Qucs Help Documentation

Vydání 0.0.19

Qucs Team (2014)

25.07.2018

Obsah

Contents:

KAPITOLA 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 ©.

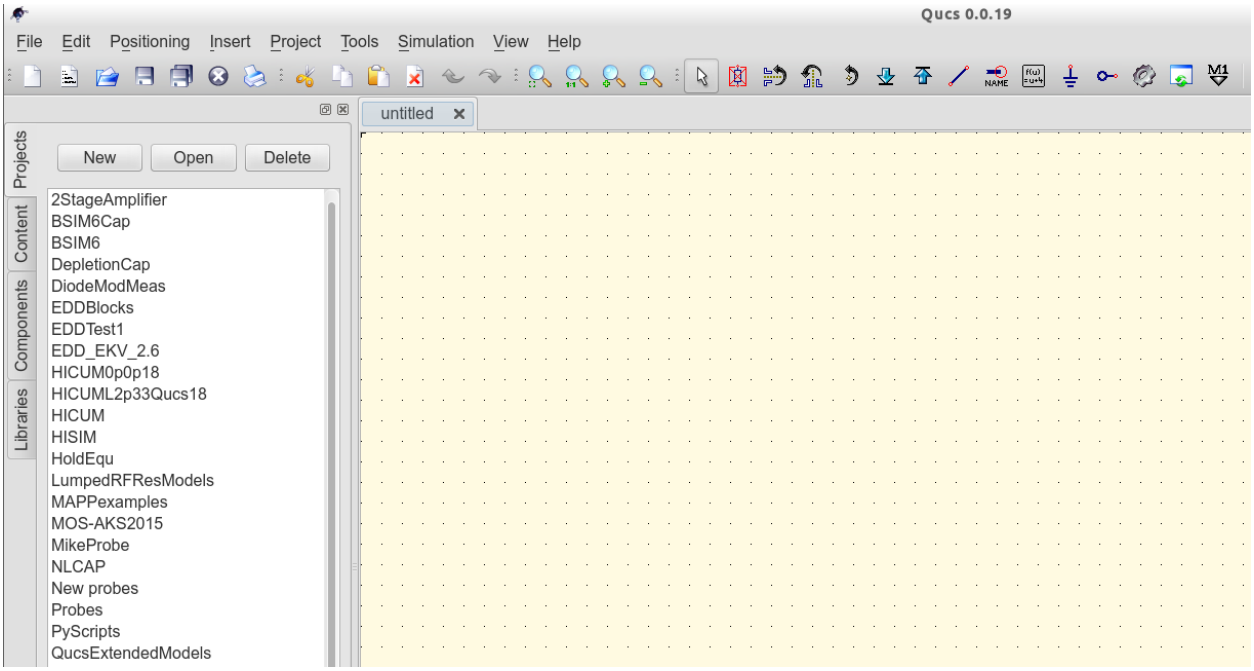
Qucs is a large software package which takes time to learn. Incidentally, this statement is also true for other GPL circuit simulators. New users must realise that to get the best from the software some effort is required on their part. In particular, one of the best ways to become familiar with Qucs 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.

KAPITOLA 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.



Obrázek 1 - Qucs - hlavní okno

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.

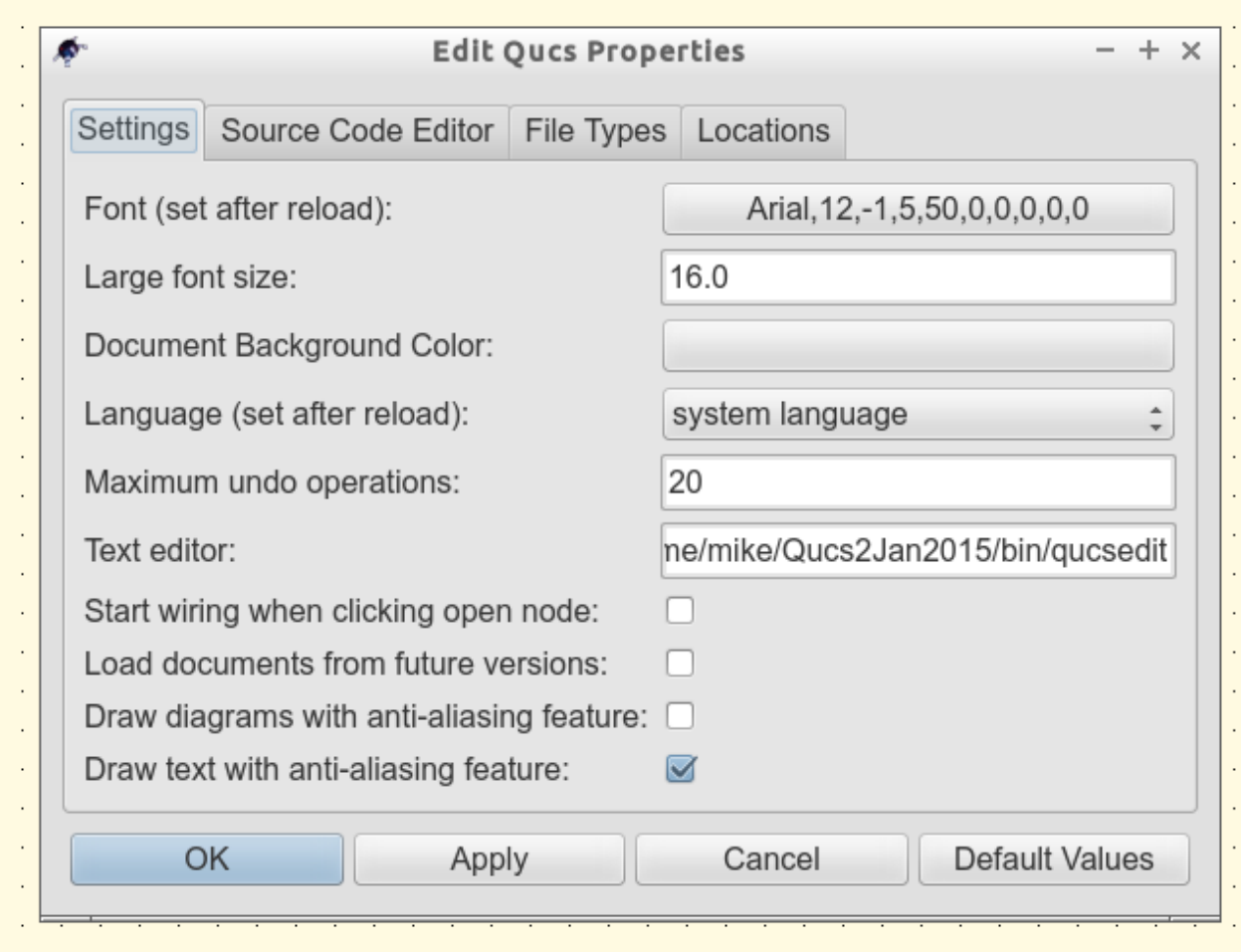


Figure 2 - QucsEditProperties window

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.

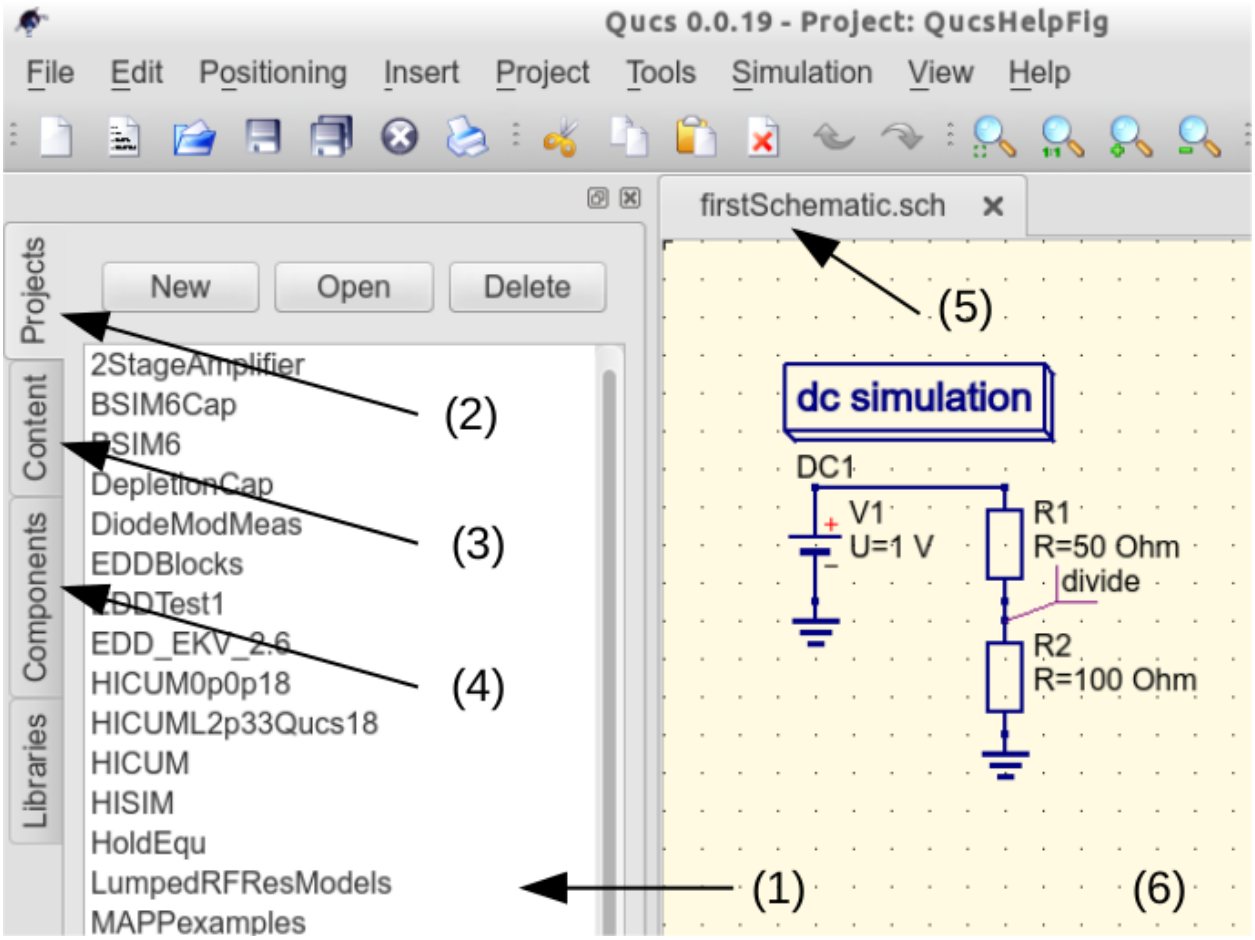


Figure 3 - Qucs main window with working areas labelled

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** $\longrightarrow$ **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.

