

---

# **Qucs Help Documentation**

***Release 0.0.19***

**Qucs Team (2014)**

**jul 25, 2018**



<b>1</b>	<b>Background</b>	<b>3</b>
<b>2</b>	<b>Getting Started with Qucs Analogue Circuit Simulation</b>	<b>5</b>
<b>3</b>	<b>Começando com Otimizações</b>	<b>11</b>
<b>4</b>	<b>Começando com o Octave Scripts</b>	<b>23</b>
<b>5</b>	<b>Descrição Rápida das Ações</b>	<b>25</b>
<b>6</b>	<b>Trabalhando com Subcircuitos</b>	<b>29</b>
<b>7</b>	<b>Começando com Simulações Digitais</b>	<b>33</b>
<b>8</b>	<b>Descrição Rápida das Funções Matemáticas</b>	<b>35</b>
<b>9</b>	<b>Lista de Caracteres Especiais</b>	<b>43</b>
<b>10</b>	<b>Circuitos de Casamento</b>	<b>47</b>
<b>11</b>	<b>Arquivos Instalados</b>	<b>49</b>
<b>12</b>	<b>Formato dos Arquivos de Diagrama Elétrico</b>	<b>51</b>
<b>13</b>	<b>Subcircuit and Verilog-A RF Circuit Models for Axial and Surface Mounted Resistors</b>	<b>55</b>





Conteúdo:



---

## 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 whose 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 understanding 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.

---

## 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 its 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.

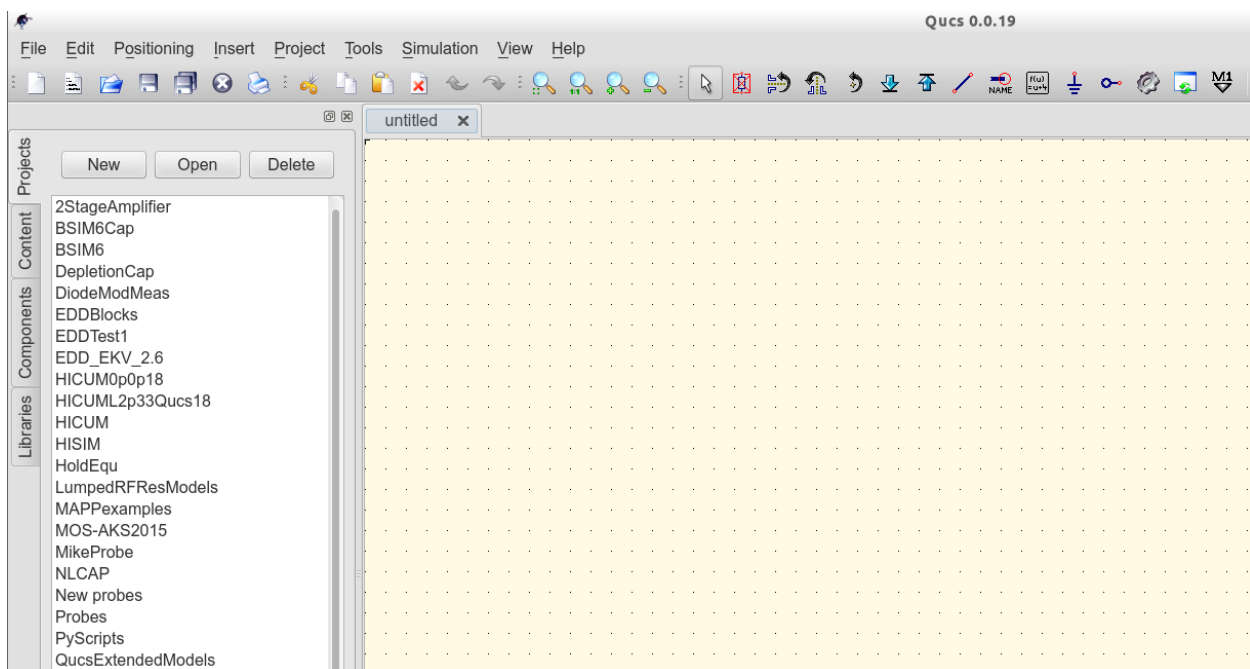


Figura 1 - Janela principal do Qucs

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.

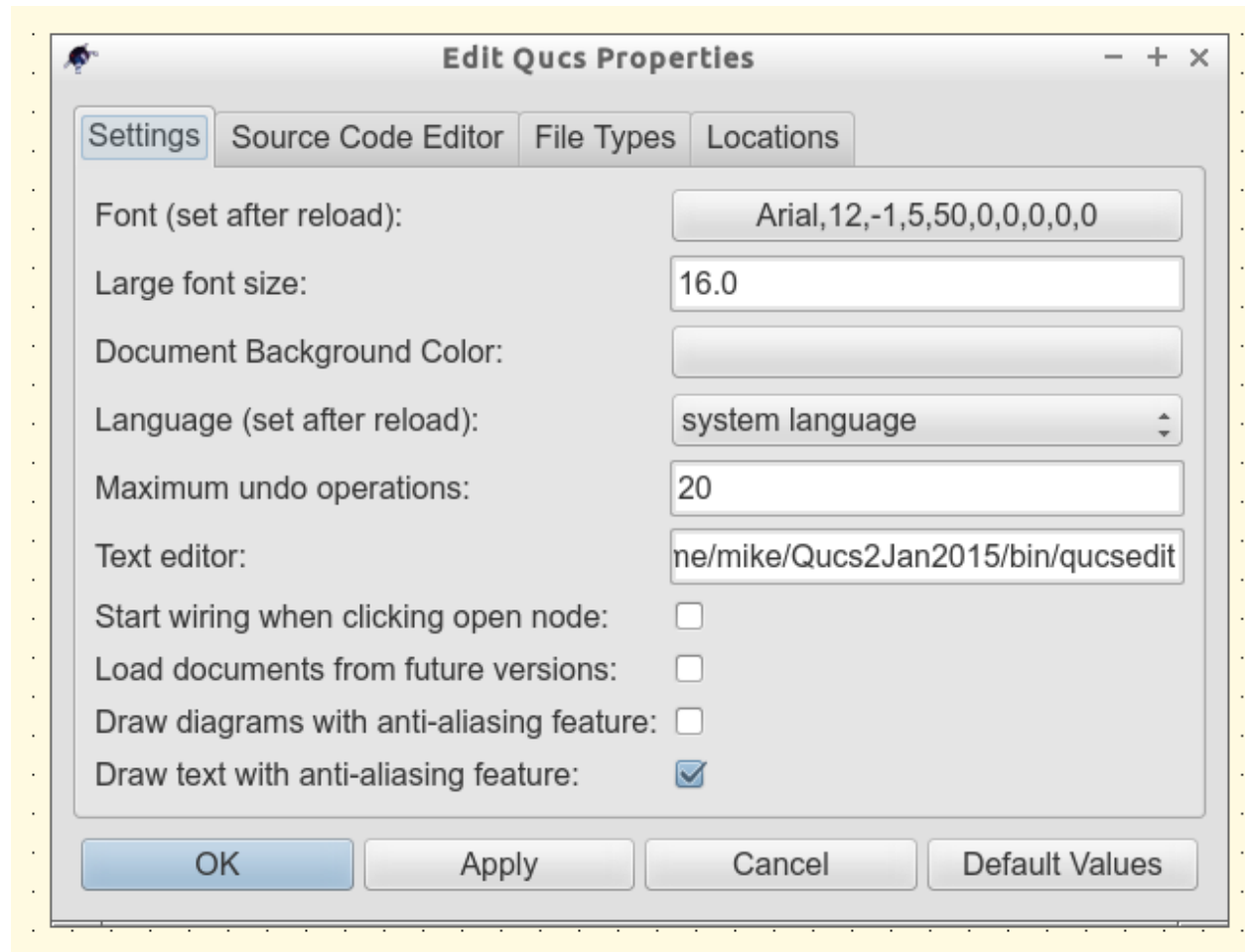


Figure 2 - QucsEditProperties window

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 numerical 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.

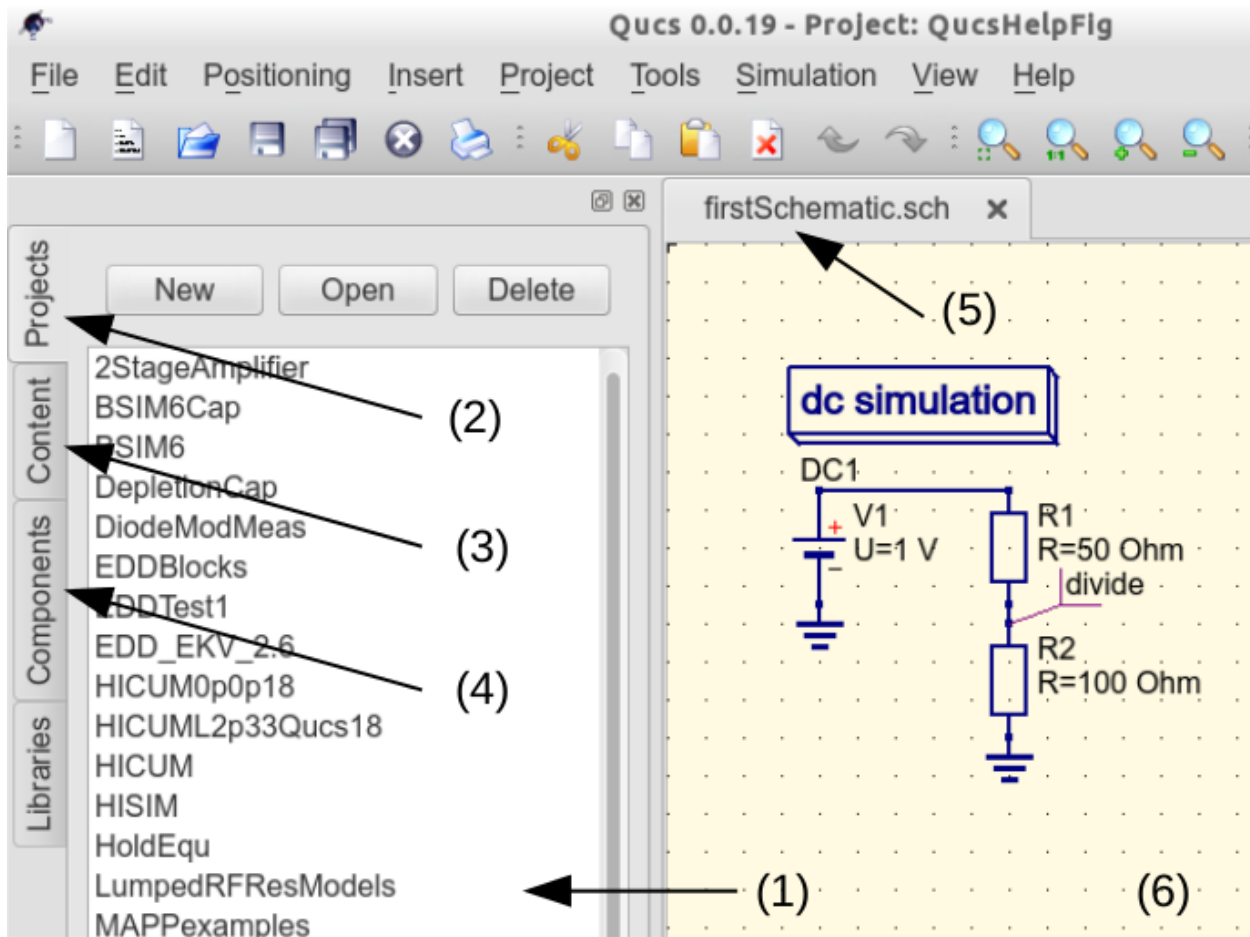


Figure 3 - Qucs main window with working areas labelled

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 its 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 chosen 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, whose 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.



