Qucs Help Documentation

Реліз 0.0.19

Qucs Team

лист. 05, 2017

Зміст

1 Background 3

2 Getting Started with Qucs Analogue Circuit Simulation 5

3 Швидкий старт в оптимізації 11

4 Getting Started with Octave Scripts 23

5 Короткий опис дій 25

6 Фундаментальна обізнаність із підсхемами 29

7 Швидкий старт в цифровому моделюванні 33

8 Короткий опис математичних функцій 37

9 Перелік спеціальних символів 45

10 Узгодження електричних кіл 49

11 Встановлені файли 51

12 Qucs File Formats 53

13 Subcircuit and Verilog-A RF Circuit Models for Axial and Surface Mounted Resistors 61

Contents:

Розділ 1

Background

The 'Quite universal circuit simulator' Qucs (pronounced: kju:ks) is an open source circuit simulator developed by a group of engineers, scientists and mathematicians under the GNU General Public License (GPL). Qucs is the brain-child of German Engineers Michael Margraf and Stefan Jahn. Since its initial public release in 2003 around twenty contributors, from all regions of the world, have invested their expertise and time to support the development of the software. Both binary and source code releases take place at regular intervals. Qucs numbered releases and day-to-day development code snapshots can be downloaded from (http://qucs.sourceforge.net). Versions are available for Linux (Ubuntu and other distributions), Mac OS X © and the Windows © 32 bit operating system.

In the period since Qucs was first released it has evolved into an advanced circuit simulation and device modelling tool with a user friendly "graphical user interface" (GUI) for circuit schematic capture, for investigating circuit and device properties from DC to RF and beyond, and for launching other circuit simulation software, including the FreeHDL (VHDL) and Icarus Verilog digital simulators. Qucs includes built-in code for processing and visualising simulation output data. Qucs also allows users to process post-simulation data with the popular Octave numerical data analysis package. Similarly, circuit performance optimisation is possible using the A SPICE Circuit Optimizer (ASCO) package or Python code linked to Qucs.

Between 2003, and January 2015, the sourceforge Qucs download statistics show that over one million downloads of the software have been recorded. As well as extensive circuit simulation capabilities Qucs supports a full range of device modelling features, including non-linear and RF equation-defined device modelling and the use of the Verilog-A hardware description language (HDL) for compact device modelling and macromodelling. Recent extensions to the software aim to diversify the Qucs modelling facilities by running the Berkeley "Model and Algorithm Prototyping Platform" (MAPP) in parallel with Qucs, using Octave launched from the Qucs GUI. In the future, as the Qucs project evolves, the software will also provide circuit designers with a choice of simulation engine selected from the Qucs built-in code, ngspice and Xyce ©.

Qucs is a large software package which takes time to learn. Incidentally, this statement is also true for other GPL circuit simulators. New users must realise that to get the best from the software some effort is required on their part. In particular, one of the best ways to become familiar with Qucs is to learn a few basic user rules and how to apply them. Once these have been mastered users can move on with confidence to next level of understanding. Eventually, a stage will be reached which allows Qucs to be used productively to model devices and to investigate the performance of circuits. Qucs is equally easy to use by absolute beginners, like school children learning the physics of electrical circuits consisting of a battery and one or more resistors,

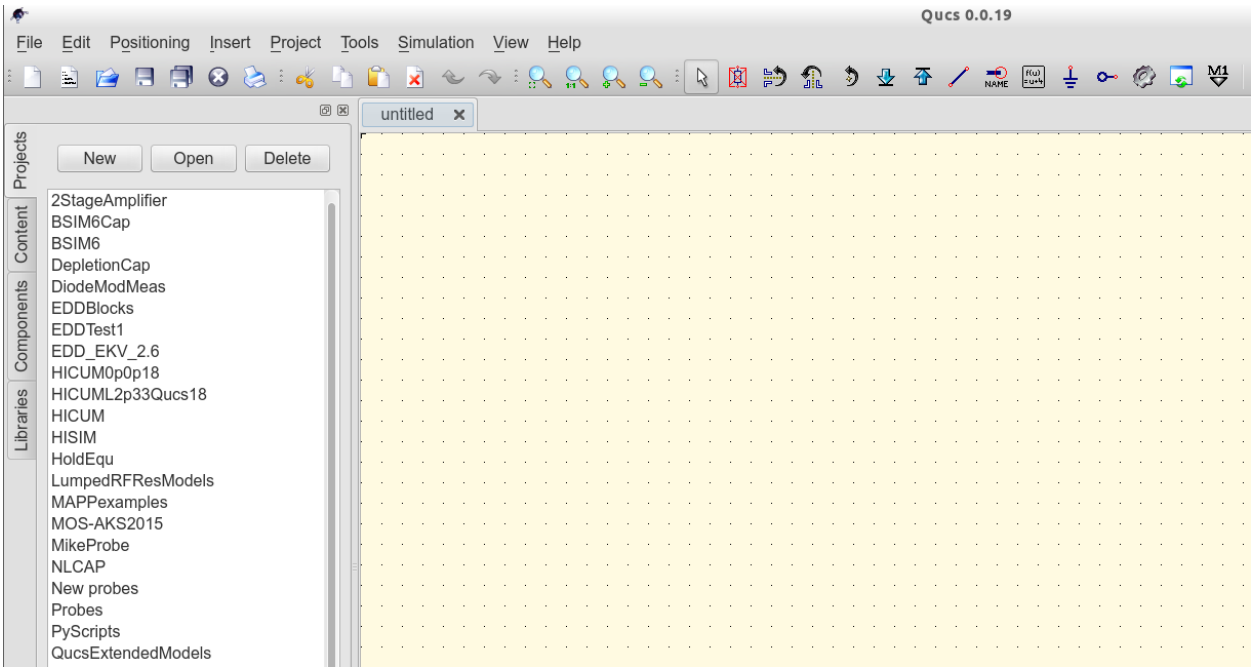
as it is by cutting edge engineers working on the modelling of sub-nano sized RF MOS transistors with hundreds of physical parameters.

The primary purpose of these notes is to provide Qucs users with a source of reference for the operation and capabilities of the software. The information provided also indicates any known limitations and, if available, provides details of any work-arounds. Qucs is a high level scientific/engineering tool who's operation and performance does require users to understand the basic mathematical, scientific and engineering principles underlying the operation of electronic devices and the design and analysis of electronic circuits. Hence, the individual sections of the Qucs-Help document include material of a technical nature mixed in with details of the software operation. Most sections introduce a number of worked design and simulation examples. These have been graded to help readers with different levels of understand get the best from the Qucs circuit simulator. Qucs-Help is a dynamic document which will change with every new release of the Qucs software. At this time, Qucs release 0.0.19, the document is far from complete but given time it will improve.

Розділ 2

Getting Started with Qucs Analogue Circuit Simulation

Qucs is a scientific/engineering software package for analogue and digital circuit simulation, including linear and non-linear DC analysis, small signal S parameter circuit analysis, time domain transient analysis and VHDL/Verilog digital circuit simulation. This section of the Qucs-Help document introduces readers to the basic steps involved in Qucs analogue circuit simulation. When Qucs is launched for the first time, it creates a directory called `.qucs` within the user's home directory. All files involved in Qucs simulations are saved in the `.qucs` directory or in one of it's sub-directories. After Qucs has been launched, the software displays a Graphical User Interface window (GUI) similar, or the same, to the one shown in Figure 1.



Мал. 1 - Головне вікно Qucs

Before using Qucs it is advisable to set the program application settings. This is done from the **File** → **Application Settings** menu. Clicking on **Application Settings** causes the **EditQucsProperties** window

to be displayed, see Figure 2. Complete, with appropriate entries for your Qucs installation, the **Settings**, **Source Code Editor**, **File Types** and **Locations** menus.

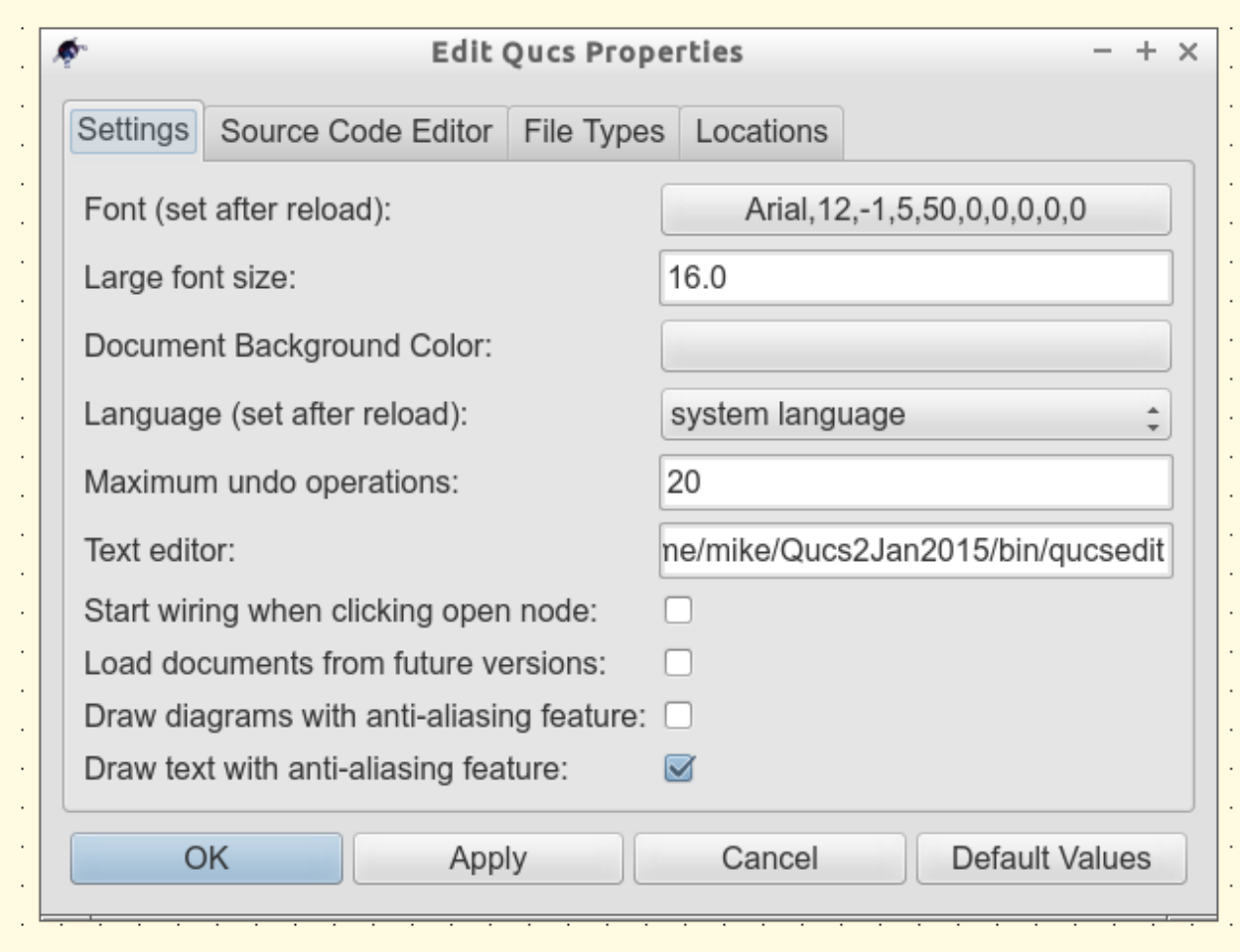


Figure 2 - QucsEditProperties window

On launching Qucs a working area labelled (6) appears at the centre of the GUI. This window is used for displaying schematics, numerical and algebraic model and circuit design data, numerical output data, and signal waveforms and numerial data visualised as graphs, see Figure 3. Clicking, with the left hand mouse button on any of the entries in the tabular bar labelled (5) allows users to quickly switch between the currently open documents. On the left hand side of the Qucs main window is a third area labelled (1) whose content depends on the status of **Projects** (2), **Content** (3), **Components** (4) or **Libraries**. After running Qucs, the **Projects** tab is activated. However note, when Qucs is launched for the first time the **Projects** list is empty.

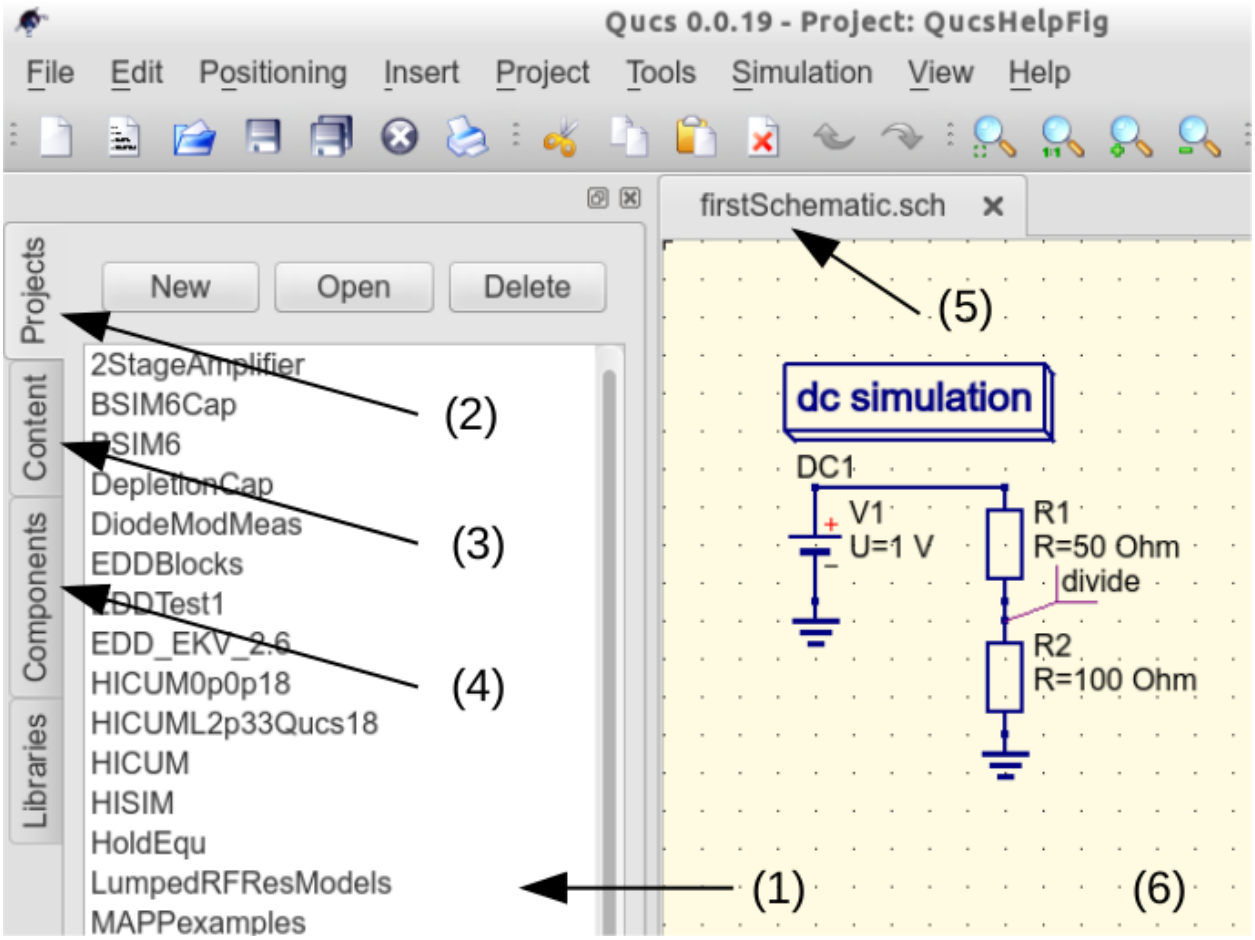


Figure 3 - Qucs main window with working areas labelled

To enter a new project left click on the **New** button located on the right above window (1). This action causes a Qucs GUI dialogue to open. Enter the name of a Qucs project in the box provided, for example enter *QucsHelpFig* and click on the **OK** button. Qucs then creates a project directory in the `~/.qucs` directory. In the example shown in Figure 3 this is called **QucsHelpFig_prj**. Every file belonging to this new project is saved within the **QucsHelpFig_prj** directory. On creation a new project is immediately opened and it's name displayed on the Qucs window title bar. The left tabular bar is then switched to **Content**, and the content of the currently opened project displayed. To save an open document click on the **save** button (or use the main menu: **File** ⟶ **Save**). This step initiates a sequence which saves the document displayed in area (6). To complete the save sequence the program will request the name of your new document. Enter *firstSchematic*, or some other suitable name, and click on the **OK** button to complete the save sequence.

As a first example to help you get started with Qucs enter and run the simple DC circuit shown in Figure 3. The circuit illustrated is a two resistor voltage divider network connected to a fixed value DC voltage source. Start by clicking on the **Components** tab. This action causes a combo box to be displayed from which a component group may be choosen and the required components selected. Choose components group **lumped components** and click on the first symbol: **Resistor**. Next move the mouse cursor into area (6). Pressing the right mouse button rotates the **Resistor** symbol. Similarly, pressing the left mouse button places the component onto the schematic at the place the mouse cursor is pointing at. Repeat this process for all components shown in Figure 3. The independent DC voltage source is located in the **sources** group. The ground symbol can be found in the **lumped components** group or selected from the Qucs toolbar. The icon requesting DC simulation is listed in the **simulations** group. To edit the parameters of the second resistor, double-click on it. A dialogue opens which allows the resistor value to be changed; enter *100 Ohm* in the edit field on the right hand side and click enter.

To connect the circuit components shown in Figure 3, click on the wire toolbar button (or use the main menu: **Insert** ⟶ **Wire**). Move the cursor onto an open component port (indicated by a small red circle at the end of a blue wire). Clicking on it starts the wire drawing sequence. Now move the drawing cursor to the end point of a wire (normally this is a second red circle attached to a placed component) and click again. Two components are now connected. Repeat the drawing sequence as many times as required to wire up the example circuit. If you want to change the corner direction of a wire, click on the right mouse button before moving to an end point. You can also end a wire without clicking on an open port or on a wire; just double-click the left mouse button.

As a final step before DC simulation label the node, or nodes, who's DC voltage is required, for example the wire connecting resistors **R1** and **R2**. Click on the label toolbar button (or use the menu: **Insert** ⟶ **Wire Label**). Now click on the chosen wire. A dialogue opens allowing a node name to be entered. Type *divide* and click the **OK** button. If you have drawn the test schematic correctly the entered schematic should look the same, or be similar to, the one shown in Figure 3.

To start DC simulation click on the **Simulate** toolbar button (or use menu: **Simulation** → **Simulate**). A simulation window opens and a sliding bar reports simulation progress. Normally, all this happens so fast that you only see a short flickering on the PC display (this depends on the speed of your PC). After finishing a simulation successfully Qucs opens a data display window. This replaces the schematic entry window labelled (6) in Figure 3. Next the **Components** ⟶ **diagrams** toolbar is opened. This allows the simulation results to be listed. Click on the **Tabular** item and move it to the display working area, placing it by clicking the left hand mouse button. A dialogue opens allowing selection of the named signals you wish to list, see Figure 4. On the left hand side of the **Tabular** dialogue (called Edit Diagram Properties) is listed the node name: **divide.V**. Double-click on it and it will be transferred to the right hand side of the dialogue. Leave the dialogue by clicking the **OK** button. The DC simulation voltage data for node **divide** should now be listed in a box on the data display window, with a value of 0.666667 volts.

