `attr(info="body_effect_parameter" unit="V**(1/2)");
 parameter real Phi = 0.97 from [0.3 : 2.0]
        `attr(info="bulk_Fermi_potential" unit="V");
 parameter real Kp = 150e-6 from [10e-6 : inf]
        `attr(info="transconductance_parameter" unit = "A/V**2");
 parameter real Theta = 50e-3 from [0.0 : inf]
        `attr(info="mobility_reduction_coefficient" unit = "1/V");
 parameter real EO = 88.0e6 from [1.0e6 : inf]
        `attr(info="mobility_coefficient" unit="V/m");
 parameter real Ucrit = 4.5e6 from [2.0e6 : 25.0e6]
        `attr(info="longitudinal_critical_field" unit="V/m");
// Channel length and charge sharing parameters
 parameter real Lambda = 0.23 from [0.1 : inf]
        `attr(info="depletion_length_coefficient");
 parameter real Weta = 0.05 from [0.0 : inf]
        `attr(info="narrow-channel_effect_coefficient");
 parameter real Leta = 0.28 from [0.0 : inf]
        `attr(info="longitudinal_critical_field");
// Reverse short channel effect parameters
 parameter real Q0 = 280e-6 from [0.0 : inf]
        `attr(info="reverse_short_channel_charge_density" unit="A*s/m**2");
 parameter real Lk = 0.5e-6 from [0.0 : inf]
        `attr(info="characteristic_length" unit="m");
// Intrinsic model temperature parameters
 parameter real Tcv = 1.5e-3
        `attr(info="threshold_voltage_temperature_coefficient" unit="V/K");
 parameter real Bex = -1.5
        `attr(info="mobility_temperature_coefficient");
 parameter real Ucex = 1.7
        `attr(info="Longitudinal_critical_field_temperature_exponent");
 parameter real Ibbt = 0.0
        `attr(info="Ibb_temperature_coefficient" unit="1/K");
// Series resistance calculation parameters
 parameter real Hdif = 0.9e-6 from [0.0 :inf]
        `attr(info="heavily_doped_diffusion_length" unit = "m");
 parameter real Rsh = 510.0 from [0.0 : inf]
        `attr(info="drain/source_diffusion_sheet_resistance" unit="Ohm/square");
```

```
  parameter real Rsc = 0.0 from [0.0 : inf]
        'attr(info="source_contact_resistance" unit="Ohm");
  parameter real Rdc = 0.0 from [0.0 : inf]
        'attr(info="drain_contact_resistance" unit="Ohm");
// Gate overlap capacitances
  parameter real Cgso = 1.5e-10 from [0.0 :inf]
        'attr(info="gate_to_source_overlap_capacitance" unit = "F/m");
  parameter real Cgdo = 1.5e-10 from [0.0 : inf]
        'attr(info="gate_to_drain_overlap_capacitance"  unit=  "F/m");
  parameter real Cgbo = 4.0e-10 from [0.0 : inf]
        'attr(info="gate_to_bulk_overlap_capacitance"    unit=   "F/m");
// Impact ionization related parameters
  parameter real Iba = 2e8 from [0.0 :inf]
        'attr(info="first_impact_ionization_coefficient" unit = "1/m");
  parameter real Ibb = 3.5e8 from [1.0e8 : inf]
        'attr(info="second_impact_ionization_coefficient" unit="V/m");
  parameter real Ibn = 1.0 from [0.1 : inf]
        'attr(info="saturation_voltage_factor_for_impact_ionization");
// Flicker noise parameters
  parameter real Kf = 1.0e-27 from [0.0 :inf]
        'attr(info="flicker_noise_coefficient");
  parameter real Af = 1.0 from [0.0 : inf]
        'attr(info="flicker_noise_exponent" );
// Matching parameters
  parameter real Avto = 0.0 from [0.0 :inf]
        'attr(info="area_related_theshold_voltage_mismatch_parameter" unit = "V*m");
  parameter real Akp = 0.0 from [0.0 : inf]
        'attr(info="area_related_gain_mismatch_parameter" unit="m");
  parameter real Agamma = 0.0 from [0.0 : inf]
        'attr(info="area_related_body_effect_mismatch_parameter" unit="sqrt(V)*m");
// Diode parameters
  parameter real N=1.0 from [1e-6:inf]
        'attr(info="emission_coefficient");
  parameter real Is=1e-14 from [1e-20:inf]
        'attr(info="saturation_current" unit="A" );
  parameter real Bv=100 from [1e-6:inf]
        'attr(info="reverse_breakdown_voltage" unit="V");
  parameter real Ibv=1e-3 from [1e-6:inf]
        'attr(info="current_at_reverse_breakdown_voltage" unit="A");
  parameter real Vj=1.0 from [1e-6:inf]
        'attr(info="junction_potential" unit="V");
  parameter real Cj0=300e-15 from [0:inf]
        'attr(info="zero-bias_junction_capacitance" unit="F");
  parameter real M=0.5 from [1e-6:inf]
        'attr(info="grading_coefficient");
  parameter real Area=1.0 from [1e-3:inf]
        'attr(info="diode_relative_area");
  parameter real Fc=0.5 from [1e-6:inf]
        'attr(info="forward-bias_depletion_capcitance_coefficient");
  parameter real Tt=0.1e-9 from [1e-20:inf]
        'attr(info="transit_time" unit="s" );
  parameter real Xti=3.0 from [1e-6:inf]
        'attr(info="saturation_current_temperature_exponent");
// Temperature parameters
  parameter real Tnom = 26.85
        'attr(info="parameter_measurement_temperature" unit = "Celsius");
// Local variables
real epsilonsi, epsilonox, Tnomk, T2, Tratio, Vto_T, Ucrit_T, Egnom, Eg, Phi_T;
real Weff, Leff, RDeff, RSeff, con1, con2, Vtoa, Kpa,Kpa_T,Gammaa, C_epsilon, xi;
real nnn, deltaV_RSCE, Vg, Vs, Vd, Vgs, Vgd, Vds, Vdso2, VG, VS, VD;
real VGprime, VP0, VSprime, VDprime, Gamma0, Gammaprime, Vp;
real n, X1, iff, X2, ir, Vc, Vdss, Vdssprime, deltaV, Vip;
```

```
real Lc, DeltaL, Lprime, Lmin, Leq, X3, irprime, Beta0, eta;
real Qb0, Beta0prime, nq, Xf, Xr, qD, qS, qI, qB, Beta, Ispecific, Ids, Vib, Idb, Ibb_T;
real A, B, Vt_T2, Eg_T1, Eg_T2, Vj_T2, Cj0_T2, F1, F2, F3, Is_T2;
real Id1, Id2, Id3, Id4, Is1, Is2, Is3, Is4, V1, V2, Ib_d, Ib_s, Qd, Qs, Qd1, Qd2, Qs1, Qs2;
real qb, qg, qgso, qgdo, qgbo, fourkt, Sthermal, gm, Sflicker, StoDswap, p_n_MOS;
//
analog begin
// Equation initialization
p_n_MOS = 1.0; // nMOS
A=7.02e-4;
B=1108.0;
epsilonsi = 1.0359e-10;        // Eqn 4
epsilonox = 3.453143e-11;      // Eqn 5
Tnomk = Tnom+273.15;                    // Eqn 6
T2=$temperature;
Tratio = T2/Tnomk;
Vto_T = Vto-Tcv*(T2-Tnomk);
Egnom = 1.16-0.000702*Tnomk*Tnomk/(Tnomk+1108);
Eg =     1.16-0.000702*T2*T2/(T2+1108);
Phi_T = Phi*Tratio - 3.0*$vt*ln(Tratio)-Egnom*Tratio+Eg;
Ibb_T = Ibb*(1.0+Ibbt*(T2 -Tnomk));
Weff = W + Dw;    // Eqn 25
Leff = L + Dl;    // Eqn 26
RDeff = ( (Hdif*Rsh)/Weff)/Np + Rdc;
RSeff = ( (Hdif*Rsh)/Weff)/Np + Rsc;
con1 = sqrt(Np*Weff*Ns*Leff);
Vt_T2=`P_K*T2/`P_Q;
Eg_T1=Eg-A*Tnomk*Tnomk/(B+Tnomk);
Eg_T2=Eg-A*T2*T2/(B+T2);
Vj_T2=(T2/Tnomk)*Vj-(2*Vt_T2)*ln(pow((T2/Tnomk),1.5))-((T2/Tnomk)*Eg_T1-Eg_T2);
Cj0_T2=Cj0*(1+M*(400e-6*(T2-Tnomk)-(Vj_T2-Vj)/Vj));
F1=(Vj/(1-M))*(1-pow((1-Fc),(1-M)));
F2=pow((1-Fc), (1+M));
F3=1-Fc*(1+M);
Is_T2=Is*pow( (T2/Tnomk), (Xti/N))*limexp((-Eg_T1/Vt_T2)*(1-T2/Tnomk));
con2 = (Cox*Ns*Np*Weff*Leff);
fourkt = 4.0*`P_K*T2;
//
if (LEVEL == 2)
 begin
  Ucrit_T = Ucrit*pow(Tratio, Ucex);
  Vtoa = Vto+Avto/con1;                 // Eqn 27
  Kpa = Kp*(1.0+Akp/con1);              // Eqn 28
  Kpa_T = Kpa*pow( Tratio, Bex);        // Eqn 18
  Gammaa = Gamma+Agamma/con1;           // Eqn 29
  C_epsilon = 4.0*pow(22e-3, 2);        // Eqn 30
  xi = 0.028*(10.0*(Leff/Lk)-1.0);      // Eqn 31
  nnn = 1.0+0.5*(xi+ sqrt(pow(xi,2) + C_epsilon));
  deltaV_RSCE = (2.0*Q0/Cox)*(1.0/pow(nnn,2)); // Eqn 32
 end
//
// Model branch and node voltages
//

Vg = p_n_MOS*V(Gate, Bulk);
Vs = p_n_MOS*V(Source, Bulk);
Vd = p_n_MOS*V(Drain, Bulk);
VG=Vg;           // Eqn 22
if ( (Vd-Vs) >= 0.0)
        begin
            StoDswap = 1.0;
            VS=Vs;   // Eqn 23
```

```
                VD=Vd;      // Eqn 24
            end
else
            begin
            StoDswap = −1.0;
            VD=Vs;
            VS=Vd;
            end
if (LEVEL == 2)
    VGprime=VG−Vto_T−deltaV_RSCE+Phi_T+Gamma*sqrt(Phi_T);   // Eqn 33   nMOS equation
else
    VGprime=Vg−Vto_T+Phi_T+Gamma*sqrt(Phi_T);

//
if (LEVEL == 2)
 begin
  if (VGprime > 0)
        VP0=VGprime−Phi_T−Gammaa*(sqrt(VGprime+(Gammaa/2.0)*(Gammaa/2.0))
             −(Gammaa/2.0));   // Eqn 34
  else
        VP0 = −Phi_T;
  VSprime=0.5*(VS+Phi_T+sqrt(pow( (VS+Phi_T),2) + pow( (4.0*$vt),2)));   // Eqn 35
  VDprime=0.5*(VD+Phi_T+sqrt(pow( (VD+Phi_T),2) + pow( (4.0*$vt),2)));   // Eqn 35
  Gamma0=Gammaa−(epsilonsi/Cox)*((Leta/Leff)*(sqrt(VSprime)+sqrt(VDprime))
             −(3.0*Weta/Weff)*sqrt(VP0+Phi_T));   // Eqn 36
  Gammaprime = 0.5*(Gamma0+sqrt( pow(Gamma0,2) +0.1*$vt ));   // Eqn 37
  if (VGprime > 0.0 )
        Vp = VGprime−Phi_T−Gammaprime*(sqrt(VGprime+(Gammaprime/2.0)*
             (Gammaprime/2.0)) − (Gammaprime/2.0));   // Eqn 38
  else
        Vp = −Phi_T;
  n = 1.0 +Gammaa/(2.0*sqrt(Vp+Phi_T+4.0*$vt));   // Eqn 39
 end
else
  begin
   if (VGprime > 0)
        Vp=VGprime−Phi_T−Gamma*(sqrt(VGprime+(Gamma/2.0)*(Gamma/2.0))
             −(Gamma/2.0));   // Eqn 34
   else
        Vp = −Phi_T;
   n = 1.0 +Gamma/(2.0*sqrt(Vp+Phi_T+4.0*$vt));   // Eqn 39
end
//
X1 = (Vp−VS)/$vt;
iff = ln(1.0+limexp(X1/2.0))*ln(1.0+limexp(X1/2.0));     // Eqn 44
X2 = (Vp−VD)/$vt;
ir = ln(1.0+limexp(X2/2.0))*ln(1.0+limexp(X2/2.0));       // Eqn 57
//
if (LEVEL == 2)
 begin
   Vc = Ucrit_T*Ns*Leff;   // Eqn 45
   Vdss = Vc*(sqrt( 0.25 + (($vt/(Vc))*sqrt(iff)))−0.5);  // Eqn 46;
   Vdssprime = Vc*(sqrt( 0.25 + ($vt/Vc)*(sqrt(iff)−0.75*ln(iff))) − 0.5)
             +$vt*(ln(Vc/(2.0*$vt)) − 0.6 );  // Eqn 47
   if (Lambda*(sqrt(iff) > (Vdss/$vt) ) )
        deltaV = 4.0*$vt*sqrt(Lambda*(sqrt(iff) −(Vdss/$vt))
                 + (1.0/64.0) );   // Eqn 48
   else
        deltaV = 1.0/64.0;
   Vdso2 = (VD−VS)/2.0;            // Eqn 49
   Vip = sqrt( pow(Vdss, 2) + pow( deltaV,2)) − sqrt( pow( (Vdso2 − Vdss), 2)
         + pow (deltaV, 2));   // Eqn 50
```

153

```
     Lc = sqrt( (epsilonsi/Cox)*Xj);   // Eqn 51
     DeltaL = Lambda*Lc*ln(1.0+((Vdso2−Vip)/(Lc*Ucrit_T)));   // Eqn 52
     Lprime = Ns*Leff − DeltaL + ( (Vdso2+Vip)/Ucrit_T);  // Eqn 53
     Lmin = Ns*Leff/10.0;   // Eqn 54
     Leq = 0.5*(Lprime + sqrt( pow(Lprime, 2) + pow(Lmin, 2)));   // Eqn 55
     X3 = (Vp−Vdso2−VS−sqrt( pow(Vdssprime, 2) + pow( deltaV, 2))
         + sqrt( pow( (Vdso2−Vdssprime), 2) + pow(deltaV,2)))/$vt;
     irprime = ln(1.0+limexp(X3/2.0))*ln(1.0+limexp(X3/2.0));   // Eqn 56
     Beta0 = Kpa_T*(Np*Weff/Leq);      // Eqn 58
     eta = 0.5;   // Eqn 59 − nMOS
     Qb0 = Gammaa*sqrt(Phi_T);   // Eqn 60;
     Beta0prime = Beta0*(1.0 +(Cox/(EO*epsilonsi))*Qb0);   // Eqn 61
     nq = 1.0 +Gammaa/(2.0*sqrt(Vp+Phi_T+1e−6));       // Eqn 69
  end
 else
 nq = 1.0 +Gamma/(2.0*sqrt(Vp+Phi_T+1e−6));             // Eqn 69
//
Xf = sqrt(0.25+ iff);      // Eqn 70
Xr = sqrt(0.25+ ir);        // Eqn 71
qD = −nq*( (4.0/15.0)*((3.0*pow( Xr,3) + 6.0*pow( Xr, 2)*Xf + 4.0*Xr*pow( Xf, 2)
     + 2.0*pow(Xf, 3))/(pow( (Xf+Xr), 2) ) ) −0.5);  // Eqn 72
qS = −nq*( (4.0/15.0)*((3.0*pow( Xf,3) + 6.0*pow( Xf, 2)*Xr + 4.0*Xf*pow( Xr, 2)
     + 2.0*pow(Xr, 3))/(pow( (Xf+Xr), 2) ) ) −0.5);  // Eqn 73
qI = −nq*( (4.0/3.0)*( (pow(Xf,2)+(Xf*Xr)+pow(Xr,2))/(Xf+Xr)) − 1.0);      // Eqn 74
if (LEVEL == 2)
  if (VGprime > 0)
       qB = (−Gammaa*sqrt(Vp+Phi_T+1e−6))*(1.0/$vt) − ( (nq−1.0)/nq)*qI;   // Eqn 75
  else
       qB = −VGprime/$vt;
 else
    if (VGprime > 0)
       qB = (−Gamma*sqrt(Vp+Phi_T+1e−6))*(1.0/$vt) − ( (nq−1.0)/nq)*qI;   // Eqn 75
  else
       qB = −VGprime/$vt;
//
if (LEVEL == 2)
    Beta = Beta0prime/(1.0 + (Cox/ (EO*epsilonsi))*$vt*abs(qB+eta*qI));      // Eqn 62
else
    Beta = Kp*(Weff/Leff)/(1+Theta*Vp);
//
Ispecific = 2.0*n*Beta*pow( $vt, 2);     // Eqn 65
//
if (LEVEL == 2)
  begin
    Ids = Ispecific*(iff−irprime);         // Eqn 66
    Vib = VD−VS−Ibn*2.0*Vdss;            // Eqn 67
    if ( Vib > 0.0)
     Idb = Ids*(Iba/Ibb_T)*Vib*exp( (−Ibb_T*Lc)/Vib);    // Eqn 68
    else
    Idb = 0.0;
  end
else
    Ids = Ispecific*(iff−ir);            // Eqn 66
//
Sthermal = fourkt*Beta*abs(qI);
gm = Beta*$vt*(sqrt( (4.0*iff/Ispecific) +1.0) − sqrt( (4.0*ir/Ispecific) + 1.0) );
Sflicker = (Kf*gm*gm)/(Np*Weff*Ns*Leff*Cox);
//
qb = con2*$vt*qB;
qg = con2*$vt*(−qI−qB);
qgso = Cgso*Weff*Np*(VG−VS);
qgdo = Cgdo*Weff*Np*(VG−VD);
```

154

```
qgbo = Cgbo*Leff*Np*VG;
// Drain and source diodes
if (StoDswap > 0.0)
    begin
        V1=p_n_MOS*V(Bulk, Drain_int);
        V2=p_n_MOS*V(Bulk, Source_int);
    end
else
    begin
        V2=p_n_MOS*V(Bulk, Drain_int);
        V1=p_n_MOS*V(Bulk, Source_int);
    end
Id1= (V1>-5.0*N*$vt) ? Area*Is_T2*(limexp( V1/(N*Vt_T2) )-1.0) : 0;
Qd1=(V1<Fc*Vj)? Tt*Id1+Area*(Cj0_T2*Vj_T2/(1-M))*(1-pow((1-V1/Vj_T2),(1-M))):0;
Id2=(V1<=-5.0*N*$vt) ? -Area*Is_T2 : 0;
Qd2=(V1>=Fc*Vj)? Tt*Id1+Area*Cj0_T2*(F1+(1/F2)*(F3*(V1-Fc*Vj_T2)+(M/(2.0*Vj_T2))
    *(V1*V1-Fc*Fc*Vj_T2*Vj_T2))):0;
Id3=(V1 == -Bv) ? -Ibv   : 0 ;
Id4=(V1<-Bv) ?-Area*Is_T2*(limexp(-(Bv+V1)/Vt_T2)-1.0+Bv/Vt_T2) : 0;
Ib_d = Id1+Id2+Id3+Id4;
Qd = Qd1+Qd2;
//
Is1= (V2>-5.0*N*$vt) ? Area*Is_T2*(limexp( V2/(N*Vt_T2) )-1.0) : 0;
Qs1=(V2<Fc*Vj)? Tt*Is1+Area*(Cj0_T2*Vj_T2/(1-M))*(1-pow((1-V2/Vj_T2),(1-M))):0;
Is2=(V2<=-5.0*N*$vt) ? -Area*Is_T2 : 0;
Qs2=(V2>=Fc*Vj)? Tt*Is1+Area*Cj0_T2*(F1+(1/F2)*(F3*(V2-Fc*Vj_T2)+(M/(2.0*Vj_T2))
    *(V2*V2-Fc*Fc*Vj_T2*Vj_T2))):0;
Is3=(V2 == -Bv) ? -Ibv   : 0 ;
Is4=(V2<-Bv) ?-Area*Is_T2*(limexp(-(Bv+V2)/Vt_T2)-1.0+Bv/Vt_T2) : 0;
Ib_s = Is1+Is2+Is3+Is4;
Qs = Qs1+Qs2;
// Current and noise contributions
if ( StoDswap > 0.0)
 begin
     if (RDeff > 0.0)
        I(Drain, Drain_int) <+ V(Drain, Drain_int)/RDeff;
     else
        I(Drain, Drain_int) <+ V(Drain, Drain_int)/1e-7;
     if (RSeff > 0.0)
        I(Source, Source_int) <+ V(Source, Source_int)/RSeff;
     else
        I(Source, Source_int) <+ V(Source, Source_int)/1e-7;
    I(Drain_int, Source_int) <+ p_n_MOS*Ids;
    if (LEVEL == 2)
        I(Drain_int, Bulk) <+ p_n_MOS*Idb;
    I(Gate, Drain_int) <+ p_n_MOS*0.5*ddt(qg);
    I(Gate, Source_int) <+ p_n_MOS*0.5*ddt(qg);
    I(Drain_int, Bulk) <+ p_n_MOS*0.5*ddt(qb);
    I(Source_int, Bulk) <+ p_n_MOS*0.5*ddt(qb);
    I(Gate, Source_int) <+ p_n_MOS*ddt(qgso);
    I(Gate, Drain_int)  <+ p_n_MOS*ddt(qgdo);
    I(Gate, Bulk)     <+ p_n_MOS*ddt(qgbo);
    I(Bulk, Drain_int) <+ p_n_MOS*Ib_d;
    I(Bulk, Drain_int) <+ p_n_MOS*ddt(Qd);
    I(Bulk, Source_int) <+ p_n_MOS*Ib_s;
    I(Bulk, Source_int) <+ p_n_MOS*ddt(Qs);
    I(Drain_int, Source_int) <+ white_noise(Sthermal,"thermal");
    I(Drain_int, Source_int) <+ flicker_noise(Sflicker, Af, "flicker");
    I(Drain, Drain_int) <+ white_noise(fourkt/RDeff, "thermal");
    I(Source, Source_int) <+ white_noise(fourkt/RSeff, "thermal");
end
else
```

```
begin
    if (RSeff > 0.0)
        I(Drain, Drain_int) <+ V(Drain, Drain_int)/RSeff;
    else
        I(Drain, Drain_int) <+ V(Drain, Drain_int)/1e-7;
    if (RDeff > 0.0)
        I(Source, Source_int) <+ V(Source, Source_int)/RDeff;
    else
        I(Source, Source_int) <+ V(Source, Source_int)/1e-7;
    I( Source_int, Drain_int) <+ p_n_MOS*Ids;
    if (LEVEL == 2)
        I(Source_int, Bulk) <+ p_n_MOS*Idb;
    I( Gate, Source_int) <+ p_n_MOS*0.5*ddt(qg);
    I( Gate, Drain_int) <+ p_n_MOS*0.5*ddt(qg);
    I( Source_int, Bulk) <+ p_n_MOS*0.5*ddt(qb);
    I( Drain_int, Bulk) <+ p_n_MOS*0.5*ddt(qb);
    I( Gate, Drain_int) <+ p_n_MOS*ddt(qgso);
    I( Gate, Source_int)  <+ p_n_MOS*ddt(qgdo);
    I( Gate, Bulk)    <+ p_n_MOS*ddt(qgbo);
    I( Bulk, Source_int) <+ p_n_MOS*Ib_d;
    I( Bulk, Source_int) <+ p_n_MOS*ddt(Qd);
    I( Bulk, Drain_int) <+ p_n_MOS*Ib_s;
    I( Bulk, Drain_int) <+ p_n_MOS*ddt(Qs);
    I( Source_int, Drain_int) <+ white_noise(Sthermal,"thermal");
    I( Source_int, Drain_int) <+ flicker_noise(Sflicker, Af, "flicker");
    I( Source_int, Source) <+ white_noise(fourkt/RDeff, "thermal");
    I( Drain_int, Drain) <+ white_noise(fourkt/RSeff, "thermal");
  end
end
endmodule
```

6.7.2 pMOS: EKV equation numbers are given on the right-hand side of code lines

```
//     Qucs EPFL-EKV 2.6 pMOS model:
//
//     The structure and theoretical background to the EKV 2.6
//     Verilog-a model is presented in the Qucs EPL-EKV 2.6 report.
//     Typical parameters are for 0.5um CMOS (C) EPLFL-LEG 1999.
//     Geometry range: Short channel: W>= 0.8um, L >= 0.5um
//                     Long channel:  W >= 2um,  L >= 2um
//     Voltage range:  |Vgb| < 3.3V, |Vdb| < 3.3V, |Vsb| < 2V
//
//     This is free software; you can redistribute it and/or modify
//     it under the terms of the GNU General Public License as published by
//     the Free Software Foundation; either version 2, or (at your option)
//     any later version.
//
//     Copyright (C), Mike Brinson, mbrin72043@yahoo.co.uk, May 2008.
//
'include "disciplines.vams"
'include "constants.vams"

//
 module EKV26pMOS (Drain, Gate, Source, Bulk);
 inout Drain, Gate, Source, Bulk;
electrical Drain, Gate, Source, Bulk;
// Internal nodes
electrical Drain_int, Source_int;
'define attr(txt) (*txt*)
```

```verilog
// Device dimension parameters
 parameter real LEVEL = 1 from [1 : 2]
        'attr(info="long_=_1,_short_=_2");
 parameter real L = 0.5e-6 from [0.0 : inf]
        'attr(info="length_parameter" unit = "m" );
 parameter real W = 10e-6 from [0.0 : inf]
        'attr(info="Width_parameter" unit = "m");
 parameter real Np = 1.0 from [1.0 : inf]
        'attr(info="parallel_multiple_device_number");
 parameter real Ns = 1.0 from [1.0 : inf]
        'attr(info="series_multiple_device_number");
// Process parameters
 parameter real Cox = 3.45e-3 from [0 : inf]
        'attr(info="gate_oxide_capacitance_per_unit_area" unit = "F/m**2" );
 parameter real Xj = 0.15e-6 from [0.01e-6 : 1.0e-6]
        'attr(info="metallurgical_junction_depth" unit = "m");
 parameter real Dw = -0.03e-6 from [-inf : 0.0]
        'attr(info="channel_width_correction" unit = "m");
 parameter real Dl = -0.05e-6 from [-inf : 0.0]
        'attr(info="channel_length_correction" unit = "m");
// Basic intrinsic model parameters
 parameter real Vto = -0.55 from [-inf : -1e-6]
        'attr(info="long_channel_threshold_voltage" unit="V" );
 parameter real Gamma = 0.69 from [0.0 : 2.0]
        'attr(info="body_effect_parameter" unit="V**(1/2)");
 parameter real Phi = 0.87 from [0.3 : 2.0]
        'attr(info="bulk_Fermi_potential" unit="V");
 parameter real Kp = 35e-6 from [10e-6 : inf]
        'attr(info="transconductance_parameter" unit = "A/V**2");
 parameter real Theta = 50e-3 from [0.0 : inf]
        'attr(info="mobility_reduction_coefficient" unit = "1/V");
 parameter real EO = 51.0e6 from [1.0e6 : inf]
        'attr(info="mobility_coefficient" unit="V/m");
 parameter real Ucrit = 18.0e6 from [2.0e6 : 25.0e6]
        'attr(info="longitudinal_critical_field" unit="V/m");
// Channel length and charge sharing parameters
 parameter real Lambda = 1.1 from [0.1 : inf]
        'attr(info="depletion_length_coefficient");
 parameter real Weta = 0.0   from [0.0 : inf]
        'attr(info="narrow-channel_effect_coefficient");
 parameter real Leta = 0.45 from [0.0 : inf]
        'attr(info="longitudinal_critical_field");
// Reverse short channel effect parameters
 parameter real Q0 = 200e-6 from [0.0 : inf]
        'attr(info="reverse_short_channel_charge_density" unit="A*s/m**2");
 parameter real Lk = 0.6e-6 from [0.0 : inf]
        'attr(info="characteristic_length" unit="m");
// Intrinsic model temperature parameters
 parameter real Tcv = -1.4e-3
        'attr(info="threshold_voltage_temperature_coefficient" unit="V/K");
 parameter real Bex = -1.4
        'attr(info="mobility_temperature_coefficient");
 parameter real Ucex = 2.0
        'attr(info="Longitudinal_critical_field_temperature_exponent");
 parameter real Ibbt = 0.0
        'attr(info="Ibb_temperature_coefficient" unit = "1/K");
// Series resistance calculation parameters
 parameter real Hdif = 0.9e-6 from [0.0 :inf]
        'attr(info="heavily_doped_diffusion_length" unit = "m");
 parameter real Rsh = 990.0 from [0.0 : inf]
        'attr(info="drain/source_diffusion_sheet_resistance" unit="Ohm/square");
 parameter real Rsc = 0.0 from [0.0 : inf]
```

```
                'attr(info="source_contact_resistance" unit="Ohm");
    parameter real Rdc = 0.0 from [0.0 : inf]
                'attr(info="drain_contact_resistance" unit="Ohm");
// Gate overlap capacitances
    parameter real Cgso = 1.5e-10 from [0.0 : inf]
                'attr(info="gate_to_source_overlap_capacitance" unit = "F/m");
    parameter real Cgdo = 1.5e-10 from [0.0 : inf]
                'attr(info="gate_to_drain_overlap_capacitance"  unit= "F/m");
    parameter real Cgbo = 4.0e-10 from [0.0 : inf]
                'attr(info="gate_to_bulk_overlap_capacitance"   unit= "F/m");
// Impact ionization related parameters
    parameter real Iba = 0.0 from [0.0 : inf]
                'attr(info="first_impact_ionization_coefficient" unit = "1/m");
    parameter real Ibb = 3.0e8 from [1.0e8 : inf]
                'attr(info="second_impact_ionization_coefficient" unit="V/m");
    parameter real Ibn = 1.0 from [0.1 : inf]
                'attr(info="saturation_voltage_factor_for_impact_ionization");
// Flicker noise parameters
    parameter real Kf = 1.0e-28 from [0.0 : inf]
                'attr(info="flicker_noise_coefficient");
    parameter real Af = 1.0 from [0.0 : inf]
                'attr(info="flicker_noise_exponent");
// Matching parameters
    parameter real Avto = 0.0 from [0.0 : inf]
                'attr(info="area_related_theshold_voltage_mismatch_parameter" unit = "V*m");
    parameter real Akp = 0.0 from [0.0 : inf]
                'attr(info="area_related_gain_mismatch_parameter" unit="m");
    parameter real Agamma = 0.0 from [0.0 : inf]
                'attr(info="area_related_body_effect_mismatch_parameter" unit="sqrt(V)*m");
// Diode parameters
    parameter real N=1.0 from [1e-6:inf]
                'attr(info="emission_coefficient");
    parameter real Is=1e-14 from [1e-20:inf]
                'attr(info="saturation_current" unit="A");
    parameter real Bv=100 from [1e-6:inf]
                'attr(info="reverse_breakdown_voltage" unit="V");
    parameter real Ibv=1e-3 from [1e-6:inf]
                'attr(info="current_at_reverse_breakdown_voltage" unit="A");
    parameter real Vj=1.0 from [1e-6:inf]
                'attr(info="junction_potential" unit="V");
    parameter real Cj0=300e-15 from [0:inf]
                'attr(info="zero-bias_junction_capacitance" unit="F");
    parameter real M=0.5 from [1e-6:inf]
                'attr(info="grading_coefficient");
    parameter real Area=1.0 from [1e-3:inf]
                'attr(info="diode_relative_area");
    parameter real Fc=0.5 from [1e-6:inf]
                'attr(info="forward-bias_depletion_capcitance_coefficient");
    parameter real Tt=0.1e-9 from [1e-20:inf]
                'attr(info="transit_time" unit="s");
    parameter real Xti=3.0 from [1e-6:inf]
                'attr(info="saturation_current_temperature_exponent");
// Temperature parameters
    parameter real Tnom = 26.85
                'attr(info="parameter_measurement_temperature" unit = "Celsius");
// Local variables
real epsilonsi, epsilonox, Tnomk, T2, Tratio, Vto_T, Ucrit_T, Egnom, Eg, Phi_T;
real Weff, Leff, RDeff, RSeff, con1, con2, Vtoa, Kpa,Kpa_T,Gammaa, C_epsilon, xi;
real nnn, deltaV_RSCE, Vg, Vs, Vd, Vgs, Vgd, Vds, Vdso2, VG, VS, VD;
real VGprime, VP0, VSprime, VDprime, Gamma0, Gammaprime, Vp;
real n, X1, iff, X2, ir, Vc, Vdss, Vdssprime, deltaV, Vip;
real Lc, DeltaL, Lprime, Lmin, Leq, X3, irprime, Beta0, eta;
```