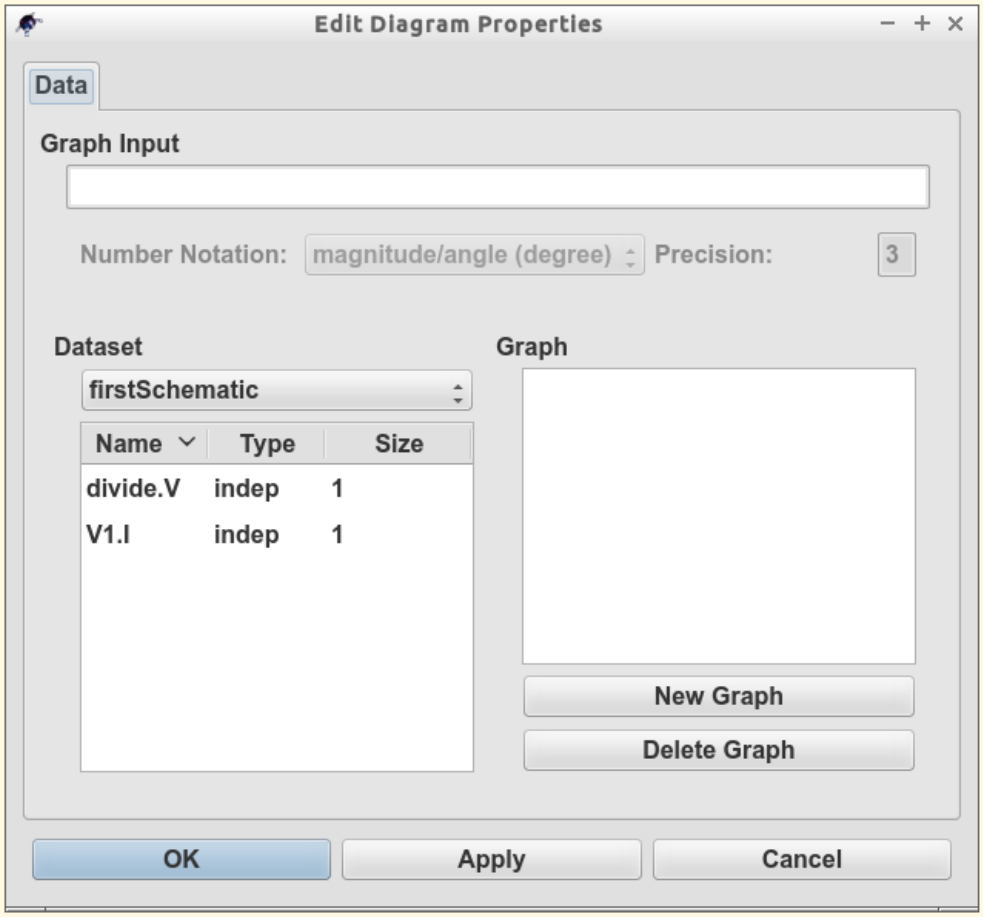


Figure 4 - Qucs data display window showing a **Tabular** dialogue

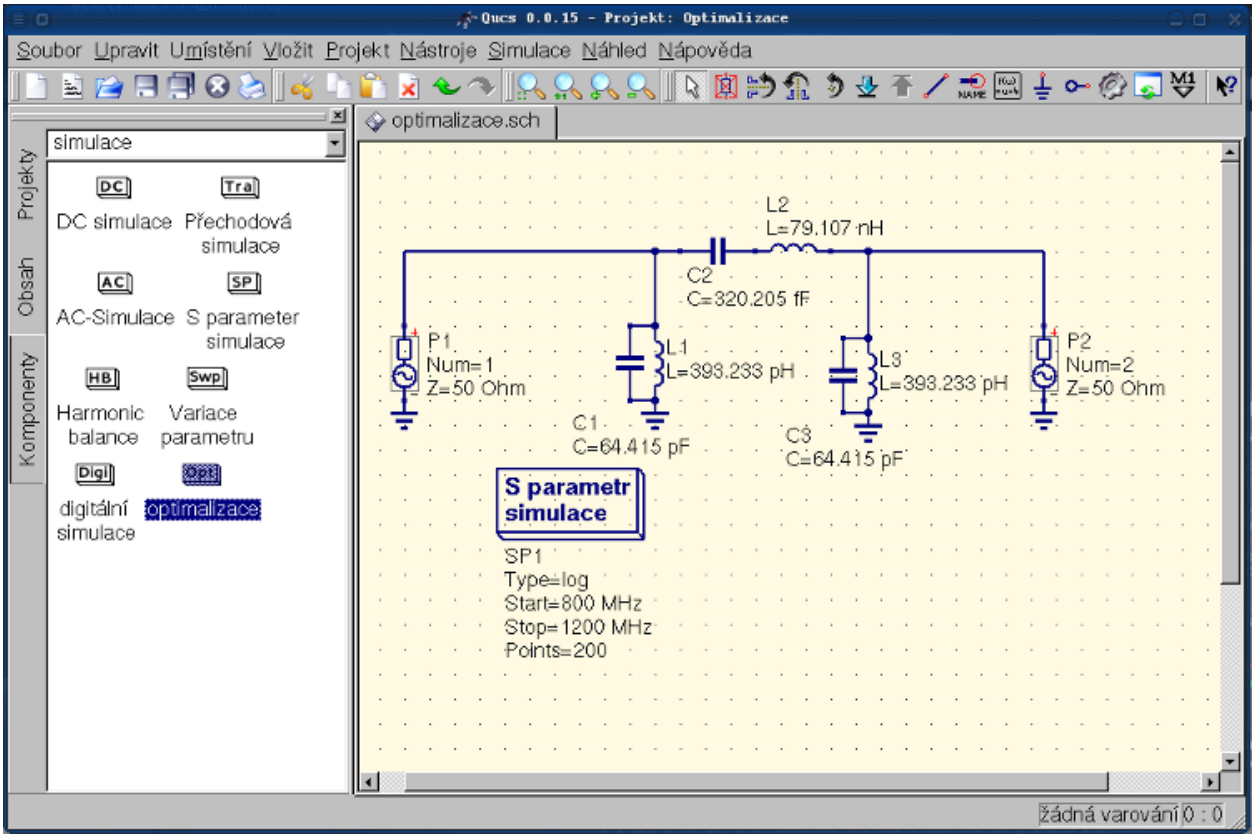
KAPITOLA 3

Začínáme s optimalizací

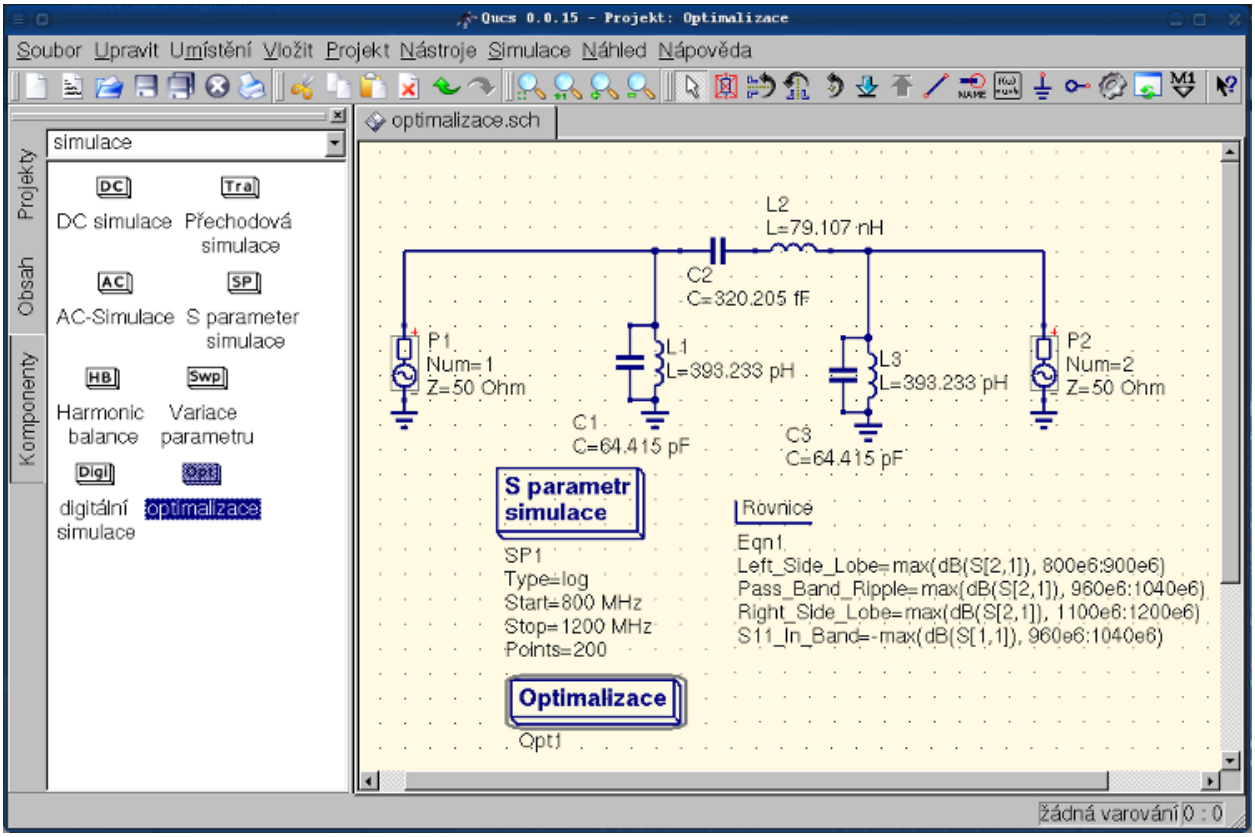
Pro optimalizaci obvodu používá Qucs utilitku ASCO (http://asco.sourceforge.net/). Zde předložím popis jak připravit vaše schéma, jak to celé spustit a jak porozumět výsledkům. Předtím než budete chtít využívat tuto funkci, musíte ASCO nainstalovat na váš počítač.

Obvod s minimalizací není nic víc, než program, který minimalizuje počet funkcí. Může to být každé časové zpoždění, nebo doba náběžné hrany v digitálních obvodech, nebo jakýkoli zdroj v analogových obvodech. Další možností je definovat optimalizační problém jako skládání funkcí, nebo jako v tomto případě, definovat „figure-of-metir“.

Pro nastavení optimalizace musíme dvě věci do schématu přidat: rovnici/rovnice (Vložit -> Vložit rovnici) a velký čtverec s nápisem „Optimalizace“ (Ve skupině „Simulace“). Sestavte schéma podle obrázku 1 a hrajte si s Qucs, dikud vaše schéma nebude vypadat jako na obrázku 2 ;-) . Dejte si pozor na znaménko minus u S11_In_Band!

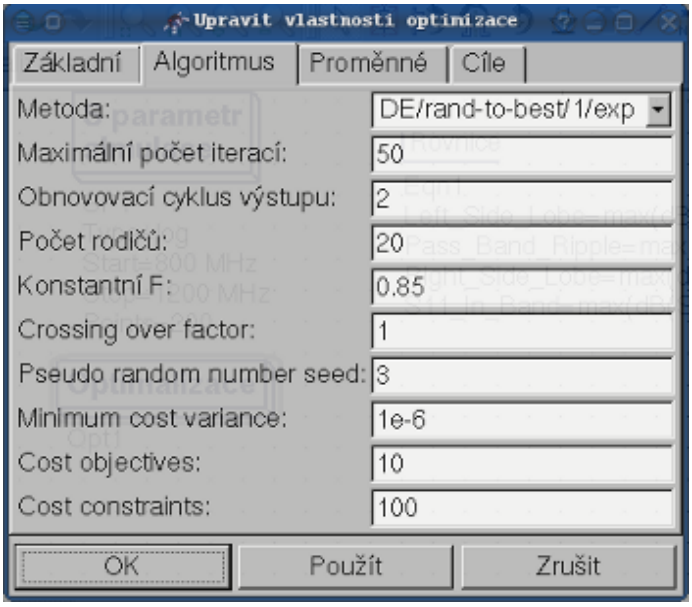


Obrázek 1 - Počáteční schéma



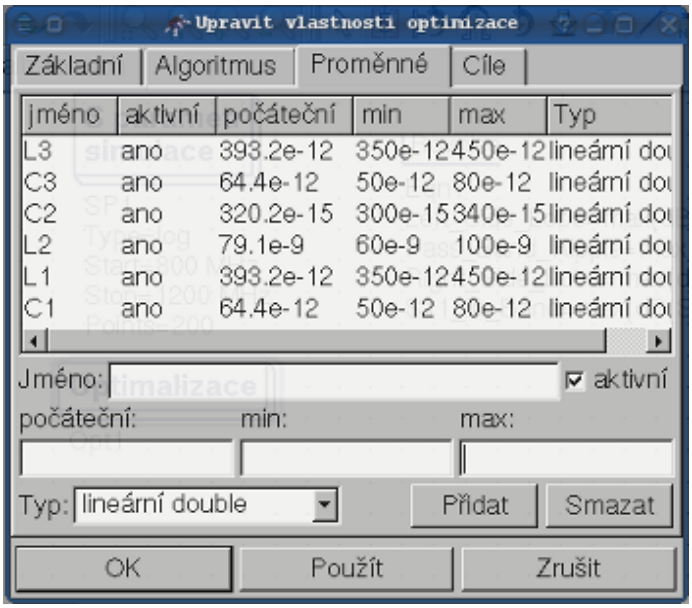
Obrázek 2 - Připravené schéma

Nyní vyberte ze skupin komponent komponentu „Optimalizace“. Z existujících parametrů je třeba věnovat speciální pozornost „Maximální počet opakování“ , „Konstanta F“ a „Přechod přes faktor“. Někdy může optimalizace trvat jen chvilinku, ale někdy i celkem dlouho.



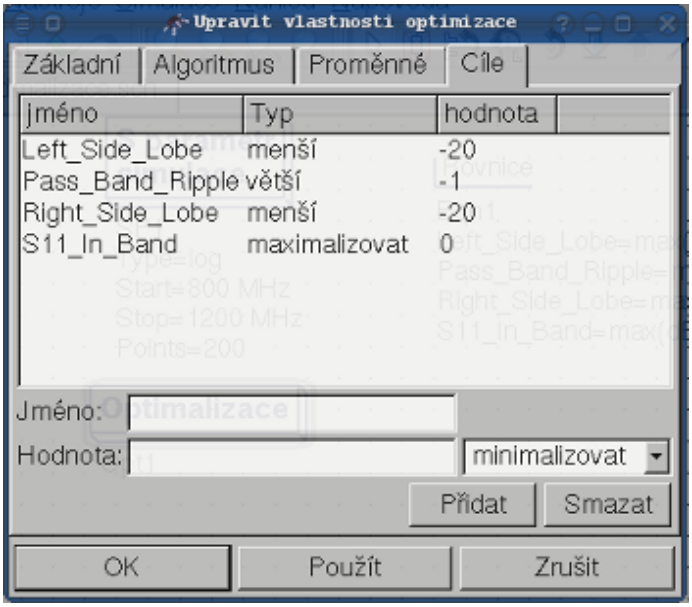
Obrázek 3 - Optimalizace - dialog, možnosti algoritmu.

V tabulce Proměnné, které definují jednotlivé komponenty bude vybráno z rozsahu, jak je zobrazeno na obrázku 4. Jména proměnných jsou ve shodě s jejich identifikátory umístěných ve vlastnostech komponent a **NE** jmen konponent.



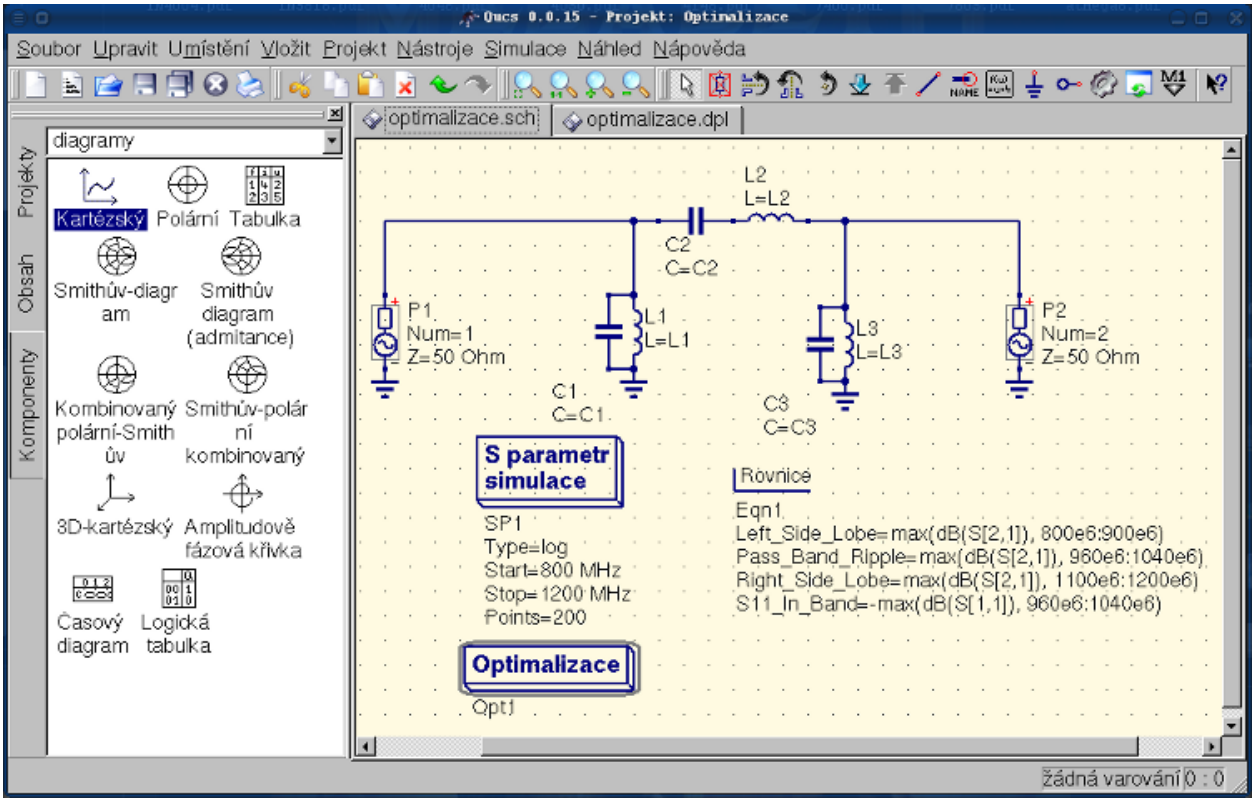
Obrázek 4 - Optimalizace - dialoh, možnosti proměnných.

Konečně, se dostáváme do cíle kde objekty optimalizace (maximalizace, minimalizace) a omezení (menší, větší, rovno) jsou definovány. Poté ASCO je automaticky zkombinuje do jediné funkce a to je minimalizace.