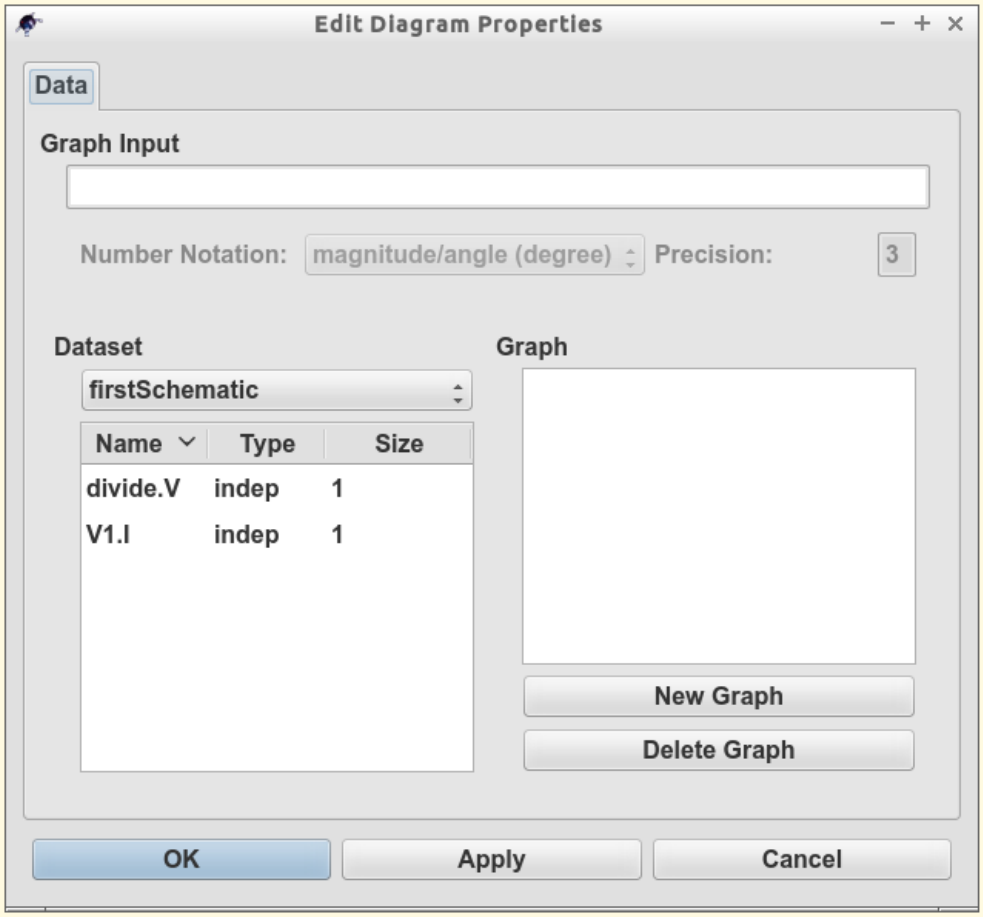


Figure 4 - Qucs data display window showing a **Tabular** dialogue

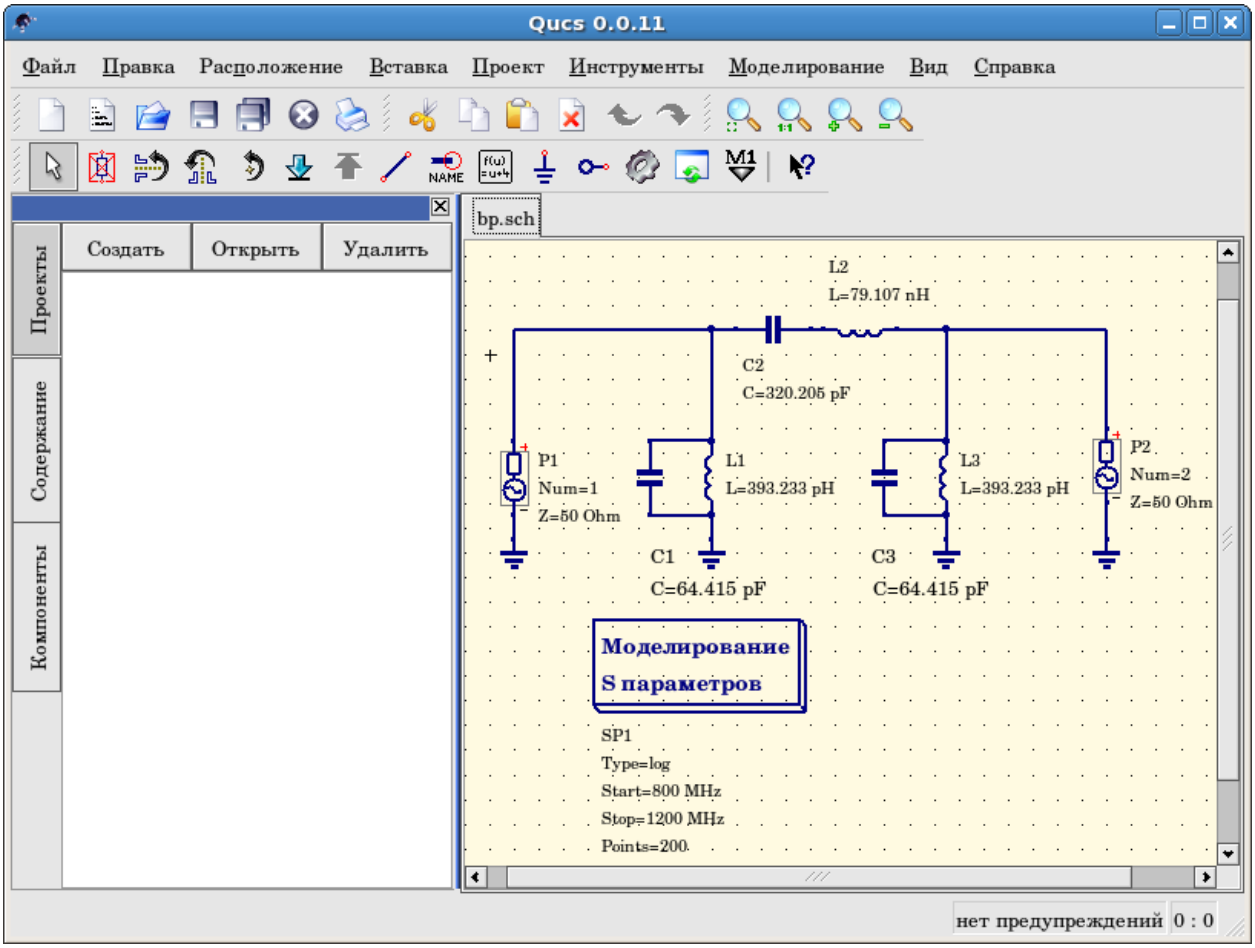
Розділ 3

Швидкий старт в оптимізації

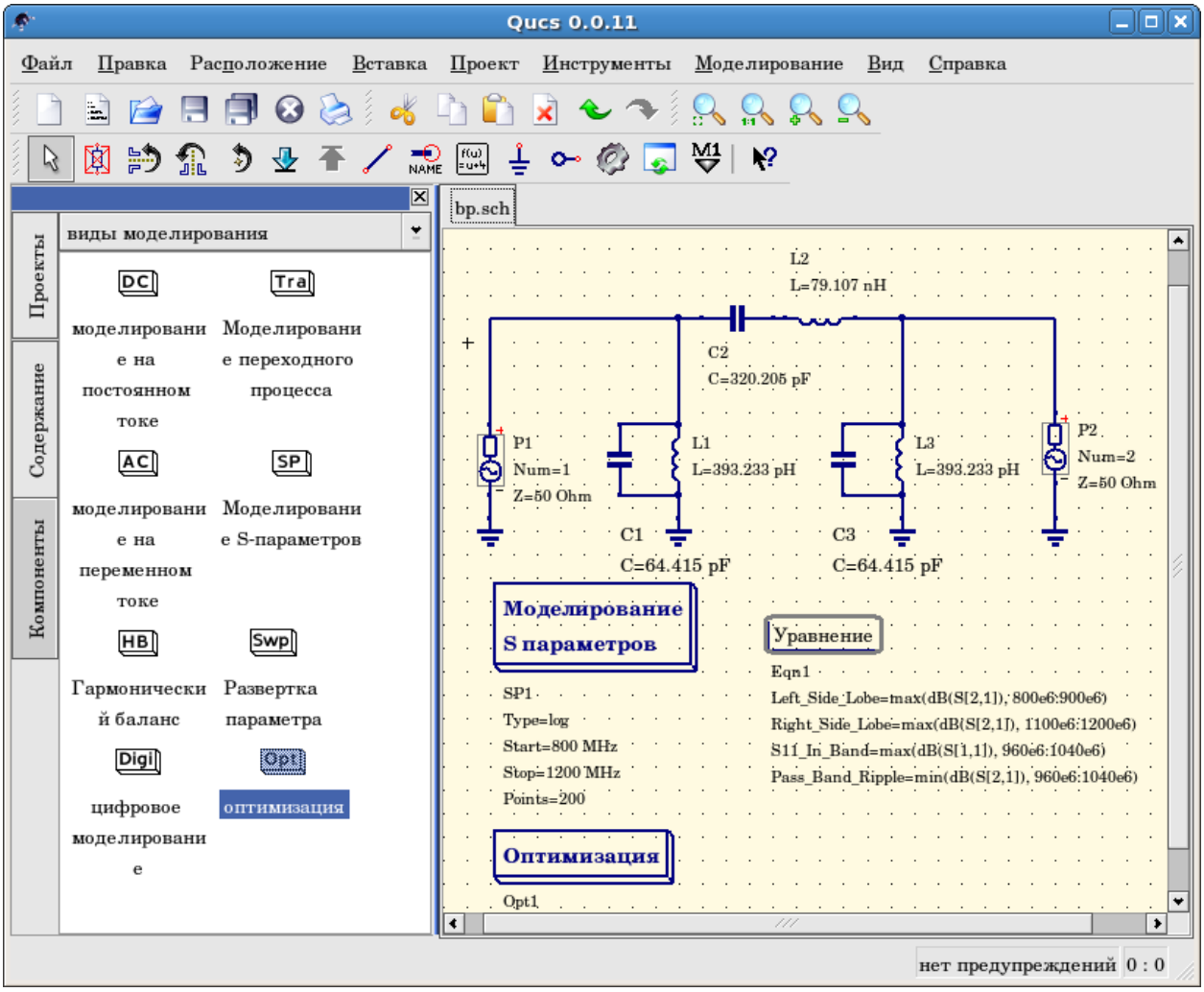
Для оптимізації ланцюгів в Qucs використовується утиліта ASCO (http://asco.sourceforge.net/). Нижче подається стислий опис того, як підготувати схему, виконати утиліту й інтерпретувати результат. Перед цим у системі слід встановити ASCO.

Для підготовки списку зв'язків до оптимізації дві речі потрібно додати до існуючої схеми: потрібно вставити рівняння та блок компонента оптимізації. Візьміть схему з мал. 1 і внесіть у неї зміни такі щоб отримати у результаті схему на мал. 2.

Оптимізація електричних кіл - це ніщо інше, як мінімізація функції вартості. Це може бути час затримки чи наростання цифрового кола, або потужність чи підсилення аналогового електричного кола. Ще один спосіб - визначити завдання оптимізації як поєднання функцій, що в цьому разі веде до визначення показника добротності.

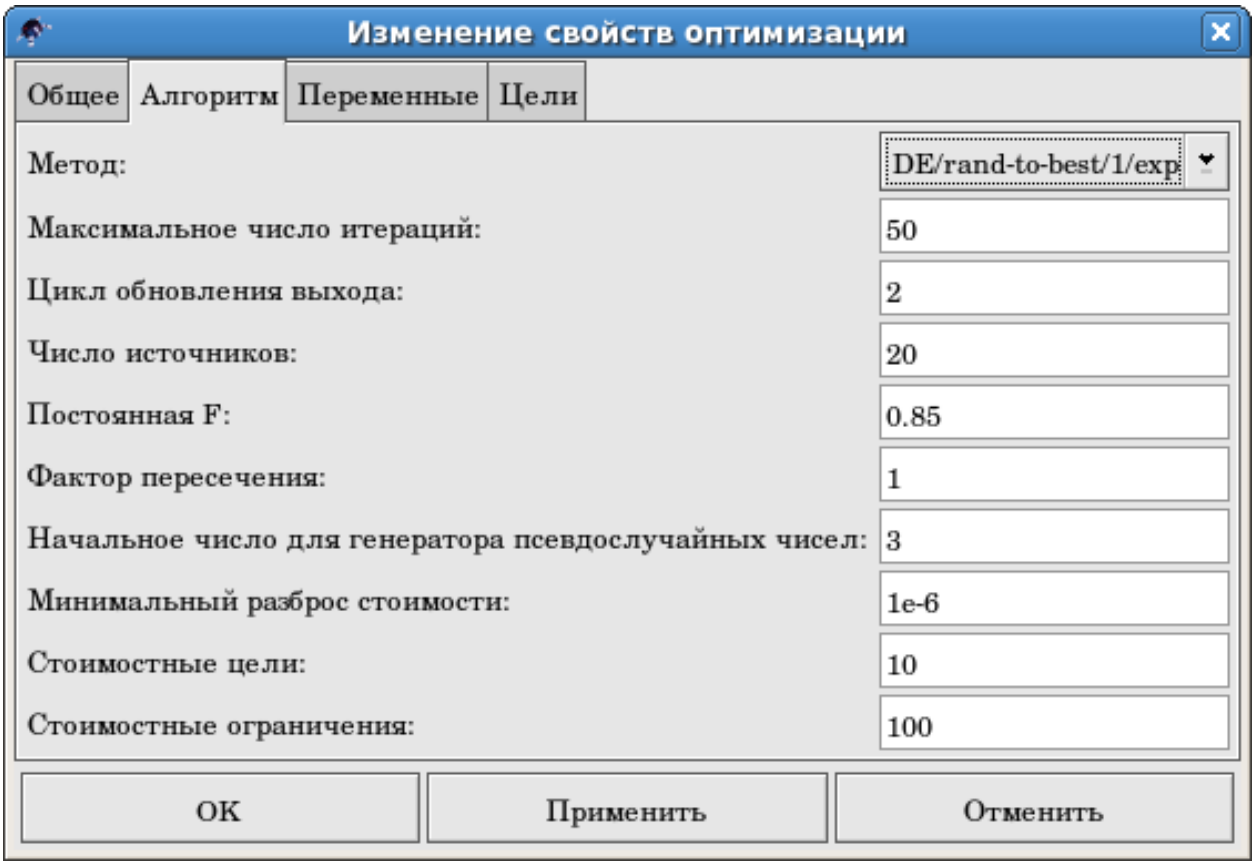


Мал. 1 - Вхідна схема.



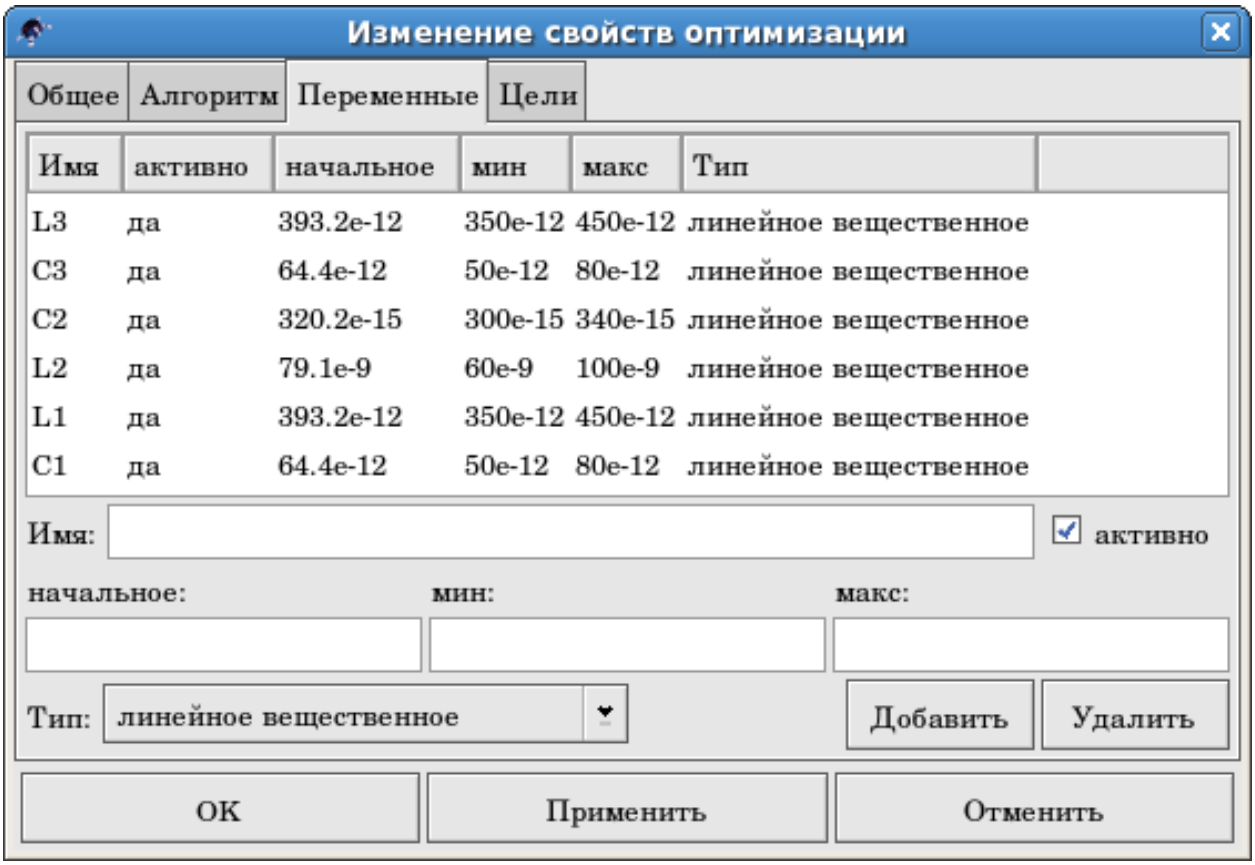
Мал. 2 - Підготовлена схема.

Тепер відкрийте компонент оптимізації і виберіть вкладку оптимізації. З наявних параметрів особливу увагу слід приділити “Максимальному числу ітерацій”, “Константі F” і “Фактору перетину”. Переоцінка чи недооцінка можуть призвести до передчасної збіжності оптимізатора до локального мінімуму або до дуже тривалого часу оптимізації.



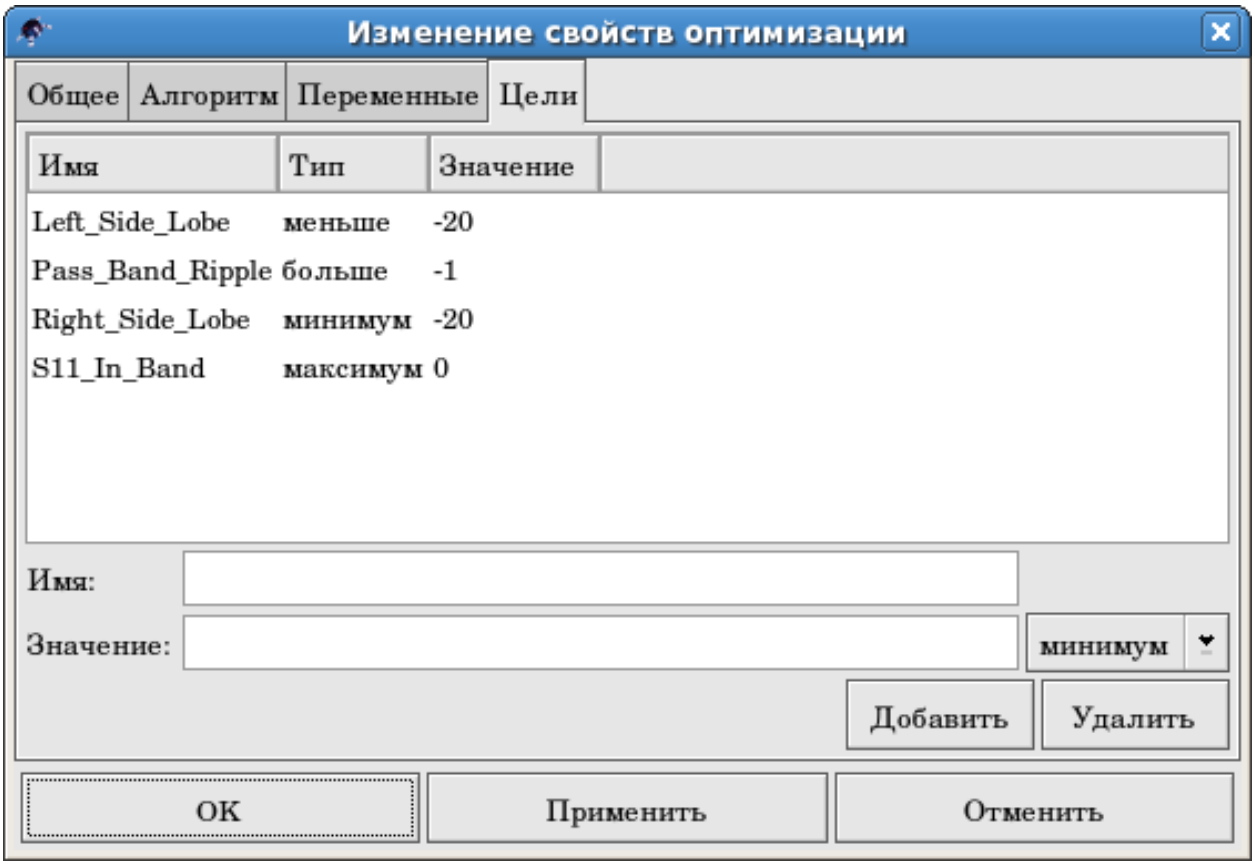
Мал. 3 - Діалог оптимізації, параметри алгоритму.

На вкладці “Змінні” визначається, які елементи кола буде обрано і діапазони їх допустимих значень (мал. 4). Імена змінних відповідають ідентифікаторам, поміщеним у властивості компонентів, а не іменам компонентів.