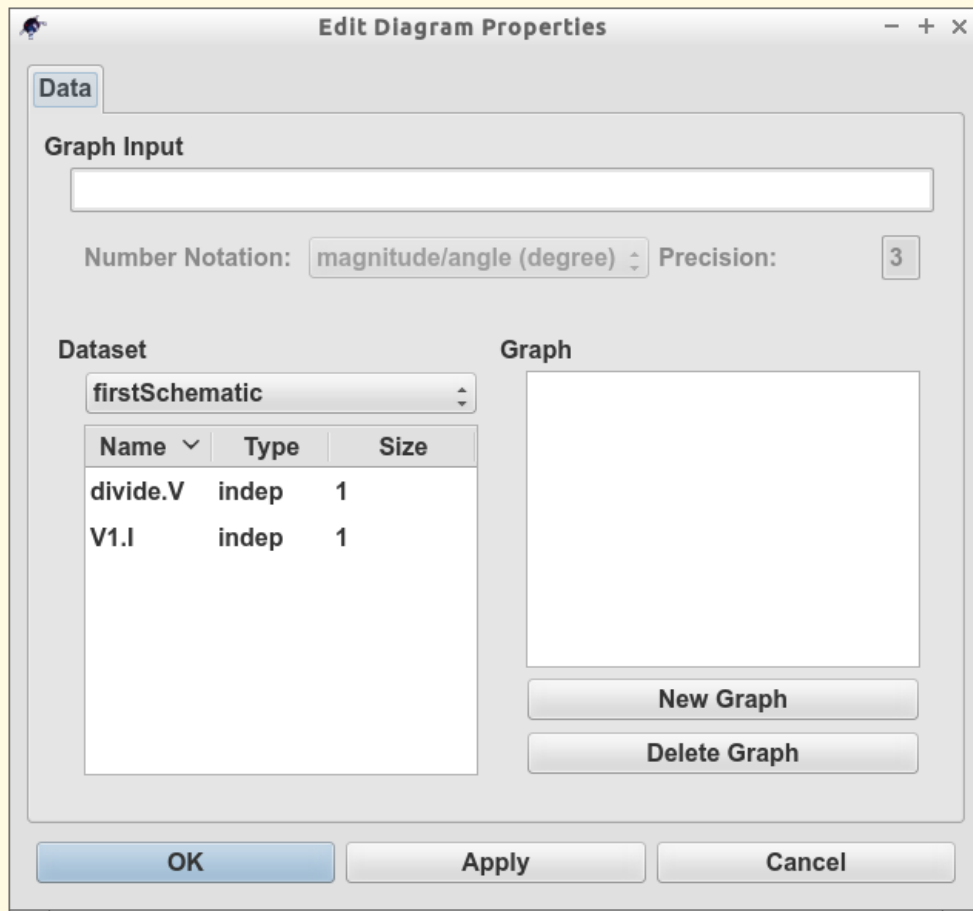


Figure 4 - Qucs data display window showing a **Tabular** dialogue



---

### Começando com Otimizações

---

Para otimizar circuitos, Qucs usa a ferramenta ASCO (<http://asco.sourceforge.net/>). Uma breve descrição de como preparar seu esquema elétrico, executar e interpretar os resultados são dados abaixo. Antes de usar esta funcionalidade, ASCO deve estar instalado no computador.

Otimização de um circuito não é nada mais do que a minimização da função custo. Ela poderia ser o atraso no tempo de subida de um circuito digital, ou a potência ou ganho de um circuito analógico. Outra possibilidade é definindo o problema de otimização como sendo uma composição de funções, conduzindo neste caso para a definição de figura-de-mérito.

Para configurar uma netlist para otimização, duas coisas devem ser adicionadas a já existente netlist: inserir a(s) equação(ões) e o bloco do componente otimização. Tomando o esquema elétrico da Figura 1 e mudando ele até que se obtenha o esquema elétrico exibido na Figura 2.

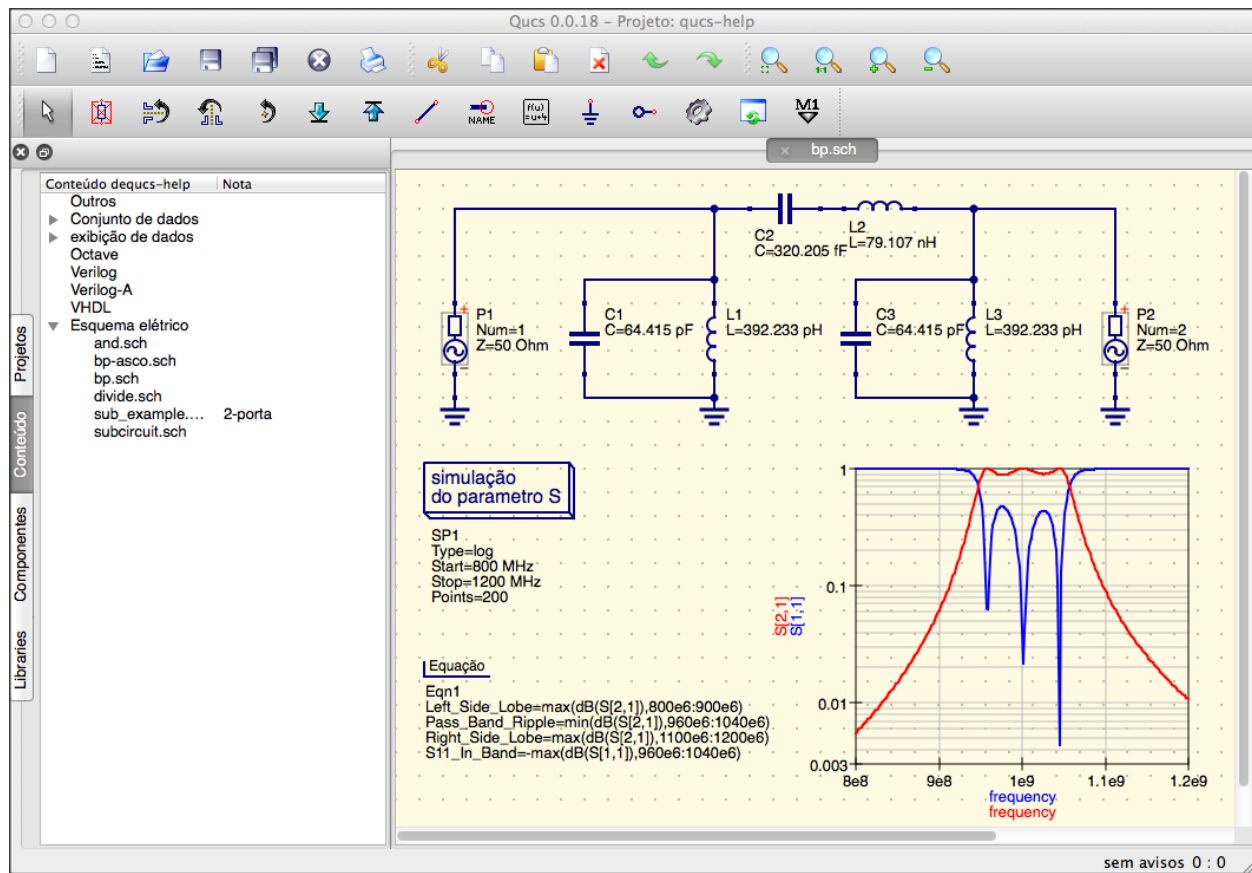


Figura 1 - Esquema elétrico inicial.