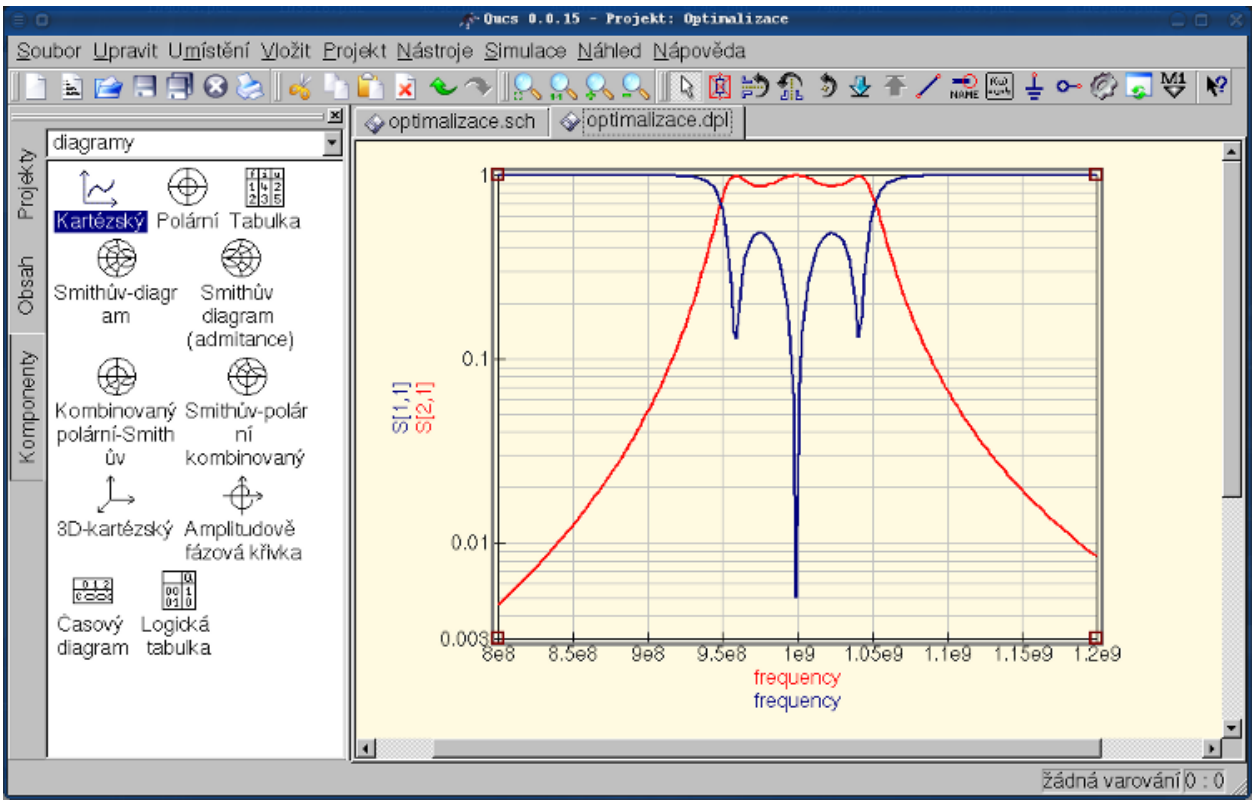
Obrázek 5 - Optimalizace - dialog, cílové možnosti.

Dalším krokem je změna schématu a definice které součásti obvodu budou optimalizovány. Výsledné schéma se zobrazeno na obrázku 6.



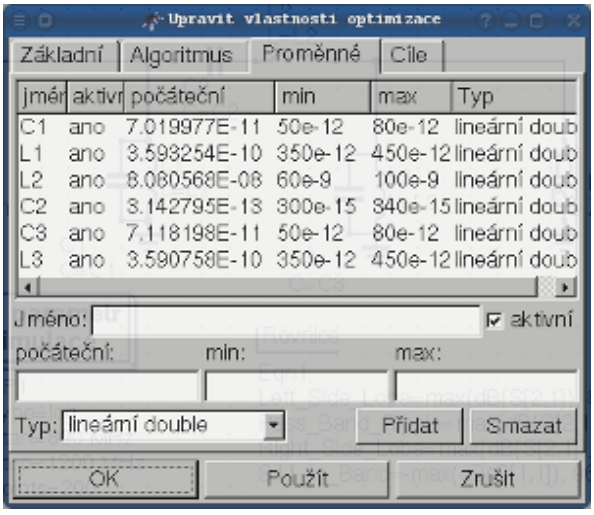
Obrázek 6 - Nové hlavní okno QUCS

Posledním krokem optimalizace je spuštění simulace stisknutím klávesy F2. Simulace může trvat několik sekund na moderním počítači. Nejlepší bude, když si výsledky necháte zobrazit do grafu. Vyberte si diagram kartézský. Přidejte do něj S[2,1] a S[1,1]. Už jen nastavit barvy a tloušťku grafu (čím větší číslo, tím tlustější).



Obrázek 7 - Qucs - okno s výsledky.

Nejlepší nalezené obvody můžete najít v optimalizačním dialogu v záložce Proměnné. Zde jsou hodnoty pro každý z uvedených prvků (Obrázek 8).



Obrázek 8 - Nejlepší nalezené obvody.

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.

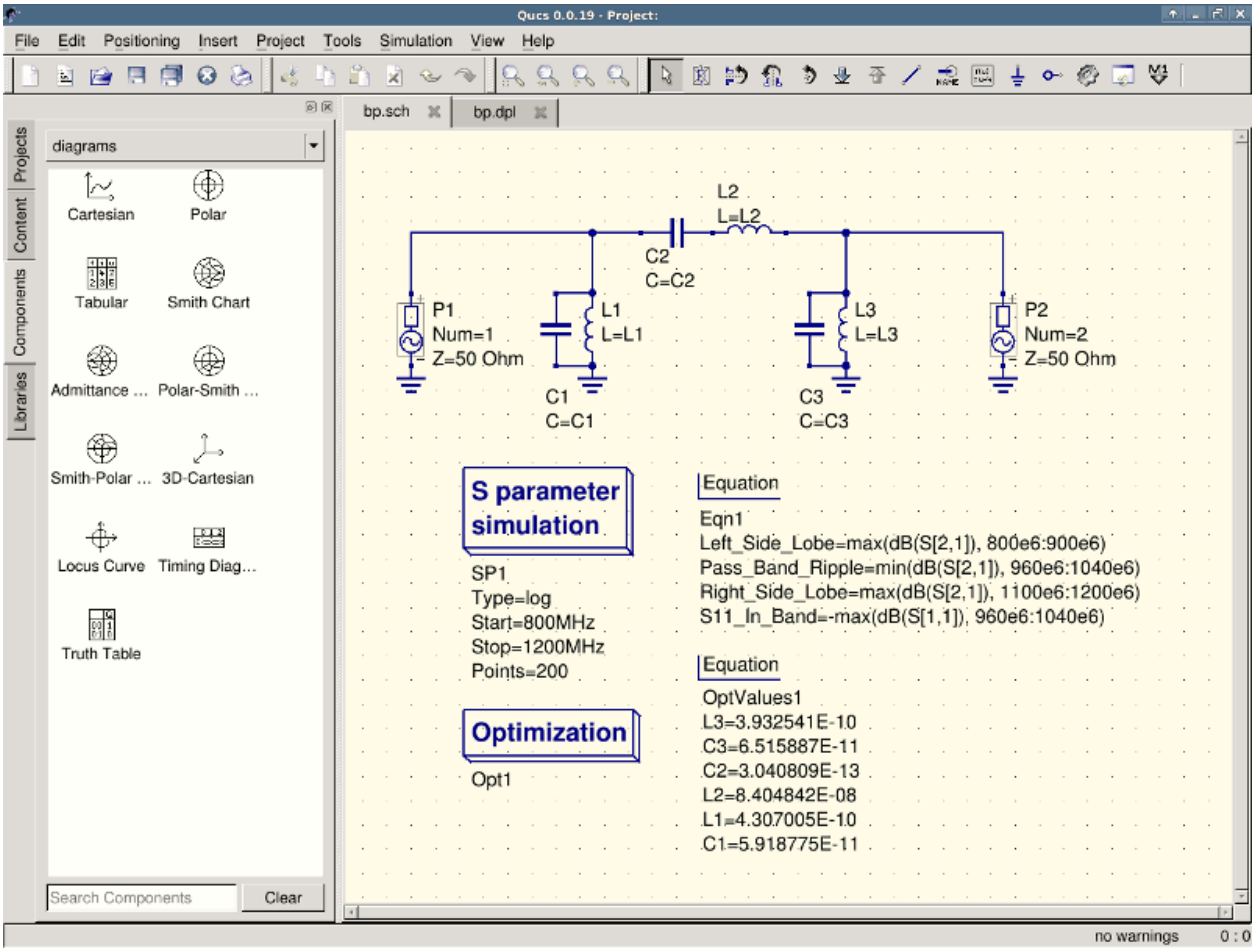


Figure 9 - Schematic with optimized values.

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.

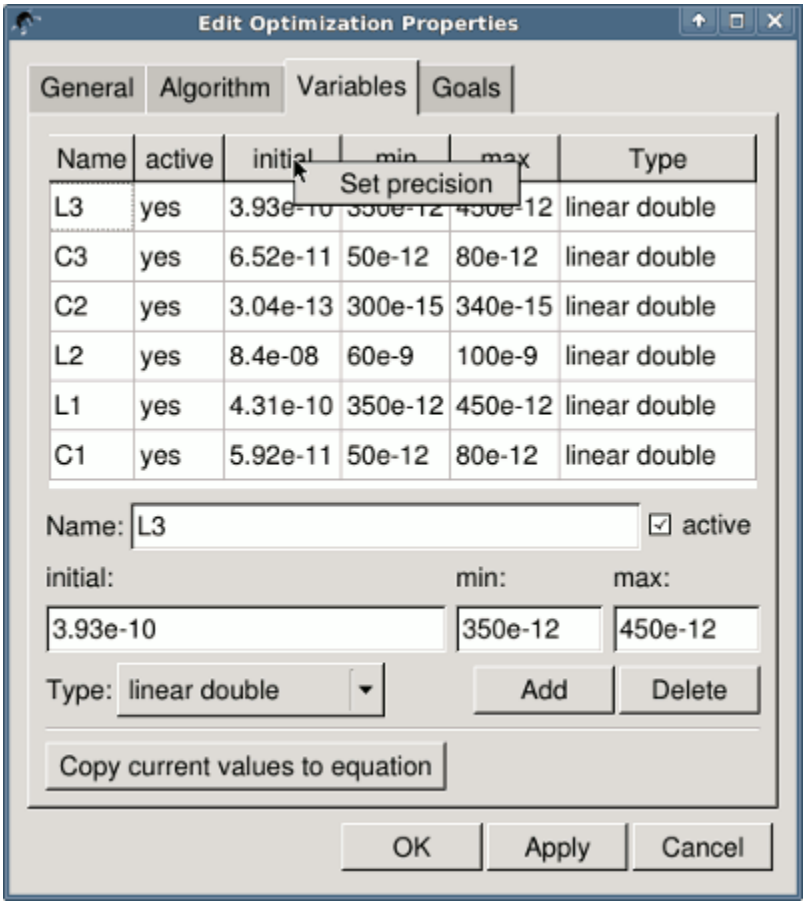


Figure 10 - Changing the displayed variables precision.

KAPITOLA 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.

KAPITOLA 5

Krátký popis k ovládání

5.1 Hlavní ovládání

(platné pro všechny módy)

kolečko myši	Posouvá vertikálně pracovní oblast.
kolečko myši + klávesa Shift	Posunuje horizontálně pracovní oblsat.
kolečko myši + klávesa Ctrl	Přiblíží, nebo oddálí pracovní oblast.
„přetáhnout“ soubor do oblasti s dokumenty (viz Začínáme - Analogové obvody obrázek 1 (5))	Qucs se pokusí otevřít soubor jako schéma, nebo jako soubor s výstupními daty.

5.2 „Výběr“-Mód

(Menu: Upravit->Vybrat)

levé tlačítko myši	Vybere komponentu pod kurzorem. Pokud je zde několik komponent, můžete klikat tak dlouho, dokud nebude vybrána taková komponenta, kterou chcete. Pokud necháte tlačítko myši stisklé, můžete komponenty pod kurzorem přesouvat. Pokud chcete nastavit přesně pozici komponent, držte klávesu CTRL během přesouvání. Tím se vypne automatické přichytávání k mřížce. Pokud budete držet tlačítko myši v prázdném poli, vytvoříte obdélník. Po uvolnění myši všechny komponenty umístěné vevnitř obdélníku budou vybrané. Vybrané schéma, nebo kresba může být zvětšena, nebo zmenšena pomocí stisklého levého tlačítka myši na jednom z jeho rohů a přesunutím kurzoru se stisklým tačítkem myši. Po kliknutí na text u komponenty, může být obsah upravován přímo. Po stisknutí Enteru automaticky skočí na další řádek. Pokud je na dalším řádku možnost si vybrat (například ze 2 možností), můžete tyto možnosti projít pomocí šipky nahoru a šipky dolů. Kliknutím v obvodu na uzel vstoupíte do „propojovacího módu“.
Levé tlačítko myši + klávesa Ctrl	Povolí vybrat více jak jednu komponentu. Například vyberete několik komponent, ale z tohoto výběru potřebujete mít dvě nevybrané. Kliknutím na vybranou komponentu ji odeberete z výběru. Tento mód je také platný pro výběr pomocí obdélníku (podívejte se o odstavec výše).
pravé tlačíko myši	Kliknutím na vodič vyberete pouze číst vodiče namísto celého vodiče.
Dojklik levým tlačítkem myši	Otevře se okno, ve kterém můžeme měnit vlastnost (značky vodičů, parametry komponent, atd.)

5.3 „Vložit komponentu“-Mód

(Kliknout na komponentu/diagram v levé oblasti)

levé tlačítko myši	Umístí novou komponentu do schématu.
pravé tlačíko myši	Otočí komponentu. (Neplatí pro diagramy.)

5.4 „Vodič“-Mód

(Menu: Vložit->Vodič)

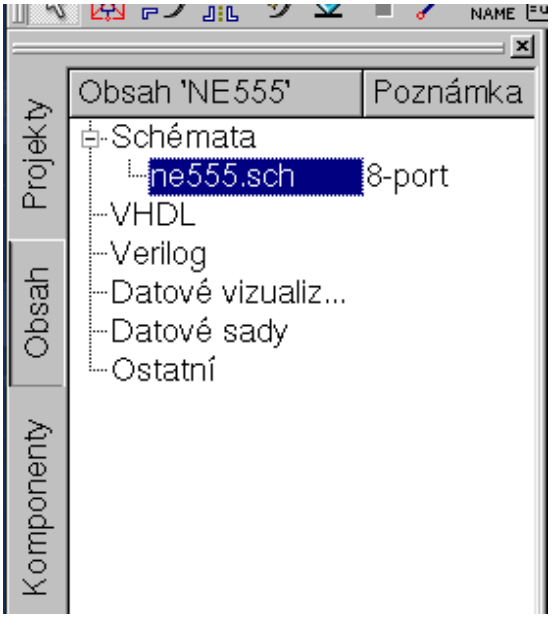
levé tlačítko myši	Nastaví začítek/konec vodiče.
pravé tlačíko myši	Změní směr vodiče v rohách (první doleva/dopravas, nebo první nahoru/dolů).
Dojklik levým tlačítkem myši	Ends a wire without being on a wire or a port.