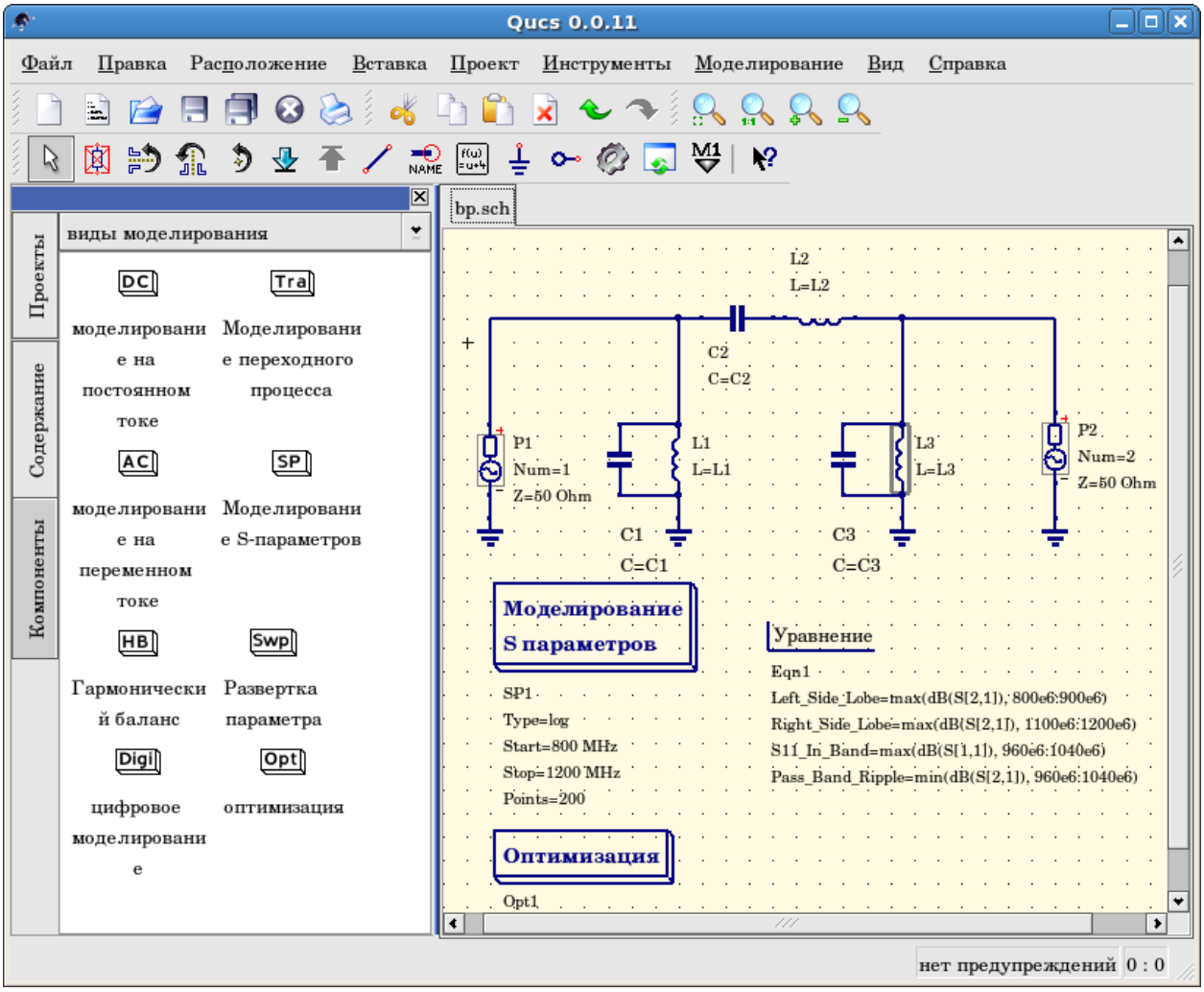
Мал. 4 - Діалог оптимізації, параметри змінних.

Нарешті, переходите до “Цілі”, де задаються цілі оптимізації (зробити максимальним, зробити мінімальним) та обмеження (менше, більше, рівно). Потім ASCO автоматично об’єднує всіх їх в одну функцію вартості, мінімум якої і шукається.



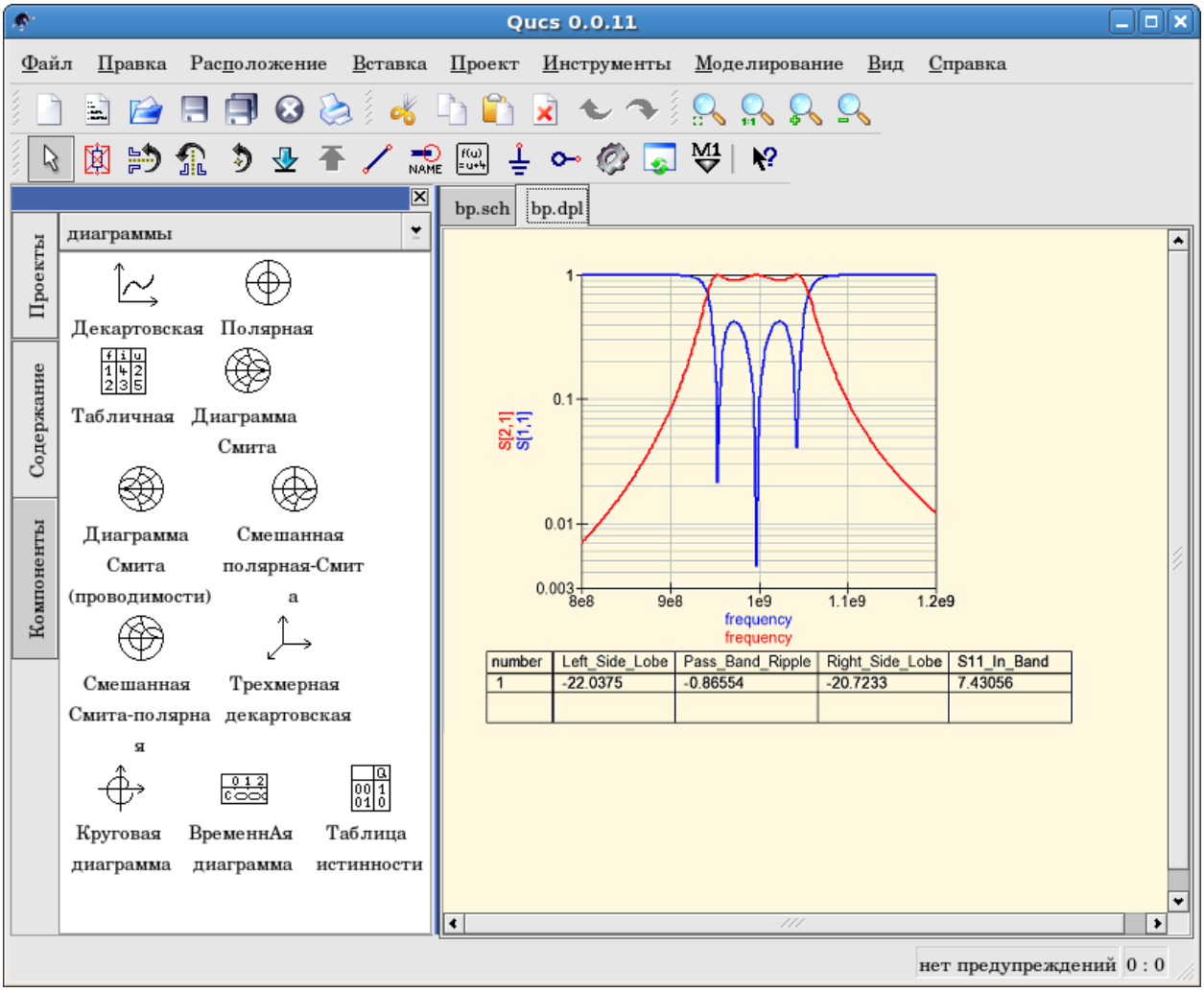
Мал. 5 - Діалог оптимізації, параметри цілей.

Наступний крок полягає у зміні схеми й визначенні, які елементи потрібно оптимізувати. Отримана внаслідок схема зображено на мал. 6.



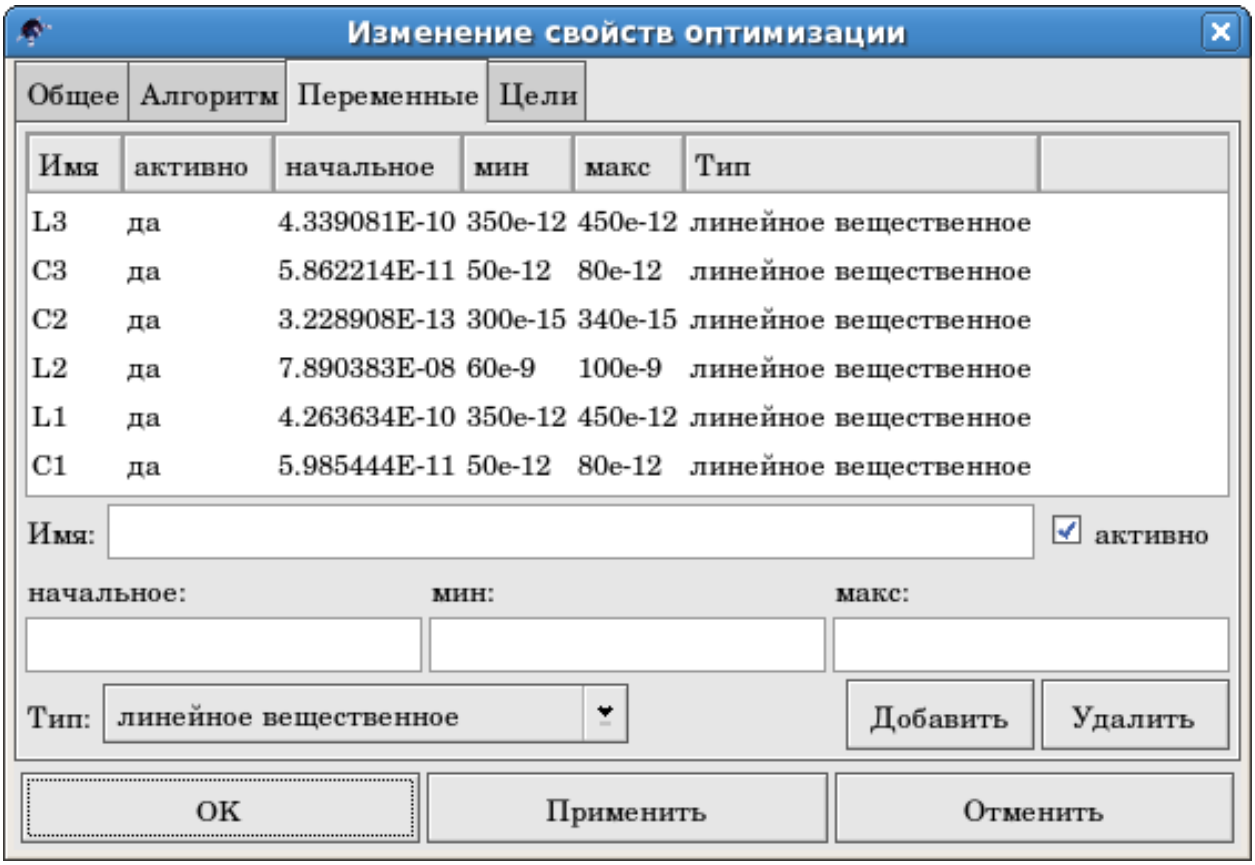
Мал. 6 - Нове головне вікно Qucs.

Останній крок - запуск оптимізації, тобто моделювання, натисканням клавіші F2. Після завершення роботи, на яку на сучасному комп'ютері піде лише кілька секунд, найкращі результати моделювання постануть в графічному вигляді.



Мал. 7 - Вікно Qucs з результатами.

Оптимальні параметри електричного кола можна знайти у діалозі оптимізації, на вкладці “Змінні”. Тепер вони є початковими значеннями кожної з представлених змінних (мал. 8).



Мал. 8 - Найкращі знайдені параметри електричного кола.

By clicking the "Copy current values to equation" button, an equation component defining all the optimization variables with the values of the "initial" column will be copied to the clipboard and can be pasted to the schematic after closing the optimization dialog. The resulting schematic will be as shown in the next figure.

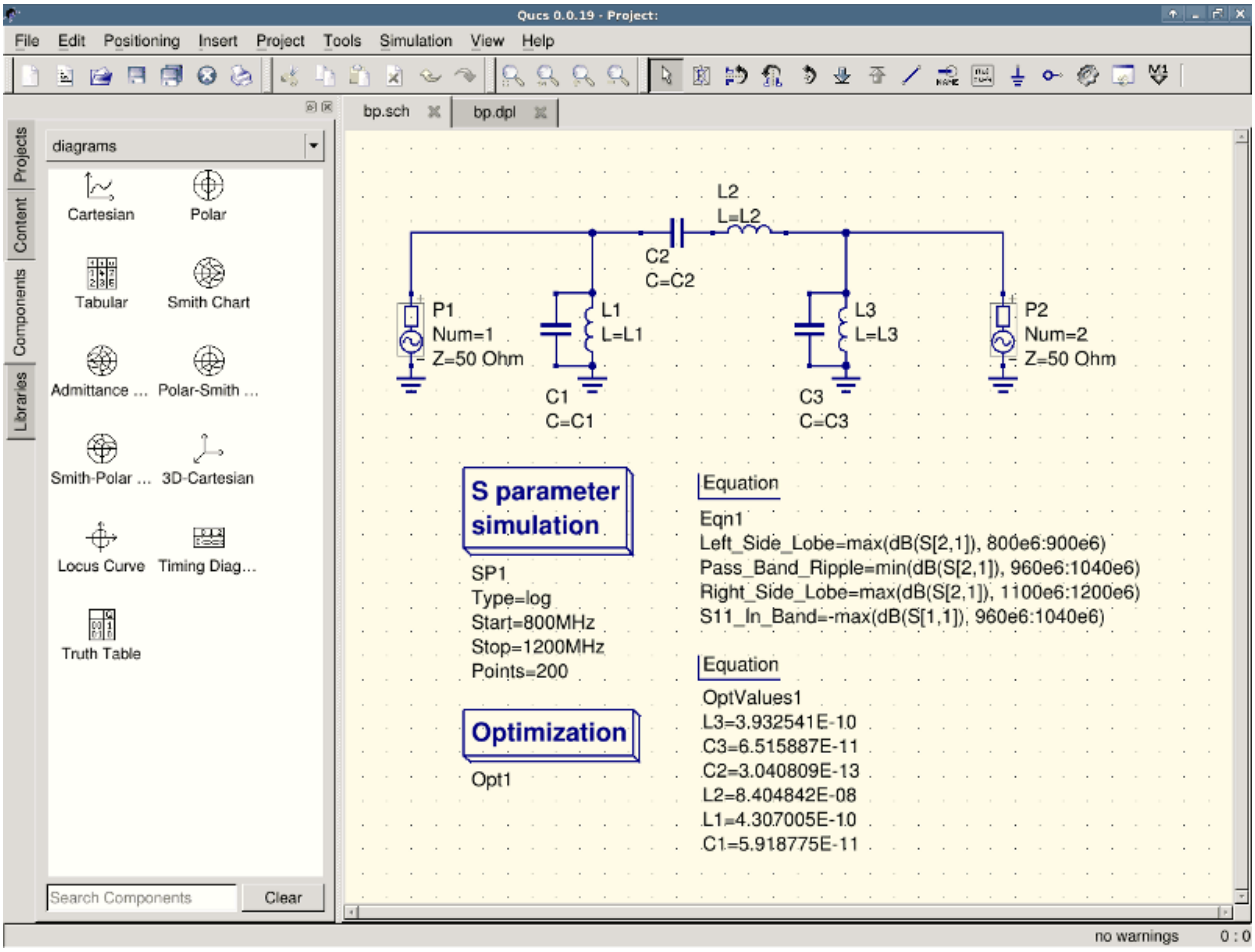


Figure 9 - Schematic with optimized values.

in case you need to do further modifications to the schematic, the optimization component can now be disabled and the optimized values from the pasted equation will be used.

You can change the number of figures shown for the optimized values in the optimization dialog by right-clicking on the “initial” table header and selecting the “Set precision” menu, as shown in the following figure.

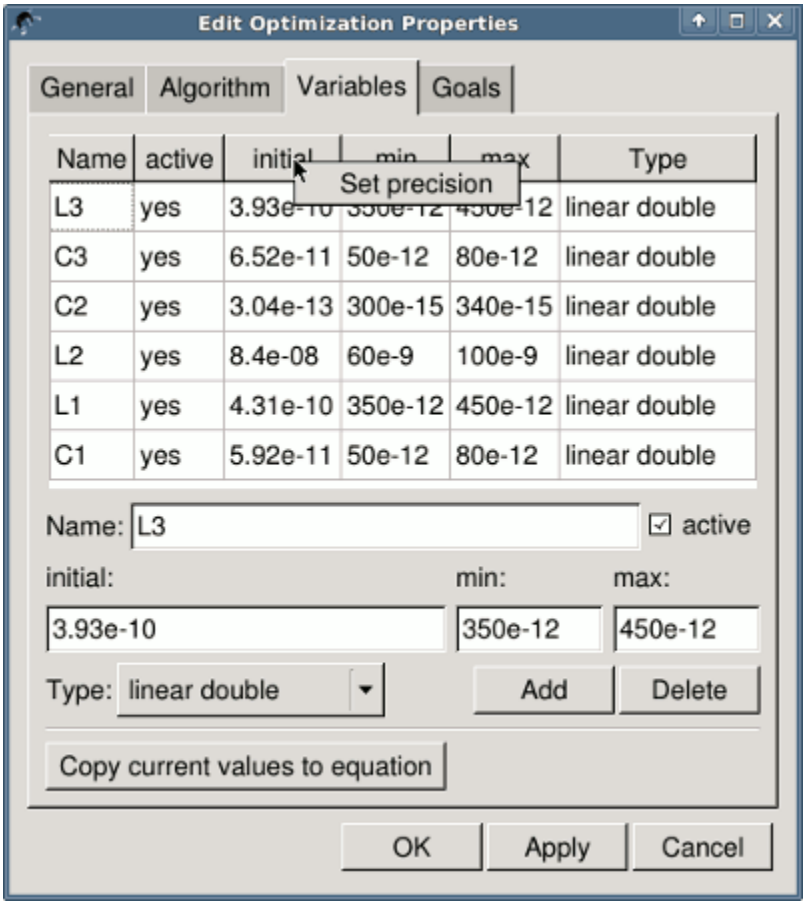


Figure 10 - Changing the displayed variables precision.

Розділ 4

Getting Started with Octave Scripts

Qucs can also be used to develop Octave scripts (see http://www.octave.org). This document should give you a short description on how to do this.

If the user creates a new text document and saves it with the Octave extension, e.g. 'name.m' then the file will be listed at the Octave files of the active project. The script can be executed with F2 key or by pressing the simulate button in the toolbar. The output can bee seen in the Octave window that opens automatically (per default on the right-hand side). At the bottom of the Octave window there is a command line where the user can enter single commands. It has a history function that can be used with the cursor up/down keys.

There are two Octave functions that load Qucs simulation results from a dataset file: loadQucsVariable() and loadQucsDataset(). Please use the help function in the Octave command line to learn more about them (i.e. type help loadQucsVariable and help loadQucsDataset).

4.1 Postprocessing

Octave can also be used for automatic postprocessing of a Qucs simulation result. This is done by editing the data display file of a schematic (Document Settings... in File menu). If the filename of an Octave script (filename extension m) from the same project is entered, this script will be executed after the simulation is finished.

Розділ 5

Короткий опис дій

5.1 Дії загального призначення

колесо миші

колесо миші	Прокручує область малювання по вертикалі. Можна прокручувати за межі поточного розміру.
колесо миші + клавіша Shift	Прокручує область малювання по горизонталі. Можна прокручувати за межі поточного розміру.
колесо миші + клавіша Ctrl	Збільшує чи зменшує масштаб області малювання.
перетягування файлу в документну область	Намагається відкрити файл як схему Qucs чи як документ показу даних.

5.2 Режим “Виділення”

(Меню: Правка->Виділити)

ліва кнопка миші	Виділяє елемент, під курсором миші. Якщо міститься кілька компонентів, можна натискати на кнопку кілька разів, аби вибрати потрібний.Тримаючи цю кнопку миші натиснутою, можна переміщати компонент, під курсором миші, і всі виділені компоненти. Якщо потрібно точно розмістити компоненти, натиснімо під час руху клавішу CTRL, і сітка буде відключена.Якщо тримати кнопку миші натиснутої без будь-яких елементів під курсором, вийде прямокутник. Після відпускання кнопки миші всі елементи, що містяться всередині цього прямокутника, виділяться.Розміри виділеної діаграми чи малюнка можуть бути змінені, якщо натиснути ліву кнопку миші над одним з кутів і рухати курсор, тримаючи кнопку натиснутою.Якщо натиснути кнопку миші на тексті компоненту, його можна безпосередньо редагувати. Натискання клавіші Enter переводить до наступної властивості. Якщо ця властивість є списком вибору, то її можна змінити лише за допомогою клавіші управління курсором (стрілки вверх/вниз).Якщо натиснути кнопку миші на вузлі електричного кола, відбудеться вхід в "режим провідника".
ліва кнопка миші + клавіша Ctrl	Дозволяє виділяти більше елементів, тобто, виділення одного елемента не знімає виділення з інших. Натискання кнопки на виділеному елементі призводить до зняття його виділення. Цей режим також доречний під час виділення з допомогою прямокутника (див. попередній пункт).
права кнопка миші	Натискання кнопки на провіднику виділяє одну пряму лінію, а не весь провідник.
подвійне натискання правої кнопки миші	Відкриває діалог редагування властивостей елемента (мітки провідників, параметри компонентів тощо).

5.3 Режим "Вставка компонента"

(Натиснімо на компонент/діаграму у лівій області)

ліва кнопка миші	Помістити новий примірник компонента на схему.
права кнопка миші	Крутити компонент. (Не діє на діаграмах).