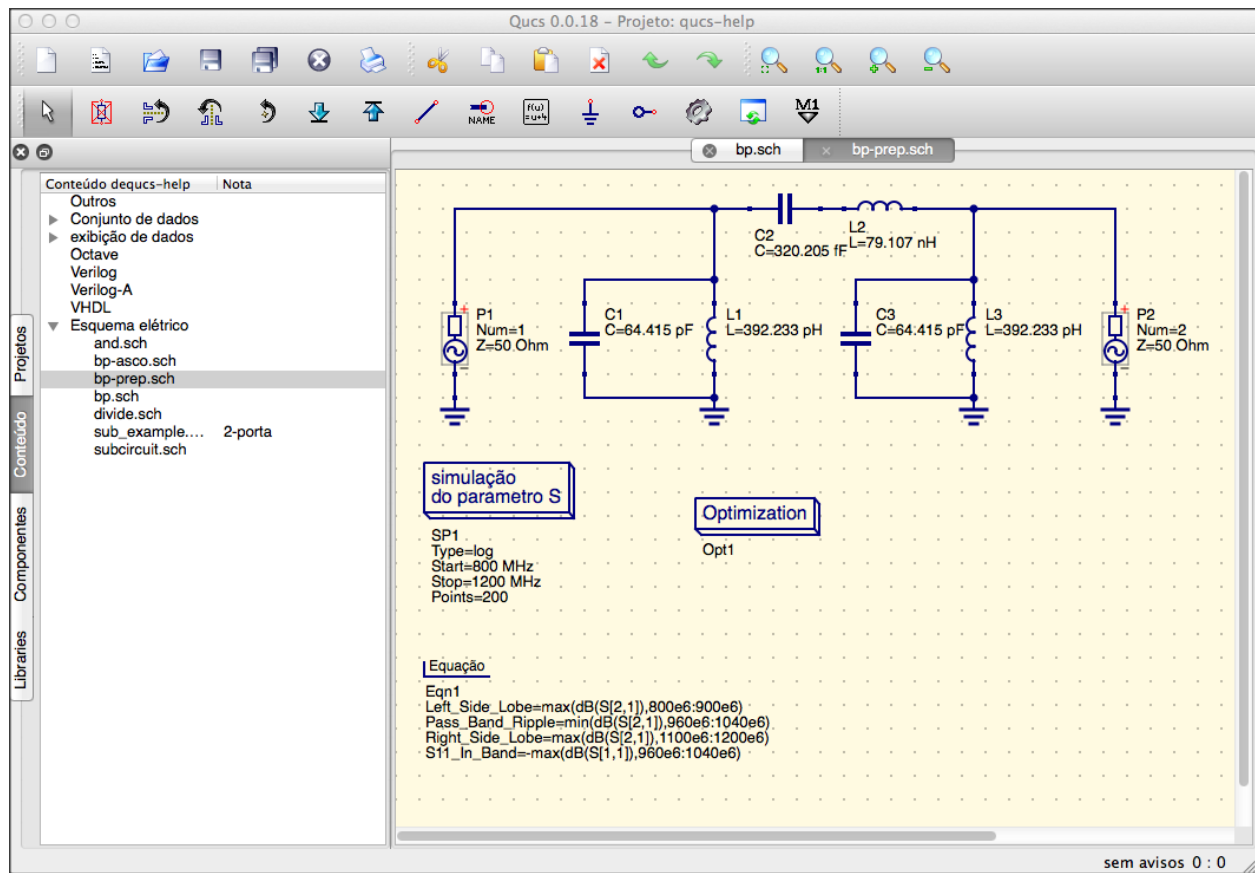


Figura 2 - Esquema elétrico preparado.

Agora, abra o componente otimização e selecione a aba Algoritmo. Dos parâmetros existentes, dê atenção especial a 'Número máximo de Iterações', 'Constante F' e 'Fator de cruzamento pelo valor'. Pelo valor- ou subestimação pode levar a uma convergência prematura do otimizador para um mínimo local ou, um tempo muito longo de otimização.

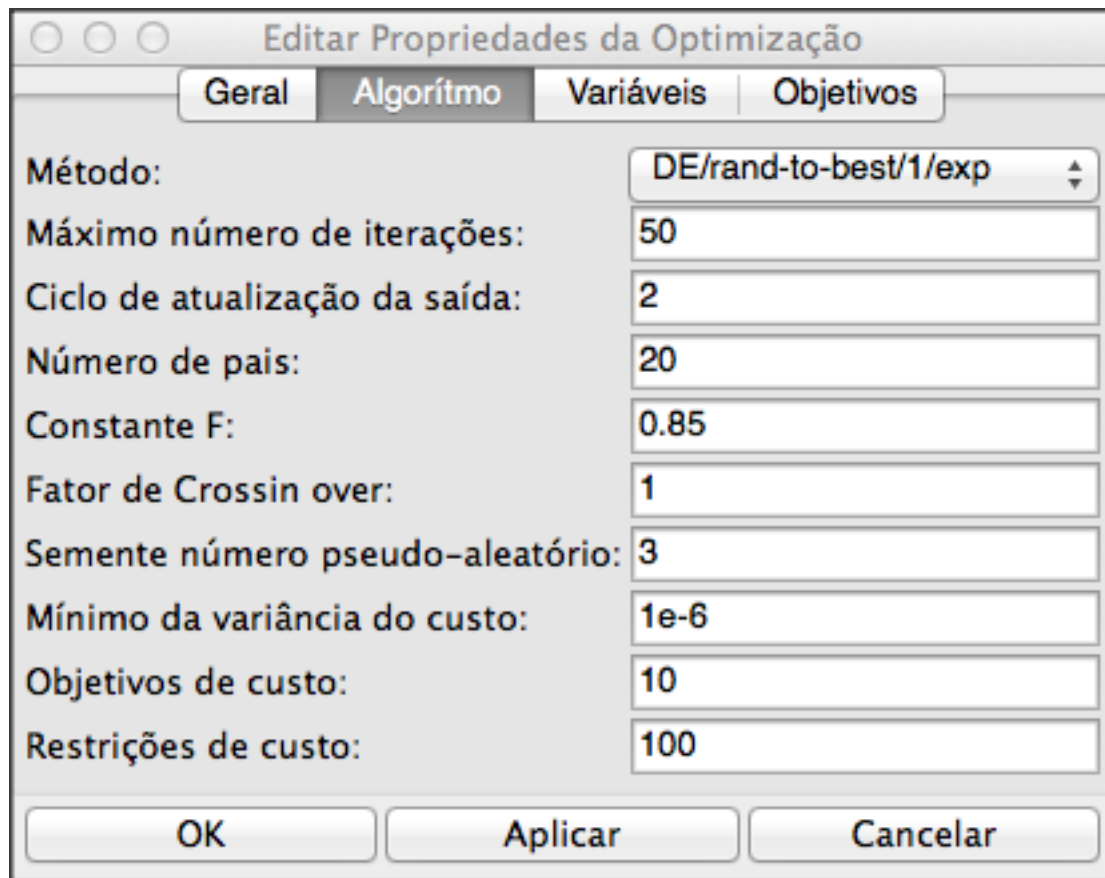


Figura 3 - Janela de otimização, opção Algoritmo.

Na aba Variáveis, definimos quais elementos do circuito serão escolhidos e suas faixas de variação, como mostrado na Figura 4. Os nomes das variáveis correspondem aos local do identificadores nas propriedades e **não** ao nome dos componentes.

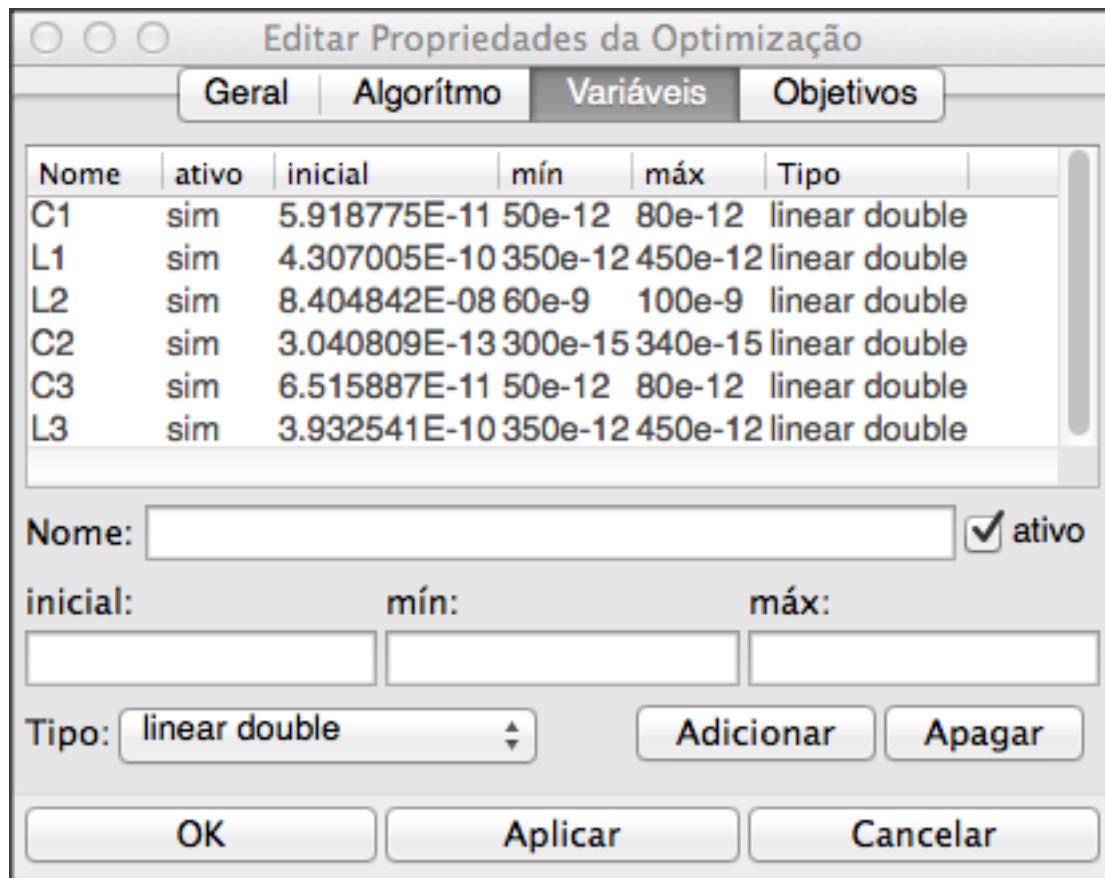


Figura 4 - Janela de otimização, opção Variáveis.

Finalmente, vá para a aba Metas onde os objetivos de otimização (maximizar, minimizar) e restrições (menor, maior, igual) são definidos. ASCO automaticamente os combinará em uma única função custo que será minimizada.