```
real Qb0, Beta0prime, nq, Xf, Xr, qD, qS, qI, qB, Beta, Ispecific, Ids, Vib, Idb, Ibb_T;
real A, B, Vt_T2, Eg_T1, Eg_T2, Vj_T2, Cj0_T2, F1, F2, F3, Is_T2;
real Id1, Id2, Id3, Id4, Is1, Is2, Is3, Is4, V1, V2, Ib_d, Ib_s, Qd, Qs, Qd1, Qd2, Qs1, Qs2;
real qb, qg, qgso, qgdo, qgbo, fourkt, Sthermal, gm, Sflicker, StoDswap, p_n_MOS;
//
analog begin
// Equation initialization
p_n_MOS = -1.0; // pMOS
A=7.02e-4;
B=1108.0;
epsilonsi = 1.0359e-10;        // Eqn 4
epsilonox = 3.453143e-11;     // Eqn 5
Tnomk = Tnom+273.15;                     // Eqn 6
T2=$temperature;
Tratio = T2/Tnomk;
Vto_T = -Vto+Tcv*(T2-Tnomk); // Signs of Vto and Tcv changed for pMOS
Egnom = 1.16-0.000702*Tnomk*Tnomk/(Tnomk+1108);
Eg =      1.16-0.000702*T2*T2/(T2+1108);
Phi_T = Phi*Tratio - 3.0*$vt*ln(Tratio)-Egnom*Tratio+Eg;
Ibb_T = Ibb*(1.0+Ibbt*(T2 -Tnomk));
Weff = W + Dw;    // Eqn 25
Leff = L + Dl;    // Eqn 26
RDeff = ( (Hdif*Rsh)/Weff)/Np + Rdc;
RSeff = ( (Hdif*Rsh)/Weff)/Np + Rsc;
con1 = sqrt(Np*Weff*Ns*Leff);
Vt_T2='P_K*T2/'P_Q;
Eg_T1=Eg-A*Tnomk*Tnomk/(B+Tnomk);
Eg_T2=Eg-A*T2*T2/(B+T2);
Vj_T2=(T2/Tnomk)*Vj-(2*Vt_T2)*ln(pow((T2/Tnomk),1.5)) -((T2/Tnomk)*Eg_T1-Eg_T2);
Cj0_T2=Cj0*(1+M*(400e-6*(T2-Tnomk)-(Vj_T2-Vj)/Vj));
F1=(Vj/(1-M))*(1-pow((1-Fc),(1-M)));
F2=pow((1-Fc), (1+M));
F3=1-Fc*(1+M);
Is_T2=Is*pow( (T2/Tnomk), (Xti/N))*limexp((-Eg_T1/Vt_T2)*(1-T2/Tnomk));
con2 = (Cox*Ns*Np*Weff*Leff);
fourkt = 4.0*'P_K*T2;
//
if (LEVEL == 2)
 begin
  Ucrit_T = Ucrit*pow(Tratio, Ucex);
  Vtoa = Vto+Avto/con1;                    // Eqn 27
  Kpa = Kp*(1.0+Akp/con1);                 // Eqn 28
  Kpa_T = Kpa*pow( Tratio, Bex);        // Eqn 18
  Gammaa = Gamma+Agamma/con1;         // Eqn 29
  C_epsilon = 4.0*pow(22e-3, 2);      // Eqn 30
  xi = 0.028*(10.0*(Leff/Lk)-1.0);    // Eqn 31
  nnn = 1.0+0.5*(xi+ sqrt(pow(xi,2) + C_epsilon));
  deltaV_RSCE = (2.0*Q0/Cox)*(1.0/pow(nnn,2)); // Eqn 32
 end
//
// Model branch and node voltages
//

Vg = p_n_MOS*V(Gate, Bulk);
Vs = p_n_MOS*V(Source, Bulk);
Vd = p_n_MOS*V(Drain, Bulk);
VG=Vg;        // Eqn 22
if ( (Vd-Vs) >= 0.0)
        begin
           StoDswap = 1.0;
           VS=Vs;    // Eqn 23
           VD=Vd;    // Eqn 24
```

```verilog
            end
else
            begin
               StoDswap = −1.0;
               VD=Vs;
               VS=Vd;
            end
if (LEVEL == 2)
     VGprime=VG−Vto_T−deltaV_RSCE+Phi_T+Gamma∗sqrt(Phi_T);   // Eqn 33   nMOS equation
else
     VGprime=Vg−Vto_T+Phi_T+Gamma∗sqrt(Phi_T);

if (LEVEL == 2)
 begin
  if (VGprime > 0)
        VP0=VGprime−Phi_T−Gammaa∗(sqrt(VGprime+(Gammaa/2.0)∗(Gammaa/2.0))−(Gammaa/2.0));
// Eqn 34
   else
        VP0 = −Phi_T;
  VSprime=0.5∗(VS+Phi_T+sqrt(pow( (VS+Phi_T),2) + pow( (4.0∗$vt),2)));   // Eqn 35
  VDprime=0.5∗(VD+Phi_T+sqrt(pow( (VD+Phi_T),2) + pow( (4.0∗$vt),2)));   // Eqn 35
  Gamma0=Gammaa−(epsilonsi/Cox)∗((Leta/Leff)∗(sqrt(VSprime)+sqrt(VDprime))
           −(3.0∗Weta/Weff)∗sqrt(VP0+Phi_T));   // Eqn 36
  Gammaprime = 0.5∗(Gamma0+sqrt( pow(Gamma0,2) +0.1∗$vt ));   // Eqn 37
  if (VGprime > 0.0 )
          Vp = VGprime−Phi_T−Gammaprime∗(sqrt(VGprime+(Gammaprime/2.0)∗(Gammaprime/2.0))
               − (Gammaprime/2.0));  // Eqn 38
  else
        Vp = −Phi_T;
  n = 1.0 +Gammaa/(2.0∗sqrt(Vp+Phi_T+4.0∗$vt));   // Eqn 39
 end
else
 begin
  if (VGprime > 0)
        Vp=VGprime−Phi_T−Gamma∗(sqrt(VGprime+(Gamma/2.0)∗(Gamma/2.0))−(Gamma/2.0));
// Eqn 34
   else
        Vp = −Phi_T;
  n = 1.0 +Gamma/(2.0∗sqrt(Vp+Phi_T+4.0∗$vt));   // Eqn 39
end
//
X1 = (Vp−VS)/$vt;
iff = ln(1.0+limexp(X1/2.0))∗ln(1.0+limexp(X1/2.0));   // Eqn 44
X2 = (Vp−VD)/$vt;
ir = ln(1.0+limexp(X2/2.0))∗ln(1.0+limexp(X2/2.0));    // Eqn 57
//
if (LEVEL == 2)
 begin
  Vc = Ucrit_T∗Ns∗Leff;   // Eqn 45
  Vdss = Vc∗(sqrt( 0.25 + (($vt/(Vc))∗sqrt(iff)))−0.5);   // Eqn 46;
  Vdssprime = Vc∗(sqrt( 0.25 + ($vt/Vc)∗(sqrt(iff)−0.75∗ln(iff))) − 0.5)
            +$vt∗(ln(Vc/(2.0∗$vt)) − 0.6 ); // Eqn 47
  if (Lambda∗(sqrt(iff) > (Vdss/$vt) ) )
        deltaV = 4.0∗$vt∗sqrt(Lambda∗(sqrt(iff) −(Vdss/$vt)) + (1.0/64.0) );   // Eqn 48
  else     deltaV = 1.0/64.0;
  Vdso2 = (VD−VS)/2.0;            // Eqn 49
  Vip = sqrt( pow(Vdss, 2) + pow( deltaV,2)) − sqrt( pow( (Vdso2 − Vdss), 2)
       + pow (deltaV, 2));   // Eqn 50
  Lc = sqrt( (epsilonsi/Cox)∗Xj);   // Eqn 51
  DeltaL = Lambda∗Lc∗ln(1.0+((Vdso2−Vip)/(Lc∗Ucrit_T)));   // Eqn 52
  Lprime = Ns∗Leff − DeltaL + ( (Vdso2+Vip)/Ucrit_T); // Eqn 53
  Lmin = Ns∗Leff/10.0;   // Eqn 54
```

```
   Leq = 0.5*(Lprime + sqrt( pow(Lprime, 2) + pow(Lmin, 2)));     // Eqn 55
   X3 = (Vp-Vdso2-VS-sqrt( pow(Vdssprime, 2) + pow( deltaV, 2))
        + sqrt( pow( (Vdso2-Vdssprime), 2) + pow(deltaV,2)))/$vt;
   irprime = ln(1.0+limexp(X3/2.0))*ln(1.0+limexp(X3/2.0));    // Eqn 56
   Beta0 = Kpa_T*(Np*Weff/Leq);        // Eqn 58
   eta = 0.3333333;    // Eqn 59 - pMOS
   Qb0 = Gammaa*sqrt(Phi_T);    // Eqn 60;
   Beta0prime = Beta0*(1.0 +(Cox/(EO*epsilonsi))*Qb0);    // Eqn 61
   nq = 1.0 +Gammaa/(2.0*sqrt(Vp+Phi_T+1e-6));             // Eqn 69
  end
 else
   nq = 1.0 +Gamma/(2.0*sqrt(Vp+Phi_T+1e-6));             // Eqn 69
//
Xf = sqrt(0.25+ iff);      // Eqn 70
Xr = sqrt(0.25+ ir);       // Eqn 71
qD = -nq*( (4.0/15.0)*((3.0*pow( Xr,3) + 6.0*pow( Xr, 2)*Xf + 4.0*Xr*pow( Xf, 2)
        + 2.0*pow(Xf, 3))/(pow( (Xf+Xr), 2) ) ) -0.5);  // Eqn 72
qS = -nq*( (4.0/15.0)*((3.0*pow( Xf,3) + 6.0*pow( Xf, 2)*Xr + 4.0*Xf*pow( Xr, 2)
        + 2.0*pow(Xr, 3))/(pow( (Xf+Xr), 2) ) ) -0.5);  // Eqn 73
qI = -nq*( (4.0/3.0)*( (pow(Xf,2)+(Xf*Xr)+pow(Xr,2))/(Xf+Xr)) - 1.0);     // Eqn 74
if (LEVEL == 2)
   if (VGprime > 0)
        qB = (-Gammaa*sqrt(Vp+Phi_T+1e-6))*(1.0/$vt) - ( (nq-1.0)/nq)*qI;   // Eqn 75
   else
        qB = -VGprime/$vt;
else
     if (VGprime > 0)
        qB = (-Gamma*sqrt(Vp+Phi_T+1e-6))*(1.0/$vt) - ( (nq-1.0)/nq)*qI;   // Eqn 75
   else
        qB = -VGprime/$vt;
//
if (LEVEL == 2)
   Beta = Beta0prime/(1.0 + (Cox/ (EO*epsilonsi))*$vt*abs(qB+eta*qI));       // Eqn 62
else
   Beta = Kp*(Weff/Leff)/(1+Theta*Vp);
//
Ispecific = 2.0*n*Beta*pow( $vt, 2);      // Eqn 65
//
if (LEVEL == 2)
  begin
    Ids = Ispecific*(iff-irprime);           // Eqn 66
    Vib = VD-VS-Ibn*2.0*Vdss;                // Eqn 67
    if ( Vib > 0.0)
        Idb = Ids*(Iba/Ibb_T)*Vib*exp( (-Ibb_T*Lc)/Vib);    // Eqn 68
    else
        Idb = 0.0;
  end
else
    Ids = Ispecific*(iff-ir);           // Eqn 66
//
Sthermal = fourkt*Beta*abs(qI);
gm = Beta*$vt*(sqrt( (4.0*iff/Ispecific) +1.0) - sqrt( (4.0*ir/Ispecific) + 1.0) );
Sflicker = (Kf*gm*gm)/(Np*Weff*Ns*Leff*Cox);
//
qb = con2*$vt*qB;
qg = con2*$vt*(-qI-qB);
qgso = Cgso*Weff*Np*(VG-VS);
qgdo = Cgdo*Weff*Np*(VG-VD);
qgbo = Cgbo*Leff*Np*VG;
// Drain and source diodes
if (StoDswap > 0.0)
     begin
```

```verilog
            V1=p_n_MOS*V(Bulk, Drain_int);
            V2=p_n_MOS*V(Bulk, Source_int);
        end
else
        begin
            V2=p_n_MOS*V(Bulk, Drain_int);
            V1=p_n_MOS*V(Bulk, Source_int);
        end
Id1= (V1>-5.0*N*$vt) ? Area*Is_T2*(limexp( V1/(N*Vt_T2) )-1.0) : 0;
Qd1=(V1<Fc*Vj)? Tt*Id1+Area*(Cj0_T2*Vj_T2/(1-M))*(1-pow((1-V1/Vj_T2),(1-M))):0;
Id2=(V1<=-5.0*N*$vt) ? -Area*Is_T2 : 0;
Qd2=(V1>=Fc*Vj)? Tt*Id1+Area*Cj0_T2*(F1+(1/F2)*(F3*(V1-Fc*Vj_T2)+(M/(2.0*Vj_T2))
    *(V1*V1-Fc*Fc*Vj_T2*Vj_T2))):0;
Id3=(V1 == -Bv) ? -Ibv : 0 ;
Id4=(V1<-Bv) ?-Area*Is_T2*(limexp(-(Bv+V1)/Vt_T2)-1.0+Bv/Vt_T2) : 0;
Ib_d = Id1+Id2+Id3+Id4;
Qd = Qd1+Qd2;
//
Is1= (V2>-5.0*N*$vt) ? Area*Is_T2*(limexp( V2/(N*Vt_T2) )-1.0) : 0;
Qs1=(V2<Fc*Vj)? Tt*Is1+Area*(Cj0_T2*Vj_T2/(1-M))*(1-pow((1-V2/Vj_T2),(1-M))):0;
Is2=(V2<=-5.0*N*$vt) ? -Area*Is_T2 : 0;
Qs2=(V2>=Fc*Vj)? Tt*Is1+Area*Cj0_T2*(F1+(1/F2)*(F3*(V2-Fc*Vj_T2)+(M/(2.0*Vj_T2))
    *(V2*V2-Fc*Fc*Vj_T2*Vj_T2))):0;
Is3=(V2 == -Bv) ? -Ibv : 0 ;
Is4=(V2<-Bv) ?-Area*Is_T2*(limexp(-(Bv+V2)/Vt_T2)-1.0+Bv/Vt_T2) : 0;
Ib_s = Is1+Is2+Is3+Is4;
Qs = Qs1+Qs2;
// Current contributions
if ( StoDswap > 0.0)
 begin
        if (RDeff > 0.0)
            I(Drain, Drain_int) <+ V(Drain, Drain_int)/RDeff;
        else
            I(Drain, Drain_int) <+ V(Drain, Drain_int)/1e-7;
        if (RSeff > 0.0)
            I(Source, Source_int) <+ V(Source, Source_int)/RSeff;
        else
            I(Source, Source_int) <+ V(Source, Source_int)/1e-7;
     I(Drain_int, Source_int) <+ p_n_MOS*Ids;
        if (LEVEL == 2)
          I(Drain_int, Bulk) <+ p_n_MOS*Idb;
        I(Gate, Drain_int) <+ p_n_MOS*0.5*ddt(qg);
        I(Gate, Source_int) <+ p_n_MOS*0.5*ddt(qg);
        I(Drain_int, Bulk) <+ p_n_MOS*0.5*ddt(qb);
        I(Source_int, Bulk) <+ p_n_MOS*0.5*ddt(qb);
        I(Gate, Source_int) <+ p_n_MOS*ddt(qgso);
        I(Gate, Drain_int)  <+ p_n_MOS*ddt(qgdo);
        I(Gate, Bulk)    <+ p_n_MOS*ddt(qgbo);
        I(Bulk, Drain_int) <+ p_n_MOS*Ib_d;
        I(Bulk, Drain_int) <+ p_n_MOS*ddt(Qd);
        I(Bulk, Source_int) <+ p_n_MOS*Ib_s;
        I(Bulk, Source_int) <+ p_n_MOS*ddt(Qs);
        I(Drain_int, Source_int) <+ white_noise(Sthermal,"thermal");
        I(Drain_int, Source_int) <+ flicker_noise(Sflicker, Af, "flicker");
        I(Drain, Drain_int) <+ white_noise(fourkt/RDeff, "thermal");
        I(Source, Source_int) <+ white_noise(fourkt/RSeff, "thermal");
end
else
 begin
        if (RSeff > 0.0)
            I(Drain, Drain_int) <+ V(Drain, Drain_int)/RSeff;
        else
```

```verilog
            I(Drain, Drain_int) <+ V(Drain, Drain_int)/1e−7;
        if (RDeff > 0.0)
            I(Source, Source_int) <+ V(Source, Source_int)/RDeff;
        else
            I(Source, Source_int) <+ V(Source, Source_int)/1e−7;
    I( Source_int , Drain_int ) <+ p_n_MOS∗Ids ;
    if (LEVEL == 2)
        I(Source_int , Bulk) <+ p_n_MOS∗Idb ;
    I( Gate , Source_int) <+ p_n_MOS∗0.5∗ddt(qg );
    I( Gate , Drain_int ) <+ p_n_MOS∗0.5∗ddt(qg );
    I( Source_int , Bulk) <+ p_n_MOS∗0.5∗ddt(qb );
    I( Drain_int , Bulk) <+ p_n_MOS∗0.5∗ddt(qb );
    I( Gate , Drain_int ) <+ p_n_MOS∗ddt(qgso );
    I( Gate , Source_int)  <+ p_n_MOS∗ddt(qgdo );
    I( Gate , Bulk)    <+ p_n_MOS∗ddt(qgbo );
    I( Bulk , Source_int) <+ p_n_MOS∗Ib_d ;
    I( Bulk , Source_int) <+ p_n_MOS∗ddt(Qd);
    I( Bulk , Drain_int ) <+ p_n_MOS∗Ib_s ;
    I( Bulk , Drain_int ) <+ p_n_MOS∗ddt(Qs);
    I( Source_int , Drain_int) <+ white_noise(Sthermal ,"thermal");
    I( Source_int , Drain_int) <+ flicker_noise(Sflicker , Af, "flicker");
    I( Source_int , Source) <+ white_noise(fourkt/RDeff , "thermal");
    I( Drain_int , Drain) <+ white_noise(fourkt/RSeff , "thermal");
 end
end
endmodule
```

6.8 Update number one: September 2008

The first version of the Qucs EPFL-EKV v2.6 model provided Qucs users with reasonably complete long and short channel models for nMOS and pMOS devices. In no respect were these models optimized for minimum simulation run time or were they flexible enough to allow users to select the style of charge partitioning employed by the EKV model. Recent work on the Qucs implementation of the EKV v2.6 MOSFET model and the Qucs ADMS/XML interface has resulted in a significant reduction in simulation run time overhead, particularly in transient and small signal analysis. The addition of a SPICE BSIM style partition parameter Xpart to the Qucs version of the EKV v2.6 model now allows users to set the style of charge partitioning employed by the EKV model. These notes explain the function of the first EKV v2.6 update and introduce a series of test simulations that demonstrate the effects these changes have on the operation of the Qucs port of the EKV V2.6 model.

6.8.1 Model initialisation

Readers who have looked through the EKV v2.6 Verilog-A code listed in the previous sections of these notes will probably have been struck by the quantity of calculations involved each time the code is evaluated during simulation. In the case of transient analysis it is calculated at least once per time step, often resulting in many thousands of passes through the code. The more MOS devices included in a circuit the greater the time overhead becomes. Obviously, a sensible approach would be to minimize the amount of calculation by only evaluating once those parts of the EKV model equations which result in constant values during simulation. The Verilog-A hardware description language provides a model initialisation feature which selects those parts of a device model code which are to be evaluated prior to the start of a simulation. The resulting calculated variables are then available for use by other sections of the Verilog-A model code during simulation. In transient and small signal analysis this is particularly important as it significantly reduces simulation calculation time. Verilog-A employs the "at" (@(`initial_step`) or @(`initial_model`)) language construction coupled with a begin ... end block to signify the Verilog-A code that is to be evaluated only at model initialisation. Although this technique does greatly improve model simulation speed it does imply significantly more work for the model developer in that the Verilog-A device code has to be split into initialisation and dynamic simulation sections. Readers interested in the detail of how this split can be achieved should compare the latest EKV v2.6 Verilog-A CVS code given at the Qucs Web site with that presented in previous sections of these notes.

6.8.2 Charge partitioning

The MOSFT is a four terminal device with a dynamic performance that requires accurate calculation of the charge at each terminal. Previous notes indicated that the intrinsic channel charge equals the sum of the drain and source charges. However, the exact proportion of intrinsic channel charge that belongs to the drain or to the source is often not known. The assignment of the proportion of the channel charge to the drain and source charges is called charge partitioning. The first release of the Qucs EKV v2.6 model used the 50/50 partitioning scheme where 50% of the channel charge is arbitrarily assigned to both drain and source. It's interesting to note that this partitioning scheme has no physical basis but depends entirely on convenience. A second partitioning scheme, called the 40/60 partitioning, does however, have a strong physical basis[9]. Yet a third charge partitioning is often employed for digital circuit simulation; this is known as the 0/100 partition. The second release of the Qucs EPFL-EKV v2.6 model includes an extra parameter called Xpart which allows users to set the partitioning scheme for dynamic simulation calculations. Xpart default is set at 0.4 which corresponds to the 40/60 partitioning scheme. Figure 6.10 illustrates a test circuit for determining the S-Parameters of an nMOS device connected as a capacitance. Both the device capacitance and associated series resistance can be extracted from S[1,1]. Qucs equation block Eqn1 gives the equations for extracting these properties. Other equations in Eqn1 show how the extracted capacitance can be represented as a ratio of the basic parallel capacitance given by

$$C\_parallel\_plate = W \cdot L \cdot Cox. \tag{6.30}$$

6.8.3 Modelling EKV v2.6 charge partitioning using Qucs EDD

Complex simulation results like those shown in Fig 6.10 suggest the question "How do we check the accuracy of the model being simulated?". One possible approach is to develop a second model of the same device based on the same physical principles and equations but using a different approach like the Qucs EDD/subcircuit modelling route shown in Fig. 6.11. It is an EDD/subcircuit model of a long channel EKV v2.6 nMOS device which includes charge partitioning. Figure 6.12 illustrated the same test circuit as Fig. 6.10 and the extracted capacitance and resistance values for the EDD model of the long channel nMOS device. A number of features observed from Fig. 6.10 and Fig. 6.11 are worth commenting on; firstly that good agreement is recorded between the two sets of results, secondly that

---

[9]William Liu, MOSFET models for SPICE simulation, including BSIM3v3 and BSIM4, 2001, Wiley-Interscience publications, ISBN 0-471-39697-4.

number	C_parallel_plate
1	6.9e-14

S parameter simulation

SP1
Type=const
Values=[1MHz]

Parameter sweep

SW1
Sim=SP1
Type=lin
Param=Vgs
Start=-2
Stop=2
Points=81

P1
Num=1
Z=50 Ohm

X1

R1
R=1

V1
U=Vgs

Is

EKV26nMOS1
L=1e-6
W=20e-6
Dw=-0.02e-6
Dl=-0.05e-6
Cgso=1.5e-10
Cgdo=1.5e-10
Cgbo=4.0e-10
Xpart=0.4

Equation

Eqn1
PL_Cap=PlotVs(Cap, Vgs)
y=stoy(S)
Omega=2*pi*frequency
Cap=imag(y[1,1])/Omega
Rin=real(y[1,1])/(imag(y[1,1])*imag(y[1,1]))
L=1e-6
W=20e-6
Cox=3.45e-3
C_parallel_plate=W*L*Cox
PL_Cap_ratio=PlotVs(Cap/C_parallel_plate, Vgs)

Vgs	frequency	Rin
-0.15	1e6	5.18
-0.1	1e6	5.25
-0.05	1e6	5.31
0	1e6	5.37
0.05	1e6	5.43
0.1	1e6	5.49

Versus.0001: 0
PL_Cap: 2.76e-14

Versus.0001: -1.5
PL_Cap: 7.19e-14

Versus.0001: 1.5
PL_Cap: 7.18e-14

Versus.0002: 0
PL_Cap_ratio: 0.401

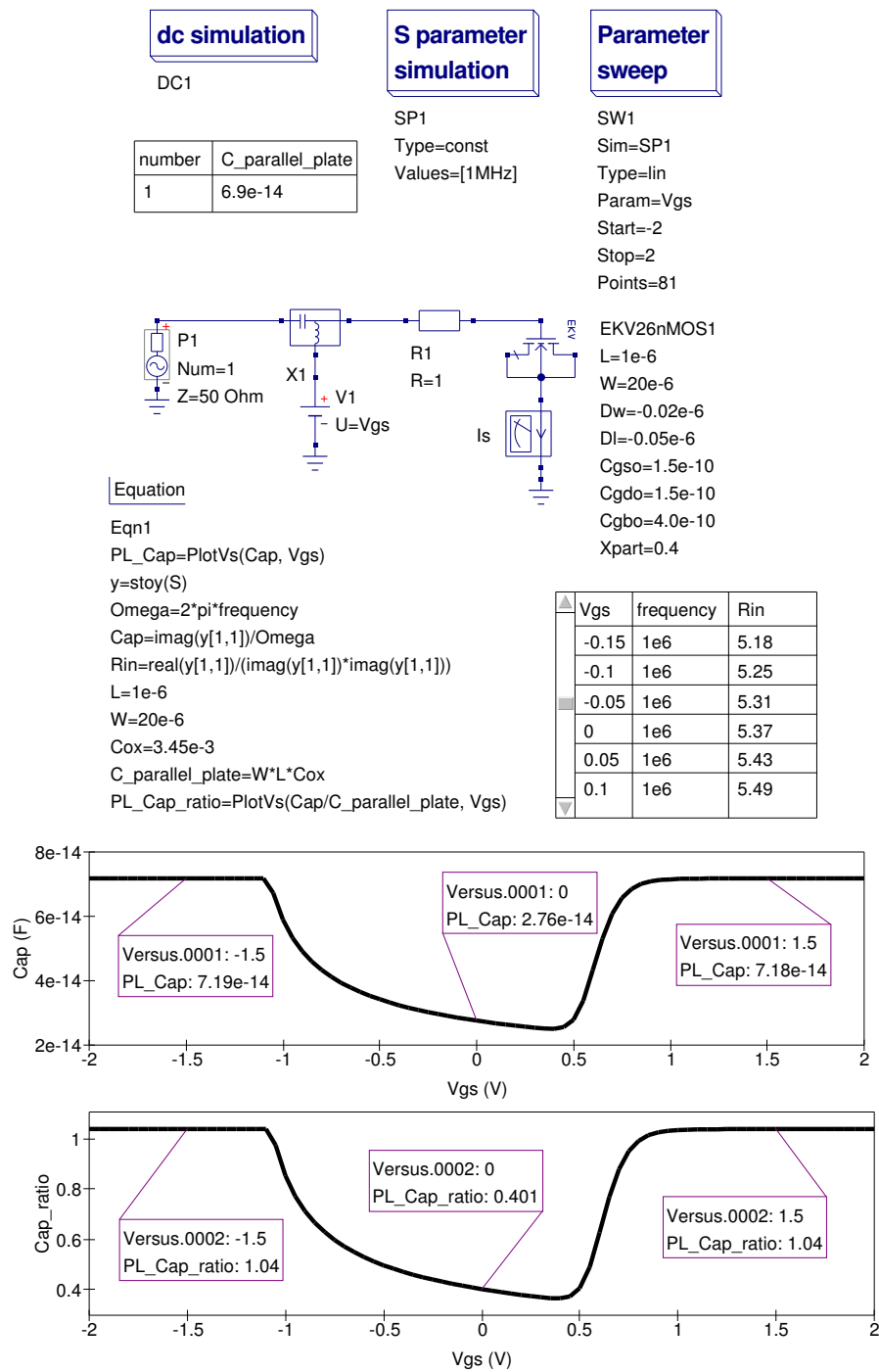Versus.0002: -1.5
PL_Cap_ratio: 1.04

Versus.0002: 1.5
PL_Cap_ratio: 1.04

Figure 6.10: Test circuit for simulating EKV v2.6 charge partitioning effects: Xpart
= 0.4 or QD/QS = 40/60

the Verilog-A model includes both overlap capacitance and drain and gate source resistances. Hence the slight difference in the capacitance ratio and the recorded values of Rin above one Ohm for the Verilog-A model.
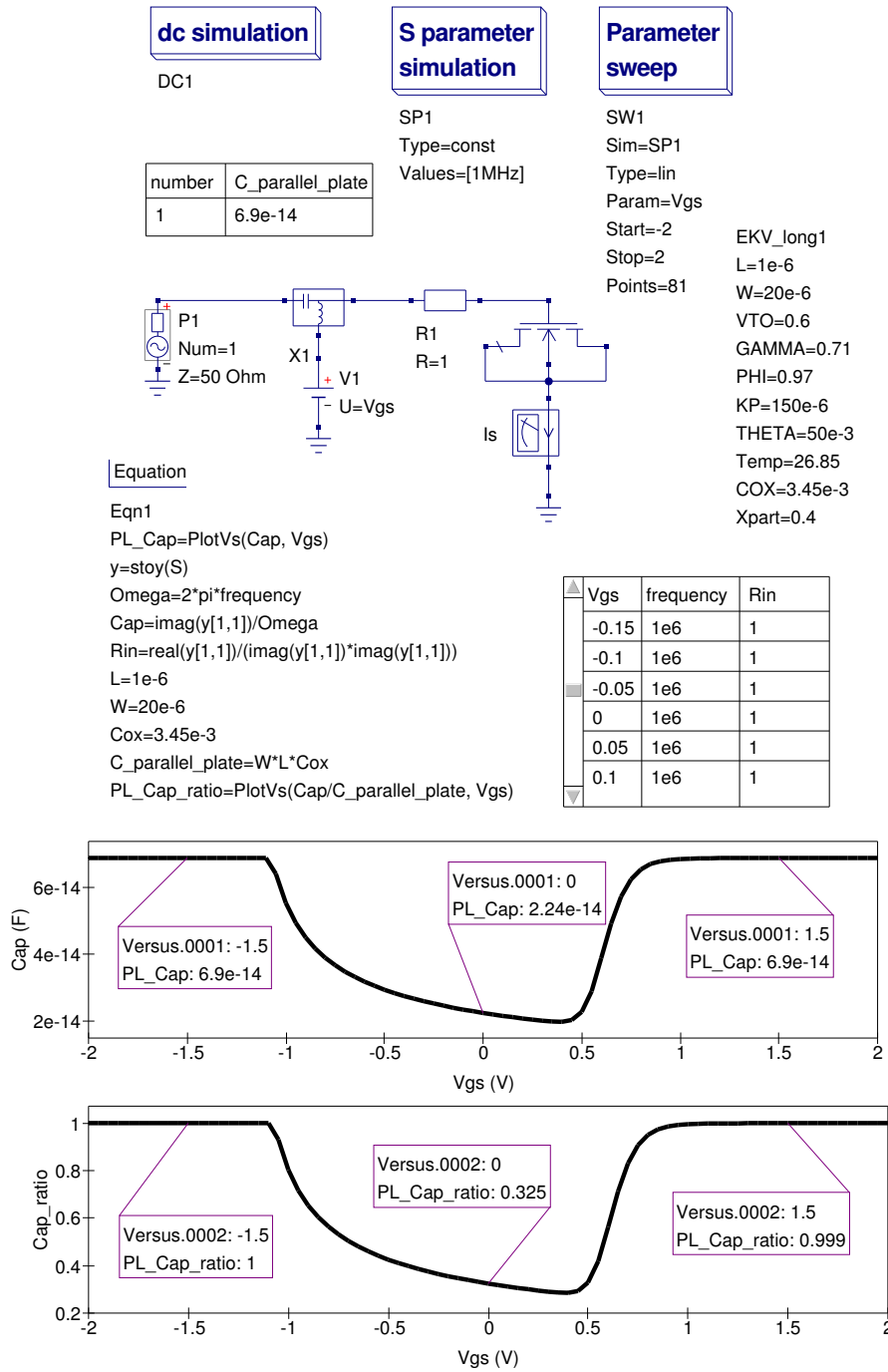
Figure 6.11: Qucs EDD model for a EKV v2.6 long channel nMOS device with charge partitioning

dc simulation

DC1

number	C_parallel_plate
1	6.9e-14

S parameter simulation

SP1
Type=const
Values=[1MHz]

Parameter sweep

SW1
Sim=SP1
Type=lin
Param=Vgs
Start=-2
Stop=2
Points=81

EKV_long1
L=1e-6
W=20e-6
VTO=0.6
GAMMA=0.71
PHI=0.97
KP=150e-6
THETA=50e-3
Temp=26.85
COX=3.45e-3
Xpart=0.4

P1
Num=1
Z=50 Ohm

X1

V1
U=Vgs

R1
R=1

Is

Equation

Eqn1
PL_Cap=PlotVs(Cap, Vgs)
y=stoy(S)
Omega=2*pi*frequency
Cap=imag(y[1,1])/Omega
Rin=real(y[1,1])/(imag(y[1,1])*imag(y[1,1]))
L=1e-6
W=20e-6
Cox=3.45e-3
C_parallel_plate=W*L*Cox
PL_Cap_ratio=PlotVs(Cap/C_parallel_plate, Vgs)

Vgs	frequency	Rin
-0.15	1e6	1
-0.1	1e6	1
-0.05	1e6	1
0	1e6	1
0.05	1e6	1
0.1	1e6	1

Versus.0001: -1.5
PL_Cap: 6.9e-14

Versus.0001: 0
PL_Cap: 2.24e-14

Versus.0001: 1.5
PL_Cap: 6.9e-14

Versus.0002: -1.5
PL_Cap_ratio: 1

Versus.0002: 0
PL_Cap_ratio: 0.325

Versus.0002: 1.5
PL_Cap_ratio: 0.999

Figure 6.12: Test circuit for simulating EKV v2.6 EDD model charge partitioning effects: Xpart = 0.4 or QD/QS = 40/60

6.9 End note

The first update of the Qucs EKV v2.6 model provides users with a more optimised model, with improved simulation performance and a more complete charge partitioning scheme. Even with these changes the model is still not complete. The nMOS and pMOS Verilog-A code needs to be unified and a number of optional parameters need to be added to the Qucs implementation of the EKV v2.6 model. The next update of the model is scheduled for the near future, following correction of bug reports sent in by Qucs users. Once again my thanks to Stefan Jahn for all his help and support during the first EKV v2.6 update development phase.

7 Compact Verilog-A pn junction photodiode model

7.1 Introduction

Optoelectronic devices are not included in Qucs version 0.0.14 or earlier releases of the software. With the growing importance of these devices, and indeed the fact that they are present in an increasing number of electronic systems, this is a significant omission. This report presents the structure and physical details of a Qucs implementation of a pn junction photodiode model. The photodiode model is the first in a planned series of Verilog-a compact device models for optoelectronic components. The report also introduces the concept of a light bus and shows how light paths can be added to Qucs simulation schematics. A number of example schematics are also included in the report to demonstrate the performance of the new Verilog-A pn junction photodiode model. The background to the work outlined in this report was first published in the International Journal of Numerical Modelling: Electronic Networks, Devices and Fields in September 2008[1].

7.2 pn junction photodiode effects modelled

The Qucs pn junction photodiode model includes the following features:

- Diode photocurrent response characteristics expressed as a function of light power and wavelength.

- Diode DC I-V characteristics in the forward and reverse bias regions including avalanche breakdown in reverse bias.

- Diode bias dependent capacitance.

- Diode shunt resistance.

---

[1]Brinson M.E. and Jahn S., Qucs: A GPL software package for circuit simulation, compact device modelling and circuit macromodelling from DC to RF and beyond, published online, 5 September 2008, `http://www3.interscience.wiley.com/journal/121397825/abstract`.

- Diode package series resistance.

- Device noise, including thermal, shot, flicker and quantum contributions.

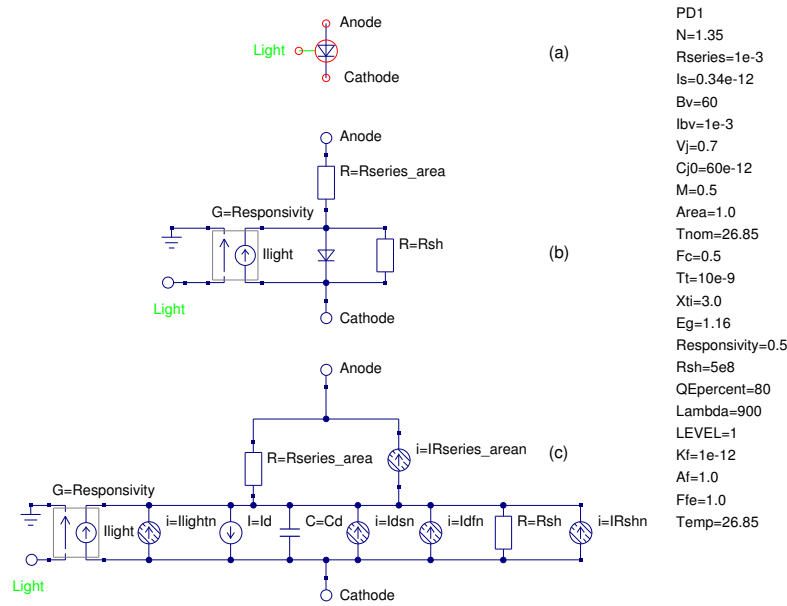## 7.3 The Qucs pn junction photodiode model

The schematic capture symbol and equivalent electrical circuit for the Qucs pn junction photodiode is shown in Fig. 7.1. In this model the DC properties and capacitance of the photodiode are represented by a semiconductor diode with a parallel shunt resistance Rsh. Device lead resistance is represented by series resistance Rseries_area. A voltage-controlled current source models the diode photocurrent. The gain of the controlled source is set as the responsivity of the photodiode. The equivalent circuit shown in Fig. 7.1(c) presents the complete photodiode model with thermal, shot and flicker noise sources included. One interesting, and unusual, feature of the Qucs photodiode noise model is the inclusion of quantum shot noise, modelled by noise current source *Ilightn*. Photocurrent *Ilight* is given by

$$Ilight = Light \cdot Responsivity \tag{7.1}$$

where *Light* is the optical signal in W and *Responsivity* is the the spectral responsivity in A/W. Responsivity can also be written in terms of the photodiode quantum efficiency given by.

$$Responsivity = \frac{QE \cdot q \cdot \lambda}{h \cdot c} = \frac{QE_{percent} \cdot \lambda}{1.2398e5} \tag{7.2}$$

where QE is the photodiode quantum efficiency, λ is the light wavelength in nm, h is Plank's constant, c is the speed of light in a vacuum, and $QE_{percent}$ is the quantum efficiency given in percentage. In Fig. 7.1 the optical signal path is shown as a green line connected at the side of the photodiode symbol. In Qucs simulations optical signals are modelled as voltage quantities expressed as real numbers, even though they represent a power quantity, and are shown on a schematic as green connecting lines between components. This has the effect of clearly identifying optical signal paths on a schematic, when compared to the blue lines which indicate connecting wires characterised by conventional current and voltage signals. Similarly, signal sources which input light to a circuit are coloured green, see later example simulations.

PD1
N=1.35
Rseries=1e-3
Is=0.34e-12
Bv=60
Ibv=1e-3
Vj=0.7
Cj0=60e-12
M=0.5
Area=1.0
Tnom=26.85
Fc=0.5
Tt=10e-9
Xti=3.0
Eg=1.16
Responsivity=0.5
Rsh=5e8
QEpercent=80
Lambda=900
LEVEL=1
Kf=1e-12
Af=1.0
Ffe=1.0
Temp=26.85

Figure 7.1: Qucs pn junction photodiode model: (a) schematic capture symbol, (b) basic model circuit, (c) full equivalent circuit, including noise

7.3.1 Photodiode parameters

Name	Symbol	Description	Unit	Default
N	N	photodiode emission coefficient		1.35
Rseries	$Rseries$	series lead resistance	Ω	$1e-3$
Is	Is	diode dark current	A	$0.34e-12$
Bv	Bv	reverse breakdown voltage	V	60.0
Ibv	Ns	current at reverse breakdown voltage	A	$1e-3$
Vj	Vj	junction potential	V	0.7
Cj0	$Cj0$	zero bias junction capacitance	F	$60e-12$
M	M	grading coefficient		0.5
Area	$Area$	diode relative area		1
Tnom	$Tnom$	parameter measurement temperature	$^\circ$C	26.85
Fc	Fc	forward-bias depletion capacitance coefficient		0.5
Tt	Tt	transit time	s	$10e-9$
Xti	Xti	saturation current temperature exponent		3.0
Eg	Eg	energy gap	eV	1.16
Responsivity	$Responsivity$	responsivity	A/W	0.5
Rsh	Rsh	shunt resistance	Ω	$5e8$
QEpercent	$QEpercent$	quantum efficiency	%	80.0
Lambda	$Lambda$	light wavelength	nm	900.0
LEVEL		responsivity calculator selector*		1
Kf	KF	flicker noise coefficient		$1e-12$
Af	Af	flicker noise exponent		1.0
Ffe	Ffe	flicker noise frequency exponent		1.0
Temp	$Temp$	device temperature	$^\circ$C	26.85

* Parameter LEVEL is used to select how the photodiode *Responsivity* is determined: with LEVEL = 1 the model uses the listed value of parameter *Responsivity* or calculates it's value if QEpercent is not equal to zero; with LEVEL = 2 *Responsivity* is always calculated using QEpercent.

7.3.2 pn junction photodiode model equations

- Basic semiconductor DC characteristics

$$Id = I1 + I2 + I3 + I4 \tag{7.3}$$

Where

1. $I1 = Area \cdot Is(T2) \cdot \left[limexp\left(\dfrac{Vd}{N \cdot Vt(T2)}\right) - 1\right] + Vd \cdot GMIN \quad \forall\,(Vd > -5 \cdot N \cdot Vt)$

2. $I2 = -Area \cdot Is(T2) + Vd \cdot GMIN \quad \forall\,(-Bv < Vd) \text{ and } (Vd > -5 \cdot N \cdot Vt)$

3. $I3 = -Ibv \quad \forall\,(Vd == -Bv)$

1. $Cdiff = \dfrac{dQdiff}{dVd} = Tt \cdot \dfrac{dId}{dVd}$

2. $Cdep = \dfrac{dQdep}{dVd} = Area \cdot Cj0 \cdot \left(1 - \dfrac{Vd}{Vj}\right)^{-M}$

where $Qd = Qdep + Qdiff$, and

$$Qd = Tt \cdot Id + \dfrac{Area \cdot Cj0 \cdot Vj}{1-M} \cdot \left\{ 1 - \left(1 - \dfrac{Vd}{Vj}\right)^{1-M} \right\} \qquad \forall \, (Vd < Fc \cdot Vj)$$

$$Qd = Tt \cdot Id + Area \cdot Cj0 \cdot \left\{ F1 + \dfrac{1}{F2} \cdot \left(F3 \cdot (Vd - Fc \cdot Vj) + \left[\dfrac{M}{2 \cdot Vj}\right] \cdot [Vd^2 - (Fc \cdot Vj)^2] \right) \right.$$

$$\forall \, (Vd >= Fc \cdot Vj)$$

Where

$F1 = \dfrac{Vj}{1-M} \cdot \left[1 - (1-Fc)^{1-M} \right]$

$F2 = [1 - Fc]^{1+M}$

$F3 = 1 - Fc \cdot (1+M)$

and GMIN = 1e-12S.

- Diode area factors

  1. $Is\_area = Is \cdot Area$

  2. $Cjo\_area = Cj0 \cdot Area$

  3. $Rseries\_area = \dfrac{Rseries}{Area}$

- Diode temperature factors

  1. $Is(T2) = Is \cdot \left\{ \dfrac{T2}{T1} \right\}^{\frac{Xti}{N}} \cdot limexp \left\{ \dfrac{-Eg(T1)}{Vt(T2)} \cdot \left[1 - \dfrac{T2}{T1} \right] \right\}$

  2. $Vj(T2) = \dfrac{T2}{T1} \cdot Vj - 2 \cdot Vt(T2) \cdot ln \left(\dfrac{T2}{T1}\right)^{1.5} - \left\{ \dfrac{T2}{T1} \cdot Eg(T1) - Eg(T2) \right\}$

  3. $Cj0(T2) = Cj0 \cdot \left\{ 1 + M \cdot \left[400e - 6 \cdot (T2 - T1) - \dfrac{Vj(T2)}{Vj} \right] \right\}$

  Where

  $T1 = Tnom + 273.15, \quad T2 = Temp + 273.15$

  $Eg(T) = EG(0) - \dfrac{7.02e - 4 \cdot T^2}{1108 + T}$

$$Vt = \frac{K \cdot T1}{q}, \quad Vt(T2) = \frac{K \cdot T2}{q} \text{ and } K \text{ and } q \text{ have their usual mean-}$$
ing.

- Diode photocurrent

$$Ilight = Light \cdot Responsivity \tag{7.5}$$

Where $Responsivity = \dfrac{QE \cdot q \cdot \lambda}{h \cdot c} = \dfrac{QEpercent \cdot \lambda}{1.2398e55}$

- photodiode noise

$$Ipd\_noise^2 = IRshn^2 \cdot \Delta f + Idsn^2 \cdot \Delta f + Idfn^2 \cdot \Delta f + Ilightn^2 \cdot \Delta f \tag{7.6}$$

Where $IRshn^2 = \dfrac{4 \cdot K \cdot T}{Rsh}, \; Idsn^2 = 2 \cdot q \cdot Id, \; Idfn^2 = \dfrac{Kf \cdot Id^{Af}}{f^{Ffe}},$

$Ilight^2 = 2 \cdot q \cdot Ilight$, and Δf is the noise frequency bandwidth in Hz.

7.3.3 Verilog-A model code