5.4 Режим "Провідник"

(Меню: Вставка->Провідник)

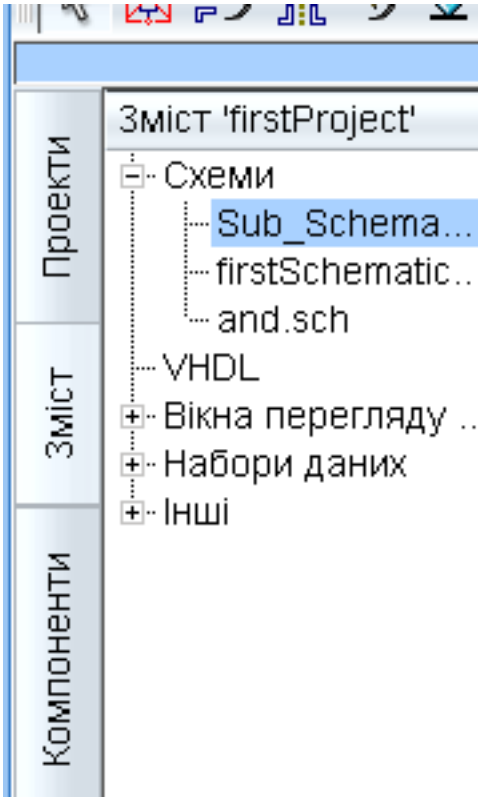
ліва кнопка миші	ліва кнопка миші
права кнопка миші	Змінює напрям вигину провідника (спочатку наліво/направо чи спочатку вверх/вниз).
подвійне натискання правої кнопки миші	Закінчує провідник, не перебуваючи на провіднику чи виводі.

5.5 Режим "Вставка"

(Меню: Правка->Вставити)

ліва кнопка миші	(Меню: Правка->Вставити)
права кнопка миші	Крутити елементи.

5.6 Миша у вкладці “Зміст”



натискання лівої кнопки	Виділяє файл.	
[image]	Відкриває файл.	
натискання правої кнопки	Відображає меню з:	
	“відкрити”	• відкрити виділений файл
	“перейменувати”	• змінити ім’я виділеного файлу
	“видалити”	• видалити виділений файл
	“copy file”	• copy schematic file. Only file operations are performed. Dataset and dispaly settings in schematic properties are kept untouched.

5.7 Клавіатура

Багато дій може бути викликано/зроблено з допомогою клавіш клавіатури. Дії що виконуються пояснюються в рядку статусу при виборі команди із меню. Деякі додаткові команди, що виконуються з допомогою клавіш, наводяться в наступному списку:

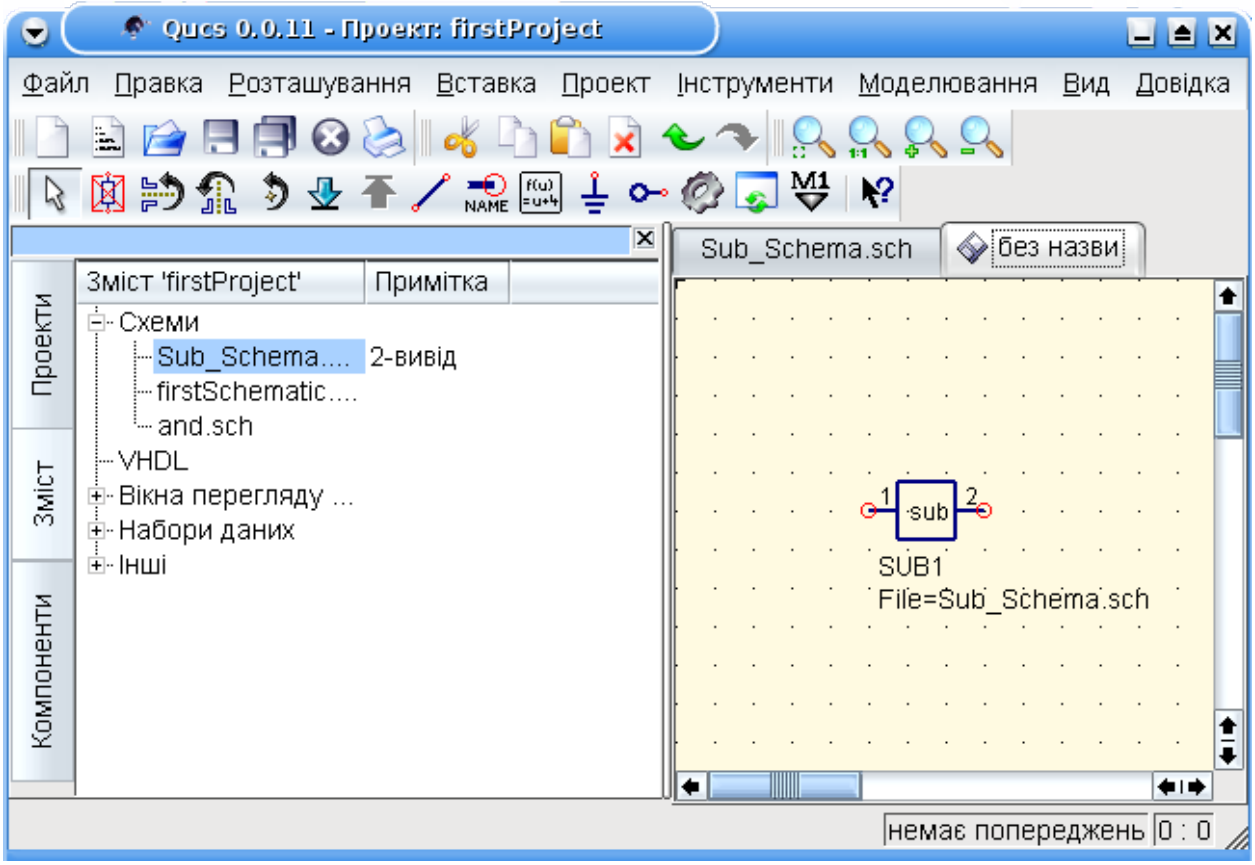
“Delete” чи “Backspace”	Видаляє виділені елементи чи входить у режим видалення, якщо жоден елемент не виділено.
Клавіші зі стрілками вліво/вправо	Змінюють розташування виділених маркерів на графіках.Якщо жоден маркер не виділено, переміщають виділені елементи.Якщо жоден елемент не виділено, прокручують область документа.
Клавіші зі стрілками вверх/вниз	Змінюють розташування виділених маркерів на багатомірних графіках.Якщо жоден маркер не виділено, переміщають виділені елементи.Якщо жоден елемент не виділено, прокручують область документа.
Клавіша Tab	Переходить до наступного відкритого документу (у відповідності до вкладків).

Розділ 6

Фундаментальна обізнаність із підсхемами

Підсхеми використовуються, щоб зробити більш зрозумілою схему. Це дуже корисно у великих схемах чи в схемах де який не будь блок компонентів з'являється по кілька разів.

У Qucs кожна схема, що містить вивід підсхеми, є підсхемою. Вивід підсхеми можна отримати з допомогою панелі інструментів, списку компонентів (в дискретних компонентах) чи меню (Вставка->Вставити вивід). Після того, як вставлені всі виводи підсхеми (наприклад, два), треба зберегти підсхему (наприклад, натиснувши CTRL-S). Якщо глянути в перегляд вмісту проекту (мал. 1), помітно, що тепер праворуч від імені схеми стоїть "2-виводи" (стовпець "Примітка"). Ця позначка є у всіх документів, що є підсхемами. Тепер перейдіть в схему, де Ви хочете використати цю підсхему. Потім натиснімо на ім'я підсхеми (в перегляді вмісту). Знову зайшовши у область документів, Ви бачите, що зараз можна помістити підсхему в головну схему. Зробіть так і закінчіть схему. Тепер можна виконати моделювання. Результат буде таким же, коли б всі компоненти підсхем були розміщені безпосередньо на схемі.



Мал. 1 - Одержання доступу до підсхеми

Якщо вибрати компонент-підсхему (натиснувши на її позначення у схемі), можна ввійти у підсхему, натиснувши CTRL-I (звісно, ця функція доступна через панель інструментів, і через меню). Можна повернутися назад, натиснувши CTRL-H.

Якщо Вам не подобається позначення компонента підсхеми, то можете намалювати свій власний і помістити текст компонента туди, де Вам подобається. Просто зробіть схему підсхеми поточним документом і перейдіть до меню: Файл->Змінити позначення схеми. Якщо ви не намалювали позначення для цієї схеми, то автоматично буде створено просте позначення. Це позначення можна редагувати, малюючи лінії та дуги. Після закінчення, збережіть його. Тепер помістіть його на іншу схему, і вже у Вас є нове позначення.

Як і в усіх інших компонентів, у підсхем можуть бути параметри. Для формування власних параметрів, поверніться в редактор, де ви редагували позначення підсхеми, і двічі натисніть ліву кнопку на тексті параметра підсхеми. З'явиться діалогове вікно, у якому можете заповнити параметри початковими значеннями і описами. Коли Ви це закінчите, закрийте діалогове вікно і збережіть підсхему. Скрізь, де вставляється підсхема, у неї будуть ці нові параметри, і можна редагувати їх, як і у всіх інших компонентів.

6.1 Subcircuits with Parameters

A simple example using subcircuits with parameters and equations is provided here.

Create a subcircuit:

- Create a new project

- New schematic (for subcircuit)
- Add a resistor, inductor, and capacitor, wire them in series, add two ports
- Save the subcircuit as RLC.sch
- Give value of resistor as 'R1'
- Add equation 'ind = L1',
- Give value of inductor as 'ind'
- Give value of capacitor as 'C1'
- Save
- File > Edit Circuit Symbol
- Double click on the 'SUB File=name' tag under the rectangular box
  - Add name = R1, default value = 1
  - Add name = L1, default value = 1
  - Add name = C1, default value = 1
  - Гаразд

Insert subcircuit and define parameters:

- New schematic (for testbench)
- Save Test_RLC.sch
- Project Contents > pick and place the above RLC subcircuit
- Add AC voltage source (V1) and ground
- Add AC simulation, from 140Hz to 180Hz, 201 points
- Set on the subcircuit symbol
  - R1=1
  - L1=100e-3
  - C1=10e-6
- Моделювати
- Add a Cartesian diagram, plot V1.i
- The result should be the resonance of the RLC circuit.
- The parameters of the RLC subcircuit can be changed on the top schematic.

Розділ 7

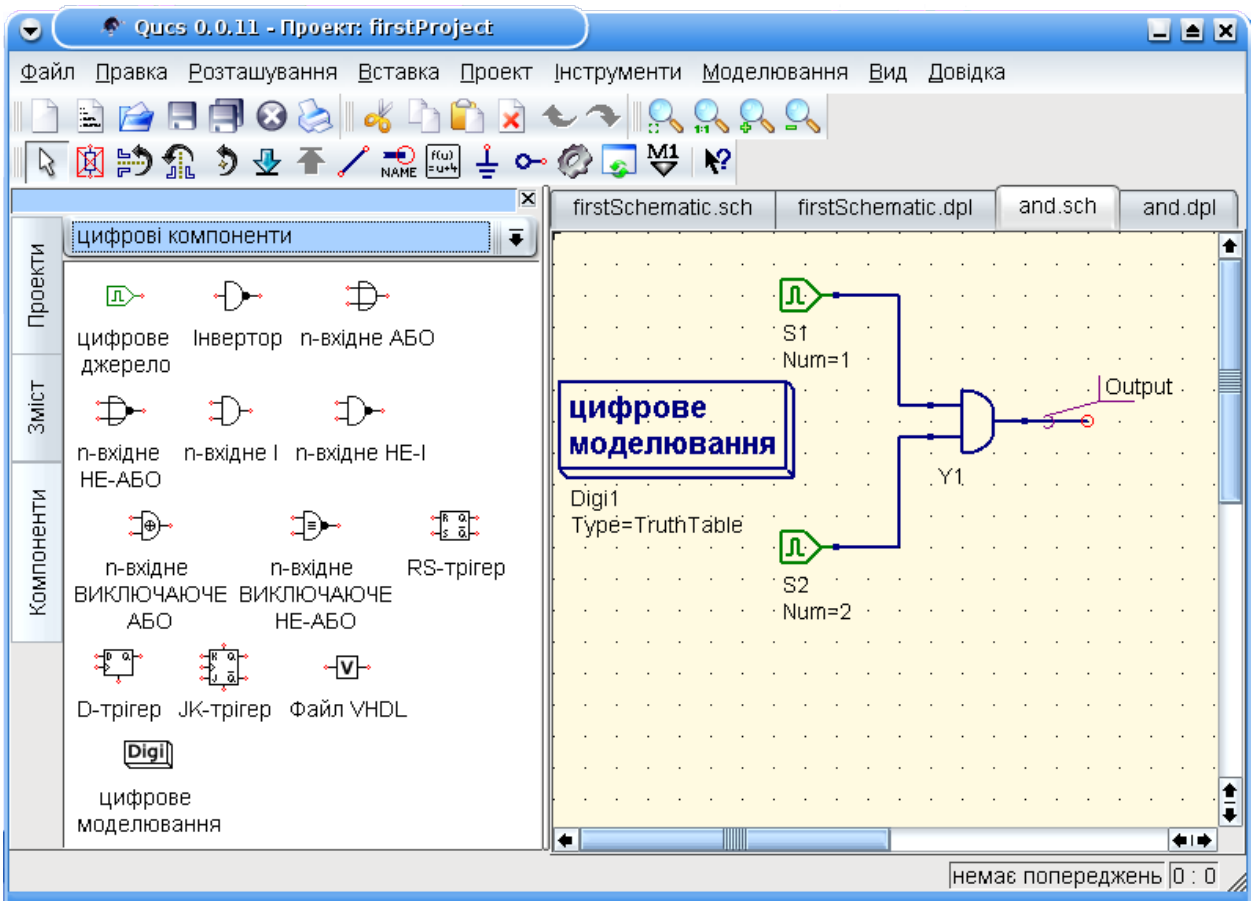
Швидкий старт в цифровому моделюванні

Qucs - надає також графічний інтерфейс користувачу для виконання цифрового моделювання. Даний документ коротенько описує, як для цього користовуватись Qucs.

For digital simulations Qucs uses the FreeHDL program (http://www.freehdl.seul.org). So the FreeHDL package as well as the GNU C++ compiler must be installed on the computer.

There is no big difference in running an analog or a digital simulation. So having read the Getting Started for analog simulations, it is now easy to get a digital simulation work. Let us compute the truth table of a simple logical AND cell. Select the digital components in the combobox of the components tab on the left-hand side and build the circuit shown in figure 1. The digital simulation block can be found among the other simulation blocks.

Цифрові джерела S1 і S2 підключені до входів, вузол з міткою Output є виходом. Після виконання моделювання відкривається сторінка показу даних. Помістіть на неї діаграму Таблиця істинності і вставте зміну Output. Тепер показується таблиця істинності дво-вхідного елемента I. Поздоровлення, перше цифрове моделювання зроблено!



Мал. 1 - Головне вікно Qucs

Таблиця істинності - далеко не єдиний вид моделювання, котрий можна виконати в Qucs. Можливо також подати на вхід випадковий сигнал і подивитися вихідний сигнал в часовій діаграмі. Для цього, треба поміняти параметр Type блоку моделювання на TimeList і у наступному параметрі слід ввести тривалість моделювання. Тепер в цифрових джерел інший зміст: вони можуть видавати випадкову послідовність бітів, для цього їм потрібно вказати перший біт (низький чи високий) і список моментів часу наступної зміни стану. Зверніть увагу, що цей перелік повторюється після кінця. Тому, щоб отримати тактові імпульси з частотою 1 ГГц і скважністю 1:1, у списку має бути записано: 0.5ns; 0.5ns.

Для відображення результатів цього моделювання є часова діаграма. У ній результати всіх вихідних сигналів можуть бути зображені пострічково в одній діаграмі. Отож успіхів у цій справі...

7.1 Файловий компонент VHDL

Більш складні і більш універсальні види моделювання можуть бути виконані з допомогою компонента "файл VHDL". Цей компонент можна взятий з списку компонентів (розділ "цифрові компоненти"). Проте, рекомендується наступний спосіб: файл VHDL повинен бути в складі проекту. Потім перейдіть в перегляд вмісту проекту й натисніть ім'я файла. Зайшовши у область побудови схем, помістіть компонент VHDL.

Останній об'єктний блок в файлі VHDL визначає інтерфейс, тобто тут повинні бути оголошені всі вхідні і вихідні виводи. Такі виводи показуються також на схемному позначенні і можуть бути з'єднані з іншою схемою. Під час моделювання вихідний код файла VHDL поміститься у VHDL-файл верхнього

рівня. Це треба враховувати, оскільки це веде до деяких обмежень. Наприклад, імена об'єктів у VHDL-файлі мають відрізнятись від іменам, вже даних підсхемам. (Після моделювання повний вихідний код можна переглянути, натиснувши F6. Користуйтеся цим, щоб відчути процес.)

Розділ 8

Короткий опис математичних функцій

У рівняннях Qucs можна застосовувати наступні операції, та функції. Для детальнішого опису будь-ласка зверніться до “Measurement Expressions Reference Manual”. Параметри в квадратних дужках “[]” не обовязкові.

8.1 Оператори

8.1.1 Арифметичні оператори

`+x`	унарний плюс
`-x`	унарний мінус
`x+y`	додавання
`x-y`	віднімання
`x*y`	множення
`x/y`	ділення
`x%y`	залишок від ділення
`x^y`	піднесення до степеня

8.1.2 Логічні оператори

`!x`	Заперечення
`x&&y`	І
`x||y`	Або
`x^^y`	Виключаюче або
`x?y:z`	Abbreviation for conditional expression - if `x` then `y` else `z`
`x==y`	Тотожність
`x!=y`	Не дорівнює
`x<y`	Менше
`x<=y`	Менше рівно
`x>y`	Більше
`x>=y`	Більше рівно

8.2 Математичні функції

8.2.1 Вектори та матриці: Створення

`eye(n)`	n x n одинична матриця
`length(y)`	Returns the length of the `y` vector
`linspace(from,to,n)`	Real vector with `n` lin spaced components between `from` and `to`
`logspace(from,to,n)`	Real vector with `n` log spaced components between `from` and `to`

8.2.2 Вектори та матриці: Базові матричні функції

`adjoint(x)`	сполучена матриця (транспонована і комплексно-спряжена)
`det(x)`	детермінант x
`inverse(x)`	інверсія матриці x
`transpose(x)`	транспонована матриця x (рядки - і стовпчики міняються місцями)

8.2.3 абсолютне значення, модуль комплексного числа

`abs(x)`	абсолютне значення, модуль комплексного числа
`angle(x)`	фаза в радіанах
`arg(x)`	спряжене комплексне число
`conj(x)`	спряжене комплексне число
`deg2rad(x)`	Функція Евклідової відстані
`hypot(x,y)`	Функція Евклідової відстані
`imag(x)`	те саме, що і abs(x)
`mag(x)`	квадрат mag(x)
`norm(x)`	фаза в градусах
`phase(x)`	фаза в градусах
`polar(m,p)`	Transform polar coordinates `m` and `p` into a complex number
`rad2deg(x)`	перетворює радіани в градуси
`real(x)`	дійсна частина комплексного числа
`sign(x)`	обчислення знаку функції
`sqr(x)`	квадратний корінь
`sqrt(x)`	квадратний корінь
`unwrap(p[,tol[,step]])`	Unwrap angle `p` (radians) – defaults `step` = 2pi, `tol` = pi

8.2.4 експоненціальна функція з основою e

exp(x)	експоненціальна функція з основою e
limexp(x)	Обмежена експоненціальна функція
log10(x)	десятковий логарифм
log2(x)	логарифм з основою два
ln(x)	натуральний логарифм

8.2.5 Елементарні математичні функції: Тригонометричні функції

cos(x)	косеканс
cosec(x)	косеканс
cot(x)	котангенс
sec(x)	секанс
sin(x)	тангенс
tan(x)	тангенс

8.2.6 Елементарні математичні функції: Оберенні тригонометричні функції

arccos(x)	арккосинус
arccosec(x)	арккосеканс
arccot(x)	арккотангенс
arcsec(x)	арксеканс
arcsin(x)	арксинус
arctan(x[,y])	арктангенс

8.2.7 косинус гіперболічний

cosh(x)	косинус гіперболічний
cosech(x)	косеканс гіперболічний
coth(x)	котангенс гіперболічний
sech(x)	секанс гіперболічний
sinh(x)	синус гіперболічний
tanh(x)	тангенс гіперболічний

8.2.8 арккосинус гіперболічний

arcosh(x)	арккосинус гіперболічний
arcosech(x)	арккосеканс гіперболічний
arcoth(x)	арккотангенс гіперболічний
arsech(x)	арксеканс гіперболічний
arsinh(x)	арксинус гіперболічний
artanh(x)	арктангенс гіперболічний

8.2.9 Елементарні математичні функції: Округлення

`ceil(x)`	округлення до найближчого більшого цілого
`fix(x)`	відкидає дробові розряди дійсного числа
`floor(x)`	округлення до найближчого меншого цілого
`round(x)`	округлення до найближчого цілого