5.5 „Vložit“-Mód

(Menu: Upravit->Vložit)

levé tlačítko myši	Umístí komponentu do schématu (ze schránky).
pravé tlačíko myši	klik levým tlačítkem

5.6 Myš v záložce „Obsah“



dvojklik	Otevře soubor.
dvojklik	Otevře soubor.
Klávesnice	Zobrazí se menu:

5.7 Klávesnice

Mnoho akcí můžeme aktivovat/potrvrdit pomocí klávesnice. Tyto klávesové zkratky můžete najít v hlavním menu. Další klávesové zkratky najdete v tomto seznamu:

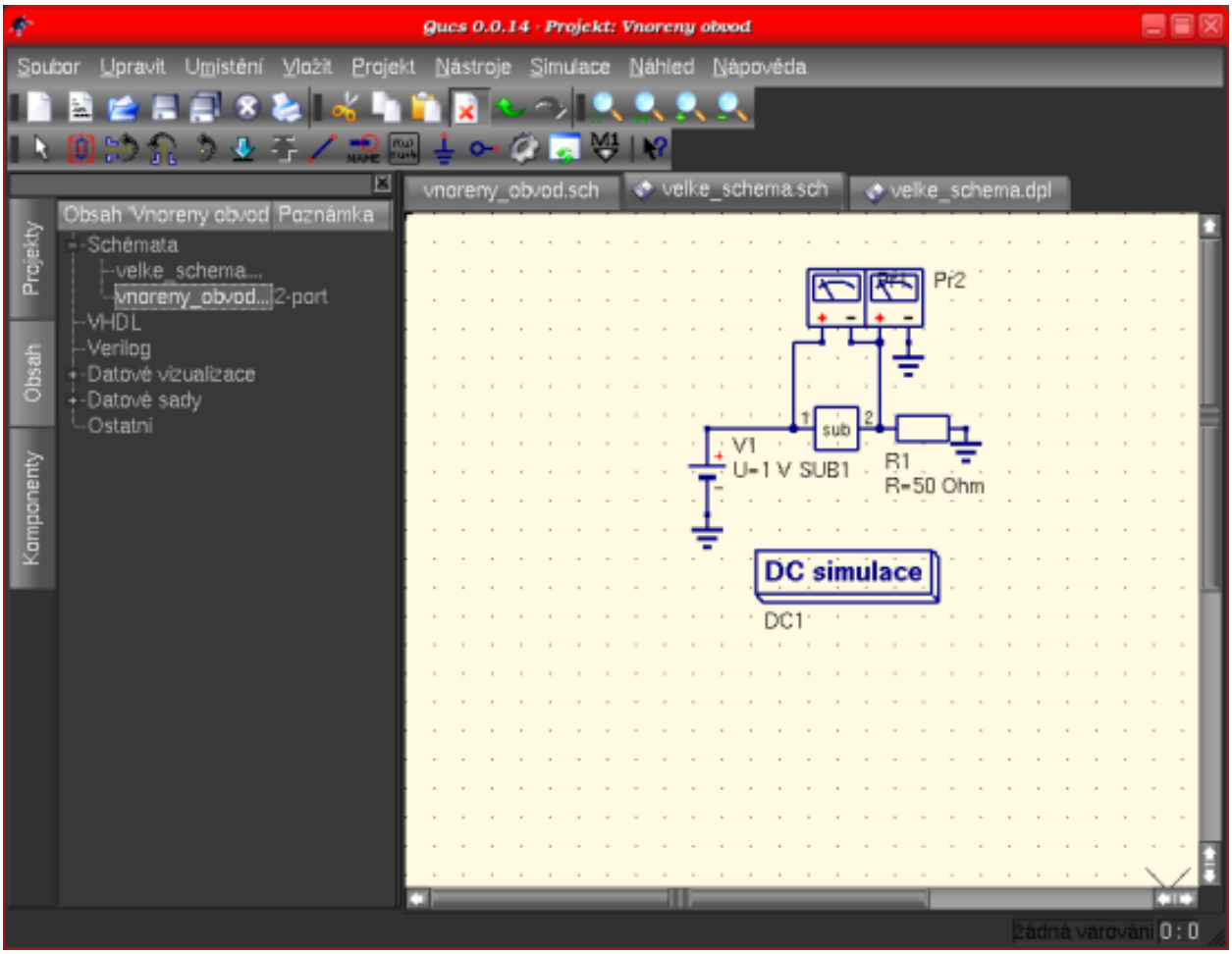
Šipky do-leva/doprava	Smaže vybrané komponenty, nebo se zapne mazací mód, pokud není nic vybráno.
Šipky na-horu/dolů	Změní pozici vybraných popisovačů v jejich grafech. Pokud není žádny popisovač vybrán, posune vybrané objekty. Pokud není žádná komponenta vybrána, bude se rolovat mezi záložkami dokumentů.
Tabulátor	Změní pozici vybraných popisovačů ve více rozměrných grafech.Pokud není žádný popisovač vybrán, po Changes the position of selected markers on more-dimensional graphs.Pokud není žádny popisovač vybrán, posune vybrané objekty. Pokud není žádná komponenta vybrána, bude se rolovat mezi záložkami dokumentů.
Na-horu	Skočí na další otevřený dokument. (souhlasně s další záložkou).

KAPITOLA 6

Práce s vnořenými obvody

Vnořené obvody se používají pro větší přehlednost ve schématu. Toto je velice užitečné při sestavování rozsáhlejších obvodů, kde se mnohokrát objevují stejné bloky součástek.

Nejprve vytvoríme vnořený obvod samotný. Vytváří se stejně jako každé jiné schéma. Nejdříve ale musíme programů říct, kolik bude mít vlastně vnořený obvod pinů (pro začátek stačí třeba 2). Tyto piny najdete v liště nástrojů „Vložit připojení“, nebo v menu Vložit->Vložit připojení. Na ty červené kroužky připojte například dva rezistory 50 Ohmů zapojeny paralelně. Nyní schéma uložte (například vnoreny_obvod.sch). Všiměnte si, že v záložce „Obsah“ je u názvu schématu poznámka „2-port“. To znamená, že se jedná o vnořený obvod se dvěma piny (konektory). Teď už jen zbývá otevřít (popřípadě vytvořit nové) schéma a do něj vložit komponentu „Podobvod“. Klikněte na náš vnořený obvod následně dvakrát klikněte na schéma, kam chcete vložit vnořený obvod. Komponenta „Podobvod“ by se měla sama nabídnout ke vložení. Nyní už zbývá vybrat pro „Podobvod“ vybrat vhodné místo a napojit k němu další součástky. Vzorový příklad můžete vidět na obrázku 1. Nyní můžete obvod odsimulovat. Výsledky jsou naprosto shodné, jako kdyby součástky ze vnořeného obvodu byli připojené přímo.



Obrázek 1 - Obvod s vnořeným obvodem

Pokud vyberete komponentu „Podobvod“ (jednou kliknout na symbol ve schématu) můžete se snadno a rychle dostat do vnořeného obvodu, pokud stisknete klávesu CTRL současně s klávesou I (Samozřejmě, že tato funkce je dostupná z lišty nástrojů i hlavního menu). Vrátit zpět se můžete pomocí CTRL-H (současně stisknout CTRL a klávesu H).

Pokud se vám nelíbí symbol komponenty pro vnořený obvod, můžete si nakreslit svůj vlastní symbol a vložit vlastní popisky komponenty na vaše oblíbené místo. Pouze vytvořte vnořený obvod a v menu kliněte na Soubor->Upravit symbol komponenty. Pokud ještě nemáte nakreslený symbol pro tento obvod, jednoduchý symbol je vytvořen automaticky. Nyní můžete upravovat symbol kreslením čar a vlastně čímkoliv, co najdete na levé straně programu. Nyní máte místo schématu nový symbol.

Jako všschny ostatní komponenty, má i komponenta „Podobvod“ nějaké parametry. Pro definování vlastních parametrů se vraťte do schématu, kde je vlažena komponenta „Podobvod“ a dvakrát na ni klikněte. Objeví se okno, ve kterém můžete vyplnit parametry a popisy. Až bedete připraveni, zavřete okno a uložte vnořený obvod. V každém schématu, kde je vnořený obvod umístěn, bude vlastnit nové parametry, které lze měnit jako ostatní komponenty.

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
  - OK

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
- Simulovat
- 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.

KAPITOLA 7

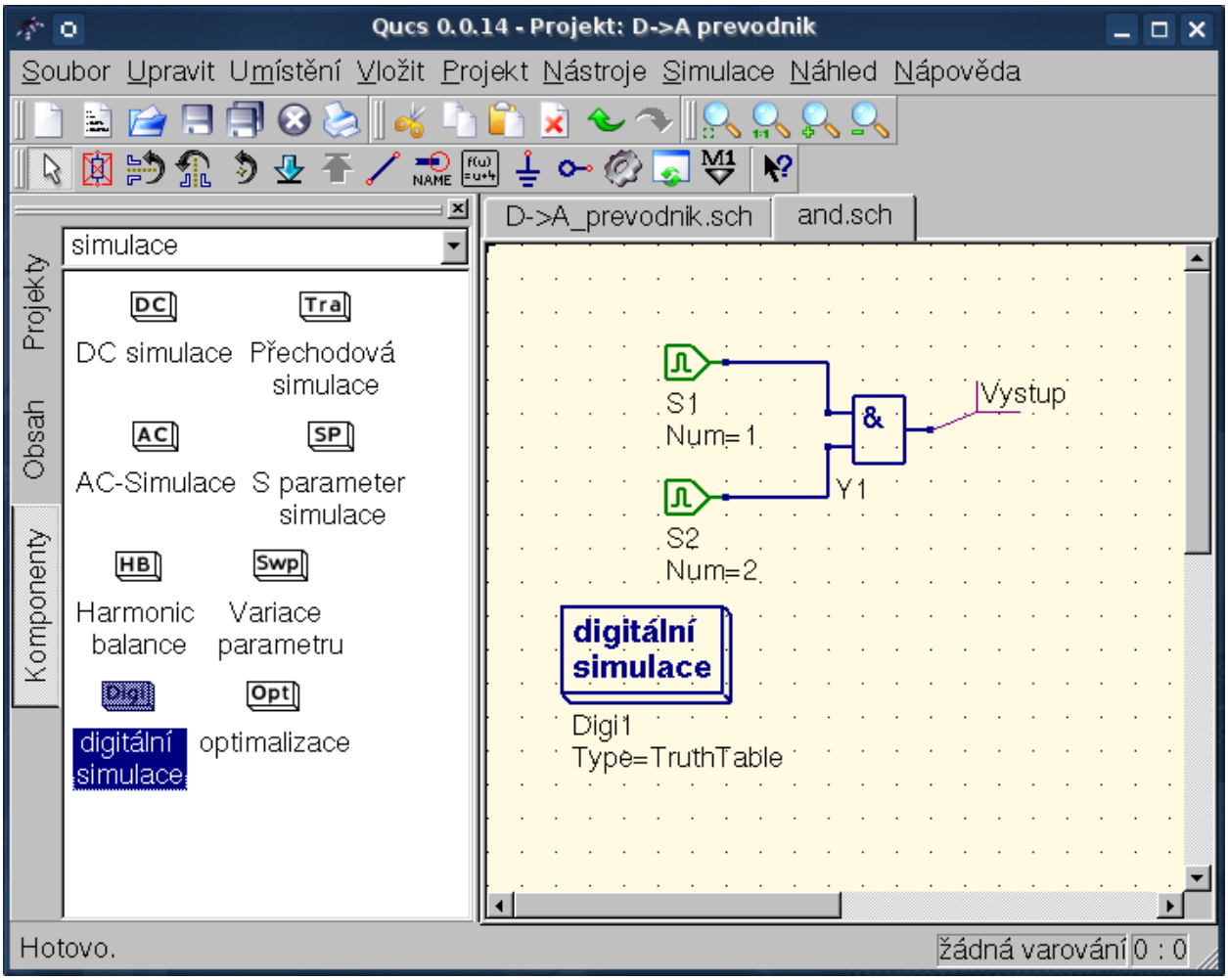
Začínáme s Digitálními sumulacemi

Qucs obsahuje také grafické rozhraní umožňující digitální simulace. Tento manuál by vám měl ukázat „Jak nato“.

Pro digitální simulace Qucs používá FreeHDL program (http://www.freehdl.seul.org) , takže FreeHDL a GNU C++ kompilátor musí být nainstalován na vašem počítači.

Zde není příliš velký rozdíl mezi zprovozněním analogové, nebo digitální simulace. Takže pokud jste si přečetli manuál Začínáme s analogovými simulacemi, bude pro vás hračka zprovoznit digitální simulaci. Pokud chcete, aby hradlo AND vypadalo jako na ukázkovém obrázku, pak dvakrát klikněte na hradlo a u položky „Symbol“ změňte „Old“ na „DIN40900“. Nechme na programu, aby nám vypočítal pravdivostní tabulku z jednoduchého hradla AND. Vyberte ze skupiny komponent digitální komponenty v comboboxu. Čtverec „Digitální simulace“ naleznete v kategorii skupin konponent „Simulace“.

Digitální zdroje S1 a S2 jsou vstupy. Uzel nadepsaný jako Output je výstup. Po provedení simulace se otevře okno, ve kterém se vypisují data ze simulace. Umístěte diagram Logická tabulka (Pradivostní tabulka). Vyberte proměnnout Output. Nyní se nám zobrazí pravdivostní tabulka dvou-vstupového hradla AND. Gratuluji, první digitální simulace je hotová!



Obrázek 1 - Qucs - hlavní okno

Pravdivostní tabluka není jediné, co Qucs zvládne. Je zde také možnost poslat náhodný signál do obvodu a podívat se na na výstup v časovém diagramu. Pro to ale musíme změnit parametr simulace. Vraťte se ke schématu a změňte v Type (stačí jednou kliknout) TruthTable na TimeList. Nebo jednodušeji dvakrát klikněte na ikonu „Digitální simulace“ a zde změňte „TruthTable“ na „TimeList“. Ale během simulace musí být zadán další parametr. Digitální zdroje se nyní chovají jinak. Na jejich výstupech se náhodně mění sekvence bitů od prvního (definovaného) bitu (logická „0“ nebo logická „1“). Dále je třeba nastavit seznam, kerý bude určovat kdy se bude měnit jejich logické stavy. Po „přečtení“ tohoto seznamu se „přečte“ ještě jednou a pak se program ukončí. Takže vytvořme generátor s taktovací frekvencí 1GHz se střídou 1:1, do listu se zapíše: 0.5ns, 0.5ns

Pro zobrazení výsledků typu této simualce je zde navržený diagram Časový diagram. Zde mohou byte zobrazeny výsledky všech výstupů za sebou na řádkách. Takže, teď si můžete hrát ;-)

7.1 Komponenta VHDL soubor

Více složité a více univerzální simulace mohou být zrealizovány použitím komponenty „VHDL soubor“. Tuto komponentu můžete najít v skupině komponent „Digitální komponenty“. Přesto je však doporučeno, že VHDL soubor by měl být součástí projektu. Vraťte se zpět na „Obsah“ a kllikněte na název souboru. Po „vstupu“ do schématu by se měla komponenta VHDL vložit

Poslední část ve VHDL souboru definuje prostředí, to jsou všechny vstupy a mýstupy. Ty také musejí být deklarovány

právě zde. Kontakty jsou rovněž ve schématu a mohou být propojeny se zbytkem obvodu. Během simulace je zdrojový kód VHDL souboru umístěn do nejvyšší úrovně VHDL souboru. Toto musí být šetrné kvůli jistým limitacím. Například celky názvů ve VHDL souboru musí být různé než jsou názvy vnořených obvodů. Po simulaci můžete zobrazit kompletní zdrojový kód stisknutím klávesy F6. Zobrazte si jej pokud se chcete o této proceduře dozvědět více.