```
//    Qucs compact photodiode model
//    The structure and theoretical background to the photodiode
//    Verilog-a model are presented in the Qucs photodiode report.
//
//    This is free software; you can redistribute it and/or modify
//    it under the terms of the GNU General Public License as published by
//    the Free Software Foundation; either version 2, or (at your option)
//    any later version.
//
//    Copyright (C), Mike Brinson, mbrin72043@yahoo.co.uk, October 2008.
//
'include "disciplines.vams"
'include "constants.vams"
module photodiode (Anode, Cathode, Light);
 inout Anode, Cathode, Light;
 electrical Anode, Cathode, Light;
 electrical n1;
 'define attr(txt) (*txt*)
//
 parameter real N=1.35 from [1e-6:inf]
         'attr(info="photodiode emission coefficient");
 parameter real Rseries=1e-3 from [1e-6:inf]
         'attr(info="series lead resistance" unit = "Ohm");
 parameter real Is=0.34e-12 from [1e-20:inf]
         'attr(info="diode dark current" unit="A" );
 parameter real Bv=60 from [1e-6:inf]
         'attr(info="reverse breakdown voltage" unit="V");
 parameter real Ibv=1e-3 from [1e-6:inf]
```

```
                    'attr(info="current_at_reverse_breakdown_voltage" unit="A");
  parameter real Vj=0.7 from [1e-6:inf]
                    'attr(info="junction_potential" unit="V");
  parameter real Cj0=60e-12 from [0:inf]
                    'attr(info="zero-bias_junction_capacitance" unit="F");
  parameter real M=0.5 from [1e-6:inf]
                    'attr(info="grading_coefficient");
  parameter real Area=1.0 from [1.0:inf]
                    'attr(info="diode_relative_area");
  parameter real Tnom=26.85 from [-273:inf]
                    'attr(info="parameter_measurement_temperature" unit="Celsius");
  parameter real Fc=0.5 from [1e-6:inf]
                    'attr(info="forward-bias_depletion_capcitance_coefficient");
  parameter real Tt=10e-9 from [1e-20:inf]
                    'attr(info="transit_time" unit="s" );
  parameter real Xti=3.0 from [1e-6:inf]
                    'attr(info="saturation_current_temperature_exponent");
  parameter real Eg= 1.16 from [1e-6:inf]
                    'attr(info="energy_gap" unit="eV");
  parameter real Responsivity=0.5 from [1e-6:inf]
                    'attr(info="responsivity" unit="A/W");
  parameter real Rsh=5e8 from [1e-6:inf]
                    'attr(info="shunt_resistance" unit="Ohm");
  parameter real QEpercent=80 from [0:100]
                    'attr(info="quantum_efficiency" unit="%");
  parameter real Lambda=900 from [100:2000]
                    'attr(info="light_wavelength" unit="nm");
  parameter integer LEVEL=1 from [1:2]
                    'attr(info="responsivity_calculator_selector");
  parameter real Kf=1e-12 from [0:inf]
                    'attr(info="flicker_noise_coefficient");
  parameter real Af=1.0 from [0:inf]
                    'attr(info="flicker_noise_exponent");
  parameter real Ffe=1.0 from [0:inf]
                    'attr(info="flicker_noise_frequency_exponent");
//
real A, B, T1, T2, F1, F2, F3, Rseries_Area, Eg_T1, Eg_T2,
real Vt_T2, Vj_T2, Cj0_T2, Is_T2, GMIN;
real I1, I2, I3, I4, I5, Id, V1, Q1, Q2, fourkt, TwoQ, Res1,
real Res2, Res, Vt, I_flicker;
real con1, con2, con3, con4, con5, con6;
// Model branches
branch (Anode, n1) b6;
branch (n1, Cathode) b1;
//
analog begin
// Model equations
@(initial_step)
  begin
    Rseries_Area=(Rseries+1e-10)/Area;
    A=7.02e-4;
    B=1108.0;
    T1=Tnom+273.15;
    T2=$temperature;
    Vt='P_K*300.0/'P_Q;
    Vt_T2='P_K*T2/'P_Q;
    F1=(Vj/(1-M))*(1-pow((1-Fc),(1-M)));
    F2=pow((1-Fc), (1+M));
    F3=1-Fc*(1+M);
    Eg_T1=Eg-A*T1*T1/(B+T1);
    Eg_T2=Eg-A*T2*T2/(B+T2);
    Vj_T2=(T2/T1)*Vj-2*$vt*ln(pow((T2/T1),1.5))-((T2/T1)*Eg_T1-Eg_T2);
```

```verilog
    GMIN=1e−12;
    Cj0_T2=Cj0*(1+M*(400e−6*(T2−T1)−(Vj_T2−Vj)/Vj));
    Is_T2=Is*pow( (T2/T1), (Xti/N))*limexp(−(Eg_T1)/$vt*(1−T2/T1));
    Res1=(QEpercent != 0) ? QEpercent*Lambda/1.2398e5:Responsivity;
    Res2=QEpercent*Lambda/1.2938e5;
    Res=(LEVEL==1) ? Res1 : Res2;
    con1=−5.0*N*Vt;
    con2=Area*Is_T2;
    con3=Area*Cj0_T2;
    con4=Fc*Vj;
    con5=Fc*Vj_T2;
    con6=Bv/Vt_T2;
  end;
// Current contributions
V1=V(b1);
I1=(V1 > con1) ? con2*(limexp(V1/(N*Vt_T2))−1.0)+GMIN*V1: 0;
I2=(V1 <= con1) ? −con2+GMIN*V1 :0;
I3=(V1 == −Bv)?−Ibv: 0;
I4=(V1<−Bv)?−con2*(limexp(−(Bv+V1)/Vt_T2)−1.0+con6):0;
Q1=(V1<con4)  ? Tt*I1 + con3*(Vj_T2/(1−M))*(1−pow((1−V1/Vj_T2),(1−M))):0;
Q2=(V1>=con4) ? Tt*I1 + con3*(F1+(1/F2)*(F3*(V1−con5)+
                (M/(2.0*Vj_T2))*(V1*V1−con5*con5))):0;
I5=V(Light)*Res;
Id=I1+I2+I3+I4;
I(b1) <+ −I5;
I(b1) <+ V(b1)/Rsh;
I(b6)<+V(b6)/Rseries_Area;
I(b1)<+Id;
I(b1)<+ddt(Q1+Q2);
I(Light)<+V(Light)/1e10;
// Noise contributions
fourkt=4.0*`P_K*$temperature;
TwoQ=2.0*`P_Q;
I_flicker=pow(Id, Af);
I(b6)<+white_noise(fourkt/Rseries_Area , "thermal");
I(b1)<+white_noise(fourkt/Rsh , "thermal");
I(b1)<+white_noise(TwoQ*Id , "shot");
I(b1)<+flicker_noise(Kf*I_flicker , Ffe , "flicker");
I(b1)<+white_noise(TwoQ*I5 , "shot");
//
end
endmodule
```
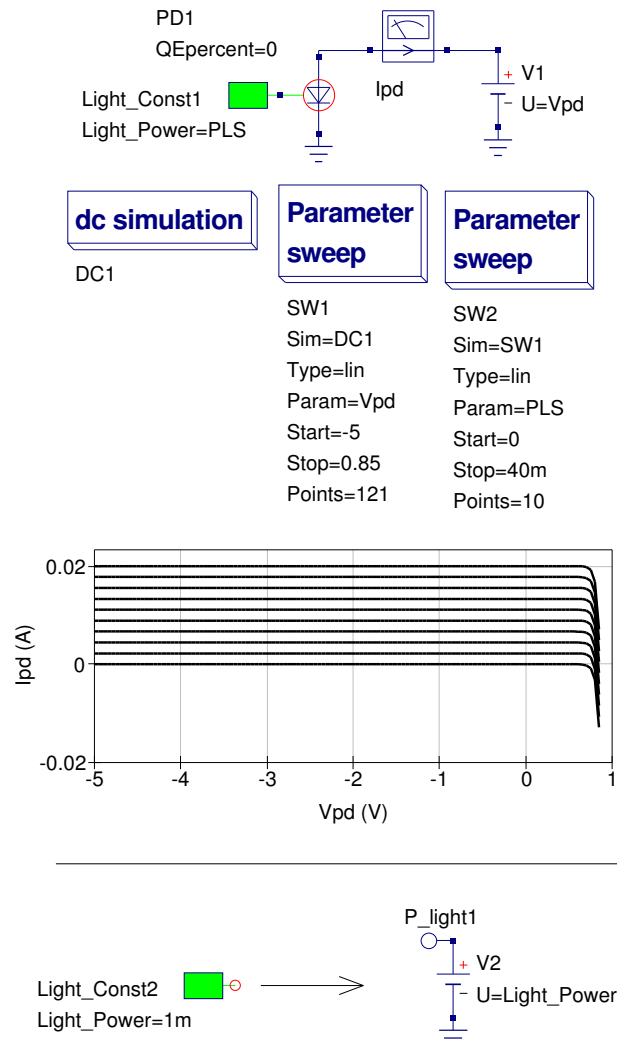
7.4 Example photodiode circuits

Figure 7.2: Basic photodiode test circuit for simulating device output current as a function of light input level and applied DC bias
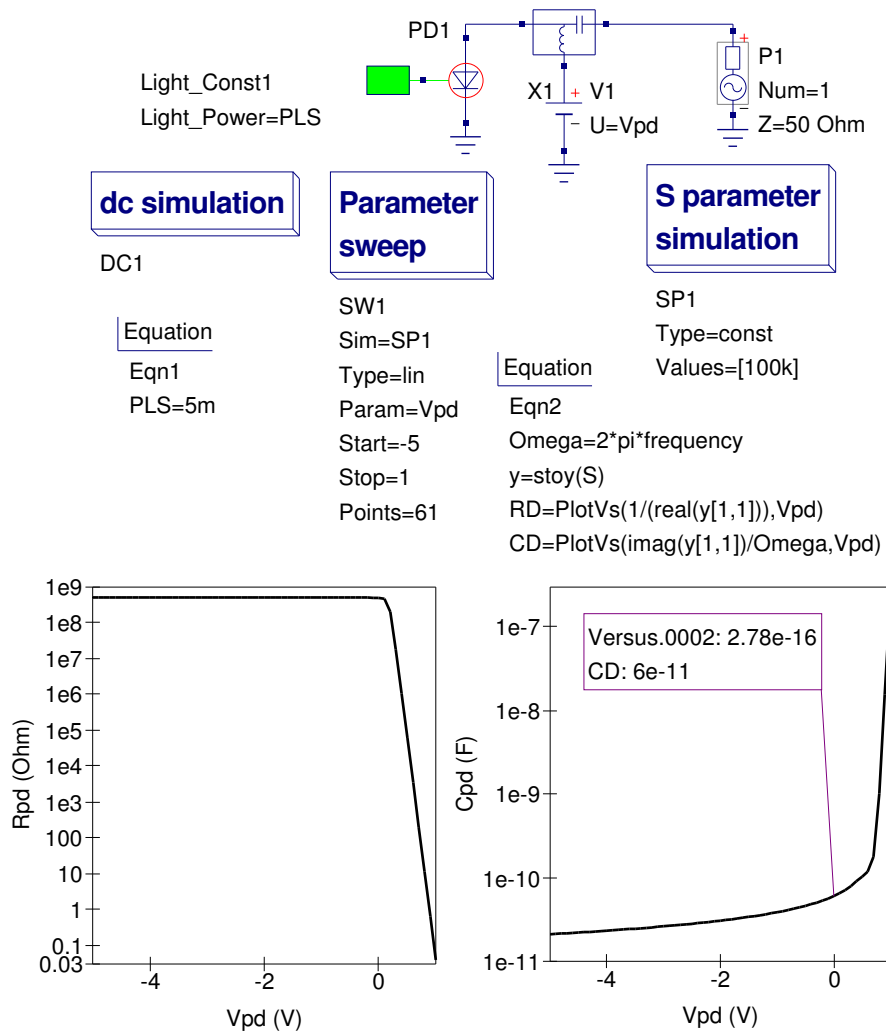
Figure 7.3: Photodiode test circuit for extracting device capacitance and resistance as a function of applied DC bias
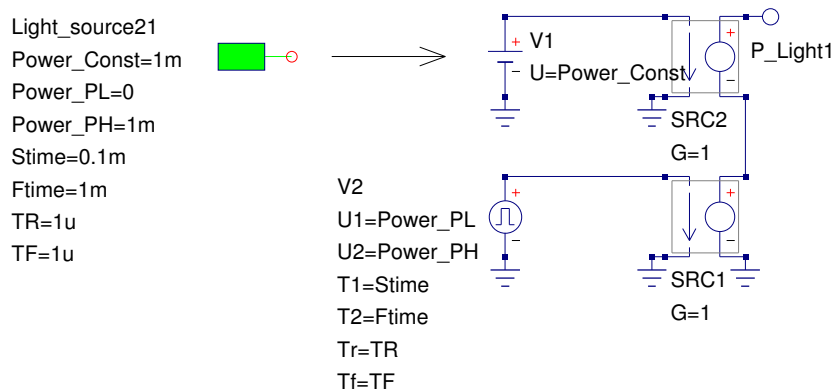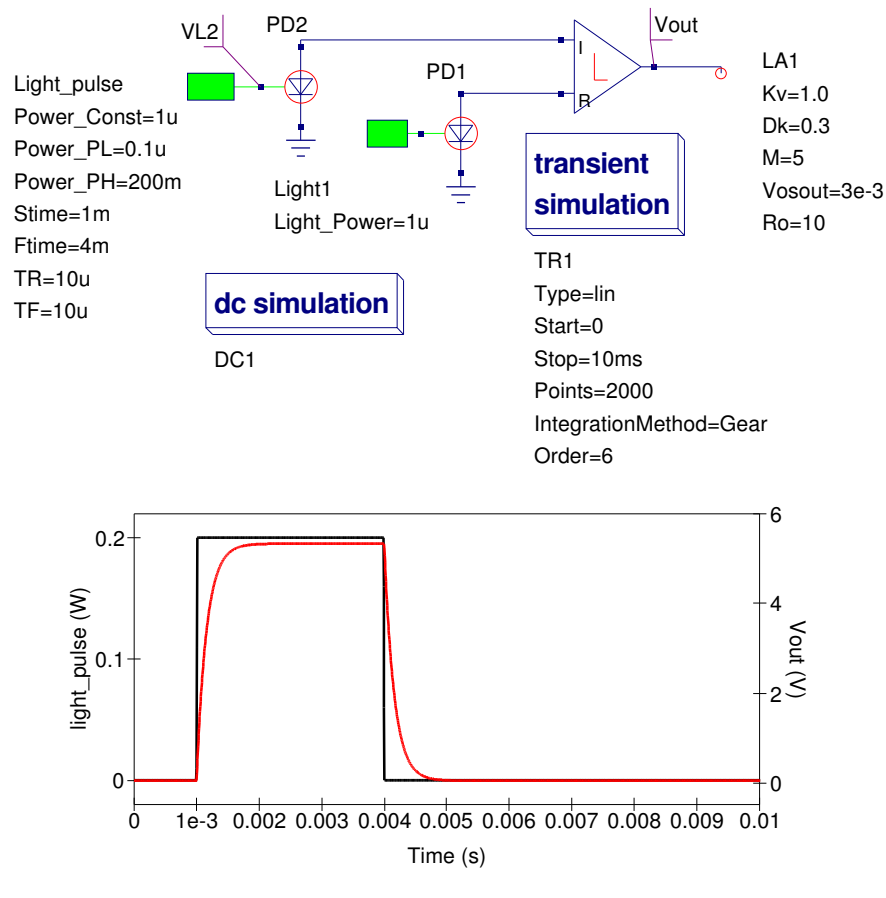
VL2 PD2   Vout

Light_pulse
Power_Const=1u
Power_PL=0.1u
Power_PH=200m
Stime=1m
Ftime=4m
TR=10u
TF=10u

PD1

LA1
Kv=1.0
Dk=0.3
M=5
Vosout=3e-3
Ro=10

Light1
Light_Power=1u

transient simulation

dc simulation

DC1

TR1
Type=lin
Start=0
Stop=10ms
Points=2000
IntegrationMethod=Gear
Order=6

Light_source21
Power_Const=1m
Power_PL=0
Power_PH=1m
Stime=0.1m
Ftime=1m
TR=1u
TF=1u

V1
U=Power_Const

P_Light1

SRC2
G=1

V2
U1=Power_PL
U2=Power_PH
T1=Stime
T2=Ftime
Tr=TR
Tf=TF

SRC1
G=1

Figure 7.4: Pulsed light power comparator circuit using two photodiodes and a logarithmic amplifier

180
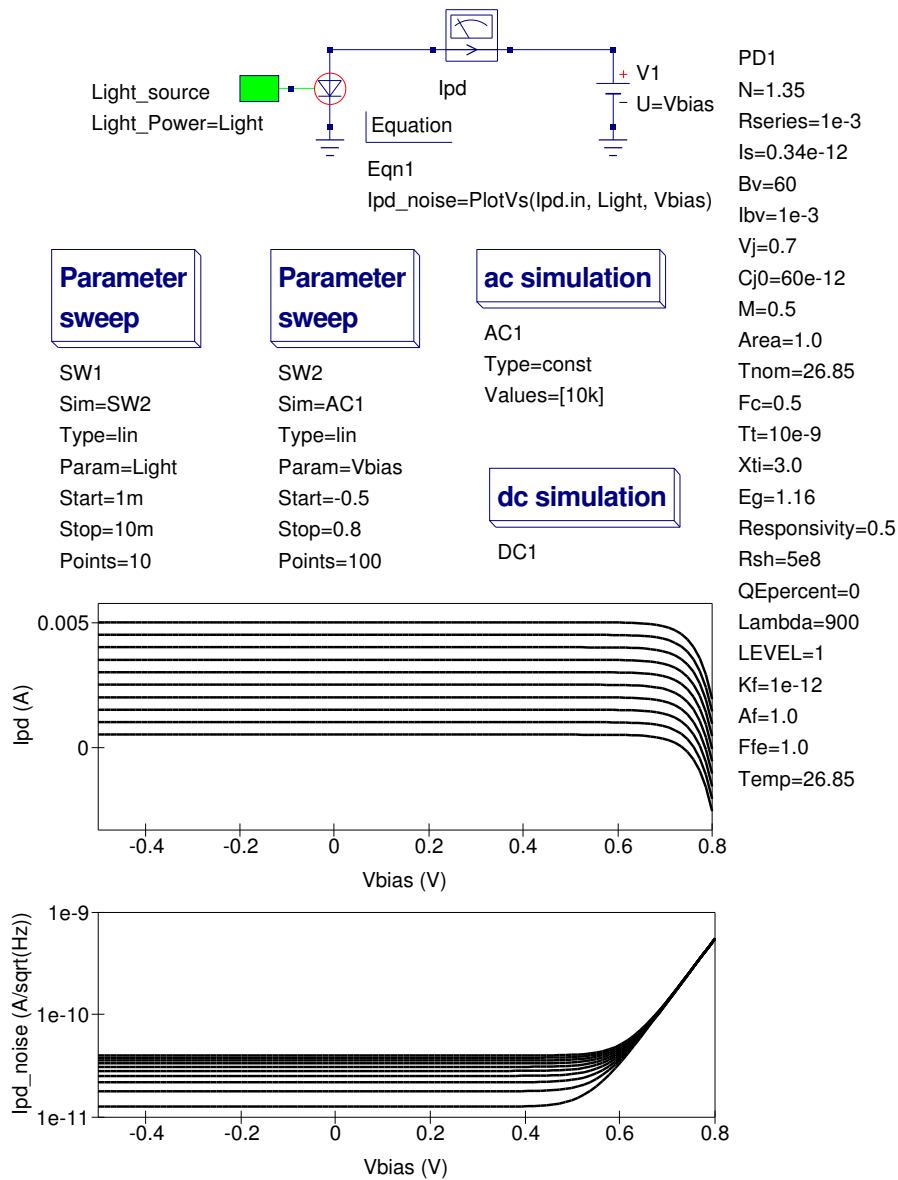
Figure 7.5: Photodiode noise test circuit and simulated noise characteristics

7.5 End Note

Optoelectronic devices are an important group of electronic components and as such deserve more attention than has been given to them in the past by the Qucs development team. This situation should improve over the coming year. The pn junction photodiode model is the first in a planned series of optoelectronic models, including models for transimpedance amplifiers, optical actuators and optical media. Given time it should be possible to significantly improve Qucs optoelectronic capabilities. The photodiode model reported here is an interesting model in that it is one of the first Verilog-A models to fully utilize the recent changes to the ADMS/Qucs interface which allow proper initialisation of model parameters. A new procedure for combining Verilog-A generated models also introduces for the first time the initial stages towards a fully modularized approach to linking complex models with the main body of the Qucs code. Once again my thanks to Stefan Jahn for all his encouragement and help during the period I worked on the photodiode model.

8 SPICE to Qucs conversion: Test File 1

8.1 Introduction

8.1.1 Title

DC and independent voltage pulse generator test.

8.1.2 Test file name

8.1.3 SPICE specification

Format:
VX N+ N- [[DC] DC/TRAN VALUE] [AC [ACMAG [ACPHASE]]]

Notes:

1. Characters [and] enclose optional items

2. Character / denotes OR

3. Independent voltage source names begin with the letter V

4. X denotes name of source

5. N+ and N- are the positive and negative nodes respectively

6. Voltage sources need not be grounded

Specification of SPICE statement being tested:

VX N+ N- [[DC] VALUE] [PULSE(V1 V2 [TD [TR [TF [PW [PER]]]]])] Notes:

1. PULSE generates a periodic pulse, where

2. V1 is the initial value; default: must be specified

3. V2 is the pulsed value; default: must be specified

4. TD is the delay time; default value = TSTEP

5. TR is the rise time; default value = TSTEP

6. TF is the fall time; default value = TSTEP

7. PW is the pulse width; default value = TSTOP

8. PER is the period; default value = TSTOP

8.2 Test code and schematic

SPICE code: File `S2Q_test1.cir`