8.2.10 Елементарні математичні функції: Спеціальні математичні функції

`besseli0(x)`	модифікована функція Бесселя нульового порядку
`besselj(n,x)`	функція Бесселя 1-го роду n-го порядку
`bessely(n,x)`	функція Бесселя 2-го роду n-го порядку
`erf(x)`	функція помилки
`erfc(x)`	компліментарна функція помилки
`erfinv(x)`	інверсна функція помилки
`erfcinv(x)`	інверсна компліментарна функція помилки
`sinc(x)`	повертає `sin(x)/x` і одиницю при `x=0`
`step(x)`	avg(x[,range])

8.2.11 Аналіз даних: Базова статистика

`avg(x[,range])`	Average of vector `x`. If range given `x` must have a single data dependency
`cumavg(x)`	накопичувальне середнє значення в векторі
`max(x,y)`	повертає більше з значень x і y
`max(x[,range])`	Maximum of vector `x`. If range given `x` must have a single data dependency
`min(x,y)`	повертає менше з значень x і y
`min(x[,range])`	Minimum of vector `x`. If range is given `x` must have a single data dependency
`rms(x)`	среднеквадратичне значення по вектору
`runavg(x)`	ковзне середнє значення в векторі
`stddev(x)`	видає випадкове число
`variance(x)`	розбіжність значень в векторі
`random()`	випадкове число між 0.0 та 1.0
`srandom(x)`	видає випадкове число

8.2.12 Аналіз даних: Базові операції

`cumprod(x)`	накопичувальний добуток значень в векторі
`cumsum(x)`	накопичувальна сума значень в векторі
`interpolate(f,x[,n])`	Spline interpolation of vector `f` using `n` equidistant points of `x`
`prod(x)`	добуток значень в векторі
`sum(x)`	сума значень в векторі
`xvalue(f,yval)`	Returns x-value nearest to `yval` in single dependency vector `f`
`yvalue(f,xval)`	Returns y-value nearest to `xval` in single dependency vector `f`

8.2.13 Аналіз даних: Диференциювання та інтегрування

`ddx(expr,var)`	Derives mathematical expression `expr` with respect to the variable `var`
`diff(y,x[,n])`	Differentiate vector `y` with respect to vector `x` `n` times. Defaults to `n` = 1
`integrate(x,h)`	чисельно інтегрує вектор x, приймаючи постійний розмір кроку h

8.2.14 Аналіз даних: Обробка сигналів

`dft(x)`	обчислює дискретне перетворення Фур'є (DFT) вектора x
`fft(x)`	обчислює швидке перетворення Фур'є (FFT) вектора x
`fftshift(x)`	Shuffles the FFT values of vector `x` to move DC to the center of the vector
`Freq2Time(V, f)`	обчислює зворотне дискретне перетворення Фур'є функції V(f) з фізичною інтерпретацією
`idft(x)`	обчислює зворотне дискретне перетворення Фур'є (DFT) вектора x
`ifft(x)`	обчислює зворотне швидке перетворення Фур'є (IFFT) вектора x
`kbd(x[,n])`	децибели напруги
`Time2Freq(v, t)`	обчислює дискретне перетворення Фур'є функції v(t) з фізичною інтерпретацією

8.3 Функції Електроніки

8.3.1 Перетворення величин

`dB(x)`	децибели напруги
`dbm(x)`	перетворити напругу у потужність в дБ
`dbm2w(x)`	перетворити потужність в dBm в потужність в ваттах
`w2dbm(x)`	перетворити потужність в ваттах в потужність в dBm
`vt(t)`	Функція залежності опору від температури t в Кельвінах

8.3.2 Коефіцієнти відображення та VSWR

`rtoswr(x)`	перетворює коефіцієнт відображення в коефіцієнт стоячої хвилі (за напругою) (КСВ чи КСВН)
`rtoy(x[, zref])`	Converts reflection coefficient to admittance; default `zref` = 50 ohms
`rtoz(x[, zref])`	Converts reflection coefficient to impedance; default `zref` = 50 ohms
`ytor(x[, zref])`	Converts admittance to reflection coefficient; default `zref` = 50 ohms
`ztor(x[, zref])`	Converts impedance to reflection coefficient; default `zref` = 50 ohms

8.3.3 Робота з N-портовою матрицею

`stos(s,zref[,z0])`	Converts S-parameter matrix to S-parameter matrix with a different Z0
`stoy(s[,zref])`	перетворює матрицю s-параметрів в матрицю y-параметрів
`stoz(s[,zref])`	перетворює матрицю s-параметрів в матрицю z-параметрів
`twoport(m,from,to)`	Converts a two-port matrix: `from` and `to` are 'Y', 'Z', 'H', 'G', 'A', 'S' and 'T'.
`ytos(y[,z0])`	перетворює матрицю y-параметрів в матрицю s-параметрів
`ytoz(y)`	перетворює матрицю y-параметрів в матрицю z-параметрів
`ztos(z[,z0])`	перетворює матрицю z-параметрів в матрицю s-параметрів
`ztoy(z)`	перетворює матрицю z-параметрів в матрицю y-параметрів

8.3.4 Підсилювачі

`GaCircle(s,Ga[,arcs])`	Available power gain `Ga` circles (source plane)
`GpCircle(s,Gp[,arcs])`	Operating power gain `Gp` circles (load plane)
`Mu(s)`	Mu чинник стійкості для матриці x (матриця S-параметрів чотирьохполюсника)
`Mu2(s)`	Mu' чинник стійкості для матриці x (матриця S-параметрів чотирьохполюсника)
`NoiseCircle(Sopt,Fmin,Rn, F[,Arcs])`	Noise Figure(s) `F` circles
`PlotVs(data,dep)`	Returns data selected from `data`: dependency `dep`
`Rollet(s)`	Чинник стійкості Роллета для матриці x (матриця S-параметрів чотирьохполюсника)
`StabCircleL(s[,arcs])`	окружність стійкості у площині навантаження
`StabCircleS(s[,arcs])`	окружність стійкості у площині джерела
`StabFactor(s)`	Stability factor of a two-port S-parameter matrix
`StabMeasure(s)`	Границі стабільності B1 для двохпортової матриці S-параметру

8.4 Номенклатура

8.4.1 Інтервали

`LO:HI`	інтервал від LO до HI
`:HI`	аж до HI
`LO:`	від LO
`:`	немає границі інтервалу

8.4.2 Матриці

`M`	вся матриця M
`M[2,3]`	елемент, який перебуває у 2-гому рядку і 3-му стовпці матриці M
`M[:,3]`	вектор, утворений з 3-го стовпця матриці M

8.4.3 Назви величин

`2.5`	Real number
`1.4+j5.1`	Complex number
`[1,3,5,7]`	Vector
`[11,12;21,22]`	Matrix

8.4.4 Number suffixes

`E`	exa, 1e+18
`P`	peta, 1e+15
`T`	tera, 1e+12
`G`	giga, 1e+9
`M`	mega, 1e+6
`k`	kilo, 1e+3
`m`	milli, 1e-3
`u`	micro, 1e-6
`n`	nano, 1e-9
`p`	pico, 1e-12
`f`	femto, 1e-15
`a`	atto, 1e-18

8.4.5 Назви величин

`S[1,1]`	значення S-параметра
nodename.`V`	постійна напруга в вузлі nodename
name.`I`	постійний струм через компонент name
nodename.`v`	змінна напруга у вузлі nodename
name.`i`	змінний струм через компонент name
nodename.`vn`	шумова напруга змінного струму в вузлі nodename
name.`in`	шумовий змінний струм через компонент name
nodename.`Vt`	перехідна напруга у вузлі nodename
name.`It`	перехідний струм через компонент name

Примітка: Усі напруги і струми виражені піковими значеннями. Примітка: Шумові напруги виражені СКЗ значеннями в смузі частот в 1Гц.

8.5 Константи

`i, j`	уявна одиниця (“квадратний корінь з -1”)
`pi`	4*arctan(1) = 3.14159...
`e`	Euler = 2.71828...
`kB`	Постійна Больцмана = 1.38065e-23
`q`	Елементарний заряд = 1.6021765e-19 C

Розділ 9

Перелік спеціальних символів

У компоненті “Текст” й у тексті міток осей діаграм можна використовувати спеціальні символи. Це робиться з допомогою тегів LaTeX. У наступній таблиці наводиться перелік наявних на сьогодні символів.

Примітка: Правильне відображення цих символів залежить від шрифту, що використовується у Qucs!

Друковані грецькі літери

Тег LaTeX	Юнікод	Опис
`\alpha`	0x03B1	alpha
`\beta`	0x03B2	beta
`\gamma`	0x03B3	gamma
`\delta`	0x03B4	delta
`\epsilon`	0x03B5	epsilon
`\zeta`	0x03B6	zeta
`\eta`	0x03B7	eta
`\theta`	0x03B8	theta
`\iota`	0x03B9	iota
`\kappa`	0x03BA	kappa
`\lambda`	0x03BB	lambda
`\mu`	0x03BC	mu
`\textmu`	0x00B5	mu
`\nu`	0x03BD	nu
`\xi`	0x03BE	xi
`\pi`	0x03C0	pi
`\varpi`	0x03D6	pi
`\rho`	0x03C1	rho
`\varrho`	0x03F1	rho
`\sigma`	0x03C3	sigma
`\tau`	0x03C4	tau
`\upsilon`	0x03C5	upsilon
`\phi`	0x03C6	phi
`\chi`	0x03C7	chi
`\psi`	0x03C8	psi
`\omega`	0x03C9	omega

Прописні грецькі літери

Тег LaTeX	Юнікод	Опис
`\Gamma`	0x0393	Gamma
`\Delta`	0x0394	Delta
`\Theta`	0x0398	Theta
`\Lambda`	0x039B	Lambda
`\Xi`	0x039E	Xi
`\Pi`	0x03A0	Pi
`\Sigma`	0x03A3	Sigma
`\Upsilon`	0x03A5	Upsilon
`\Phi`	0x03A6	Phi
`\Psi`	0x03A8	Psi
`\Omega`	0x03A9	Omega

Математичні символи

Тег LaTeX	Юнікод	Опис
`\cdot`	0x00B7	знак множення - точка (центрована точка)
`\times`	0x00D7	знак множення - хрестик
`\pm`	0x00B1	знак плюс-мінус
`\mp`	0x2213	знак мінус плюс
`\partial`	0x2202	знак часткового диференціювання
`\nabla`	0x2207	набла-оператор
`\infty`	0x221E	знак нескінченності
`\int`	0x222B	знак інтеграла
`\approx`	0x2248	символ наближення (хвилястий знак рівності)
`\neq`	0x2260	знак не рівно
`\in`	0x220A	символ “міститься у”
`\leq`	0x2264	знак меньше-рівно
`\geq`	0x2265	знак більше-рівно
`\sim`	0x223C	(центральноєвропейський) знак пропорційності
`\propto`	0x221D	(американський) знак пропорційності
`\diameter`	0x00F8	знак діаметра (також знак середнього)
`\onehalf`	0x00BD	половина
`\onequarter`	0x00BC	чверть
`\twosuperior`	0x00B2	квадрат (степінь 2)
`\threesuperior`	0x00B3	степінь 3
`\ohm`	0x03A9	одиниця для опору (прописна грецька омега)

Розділ 10

Узгодження електричних кіл

Створення узгоджених електричних кіл часто потрібне для мікрохвильової технології. Qucs може робити це автоматично. Необхідні для цього кроки:

Виконати моделювання S-параметрів, щоб розрахувати коефіцієнт відображення.

Вставити діаграму, щоб показати коефіцієнт відображення (тобто, S[1,1] для порту 1, S[2,2] для порту 2 тощо.)

Помістити на графік маркер і рухатися кроками до необхідної частоти.

Натиснути праву кнопку миші на маркері і вибрати "узгодження потужності" в меню яке з'явився.

З'являється діалогове вікно, де можна налаштувати значення, наприклад, повний опорний опір, можливо вибрати відмінним від 50 Ом.

Після натискання на кнопку "створити" відбувається повернення до схеми, і з допомогою курсору миші можна вибрати місце для вставки узгодженого електричного кола.

З'являється діалогове вікно, де можна налаштувати значення, наприклад, повний опорний опір, можливо вибрати відмінним від 50 Ом.

If the marker points to a variable called "Sopt", the menu shows the option "noise matching". Note that the only different to "power matching" is the fact that the conjugate complex reflexion coefficient is taken. So if the variable has another name, noise matching can be chosen by re-adjusting the values in the dialog.

Діалог здійснення узгодження може бути також викликаний з допомогою меню (Інструменти->узгодження електричних кіл) чи з допомогою комбінації клавіш (CTRL-5). Але всі значення повинні вводитися вручну.

10.1 Узгодження чотирьохполюсників

Якщо ім'я змінної з тексту маркера є S-параметром, то існує можливість одночасного узгодження входу і виходу чотирьохполюсного ланцюга. Це працює досить схоже на вищеописані кроки. Результатом є два узгоджені ланцюга: самий лівий вузол повинен з'єднуватися з виводом 1, більш правий вузол - з виводом 2, а через два вузла у середині повинні з'єднуватися з чотирьохполюсником.

Розділ 11

Встановлені файли

До складу Qucs входять кілька програм. Їх встановлюють під час процесу інсталяції. Шлях, куди встановлюється Qucs, визначається при встановленні (скриптом configure). У наступних поясненнях приймається шлях за замовчуванням (/usr/local/).

- /usr/local/bin/qucs - графічний інтерфейс
- /usr/local/bin/qucsator - симулятор (консольна програма)
- /usr/local/bin/qucsedit - простий текстовий редактор
- /usr/local/bin/qucshelp - невеличка програма для перегляду довідкової інформації
- /usr/local/bin/qucstrans - програма для розрахунку параметрів ліній передач
- /usr/local/bin/qucsfilter - програма синтезу фільтрів
- /usr/local/bin/qucsconv - перетворювач форматів файлів (консольна програма)

Усі програми є самостійними додатками і можуть працювати незалежно один від одного. Головна програма (графічний інтерфейс)

- викликає qucsator при виконанні моделювання,
- викликає qucsedit, коли демонструються текстові файли,
- викликає qucshelp, коли показується довідкова система,
- викликає qucstrans при виклику цієї програми з меню “Інструменти”,
- викликає qucsfilter при виклику цієї програми з меню “Інструменти”,
- викликає qucsconv, коли вставляється компонент SPICE і коли виконується моделювання з допомогою компонента SPICE.

Крім цього, за умови встановлення створюються такі папки:

- /usr/local/share/qucs/bitmaps - містить всі растрові зображення (значки тощо.)
- /usr/local/share/qucs/docs - містить HTML-документи довідкової системи
- /usr/local/share/qucs/lang - містить файли перекладів

11.1 Аргументи командного рядку

qucs [файл1 [файл2 ...]]

qucsator [-b] -i список_кіл -o набір_даних (b = смуга прогресу)

qucsedit [-r] [файл] (r = лише читання)

qucshelp (без аргументів)

qucsconv -if spice -of qucs -i netlist.inp -o netlist.net

Розділ 12

Qucs File Formats

This document describes the schematic and library file formats of Qucs.

12.1 Schematic file format

This format is used for schematics (usually with suffix `.sch`) and for data displays (usually with suffix `.dpl`). The following text shows a short example of a schematic file.

```
<Qucs Schematic 0.0.6>
<Properties>
  <View=0,0,800,800,1,0,0>
</Properties>
<Symbol>
  <.ID -20 14 SUB>
</Symbol>
<Components>
  <R R1 1 180 150 15 -26 0 1 "50 Ohm" 1 "26.85" 0 "european" 0>
  <GND * 1 180 180 0 0 0 0>
</Components>
<Wires>
  <180 100 180 120 "" 0 0 0 "">
  <120 100 180 100 "Input" 170 70 21 "">
</Wires>
<Diagrams>
  <Polar 300 250 200 200 1 #c0c0c0 1 00 1 0 1 1 1 0 5 15 1 0 1 1 315 0 225 "" "" "">
    <"acnoise2:S[2,1]" #0000ff 0 3 0 0 0>
    <Mkr 6e+09 118 -195 3 0 0>
  </Polar>
</Diagrams>
<Paintings>
  <Arrow 210 320 50 -100 20 8 #000000 0 1>
</Paintings>
```

У файлі є кілька розділів. Усі вони пояснюються нижче. Кожна лінія складається лише з одного інформаційного блоку, який починається знаком "менше" (<) і який закінчується знаком "більше" (>).

12.1.1 Властивості

Перший розділ починається з `<Properties>` і закінчується `</Properties>`. Він містить властивості документа, що знаходиться у файлі. Кожний рядок необов'язковий. Підтримуються такі властивості:

- `<View=x1,y1,x2,y2,scale,xpos,ypos>` містить становище у пікселях вікна схеми (перші чотири числа), його поточний масштаб і поточний стан верхнього лівого кута (останні два числа).
- `<Grid=x,y,on>` містить крок сітки в пікселях (перші два числа) і включена сітка (останнє число = 1) чи виключена (останнє число = 0).
- `<DataSet=name.dat>` містить ім'я файлу набору даних, зв'язаного з цією схемою.
- `<DataDisplay=name.dpl>` містить ім'я файлу з сторінкою перегляду даних, зв'язаного з цією схемою (чи ім'я схемного файла, чи цей документ є переглядом даних).
- `<OpenDisplay=yes>` містить 1, якщо сторінка показу даних відкривається автоматично після моделювання, інакше - 0.
- `<Script=name.m>` contains the file name of the octave script associated with this schematic.
- `<RunScript=0>` contains 1 if the octave script is executed after the simulation.
- `<showFrame=0>` specify if a frame is drawn and if so which size it is. valid values are 0 (do not show a frame), 1 (A5 landscape), 2 (A5 portrait), 3 (A4 landscape), 4 (A4 portrait), 5 (A3 landscape), 6 (A3 portrait), 7 (letter landscape) and 8 (letter portrait).
- `<FrameText0=NE555 sub-circuit model>`, `FrameText1=Draw by: anonymous`, `FrameText2=Date: 1984`, and `<FrameText3=Revision: 42>` specifiy the texts to be placed into the frame text boxes.

12.1.2 Symbol

Цей поділ починається з `<Symbol>` і закінчується `</Symbol>`. Він містить елементи малювання, складові схемне позначення для файла. Це зазвичай використовується лише у схемних файлів, які вважають підсхемою.

Refers to "Symbol definition" in the "Shared file format" section at the end of this document.

12.1.3 Компоненти

Цей розділ починається з `<Components>` і закінчується `</Components>`. Він містить компоненти ланцюгів схеми. Формат рядки:

```
<type name active x y xtext ytext mirrorX rotate "Value1" visible "Value2" visible ...>
```

- `type` ("тип") означає компонент, наприклад, `R` для резистора, `3` для конденсатора.
- `name` ("ім'я") - унікальне позначення компонента на схемою, наприклад, `R1` на першому резистора.
- `1` на полі `active` ("активний") показує, що це компонент активний, тобто використовують у моделюванні. `0` показує, що він неактивний.
- Наступні два числа є x і y координатами центру компонента.
- Наступні два числа є x і y координатами верхнього лівого кута тексту компонента. Вони відраховуються від центру компонента.

- Наступні два числа свідчить про дзеркальне відображення щодо осі x (`1` - дзеркальне відображення, `0` - немає дзеркального відображення) і обертання проти годинниковий стрілки (кратно 90 градусів, тобто 0...3).
- Наступні параметри є значеннями властивостей компонента (у лапках), що їх слід 1, якщо це властивість певне на схемою (інакше 0).

12.1.4 Wires

Цей розділ починається з `<Wires>` і закінчується `</Wires>`. Він містить провідники (електричне з'єднання між компонентами електричного кола), їх мітки і вузли. Формат рядку:

```
<x1 y1 x2 y2 "label" xlabel ylabel dlabel "node set">
```

- Цей розділ починається з `<Wires>` і закінчується `</Wires>`. Він містить провідники (електричне з'єднання між компонентами електричного кола), їх мітки і вузли. Формат рядку: <code><x1 y1 x2 y2 "label" xlabel ylabel dlabel "node set"></code>
- Перший рядок у лапках - ім'я мітки. Вона порожня, якщо користувач не встановив мітку на цей провідник.
- Наступні два числа - x- і y-координати мітки чи нуль, якщо мітки немає.
- The next number is the distance between the wire starting point and the point where the label is set on the wire.
- Останній рядок у лапках - параметри кола провідника, тобто початкова напруга вузла, що використовується ядром симулятора для пошуку рішення. Цей рядок порожній, якщо користувач не встановив параметри вузла електричного кола для цього провідника.

12.1.5 Diagrams

Цей розділ починається з `<Diagrams>` і закінчується `</Diagrams>`. Він містить діаграми з своїми графіками і маркерами.

```
<diatype x y width height grid gridcolor gridstyle log xAutoscale xmin
 xstep xmax yAutoscale ymin ystep ymax zAutoscale zmin zstep zmax
 xrotate yrotate zrotate "xlabel" "ylabel" "zlabel" "[freq Hz;]*">
    <"graphvar" color thickness precision numberformat style axisside>
    <Mkr x y precision numberformat transparent>
</diatype>
```

Diagram line format:

- The `diatype` token specifies the type of diagram.
- The `x` and `y` numbers are the coordinate of lower left corner.
- The `width` and `height` numbers of diagram boundings.
- The `grid` flags with 1 if grid is on and 0 if grid is off.
- The `gridColor` in 24 bit hexadecimal RGB value, e.g. #FF0000 is red.
- The `gridstyle` is the line style sued of the grid.
- The `log` has two field to flag which axes have logarithmical scale.
- The `xAutoscale`, `xmin`, `xstep`, `xmax` configure the x-axis scaling, limits.