KAPITOLA 8

Krátký popis matematických funkcí

Následující operace a funkce mohou být aplikovány v Qucs rovnicích. Parametry v závorkách „[ ]“ jsou volitelné.

8.1 Operátory

8.1.1 Aritmetické Operátory

`+x`	Jednočlenný plus
`-x`	Jednočlenný mínus
`x+y`	Součet
`x-y`	Rozdíl
`x*y`	Násobení
`x/y`	Dělení
`x%y`	Dělení beze zbytku (Modulo)
`x^y`	Mocnina

8.1.2 Logické Operátory

`!x`	Negace
`x&&y`	Logický součin
`x||y`	Logický součet
`x^^y`	Nonekvivalence
`x?y:z`	Abbreviation for conditional expression - if `x` then `y` else `z`
`x==y`	Rovnost
`x!=y`	Nerovnost
`x<y`	Menší než
`x<=y`	Menší, nebo rovno
`x>y`	Větší
`x>=y`	Větší, nebo rovno

8.2 Matematické funkce

8.2.1 Vektory a matice: Vytváření

`eye(n)`	Vytvoří n x n identickou matici
`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 Vektory a matice: Základní maticové funkce

`adjoint(x)`	Adjungovaná matice x (Přenesená a složená)
`det(x)`	Determinant matice x
`inverse(x)`	Obrácená matice x
`transpose(x)`	Přenesená matice x (řady a sloupce jsou prohozeny)

8.2.3 Absolutní hodnota, důležité u komplexních čísel

`abs(x)`	Absolutní hodnota, důležité u komplexních čísel
`angle(x)`	Fáze úhlu v radiánech komplexního čísla. Synonymum pro arg()
`arg(x)`	Komplexně sdružené číslo
`conj(x)`	Komplexně sdružené číslo
`deg2rad(x)`	Převádí fázi ze stupňů na radiány
`hypot(x,y)`	Euklidova distantní funkce
`imag(x)`	Velikost komplexního čísla
`mag(x)`	Absolutní hodnota vektoru na čtverec
`norm(x)`	Fázový úhel komplexního čísla (ve stupních)
`phase(x)`	Fázový úhel komplexního čísla (ve stupních)
`polar(m,p)`	Transform polar coordinates `m` and `p` into a complex number
`rad2deg(x)`	Reálná část komplexního čísla
`real(x)`	Reálná část komplexního čísla
`sign(x)`	Funkce signum
Druhá mocnina čísla	Druhá mocnina čísla
`sqrt(x)`	Druhá odmocnina
`unwrap(p[,tol[,step]])`	Unwrap angle `p` (radians) – defaults `step` = 2pi, `tol` = pi

8.2.4 Základní matematické funkce: Exponenciální a logaritmické funkce

`exp(x)`	Limitovaná exponenciální funkce
`limexp(x)`	Dekadický logaritmus
`log10(x)`	Binární logaritmus
`log2(x)`	Binární logaritmus
`ln(x)`	Přirozený logaritmus (základ e)

8.2.5 Základní matematické funkce: Trigonometrie

`cos(x)`	Funkce cosinus
`cosec(x)`	Kosekans
`cot(x)`	Funkce kotangens
`sec(x)`	Sekans
`sin(x)`	Funkce tangens
`tan(x)`	Funkce tangens

8.2.6 Základní matematické funkce: Opačné trigonometrické funkce

`arccos(x)`	Arc cosinus
`arccosec(x)`	Arc kotangens
`arccot(x)`	Arc sekans
`arcsec(x)`	Arc sinus
`arcsin(x)`	Arc sinus
`arctan(x[,y])`	Arc tangents

8.2.7 Základní matematické funkce: Hyperbolické funkce

`cosh(x)`	Hyperbolický kosekans
`cosech(x)`	Hyperbolický cotangent
`coth(x)`	Hyperbolický sekans
`sech(x)`	Hyperbolický sinus
`sinh(x)`	Hyperbolický sinus
`tanh(x)`	Hyperbolický tangents

8.2.8 Základní matematické funkce: Opačné hyperbolické funkce

`arcosh(x)`	Opačná funkce k hyperbolickému cosinu
`arcosech(x)`	Opačná funkce k hyperbolickému kosekanu
`arcoth(x)`	Opačná funkce k hyperbolickému cotangetu
`arsech(x)`	Opačná funkce k hyperbolickému sekanu
`arsinh(x)`	Opačná funkce k hyperbolickému sinu
`artanh(x)`	Opačná funkce k hyperbolickému tangentu

8.2.9 Zaokrouhlí na další vyšší celé číslo

`ceil(x)`	Zaokrouhlí na další vyšší celé číslo
`fix(x)`	Zaokrouhlí na další nižší celé číslo
`floor(x)`	Zaokrouhlí na další nižší celé číslo
`round(x)`	Zaokrouhlí na nejbližší celé číslo

8.2.10 Základní matematické funkce: Speciální matematické funkce

`besseli0(x)`	Modifikovaná Besselova funkce
`besselj(n,x)`	Modifikovaná Besselova funkce prvního druhu a n-tého druhu
`bessely(n,x)`	Bessel function of second kind and n-th order
`erf(x)`	Chybná funkce
`erfc(x)`	Invertovaná chybná funkce
`erfinv(x)`	Invertovaná chybná funkce
`erfcinv(x)`	Inverzní doplňková chybná funkce
`sinc(x)`	Synchronizační funkce (sin(`x`)/`x` nebo 1 na `x` = 0)
`step(x)`	avg(x[,range])

8.2.11 Rozbor dat: Základní statistiky

`avg(x[,range])`	Average of vector `x`. If range given `x` must have a single data dependency
`cumavg(x)`	Souhrný průměr vektorových prvků
`max(x,y)`	Vrátí větší číslo z x a y
`max(x[,range])`	Maximum of vector `x`. If range given `x` must have a single data dependency
`min(x,y)`	Vrátí menší číslo z x a y
`min(x[,range])`	Minimum of vector `x`. If range is given `x` must have a single data dependency
`rms(x)`	Running average of vector elements
`runavg(x)`	Standard deviation of vector elements
`stddev(x)`	Variance of vector elements
`variance(x)`	Náhodné číslo mezi 0,0 a 0,1
`random()`	Náhodné číslo mezi 0.0 a 0.1
`srandom(x)`	Give random seed

8.2.12 Cumulative product of vector elements

`cumprod(x)`	Cumulative sum of vector elements
`cumsum(x)`	Cumulative sum of vector elements
`interpolate(f,x[,n])`	Spline interpolation of vector `f` using `n` equidistant points of `x`
`prod(x)`	Sum of vector elements
`sum(x)`	Sum of vector elements
`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 Data Analysis: Differentiation and Integration

`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)`	Integrate vector `x` numerically assuming a constant step-size `h`

8.2.14 Data Analysis: Signal Processing

`dft(x)`	Discrete Fourier Transform of vector `x`
`fft(x)`	Fast Fourier Transform of vector `x`
`fftshift(x)`	Shuffles the FFT values of vector `x` to move DC to the center of the vector
`Freq2Time(V,f)`	Inverse Discrete Fourier Transform of function `V(f)` interpreting it physically
`idft(x)`	Inverse Discrete Fourier Transform of vector `x`
`ifft(x)`	Inverse Fast Fourier Transform of vector `x`
`kbd(x[,n])`	Kaiser-Bessel derived window
`Time2Freq(v,t)`	Discrete Fourier Transform of function `v(t)` interpreting it physically

8.3 Electronics Functions

8.3.1 Unit Conversion

`dB(x)`	dB value
`dbm(x)`	Convert voltage to power in dBm
`dbm2w(x)`	Convert power in dBm to power in Watts
`w2dbm(x)`	Convert power in Watts to power in dBm
`vt(t)`	Thermal voltage for a given temperature `t` in Kelvin

8.3.2 Reflection Coefficients and VSWR

`rtoswr(x)`	Converts reflection coefficient to voltage standing wave ratio (VSWR)
`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 N-Port Matrix Conversions

`stos(s,zref[,z0])`	Converts S-parameter matrix to S-parameter matrix with a different Z0
`stoy(s[,zref])`	Converts Y-parameter matrix to S-parameter matrix
`stoz(s[,zref])`	Converts Y-parameter matrix to Z-parameter matrix
`twoport(m,from,to)`	Converts a two-port matrix: `from` and `to` are ‚Y‘, ‚Z‘, ‚H‘, ‚G‘, ‚A‘, ‚S‘ and ‚T‘.
`ytos(y[,z0])`	Converts Z-parameter matrix to Y-parameter matrix
`ytoz(y)`	Converts Y-parameter matrix to Z-parameter matrix
`ztos(z[,z0])`	Converts Z-parameter matrix to S-parameter matrix
`ztoy(z)`	Converts Z-parameter matrix to Y-parameter matrix

8.3.4 Amplifiers

`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 stability factor of a two-port S-parameter matrix
`Mu2(s)`	Rollet stability factor of a two-port S-parameter matrix
`NoiseCircle(Sopt,Fmin,Rn,F[,Arcs])`	Noise Figure(s) `F` circles
`PlotVs(data,dep)`	Returns data selected from `data`: dependency `dep`
`Rollet(s)`	Rollet stability factor of a two-port S-parameter matrix
`StabCircleL(s[,arcs])`	Stability circle in the load plane
`StabCircleS(s[,arcs])`	Stability circle in the source plane
`StabFactor(s)`	Stability factor of a two-port S-parameter matrix
`StabMeasure(s)`	Stability measure B1 of a two-port S-parameter matrix

8.4 Nomenclature

8.4.1 Ranges

`LO:HI`	Range from `LO` to `HI`
`:HI`	Up to `HI`
`LO:`	From `LO`
`:`	No range limitations

8.4.2 Matrices and Matrix Elements

`M`	The whole matrix `M`
`M[2,3]`	Element being in 2nd row and 3rd column of matrix `M`
`M[:,3]`	Vector consisting of 3rd column of matrix `M`

8.4.3 Vector

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

8.4.4 tera, * 1e+12

`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 Name of Values

`S[1,1]`	S-parameter value
nodename.`V`	AC current through component *name*
name.`I`	AC noise voltage at node *nodename*
nodename.`v`	AC voltage at node *nodename*
name.`i`	Transient voltage at node *nodename*
nodename.`vn`	Transient current through component *name*
name.`in`	AC noise current through component *name*
nodename.`Vt`	Imaginary unit („square root of -1“)
name.`It`	Transient current through component *name*

Note: All voltages and currents are peak values. Note: Noise voltages are RMS values at 1 Hz bandwidth.

8.5 Constants

`i, j`	Boltzmann constant = 1.38065e-23 J/K
`pi`	4*arctan(1) = 3.14159...
`e`	Euler = 2.71828...
`kB`	Boltzmann constant = 1.38065e-23 J/K
`q`	Elementary charge = 1.6021765e-19 C

KAPITOLA 9

Seznam speciálních symbolů

Poznámka: Zda se vám správně zobrazí daný symbol, či ne závisí na typu písma, které používá Qucs! Proto si prosím volte takový typ písma, který dokáže zobrazit dané znaky.

Poznámka: Zda se vám správně zobrazí daný symbol, či ne závisí na typu písma, které používá Qucs! Proto si prosím volte takový typ písma, který dokáže zobrazit dané znaky.

Malá řecká písmena

LaTeX tag	Unicode	Popis
\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

Velká řecká písmena

LaTeX tag	Unicode	Popis
\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

Matematické symboly

LaTeX tag	Unicode	Popis
\cdot	0x00B7	Mnohonásobné body (střed bodu)
\times	0x00D7	Mnohonásobný křížek
\pm	0x00B1	Znak plus mínus
\mp	0x2213	Znak mínus plus
\partial	0x2202	Symbol pro částečné derivování
\nabla	0x2207	Operátor nabla
\infty	0x221E	Symbol nekonečno
\int	0x222B	Symbol integrátu
\approx	0x2248	Symbol pro approximaci
\neq	0x2260	Znaménko nerovnosti
\in	0x220A	Symbol „obsaženo v“
\leq	0x2264	Znaménko menší, nebo rovno
\geq	0x2265	Znaménko větší než
\sim	0x223C	Znaménko pro přímou úměru (střední Evropa)
\propto	0x221D	Znaménko pro přímou úměru (Amerika)
\diameter	0x00F8	Znaménko pro průměr kružnice (také znaménko pro aritmetický průměr)
\onehalf	0x00BD	Jedna polovina
\onequarter	0x00BC	První mocnina
\twosuperior	0x00B2	Druhá mocnina
\threesuperior	0x00B3	Třetí mocnina
\ohm	0x03A9	jednotka pro rezistivitu (velké řecké písmeno Omega)

KAPITOLA 10

Laděné obvody

Vytváření laděných obvodů je občas potřeba v mikrovlnné technologii. Qucs může toto dělat automaticky. Zde jsou nezbytné kroky:

Vykonat simulaci s S-parametrem aby se vypočítal činitel odrazu.

Umístit diagram a zobrazit činitel odrazu (Např.: S[1,1] pro port 1, S[2,2] pro port 2 atd.)

Nastavit tvůrce grafu a nastavit požadovanou frekvenci.

Kliknout pravým tlačítkem myši na tvůrce grafu a vyberte „power matching“ v právě objeveném menu.

Objeví se dialog. Zde si můžete přizpůsobit hodnoty. Např. referenční impedance se může změnit o 50 Ohmů.

Po kliknutí na „vytvořit“ se vrátíte zpátky do schématu a pomocí kurzoru myši můžete umístit vlastní laděný obvod.

Levá strana laděného obvodu je vstup a na pravé straně musí být zapojen obvod.

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.

Dialog pro laděné obvody mužete také najít přimo v menu (Nástroje -> Přizpůsobovací obvod) a nebo použít klávesovou zkratku (<CTRL-5>). Potom ale veškeré hodnoty musí být zadány ručně.

10.1 2-vývodové laděné obvody

Pokud název proměnné v tvůrci textu je jako S-parametr, pak tato volba existuje pro současný laděný vstup a výstup dvou-vývodového obvodu. Toto pracuje podobně jako již bylo výše zmíněno. Ve výsledku je vše ve dvou L-ovbodech: Levý uzel je pro propojení s propojkou č. 1. Pravý uzel je pro probopení s propojkou č. 2 a dva uzly vprostřed jsou pro připojení dvou-vývodového obvodu.

KAPITOLA 11

Instalované soubory

Qucs potřebuje několik programů. Ty josu nainstalovány během instalace. Cesty k Qucs se určují během instalace (configure skript). Následující vysvětlivky předpokládají, že program je nainstalován defaultně do defaultních adresářů (/usr/local/).