```
* SPICE to Qucs syntax test file 1.
* DC and independent voltage pulse sources, plus resistors.
*
.subckt S2Q_test1 p01 p02 p03 p04 p05 p06 p07 p08 p09 p10 p11
v1 p01 0 1v
r1 p01 0 10k
*
v2 p02 0 dc 1v
r2 p02 0 10k
*
v3 p03 0 pulse(0 5)
r3 p03 0 10k
*
v4 p04 0 pulse( 0 5 20n)
r4 p04 0 10k
*
v5 p05 0 pulse(0 5 20n 10n)
r5 p05 0 10k
*
v6 p06 0 pulse(0 5 20n 10n 10n)
r6 p06 0 10k
*
v7 p07 0 pulse(0 5 20n 10n 10n 50n)
r7 p07 0 10k
*
v8 p08 0 pulse(0 5 20n 10n 10n 50n 100n)
r8 p08 0 10k
```
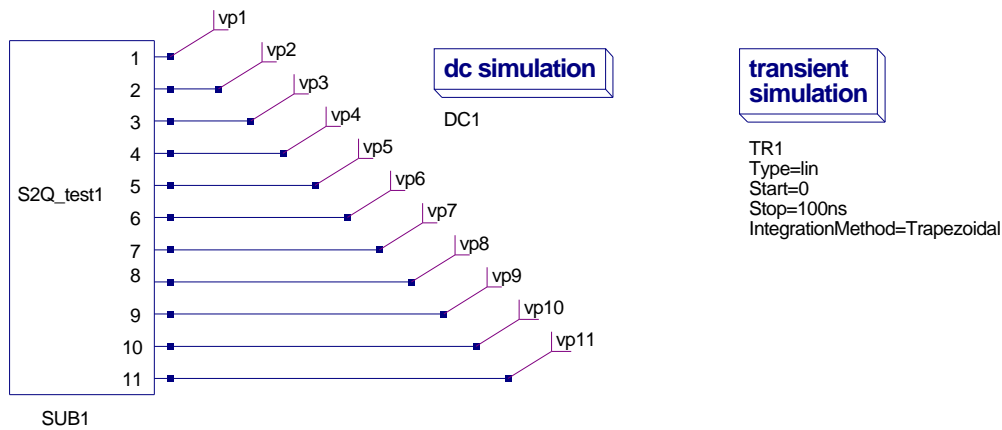
Figure 8.1: SPICE to Qucs conversion: Test1

```
*
v9  p09  0  pulse(0 5 20n 1n 1n 20n 40n)
r9  p09  0  10k
*
v10  p10  0  pulse(0 5 20n 0.1n 0.1n 5n 50n)
r10  p10  0  10k
*
v11  p11  0  dc 5v pulse(0 5 20n 0.5n 0.5n 10n 20n)
r11  p11  0  10k
.ends
.end
```

8.3 History of simulation results

8.3.1 March 8 2007, Simulation tests by Mike Brinson

1. Test 1 : Vp1.Vt; Pass correct result.

2. Test 2 : Vp2.Vt; Pass correct result.

3. Test 3 : Vp3.Vt; **Fail** TR and TF should default to TSTEP [TSTEP=1nS in test]

4. Test 4 : Vp4.Vt; **Fail** TR and TF should default to TSTEP [TSTEP=1nS in test]

5. Test 5 : Vp5.Vt; Pass.

6. Test 6 : Vp6.Vt; Pass.

7. Test 7 : Vp7.Vt; Pass.

8. Test 8 : Vp8.Vt; Pass.

9. Test 9 : Vp9.Vt; **Fail** - waveform should repeat after 60ns.

10. Test 10 : Vp10.Vt; **Fail** - waveform should repeat after 70ns.

11. Test 11 : Vp11.Vt; **Fail**

    1. waveform should repeat after 40ns,
    2. `Vdc:V11 _cnet8 _ref U="0"` incorrect,

       should be `Vdc:V11 _cnet8 _ref U="5"`

8.3.2 March 10 2007, Simulation tests by Mike Brinson

Code modified * `check_spice.cpp`: Handling periodic pulse sources correctly. Also default Tr/Tf values for these sources to a given .TRAN step value : Stefan Jahn.

Restriction on SPICE code: $TD + TR + PW + TF < PER$, otherwise a negative TL time for the repetitive pulse occurs and simulation fails.

SPICE test file `S2Q_test1.cir` modified: Mike Brinson

1. Test 1 : Vp1.Vt; Pass.

2. Test 2 : Vp2.Vt; Pass.

3. Test 3 : Vp3.Vt; Pass

4. Test 4 : Vp4.Vt; Pass

5. Test 5 : Vp5.Vt; Pass.

6. Test 6 : Vp6.Vt; Pass.

7. Test 7 : Vp7.Vt; Pass.

8. Test 8 : Vp8.Vt; Pass.

```
# Qucs 0.0.11   /media/hda2/spice_to_qucs_prj/s2Q(test1).sch

.Def:S2Q_test1 _net0 _net1 _net2 _net3 _net4 _net5 _net6
                _net7 _net8 _net9 _net10
Sub:X1 _net0 _net1 _net2 _net3 _net4 _net5 _net6 _net7
                _net8 _net9 _net10 gnd Type="S2Q_test1_cir"
.Def:End

.Def:S2Q_test1_cir _netP01 _netP02 _netP03 _netP04 _netP05
                _netP06 _netP07 _netP08 _netP09 _netP10 _netP11 _ref
   .Def:S2Q_TEST1 _ref _netP01 _netP02 _netP03 _netP04 _netP05
                _netP06 _netP07 _netP08 _netP09 _netP10 _netP11
   Vpulse:V11 _netP11 _cnet8 U1="0" U2="5" T1="20n"
                Tr="0.5n" Tf="0.5n" T2="3.1e-08"
   Vpulse:V10 _netP10 _cnet7 U1="0" U2="5" T1="20n"
                Tr="0.1n" Tf="0.1n" T2="2.52e-08"
   Vpulse:V9 _netP09 _cnet6 U1="0" U2="5" T1="20n"
                Tr="1n" Tf="1n" T2="4.2e-08"
   Vpulse:V8 _netP08 _cnet5 U1="0" U2="5" T1="20n"
                Tr="10n" Tf="10n" T2="9e-08"
   Vpulse:V7 _netP07 _cnet4 U1="0" U2="5" T1="20n"
                Tr="10n" Tf="10n" T2="9e-08"
   Vpulse:V6 _netP06 _cnet3 U1="0" U2="5" T1="20n"
                Tr="10n" Tf="10n" T2="4e-08"
   Vpulse:V5 _netP05 _cnet2 U1="0" U2="5"
                T1="20n" Tr="10n" T2="3e-08"
   Vpulse:V4 _netP04 _cnet1 U1="0" U2="5"
                T1="20n" T2="2e-08"
   Vpulse:V3 _netP03 _cnet0 U1="0"
                U2="5" T2="0" T1="0"
  Vdc:V1 _netP01 _ref U="1V"
  R:R1 _netP01 _ref R="10k"
  Vdc:V2 _netP02 _ref U="1V"
  R:R2 _netP02 _ref R="10k"
  Vdc:V3 _cnet0 _ref U="0"
  R:R3 _netP03 _ref R="10k"
  Vdc:V4 _cnet1 _ref U="0"
  R:R4 _netP04 _ref R="10k"
  Vdc:V5 _cnet2 _ref U="0"
  R:R5 _netP05 _ref R="10k"
  Vdc:V6 _cnet3 _ref U="0"
  R:R6 _netP06 _ref R="10k"
  Vdc:V7 _cnet4 _ref U="0"
  R:R7 _netP07 _ref R="10k"
  Vdc:V8 _cnet5 _ref U="0"
  R:R8 _netP08 _ref R="10k"
  Vdc:V9 _cnet6 _ref U="0"
  R:R9 _netP09 _ref R="10k"
  Vdc:V10 _cnet7 _ref U="0"
  R:R10 _netP10 _ref R="10k"
  Vdc:V11 _cnet8 _ref U="0"
  R:R11 _netP11 _ref R="10k"
   .Def:End
  Sub:X1 _ref _netP01 _netP02 _netP03 _netP04 _netP05
         _netP06 _netP07 _netP08 _netP09 _netP10 _netP11 Type="S2Q_TEST1"
.Def:End


.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA" vntol="1_uV"
saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
Sub:SUB1 vp1 vp2 vp3 vp4 vp5 vp6 vp7 vp8 vp9 vp10 vp11 Type="S2Q_test1"
.TR:TR1 Type="lin" Start="0" Stop="100ns" Points="500"
IntegrationMethod="Trapezoidal" Order="2" InitialStep="1_ns"
MinStep="1e-16" MaxIter="150" reltol="0.001" abstol="1_pA"
vntol="1_uV" Temp="26.85" LTEreltol="1e-3" LTEabstol="1e-6" LTEfactor="1"
Solver="CroutLU" relaxTSR="no" initialDC="yes" MaxStep="0"
```

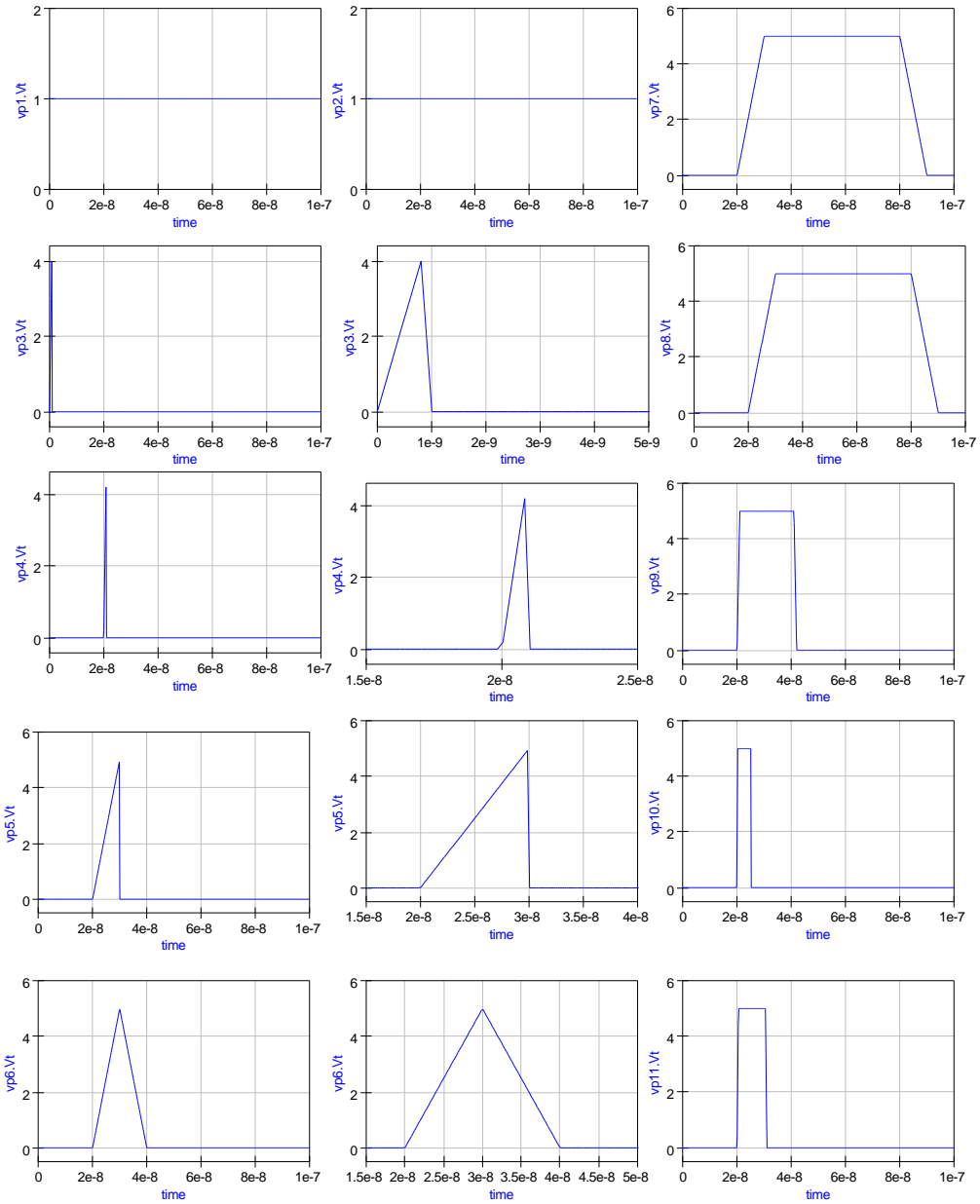187

Figure 8.2: Qucs netlist [Edited to fit on page width]

Figure 8.3: SPICE to Qucs conversion: Test1 simulation waveforms

```
* SPICE to Qucs syntax test file 1.
* DC and independent voltage pulse sources, plus resistors.
*
.subckt S2Q_test1 p01 p02 p03 p04 p05 p06 p07 p08 p09 p10 p11
v1 p01 0 1v
r1 p01 0 10k
*
v2 p02 0 dc 1v
r2 p02 0 10k
*
v3 p03 0 pulse(0 5)
r3 p03 0 10k
*
v4 p04 0 pulse( 0 5 20n)
r4 p04 0 10k
*
v5 p05 0 pulse(0 5 20n 10n)
r5 p05 0 10k
*
v6 p06 0 pulse(0 5 20n 10n 10n)
r6 p06 0 10k
*
v7 p07 0 pulse(0 5 20n 10n 10n 50n)
r7 p07 0 10k
*
v8 p08 0 pulse(0 5 20n 10n 10n 50n 100n)
r8 p08 0 10k
*
v9 p09 0 pulse(0 5 10n 1n 1n 20n 40n)
r9 p09 0 10k
*
v10 p10 0 pulse(0 5 20n 0.1n 0.1n 5n 50n)
r10 p10 0 10k
*
v11 p11 0 dc 5v pulse(-3 5 20n 0.5n 0.5n 10n 40n)
r11 p11 0 10k
.ends
.end
```

Figure 8.4: Modified SPICE test1 netlist

9. Test 9 : Vp9.Vt; Pass.

10. Test 10 : Vp10.Vt; Pass.
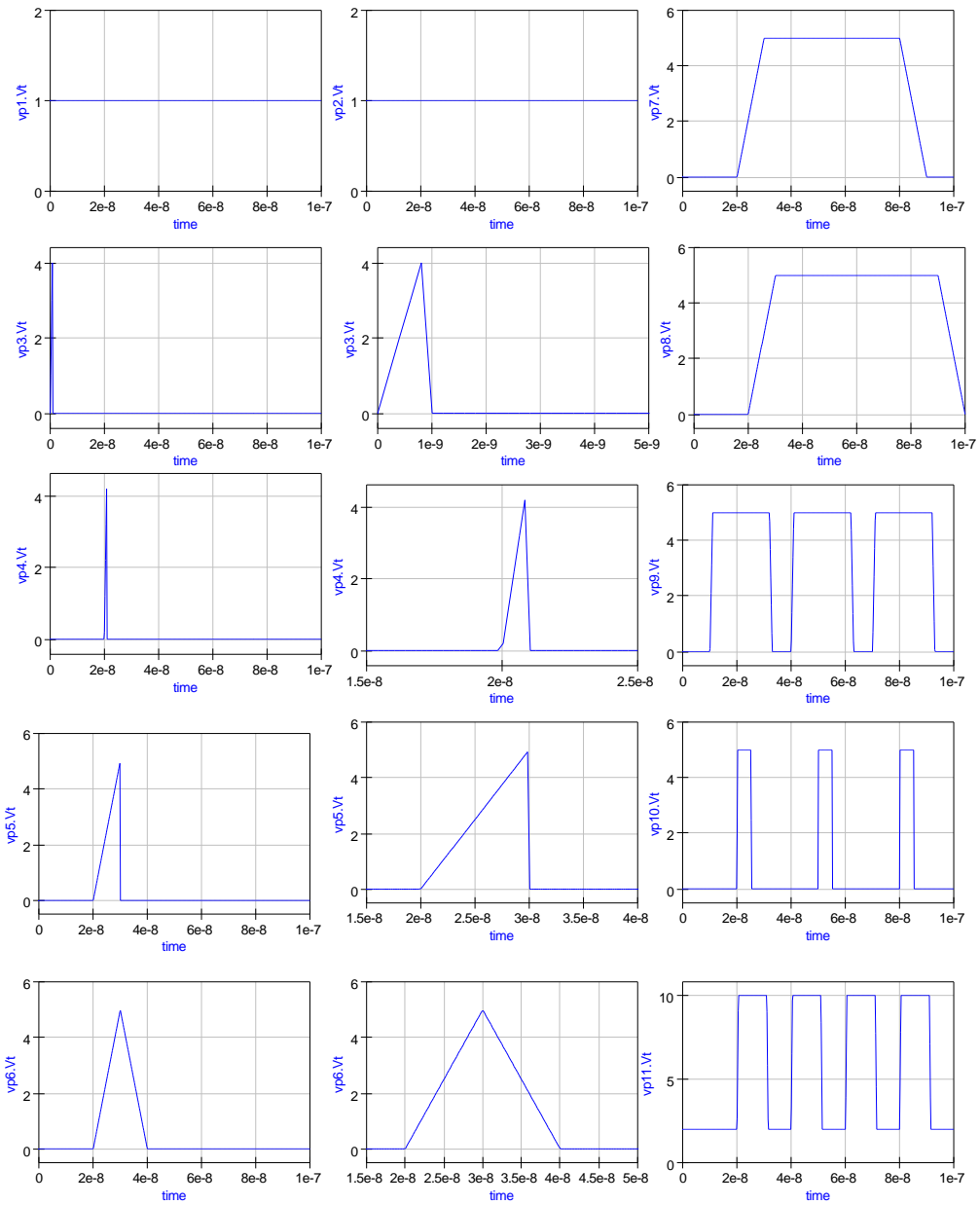
11. Test 11 : Vp11.Vt; Pass

Figure 8.5: SPICE to Qucs conversion: Modified test1 simulation waveforms

```
# Qucs 0.0.11  /media/hda2/spice_to_qucs_prj/s2Q(test1).sch

.Def:S2Q_test1 _net0 _net5 _net1 _net6 _net2 _net7 _net3
_net8 _net4 _net9 _net10
Sub:X1 _net0 _net5 _net1 _net6 _net2 _net7 _net3 _net8
_net4 _net9 _net10 gnd Type="S2Q_test1_cir"
.Def:End


.Def:S2Q_test1_cir _netP01 _netP02 _netP03 _netP04 _netP05
_netP06 _netP07 _netP08 _netP09 _netP10 _netP11 _ref
 .Def:S2Q_TEST1 _ref _netP01 _netP02 _netP03 _netP04
_netP05 _netP06 _netP07 _netP08 _netP09 _netP10 _netP11
   Vrect:V11 _netP11 _cnet8 U="8" Td="20n" Tr="0.5n" Tf="0.5n" TH="1.1e-08" TL="9e-09"
   Vrect:V10 _netP10 _cnet7 U="5" Td="20n" Tr="0.1n" Tf="0.1n" TH="5.2e-09" TL="2.48e-08"
   Vrect:V9 _netP09 _cnet6 U="5" Td="10n" Tr="1n" Tf="1n" TH="2.2e-08" TL="8e-09"
   Vrect:V8 _netP08 _cnet5 U="5" Td="20n" Tr="10n" Tf="10n" TH="7e-08" TL="1e-08"
   Vpulse:V7 _netP07 _cnet4 U1="0" U2="5" T1="20n" Tr="10n" Tf="10n" T2="9e-08"
   Vpulse:V6 _netP06 _cnet3 U1="0" U2="5" T1="20n" Tr="10n" Tf="10n" T2="4e-08"
   Vpulse:V5 _netP05 _cnet2 U1="0" U2="5" T1="20n" Tr="10n" T2="3e-08"
   Vpulse:V4 _netP04 _cnet1 U1="0" U2="5" T1="20n" T2="2e-08"
   Vpulse:V3 _netP03 _cnet0 U1="0" U2="5" T2="0" T1="0"
   Vdc:V1 _netP01 _ref U="1V"
   R:R1 _netP01 _ref R="10k"
   Vdc:V2 _netP02 _ref U="1V"
   R:R2 _netP02 _ref R="10k"
   Vdc:V3 _cnet0 _ref U="0"
   R:R3 _netP03 _ref R="10k"
   Vdc:V4 _cnet1 _ref U="0"
   R:R4 _netP04 _ref R="10k"
   Vdc:V5 _cnet2 _ref U="0"
   R:R5 _netP05 _ref R="10k"
   Vdc:V6 _cnet3 _ref U="0"
   R:R6 _netP06 _ref R="10k"
   Vdc:V7 _cnet4 _ref U="0"
   R:R7 _netP07 _ref R="10k"
   Vdc:V8 _cnet5 _ref U="0"
   R:R8 _netP08 _ref R="10k"
   Vdc:V9 _cnet6 _ref U="0"
   R:R9 _netP09 _ref R="10k"
   Vdc:V10 _cnet7 _ref U="0"
   R:R10 _netP10 _ref R="10k"
   Vdc:V11 _cnet8 _ref U="2"
   R:R11 _netP11 _ref R="10k"
   .Def:End
   Sub:X1 _ref _netP01 _netP02 _netP03 _netP04 _netP05 _netP06
   _netP07 _netP08 _netP09 _netP10 _netP11 Type="S2Q_TEST1"
.Def:End


.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA" vntol="1_uV"
saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
.TR:TR1 Type="lin" Start="0" Stop="100ns" Points="500"
IntegrationMethod="Trapezoidal" Order="2" InitialStep="1_ns"
MinStep="1e-16" MaxIter="150" reltol="0.001" abstol="1_pA"
vntol="1_uV" Temp="26.85" LTEreltol="1e-3" LTEabstol="1e-6"
LTEfactor="1" Solver="CroutLU" relaxTSR="no" initialDC="yes" MaxStep="0"
Sub:SUB1 vp1 vp2 vp3 vp4 vp5 vp6 vp7 vp8 vp9 vp10 vp11 Type="S2Q_test1"
```

Figure 8.6: Qucs netlist for modified test1 SPICE netlist [Edited to fit on page width]

192

8.4 References

1. A. Vladimirescu, Kaihe Zhang, A.R. Newton, D.O Pederson A. Sangiovanni-Vincentelli, SPICE 2G User's Guide (10 Aug 1981), Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca., 94720.

2. B. Johnson, T. Quarles, A.R. Newton, P.O. Pederson, A.Sangiovanni-Vincentelli, SPICE3 Version 3f User's Manual (October 1972), Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca., 94720.

3. Andrei Vladimirescu, THE SPICE book,1994, John Wiley and Sons. Inc., ISBN 0-471-609-26-9.

9 SPICE to Qucs conversion: Test File 2

9.1 Introduction

9.1.1 Title

DC and independent voltage sin generator test.

9.1.2 SPICE specification

Format:
VX N+ N- [[DC] DC/TRAN VALUE] [AC [ACMAG [ACPHASE]]]

Notes:

1. Characters [and] enclose optional items

2. Character / denotes OR

3. Independent voltage source names begin with the letter V

4. X denotes name of source

5. N+ and N- are the positive and negative nodes respectively

6. Voltage sources need not be grounded

Specification of SPICE statement being tested:

VX N+ N- [[DC] VALUE] [SIN(VO VA [FREQ [TD [KD]]]]

Notes:

1. SIN generates a periodic sinusoidal signal, where

2. VO is the DC offset; default: must be specified

3. VA is the signal amplitude; default: must be specified

4. FREQ is the signal frequency; default: value = 1/TSTOP

5. TD is initial delay before sinusoidal signal starts; default: value = 0 seconds

6. KD is the damping coefficient; default: value = 0. The damping factor has dimension 1/time.

9.2 Test code and schematic

SPICE code: File `S2Q_test2.cir`

```
* SPICE to Qucs syntax test file 2
* DC and independent voltage sin sources, plus resistors.
*
.subckt S2Q_test2 p01 p02 p03 p04 p05 p06 p07 p08 p09 p10 p11
v1 p01 0 1v
r1 p01 0 10k
*
v2 p02 0 dc 1v
r2 p02 0 10k
*
*v3 p03 0 sin(0 5)
r3 p03 0 10k
*
v4 p04 0 sin( 0 5 1k)
r4 p04 0 10k
*
v5 p05 0 sin(0 5 1k 0.5m)
r5 p05 0 10k
*
v6 p06 0 sin(0 5 1k 0.5m 100)
r6 p06 0 10k
*
v7 p07 0 sin(0 5 1k 0.5m 1000)
r7 p07 0 10k
*
v8 p08 0 dc 5v sin(0 5 1k 0.5m 1000)
r8 p08 0 10k
*
v9 p09 0 sin(-5 5 1k 0.5m 1000)
r9 p09 0 10k
```
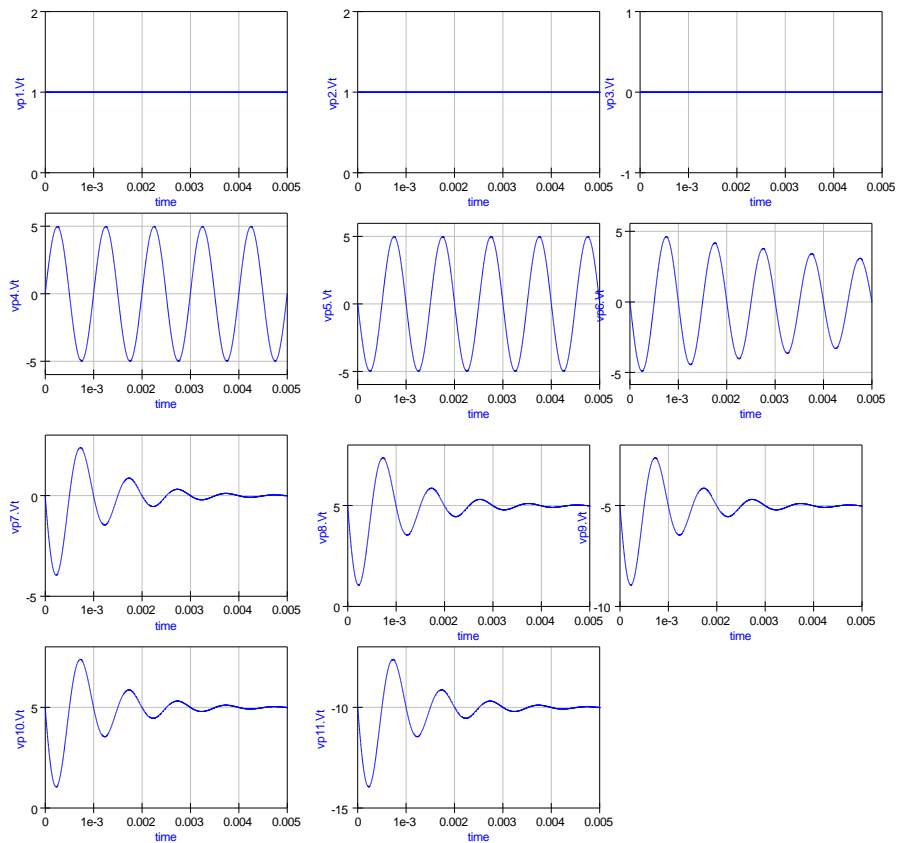
Figure 9.1: March 11: SPICE to Qucs conversion: Test2 waveforms

```
*
v10  p10  0  sin(5  5  1k  0.5m  1000)
r10  p10  0  10k
*
v11  p11  0  dc  −10  sin(5  5  1k  0.5m  1000)
r11  p11  0  10k
.ends
.end
```

9.3 History of simulation results

9.3.1 March 11 2007, Simulation tests by Mike Brinson

1. Test 1 : Vp1.Vt; Pass correct result.

2. Test 2 : Vp2.Vt; Pass correct result.

3. Test 3 : Vp3.Vt; **Fail** ERROR: line 17: checker error, no such variable 'nan' used in a 'Vac:V3' property NOTE error occurs when v3 uncommented.

4. Test 4 : Vp4.Vt; Pass

5. Test 5 : Vp5.Vt; **Fail** TD should be 0.5m seconds - otherwise OK.

6. Test 6 : Vp6.Vt; **Fail** TD should be 0.5m seconds - otherwise OK.

7. Test 7 : Vp7.Vt; **Fail** TD should be 0.5m seconds - otherwise OK.

8. Test 8 : Vp8.Vt; **Fail** TD should be 0.5m seconds - otherwise OK

9. Test 9 : Vp9.Vt; **Fail** TD should be 0.5m seconds - otherwise OK

10. Test 10 : Vp10.Vt; **Fail** TD should be 0.5m seconds - otherwise OK

11. Test 11 : Vp11.Vt; **Fail** TD should be 0.5m seconds, plus DC level wrong.

9.3.2 March 12 2007, Simulation tests by Mike Brinson

Code modifications:

1. * `check_spice.cpp`: Fixed DC offset of sinuasoidal voltage and current sources. Also apply default frequency if a transient analysis is given. Stefan Jahn

2. * vac.cpp, iac.cpp: Adjusted time dependency of damping factor Stefan Jahn

1. Test 1 : Vp1.Vt; Pass correct result.

2. Test 2 : Vp2.Vt; Pass correct result.

3. Test 3 : Vp3.Vt; Pass - see note 1 below.

4. Test 4 : Vp4.Vt; Pass.

5. Test 5 : Vp5.Vt; Pass - see note 2 below.

6. Test 6 : Vp6.Vt; Pass - see note 2 below.

7. Test 7 : Vp7.Vt; Pass - see note 2 below.

8. Test 8 : Vp8.Vt; Pass - see note 2 below.

9. Test 9 : Vp9.Vt; Pass - see note 2 below.