- The `yAutoscale`, `ymin`, `ystep`, `ymax` configure the y-axis scaling, limits.
- The `zAutoscale`, `zmin`, `zstep`, `zmax` configure the z-axis scaling, limits.
- The `xrotate`, `yrotate`, `zrotate` numbers set the 3D rotation.
- The `xlabel`, `ylabel`, `zlabel` hold the labels used on each axis.
- The list of frequencies `"[freq Hz;]*"` is used by `Phasor` and `Waveac`.

Here is a list of known diagram types:

- `Curve` for a locus curve diagram.
- `Smith` for an impedance Smith diagram.
- `ySmith` for an admittance Smith diagram.
- `PS` for a mixed polar/smith diagram.
- `SP` for a upper-half mixed polar/smith diagram.
- `Polar` for a polar diagram.
- `Rect` for a 2D-cartesian diagram.
- `Rect3D` for a 3D-cartesain diagram.
- `Tab` for a tabular diagram.
- `Time` for a timing diagram.
- `Truth` for a truth-table diagram.
- `Phasor` for a complex phasor diagram.
- `Waveac` for a wave as temporal diagram.

Graph line format:

- The `graphvar` specify the variable this graph is plotting for.
- The `color`, `thickness` and `style` refers to the pen used to draw the curve.
- The `precision` specify the number of digits used when displaying data values.
- The `numberformat` is an integer that specify how the number are formated (0 for real/imag, 1 for polar/deg and 2 for polar/rad).
- The `axisside` is an integer indicating on which side the Y axis should be placed ().

Marker line format:

- The `x` and `y` are the location of the marker.
- The `precision` ...
- The `numberformat` ...
- The `transparent`

12.1.6 Paintings

Цей розділ починається з `<Paintings>` і закінчується `</Paintings>`. Він містить елементи малювання, які є в схемі.

Refers to “Shared file format” section below.

12.2 Library file format

This format is used for libraries (usually with suffix `.lib`). The following text shows a short example of a library file.

```
<Qucs Library 0.0.14 "Ideal">
<DefaultSymbol>
  <.ID -26 13 D>
  <Line -30 0 60 0 #000080 2 1>
  <Line -6 -9 0 18 #000080 2 1>
  <Line 6 -9 0 18 #000080 2 1>
  <Line -6 0 12 -9 #000080 2 1>
  <Line -6 0 12 9 #000080 2 1>
  <Line -6 9 4 0 #000080 2 1>
  <.PortSym -30 0 1 0>
  <.PortSym 30 0 2 180>
</DefaultSymbol>
<Component VSum>
  <Description>
Voltage adder
  </Description>
  <Model>
.Def:Ideal_AP1 _net3 _net2 fc="1E3"
Sub:VSUB1 _net0 _net1 _net2 Type="VSub"
Sub:LP1F1 _net3 _net0 Type="LP1" fc="fc2" V0="0"
Sub:HP1F1 _net3 _net1 Type="HP1" fc="fc2"
Eqn:Eqn1 fc2="fc/0.6436" Export="yes"
.Def:End
  </Model>
  <ModelIncludes "HP1.sch.lst" "LP1.sch.lst" "VSub.sch.lst">
  <Symbol>
    <Ellipse -20 -20 40 40 #000080 2 1 #c0c0c0 1 0>
    <Line -10 0 20 0 #000080 1 1>
    <Line 0 -10 0 20 #000080 1 1>
    <Line 0 30 0 -10 #000080 2 1>
    <.PortSym 0 30 2 0>
    <.PortSym 30 0 3 180>
    <Line 20 0 10 0 #000080 2 1>
    <.ID 10 14 VADD>
    <Line 0 -20 0 -10 #000080 2 1>
    <.PortSym 0 -30 1 0>
  </Symbol>
</Component>
```

The first line specify that this file is a Qucs library file generated by Qucs 0.0.14 and that the library is named "Ideal".

The file contains on optional `DefaultSymbol` section, followed by `Component` sections. Each section is explained below.

12.2.1 Default symbol

This section starts with `<DefaultSymbol>` and ends with `</DefaultSymbol>`. It contains painting elements creating a default schematic symbol for any subsequent component declaration that doesn't define its own.

Refers to "Shared file format" section below.

12.2.2 Component

This section starts with `<Component>` and ends with `</Component>`. It contains the component definition for use with schematic documents.

The component section is an aggregation of the following sub-sections:

- `<Description>` and `</Description>` contain lines of free text describing the component function.
- `<Model>` and `</Model>` contain the Qucsator netlist lines for this component.
- `<ModelIncludes "value0value1"...>` ...
- `<Spice>` and `</Spice>`` are optional and contain the Spice netlist lines for this component.
- `<Symbol>` and `</Symbol>` are optional and contain painting elements defining the schematic symbol to be used with this component. Refers to "Symbol definition" section below.

12.3 Shared file format

12.3.1 Painting elements

A painting line can be found in:

- The `Paintings` section of a schematic file.
- The `Symbol` sections of a schematic file.
- The `DefaultSymbol` section of a library file.
- The `Symbol` section (sub-section of `Component`) of a library file.

A painting line has one of the following format:

- `<Rectangle x y width height pencolor penwidth penstyle brushcolor brushstyle filled>` ...
- `<Ellipse x y width height pencolor penwidth penstyle brushcolor brushstyle filled>` ...
- `<EArc x y startangle spanangle width height pencolor penwidth penstyle brushcolor brushstyle filled>` ...
- `<Text x y size color angle "text">` ...
- `<Line x1 y1 x2 y2 pencolor penwidth penstyle >` ...
- `<Arrow x1 y1 x2 y2 x3 y3 pencolor penwidth penstyle >` ...

12.3.2 Symbol definition

A symbol definition can contains any painting element as described in the previous section. In addition to the painting elements, a symbol definition must contain one `.ID` line and one or more `.PortSym` lines.

The `.ID` line has the following format:

```
<.ID x y name "property1" "property2" ...>
```

Where:

- `x` and `y` are the center coordinates of the symbol.

- `name` will be used as a name prefix when instanciating this symbol on a schematic sheet.
- `propertyX` are used for symbol definition within a schematic file, these parameter will be associated with the symbol instance and communicated to the sub-schematic. The format for such a property is `` `displayed=name=value=description=unknown` ``.

The `.PortSym` line has the following format:

```
<.PortSym x y caption angle>
```

Where:

- `x` and `y` are the coordinates of the port.
- `caption` is the name/caption of the port.
- `angle` is an angle value, it is ignored (backward compatibility).

Розділ 13

Subcircuit and Verilog-A RF Circuit Models for Axial and Surface Mounted Resistors

Mike Brinson

13.1 Introduction

Resistors are one of the fundamental building blocks in electronic circuit design. In most instances conventional resistor circuit simulation models are characterized by I/V characteristics specified by Ohm's law. In reality the impedance of RF resistors is frequency dependent, being determined by component physical properties, component manufacturing technology and how components are connected in a circuit. At low frequencies fixed resistors have a nominal value at roomtemperature and can be modelled accurately by Ohm's law. At RF frequencies the fact that a resistor acts more like an inductance or a capacitance can play a crucial role in determining whether or not a circuit operates as designed. Similarly, if a resistor is modelled as an ideal component at a frequency where it exhibits significant reactive properties then the resulting simulation data are likely to be incorrect. The subcircuit and Verilog-A compact resistor models introduced in this Qucs note are designed to give good performance from low frequencies to RF frequencies not greater than a few GHz.

13.2 RF Resistor Models

The schematic symbol, I/V equation and parameters of the Qucs linear resistor model are shown in Figure 1. In contrast to this model Figure 2 illustrates the structure of a printed circuit board (PCB) mounted metal film (MF) axial RF resistor (a), its Qucs schematic symbol (b) and its equivalent circuit model (c). A thin film surface mounted (SMD) resistor can also be represented by the model shown in Figure 2 (c).

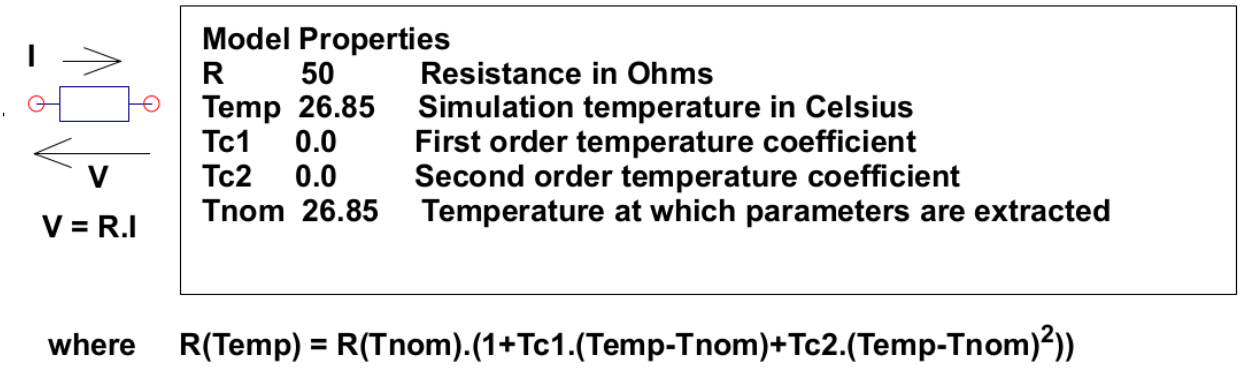


Figure 1 - Qucs built-in resistor model.

At signal frequencies where the largest dimension of an axial or SMD resistor is less than approximately 20 times the smallest signal wavelength a resistor can be modelled by a lumped passive circuit consisting of a resistor **Rs** in series with a small inductance **Ls** with the combination shunted by parasitic capacitor **Cp**. In Figure 2 **Rs** is the nominal value of resistor at its parameter extraction temperature **Tnom**, **Ls** represents the inductance associated with **Rs** where the value of **Ls** is largely determined by the trimming method employed during component manufacture to set the value of **Rs** to a specified tolerance. Similarly, capacitor **Cp** models a parasitic capacitance associated with **Rs** where the value of **Cp** is a function of the physical size of **Rs**. At RF frequencies it is important, for accurate operation, to add lead parasitic elements to the intrinsic equivalent circuit model shown within the red box draw in Figure 2. In Figure 2 **Llead** and **Cshunt** represent resistor series lead inductance and shunt capacitance to ground respectively.

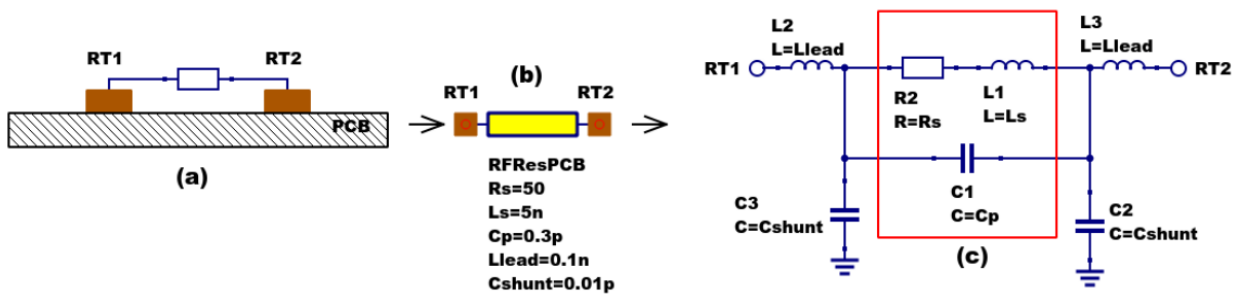


Figure 2 - PCB mounted resistor: (a) axial component mounting, (b) Qucs symbol and (c) equivalent circuit model.

A typical set of model parameters for a 51 Ω 5 % MF axial resistor are (1) **Ls** = **8nH**, **Cp** = **1pF**, **Llead** = **1nH** and **Cshunt** = **0.1pF**. Illustrated in Figure 3 is a basic S parameter test bench circuit for measuring the S parameters of an RF resistor over a frequency range 1 MHz to 1.3 GHz. This example also demonstrates how the real and imaginary parts of a resistor model impedance can be extracted from S parameter simulation data. The graphs in Figure 3 clearly demonstrate that the impedance of the typical MF RF resistor described in previous text and modelled by the equivalent circuit shown in Figure 2 is a strong function of frequency at higher frequencies in the band 1 MHz to 1.3 GHZ.

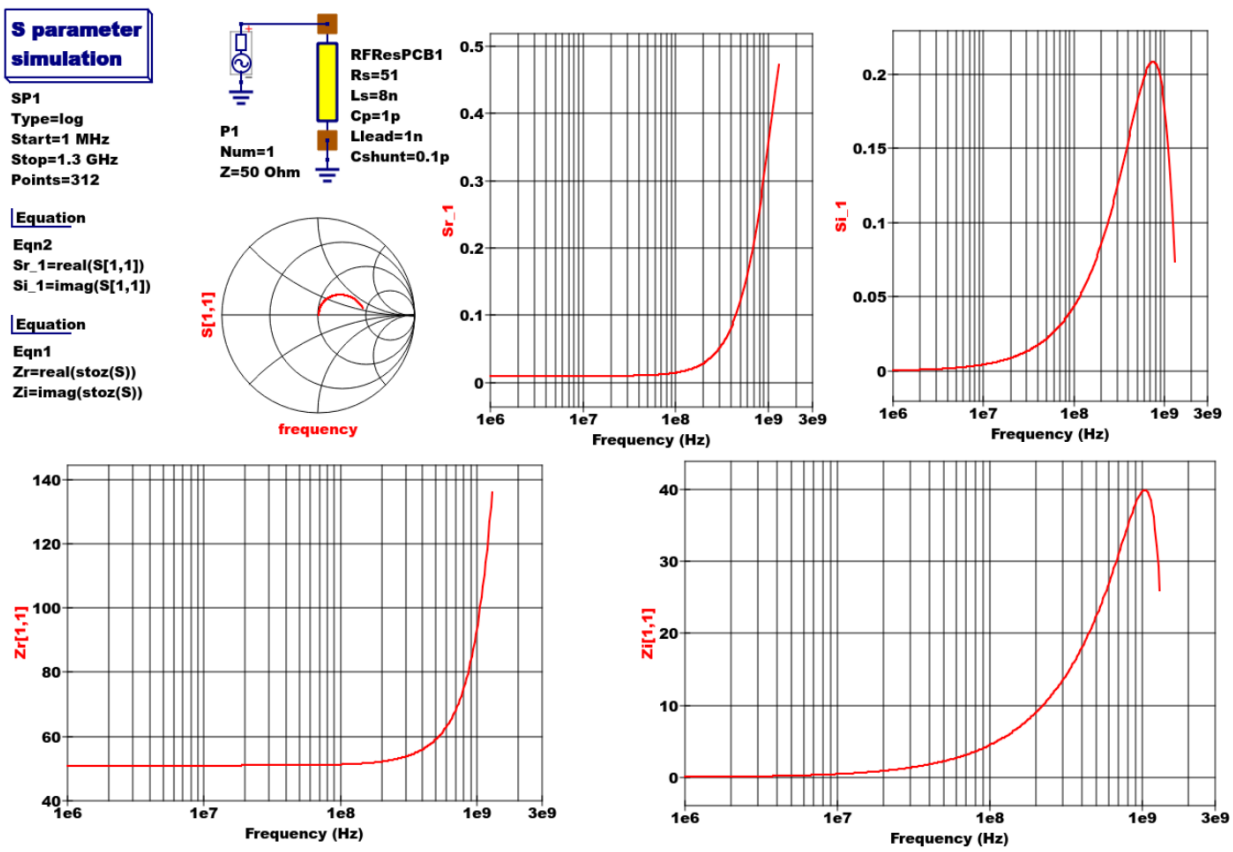


Figure 3 - Qucs S parameter simulation test circuit and plotted output data for a MF axial resistor: Rs=51Ω, Ls=8nH, Cp=1pF, $Llead$=1nH and $Cshunt$=0.1pF.

13.3 Analysis of the RF resistor model

A component level version of the proposed RF resistor model is shown in Figure 4, where

$$Z1 = j \cdot \omega \cdot Llead$$

$$Z2 = \frac{Rs + j \cdot \omega \cdot Ls \cdot (1 - \omega^2 \cdot Cp \cdot Ls) - j \cdot \omega \cdot Cp \cdot Rs^2}{(1 - \omega^2 \cdot Cp \cdot Ls)^2 + (\omega \cdot Cp \cdot Rs)^2}$$

$$Z3 = \frac{j \cdot \omega \cdot Llead}{(1 - \omega^2 \cdot Llead \cdot Cshunt)}$$

$$Zseries = Z1 + Z2 = Rseries + j \cdot Xseries$$

$$Zb = Zseries || XCshunt = \frac{Zseries}{(1 + j \cdot \omega \cdot Cshunt \cdot Zseries)} = ZBR + j \cdot \omega \cdot ZBI,$$

$$Z = j \cdot \omega \cdot Llead + Zb = ZR + j \cdot \omega \cdot ZI.$$

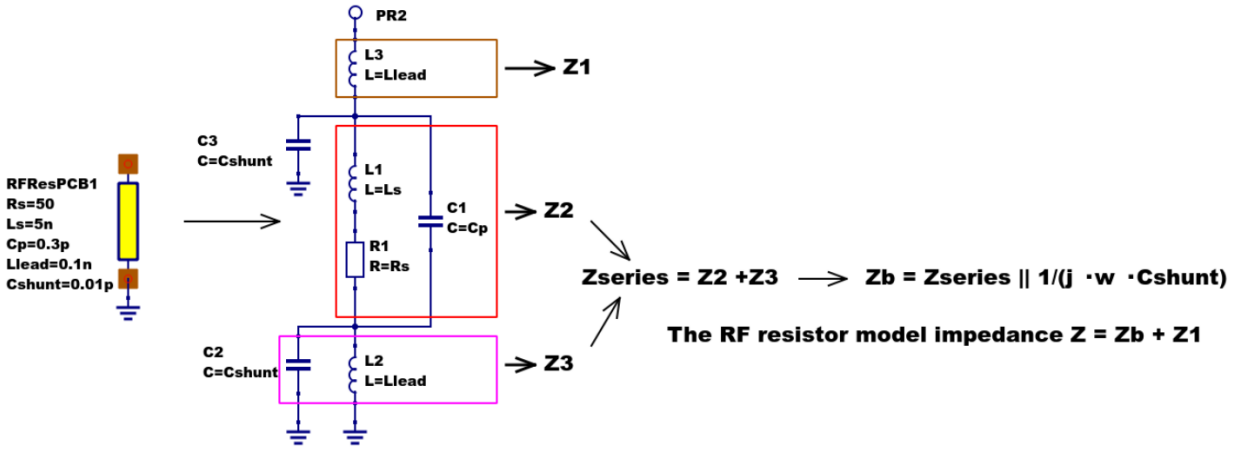


Figure 4 - RF resistor model rotated through 90 degrees and connected with one terminal grounded, similar to the test circuit in Figure. Sections of the model are shown grouped for calculation of the model impedance Z.

Figure 5 illustrates how a set of theoretical equations can be converted into Qucs equations for model simulation and post simulation data processing. In this example Qucs equation **Eqn1** holds values for RF resistor model parameters and Qucs equation **Eqn2** lists the model equations introduced at the start of this section. Figure 5 also gives a set of cartesian graphs of post simulation output data which illustrate how **ZR** and **ZI**, and other calculated items, vary with frequency over the range 1 MHz to 1.3 GHz.