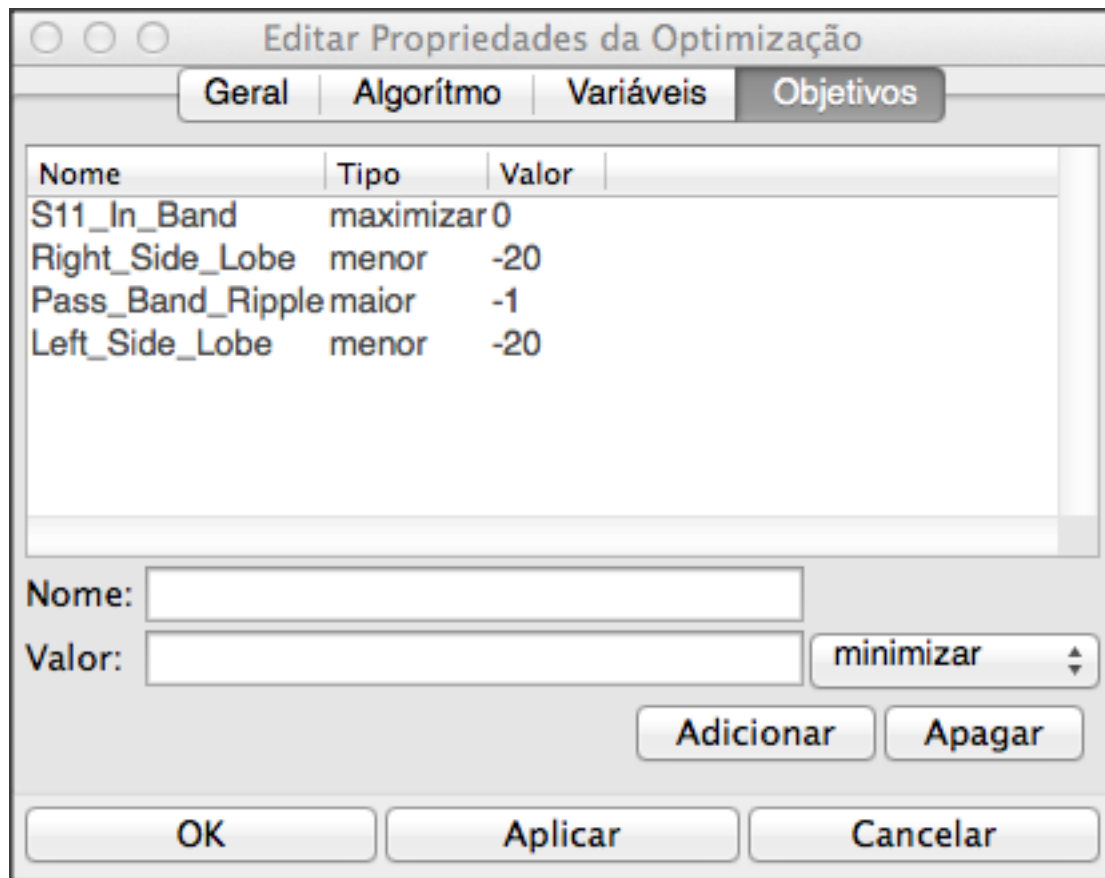


Figura 5 - Janela otimização, opção Metas.

O próximo passo é mudar o esquema elétrico, e definir quais elementos do circuito serão otimizados. O esquema elétrico resultante é mostrado na Figura 6.



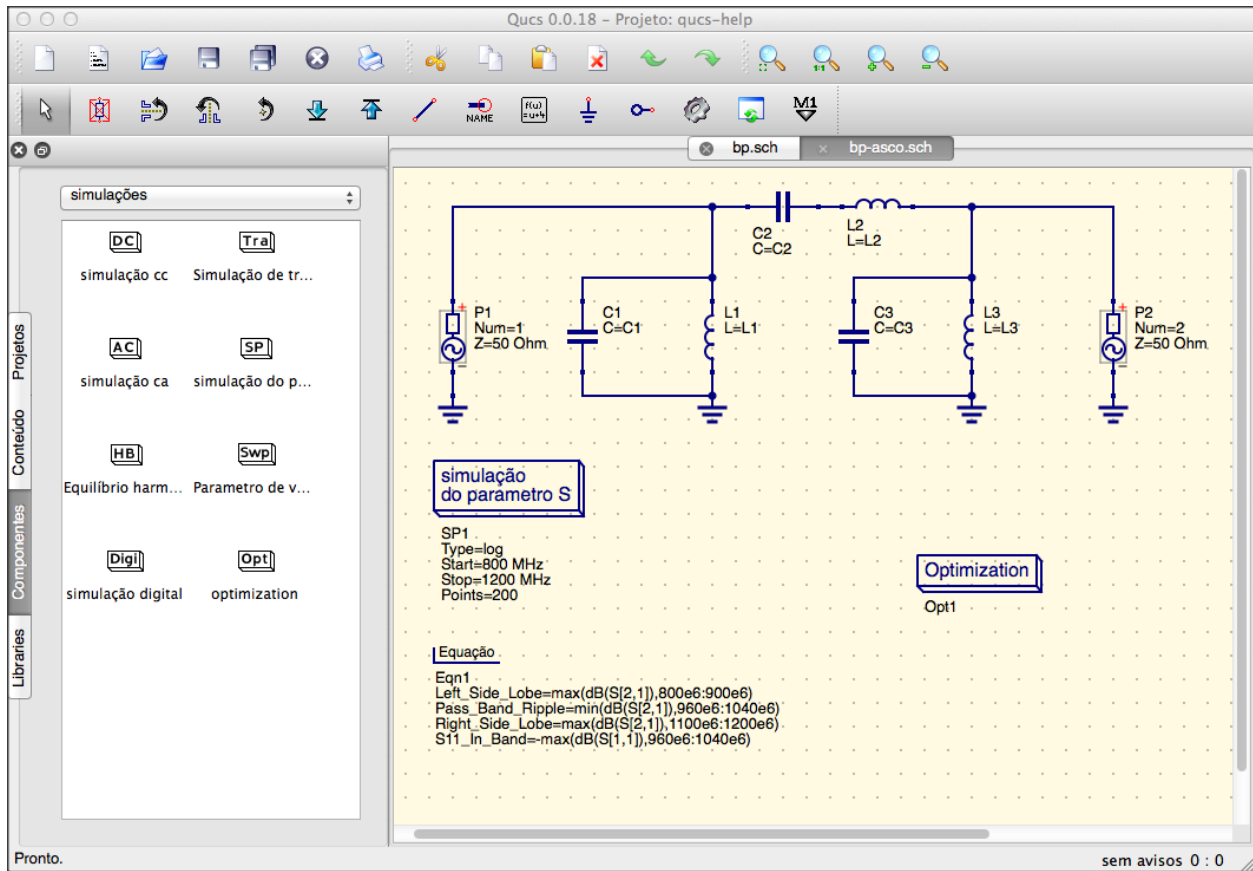


Figura 6 - Nova janela principal do Qucs.

O último passo é executar a otimização, isto é, executar a simulação pressionando F2. Quando finalizado, que demora poucos segundos em um computador moderno, os melhores resultados da simulação são exibidos em um visualizador de formas de onda gráfico.

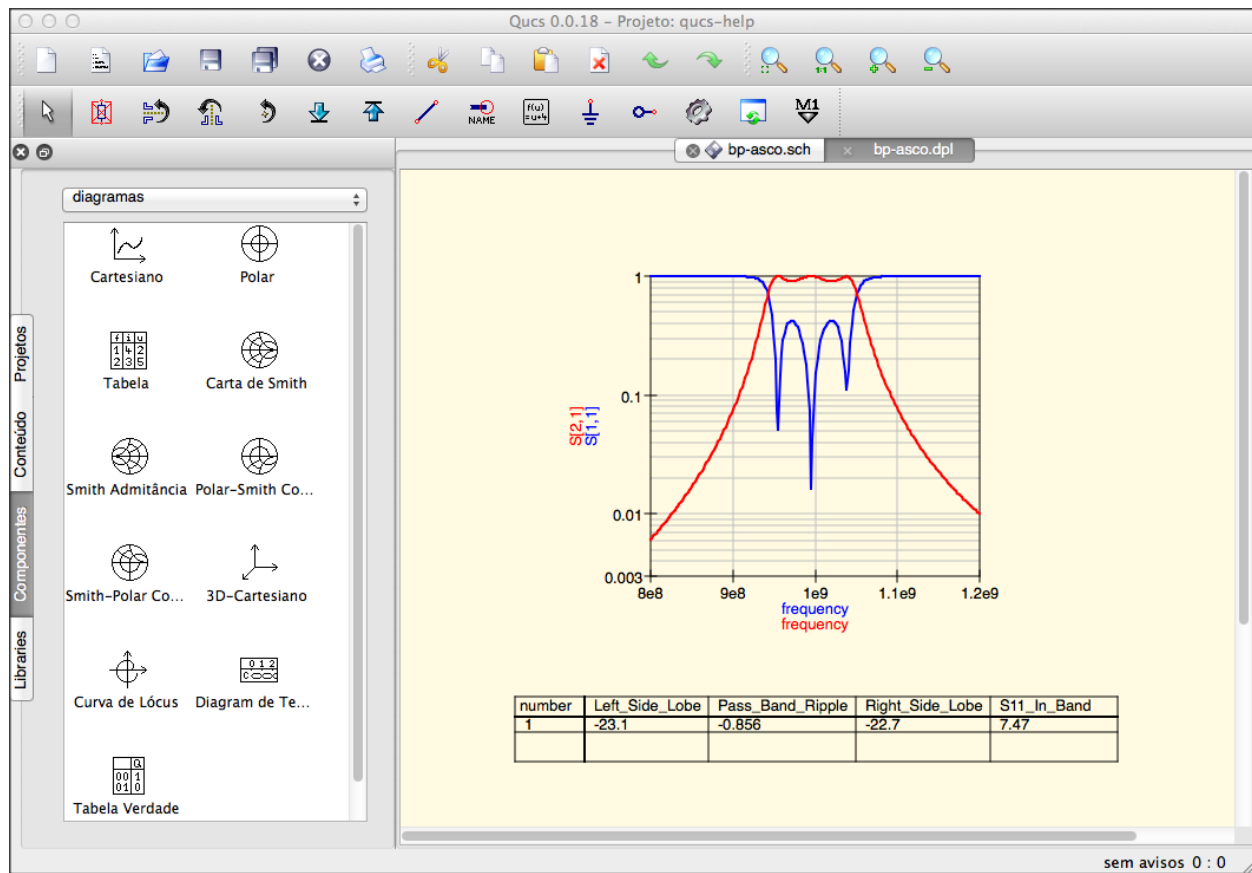


Figura 7 - Janela de resultados do Qucs.

Os melhores valores para o circuito encontrados podem ser encontrados na janela de otimização, na aba Variáveis. Agora eles são os valores iniciais para cada uma das variáveis introduzidas (Figura 8).