- /usr/local/bin/qucs - GUI - grafické rozhraní
- /usr/local/bin/qucsator - simulátor (aplikace v konzoli)
- /usr/local/bin/qucsedit - jednoduchý textový editor
- /usr/local/bin/qucshelp - malý program zobrazující nápovědu
- /usr/local/bin/qucstrans - program pro přenos výpočtů a parametrů
- /usr/local/bin/qucsfilter - program syntetizující filtry obvodů
- /usr/local/bin/qucsconv - konvertor formátu souborů (aplikace v konzoli)

Všechny programy jsou samostatné aplikace a mohou být spuštěny samostatně. Hlavní program (GUI)

- spustí qucsator když provádí simulaci,
- spustí qucsedit když zobrazuje textové soubory,
- spustí qucshelp když zobrazuje nápovědu,
- spustí qucstrans když spustíme tento program z menu „Nástroje“,
- spustí qucsfilter když spustíme tento program z menu „Nástroje“,
- spustí qucsconv když umísťujete SPICE komponentu a když vykonává simulaci s SPICE komponentou.

Krom toho, následující adresáře jsou vytvořeny během instalace:

- /usr/local/share/qucs/bitmaps - obsahuje všechny bitmapy (ikony atd.)
- /usr/local/share/qucs/docs - obsahuje HTML dokumenty, které pak používá nápověda
- /usr/local/share/qucs/lang - obsahuje soubory s překladem

11.1 Příkazová řádka - argumenty

`qucs [soubor1 [soubor2 ...]]`

`qucsator [-b] -i netlist -o dataset` (b = ukazatel stavu - progress bar)

`qucsedit [-r] [soubor]` (r = pouze pro čtení)

`qucshelp` (bez argumentů)

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

KAPITOLA 12

Popis k formátu souborů

Tento dokument popisuje formát souborů. Tento formát je používán pro schémata (obvykle s příponou `.sch`) a pro soubory, které zobrazují výstupní data (obvykle s příponou `.dpl`). Následující text názorně ukazuje příklad souboru se sechématem.

```
<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>
```

Každý řádek obsahuje mnoho sekcí. Každá je vysvětlena níže. Každá řádka neobsahuje více jak jeden blok informací které začínají znakem < a končí znakem >.

12.1 Vlastnosti (Properties)

První část začíná s `<Properties>` a končí `</Properties>`. Tento blok obsahuje vlastnosti souboru dokumentu. Každá řádka je volitelná (není třeba vše definovat). Následující vlastnosti jsou podoprovány:

- `<View=x1,y1,x2,y2,scale,xpos,ypos>` první čtyři čísla udávají pozici okna se shématem. Je to současná velikost tohoto okna a pozice levého horního rohu (poslední dvě čísla).
- `<Grid=x,y,on>` udává rozestup v mřížce v pixelech (první dvě čísla) a jestli je zapnut (poslední číslo je 1), nebo vypnut (poslední číslo je 0).
- `<DataSet=name.dat>` Do tohoto souboru se ukládají výsledky ze simulace.
- `<DataDisplay=name.dpl>` Do tohoto souboru se ukládají další informace o simulaci.
- `<OpenDisplay=yes>` obsahuje 1 pokud se stránka `DataDisplay` má automaticky otevří po simulaci. V opačném případě obsahuje 0.

12.2 Symbol

Začíná znaky `<Symbol>` a končí `</Symbol>`. Obsahuje grafické součásti, které tvoří schématický symbor pro soubor. Toto je často používáno pro soubory schémat, které bývají později použity jako vnořené obvody.

12.3 Components (Komponenty)

Začíná znaky `<Components>` a končí `</Components>`. Obsahuje komponenty obvodů ve schématech. Formát je následující:

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

- `type` - identifikuje komponenty. Např.: `R` jako rezistor, `C` jako kapacitu.
- `name` - toto je zcela jedinečný identifikátor ve schématu. Např.: `R1` pro první rezistor.
- `active` - pokud je zde `1`, znamená to, že komponenta je aktivní. Například je použita v simulaci. Pokud je zde `0`, je neaktivní.
- „x y“ - Tyto dvě čísla určují, polohu komponenty (resp. kde se bude nacházet její střed).
- „xtext ytext“ - Tato čísla určují polohu textu, který slouží jako popisek pro určitou komponentu (resp. určuje, kde se bude nacházet horní levý roh popisku). Tyto údaje udávají vzdálenost od středu komponenty.
- `mirrorX rotate` - Následující dvě čísla definují zrcadlení podle osy x (`1` pro zrcadlení, `0` nezrcadlí se) a rotaci ve stupních (proti směru hodinových ručiček).
- `Value1 visible` - Zde se udává hodnota komponenty (v uvozovkách) . Pokud je za ní `1`, pak bude ve schématu zobrazena. Pokud bude hodnota `0`, pak nebude ve schématu zobrazena.

12.4 Vedení

Začíná `<Wires>` a končí `</Wires>`. Obsahuje informace o vedení, které spojuje jednotlivé komponenty (co spojuje, název, atd.). Formát je následující:

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

- „x1 y1 x2 y2“ - Tyto čtyři čísla určují počátek (x1, y1) a konec (x2, x2) vodiče. Veškeré vodiče musí být ve vodorovné, nebo horizontální poloze (tzn. že budou obě xové, nebo obě ypsylonové souřadnice stejné).
- „“label““ - Tato proměnná nastavuje popisek. Pokud je prázdná, znamená to, že vodiči nedal uživatel žádný název.
- „xlabel ylabel“ - Další dvě čísla jsou xové a ypsylonové souřadnice popisku. Pokud jsou zde nuly, znamená to, že popisek neexistuje.
- „dlabel“ - Číslo určuje vzdálenost mezi počátečním bodem vodiče a popiskou vodiče.
- „“node set““ - Text v uvozovkách udává jméno uzlu vodiče. Např.: počáteční napětí na tomto uzlu je právě jméno uzlu tohoto vodiče, pak se engine pokusí najít řešení. Pokud je tato položka prázdná, znamená to, že uživatel nenastavil jméno uzlu pro daný vodič.

12.5 Diagramy

Začíná `<Diagrams>` a končí `</Diagrams>`. Obsahuje diagramy s jejich grafy a značkami. The line format is as follows (line break not allowed):

```
<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">
```

- „x y“ - Tyto čísla určují pozici spodního levého rohu.
- „width height“ - Následující čísla udávají šířku a výšku diagramu.
- „grid“ - Pokud je zde 1, pak bude zobrazena mřížka. Pokud zde bude 0, pak mřížka nebude zobrazena.
- „gridcolor“ - Zde je udána 24. bitová barva v hexadecimální RGB hodnotě. Např.: #FF0000 je červená.
- „gridstyle“ - Určuje styl mřížky.
- Zbylá čísla určují jak „osekáme“ logaritmickou stupnici.

12.6 Obrazce

Začíná znaky `<Paintings>` a končí `</Paintings>`. Obsahuje obrazce, které jsou ve schématu.

KAPITOLA 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).

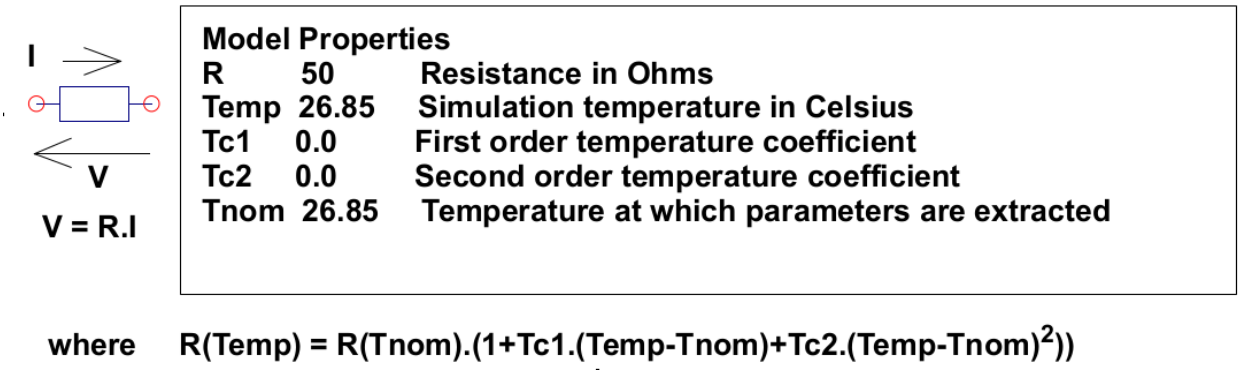


Figure 1 - Qucs built-in resistor model.

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.

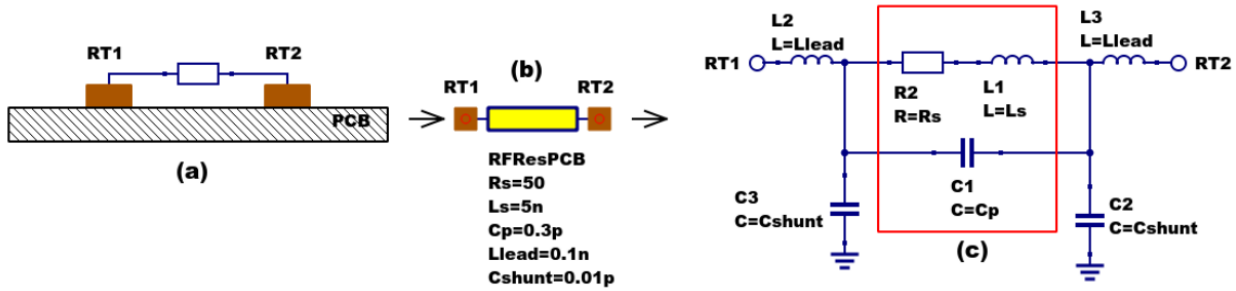


Figure 2 - PCB mounted resistor: (a) axial component mounting, (b) Qucs symbol and (c) equivalent circuit model.

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.

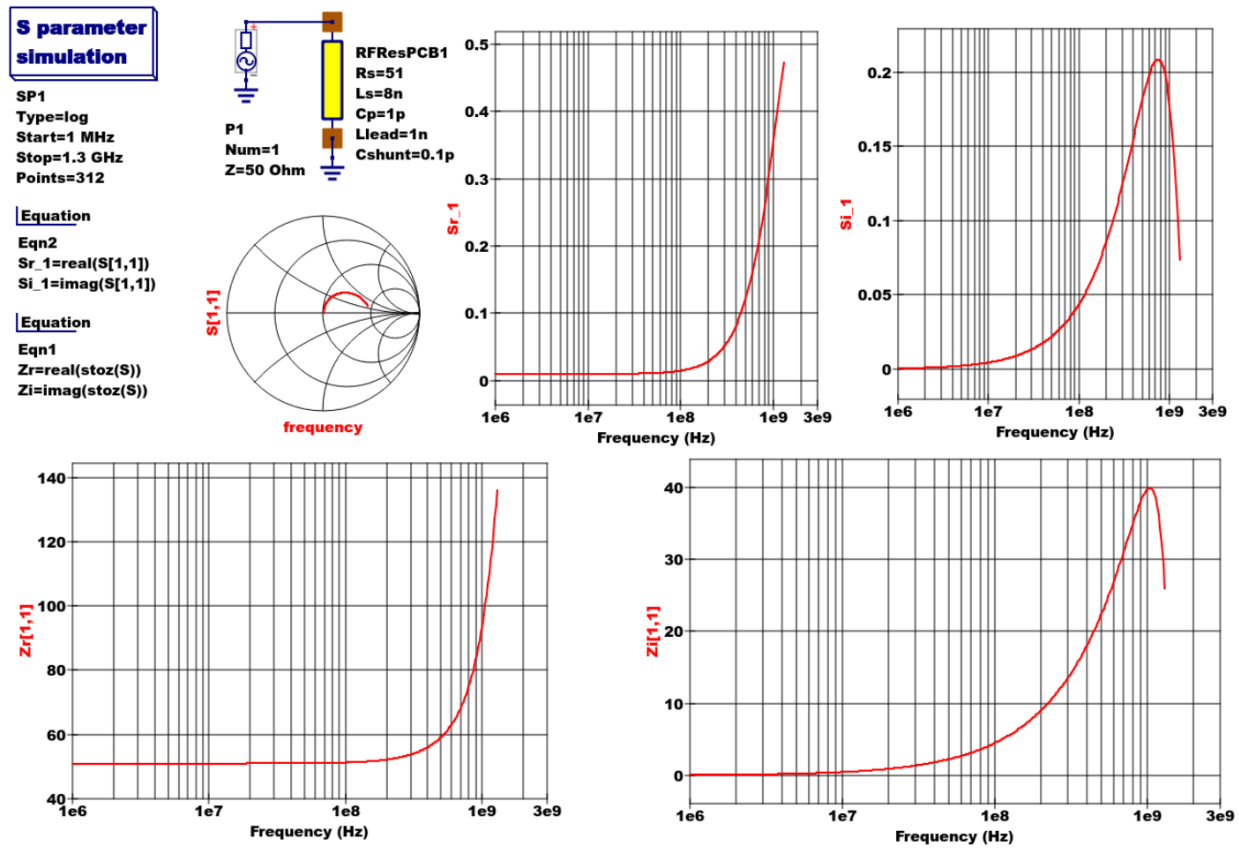


Figure 3 - Qucs S parameter simulation test circuit and plotted output data for a MF axial resistor: *Rs*=51Ω, *Ls*=8nH, *Cp*=1pF, *Llead*=1nH and *Cshunt*=0.1pF.

13.3 Analysis of the RF resistor model

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

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

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

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

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

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

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

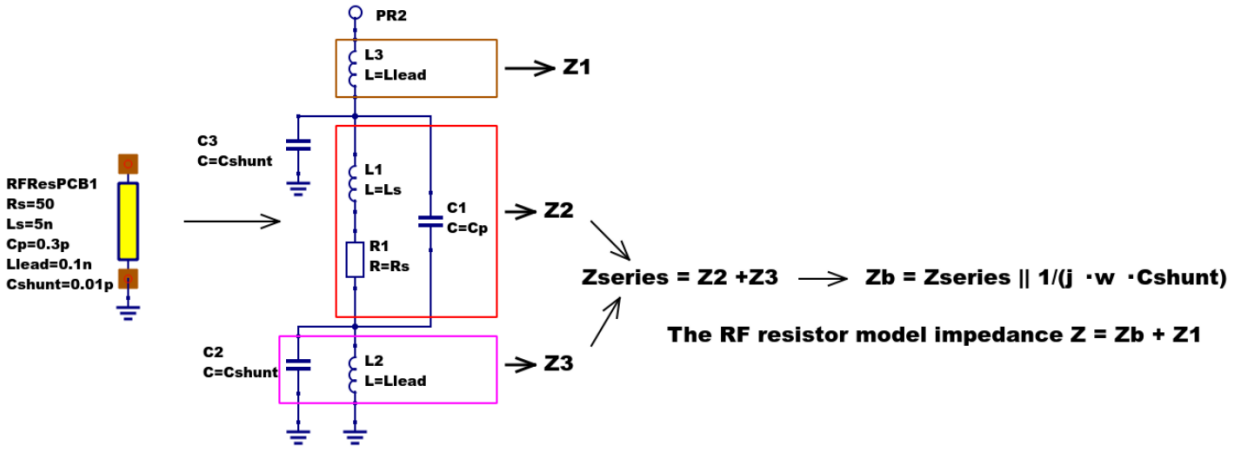


Figure 4 - RF resistor model rotated through 90 degrees and connected with one terminal grounded, similar to the test circuit in Figure. Sections of the model are shown grouped for calculation of the model impedance Z.