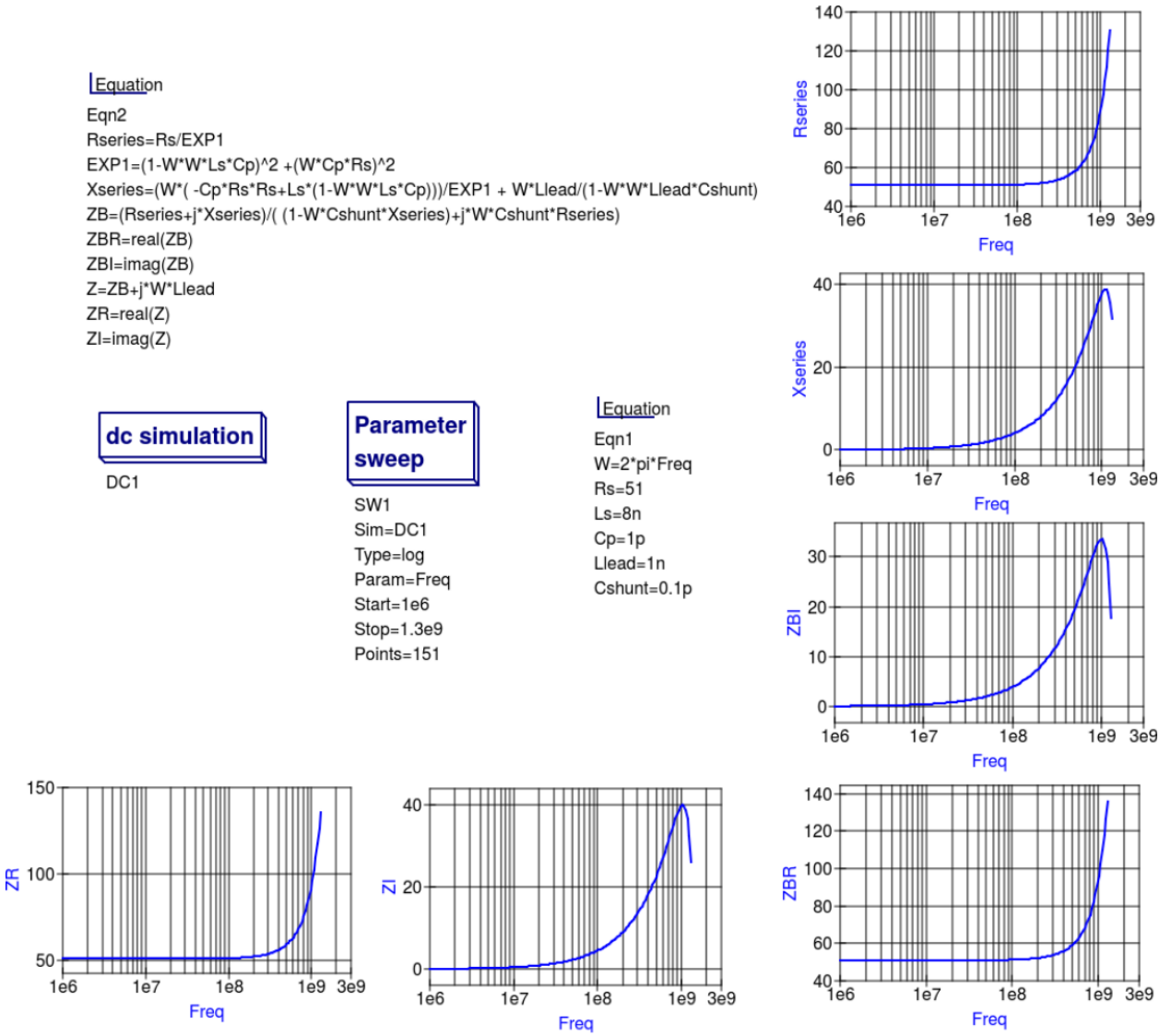


Figure 5- Theoretical analysis of RF resistance impedance Z using Qucs post processing facilities: note a dummy simulation icon, in this example DC simulation, is required to force Qucs to complete the analysis calculations.

13.4 Direct measurement of RF resistor impedance using a simulated impedance meter

A simple impedance meter for measuring the real and imaginary components of component and circuit impedance, using small signal AC simulation, is shown in Figure 6. The impedance measuring technique uses a 1 Amp AC constant current source applied to one terminal of a two port electrical network. The second terminal is grounded. A parallel high resistance resistor (1E9 Ω in Figure 6) shunts the network under measurement to ensure that there is always a direct current path to ground as required by the Qucs simulator during the calculation of simulation results. If required the 1 Amp AC source can be set at a lower value. In such cases the value of **VRes** must also be scaled to give the network impedance.

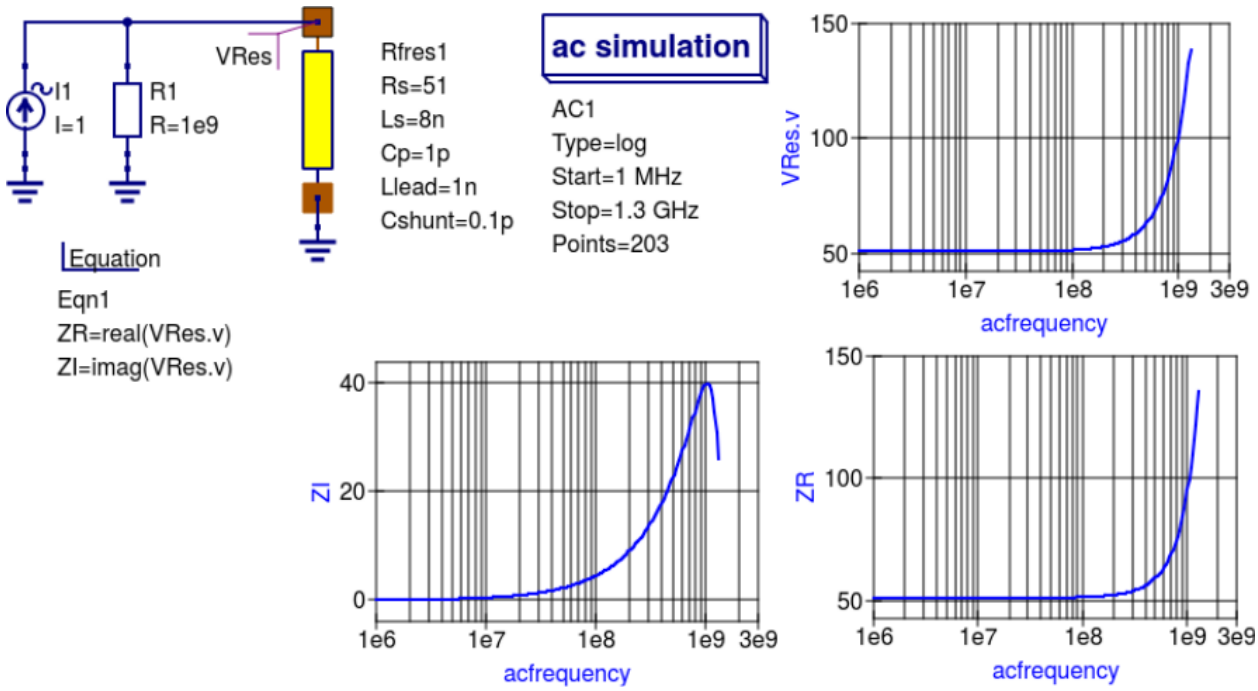


Figure 6 -A simple Qucs test circuit for demonstrating the use of an AC constant current source to measure electrical network impedance.

13.5 Extraction of RF resistance data from measured S parameters

In the past the cost of Vector Network Analyser systems for measuring S parameters has been prohibitively expensive for individual engineers to purchase. However, this scene is changing with the introduction of low cost systems like the DGSAQ Vector Network Analyser (VNWA)[1] . This instrument operates over a frequency band width of 1.3 GHz, providing a range of useful functions with highest accuracy at frequencies up to 500 MHz. This form of VNWA is particularly suited to Radio Amateur requirements and Qucs users interested in RF circuit analysis and design. Such equipment is ideal for measuring RF circuit S parameters and providing measured data for subcircuit and Verilog-A compact devicemodel parameter extraction. Shown in Figure 7 is a graph of measured S parameter data for a nominal 47 Ω resistor[2] . As well as displaying, and printing, measured data the DGSAQ Vector Network Analyser software can output data tabulated in Touchstone"SnP"[3] file format. These files can be read by Qucs and their contents attached to an S parameter file icon for inclusion in circuit schematic diagrams. Figure 8 shows this process as part of an RF resistor model parameter extraction technique involving DGSAQ VNWA measured S parameter data and Qucs simulated S parameter data.

The brown "Test circuits" box shows test circuits for firstly reading and processing the DGSAQ VNWA measured data listed in file mike3.s1p, and for secondly generating simulated S parameter data for an RF resistor specified by parameters **Ls =L**, **Cp = C**, **Llead = LL**, **Cshunt = 0.08 pF**, and **Rs = 47.3** Ω. Presented in Figure 9 are the Qucs Optimization controls" which are used to set the range of** L**, **C** and **LL** values that optimizer ASCO will select from to obtain the best fit between the measured and simulated S parameter data. Note in this parameter extraction system that **S[1,1]** refers to measured S parameter data and **S[2,2]** to simulated S parameter data. Two least squares cost functions called **CF1** and **CF2** are used as targets in the minimisation process. Values for **CF1** and **CF2** can be found in the red box called "Simulation

[1] DG8SAQ VNWA 3 & 3E- Vector Network Analysers, SDR Kits Limited, Grangeside Business Centre, 129 Devizes Road, Trowbridge, Wilts, BA14-7sZ, United Kingdom, 2014.

[2] See DG8SAQ VNWA 3 & 3E- Vector Network Analysers- Getting Started Manual for Windows 7, Vista and Windows XP.

[3] (http://www.vhdl.org/ibis/connector/touchstone_spec11.pdf).

Controls‘‘. In this parameter extraction example the least squares cost function **CF1** is employed to minimize the square of the difference between the real values of the S parameters and least squares cost function **CF2** is employed to minimize the square of the difference between the imaginary values of the S parameters. Qucs post-simulation processing is also used to extract values for the real and imaginary components of the RF resistor impedance. Both the S parameter data and the impedance data are displayed as graphs in Figure 8.

Notice in this example the SPICE optimizer ASCO is used to find the values of **L**, **C** and **LL** which minimize **CF1** and **CF2**. Also note that **Rs** and **Cshunt** are held at fixed values during optimization. In the case of *Rs* its nominal value can be found from DC or low frequency AC measurements. Similarly the value selected for *Cshunt* has been chosen to give a very small but representative value of the parasitic shunt capacitance.. After optimization finishes the minimized values of **L**, **C** and **LL** are given in the initial value column of the Qucs optimization Variables list, see Figure 9. For the 47 Ω resistor the post-minimization RF resistor model parameters are **Rs = 47.3** Ω, **Ls = 10.43 nH**, **Cp = 0.69 pF**, **Llead = 1.46 nH** and **Cshunt = 0.08 pF**. The theoretical simulation data illustrated in Figure 10 shows good agreement with the measured and the optimized simulation data.

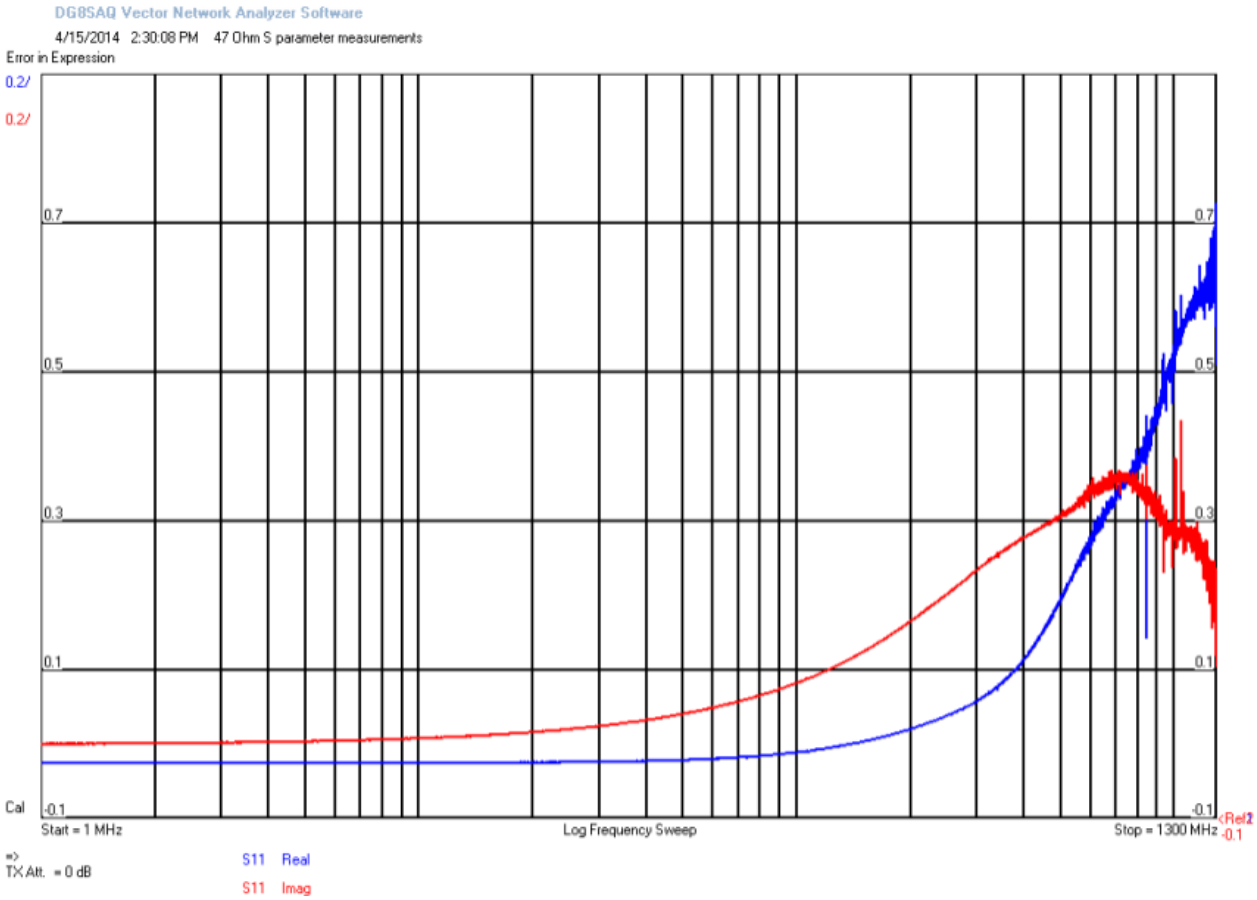


Figure 7 - DGSAQ Vector Network Analyser S parameter measurements for a 47 Ω axial RF resistor.

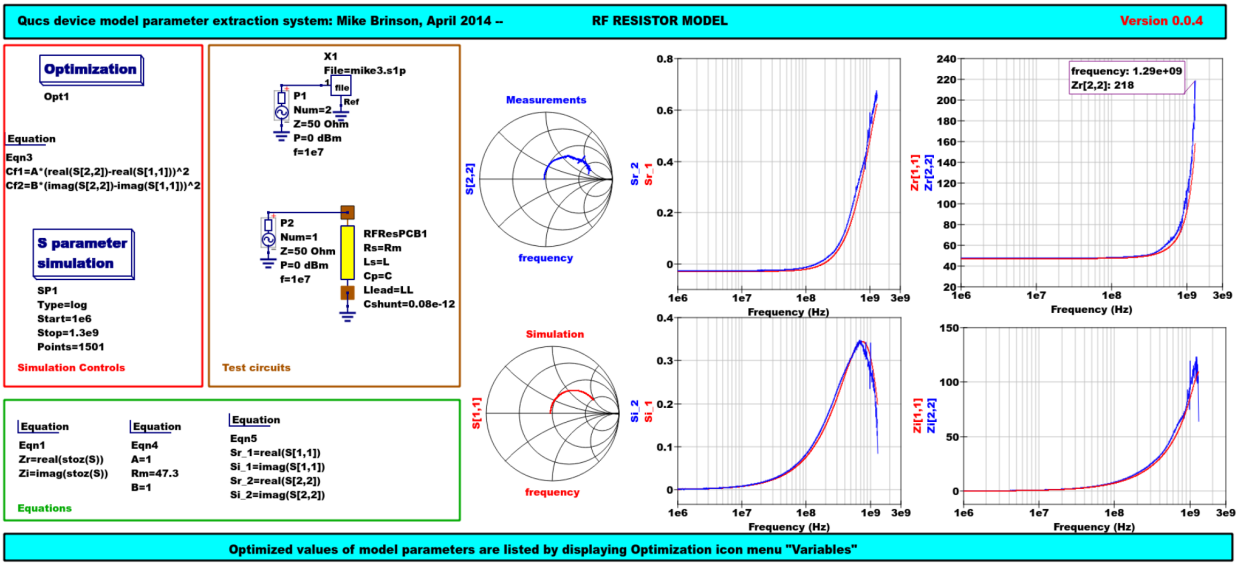


Figure 8 - Qucs device model parameter extraction system applied to a nominal 47 Ω resistor represented by the subcircuit model illustrated in Figure 2 (c). Fixed model parameter values: **Rs = Rm = 47.3** Ω, **CShunt = 0.08pF**; Optimised values: **Ls = L = 10.43nH**, **Llead = LL = 1.47nH**, **Cp = C = 0.69pF**. To reduce simulation time the ASCO cost variance was set to 1e-3. The ASCO method was set to DE/best/1/exp.

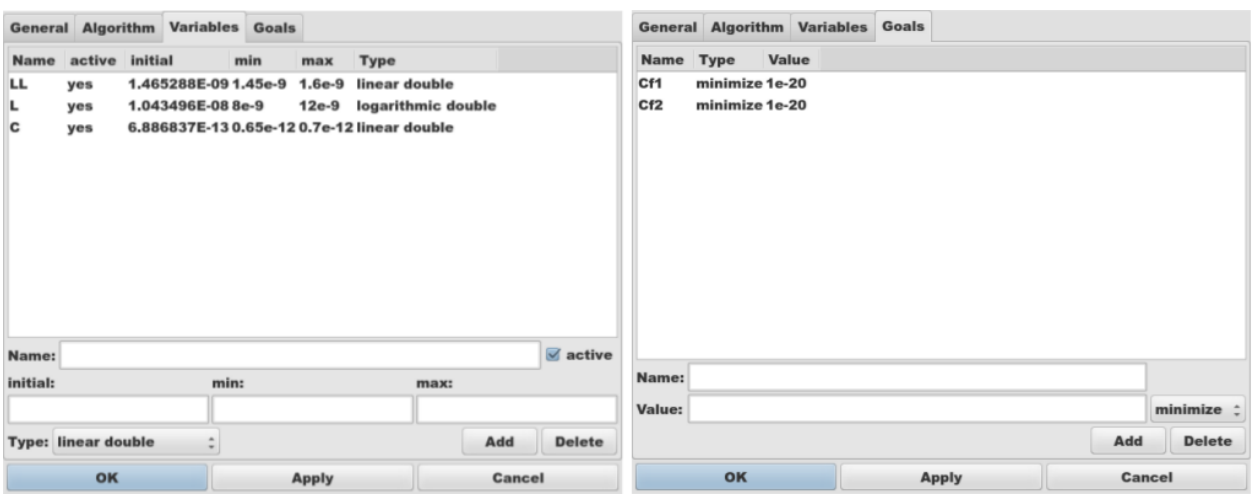


Figure 9 - Qucs Minimization Icon drop down menus: left ”Variables“ and right ”Goals‘‘.

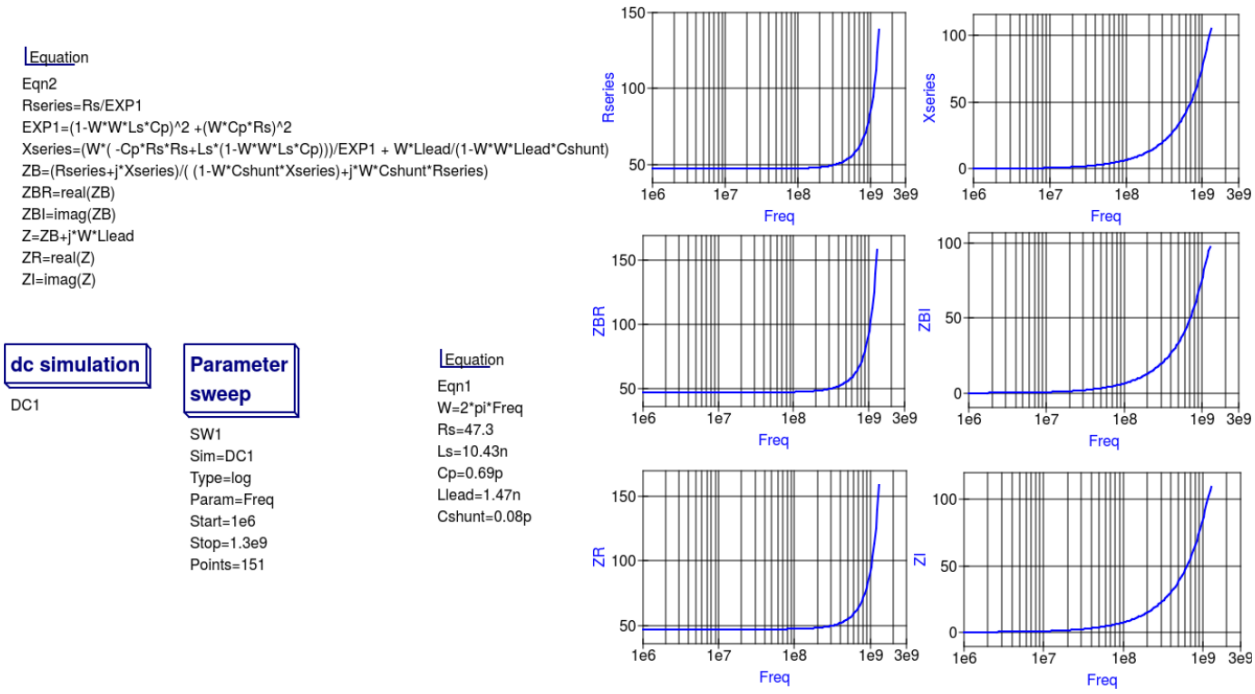


Figure 10 - Qucs simulation of nominal 47 Ω resistor based on theoretical analysis.