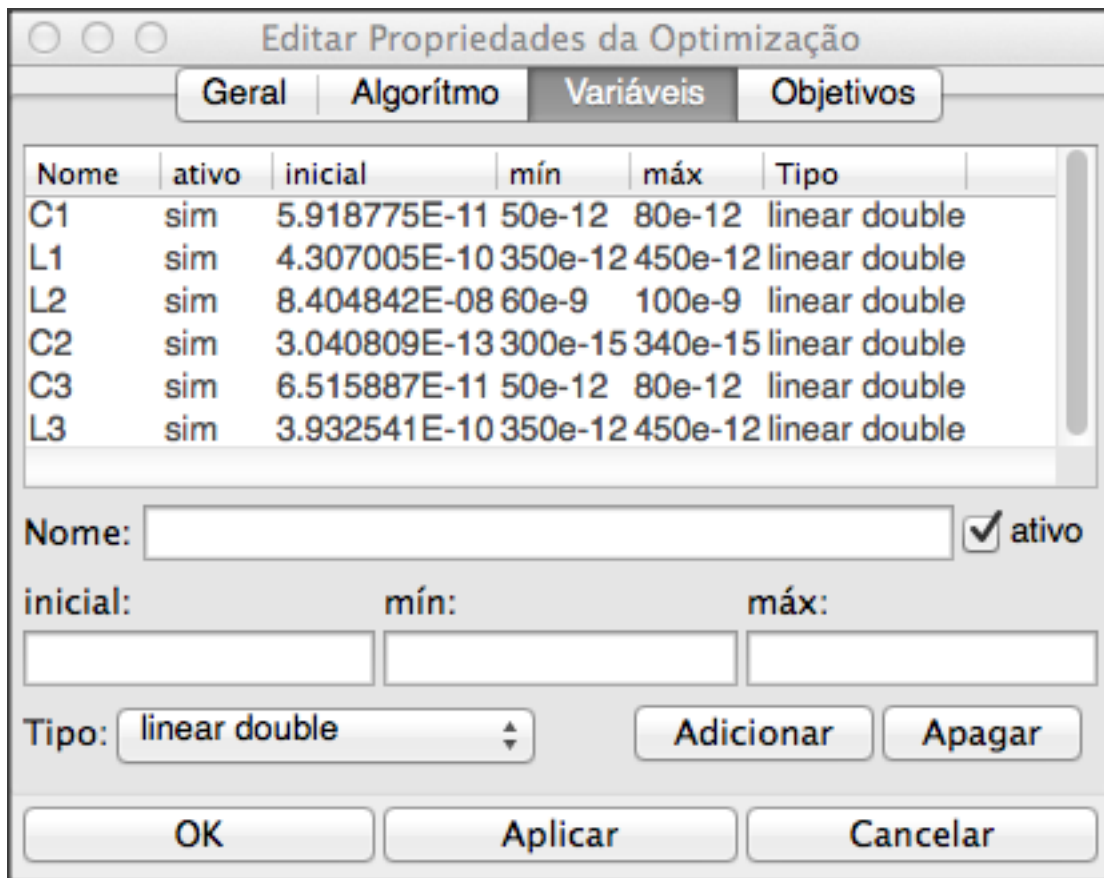


Figura 8 - Os melhores valores encontrados para o circuito.

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.

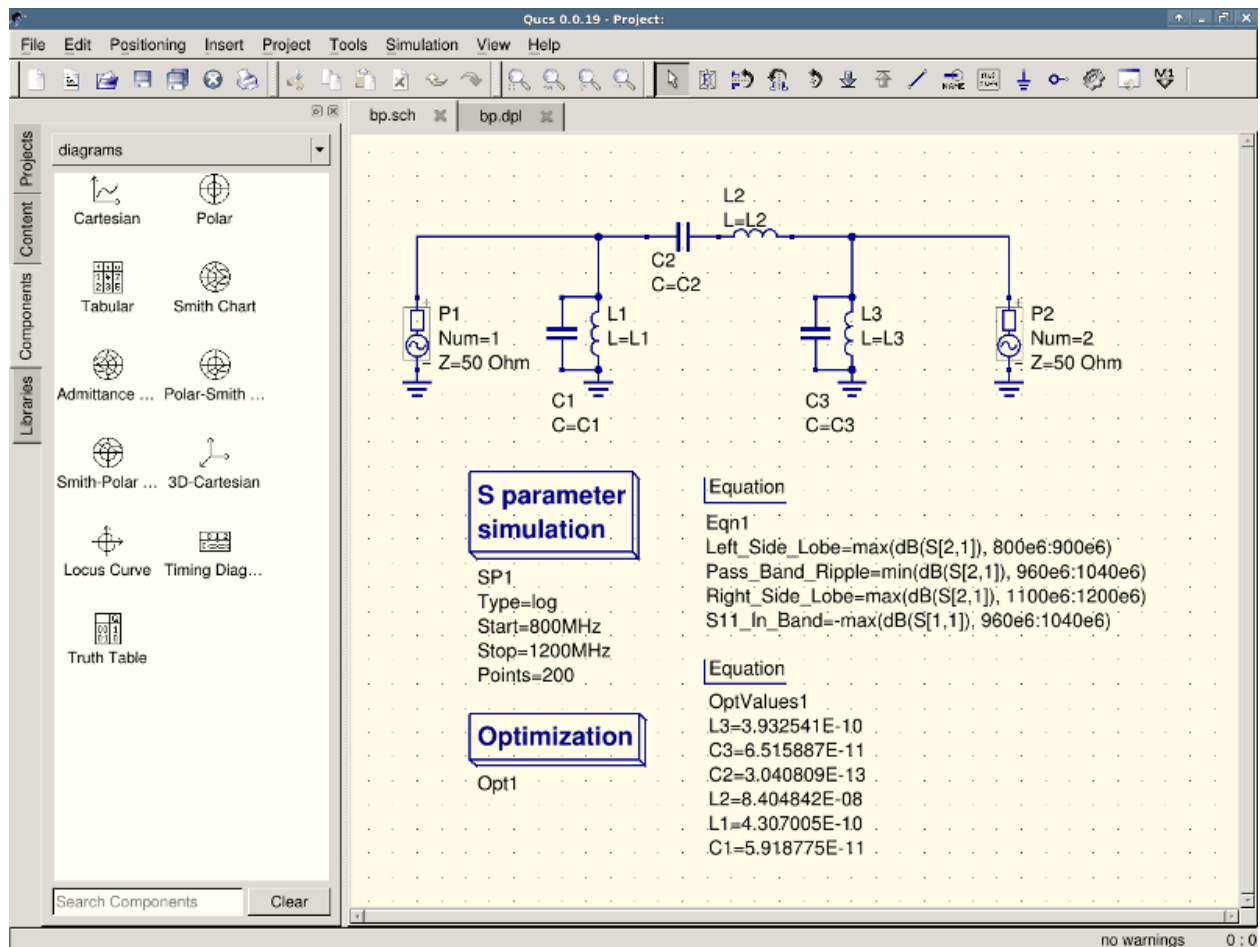


Figure 9 - Schematic with optimized values.

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.

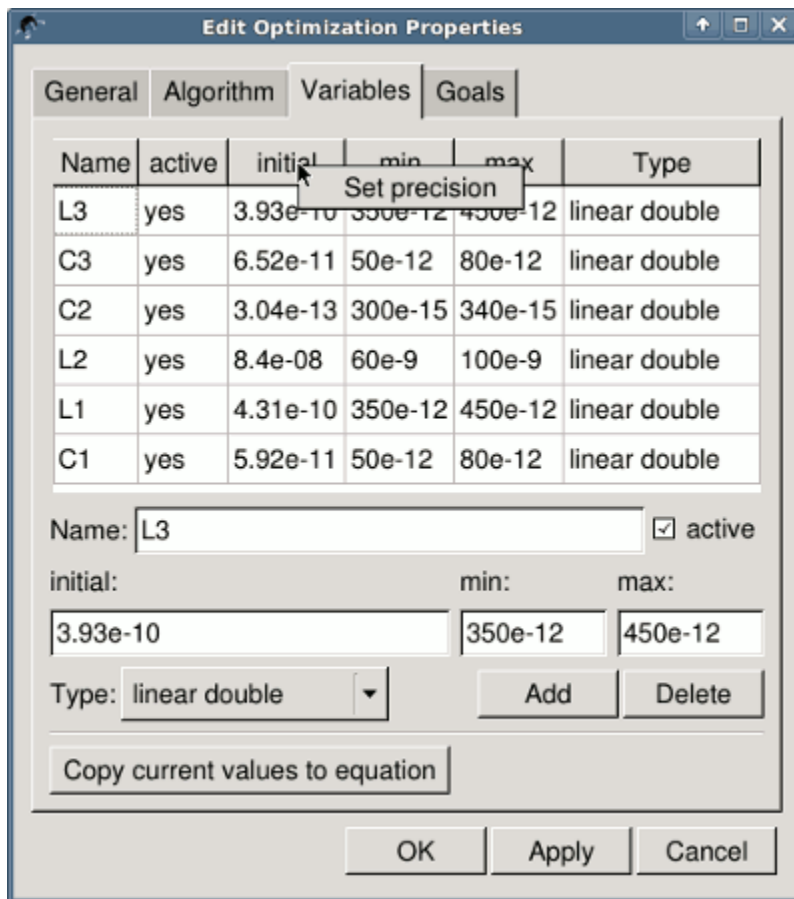


Figure 10 - Changing the displayed variables precision.



---

### Começando com o Octave Scripts

---

Qucs também pode ser usado para desenvolver scripts com Octave (consulte '<http://www.octave.org> < <http://www.octave.org> >' \_). Este documento deve te dar uma breve descrição sobre como fazer isso.

Se o usuário cria um novo documento de texto e salva com a extensão do Octave, por exemplo, 'name.m', em seguida, o arquivo será listado em arquivos do Octave no projeto ativo. O script pode ser executado com a tecla F2 ou pressionando o botão simular na barra de ferramentas. A saída pode ser vista na janela Octave que abre automaticamente (por padrão no lado direito). Na parte inferior da janela do Octave, há uma linha de comando, onde o usuário pode digitar comandos simples. Ele tem uma função de história que pode ser usada com o cursor para cima/baixo teclas.

Existem duas funções do Octave que carregar resultados de simulação realizada com Qucs a partir de um arquivo de conjunto de dados: `loadQucsVariable()` e `loadQucsDataset()`. Por favor, use a função de ajuda na linha de comando Octave para aprender mais sobre elas (ou seja, digite '`help loadQucsVariable`' e '`help loadQucsDataset`').