```
# Qucs 0.0.11   /media/hda2/S2Q_test2_prj/S2Q(test2).sch

.Def:S2Q_test2 _net0 _net1 _net2 _net3 _net4 _net5
_net6 _net7 _net8 _net9 _net10
Sub:X1 _net0 _net1 _net2 _net3 _net4 _net5 _net6
_net7 _net8 _net9 _net10 gnd Type="S2Q_test2_cir"
.Def:End


.Def:S2Q_test2_cir _netP01 _netP02 _netP03 _netP04 _netP05
_netP06 _netP07 _netP08 _netP09 _netP10 _netP11 _ref
  .Def:S2Q_TEST2 _ref _netP01 _netP02 _netP03 _netP04 _netP05
_netP06 _netP07 _netP08 _netP09 _netP10 _netP11
  Vac:V11 _netP11 _cnet8 U="5" f="1k" Phase="−180" Theta="1"
  Vac:V10 _netP10 _cnet7 U="5" f="1k" Phase="−180" Theta="1"
  Vac:V9 _netP09 _cnet6 U="5" f="1k" Phase="−180" Theta="1"
  Vac:V8 _netP08 _cnet5 U="5" f="1k" Phase="−180" Theta="1"
  Vac:V7 _netP07 _cnet4 U="5" f="1k" Phase="−180" Theta="1"
  Vac:V6 _netP06 _cnet3 U="5" f="1k" Phase="−180" Theta="0.1"
  Vac:V5 _netP05 _cnet2 U="5" f="1k" Phase="−180" Theta="0"
  Vac:V4 _netP04 _cnet1 U="5" f="1k" Phase="−0" Theta="0"
  Vac:V3 _netP03 _cnet0 U="5" Phase="−0" Theta="nan" f="1e+09"
  Vdc:V1 _netP01 _ref U="1V"
  R:R1 _netP01 _ref R="10k"
  Vdc:V2 _netP02 _ref U="1V"
  R:R2 _netP02 _ref R="10k"
  Vdc:V3 _cnet0 _ref U="0"
  R:R3 _netP03 _ref R="10k"
  Vdc:V4 _cnet1 _ref U="0"
  R:R4 _netP04 _ref R="10k"
  Vdc:V5 _cnet2 _ref U="0"
  R:R5 _netP05 _ref R="10k"
  Vdc:V6 _cnet3 _ref U="0"
  R:R6 _netP06 _ref R="10k"
  Vdc:V7 _cnet4 _ref U="0"
  R:R7 _netP07 _ref R="10k"
  Vdc:V8 _cnet5 _ref U="5V"
  R:R8 _netP08 _ref R="10k"
  Vdc:V9 _cnet6 _ref U="−5"
  R:R9 _netP09 _ref R="10k"
  Vdc:V10 _cnet7 _ref U="5"
  R:R10 _netP10 _ref R="10k"
  Vdc:V11 _cnet8 _ref U="−10"
  R:R11 _netP11 _ref R="10k"
  .Def:End
  Sub:X1 _ref _netP01 _netP02 _netP03 _netP04 _netP05 _netP06
_netP07 _netP08 _netP09 _netP10 _netP11 Type="S2Q_TEST2"
.Def:End


.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA" vntol="1_uV"
saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
.TR:TR1 Type="lin" Start="0" Stop="5_ms" Points="2000"
IntegrationMethod="Gear" Order="6" InitialStep="1_ns" MinStep="1e−16"
MaxIter="150" reltol="0.001" abstol="100_pA" vntol="100_uV" Temp="26.85"
LTEreltol="1e−3" LTEabstol="1e−6" LTEfactor="1" Solver="CroutLU"
relaxTSR="no" initialDC="yes" MaxStep="0"
Sub:SUB1 vp1 vp2 vp3 vp4 vp5 vp6 vp7 vp8 vp9 vp10 vp11 Type="S2Q_test2"
```

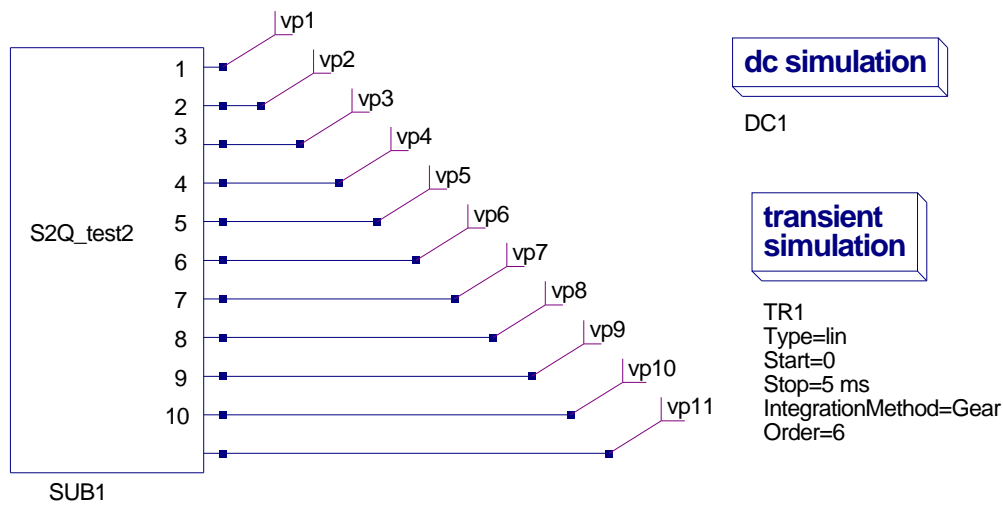Figure 9.2: March 11: Qucs netlist showing V3 error [Edited to fit on page width]

Figure 9.3: SPICE to Qucs conversion: Test2 simulation schematic

10. Test 10 : Vp10.Vt; Pass - see note 2 below.

11. Test 11 : Vp11.Vt; Pass - see note 3 below.

1. The SPICE SIN generator assumes that the frequency of the generated sinusoidal signal equals 1/TSTOP if not explicitly given. Hence, in such cases a .TRAN statement must be present in the simulated SPICE netlist; if a .TRAN statement is not included a default frequency of f = 1GHz is used. Also, when setting the transient simulation parameters using a Qucs transient analysis icon turn off SPICE simulation in the Edit SPICE Properties dialog box, otherwise two transient simulations are undertaken by Qucs.

2. SPICE parameter TD is treated differently by Qucs. In SPICE TD is the time from 0 seconds before the sinusoidal signal starts, causing the sinusoid to be non-linear. In Qucs TD is implemented as a phase shift of a linear sinusoidal signal. In the test example TD is 0.5m seconds which at f = 1kHz gives a phase shift of 180°and is clearly visible in the test results. An error will probably occur if TD is greater than one signal period, 1m second in the test example.

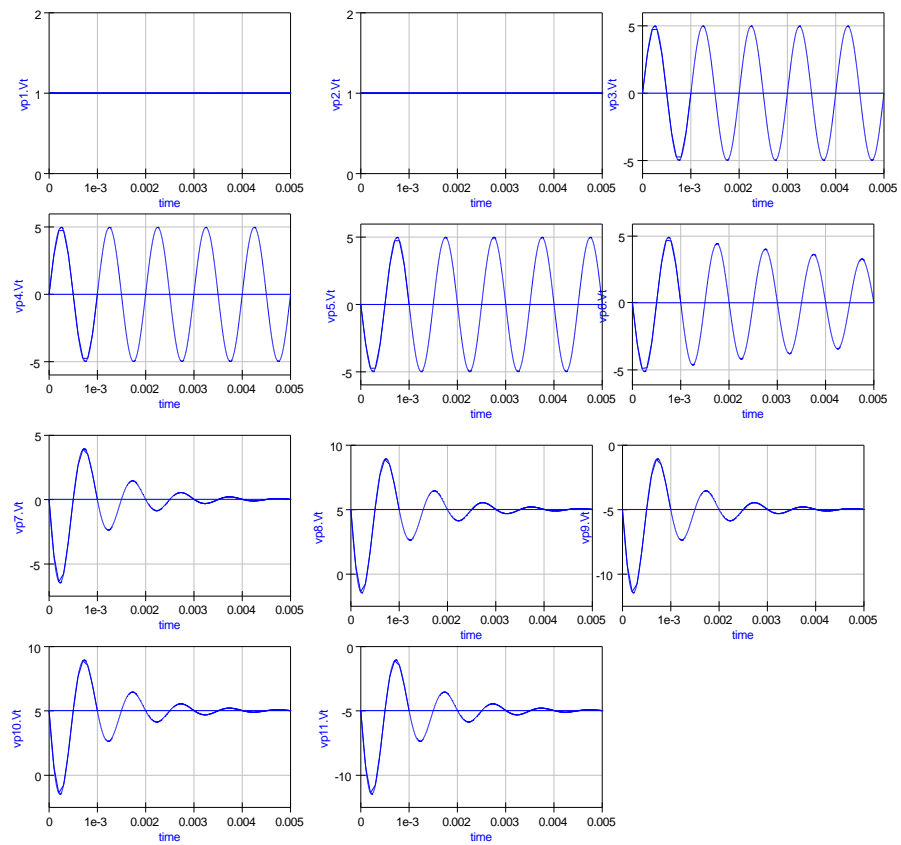3. Changes in CVS code have resulted in correct DC levels.

Figure 9.4: March 12: SPICE to Qucs conversion: Test2 waveforms

```
# Qucs 0.0.11   /media/hda2/S2Q_test2_prj/S2Q(test2).sch

.Def:S2Q_test2 _net0 _net1 _net2 _net3 _net4 _net5 _net6
_net7 _net8 _net9 _net10
Sub:X1 _net0 _net1 _net2 _net3 _net4 _net5 _net6 _net7
_net8 _net9 _net10 gnd Type="S2Q_test2_cir"
.Def:End

.Def:S2Q_test2_cir _netP01 _netP02 _netP03 _netP04 _netP05
_netP06 _netP07 _netP08 _netP09 _netP10 _netP11 _ref
  .Def:S2Q_TEST2 _ref _netP01 _netP02 _netP03 _netP04 _netP05
_netP06 _netP07 _netP08 _netP09 _netP10 _netP11
  Vac:V11 _netP11 _cnet8 U="5" f="1k" Phase="-180" Theta="1"
  Vac:V10 _netP10 _cnet7 U="5" f="1k" Phase="-180" Theta="1"
  Vac:V9 _netP09 _cnet6 U="5" f="1k" Phase="-180" Theta="1"
  Vac:V8 _netP08 _cnet5 U="5" f="1k" Phase="-180" Theta="1"
  Vac:V7 _netP07 _cnet4 U="5" f="1k" Phase="-180" Theta="1"
  Vac:V6 _netP06 _cnet3 U="5" f="1k" Phase="-180" Theta="0.1"
  Vac:V5 _netP05 _cnet2 U="5" f="1k" Phase="-180" Theta="0"
  Vac:V4 _netP04 _cnet1 U="5" f="1k" Phase="-0" Theta="0"
  Vac:V3 _netP03 _cnet0 U="5" Phase="-0" Theta="0" f="1000"
  Vdc:V1 _netP01 _ref U="1V"
  R:R1 _netP01 _ref R="10k"
  Vdc:V2 _netP02 _ref U="1V"
  R:R2 _netP02 _ref R="10k"
  Vdc:V3 _cnet0 _ref U="0"
  R:R3 _netP03 _ref R="10k"
  Vdc:V4 _cnet1 _ref U="0"
  R:R4 _netP04 _ref R="10k"
  Vdc:V5 _cnet2 _ref U="0"
  R:R5 _netP05 _ref R="10k"
  Vdc:V6 _cnet3 _ref U="0"
  R:R6 _netP06 _ref R="10k"
  Vdc:V7 _cnet4 _ref U="0"
  R:R7 _netP07 _ref R="10k"
  Vdc:V8 _cnet5 _ref U="5"
  R:R8 _netP08 _ref R="10k"
  Vdc:V9 _cnet6 _ref U="-5"
  R:R9 _netP09 _ref R="10k"
  Vdc:V10 _cnet7 _ref U="5"
  R:R10 _netP10 _ref R="10k"
  Vdc:V11 _cnet8 _ref U="-5"
  R:R11 _netP11 _ref R="10k"
  .Def:End
  Sub:X1 _ref _netP01 _netP02 _netP03 _netP04 _netP05 _netP06
_netP07 _netP08 _netP09 _netP10 _netP11 Type="S2Q_TEST2"
.Def:End

.TR:TRAN Points="11" Stop="1ms" Type="lin" Start="0"
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA" vntol="1_uV"
saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
.TR:TR1 Type="lin" Start="0" Stop="5_ms" Points="2000"
IntegrationMethod="Gear" Order="6" InitialStep="1_ns" MinStep="1e-16"
 MaxIter="150" reltol="0.001" abstol="100_pA" vntol="100_uV" Temp="26.85"
 LTEreltol="1e-3" LTEabstol="1e-6" LTEfactor="1" Solver="CroutLU"
relaxTSR="no" initialDC="yes" MaxStep="0"
Sub:SUB1 vp1 vp2 vp3 vp4 vp5 vp6 vp7 vp8 vp9 vp10 vp11 Type="S2Q_test2"
```

Figure 9.5: March 12: Qucs netlist showing V3 error [Edited to fit on page width]

9.4 References

1. A. Vladimirescu, Kaihe Zhang, A.R. Newton, D.O Pederson A. Sangiovanni-Vincentelli, SPICE 2G User's Guide (10 Aug 1981), Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca., 94720.

2. B. Johnson, T. Quarles, A.R. Newton, P.O. Pederson, A.Sangiovanni-Vincentelli, SPICE3 Version 3f User's Manual (October 1972), Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca., 94720.

3. Andrei Vladimirescu, THE SPICE book,1994, John Wiley and Sons. Inc., ISBN 0-471-609-26-9.

10 SPICE to Qucs conversion: Test File 3

10.1 Introduction

10.1.1 Title

SPICE 2g6 and 3f5 resistors.

10.1.2 SPICE specification

Format: SPICE 2g6[1]:   **RX N+ N- value [TC=TC1 [,TC2]]**

Notes:

1. Characters [and] enclose optional items

2. Resistors begin with letter R.

3. X denotes name of resistor.

4. N+ and N- are the positive and negative nodes respectively.

5. Equations:

   $TNOM$ = Nominal temperature; default 27°C.
   $\Delta T = T - TNOM$
   $R(T) = R(TNOM) + [1 + TC1 \cdot \Delta T + TC2 \cdot \Delta T^2]$

Format: SPICE 3f5[2]:

1. Standard resistors: **RX N+ N- value**

2. Semiconductor resistors:

   RX N+ N- [value] [mname] [L=length] [W=width] [TEMP=T]

---

[1]See section 6.1, SPICE 2g6 user's guide.
[2]See sections 3.1.1 and 3.1.2, SPICE 3f6 user's guide.

Notes:

1. Characters [and] enclose optional items

2. Resistors begin with letter R.

3. X denotes name of resistors.

4. N+ and N- are the positive and negative nodes respectively.

5. mname; if specified the resistance is calculated from the process information given in entry .model mname.

6. L is the length of the resistor.

7. W is the width of the resistor.

8. mname .model type R parameters:
   - TC1 : First order temperature coefficient; default $0.0. \Omega/°C$.
   - TC2 : Second order temperature coefficient; default 0.0. $\Omega/°C^2$.
   - RSH : Sheet resistance; default -. Ω per square.
   - DEFW : Default width; default 1e-6m.
   - NARROW : Narrowing due to side etching; default 0.0m.
   - TNOM : Nominal temperature; default 27 $°C$.

9. Equations $R = RSH \cdot \dfrac{L - NARROW}{W - NARROW}$

   $R(T) = R(TNOM) + [1 + TC1 \cdot \Delta T + TC2 \cdot \Delta T^2]$

   Where $\Delta T = T - T_0$: T is the circuit temperature and T_0 the nominal temperature.

10.2 Test code and schematic

SPICE code: File `S2Q_test3_a.cir`

```
* SPICE to Qucs syntax test file
* SPICE 2g6 resistors.
*
.subckt S2Q_test3_a p01 p02 p03
v1 1 0 DC 1v
```

```
r1  1  p01  10k
r2  p01  0  10k
*
v2  2  0  dc  1v
r3  2  p02   10k  tc=0.01
r4  p02  0   10k
*
v3  3  0  Dc  1v
r5  3  p03   10k  tc=0.01  0.015
r6  p03  0  10k
.ends
.end
```
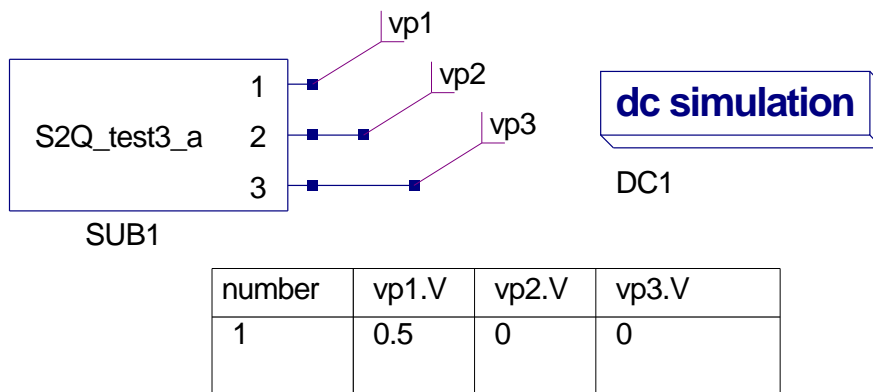
Figure 10.1: March 13: SPICE to Qucs conversion: Test3 schematic plus output
         table for SPICE 2g6 test1: linear resistors

SPICE code: File `S2Q_test3_b.cir`

```
* SPICE to Qucs syntax test file
* SPICE 3f5 resistors.
*
.subckt S2Q_test3_a p01 p02 p03
v1 1 0 DC 1v
r1 1 p01 10k
r2 p01 0 10k
*
v2 2 0 dc 1v
r3 2 p02  RMOD1 L=10u W=1u
r4 p02 0  10k
.model RMOD1 R(RSH=50 DEFW=2e-6 NARROW=1e-7)
*
.ends
.end
```

10.3 History of simulation results

10.3.1 March 13 2007, Simulation tests by Mike Brinson

A: SPICE 2g6 tests:

Test 1 [v2, r3, v3, and r5 commented]: Linear resistors: Vp1: PASS correct output voltage.

Test 2 [v3, and r5 commented]: SPICE 2g6 resistor with first order temperature coefficient: **FAIL** - inncorrect Qucs netlist.

Qucs netlist line 14: checker error, extraneous property 'TC' is invalid in 'R:R3'

Qucs netlist:

```
# Qucs 0.0.11  /media/hda2/S2Q_test3_prj/S2Q(test3_a).sch

.Def:S2Q_test3_a _net0 _net1 _net2
Sub:X1 _net0 _net1 _net2 gnd Type="S2Q_test3_a_cir"
.Def:End

.Def:S2Q_test3_a_cir _netP01 _netP02 _netP03 _ref
  .Def:S2Q_TEST3_A _ref _netP01 _netP02 _netP03
  Vdc:V1 _net1 _ref U="1V"
  R:R1 _net1 _netP01 R="10k"
  R:R2 _netP01 _ref R="10k"
  Vdc:V2 _net2 _ref U="1V"
  R:R3 _net2 _netP02 R="10k" TC="0.01"
  R:R4 _netP02 _ref R="10k"
  R:R6 _netP03 _ref R="10k"
  .Def:End
  Sub:X1 _ref _netP01 _netP02 _netP03 Type="S2Q_TEST3_A"
.Def:End


Sub:SUB1 vp1 vp2 vp3 Type="S2Q_test3_a"
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1 pA" vntol="1 uV"
 saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
```

Test 3 [v2, r2, and r3 commented]: **FAIL** - netlist not passed correctly.
Qucs error message: line 14: syntax error, unexpected Floats, expecting Eol.

Qucs netlist:

```
# Qucs 0.0.11  /media/hda2/S2Q_test3_prj/S2Q(test3_a).sch
```

```
.Def:S2Q_test3_a _net0 _net1 _net2
Sub:X1 _net0 _net1 _net2 gnd Type="S2Q_test3_a_cir"
.Def:End

.Def:S2Q_test3_a_cir _netP01 _netP02 _netP03 _ref
.Def:End


Sub:SUB1 vp1 vp2 vp3 Type="S2Q_test3_a"
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA"
vntol="1_uV" saveOPs="no" MaxIter="150" saveAll="no"
convHelper="none" Solver="CroutLU"
```

B: SPICE 3f5 tests
Test1: SPICE file `S2Q_test3_b.cir` fails to convert to Qucs netlist format,
giving the following error message:
line10 : syntax error, unexpected identifier, expecting Digits or Floats.
SPICE 3f5 Semiconductor resistors not implemented?

10.3.2 March 15 2007, Simulation tests by Mike Brinson

Qucs CVS code modifications:

- * `scan_spice.l`: Lexer modifications for the Spice 2g6 resistor syntax were
  necessary. Stefan Jahn

- * `parse_spice.y`: Allow Spice 2g6 syntax for resistors, also fixed netlist
  grammar for Spice 3f5 models. Stefan Jahn

- * `check_spice.cpp`: Handle R semiconductor model correctly as well as
  the Spice 2g6 syntax for the temperature coefficients. Stefan Jahn

SPICE code: File `S2Q_test3_a.cir`

- Vp01.V: **PASS**; correct dc output.

- Vp02.V: **PASS**; correct dc output for TEMP=TNOM.

- Vp03.V: **PASS**; correct dc output for TEMP=TNOM.

NOTES:

- The Vp02 and Vp03 test results are only correct for TEMP=TNOM.

- SPICE 2g6 simulates circuit performance at temperature set by the value of TNOM; 27 °C by default.

- Circuits can be simulated at other temperatures by using a .TEMP control statement; which has the format

  .TEMP T1 [T2 [T3 .....]]

  Unfortunately the Qucsconv program does not recognise this statement so there appears to be no way of changing the temperature of SPICE 2g6 resistors that have TC1 and TC2 temperature coefficients in their netlist entries.

- Adding SPICE 2g6 statement .TEMP 50 to file `S2Q_test3_a.cir` gives the following error:

  spice notice, no .END directive found, continuing line 18: syntax error, unexpected Identifier, expecting `$end`

Qucs netlist:

```
# Qucs 0.0.11  /media/hda2/S2Q_test3_prj/S2Q(test3_a).sch

.Def:S2Q_test3_a _net0 _net1 _net2
Sub:X1 _net0 _net1 _net2 gnd Type="S2Q_test3_a_cir"
.Def:End

.Def:S2Q_test3_a_cir _netP01 _netP02 _netP03 _ref
  .Def:S2Q_TEST3_A _ref _netP01 _netP02 _netP03
  Vdc:V1 _net1 _ref U="1V"
  R:R1 _net1 _netP01 R="10k"
  R:R2 _netP01 _ref R="10k"
  Vdc:V2 _net2 _ref U="1V"
  R:R3 _net2 _netP02 R="10k" Tc1="0.01"
  R:R4 _netP02 _ref R="10k"
  Vdc:V3 _net3 _ref U="1V"
  R:R5 _net3 _netP03 R="10k" Tc1="0.01" Tc2="0.015"
  R:R6 _netP03 _ref R="10k"
  .Def:End
  Sub:X1 _ref _netP01 _netP02 _netP03 Type="S2Q_TEST3_A"
.Def:End

.DC:DC1 Temp="50" reltol="0.001" abstol="1_pA" vntol="1_uV"
 saveOPs="no" MaxIter="150" saveAll="no" convHelper="none"
 Solver="CroutLU"
```
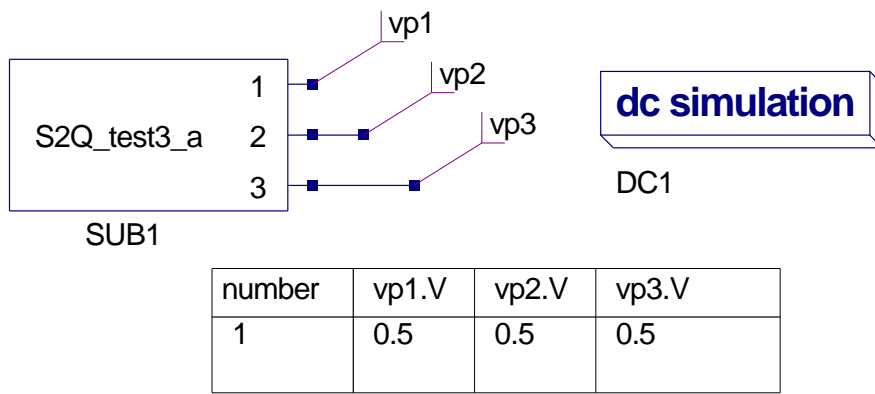
Figure 10.2: March 15: SPICE to Qucs conversion: SPICE 2g6 resistor schematic plus output table

Sub:SUB1 vp1 vp2 vp3 Type="S2Q_test3_a"

SPICE code: File `S2Q_test3_b.cir`

- Vp01.V: **PASS**; correct dc output.

- Vp02.V: **PASS**; correct dc output.

Qucs netlist:

Qucs 0.0.11   /media/hda2/S2Q_test3_prj/S2Q(test3_b).sch

```
.Def:S2Q_test3_b _net0 _net1
Sub:X1 _net0 _net1 gnd Type="S2Q_test3_b_cir"
.Def:End


.Def:S2Q_test3_b_cir _netP01 _netP02 _ref
  .Def:S2Q_TEST3_A _ref _netP01 _netP02
  Vdc:V1 _net1 _ref U="1V"
  R:R1 _net1 _netP01 R="10k"
  R:R2 _netP01 _ref R="10k"
  Vdc:V2 _net2 _ref U="1V"
  R:R3 _net2 _netP02 R="550"
  R:R4 _netP02 _ref R="10k"
  .Def:End
  Sub:X1 _ref _netP01 _netP02 Type="S2Q_TEST3_A"
.Def:End



Sub:SUB1 vp01 vp02 Type="S2Q_test3_b"
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1 pA" vntol="1 uV"
 saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
```

SPICE code: File `S2Q_test3_c.cir`

```
* SPICE to Qucs syntax test file
* SPICE 3f5 resistors : Temperature effects.
*
.subckt S2Q_test3_a p01 p02 p03 p04 p05 p06 p07 p08 p09 p10
v1 1 0 DC 1v
r1 1 p01 10k
r2 p01 0 10k
*
v2 2 0 dc 1v
r3 2 p02  10k  RMOD1
r4 p02 0  10k
.model RMOD1 R(TC1=0.01 TC2=0.015)
```

```
*
v3  3  0  dc  1v
r5  3  p03    10k RMOD1 TEMP=30
r6  p03 0  10k
*
v4  4  0  dc  1v
r7  4  p04    10k RMOD1 TEMP=40
r8  p04 0  10k
*
v5  5  0  dc  1v
r9  5  p05    10k RMOD1 TEMP=50
r10 p05 0  10k
*
v6  6  0  dc  1v
r11 6  p06    10k RMOD1 TEMP=60
r12 p06 0  10k
*
v7  7  0  dc  1v
r13 7  p07    10k RMOD1 TEMP=70
r14 p07 0  10k
*
v8  8  0  dc  1v
r15 8  p08    10k RMOD1 TEMP=80
r16 p08 0  10k
*
v9  9  0  dc  1v
r17 9  p09    10k RMOD1 TEMP=90
r18 p09 0  10k
*
v10 10 0  dc  1v
r19 10 p10    10k RMOD1 TEMP=100
r20 p10  0  10k
.ends
.end
```

- Vp01.V: **PASS**; correct dc output.

- Vp02.V: **PASS**; correct dc output.

- Vp03.V: **PASS**; correct dc output.

- Vp04.V: **PASS**; correct dc output.

- Vp05.V: **PASS**; correct dc output.

- Vp06.V: **PASS**; correct dc output.

- Vp07.V: **PASS**; correct dc output.

- Vp08.V: **PASS**; correct dc output.

- Vp09.V: **PASS**; correct dc output.

- Vp20.V: **PASS**; correct dc output.

NOTES:

- In SPICE 3f5 TEMP values attached to resistors override the global value of TEMP.

- SPICE 3f5 differs from SPICE 2g6 in that it does not allow the control statement .TEMP.

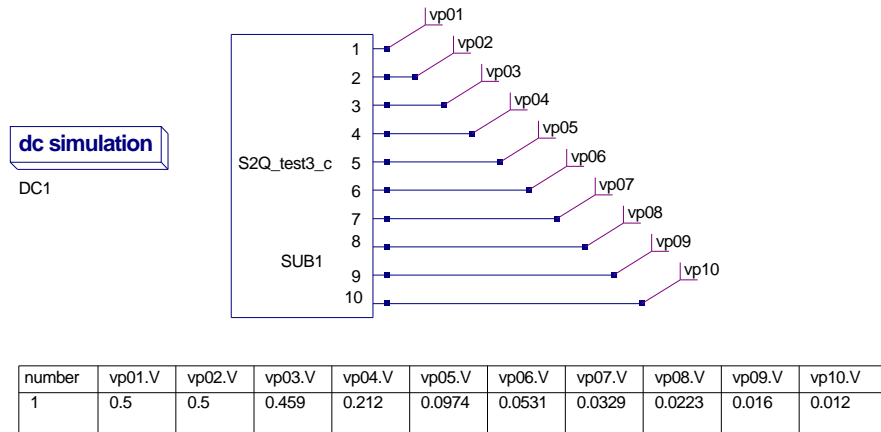| number	vp01.V	vp02.V	vp03.V	vp04.V	vp05.V	vp06.V	vp07.V	vp08.V	vp09.V	vp10.V
1	0.5	0.5	0.459	0.212	0.0974	0.0531	0.0329	0.0223	0.016	0.012

Figure 10.3: March 15: SPICE to Qucs conversion: Test3 schematic plus output
for SPICE 3f5 test c

Qucs netlist:

```
# Qucs 0.0.11  /media/hda2/S2Q_test3_prj/s2Q(test3_c).sch

.Def:S2Q_test3_c _net0 _net1 _net2 _net3 _net4 _net5 _net6 _net7 _net8 _net
Sub:X1 _net0 _net1 _net2 _net3 _net4 _net5 _net6 _net7 _net8 _net9
 gnd Type="S2Q_test3_c_cir"
.Def:End


.Def:S2Q_test3_c_cir _netP01 _netP02 _netP03 _netP04 _netP05
_netP06 _netP07 _netP08 _netP09 _netP10 _ref
  .Def:S2Q_TEST3_A _ref _netP01 _netP02 _netP03 _netP04
_netP05 _netP06 _netP07 _netP08 _netP09 _netP10
  Vdc:V1 _net1 _ref U="1V"
  R:R1 _net1 _netP01 R="10k"
  R:R2 _netP01 _ref R="10k"
  Vdc:V2 _net2 _ref U="1V"
  R:R3 _net2 _netP02 R="10k" Tc1="0.01" Tc2="0.015"
  R:R4 _netP02 _ref R="10k"
  Vdc:V3 _net3 _ref U="1V"
  R:R5 _net3 _netP03 R="10k" Temp="30" Tc1="0.01" Tc2="0.015"
  R:R6 _netP03 _ref R="10k"
  Vdc:V4 _net4 _ref U="1V"
  R:R7 _net4 _netP04 R="10k" Temp="40" Tc1="0.01" Tc2="0.015"
  R:R8 _netP04 _ref R="10k"
  Vdc:V5 _net5 _ref U="1V"
  R:R9 _net5 _netP05 R="10k" Temp="50" Tc1="0.01" Tc2="0.015"
```

```
R:R10 _netP05 _ref R="10k"
Vdc:V6 _net6 _ref U="1V"
R:R11 _net6 _netP06 R="10k" Temp="60" Tc1="0.01" Tc2="0.015"
R:R12 _netP06 _ref R="10k"
Vdc:V7 _net7 _ref U="1V"
R:R13 _net7 _netP07 R="10k" Temp="70" Tc1="0.01" Tc2="0.015"
R:R14 _netP07 _ref R="10k"
Vdc:V8 _net8 _ref U="1V"
R:R15 _net8 _netP08 R="10k" Temp="80" Tc1="0.01" Tc2="0.015"
R:R16 _netP08 _ref R="10k"
Vdc:V9 _net9 _ref U="1V"
R:R17 _net9 _netP09 R="10k" Temp="90" Tc1="0.01" Tc2="0.015"
R:R18 _netP09 _ref R="10k"
Vdc:V10 _net10 _ref U="1V"
R:R19 _net10 _netP10 R="10k" Temp="100" Tc1="0.01" Tc2="0.015"
R:R20 _netP10 _ref R="10k"
.Def:End
Sub:X1 _ref _netP01 _netP02 _netP03 _netP04 _netP05 _netP06 _netP07
_netP08 _netP09 _netP10 Type="S2Q_TEST3_A"
.Def:End


Sub:SUB1 vp01 vp02 vp03 vp04 vp05 vp06 vp07 vp08 vp09 vp10 Type="S2Q_test3_c"
.DC:DC1 Temp="26.8" reltol="0.001" abstol="1_pA" vntol="1_uV" saveOPs="no"
MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
```

10.3.3 March 18 2007, Simulation tests by Mike Brinson

More SPICE 3f5 Temperature tests.
SPICE code: File `S2Q_test3_f.cir`

```
* SPICE to Qucs syntax test file
* SPICE 3f5 resistors.
* Temperature tests.
*
.subckt S2Q_test3_f p01 p02 p03 p04
v1 1 0 DC 1v
r1 1 p01 10k
r2 p01 0 10k
*
*
v2 2 0 dc 1v
r3 2 p02  10k  RMOD1 TEMP=50
```

```
r4  p02  0    10k
.model RMOD1 R(TC1=0.01 TC2=0.015)
*
v3  3  0  dc  1v
r5  3  p03   10k   RMOD1
r6  p03  0    10k
.model RMOD1 R(TC1=0.01 TC2=0.015 TNOM=100)
*
v4  4  0  dc  1v
r7  4  p04   10k   RMOD1
r8  p04  0    10k
.model RMOD1 R(TC1=0.01 TC2=0.015)
*
.ends
.OPTION TNOM=40
.end
```

- Vp01.V: **PASS**; correct dc output.

- Vp02.V: **PASS**; correct dc output.

- Vp03.V: **FAIL**; dc output does not change with changes in .OPTION
  TEMP or TNOM.

- Vp04.V: **FAIL**; dc output does not change with changes in .OPTION
  TEMP or TNOM.

NOTES:

- In SPICE 3f5 TEMP values attached to resistors override the global value
  of circuit temperature

- Output Vp02 is correct the value of R3 being determined by TEMP=50,
  TC1=0.01 and TC2=0.015 during the resistance calculation.

- Using SPICE statements .OPTION TNOM=XX or .OPTION TEMP=XX
  appears to have no effect on resistance calculations.

- Similarly, adding TNOM=XX to a R model appears not to affect the
  resistance calculations.
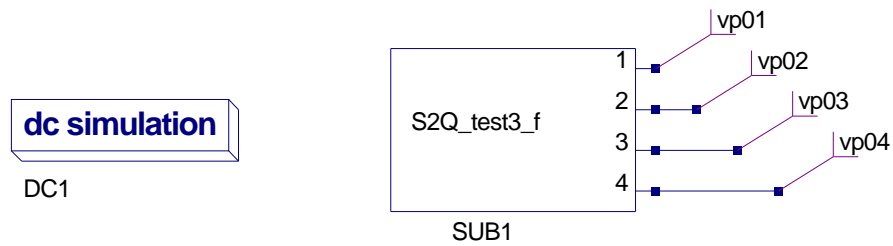
Figure 10.4: March 18: SPICE to Qucs conversion: Test3 schematic plus output
for SPICE 3f5 test f

Qucs netlist:

```
# Qucs 0.0.11  /media/hda2/S2Q_test3_prj/S2Q(test3_f).sch

.Def:S2Q_test3_f _net0 _net1 _net2 _net3
Sub:X1 _net0 _net1 _net2 _net3 gnd Type="S2Q_test3_f_cir"
.Def:End