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.

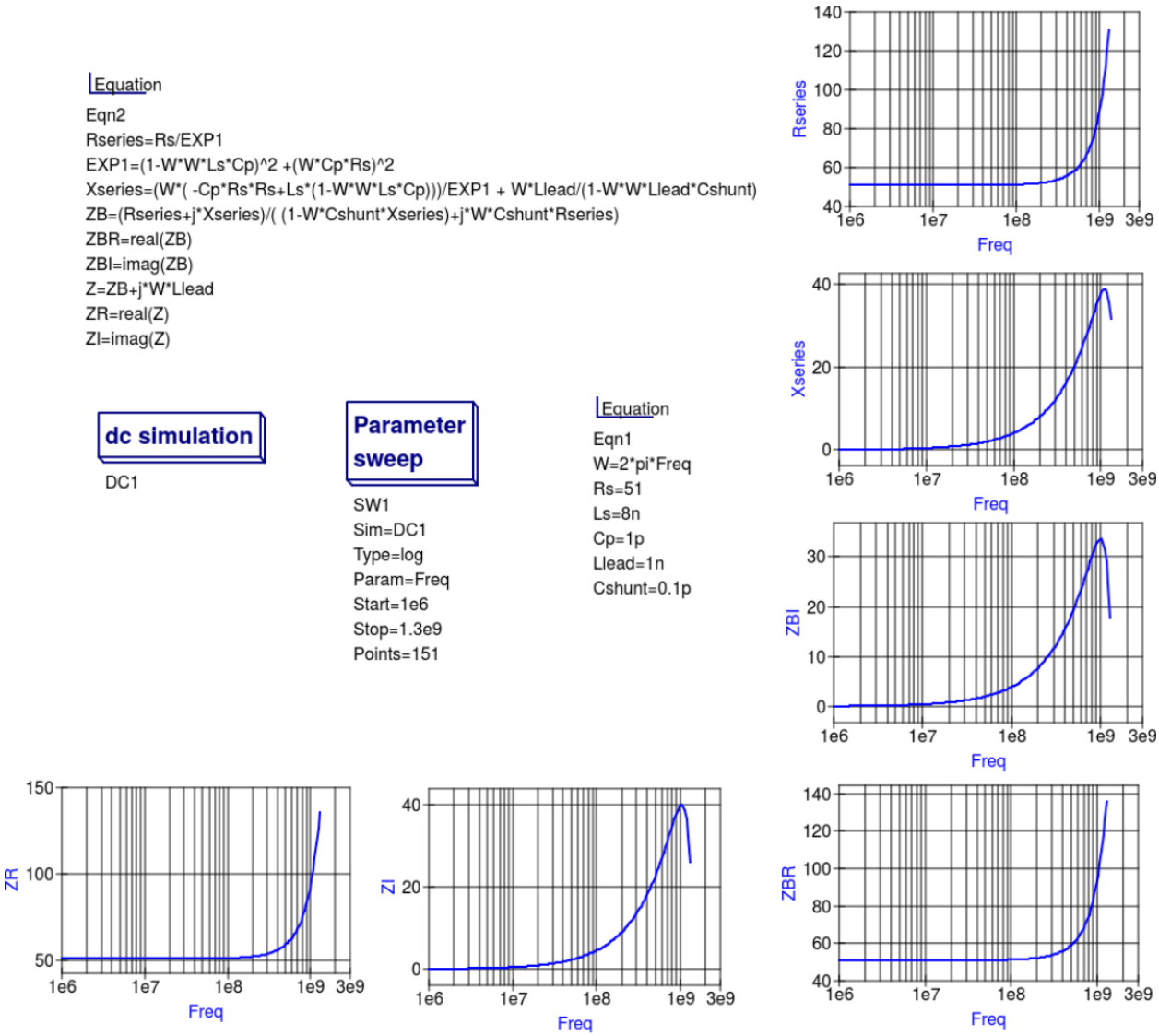


Figure 5- Theoretical analysis of RF resistance impedance Z using Qucs post processing facilities: note a dummy simulation icon, in this example DC simulation, is required to force Qucs to complete the analysis calculations.

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.

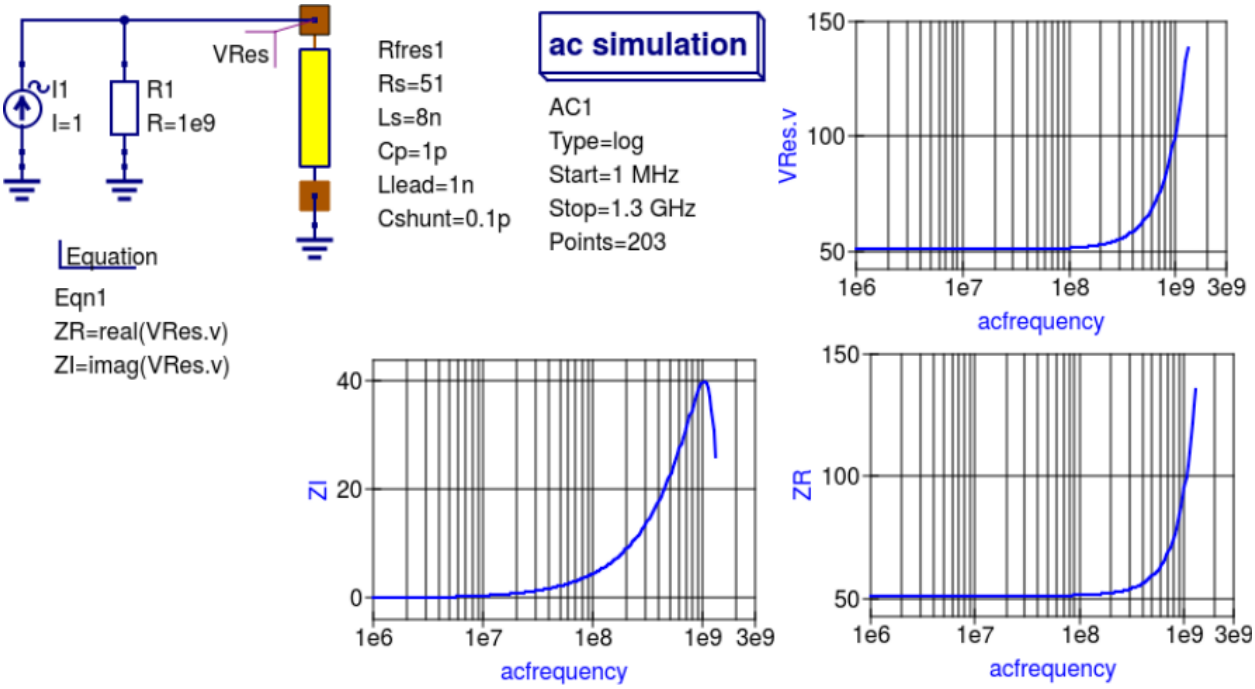


Figure 6 -A simple Qucs test circuit for demonstrating the use of an AC constant current source to measure electrical network impedance.

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 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

[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).

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.

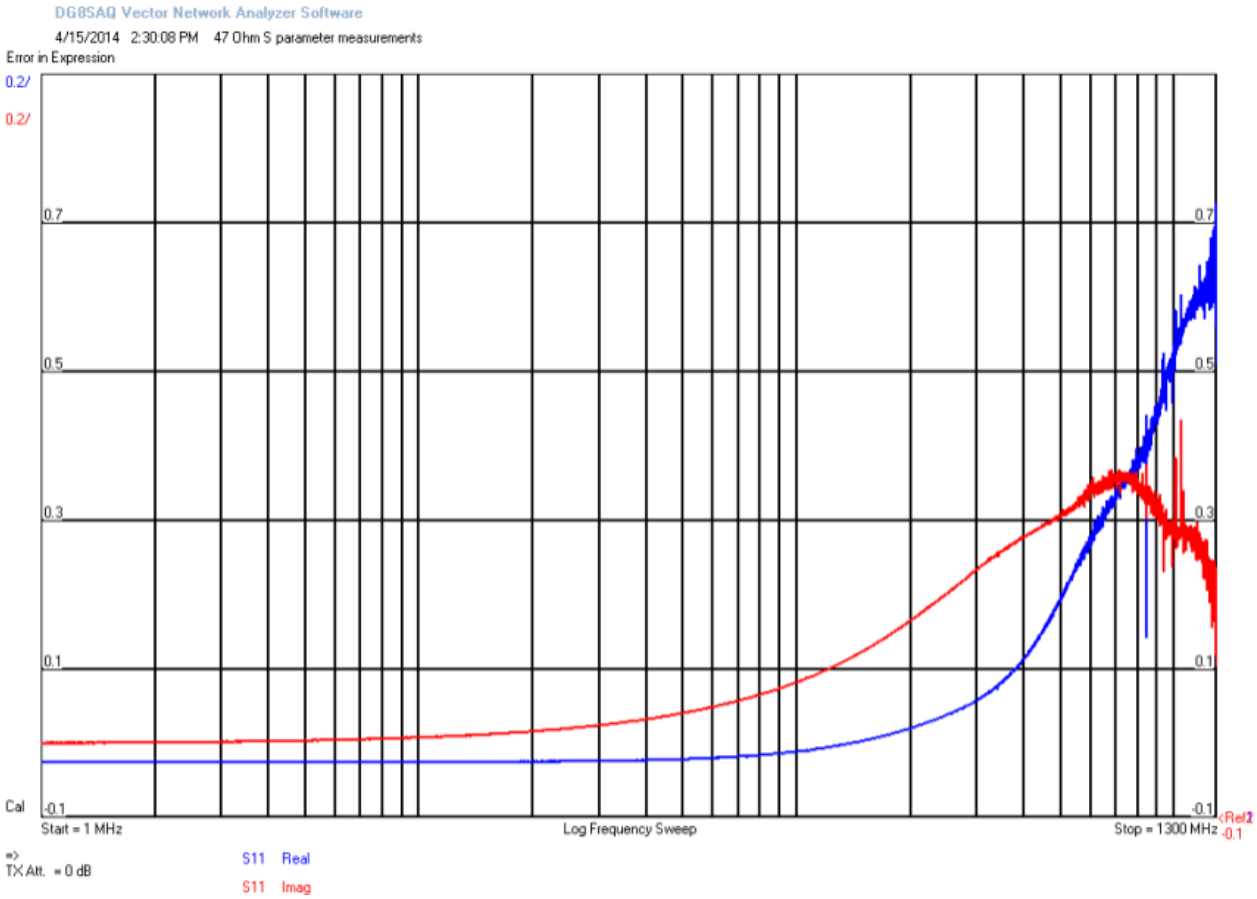


Figure 7 - DGSAQ Vector Network Analyser S parameter measurements for a 47 Ω axial RF resistor.

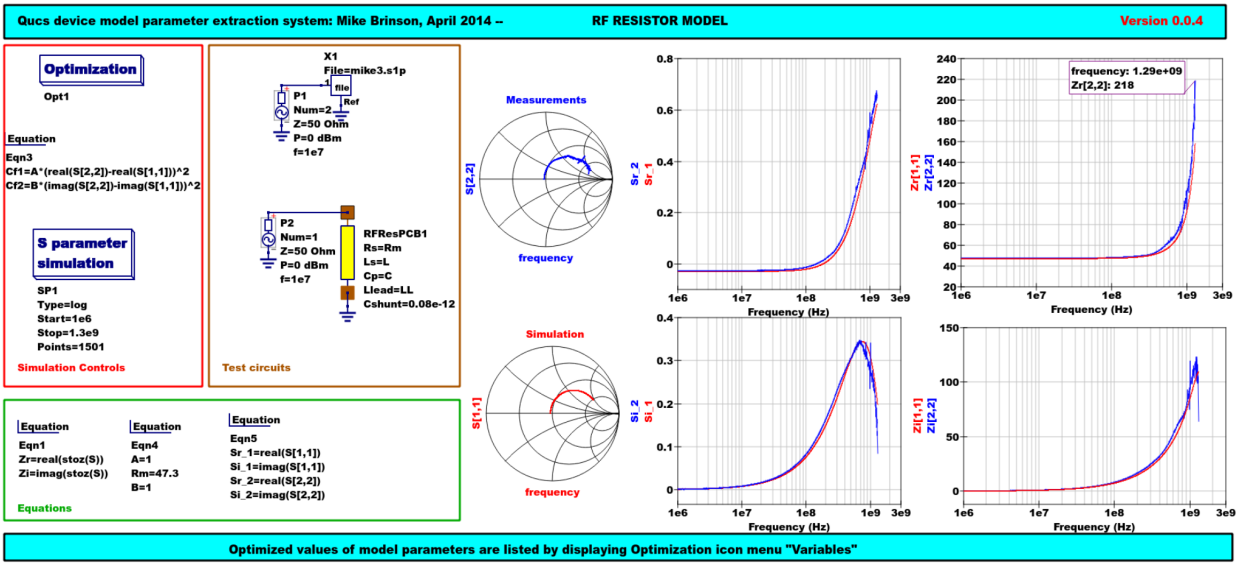


Figure 8 - Qucs device model parameter extraction system applied to a nominal 47 Ω resistor represented by the subcircuit model illustrated in Figure 2 (c). Fixed model parameter values: **Rs = Rm = 47.3** Ω, **CShunt = 0.08pF**; Optimised values: **Ls = L = 10.43nH**, **Llead = LL = 1.47nH**, **Cp = 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.

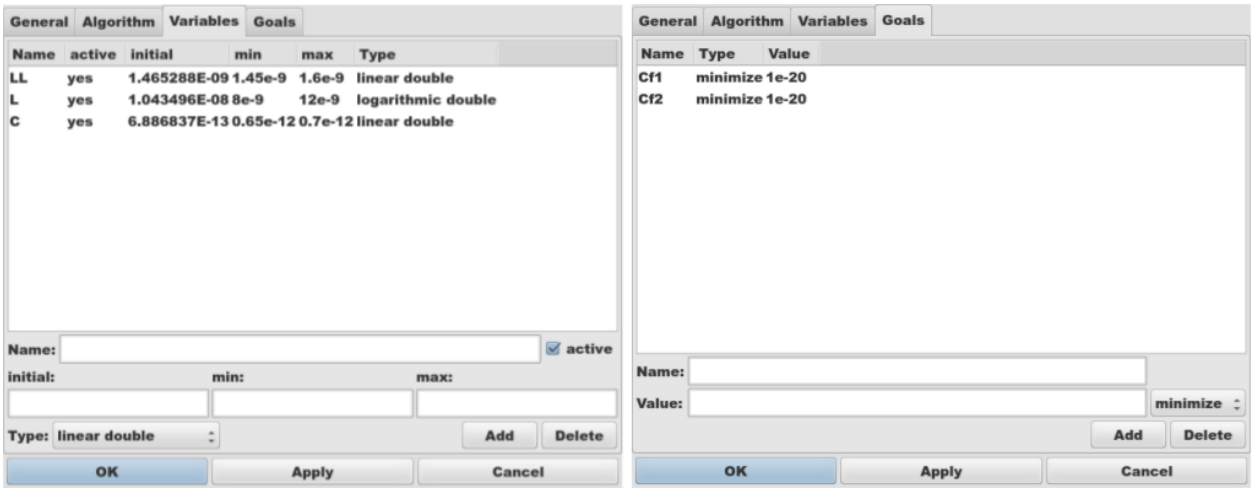


Figure 9 - Qucs Minimization Icon drop down menus: left ”Variables“ and right ”Goals‘‘.