#### 4.1 Pós-processamento

Octave também pode ser usado para pós-processamento automático de um resultado de simulação de Qucs. Isso é feito editando o arquivo de dados de exibição de um esquema (Configurações do Documento... no menu Arquivo). Se o nome do arquivo de script Octave (extensão de nome de arquivo .m) do mesmo projeto é inserido, esse script será executado após a simulação é terminada.





---

Descrição Rápida das Ações

---

## 5.1 Ações Gerais

(validas em todos os modos)

roda do mouse	Desloca verticalmente a área desenhada. Você pode também deslocar fora do tamanho atual.
roda do mouse + Botão Shift	Desloca horizontalmente a área desenhada. Você pode também deslocar fora do tamanho atual.
roda do mouse + Botão Ctrl	Aproxima ou afasta a área de desenho.
arastar e soltar um arquivo dentro da área do documento	Tenta abrir um arquivo como um esquema elétrico ou exibição de dados do Qucs.

## 5.2 Modo “Selecionar”



(Menu: Editar->Selecionar)

botão esquerdo do mouse	Seleciona o elemento abaixo do cursor do mouse. Se vários componentes estiverem colocados lá, você pode clicar várias vezes para selecionar o componente desejado. Mantendo o botão do mouse pressionado, você pode mover o componente abaixo do cursor do mouse e todos os outros selecionados. Se você quiser um ajuste fino na posição do componente, pressione a tecla CTRL durante o movimento e a grade será desativada. Mantendo o botão do mouse pressionado sem nenhum elemento por baixo, com seu movimento, aparecerá um retângulo. Após soltar o botão do mouse, todos os elementos dentro deste retângulo serão selecionados. Um diagrama selecionado ou desenhado pode ser redimensionado pressionando o botão esquerdo do mouse sobre uma de suas bordas e movido mantendo o botão pressionado. Após clicar em um componente de texto, ele pode ser editado diretamente. A tecla ENTER pula para a próxima propriedade. Se a propriedade for uma lista selecionada, ela poderá ser mudada somente com as teclas de cursor cima/baixo. Clicando em um nó do circuito, entrará em “modo fio”.
botão esquerdo do mouse + Botão Ctrl	Permite que mais de um elemento seja selecionado, isto é, selecionar um elemento não desativa a seleção dos outros. Clicando em um elemento selecionado, desativa a seleção dele. Este modo também é válido para seleção feita com retângulo (veja item anterior).
botão direito do mouse	Clicando em um fio, apenas a linha reta é selecionada ao invés da linha inteira.
clique duplo botão direito do mouse	Abre uma janela para editar as propriedades do elemento (O rótulo dos fios, o parâmetro dos componentes, etc.).

## 5.3 Modo “Inserir Componente”

(Clicar em um componente/diagrama na área esquerda)

botão esquerdo do mouse	Coloca uma nova instância do componente no esquema elétrico.
botão direito do mouse	Rotaciona o componente. (Sem efeito em diagramas.)

## 5.4 Modo “fio”



(Menu: Inserir->Fio)

botão esquerdo do mouse	Define o ponto de início/fim do fio.
botão direito do mouse	Muda a direção do canto do fio (primeiro esquerdo/direito ou primeiro cima/baixo).
clique duplo botão direito do mouse	Termina um fio sem que seja feita uma conexão com outro fio ou porta.

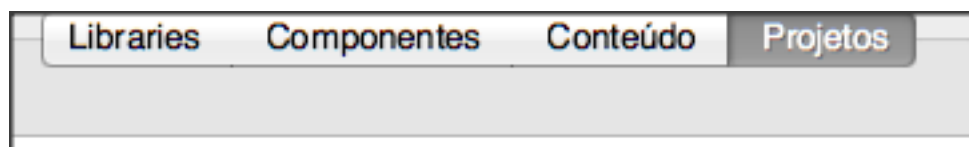
## 5.5 Modo “Colar”



(Menu: Editar->Colar)

botão esquerdo do mouse	Coloca os componentes no esquema elétrico (da área de transferência).
botão direito do mouse	Rotaciona os elementos.

## 5.6 Mouse na aba “Conteúdo”



botão esquerdo do mouse	Seleciona um arquivo.	
clique duplo com botão esquerdo do mouse	Abre arquivo selecionado.	
botão direito do mouse	Exibe menu com: “abrir”	<ul style="list-style-type: none"> <li>• abre arquivo selecionado</li> </ul>
	renomear	<ul style="list-style-type: none"> <li>• muda o nome do arquivo selecionado</li> </ul>
	excluir	<ul style="list-style-type: none"> <li>• apaga o arquivo selecionado</li> </ul>
	excluir grupo	<ul style="list-style-type: none"> <li>• apaga o arquivo selecionado e seus arquivos relacionados (esquema elétrico, exibição de dados, série de dados)</li> </ul>

## 5.7 Teclado

Muitas ações podem ser ativadas/terminadas pelo teclado. This can be seen in the main menu right beside the command. Some further key commands are shown in the following list:

“Delete” ou “Backspace”	Apaga os elementos selecionados ou entra em modo excluir se nenhum elemento for selecionado.
Cursos es-querda/direita	Muda a posição dos marcadores selecionados em gráficos. Se nenhum marcador for selecionado, move o elemento selecionado. Se nenhum elemento for selecionado, Desloca a área do documento.
Cursos para cima/ para baixo	Muda a posição de marcadores selecionados em gráficos multi-dimensionais. Se nenhum marcador for selecionado, move o elemento selecionado. Se nenhum elemento for selecionado, Desloca a área do documento.
Tabulador	Muda para o próximo documento aberto (de acordo com a barra de tabulação acima).

---

### Trabalhando com Subcircuitos

---

Subcircuitos são utilizados para trazer maior clareza ao esquema elétrico. É muito proveitoso em grandes circuitos ou em circuitos em que um bloco de componentes aparece várias vezes.

No Qucs, cada esquema elétrico contendo uma conexão para um subcircuito é um subcircuito. Você pode conseguir uma conexão para um subcircuito usando a barra de ferramentas, a listagem de componentes (nos “componentes soltos”) ou pelo Menu (Inserir->Inserir Conexão). Após colocar todas as conexões a subcircuitos (dois por exemplo) você precisará salvar o esquema elétrico (e.x. CTRL-S). Dando uma olhada no conteúdo da lista de visualização (figura 1), você agora verá um “2-portas” do lado direito do nome do esquema elétrico (coluna “Nota”). Esta Nota marca todos os documentos que são subcircuitos. Agora mude para o diagrama elétrico que você quer usar o subcircuito. Então clique no nome do subcircuito (conteúdo da lista de visualização). Movendo o mouse para dentro da área do documento, você verá que agora é possível colocar o subcircuito dentro do circuito principal. Faça então e complete o esquema elétrico. Você agora pode executar uma simulação. O resultado é o mesmo que se todos os componentes do subcircuito fossem colocados diretamente no circuito.

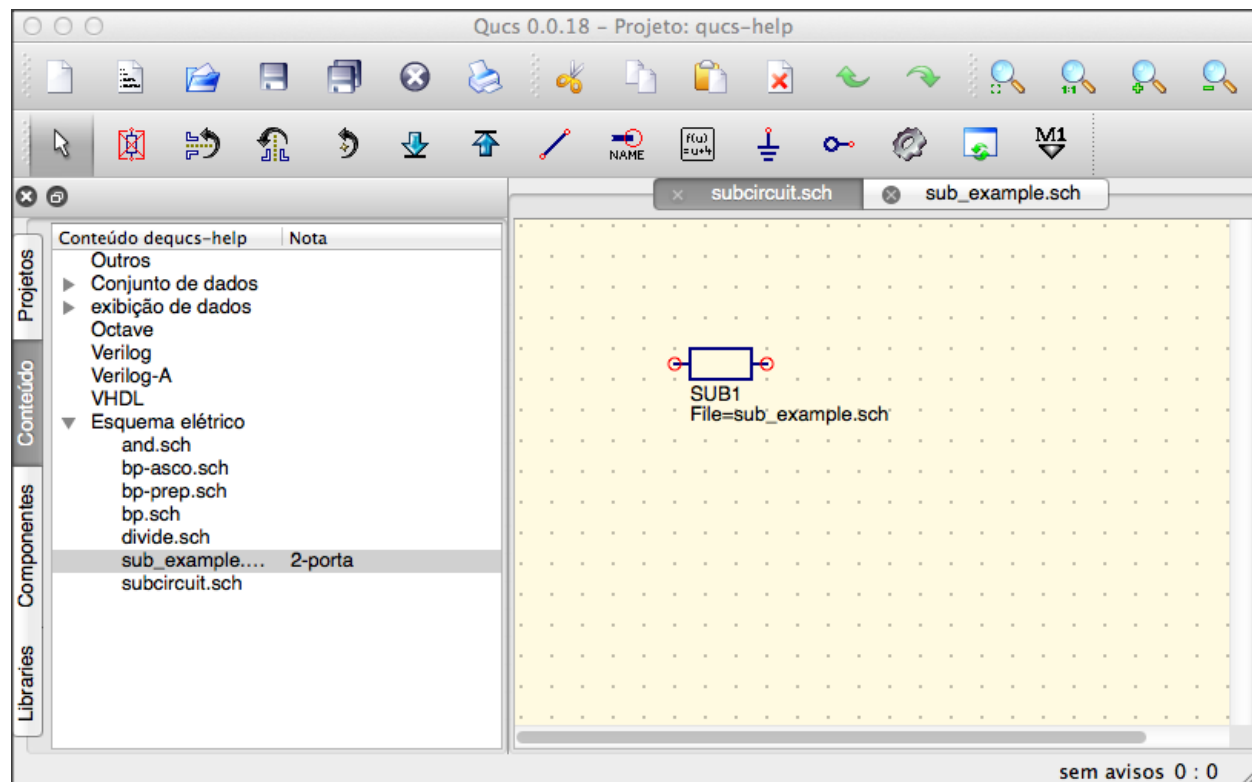


Figure 1 - Accessing a subcircuit

Se você selecionar um componente subcircuito (clique no seu símbolo no esquema elétrico) você poderá visualizar o esquema elétrico do seu subcircuito pressionando CTRL-I (claro, esta função pode ser alcançada pela barra de tarefas e via Menu). Você pode voltar pressionando CTRL-H.