.Def:S2Q_test3_f_cir _netP01 _netP02 _netP03 _netP04 _ref
  .Def:S2Q_TEST3_F _ref _netP01 _netP02 _netP03 _netP04
  Vdc:V1 _net1 _ref U="1V"
  R:R1 _net1 _netP01 R="10k"
  R:R2 _netP01 _ref R="10k"
  Vdc:V2 _net2 _ref U="1V"
  R:R3 _net2 _netP02 R="10k" Temp="50" Tc1="0.01" Tc2="0.015"
  R:R4 _netP02 _ref R="10k"
  Vdc:V3 _net3 _ref U="1V"
  R:R5 _net3 _netP03 R="10k" Tc1="0.01" Tc2="0.015"
  R:R6 _netP03 _ref R="10k"
  Vdc:V4 _net4 _ref U="1V"
  R:R7 _net4 _netP04 R="10k" Tc1="0.01" Tc2="0.015"
  R:R8 _netP04 _ref R="10k"
  .Def:End
  Sub:X1 _ref _netP01 _netP02 _netP03 _netP04 Type="S2Q_TEST3_F"
.Def:End

Sub:SUB1 vp01 vp02 vp03 vp04 Type="S2Q_test3_f"
```

217

```
.DC:DC1 Temp="26.8" reltol="0.001" abstol="1_pA" vntol="1_uV"
saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
```

10.4 March 25 2007, Simulation tests by Mike Brinson

Code modifications:

- * `scan_spice.l, parse_spice.y`: Accept .TEMP syntax (Spice 2g6) in lexer and parser. Stefan Jahn.

SPICE code: File `S2Q_test3_a.cir`
SPICE statements .OPTION TNOM=50 and .TEMP 50 are now accepted in the SPICE to Qucs translation process. HOWEVER, at the moment (Qucs 0.0.12) global circuit temperature parameters are not implemented in the Qucs simulator and cannot be changed via these statements.

SPICE code: File `S2Q_test3_f.cir`
SPICE code modification due to test bug caused by RMOD1 being referenced in each test.
New code:

```
* SPICE to Qucs syntax test file
* SPICE 3f5 resistors.
* Temperature tests.
*
.subckt S2Q_test3_f p01 p02 p03 p04
v1 1 0 DC 1v
r1 1 p01 10k
r2 p01 0 10k
v2 2 0 dc 1v
r3 2 p02 10k RMOD1 TEMP=50
r4 p02 0 10k
.model RMOD1 R(TC1=0.01 TC2=0.015)
v3 3 0 dc 1v
r5 3 p03 10k RMOD2
r6 p03 0 10k
.model RMOD2 R(TC1=0.01 TC2=0.015 TNOM=100)
v4 4 0 dc 1v
r7 4 p04 10k RMOD3
r8 p04 0 10k
.model RMOD3 R(TC1=0.01 TC2=0.015)
```
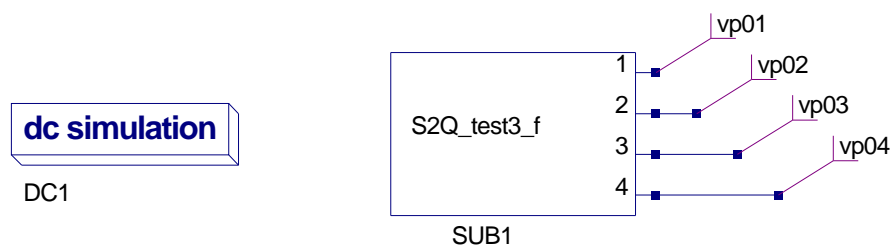
number	vp01.V	vp02.V	vp03.V	vp04.V
1	0.5	0.0974	0.0123	0.5

Figure 10.5: March 25: SPICE to Qucs conversion: Test3 schematic plus output for SPICE 3f5 test f

```
.ends
.OPTION TNOM=40
.end
```

Qucs netlist:

```
# Qucs 0.0.12  /media/hda2/S2Q_test3_prj/S2Q(test3_f).sch

.Def:S2Q_test3_f _net0 _net1 _net2 _net3
Sub:X1 _net0 _net1 _net2 _net3 gnd Type="S2Q_test3_f_cir"
.Def:End

.Def:S2Q_test3_f_cir _netP01 _netP02 _netP03 _netP04 _ref
  .Def:S2Q_TEST3_F _ref _netP01 _netP02 _netP03 _netP04
  Vdc:V1 _net1 _ref U="1V"
  R:R1 _net1 _netP01 R="10k"
  R:R2 _netP01 _ref R="10k"
  Vdc:V2 _net2 _ref U="1V"
  R:R3 _net2 _netP02 R="10k" Temp="50" Tc1="0.01" Tc2="0.015"
  R:R4 _netP02 _ref R="10k"
  Vdc:V3 _net3 _ref U="1V"
  R:R5 _net3 _netP03 R="10k" Tc1="0.01" Tc2="0.015" Tnom="100"
  R:R6 _netP03 _ref R="10k"
  Vdc:V4 _net4 _ref U="1V"
  R:R7 _net4 _netP04 R="10k" Tc1="0.01" Tc2="0.015"
  R:R8 _netP04 _ref R="10k"
  .Def:End
  Sub:X1 _ref _netP01 _netP02 _netP03 _netP04 Type="S2Q_TEST3_F"
.Def:End


Sub:SUB1 vp01 vp02 vp03 vp04 Type="S2Q_test3_f"
.DC:DC1 Temp="26.8" reltol="0.001" abstol="1_pA" vntol="1_uV"
saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
```

- Vp01.V: **PASS**; correct dc output.

- Vp02.V: **PASS**; correct dc output.

- Vp03.V: **PASS**; correct dc output.

- Vp04.V: **FAIL**; dc output does not change with changes in .OPTION
  TNOM = 40. The Qucs simulator does NOT implement a global
  temperature parameter but allocates separate temperatures to each
  component.

NOTES:

- In SPICE 3f5 TEMP values attached to resistors override the global value of circuit temperature

- Output Vp02 is correct, the value of R3 being determined by TEMP=50, TC1=0.01 and TC2=0.015.

- Output Vp03 is correct, the value of R5 being determined by TNOM=100, TC1=0.01 and TC2=0.015.

- Output VP04 is 0.5 because .OPTION TNOM=40 has no effect on the temperature of resistor R8. By default the temperature of R8 is set to $26.85°C$.

10.5 References

1. A. Vladimirescu, Kaihe Zhang, A.R. Newton, D.O Pederson A. Sangiovanni-Vincentelli, SPICE 2G User's Guide (10 Aug 1981), Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca., 94720.

2. B. Johnson, T. Quarles, A.R. Newton, P.O. Pederson, A.Sangiovanni-Vincentelli, SPICE3 Version 3f User's Manual (October 1972), Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca., 94720.

3. Andrei Vladimirescu, THE SPICE book,1994, John Wiley and Sons. Inc., ISBN 0-471-609-26-9.

11 SPICE to Qucs conversion: Test File 4

11.1 Introduction

11.1.1 Title

SPICE 2g6 and 3f5 capacitors.

11.1.2 SPICE specification

Format: SPICE 2g6[1]:

1. Linear form: **CX N+ N- value [IC = INCOND]**

2. Nonlinear form: **CX N+ N- [POLY] value [C1 [C2 ....]] [IC = INCOND]**

Notes:

1. Characters [and] enclose optional items

2. Capacitors begin with letter C.

3. X denotes name of capacitor

4. N+ and N- are the positive and negative nodes respectively.

5. Equations:

   Capacitors may be nonlinear functions of voltage, where
   $C(V) = value + C1 \cdot V + C2 \cdot V^2 + ...Cn \cdot V^n$

Format: SPICE 3f5[2]:

---

[1]See section 6.2, SPICE 2g6 user's guide.
[2]See sections 3.1.4 and 3.1.5, SPICE 3f6 user's guide.

1. Linear capacitors: **CX N+ N- value [IC = INCOND]**

2. Semiconductor capacitors:

   CX N+ N- [value] [mname] [L=length] [W=width] [IC=VAL]

Notes:

1. Characters [and] enclose optional items

2. Capacitors begin with letter C.

3. X denotes name of capacitor

4. N+ and N- are the positive and negative nodes respectively.

5. mname; if specified the capacitance is calculated from the process information given in entry .model mname.

6. L is the length of the capacitor.

7. W is the width of the capacitor.

8. mname .model type C parameters:

   - CJ : Junction bottom capacitance; default -.$F/meters^2$.
   - CJSW : Junction sidewall capacitance; default - $F/meters^2$.
   - DEFW : Default width; default 1e-6 $meters$.
   - NARROW : Narrowing due to side etching; default 0.0 $meters$.

9. Equations:
   $CAP =$
   $CJ \cdot (L - NARROW) \cdot (W - NARROW) + 2 \cdot CJSW \cdot (L + W - 2 \cdot NARROW)$

11.2 Test code and schematic

SPICE code: File `S2Q_test4_a.cir`

```
* SPICE to Qucs syntax test file
* SPICE 2g6 and 3f5 linear capacitors.
* DC and  AC tests.
*
.subckt S2Q_test4_a p01 p02 p03 p04 p05 p06
v1 1 0 AC 1v
r1 1 p01 10k
```

```
c1  p01  0  1u
v2  2  0  ac  1v
r3  2  p02   10k
c2  p02  0  1u  ic =10v
v3dc  3  0  dc  1v
v3ac  4  3  ac  1v
r4  4  p03   10k
c3  p03  0  1u
v4dc  5  0  dc  1v
v4ac  6  5  ac  1v
r5  6  p04   10k
c4  p04  0  1u  ic = 10v
v3  7  0  dc  1v  ac  1v
r6  7  p05   10k
c5  p05  0  1u
v4  8  0  dc  1v  ac  1v
r7  8  p06   10k
c6  p06  0  1u  ic = 10v
.ends
.end
```

SPICE code: File `S2Q_test4_b.cir`

```
* SPICE to Qucs syntax test file
* SPICE 2g6 and 3f5 linear capacitors.
* Pulse   tests.
*
.subckt S2Q_test4_b p01 p02 p03 p04 p05
v1  1  0  pulse(0 1 50ms 1us 1us 100ms 200ms)
r1  1  p01  10k
c1  p01  0  1u  ic=0v
v2  2  0  pulse(0 1 50ms 1us 1us 100ms 200ms)
r3  2  p02   10k
c2  p02  0  1u  ic = −1v
v3dc  3  0  1v
v3ac   4  3  pulse(0 1 50ms 1us 1us 100ms 200ms)
r4  4  p03   10k
c4  p03  0  1u  ic=0v
v4dc  5  0  dc  1v
v4ac  6  5  pulse(0 1 50ms 1us 1us 100ms 200ms)
r5  6  p04   10k
c5  p04  0  1u  ic = −1v
v5  7  0  dc  1v  pulse(0 1   50ms 1us 1us 100ms 200ms)
r6  7  p05   10k
c6  p05  0  1u  ic = −1v
.ends
.end
```
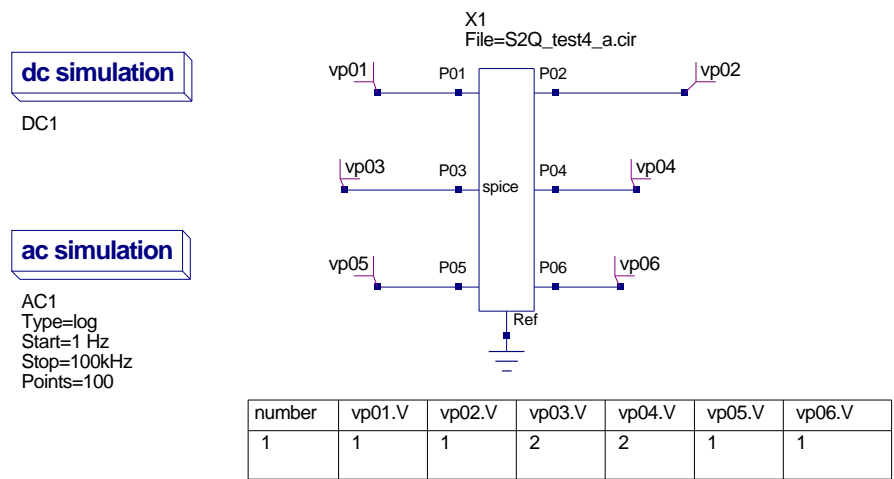
X1
File=S2Q_test4_a.cir

dc simulation

DC1

vp01   P01   P02   vp02

vp03   P03   spice   P04   vp04

ac simulation

AC1
Type=log
Start=1 Hz
Stop=100kHz
Points=100

vp05   P05   P06   vp06

Ref

number	vp01.V	vp02.V	vp03.V	vp04.V	vp05.V	vp06.V
1	1	1	2	2	1	1

Figure 11.1: March 22: SPICE to Qucs conversion: Test4 schematic plus dc output
table for SPICE 2g6 and 3f5 linear capacitors test

11.3 History of simulation results

11.3.1 March 22 2007, Simulation tests by Mike Brinson

A: SPICE 2g6 and 3f5 linear capacitor dc tests:

- Vp01.V: **FAIL**; correct dc output = 0V.

- Vp02.V: **FAIL**; correct dc output = 0V.

- Vp03.V: **FAIL**; correct dc output = 1V.

- Vp04.V: **FAIL**; correct dc output = 1V.

- Vp05.V: **FAIL**; correct dc output = 1V.

- Vp06.V: **FAIL**; correct dc output = 1V.

NOTE: It would appear that the value of a branch AC voltage source is being added to DC sources in the same branch during the DC simulation. This is incorrect. Qucs correctly computes the DC conditions from a schematic, see Fig. 11.2. ERROR in SPICE to Qucs conversion process.

dc simulation

DC1

ac simulation

AC1
Type=log
Start=1 Hz
Stop=10 MHz
Points=200

number	vp01.V
1	1

R1
R=10k

vp01

C1
C=1 uF

+ V2
U=1 V
−

Equation

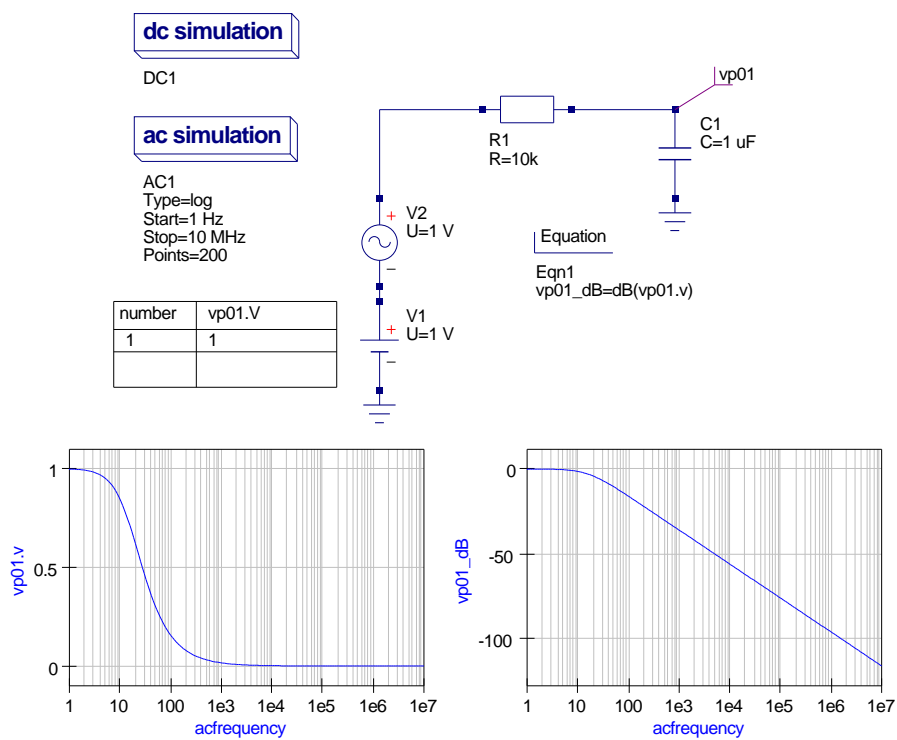Eqn1
vp01_dB=dB(vp01.v)

+ V1
U=1 V
−

Figure 11.2: March 22: Qucs RC simulation with series AC and DC sources

Qucs netlist:

```
# Qucs 0.0.11  /media/hda2/S2Q_test4_prj/S2Q(test4_a).sch

.Def:S2Q_test4_a_cir _netP01 _netP02 _netP03 _netP04 _netP05
 _netP06 _ref
  .Def:S2Q_TEST4_A _ref _netP01 _netP02 _netP03 _netP04
_netP05 _netP06
  Vac:V4 _net8 _cnet5 U="1V"
  Vac:V3 _net7 _cnet4 U="1V"
  Vac:V4AC _net6 _cnet3 U="1V"
  Vac:V3AC _net4 _cnet2 U="1V"
  Vac:V2 _net2 _cnet1 U="1V"
  Vac:V1 _net1 _cnet0 U="1V"
  Vdc:V1 _cnet0 _ref U="1"
  R:R1 _net1 _netP01 R="10k"
  C:C1 _netP01 _ref C="1u"
  Vdc:V2 _cnet1 _ref U="1"
  R:R3 _net2 _netP02 R="10k"
  C:C2 _netP02 _ref C="1u" V="10V"
  Vdc:V3DC _net3 _ref U="1V"
  Vdc:V3AC _cnet2 _net3 U="1"
  R:R4 _net4 _netP03 R="10k"
  C:C3 _netP03 _ref C="1u"
  Vdc:V4DC _net5 _ref U="1V"
  Vdc:V4AC _cnet3 _net5 U="1"
  R:R5 _net6 _netP04 R="10k"
  C:C4 _netP04 _ref C="1u" V="10V"
  Vdc:V3 _cnet4 _ref U="1V"
  R:R6 _net7 _netP05 R="10k"
  C:C5 _netP05 _ref C="1u"
  Vdc:V4 _cnet5 _ref U="1V"
  R:R7 _net8 _netP06 R="10k"
  C:C6 _netP06 _ref C="1u" V="10V"
  .Def:End
  Sub:X1 _ref _netP01 _netP02 _netP03 _netP04 _netP05 _netP06
 Type="S2Q_TEST4_A"
.Def:End


.DC:DC1 Temp="26.85" reltol="0.001" abstol="1 pA" vntol="1 uV"
saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
Sub:X1 vp01 vp02 vp03 vp04 vp05 vp06 gnd Type="S2Q_test4_a_cir"
```
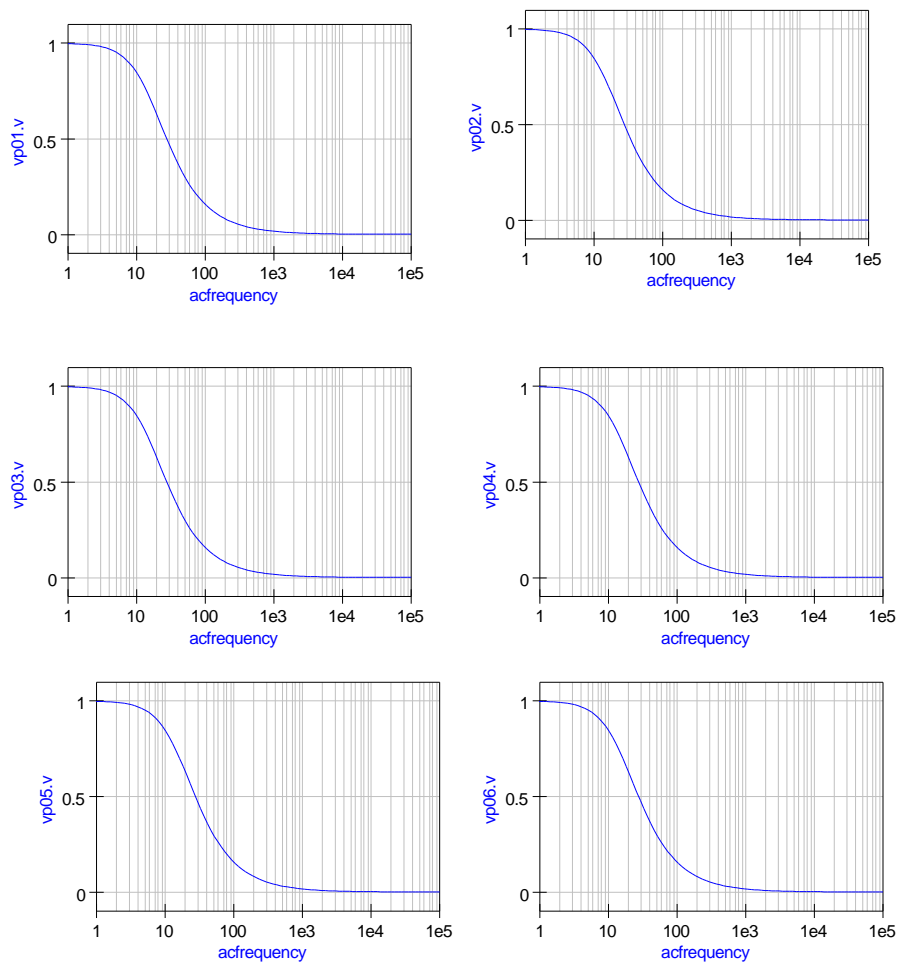
Figure 11.3: March 22: Output waveforms for ac tests

B: SPICE 2g6 and 3f5 linear capacitor ac tests:
Simulation waveforms appear to be correct, see Fig. 11.3.

C: SPICE 2g6 and 3f5 capacitor pulse tests.
SPICE code: File `S2Q_test4_b.cir`

Qucs netlist:

```
# Qucs 0.0.11  /media/hda2/S2Q_test4_prj/S2Q(test4_b).sch

.Def:S2Q_test4_b_cir _netP01 _netP02 _netP03 _netP04 _netP05 _ref
  .Def:S2Q_TEST4_B _ref _netP01 _netP02 _netP03 _netP04 _netP05
  Vrect:V5 _net7 _cnet4 U="1" Td="50ms" Tr="1us" Tf="1us"
  TH="0.100002" TL="0.049998"
```
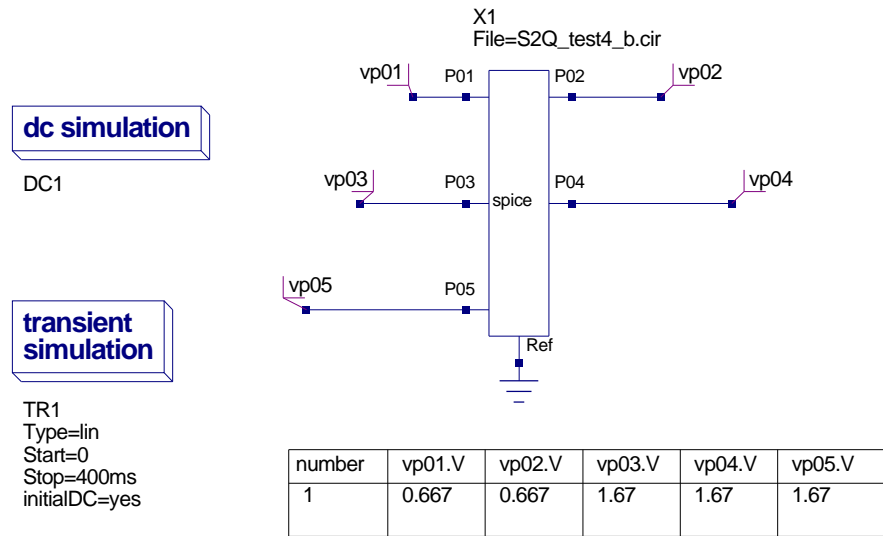
Figure 11.4: March 22: SPICE to Qucs conversion: Test4 schematic plus dc output
table for SPICE 2g6 and 3f5 linear capacitors pulse test

```
 Vrect:V4AC _net6 _cnet3 U="1" Td="50ms" Tr="1us" Tf="1us"
TH="0.100002" TL="0.049998"
 Vrect:V3AC _net4 _cnet2 U="1" Td="50ms" Tr="1us" Tf="1us"
TH="0.100002" TL="0.049998"
 Vrect:V2 _net2 _cnet1 U="1" Td="50ms" Tr="1us" Tf="1us"
TH="0.100002" TL="0.049998"
 Vrect:V1 _net1 _cnet0 U="1" Td="50ms" Tr="1us" Tf="1us"
TH="0.100002" TL="0.049998"
Vdc:V1 _cnet0 _ref U="0"
R:R1 _net1 _netP01 R="10k"
C:C1 _netP01 _ref C="1u" V="0V"
Vdc:V2 _cnet1 _ref U="0"
R:R3 _net2 _netP02 R="10k"
C:C2 _netP02 _ref C="1u" V="-1V"
Vdc:V3DC _net3 _ref U="1V"
Vdc:V3AC _cnet2 _net3 U="0"
R:R4 _net4 _netP03 R="10k"
C:C4 _netP03 _ref C="1u" V="0V"
Vdc:V4DC _net5 _ref U="1V"
Vdc:V4AC _cnet3 _net5 U="0"
R:R5 _net6 _netP04 R="10k"
C:C5 _netP04 _ref C="1u" V="-1V"
Vdc:V5 _cnet4 _ref U="1"
R:R6 _net7 _netP05 R="10k"
```

```
  C:C6 _netP05 _ref C="1u" V="−1V"
  .Def:End
  Sub:X1 _ref _netP01 _netP02 _netP03 _netP04 _netP05 Type="S2Q_TEST4_B"
.Def:End
```

```
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA" vntol="1_uV"
 saveOPs="no" MaxIter="150"
 saveAll="no" convHelper="none" Solver="CroutLU"
.TR:TR1 Type="lin" Start="0" Stop="400ms" Points="1000"
 IntegrationMethod="Gear" Order="6" InitialStep="0.01_ns"
 MinStep="1e−16" MaxIter="1500" reltol="0.001" abstol="100_pA"
 vntol="100_uV" Temp="26.85" LTEreltol="1e−3" LTEabstol="1e−6"
 LTEfactor="1" Solver="CroutLU" relaxTSR="no" initialDC="yes"
 MaxStep="0"
Sub:X1 vp01 vp02 vp03 vp04 vp05 gnd Type="S2Q_test4_b_cir"
```

C 1: DC simulation.

- Vp01.V: **FAIL**; correct dc output = 0V.

- Vp02.V: **FAIL**; correct dc output = 0V.

- Vp03.V: **FAIL**; correct dc output = 1V.

- Vp04.V: **FAIL**; correct dc output = 1V.

- Vp05.V: **FAIL**; correct dc output = 1V.

NOTE: There appears to be an error in the dc simulation of the converted SPICE test netlist. All dc output values have roughly 0.67 volts added to their correct value. This could possibly be due to the way capacitor voltages are initialised. Qucs appears not to initialise capacitor voltages and they appear to be left floating after the SPICE to Qucs conversion process if they are not set by an IC=INCD statement? Although SPICE allows capacitor voltages to be initialised via the optional parameter IC =INCOND this is only used at the start of transient analysis. The question is does it also get used in an .op dc analysis? Here there is a problem for Qucs and possibly the best solution would be for Qucs to use the IC=INCOD value for the initial capacitor voltage, if given, otherwise set capacitor dc voltages to zero at the start of dc and transient analysis. Moreover, they must be initialised and NOT left floating at an indeterminate value.

C 2: Transient simulation:
Simulation waveforms appear to be correct, see Fig. 11.5. NOTE: The transient analysis is set to use capacitor initilalisation voltages.
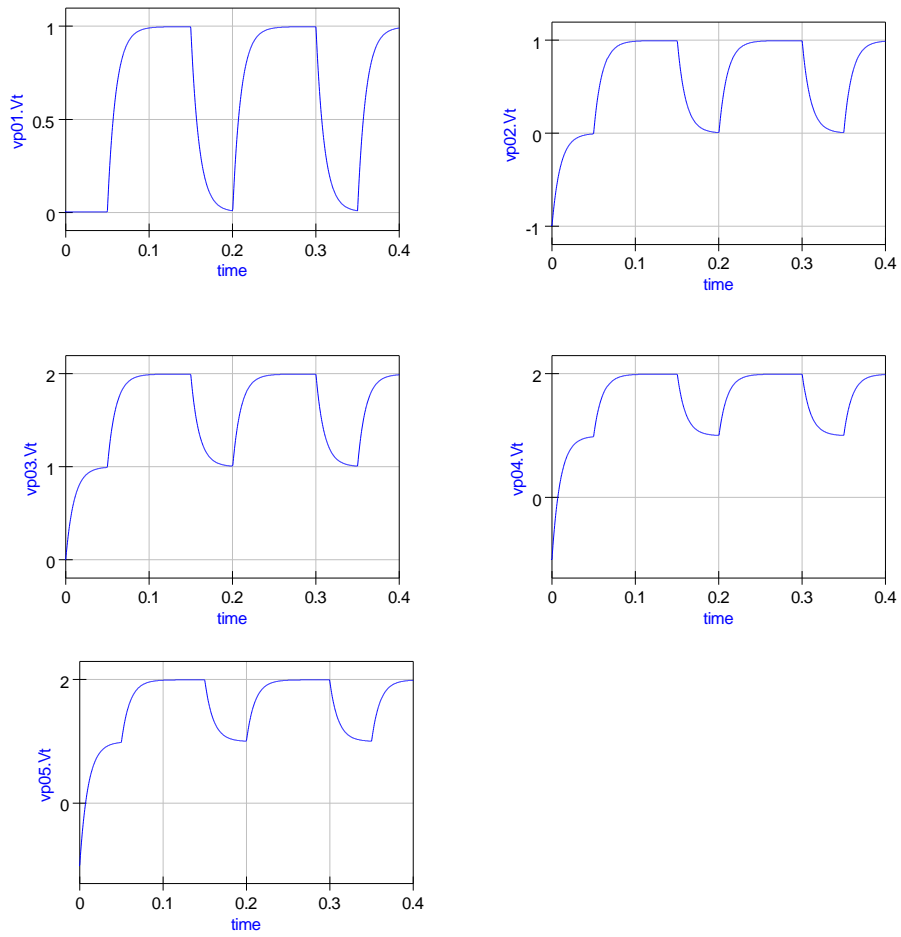
Figure 11.5: March 22: Pulse test waveforms for SPICE 2g6 and 3f5 linear capacitors

D: Combined DC, AC and transient source test.
SPICE code: File `S2Q_test4_c.cir`

```
* SPICE to Qucs syntax test file
* SPICE 2g6 and 3f5 capacitors.
* Pulse   test with dc and ac source
* in series with pulse voltage source.
*
.subckt S2Q_test4_c p06
*
v6 8 0 dc 1v ac 1v pulse(0 1 50ms 1us 1us 100ms 200ms)
r7 8 p06 10k
c7 p06 0 1u ic = −1v
.ends
.end
```

- Vp06.V: **FAIL**; correct dc output = 1V.

- Vp06.v: **PASS**; correct ac waveform.

- Vp06.Vt: **FAIL**; Transient simulation will not run.

NOTE: Vp06.Vt: If **ac 1v** is removed from voltage source statement, v6, the transient analysis gives correct result. Cause unknown.
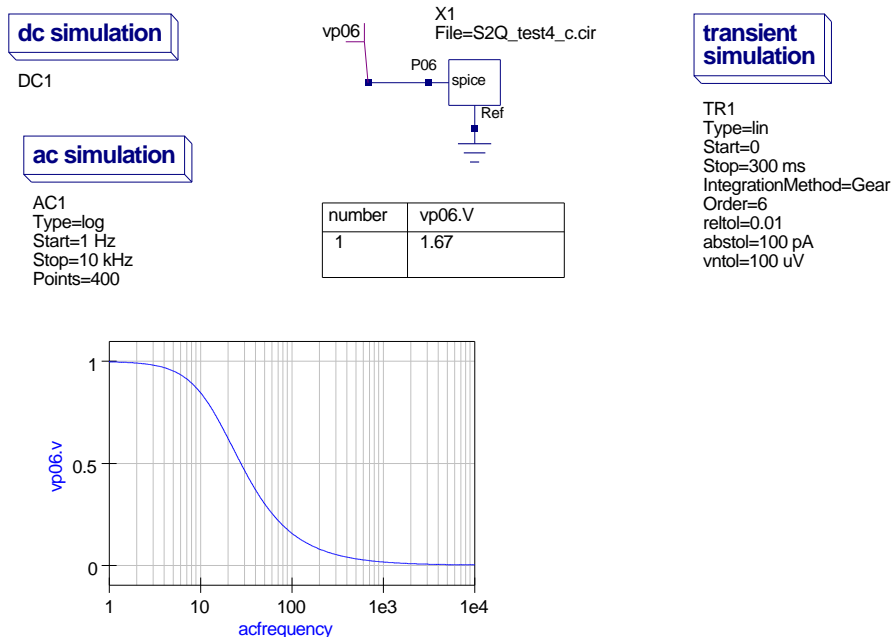
number	vp06.V
1	1.67

Figure 11.6: March 22: Test circuit for combined dc, ac and pulse source test

Qucs netlist:

```
# Qucs 0.0.11  /media/hda2/S2Q_test4_prj/S2Q(test4_c).sch

.Def:S2Q_test4_c_cir _netP06 _ref
  .Def:S2Q_TEST4_C _ref _netP06
  Vrect:V6 _net8 _cnet1 U="1" Td="50ms" Tr="1us" Tf="1us" TH="0.100002"
  TL="0.049998"
  Vac:V6 _cnet1 _cnet0 U="1V"
  Vdc:V6 _cnet0 _ref U="1"
  R:R7 _net8 _netP06 R="10k"
  C:C7 _netP06 _ref C="1u" V="-1V"
  .Def:End
  Sub:X1 _ref _netP06 Type="S2Q_TEST4_C"
.Def:End