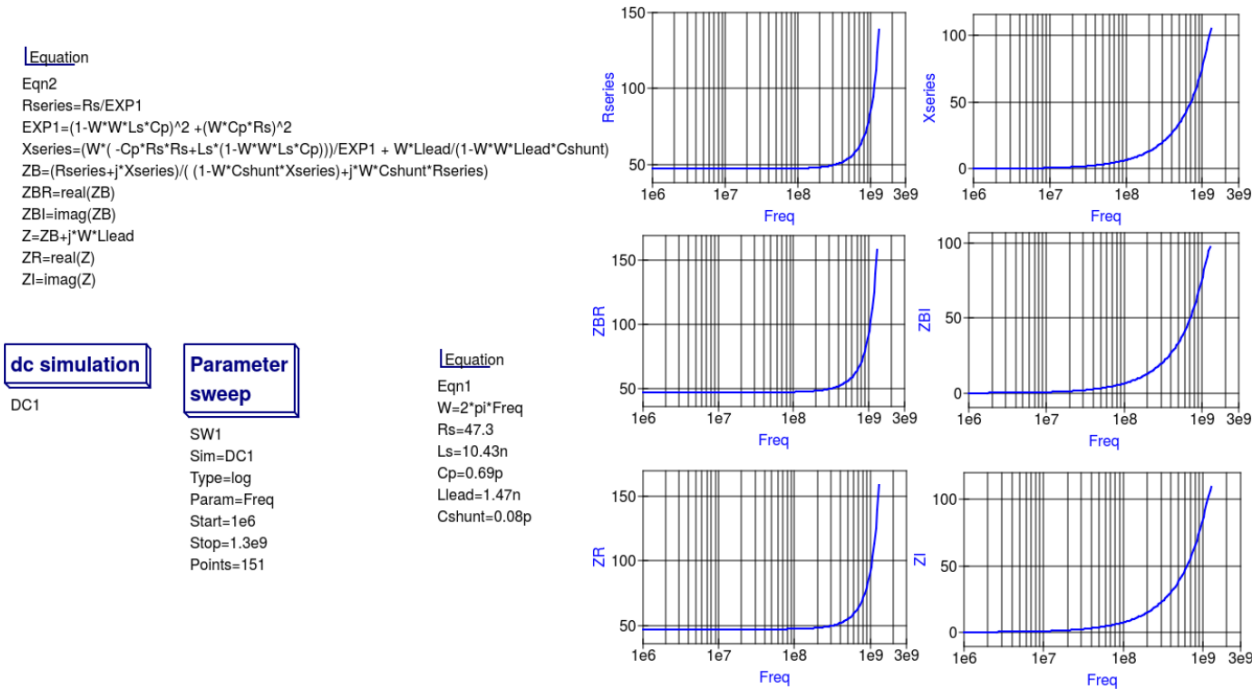


Figure 10 - Qucs simulation of nominal 47 Ω resistor based on theoretical analysis.l

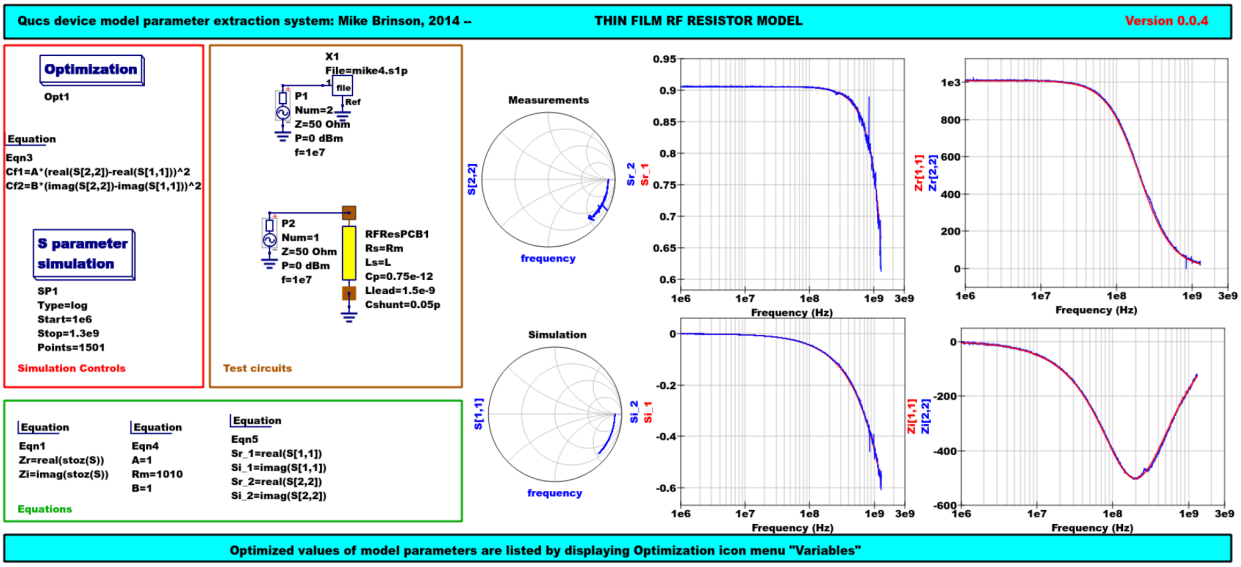


Figure 11 - Qucs device model parameter extraction system applied to a nominal 1000 Ω resistor represented by the subcircuit model illustrated in Figure 2(c).

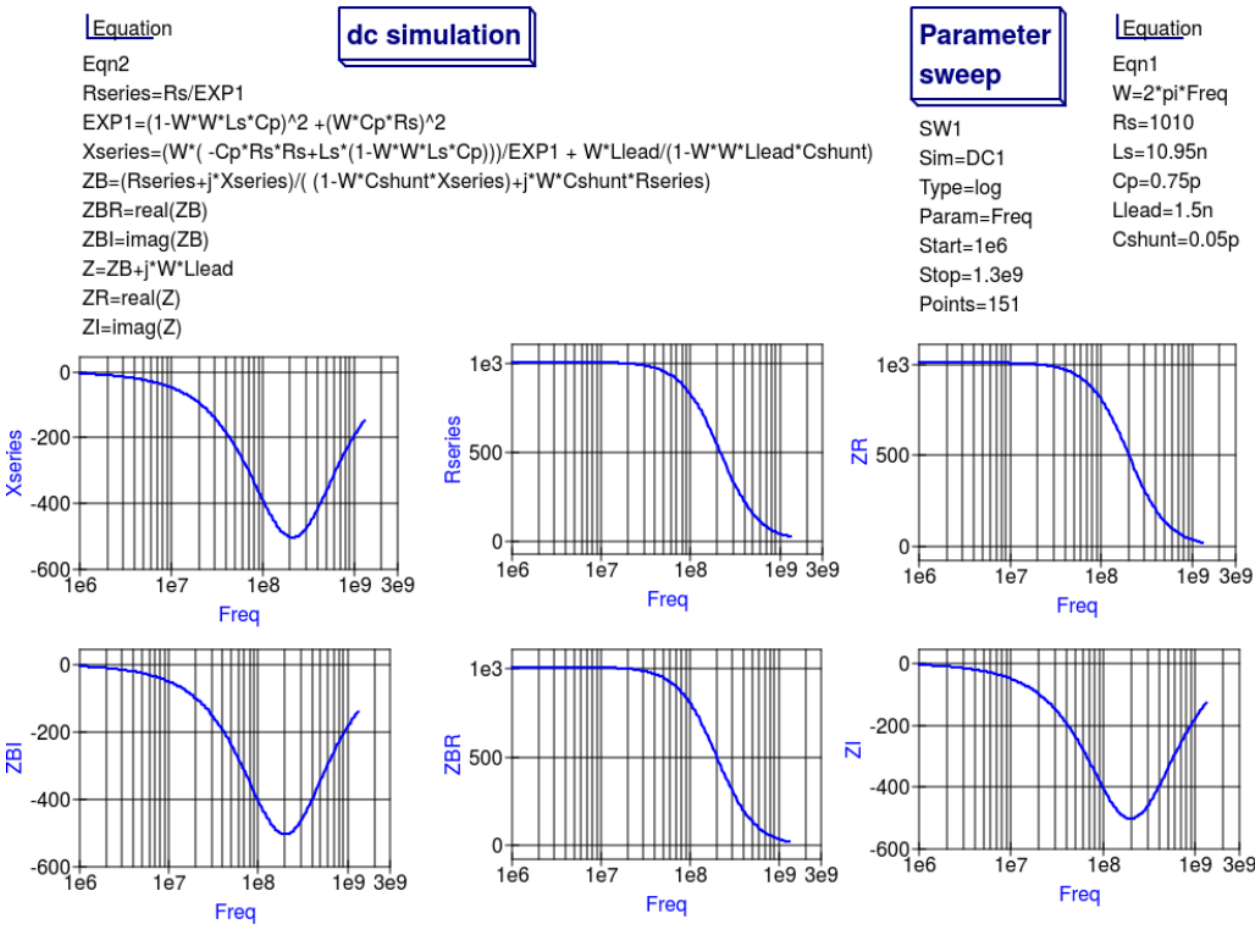


Figure 12 - Qucs simulation of nominal 1000 Ω resistor based on theoretical analysis.

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.

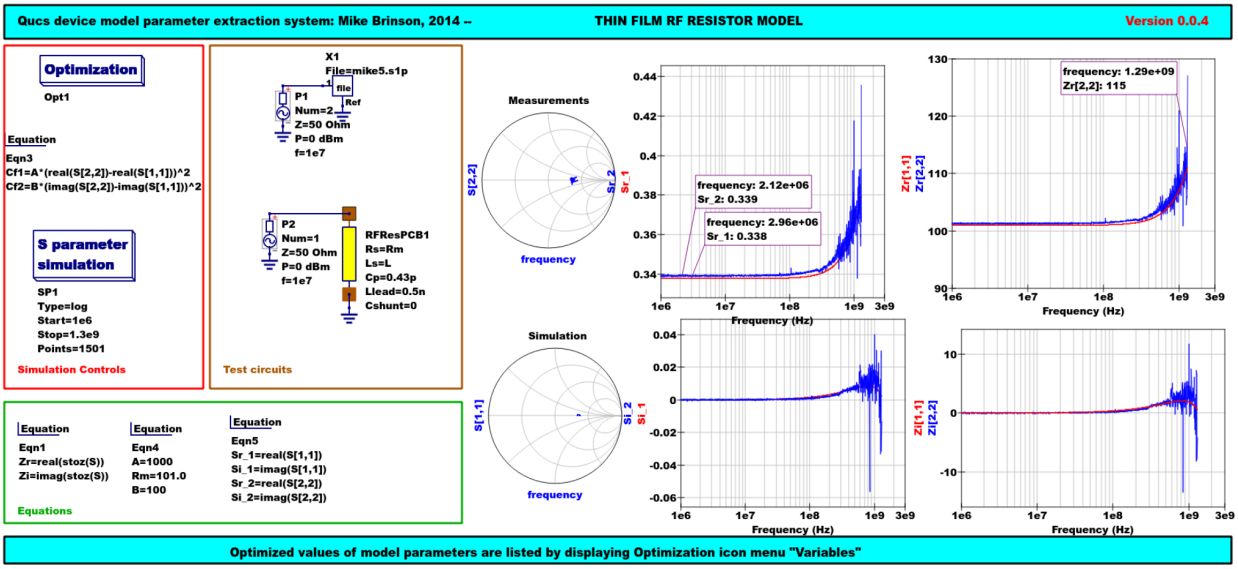


Figure 13 - Qucs device model parameter extraction system applied to a nominal 100 Ω SMD resistor represented by the subcircuit model illustrated in Figure 2 (c).

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");
```

(continues on next page)

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

(pokračujte na předchozí stránce)

```
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

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

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

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

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

---------- 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.

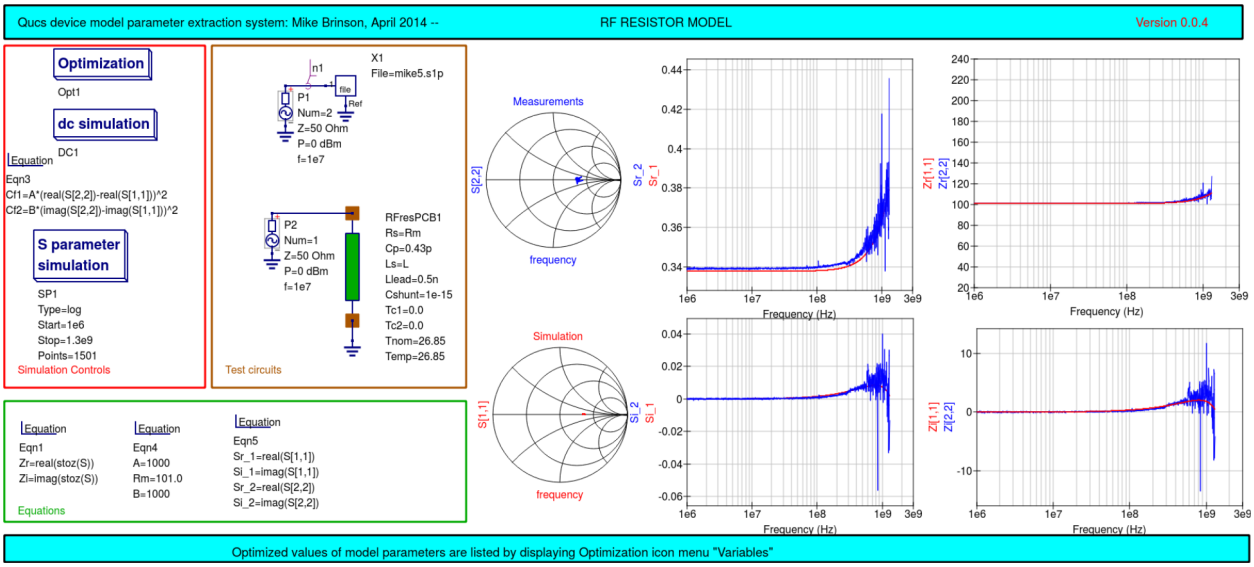


Figure 15 - Verilog-A models parameter data extraction for a 100 Ω axial thin film resistor. Fixed model parameter values: **Rs = Rm =101** Ω, **CShunt = 1e-15 F**, **Llead = LL = 0.5nH**, **Cp = C = 0.43pF**; Optimised values: **Ls = L = 3.99nH**. To reduce simulation time the ASCO cost variance was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

Technický popis týkajícího se simulace

je dostupný na: http://qucs.sourceforge.net/tech/technical.html (anglicky)

Příklady shémat

jsou dostupné na: http://qucs.sourceforge.net/download.html#example (anglicky)

[5] Qucs release 0.0.18, or greater.