Se você não gostar do símbolo de um componente subcircuito, você poderá desenhar seu próprio símbolo e colocar o texto do componente na posição favorita. Apenas abra seu esquema elétrico do subcircuito e vá no Menu: Arquivo->Editar Símbolo do Circuito. Se você nunca desenhou um símbolo para este componente, um símbolo simples é criado automaticamente. Você agora pode editar este símbolo desenhando linhas e arcos. Quando terminar, salve-o. Agora coloque-o em outro esquema elétrico, e você terá um novo símbolo.

Como todos os outros componentes, subcircuitos podem ter parâmetros. Para criar seus próprios parâmetros, volte para o editor onde você editou o símbolo do circuito e dê um duplo clique nos parâmetros de texto do subcircuito. Uma janela aparecerá e você poderá preencher os parâmetros com valores padrões e descrições. Quando estiver pronto, feche a janela e salve o subcircuito. Em cada esquema elétrico que o subcircuito for colocado, ele possuirá novos parâmetros que poderão ser editados como em todos os outros componentes.

## 6.1 Subcircuitos com parâmetros

Um exemplo simples usando subcircuitos com parâmetros e equações é fornecido aqui.

Criar um sub-circuito:

- Crie um novo projeto
- Nova esquema (para sub-circuito)
- Adicionar um resistor, indutor e capacitor, conectá-los em série, adicionar duas portas
- Salve o sub-circuito como RLC.sch

- Dar valor de resistor como 'R1'
- Adicionar equação 'ind = L1',
- Dar valor do indutor como 'ind'
- Dar o valor do capacitor como 'C1'
- Salvar
- Arquivo > Editar Símbolo do Circuito
- Duplo clique sobre o ' SUB arquivo = nome ' etiqueta sob a caixa retangular
  - Adicionar nome = R1, valor padrão = 1
  - Adicionar nome = L1, valor padrão = 1
  - Adicionar nome = C1, valor padrão = 1
  - OK

Inserir subcircuito e definir os parâmetros:

- Nova esquema (para testbench)
- Salve Test\_RLC.sch
- Projeto conteúdo > escolher e colocar o sub-circuito RLC acima
- Adicionar fonte de tensão CA (V1) e solo
- Adicionar simulação de AC, de 140Hz a 180Hz, 201 pontos
- Defina no símbolo do sub-circuito
  - R1 = 1
  - L1 = 100e-3
  - C1 = 10e-6
- Simular
- Adicionar um diagrama cartesiano, plote V1.i
- O resultado deve ser a ressonância do circuito RLC.
- Os parâmetros do subcircuito RLC podem ser alterados na parte superior esquemática.





---

## Começando com Simulações Digitais

---

Qucs também é uma interface gráfica para executar simulações digitais. Este documento irá lhe dar uma breve descrição de como usá-la.

Para simulações digitais Qucs utiliza o programa FreeHDL (<http://www.freehdl.seul.org>). Assim, o pacote FreeHDL como também o compilador GNU C++ devem estar instalados no computador.

Não há grandes diferenças em se executar uma simulação analógica ou uma digital. Tendo lido [Começando com Simulações Analógicas](#), fica fácil ter uma simulação digital funcionando. Vamos calcular a tabela verdade de uma simples porta lógica AND. Selecione a classe “componentes digitais” da aba componentes no lado esquerdo e monte o circuito mostrado na figura 1. O bloco de simulação digital pode ser encontrado entre os outros blocos de simulação. As fontes digitais *S1* e *S2* são as entradas, o nó rotulado como *Output* é a saída. Após executar a simulação, a página de exibição de dados abrirá. Coloque o diagrama *tabela verdade* nela e adicione a variável *Output*. Agora, a tabela verdade de uma porta AND de duas entradas é mostrado. Parabéns, a primeira simulação digital terminou!

As fontes digitais *S1* e *S2* são as entradas, o nó rotulado como *Output* é a saída. Após executar a simulação, a página de exibição de dados abrirá. Coloque o diagrama *tabela verdade* nela e adicione a variável *Output*. Agora, a tabela verdade de uma porta AND de duas entradas é mostrado. Parabéns, a primeira simulação digital terminou!

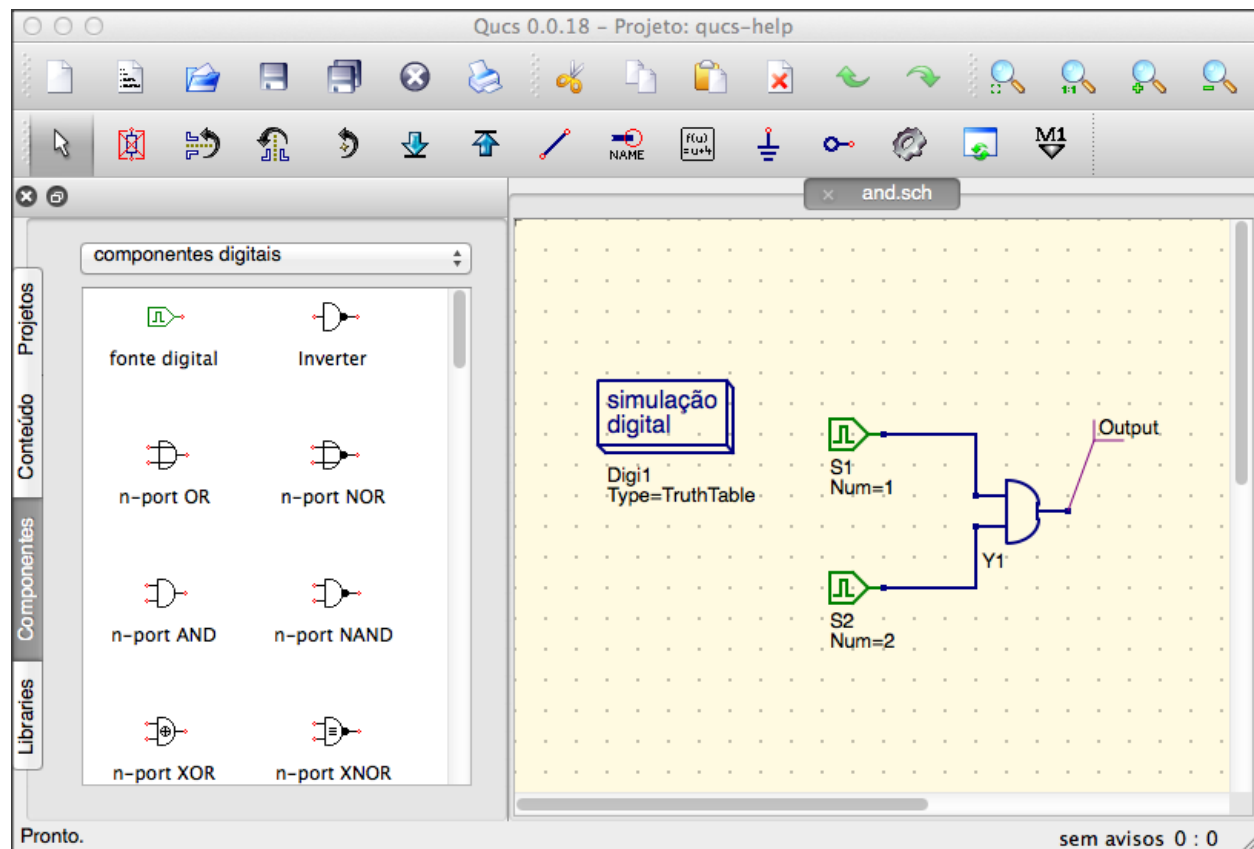


Figura 1 - Janela principal do Qucs

Tabela verdade não é a única simulação digital que o Qucs pode efetuar. Também é possível aplicar um sinal arbitrário ao circuito e visualizar o sinal de saída em um diagrama de tempo. Para fazer desta forma, o parâmetro *Tipo* do bloco de simulação deve ser mudado para *TimeList* e a duração da simulação deve ser inserido no próximo parâmetro. As fontes digitais tem agora um comportamento diferente: Elas podem gerar uma sequência aleatória de bits apenas definindo o primeiro bit (baixo ou alto) e uma lista que define os instantes até a nova mudança de estado. Note que esta lista se repete após o seu fim. Então, para criar um clock de 1GHz com razão de pulsos de 1:1, a lista escreve: 0.5ns; 0.5ns Para exibir os resultados desse tipo de simulação, há o diagrama *diagrama de tempo*. Aqui, os resultados de todas os nós de saídas podem ser exibidos linha por linha em um diagrama. Então, vamos nos divertir. .

Para exibir os resultados desse tipo de simulação, há o diagrama *diagrama de tempo*. Aqui, os resultados de todas os nós de saídas podem ser exibidos linha por linha em um diagrama. Então, vamos nos divertir. .

## 7.1 Componente em arquivo VHDL

Simulações mais complexas e mais universais podem ser executadas usando o componente “arquivo VHDL”. Este componente pode ser encontrado na lista de componentes (seção “componentes digitais”). Todavia, o uso recomendado é o seguinte: O arquivo VHDL deve ser membro do projeto. Então, vá até o visualizador de lista de conteúdo e clique no nome do arquivo. Após entrar na área de esquema elétrico, o componente VHDL pode ser colocado.

O último bloco entidade no arquivo VHDL define a interface, isto é, todas as portas de entrada e saída devem ser declaradas aqui. Essas portas também serão mostradas pelo símbolo no esquema elétrico e podem ser conectadas ao resto do circuito. Durante a simulação, o código fonte do arquivo VHDL é colocado no nível mais alto do arquivo VHDL. Isto deve ser considerado como causa de algumas limitações. Por exemplo, o nome das entidades dentro do arquivo VHDL devem ser diferentes dos nomes já dados para subcircuitos. (Após a simulação, o código fonte completo pode ser visto pressionando-se F6. Use-o para adquirir um sentimento por este tipo de procedimento.)

---

## Descrição Rápida das Funções Matemáticas

---

As seguintes operações e funções podem ser aplicadas nas equações do Qucs. Para descrição detalhada por favor veja em “Measurement Expressions Reference Manual”. Parâmetros entre colchetes “[ ]” são opcionais.