Sub:X1 vp06 gnd Type="S2Q_test4_c_cir"
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA" vntol="1_uV" saveOPs="no"
MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
.AC:AC1 Type="log" Start="1_Hz" Stop="10_kHz" Points="400" Noise="no"
.TR:TR1 Type="lin" Start="0" Stop="300_ms" Points="300" IntegrationMethod="Gear"
 Order="6" InitialStep="0.001_ns" MinStep="1e-16" MaxIter="150" reltol="0.01"
abstol="100_pA" vntol="100_uV" Temp="26.85" LTEreltol="1e-3" LTEabstol="1e-6"
LTEfactor="1" Solver="CroutLU" relaxTSR="no" initialDC="yes" MaxStep="0"
```

E: SPICE 2g6 non-linear capacitor tests.

NO results - Qucs 0.0.11 does not allow non-linear capacitors of the SPICE 2g6 form. Nonlinear capacitors will be added to both Qucs and QUCSCONV sometime in the future when the nonlinear independent voltage and current sources are implemented, see the todo list..

F: SPICE 3f5 tests SPICE code: File **S2Q_test4_e.cir**

```
* SPICE to Qucs syntax test file
* SPICE 3f5 semiconductor capacitors.
* DC and AC tests.
*
.subckt S2Q_test4_e p01 p02 p03 p04 p05 p06
v1 1 0 DC 1v AC 1v
r1 1 p01 10k
c1 p01 0 1u
*

v2 2 0 dc 1v ac 1v
r3 2 p02  10k
c2 p02 0 1u ic = 10v
*
v3dc 3 0 dc 1v
v3ac 4 3 ac 1v
r4 4 p03  10k
c3 p03 0 1u ic = -10v
*
v4dc 5 0 dc 1v
v4ac 6 5 ac 1v
r5 6 p04  10k
c4 p04 0  cmod1 L=10u W=1u
.model cmod1 C(CJ=50u CJSW=20p NARROW=0.1u)
*
v3 7 0 dc 1v ac 1v
r6 7 p05  10k
c5 p05 0 cmod1 L=10u W=1u
*
v4 8 0 dc 1v ac 1v
r7 8 p06  10k
c6 p06 0 cmod2 L=10u
.model cmod2 C(DEFW=1u CJ=50u CJSW=20p NARROW=0.1u)
.ends
.end
```

Qucs netlist (DC simulation)

```
# Qucs 0.0.11  /media/hda2/S2Q_test4_prj/S2Q(test4_e).sch
```

236

```
.Def:S2Q_test4_e_cir _netP01 _netP02 _netP03 _netP04 _netP05 _netP06 _ref
  .Def:S2Q_TEST4_E _ref _netP01 _netP02 _netP03 _netP04 _netP05 _netP06
  Vac:V4 _net8 _cnet5 U="1V"
  Vac:V3 _net7 _cnet4 U="1V"
  Vac:V4AC _net6 _cnet3 U="1V"
  Vac:V3AC _net4 _cnet2 U="1V"
  Vac:V2 _net2 _cnet1 U="1V"
  Vac:V1 _net1 _cnet0 U="1V"
  Vdc:V1 _cnet0 _ref U="1V"
  R:R1 _net1 _netP01 R="10k"
  C:C1 _netP01 _ref C="1u"
  Vdc:V2 _cnet1 _ref U="1V"
  R:R3 _net2 _netP02 R="10k"
  C:C2 _netP02 _ref C="1u" V="10V"
  Vdc:V3DC _net3 _ref U="1V"
  Vdc:V3AC _cnet2 _net3 U="1"
  R:R4 _net4 _netP03 R="10k"
  C:C3 _netP03 _ref C="1u" V="−10V"
  Vdc:V4DC _net5 _ref U="1V"
  Vdc:V4AC _cnet3 _net5 U="1"
  R:R5 _net6 _netP04 R="10k"
  C:C4 _netP04 _ref L="10u" W="1u" C="1e−12"
  Vdc:V3 _cnet4 _ref U="1V"
  R:R6 _net7 _netP05 R="10k"
  C:C5 _netP05 _ref L="10u" W="1u" C="1e−12"
  Vdc:V4 _cnet5 _ref U="1V"
  R:R7 _net8 _netP06 R="10k"
  C:C6 _netP06 _ref L="10u" C="1e−12"
  .Def:End
  Sub:X1 _ref _netP01 _netP02 _netP03 _netP04 _netP05 _netP06 Type="S2Q_TEST4_E"
.Def:End

Sub:X1 vp01 vp02 vp03 vp04 vp06 vp07 gnd Type="S2Q_test4_e_cir"
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA" vntol="1_uV"
 saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
```

RESULTS:

```
line 25: checker error, extraneous property 'L' is invalid in 'C:C4'
line 25: checker error, extraneous property 'W' is invalid in 'C:C4'
line 28: checker error, extraneous property 'L' is invalid in 'C:C5'
line 28: checker error, extraneous property 'W' is invalid in 'C:C5'
line 31: checker error, extraneous property 'L' is invalid in 'C:C6'
```

Errors in SPICE semiconductor capacitor to Qucs netlist format conversion.

11.3.2 March 25 2007, Simulation tests by Mike Brinson

Code modifications:

- * `check_spice.cpp`: Handling capacitor models correctly. Also fixed wrong DC value of sources when neither the DC directive nor an immediate value was given. Stefan Jahn.

A: SPICE 2g6 and 3f5 linear capacitor dc tests:

- Vp01.V: **PASS**; dc output = 0V.

- Vp02.V: **PASS**; dc output = 0V.

- Vp03.V: **PASS**; dc output = 1V.

- Vp04.V: **PASS**; dc output = 1V.

- Vp05.V: **PASS**; dc output = 1V.

- Vp06.V: **PASS**; dc output = 1V.

C: SPICE code: File `S2Q_test4_b.cir`
Further notes on test results:

- SPICE independent voltage sources with DC, AC and transient elements appear to be handled differently by SPICE and Qucs. In a SPICE DC operating point simulation, time domain sources have their DC values set to their value at simulation time equal to zero seconds. However, for time domain pulse source Qucs finds the DC level by integrating it's waveform over one signal cycle. This results in a DC value of 0.6667V, yielding the DC offset recorded in the original test. In Qucs a generator is added to the Qucs netlist for each of the elements specified in the original SPICE voltage source entry. Hence, it is possible to have upto three voltage sources in series. Figure 11.7 illustrates that the DC offset occurs from a pulse generator placed in series with a DC source on a schematic.

- The same comment concerning SPICE and Qucs differences introduced above applies to the AC component of a SPICE independent voltage source component. In this case there is not a DC offset but because the frequency of the source is NOT included in the SPICE code, Qucs assumes it's default value to be 1GHz. Hence, in the time domain an AC generator is included in the independent voltage source branch with a frequency of 1GHz. This in turn causes the transient analysis routines to require a very small time step for satisfactory accuracy and hence the simulation takes a long time and appears to hang. SPICE on the other hand considers the AC source to be only applicable to small signal AC analysis and does NOT include it in time domain simulation. Yet another example of the difference between SPICE and Qucs.
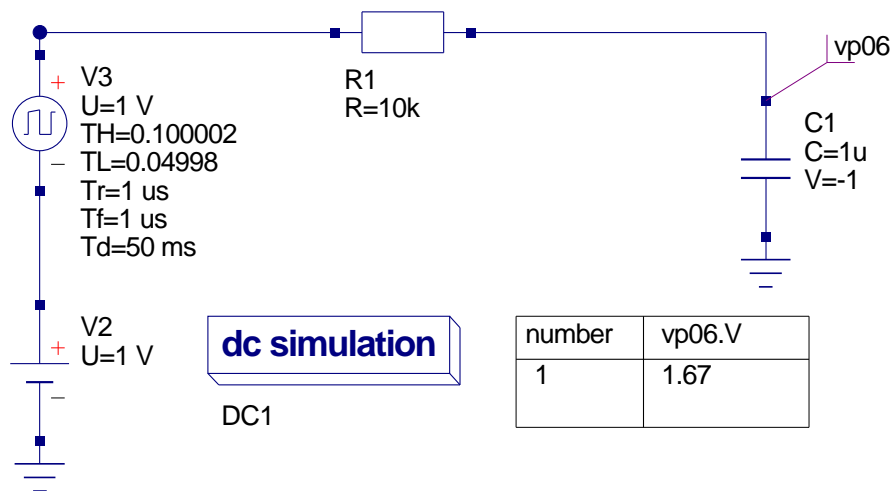
Figure 11.7: March 25: DC simulation showing 0.6667V offset

- RECOMMENDATIONS:

  1. Mix DC, AC and time domain source elements in SPICE independent voltage sources with care when converting code to be simulated with Qucs.

  2. When simulating SPICE netlists in the time domain ensure capacitors have their initial condition voltages set to known values. This ensures that the correct DC conditions form the starting point for a transient circuit simulation.

F: SPICE 3f5 tests SPICE code: File `S2Q_test4_e.cir`
DC simulation test: All outputs **PASS** with a DC value of 1V.

11.4 References

1. A. Vladimirescu, Kaihe Zhang, A.R. Newton, D.O Pederson A. Sangiovanni-Vincentelli, SPICE 2G User's Guide (10 Aug 1981), Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca., 94720.

2. B. Johnson, T. Quarles, A.R. Newton, P.O. Pederson, A.Sangiovanni-Vincentelli, SPICE3 Version 3f User's Manual (October 1972), Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca., 94720.

3. Andrei Vladimirescu, THE SPICE book,1994, John Wiley and Sons. Inc., ISBN 0-471-609-26-9.

12 SPICE to Qucs conversion: Test File 5

12.1 Introduction

12.1.1 Title

SPICE 2g6 and 3f5 inductors.

12.1.2 SPICE specification

Format: SPICE 2g6[1]:

- Linear form: **LX N+ N- value [IC = INCOND]**

- Nonlinear form: **LX N+ N- [POLY] value [L1 [L2 ....]] [IC = INCOND]**

- Coupled (mutual) inductors: **KX L1 L2 Kvalue**

Notes:

1. Characters [and] enclose optional items

2. Inductors begin with letter L.

3. X denotes name of inductor

4. N+ and N- are the positive and negative nodes respectively.

5. Value is the inductor value in Henries.

6. L1 and L2 are the names of the two coupled inductors.

7. Kvalue is the coefficient of coupling; $0 <$ Kvalue and Kvalue < 1.

---

[1]See sections 6.2 and 6.3, SPICE 2g6 user's guide.

8. The "dot" convention is used to denote coupling direction by placing a "dot" on the first node of each inductor.

9. Equations:

   Inductors may be nonlinear functions of current, where

   $$L(I) = value + L1 \cdot I + L2 \cdot I^2 + ...Ln \cdot I^n$$

   and L1, L2, L3 .... are the coefficients of a polynomial describing the element value. Also n $<=$ 20.

Format: SPICE 3f5[2]:

- Linear inductors: **LX N+ N- value [IC = INCOND]**

- Coupled (mutual) inductors: **KX L1 L2 Kvalue**

Notes: 1 to 8 above.

12.2 Test code

SPICE code: File `S2Q_test5-_a.cir`

```
∗ SPICE to Qucs syntax file
∗ SPICE 2g6 and 3f5 inductance
∗ DC and AC tests
.subckt S2Q_test5_a po1, po2, po3
v1 1 0 dc 1v
l1 1 po1 1mH
r1 po1 0 1k
∗
v2 2 0 ac 1v
l2 2 po2 1mH
r2 po2 0 1k
∗
v3 3 0 dc 1v ac 1v
l3 3 po3 1mH
r3 po3 0 1k
∗
.ends
.end
```

---

[2]See sections 3.1.7 and 3.1.8, SPICE 3f6 user's guide.

SPICE code: File `S2Q_test5_b.cir`

```
* SPICE to Qucs syntax file
* SPICE 2g6 and 3f5 inductance
* DC and transient  tests
.subckt S2Q_test5_a po1, po2, po3, po4, po5, po6, po7
v1 1 0 dc 1v
l1 1 po1 1mH ic=0mA
r1 po1 0 1k
*
v2 2 0  dc 0v
l2 2 po2 1mH ic=-10mA
r2 po2 0 1k
*
v3 3 0 dc 1v
l3 3 po3 1mH IC=0mA
r3 po3 0 1k
*
v4 4 0 dc 0v pulse(0 1 0.1u 0.1u 0.1u 10u 20u)
l4 4 po4 1mH ic=0mA
r4 po4 0 1k
*
v5 5 0 dc 1v pulse(0 1 0.1u 0.1u 0.1u 10u 20u)
l5 5 po5 1mH ic=0mA
r5 po5 0 1k
*
v6 6 0 dc 1v  pulse(0 1 0.1u 0.1u 0.1u 10u 20u)
l6 6 po6 1mH ic= -1mA
r6 po6 0 1k
*
v7 7 0 dc 0v pulse(0 1 0.1u 0.1u 0.1u 10u 20u)
l7 7 po7 1mH ic=1mA
r7 po7 0 1k
*
.ends
.end
```

SPICE code: File `S2Q_test5_c.cir`

```
* SPICE to Qucs syntax file
* SPICE 2g6 and 3f5 mutual inductance
* AC  tests
.subckt s2q_test5_c po1 po2
v1 1 0 ac 240
r1 1 2 1Ohm
l1 2 0 100mH
l2 3 0  100mH
```

```
k23  l1  l2  0.999
r2  3  po1  1Ohm
rl  po1  0  150
*
r11  1  21  1Ohm
l11  21  0  100mH
l21   0  31  100mH
k231  l11  l21  0.999
r21  31  po2  1Ohm
rl1  po2  0  150
.ends
.end
```

SPICE code: File `S2Q_test5_d.cir`

```
* SPICE to Qucs syntax file
* SPICE 2g6 and 3f5 mutual inductance
* Transient   test
.subckt s2q_test5_c po1
v1  1  0  sin(0  240  50)
r1  1  2  1Ohm
l1  2  0  100mH
l2  3  0   100mH
k23  l1  l2  0.999
r2  3  po1  1Ohm
rl  po1  0  150
.ends
.end
```

SPICE code: File `S2Q_test5_e.cir`

```
* SPICE to Qucs syntax file
* SPICE 2g6 and 3f5 mutual inductance
* Transient test − two sets of coupling
.subckt s2q_test5_c po1 po2
v1  1  0  sin(0  240  50)
r1  1  2  1Ohm
l1  2  0  100mH
l2  3  0  100mH
l3  0  4  100mH
k12  l1  l2  0.999
k13  l1  l3  0.999
k23  l2  l3  0.999
r2  3  po1  1Ohm
rl1  po1  0  150
r3  4  po2  1Ohm
rl2  po2  0  150
.ends
.end
```

12.3 History of simulation results

12.3.1 April 17 2007, Simulation tests by Mike Brinson

Summary:

- File s2q_5_a: All DC and AC results correct.

- File s2q_5_b: All DC and transient results correct. NOTE: Outputs po4.V, po5.V, po6.V and po7.V have a value of 0.513 added to the DC source values. This corresponds to the DC value of the pulse sources. Changing, for example, time parameter TD changes this value.

- File s2q_5_c: All results appear to be correct.

- File s2Q_5_d: All results appear to be correct.

- File s2q_5_e: All results appear to be correct.

- Inductive coupled networks with more than three coupling coefficients have not been tested.

- SPICE 2g6 non-linear inductors have not been tested. Future implementation of Qucs nonlinear controlled sources will allow simulation of current dependent inductors.
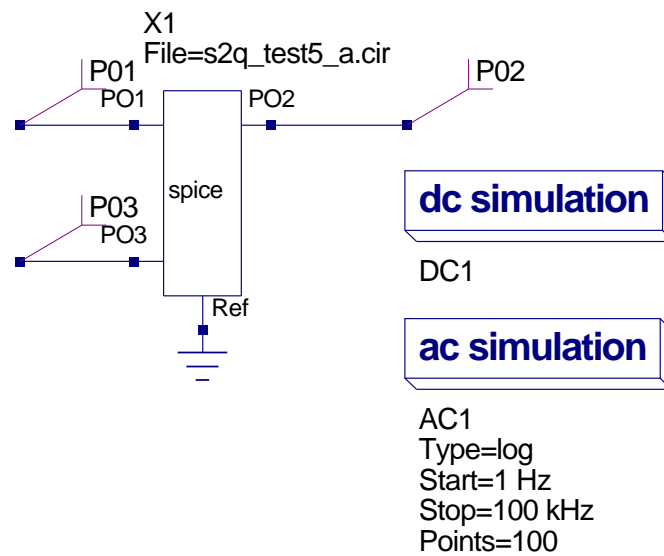
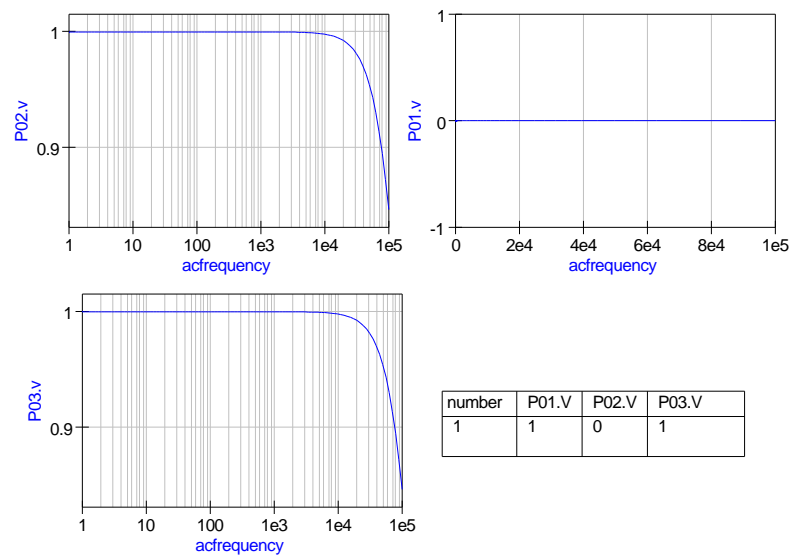Figure 12.1: April 17: Inductor DC and AC test schematic for file s2q_5_a.cir

Figure 12.2: April 17: Inductor DC and AC tests results for file s2q_5_a.cir

```
# Qucs 0.0.12   /media/hda2/S2Q_test5_prj/test_s2q_5_a.sch

.Def:s2q_test5_a_cir _netPO1 _netPO2 _netPO3 _ref
  .Def:S2Q_TEST5_A _ref _netPO1 _netPO2 _netPO3
  Vac:V3 _net3 _cnet1 U="1V"
  Vac:V2 _net2 _cnet0 U="1V"
  Vdc:V1 _net1 _ref U="1V"
  L:L1 _net1 _netPO1 L="1mH"
  R:R1 _netPO1 _ref R="1k"
  Vdc:V2 _cnet0 _ref U="0"
  L:L2 _net2 _netPO2 L="1mH"
  R:R2 _netPO2 _ref R="1k"
  Vdc:V3 _cnet1 _ref U="1V"
  L:L3 _net3 _netPO3 L="1mH"
  R:R3 _netPO3 _ref R="1k"
  .Def:End
  Sub:X1 _ref _netPO1 _netPO2 _netPO3 Type="S2Q_TEST5_A"
.Def:End


Sub:X1 P01 P02 P03 gnd Type="s2q_test5_a_cir"
.AC:AC1 Type="log" Start="1_Hz" Stop="100_kHz" Points="100" Noise="no"
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA" vntol="1_uV"
saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
```

Figure 12.3: April 17: s2q_5_a Qucs netlist
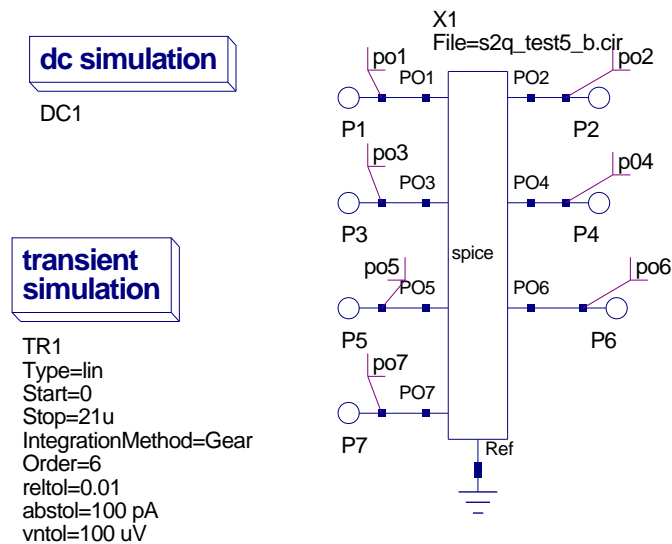

Figure 12.4: April 17: Inductor DC and transient test schematic for file s2q_5_b.cir

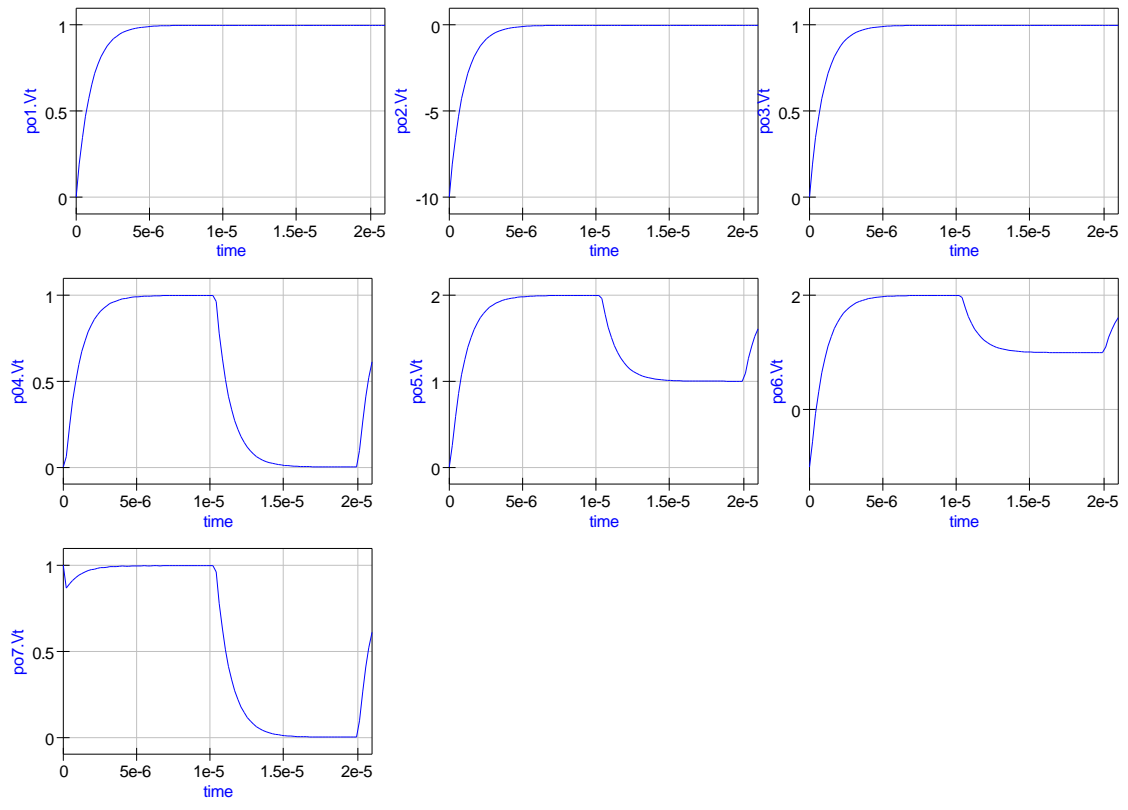| number	po1.V	po2.V	po3.V	p04.V	po5.V	po6.V	po7.V
1	1	0	1	0.513	1.51	1.51	0.513

Figure 12.5: April 17: Inductor DC and transient tests results for file s2q_5_b.cir

Qucs 0.0.12   /media/hda2/S2Q_test5_prj/test_s2q_5_b.sch

```
.Def:s2q_test5_b_cir _netPO1 _netPO2 _netPO3 _netPO4 _netPO5 _netPO6
 _netPO7 _ref
  .Def:S2Q_TEST5_A _ref _netPO1 _netPO2 _netPO3 _netPO4 _netPO5 _netPO6
_netPO7
   Vrect:V7 _net7 _cnet3 U="1" Td="0.1u" Tr="0.1u" Tf="0.1u" TH="1.02e-05"
TL="9.7e-06"
   Vrect:V6 _net6 _cnet2 U="1" Td="0.1u" Tr="0.1u" Tf="0.1u" TH="1.02e-05"
TL="9.7e-06"
   Vrect:V5 _net5 _cnet1 U="1" Td="0.1u" Tr="0.1u" Tf="0.1u" TH="1.02e-05"
TL="9.7e-06"
   Vrect:V4 _net4 _cnet0 U="1" Td="0.1u" Tr="0.1u" Tf="0.1u" TH="1.02e-05"
TL="9.7e-06"
  Vdc:V1 _net1 _ref U="1V"
  L:L1 _net1 _netPO1 L="1mH" I="0mA"
  R:R1 _netPO1 _ref R="1k"
  Vdc:V2 _net2 _ref U="0V"
  L:L2 _net2 _netPO2 L="1mH" I="-10mA"
  R:R2 _netPO2 _ref R="1k"
  Vdc:V3 _net3 _ref U="1V"
  L:L3 _net3 _netPO3 L="1mH" I="0mA"
  R:R3 _netPO3 _ref R="1k"
  Vdc:V4 _cnet0 _ref U="0"
  L:L4 _net4 _netPO4 L="1mH" I="0mA"
  R:R4 _netPO4 _ref R="1k"
  Vdc:V5 _cnet1 _ref U="1"
  L:L5 _net5 _netPO5 L="1mH" I="0mA"
  R:R5 _netPO5 _ref R="1k"
  Vdc:V6 _cnet2 _ref U="1"
  L:L6 _net6 _netPO6 L="1mH" I="-1mA"
  R:R6 _netPO6 _ref R="1k"
  Vdc:V7 _cnet3 _ref U="0"
  L:L7 _net7 _netPO7 L="1mH" I="1mA"
  R:R7 _netPO7 _ref R="1k"
  .Def:End
  Sub:X1 _ref _netPO1 _netPO2 _netPO3 _netPO4 _netPO5 _netPO6 _netPO7
Type="S2Q_TEST5_A"
.Def:End

.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA" vntol="1_uV" saveOPs="no"
 MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
.TR:TR1 Type="lin" Start="0" Stop="21u" Points="100" IntegrationMethod="Gear"
Order="6" InitialStep="1_ns" MinStep="1e-16" MaxIter="150" reltol="0.01"
abstol="100_pA" vntol="100_uV" Temp="26.85" LTEreltol="1e-3"
LTEabstol="1e-6" LTEfactor="1" Solver="CroutLU" relaxTSR="no"
initialDC="yes" MaxStep="0"
Sub:X1 po1 po2 po3 p04 po5 po6 po7 gnd Type="s2q_test5_b_cir"
```

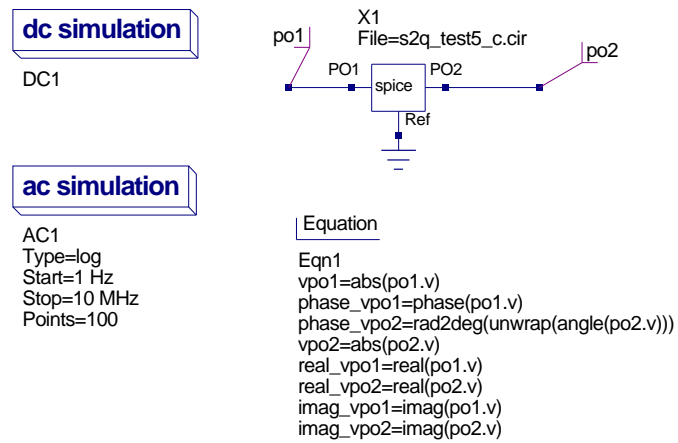Figure 12.6: April 17: s2q_5_b Qucs netlist

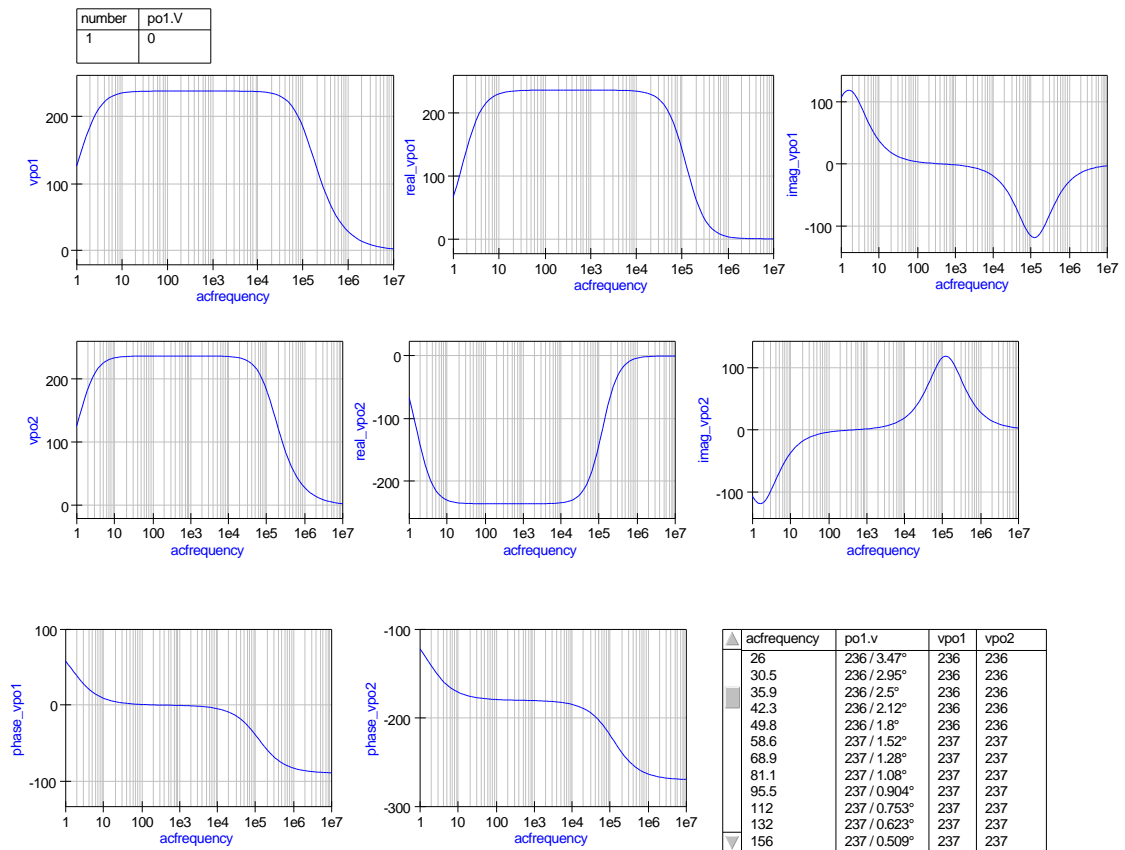Figure 12.7: April 17: Mutual inductance AC test schematic for file s2q_5_c.cir

Figure 12.8: April 17: Mutual inductance tests results for file s2q_5_c.cir

Qucs 0.0.12 /media/hda2/S2Q_test5_prj/test_s2q_5_c.sch

```
.Def:s2q_test5_c_cir _netPO1 _netPO2 _ref
  .Def:S2Q_TEST5_C _ref _netPO1 _netPO2
  L:K231 _cnet3 _ref L="0.0999"
  L:K23 _cnet1 _ref L="0.0999"
  Vac:V1 _net1 _cnet0 U="240"
  Vdc:V1 _cnet0 _ref U="0"
  R:R1 _net1 _net2 R="1Ohm"
  L:L1 _net2 _cnet1 L="0.0001"
  L:L2 _net3 _cnet2 L="0.0001"
  Tr:K23 _cnet1 _cnet2 _ref _ref T="1"
  R:R2 _net3 _netPO1 R="1Ohm"
  R:RL _netPO1 _ref R="150"
  R:R11 _net1 _net21 R="1Ohm"
  L:L11 _net21 _cnet3 L="0.0001"
  L:L21 _ref _cnet4 L="0.0001"
  Tr:K231 _cnet3 _cnet4 _net31 _ref T="1"
  R:R21 _net31 _netPO2 R="1Ohm"
  R:RL1 _netPO2 _ref R="150"
  .Def:End
  Sub:X1 _ref _netPO1 _netPO2 Type="S2Q_TEST5_C"
.Def:End
```