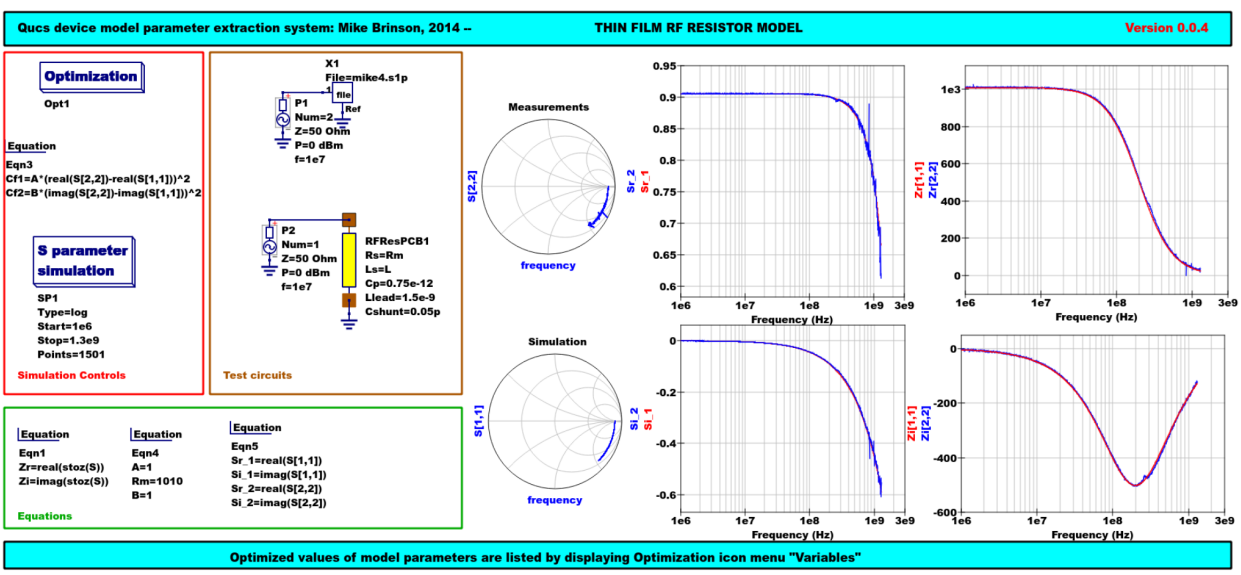


Figure 11 - Qucs device model parameter extraction system applied to a nominal 1000 Ω resistor represented by the subcircuit model illustrated in Figure 2(c).

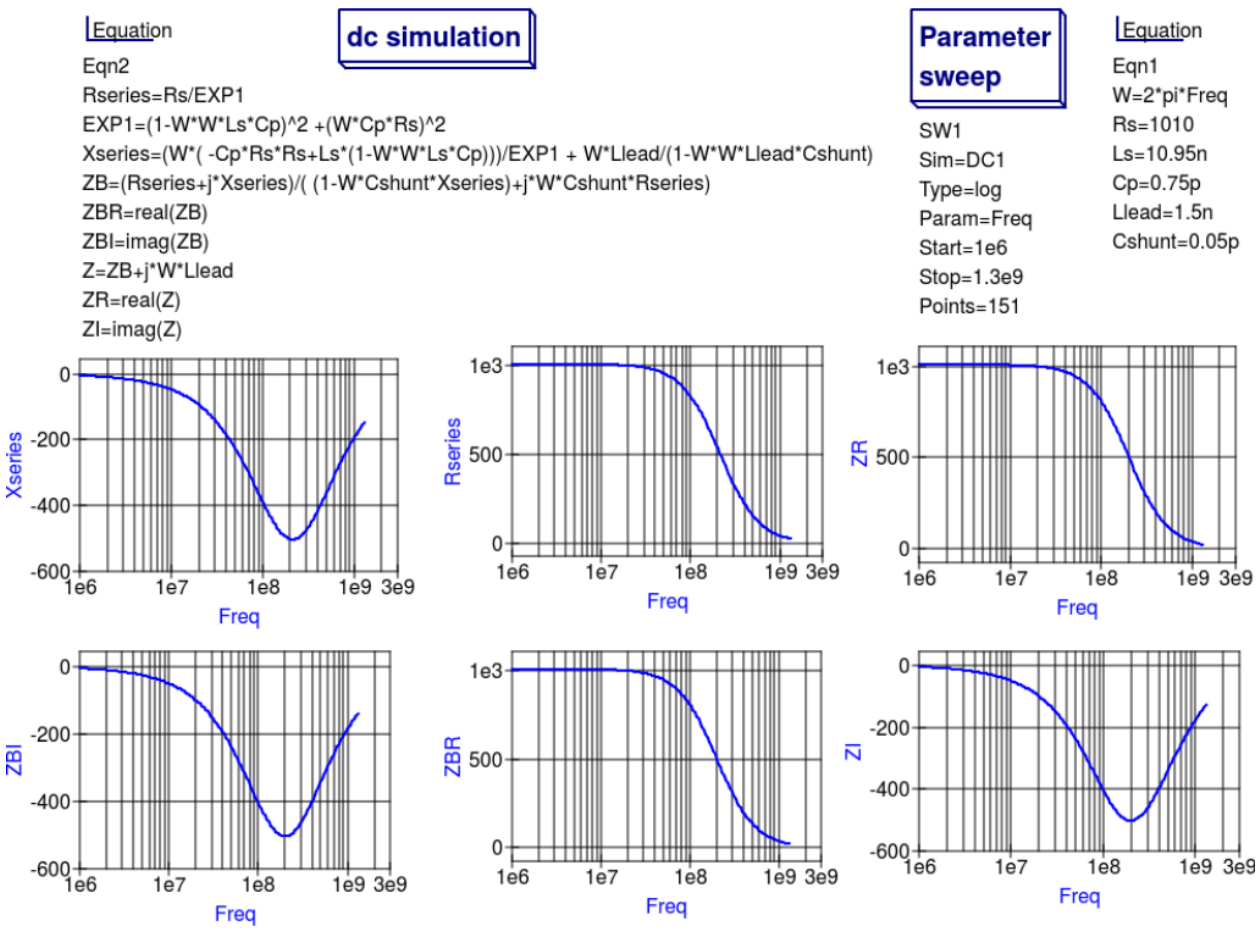


Figure 12 - Qucs simulation of nominal 1000 Ω resistor based on theoretical analysis.

13.6 Extraction of RF resistor parameters from measured S data for a nominal 1000 Ω axial resistor

At low resistance values the impedance of an RF resistor becomes inductive as the signal frequency is increased. This is due to the fact that the inductance **Ls** contribution dominates any reactance effects by **Cp**, **Llead** and **Cshunt**. However, as Rs is increased above a few hundred Ohm's the reverse becomes true with reactive effects dominated by contributions from **Cp**. Figures 11 and 12 demonstrate the dominance of **Cp** reactive effects at low to mid-range frequencies.

13.7 One more example: extraction of RF resistor parameters fro measured S data for a nominal 100 Ω SMD resistor

Figure 13 is included in this Qucs note purely for comparison purposes. SMD resistors are in general physically very small when compared to axial resistors. This results in lower values for the inductive and capacative parasitics which in turn ensures that the high frequency performance of SMD resistors is much improved.

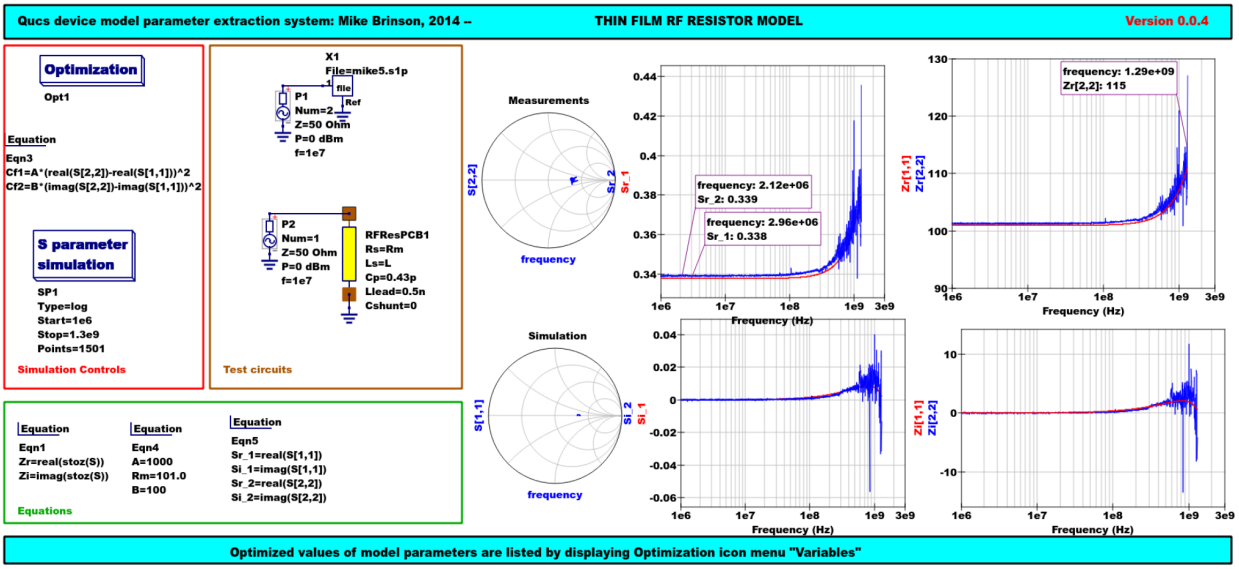


Figure 13 - Qucs device model parameter extraction system applied to a nominal 100 Ω SMD resistor represented by the subcircuit model illustrated in Figure 2 (c).

13.8 A Verilog-A RF resistor model

Listed below is an example Verilog-A code model for the RF resistor model introduced in Figure 2 (c). Due to the limitations of the Verilog-A language subset provided by version 2.3.4 of the "Analogue Device Model Synthesizer" (ADMS)[4] inductors **Ls** and **Llead** are modelled by gyrators and capacitors with values identical to **Ls** or **Llead**.

```
//  Verilog-A module statement.
//
// RFresPCB.va  RF resistor (Thin film resistor, axial type, PCB mounting)
//
//   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, April 2014.
//
`include "disciplines.vams"
`include "constants.vams"
//  Verilog-A module statement.
module RFresPCB(RT1, RT2);
inout RT1, RT2;              // Module external interface nodes.
electrical RT1, RT2;
electrical n1, n2, n3, nx, ny, nz;          // Internal nodes.
`define attr(txt) (*txt*)
parameter real Rs = 50         from [1e-20 : inf)
 `attr(info="RF resistance" unit="Ohm's");
parameter real Cp = 0.3e-12    from [0 : inf)
 `attr(info="Resistor shunt capacitance" unit="F");
parameter real Ls = 8.5e-9     from [1e-20 : inf)
 `attr(info="Series induuctance" unit="H");
```

[4] (http://sourceforge.net/projects/mot-adms/).

```
parameter real Llead = 0.1e-9  from [1e-20 : inf)
 `attr(info="Parasitic lead induuctance" unit="H");
parameter real Cshunt = 1e-10  from [1e-20 : inf)
 `attr(info="Parasitic shunt capacitance" unit="F");
parameter real Tc1 = 0.0       from [-100  : 100]
 `attr(info="First order temperature coefficient" unit ="Ohm/Celsius");
parameter real Tc2 = 0.0       from [-100  : 100]
 `attr(info="Second order temperature coefficient" unit ="(Ohm/Celsius)^2");
parameter real Tnom  = 26.85   from [-273.15 : 300]
 `attr(info="Parameter extraction temperature" unit="Celsius");
parameter real Temp  = 26.85   from [-273.15 : 300]
 `attr(info="Simulation temperature" unit="Celsius");
branch (RT1, n1)  bRT1n1;    // Branch statements
branch (n1, n2)   bn1n2;
branch (n1, n3)   bn1n3;
branch (n2, n3)   bn2n3;
branch (n3, RT2)  bn3RT2;
real Rst, FourKT, n, Tdiff, Rn;
analog begin  // Start of analog code
@(initial_model)
 begin
   Tdiff = Temp-Tnom;   FourKT =4.0*`P_K*Temp;
   Rst = Rs*(1.0+Tc1*Tdiff+Tc2*Tdiff*Tdiff);  Rn     = FourKT/Rst;
 end
 I(n1)     <+ ddt(Cshunt*V(n1));   I(bn1n2)  <+ V(bn1n2)/Rst;
 I(bn1n3)  <+ ddt(Cp*V(bn1n3));    I(n3)     <+ ddt(Cshunt*V(n3));
 I(bRT1n1)  <+ -V(nx); I(nx)      <+ V(bRT1n1);   // Llead
 I(nx)      <+ ddt(Llead*V(nx));
 I(bn2n3)   <+ -V(ny);  I(ny)      <+ V(bn2n3);  // Ls
 I(ny)      <+ ddt(Ls*V(ny));
 I(bn3RT2)  <+ -V(nz);  I(nz)      <+ V(bn3RT2);   // Llead
 I(nz)      <+ ddt(Llead*V(nz));
 I(bn1n2)    <+ white_noise(Rn, "thermal");  // Noise contribution
end  // End of analog code
endmodule
```

Module RFresPCB

Input Variables

Input Variables: instance=0 (bold) and model=9

name	description	default
Rs	RF resistance	50
Cp	Resistor shunt capacitance	0.3e-12
Ls	Series induuctance	8.5e-9
Llead	Parasitic lead induuctance	0.1e-9
Cshunt	Parasitic shunt capacitance	1e-10
Tc1	First order temperature coefficient	0.0
Tc2	Second order temperature coefficient	0.0
Tnom	Parameter extraction temperature	26.85
Temp	Simulation temperature	26.85

Output Variables

Output Variables: instance=0 (bold) and model=0 (red-underlined: temperature dependent)

name	description	dependencies

Nature/Discipline Definition

Nature

name	access	abstol	units
Current	I	1e-12	A
Charge	Q	1e-14	coul
Voltage	V	1e-6	V
Flux	Phi	1e-9	Wb
Magneto_Motive_Force	MMF	1e-12	A*turn
Temperature	Temp	1e-4	K
Power	Pwr	1e-9	W
Position	Pos	1e-6	m
Velocity	Vel	1e-6	m/s
Acceleration	Acc	1e-6	m/s^2
Impulse	Imp	1e-6	m/s^3
Force	F	1e-6	N
Angle	Theta	1e-6	rads
Angular_Velocity	Omega	1e-6	rads/s
Angular_Acceleration	Alpha	1e-6	rads/s^2
Angular_Force	Tau	1e-6	N*m

Discipline

name	potential	flow
logic		
electrical	Voltage	Current
voltage	Voltage	
current	Current	
magnetic	Magneto_Motive_Force	Flux
thermal	Temperature	Power
kinematic	Position	Force
kinematic_v	Velocity	Force
rotational	Angle	Angular_Force
rotational_omega	Angular_Velocity	Angular_Force

Model Equations

Notations used:

- *green: input parameter*
- *bar over: variable never used*
- *bar under: temperature dependent variable*
- *red: voltage dependent variable*

Initial Model

$\mathrm{Tdiff} = (\mathrm{Temp} - \mathrm{Tnom});$

$\mathrm{FourKT} = ((4.0 \cdot 1.3806503\mathrm{e}\text{-}23) \cdot \mathrm{Temp});$

$\mathrm{Rst} = (\mathrm{Rs} \cdot ((1.0 + (\mathrm{Tc1} \cdot \mathrm{Tdiff})) + ((\mathrm{Tc2} \cdot \mathrm{Tdiff}) \cdot \mathrm{Tdiff})));$

$\mathrm{Rn} = \frac{\mathrm{FourKT}}{\mathrm{Rst}};$

---------- end of Initial Model

$I(\mathrm{n1}, \mathrm{n1}) <+ \mathrm{ddt}((\mathrm{Cshunt} \cdot V(\mathrm{n1}, \mathrm{n1})));$

$I(\mathrm{n1}, \mathrm{n1}) <+ \frac{V(\mathrm{n1}, \mathrm{n1})}{\mathrm{Rst}};$

$I(\mathrm{n1}, \mathrm{n1}) <+ \mathrm{ddt}((\mathrm{Cp} \cdot V(\mathrm{n1}, \mathrm{n1})));$

$I(\mathrm{n3}, \mathrm{n3}) <+ \mathrm{ddt}((\mathrm{Cshunt} \cdot V(\mathrm{n3}, \mathrm{n3})));$

$I(\mathrm{RT1}, \mathrm{RT1}) <+ (-V(\mathrm{nx}, \mathrm{nx}));$

$I(\mathrm{nx}, \mathrm{nx}) <+ V(\mathrm{RT1}, \mathrm{RT1});$

$I(\mathrm{nx}, \mathrm{nx}) <+ \mathrm{ddt}((\mathrm{Llead} \cdot V(\mathrm{nx}, \mathrm{nx})));$

$I(\mathrm{n2}, \mathrm{n2}) <+ (-V(\mathrm{ny}, \mathrm{ny}));$

$I(\mathrm{ny}, \mathrm{ny}) <+ V(\mathrm{n2}, \mathrm{n2});$

$I(\mathrm{ny}, \mathrm{ny}) <+ \mathrm{ddt}((\mathrm{Ls} \cdot V(\mathrm{ny}, \mathrm{ny})));$

$I(\mathrm{n3}, \mathrm{n3}) <+ (-V(\mathrm{nz}, \mathrm{nz}));$

$I(\mathrm{nz}, \mathrm{nz}) <+ V(\mathrm{n3}, \mathrm{n3});$

$I(\mathrm{nz}, \mathrm{nz}) <+ \mathrm{ddt}((\mathrm{Llead} \cdot V(\mathrm{nz}, \mathrm{nz})));$

$I(\mathrm{n1}, \mathrm{n1}) <+ \mathrm{white\_noise}(\mathrm{Rn}, \text{"thermal"});$

Figure 14 - Details of the proposed RF resistor model: equations, variables and other data.

13.9 Extraction of Verilog-A RF resistor model parameters from measured S data for a 100 Ω axial resistor

This example demonstrates the use of ASCO for extracting Verilog-A model parameters from measured S parameter data. ASCO optimization yields a figure of 4nH for **L** in the model shown in Figure 2 (c). Other model parameter values are given with the test circuit, see Figure 15.

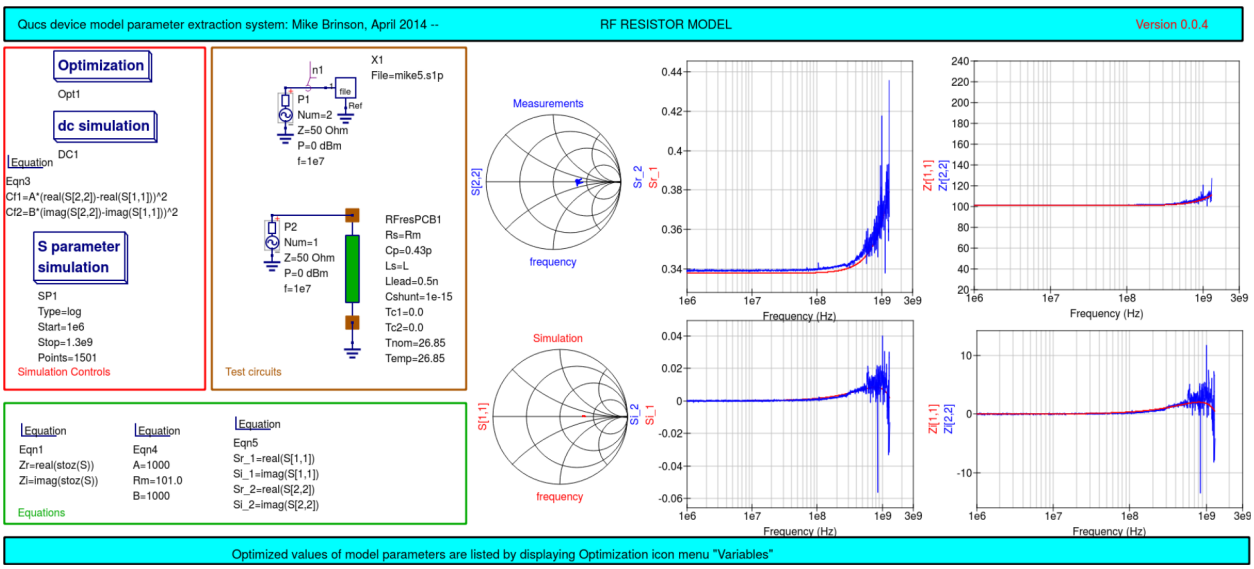


Figure 15 - Verilog-A models parameter data extraction for a 100 Ω axial thin film resistor. Fixed model parameter values: **Rs = Rm =101 Ω**, **CShunt = 1e-15 F**, **Llead = LL = 0.5nH**, **Cp = C = 0.43pF**; Optimised values: **Ls = L = 3.99nH**. To reduce simulation time the ASCO cost variance was set to 1e-3. The ASCO method was set to DE/best/1/exp.

13.10 End Notes

This brief Qucs note outlines the fundamental properties of subicircuit and verilog-A compact component models for RF resistors. The use of optimization for the extraction of subcircuit and Verilog-A compact model parameters from measured S parameters is also demonstrated. The presented techniques form part of the simulation and device modelling capabilities available with the latest Qucs release[5].

Технічні описи, що стосуються симулятора

Є на http://qucs.sourceforge.net/tech/technical.html

Приклади електричних схем

Є на http://qucs.sourceforge.net/download.html#example

[5] Qucs release 0.0.18, or greater.