```
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1 pA" vntol="1
uV" saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
.AC:AC1 Type="log" Start="1 Hz" Stop="10 MHz" Points="100" Noise="no"
Sub:X1 po1 po2 gnd Type="s2q_test5_c_cir"
Eqn:Eqn1 vpo1="abs(po1.v)" phase_vpo1="phase(po1.v)"
phase_vpo2="rad2deg(unwrap(angle(po2.v)))" vpo2="abs(po2.v)"
real_vpo1="real(po1.v)" real_vpo2="real(po2.v)" imag_vpo1="imag(po1.v)"
imag_vpo2="imag(po2.v)" Export="yes"
```

Figure 12.9: April 17: s2q_5_c Qucs netlist

dc simulation

DC1

transient simulation

TR1
Type=lin
Start=0
Stop=100 ms
IntegrationMethod=Trapezoidal

X1
File=s2q_test5_d.cir

vpo1
PO1
spice
Ref
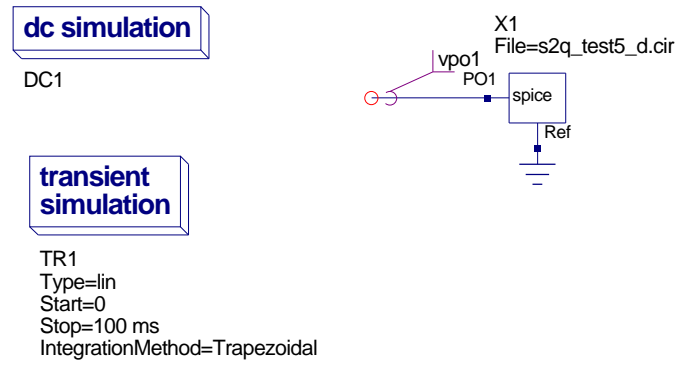
Figure 12.10: April 17: Mutual inductance transient test schematic for file s2q_5_d.cir

number	vpo1.V
1	0

time	vpo1.Vt
0.0341	-235
0.0342	-237
0.0343	-238
0.0344	-239
0.0345	-240
0.0346	-241
0.0347	-241
0.0348	-242
0.0349	-242
0.035	-242
0.0351	-241
0.0352	-240

Figure 12.11: April 17: Mutual inductance transient tests results for file s2q_5_d.cir

Qucs 0.0.12 /media/hda2/S2Q_test5_prj/test_s2q_d.sch

```
.Def:s2q_test5_d_cir _netPO1 _ref
  .Def:S2Q_TEST5_C _ref _netPO1
  L:K23 _cnet1 _ref L="0.0999"
  Vac:V1 _net1 _cnet0 U="240" f="50" Phase="−0" Theta="0"
  Vdc:V1 _cnet0 _ref U="0"
  R:R1 _net1 _net2 R="1Ohm"
  L:L1 _net2 _cnet1 L="0.0001"
  L:L2 _net3 _cnet2 L="0.0001"
  Tr:K23 _cnet1 _cnet2 _ref _ref T="1"
  R:R2 _net3 _netPO1 R="1Ohm"
  R:RL _netPO1 _ref R="150"
  .Def:End
  Sub:X1 _ref _netPO1 Type="S2Q_TEST5_C"
.Def:End


Sub:X1 vpo1 gnd Type="s2q_test5_d_cir"
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA"
vntol="1_uV" saveOPs="no" MaxIter="150" saveAll="no"
convHelper="none" Solver="CroutLU"
.TR:TR1 Type="lin" Start="0" Stop="100_ms" Points="1000"
IntegrationMethod="Trapezoidal" Order="2" InitialStep="1_ns"
MinStep="1e−16" MaxIter="150" reltol="0.001" abstol="1_pA"
vntol="1_uV" Temp="26.85" LTEreltol="1e−3" LTEabstol="1e−6"
LTEfactor="1" Solver="CroutLU" relaxTSR="no" initialDC="yes" MaxStep="0"
```

Figure 12.12: April 17: s2q_5_d Qucs netlist

dc simulation

DC1

X1
File=s2q_test5_e.cir

po1   PO1   PO2   po2
       spice
       Ref

transient
simulation

TR1
Type=lin
Start=0
Stop=100 ms

Figure 12.13: April 17: Mutual inductance transient test schematic for file s2q_5_e.cir

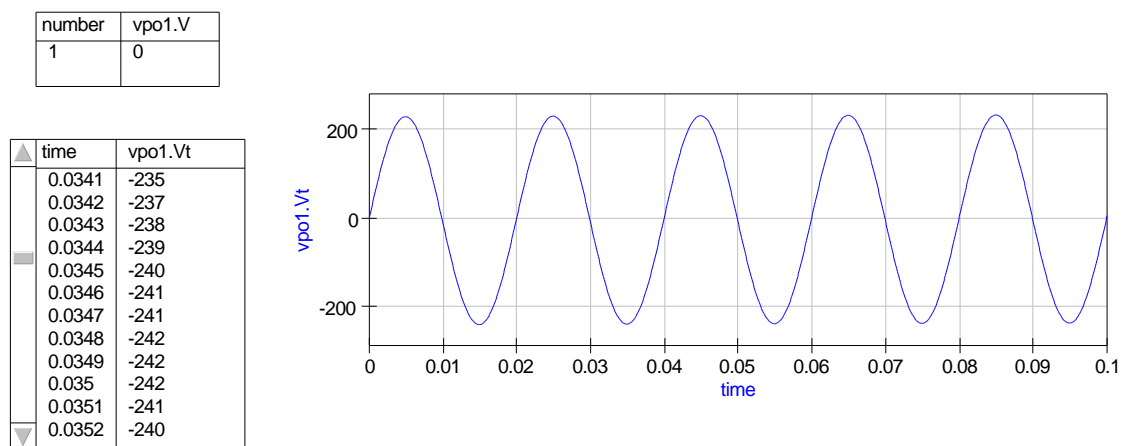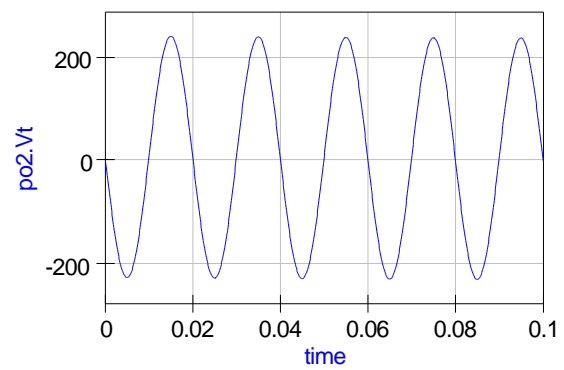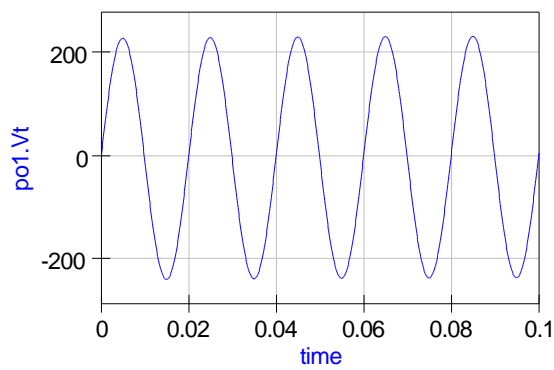| number	po1.V	po2.V
1	0	0

Figure 12.14: April 17: Mutual inductance transient tests results for file s2q_5_e.cir

```
# Qucs 0.0.12   /media/hda2/S2Q_test5_prj/test_s2q_5_e.sch

.Def:s2q_test5_e_cir _netPO1 _netPO2 _ref
  .Def:S2Q_TEST5_C _ref _netPO1 _netPO2
  Vac:V1 _net1 _cnet0 U="240" f="50" Phase="−0" Theta="0"
  Vdc:V1 _cnet0 _ref U="0"
  R:R1 _net1 _net2 R="1Ohm"
  MUT2:K12K13K23 _net2 _ref _net4 _ref _net3 _ref L1="0.1" L2="0.1"
  L3="0.1" k12="0.999" k13="0.999" k23="0.999"
  R:R2 _net3 _netPO1 R="1Ohm"
  R:RL1 _netPO1 _ref R="150"
  R:R3 _net4 _netPO2 R="1Ohm"
  R:RL2 _netPO2 _ref R="150"
  .Def:End
  Sub:X1 _ref _netPO1 _netPO2 Type="S2Q_TEST5_C"
.Def:End


Sub:X1 po1 po2 gnd Type="s2q_test5_e_cir"
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1_pA" vntol="1_uV" saveOPs="no"
MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
.TR:TR1 Type="lin" Start="0" Stop="100_ms" Points="1000"
IntegrationMethod="Trapezoidal" Order="2" InitialStep="1_ns"
MinStep="1e−16" MaxIter="150" reltol="0.001" abstol="1_pA" vntol="1_uV"
Temp="26.85" LTEreltol="1e−3" LTEabstol="1e−6" LTEfactor="1"
Solver="CroutLU" relaxTSR="no" initialDC="yes" MaxStep="0"
```

Figure 12.15: April 17: s2q_5_e Qucs netlist

13 A Curtice level 1 MESFET model

13.1 Introduction

The Metal and Semiconductor FET (MESFET) is a Schottky-barrier gate FET made from gallium arsenide. It is popular for high frequency applications because of it's high electron mobility. The device was developed by Walter R. Curtice[1] in 1980 at the RCA Laboratory in Princeton, New Jersey. USA. The MESFET model presented below is based on a Qucs equation defined device (EDD) which functions as a Curtice level 1 MESFET model with interelectrode capacitances. Basic temperature effects are also included.

13.2 Qucs EDD model for the Curtice MESFET

Parameters

Name	Symbol	Description	Unit	Default
RG	R_G	external gate resistance	Ω	1m
RD	R_D	external drain resistance	Ω	1m
RS	R_S	external source resistance	Ω	1m
VBR	V_{DR}	GS breakdown voltage	V	10^{10}
LG	L_G	external gate lead inductance	H	0
LD	L_D	external drain lead inductance	H	0
LS	L_S	external source lead inductance	H	0
Is	I_S	diode saturation current	A	10f
N	N	diode emission coefficient		1
XTI	X_{TI}	diode saturation current temperature coefficient		0
EG	E_G	diode energy gap	eV	1.11
TAU	τ	internal time delay from drain to source	s	10p
RIN	R_{IN}	series resistance to CGS	Ω	1m
CGS	C_{GS}	interelectrode gate-source bias-independent	F	300f

---

[1]W.R Curtice, 1980, A MESFET model for use in the design of GaAs integrated circuits, IEEE Transactions on Microwave Theory and Techniques, MTT-28, pp. 448-456.

Name	Symbol	Description	Unit	Default
		capacitance		
CGD	C_{GD}	interelectrode gate-drain bias-independent capacitance	F	300f
CDS	C_{DS}	interelectrode drain-source bias-independent capacitance	F	300f
Tnom	T_{NOM}	device parameter measurement temperature	°C	27
Temp	T	device temperature	°C	27
Alpha	α	coefficient of Vds in tanh function for quadratic model	1/V	0.8
Beta	β	transconductance parameter	A/V^2	3m
Lambda	λ	channel length modulation parameter for quadratic model	1/V	40m
VTO	V_{TO}	quadratic model gate threshold voltage	V	−6

Gate1
Num=2

Lg1
L=LG

Equation

Eqn2
Vt=kB/q*TK
GMIN=1e-12
TK=Temp+273.15
TnK=Tnom+273.15

D1
I1=0
Q1=CGD*V1

Rg1
R=RG

Drain1
Num=3

Rd1
R=RD

Equation

Eqn1
TR=TK/TnK
IsT=IS*exp(XTI/N*ln(TR) - EG/N/Vt*(1-TR))

Ld1
L=LD

1

D2
I1=V1<-VBR+50*Vt ? -IsT*(1+exp(-(VBR+V1)/Vt)) + GMIN*V1 : 0
Q1=0
I2=V1>=-VBR+50*Vt && V1<-5*Vt ? -IsT+GMIN*V1 : V1>=-5*Vt ? IsT*(exp(V1/(N*Vt))-1) + GMIN*V1 : 0
Q2=0
I3=V1-VT0>0 ? Beta*(V1-VT0)^2*(1+Lambda*V3)*tanh(Alpha*V3) : 0
Q3=CDS*V3 + TAU*I3
I4=0
Q4=CGS*V4

4   3   2   1

Rin1
R=RIN

Source1
Num=1

Ls1
L=LS

Rs1
R=RS

Curtice1
RG=1m
RD=1m
RS=1m
VBR=10e10
LG=0
LD=0
LS=0
IS=10f
N=1
XTI=0
EG=1.11
Beta=3m
Lambda=40m
VT0=-6
Temp=27
Alpha=0.8
CDS=300f
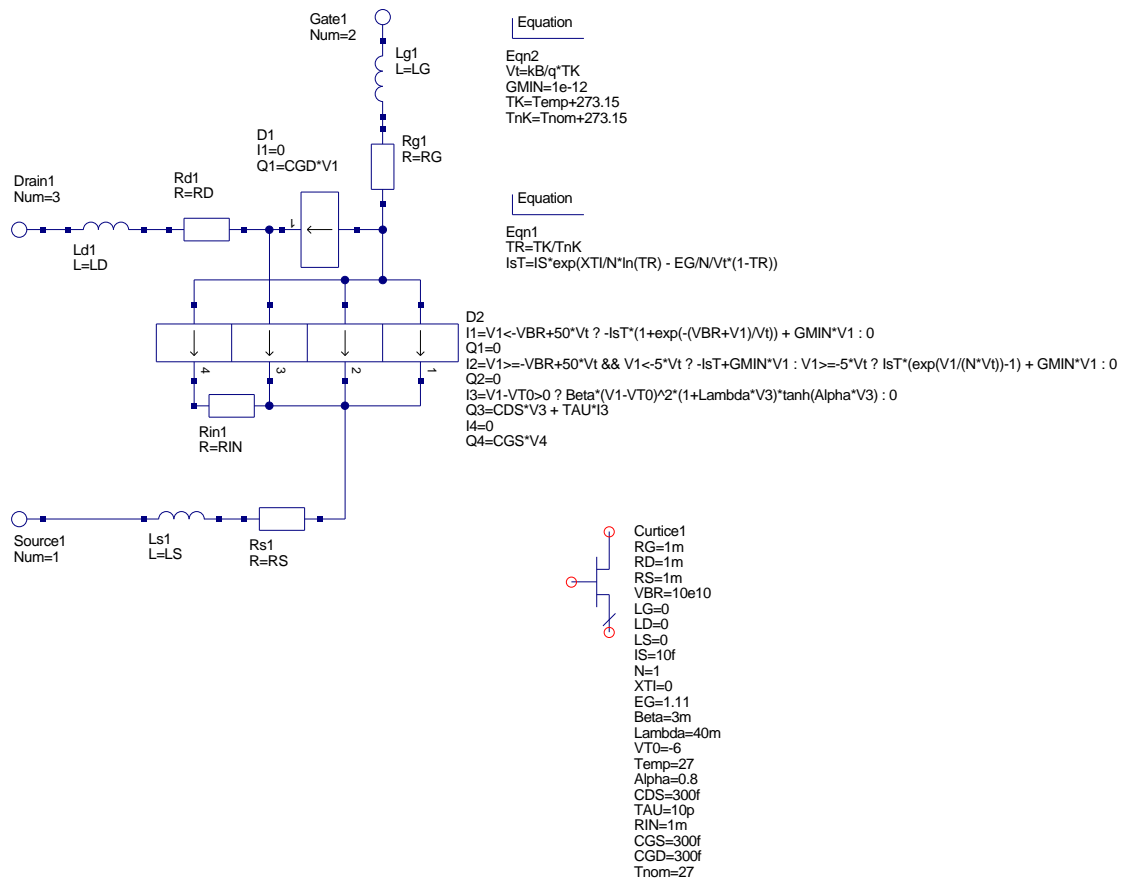TAU=10p
RIN=1m
CGS=300f
CGD=300f
Tnom=27

Figure 13.1: A Qucs EDD model for the Curtice MESFET

13.3 The MESFET equations

- DC characteristics

  1. for $(V_{GS} < -V_{BR} + 50 \cdot V_T)$

  $$I_{GS} = -I_S(T) \cdot \left(1 + \exp\left(-\frac{V_{BR} + V_{GS}}{V_T}\right)\right) + G_{MIN} \cdot V_{GS} \quad (13.1)$$

  2. for $(V_{GS} >= -V_{BR} + 50 \cdot V_T)$ and $(V_{GS} < -5 \cdot V_T)$

  $$I_{GS} = -I_S(T) + G_{MIN} \cdot V_{GS} \quad (13.2)$$

  3. for $(V_{GS} >= -5 \cdot V_T)$

  $$I_{GS} = I_S(T) \cdot \left(\exp\left(\frac{V_{GS}}{N \cdot V_T}\right) - 1\right) + G_{MIN} \cdot V_{GS} \quad (13.3)$$

  4. for $(V_{GS} - V_{TO}) > 0$

  $$I_{DS} = \beta \cdot (V_{GS} - V_{TO})^2 \cdot (1 + \lambda \cdot V_{DS}) \cdot \tanh(\alpha \cdot V_{DS}) \quad (13.4)$$

  Where

  $$I_S(T) = I_S \cdot \exp\left(\frac{X_{TI}}{N} \cdot \ln(TR) - (E_G/N/V_T) \cdot (1 - TR)\right) \quad (13.5)$$

  $$Tr = \frac{TK}{TnK} \quad \text{and} \quad TK = T + 273.15, \quad TnK = T_{NOM} + 273.15 \quad (13.6)$$

- MESFET charge equations

  1.
  $$Q_{GS} = C_{GS} \cdot V_{GS} \quad (13.7)$$

  2.
  $$Q_{GD} = C_{GD} \cdot V_{GD} \quad (13.8)$$

  3.
  $$Q_{DS} = C_{DS} \cdot V_{DS} + \tau \cdot I_{DS} \quad (13.9)$$
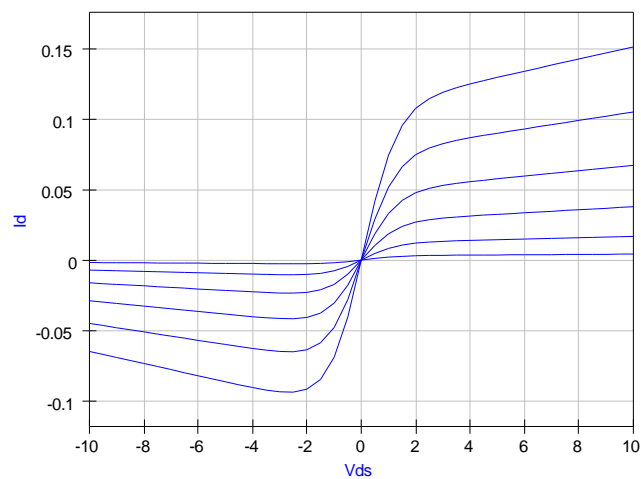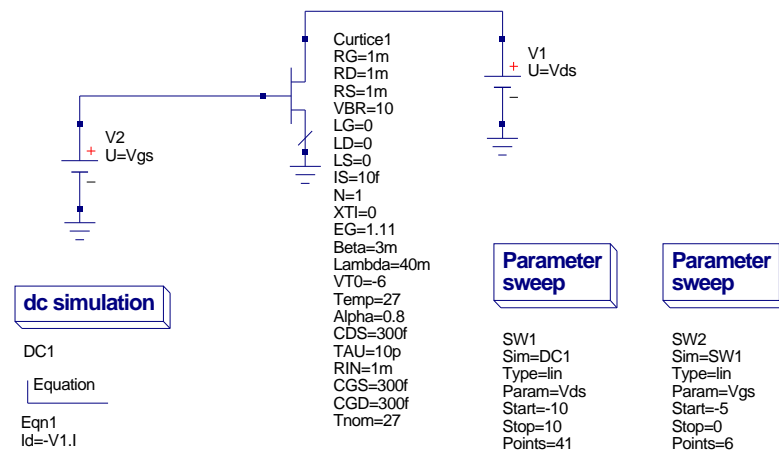
13.4 Test circuits and simulation results

Curtice1
RG=1m
RD=1m
RS=1m
VBR=10
LG=0
LD=0
LS=0
IS=10f
N=1
XTI=0
EG=1.11
Beta=3m
Lambda=40m
VT0=-6
Temp=27
Alpha=0.8
CDS=300f
TAU=10p
RIN=1m
CGS=300f
CGD=300f
Tnom=27

V1
+ U=Vds
–

V2
+ U=Vgs
–

dc simulation

DC1

Equation

Eqn1
Id=-V1.I

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Vds
Start=-10
Stop=10
Points=41

Parameter sweep

SW2
Sim=SW1
Type=lin
Param=Vgs
Start=-5
Stop=0
Points=6

Figure 13.2: DC test circuit and Id-Vds characteristics

V2
U=Vgs

Curtice1
RG=1m
RD=1m
RS=1m
VBR=10
LG=0
LD=0
LS=0
IS=10f
N=1
XTI=0
EG=1.11
Beta=3m
Lambda=40m
VT0=-6
Temp=27
Alpha=0.8
CDS=300f
TAU=10p
RIN=1m
CGS=300f
CGD=300f
Tnom=27

V1
U=Vds

dc simulation

DC1

Equation

Eqn1
Id=-V1.I

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Vgs
Start=-8
Stop=0
Points=21

Parameter sweep

SW2
Sim=SW1
Type=lin
Param=Vds
Start=1
Stop=11
Points=6
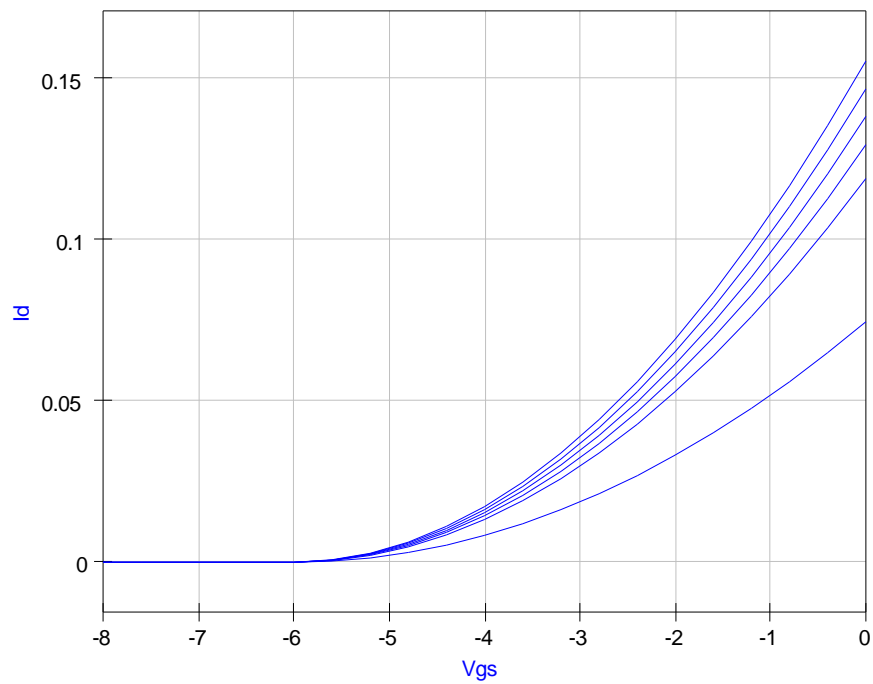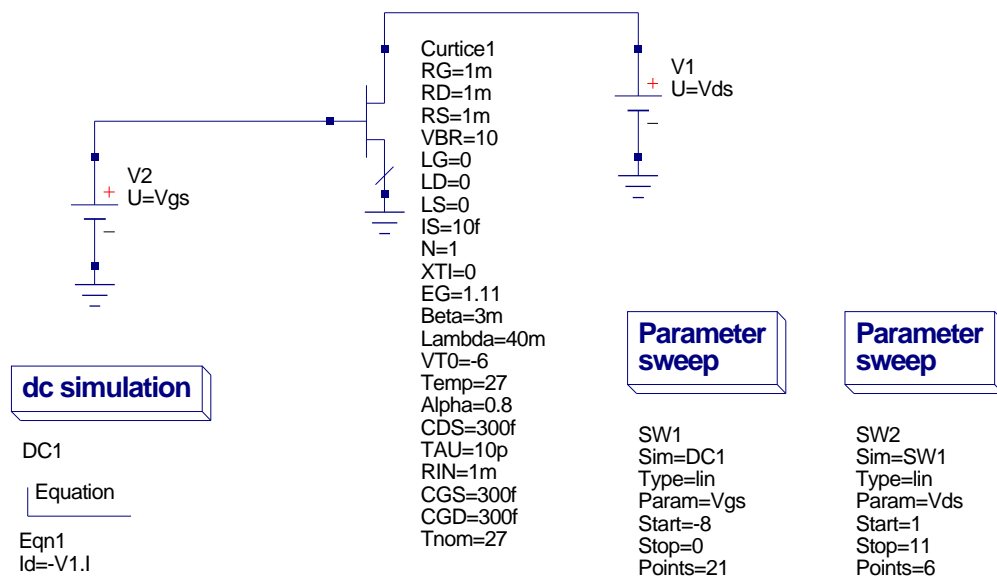
Figure 13.3: DC test circuit and Id-Vgs characteristics

Curtice1
RG=1m
RD=1m
RS=1m
VBR=0
LG=0
LD=0
LS=0
IS=10f
N=1
XTI=3
EG=1.11
Beta=3m
Lambda=40m
VT0=-6
Temp=Temp
Alpha=0.8
CDS=300f
TAU=10p
RIN=1m
CGS=300f
CGD=300f
Tnom=27

V1
U=5

V2
U=Vgs

dc simulation

DC1

Equation

Eqn1
Id=-V1.I
Ig=-V2.I

Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Vgs
Start=0.6
Stop=1.0
Points=21

Parameter sweep

SW2
Sim=SW1
Type=lin
Param=Temp
Start=-100
Stop=+100
Points=6
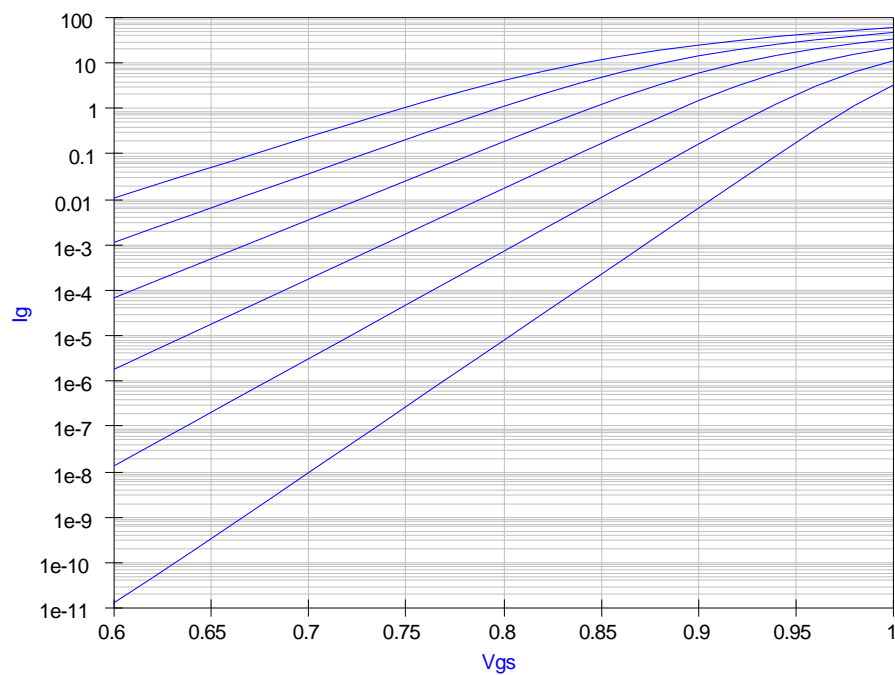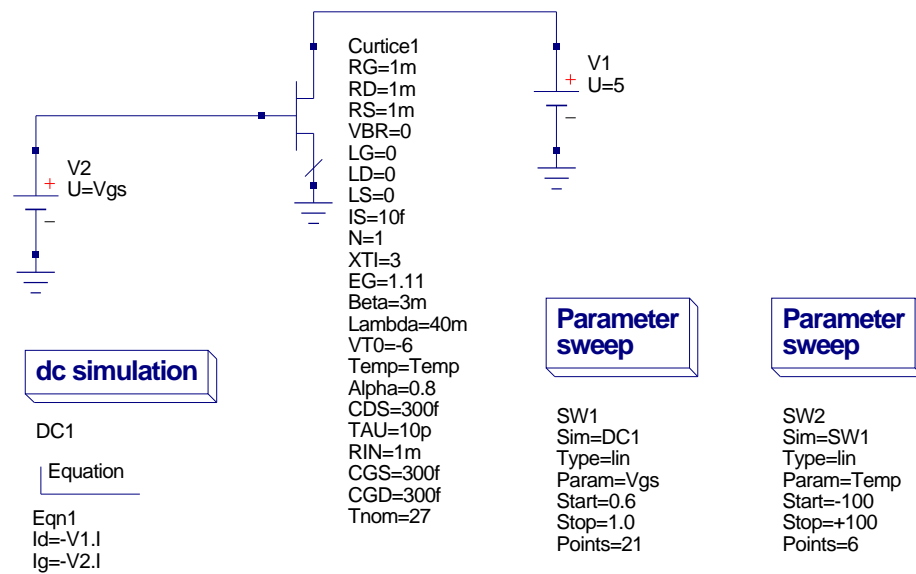
Figure 13.4: DC test circuit and Ig-Vgs characteristics

P1
Num=1
Z=50 Ohm

X1

V2
U=0

Curtice1
RG=1m
RD=1m
RS=1m
VBR=0
LG=1p
LD=5p
LS=5p
IS=10f
N=1
XTI=3
EG=1.11
Beta=3m
Lambda=40m
VT0=-6
Temp=27
Alpha=0.8
CDS=300f
TAU=10p
RIN=1m
CGS=300f
CGD=300f
Tnom=27

X2

V1
U=Vds

P2
Num=2
Z=50 Ohm

dc simulation

DC1

Equation

Eqn1
Id=-V1.I
Ig=-V2.I

**Parameter
sweep**

SW1
Sim=SP1
Type=lin
Param=Vds
Start=5
Stop=20
Points=3

**S parameter
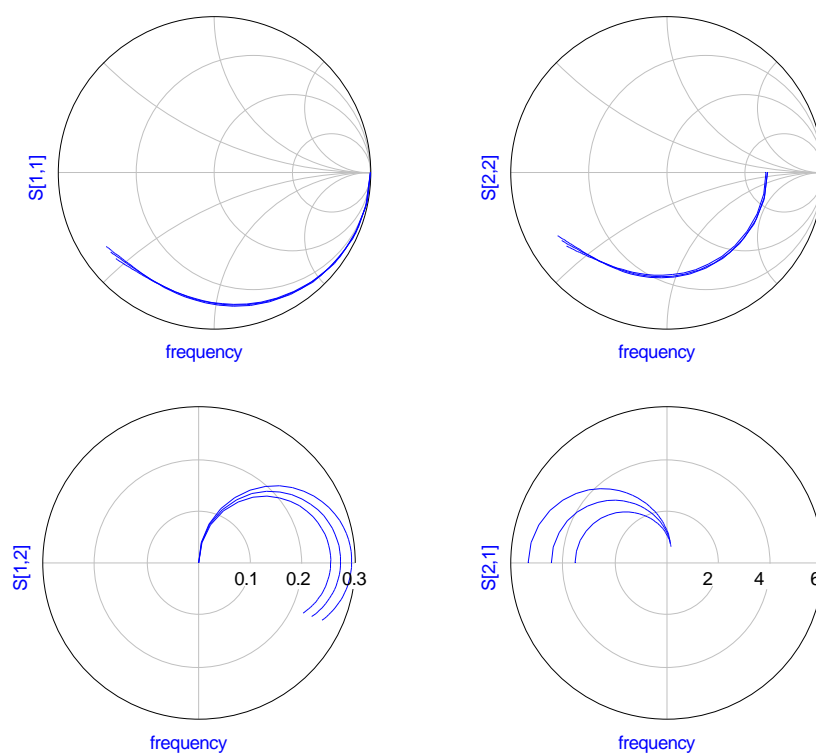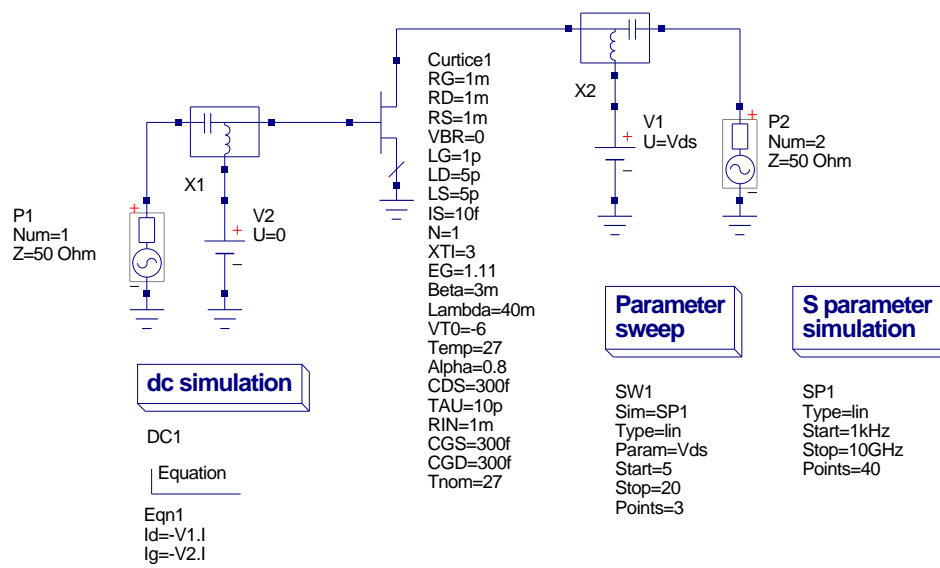simulation**

SP1
Type=lin
Start=1kHz
Stop=10GHz
Points=40

Figure 13.5: S parameter test circuit and characteristics