### 8.1 Operadores

#### 8.1.1 Operadores Aritméticos

$+x$	Soma unitária
$-x$	Subtração unitária
$x+y$	Adição
$x-y$	Subtração
$x*y$	Multiplicação
$x/y$	Divisão
$x\%y$	Módulo (resto da divisão)
$x^y$	Potência

## 8.1.2 Operadores Lógicos

<code>! x</code>	Negação
<code>x &amp; y</code>	E
<code>x   y</code>	OU
<code>x ^ y</code>	OU Exclusivo
<code>x ? y : z</code>	Abbreviation for conditional expression - if <code>x</code> then <code>y</code> else <code>z</code>
<code>x == y</code>	Igual
<code>x != y</code>	Não igual
<code>x &lt; y</code>	Menor que
<code>x &lt;= y</code>	Menor ou igual a
<code>x &gt; y</code>	Maior que
<code>x &gt;= y</code>	Maior ou igual a

## 8.2 Funções Matemáticas

### 8.2.1 Vetores e Matrizes: Criação

<code>eye (n)</code>	Cria uma matriz identidade <code>n x n</code>
<code>length (y)</code>	Returns the length of the <code>y</code> vector
<code>linspace (from, to, n)</code>	Real vector with <code>n</code> lin spaced components between <code>from</code> and <code>to</code>
<code>logspace (from, to, n)</code>	Real vector with <code>n</code> log spaced components between <code>from</code> and <code>to</code>

### 8.2.2 Vetores e Matrizes: Funções Matriciais Básicas

<code>adjoint (x)</code>	Matriz adjunta de <code>x</code> (transposta e complexo conjugado)
<code>det (x)</code>	Determinante da matriz <code>x</code>
<code>inverse (x)</code>	Matriz inversa de <code>x</code>
<code>transpose (x)</code>	Matriz transposta de <code>x</code> (linhas e colunas trocadas)

### 8.2.3 Funções Matemáticas Elementares: Funções Básicas Real e Complexa

<code>abs(x)</code>	Valor absoluto, magnitude do número complexo
<code>angle(x)</code>	Fase angular em radianos de um número complexo. Sinônimo para <code>arg()</code>
<code>arg(x)</code>	Fase angular em radianos de um número complexo
<code>conj(x)</code>	Conjugado de um número complexo
<code>deg2rad(x)</code>	Converte fase em graus para radianos
<code>hypot(x, y)</code>	Função distância Euclidiana
<code>imag(x)</code>	Parte imaginária de um número complexo
<code>mag(x)</code>	Magnitude de um número complexo
<code>norm(x)</code>	Quadrado do valor absoluto de um vetor
<code>phase(x)</code>	Fase angular em graus de um número complexo
<code>polar(m, p)</code>	Transform polar coordinates <code>m</code> and <code>p</code> into a complex number
<code>rad2deg(x)</code>	Converte fase em radianos para graus
<code>real(x)</code>	Parte real de um número complexo
<code>sign(x)</code>	Função sinal
<code>sqr(x)</code>	Quadrado (potência de dois) de um número
<code>sqrt(x)</code>	Raiz quadrada
<code>unwrap(p[, tol[, step]])</code>	Unwrap angle <code>p</code> (radians) – defaults <code>step</code> = $2\pi$ , <code>tol</code> = $\pi$

### 8.2.4 Funções Matemáticas Elementares: Funções Exponencial e Logarítmica

<code>exp(x)</code>	Função exponencial para a base $e$
<code>limexp(x)</code>	Função exponencial limitada
<code>log10(x)</code>	Logarítmo na base 10
<code>log2(x)</code>	Logarítmo na base 2
<code>ln(x)</code>	Logarítmo natural (base $e$ )

### 8.2.5 Funções Matemáticas Elementares: Funções Trigonômicas

<code>cos(x)</code>	Função cosseno
<code>cosec(x)</code>	Função cossecante
<code>cot(x)</code>	Função cotangente
<code>sec(x)</code>	Função secante
<code>sin(x)</code>	Função seno
<code>tan(x)</code>	Função tangente

### 8.2.6 Funções Matemáticas Elementares: Funções Trigonômicas Inversas

<code>arccos(x)</code>	Arco cosseno (também conhecida como “cosseno inverso”)
<code>arccosec(x)</code>	Arco cossecante
<code>arccot(x)</code>	Arco cotangente
<code>arcsec(x)</code>	Arco secante
<code>arcsin(x)</code>	Arco seno (também conhecida como “seno inverso”)
<code>arctan(x[, y])</code>	Arco tangente (também conhecida como “tangente inversa”)

### 8.2.7 Funções Matemáticas Elementares: Funções Hiperbólicas

$\cosh(x)$	Cosseno hiperbólico
$\operatorname{cosech}(x)$	Cossecante hiperbólico
$\coth(x)$	Cotangente hiperbólico
$\operatorname{sech}(x)$	Secante hiperbólico
$\sinh(x)$	Seno hiperbólico
$\tanh(x)$	Tangente hiperbólico

### 8.2.8 Funções Matemáticas Elementares: Funções Hiperbólicas Inversas

$\operatorname{arcosh}(x)$	Arco cosseno hiperbólico
$\operatorname{arcosech}(x)$	Arco cossecante hiperbólico
$\operatorname{arcoth}(x)$	Arco cotangente hiperbólico
$\operatorname{arsech}(x)$	Arco secante hiperbólico
$\operatorname{arsinh}(x)$	Arco seno hiperbólico
$\operatorname{artanh}(x)$	Arco tangente hiperbólico

### 8.2.9 Funções Matemáticas Elementares: Arredondamento

$\operatorname{ceil}(x)$	Arredondar para o próximo maior inteiro
$\operatorname{fix}(x)$	Truncar casas decimais de um número real
$\operatorname{floor}(x)$	Arredondar para o próximo menor inteiro
$\operatorname{round}(x)$	Arredondar para o inteiro mais próximo

### 8.2.10 Funções Matemáticas Elementares: Funções Matemáticas Especiais

$\operatorname{besseli0}(x)$	Função Bessel modificada de ordem zero
$\operatorname{besselj}(n, x)$	Função Bessel do tipo 1 e n-ésima ordem
$\operatorname{bessely}(n, x)$	Função Bessel do tipo 2 e n-ésima ordem
$\operatorname{erf}(x)$	Função erro
$\operatorname{erfc}(x)$	Função erro complementar
$\operatorname{erfinv}(x)$	Função inversa do erro
$\operatorname{erfcinv}(x)$	Função complementar inversa do erro
$\operatorname{sinc}(x)$	Função Sinc ( $\sin(x)/x$ ou 1 quando $x = 0$ )
$\operatorname{step}(x)$	Função passo

### 8.2.11 Análise de Dados: Estatística Básica

<code>avg(x[, range])</code>	Average of vector <code>x</code> . If range given <code>x</code> must have a single data dependency
<code>cumavg(x)</code>	Média acumulativa dos elementos de um vetor
<code>max(x, y)</code>	Retorna o maior dos valores entre <code>x</code> e <code>y</code>
<code>max(x[, range])</code>	Maximum of vector <code>x</code> . If range given <code>x</code> must have a single data dependency
<code>min(x, y)</code>	Retorna o menor de todos os valores entre <code>x</code> e <code>y</code>
<code>min(x[, range])</code>	Minimum of vector <code>x</code> . If range is given <code>x</code> must have a single data dependency
<code>rms(x)</code>	Raiz quadrada média de um vetor de elementos
<code>runavg(x)</code>	Corrida média dos elementos de um vetor
<code>stddev(x)</code>	Desvio padrão dos elementos de um vetor
<code>variance(x)</code>	Variância dos elementos de um vetor
<code>random()</code>	Número aleatório entre 0.0 e 1.0
<code>srandom(x)</code>	Semente aleatória dada

### 8.2.12 Análise de Dados: Operações Básicas

<code>cumprod(x)</code>	Produto acumulativo dos elementos de um vetor
<code>cumsum(x)</code>	Soma acumulativa dos elementos de um vetor
<code>interpolate(f, x[, n])</code>	Spline interpolation of vector <code>f</code> using <code>n</code> equidistant points of <code>x</code>
<code>prod(x)</code>	Produto dos elementos de um vetor
<code>sum(x)</code>	Soma dos elementos de um vetor
<code>xvalue(f, yval)</code>	Returns <code>x</code> -value nearest to <code>yval</code> in single dependency vector <code>f</code>
<code>yvalue(f, xval)</code>	Returns <code>y</code> -value nearest to <code>xval</code> in single dependency vector <code>f</code>

### 8.2.13 Análise de Dados: Diferenciação e Integração

<code>ddx(expr, var)</code>	Derivada matemática da expressão <code>expr</code> com relação a variável <code>var</code>
<code>diff(y, x[, n])</code>	Differentiate vector <code>y</code> with respect to vector <code>x</code> <code>n</code> times. Defaults to <code>n = 1</code>
<code>integrate(x, h)</code>	Vetor integração <code>x</code> assumindo numericamente uma constante de tamanho passo <code>h</code>

### 8.2.14 Análise de Dados: Processamento de Sinais

<code>dft(x)</code>	Transformada Discreta de Fourier do vetor <code>x</code>
<code>fft(x)</code>	Transformada Rápida de Fourier do vetor <code>x</code>
<code>fftshift(x)</code>	Shuffles the FFT values of vector <code>x</code> to move DC to the center of the vector
<code>Freq2Time(V, f)</code>	Transformada Discreta Inversa de Fourier da função <code>V(f)</code> interpretada fisicamente
<code>idft(x)</code>	Transformada Discreta Inversa de Fourier de um vetor <code>x</code>
<code>ifft(x)</code>	Transformada Rápida Inversa de Fourier do vetor <code>x</code>
<code>kbd(x[, n])</code>	Janela de derivada Kaiser-Bessel
<code>Time2Freq(v, t)</code>	Transformada Discreta de Fourier da função <code>v(t)</code> interpretada fisicamente

## 8.3 Funções Eletrônicas

### 8.3.1 Valor em dB

dB (x)	Valor em dB
dbm (x)	Converte tensão para potência em dBm
dbm2w (x)	Converte potência em Watts para potência em dBm
w2dbm (x)	Converte potência em Watts para potência em dBm
vt (t)	Tensão térmica para uma dada temperatura t em Kelvin

### 8.3.2 VSWR e coeficientes de reflexão

rtoswr (x)	Converte coeficiente de reflexão para voltage standing wave ratio (VSWR)
rtoz (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 Conversões de Matrizes de N-Portas

stos (s, zref[, z0])	Converts S-parameter matrix to S-parameter matrix with a different Z0
stoy (s[, zref])	Converte matriz de parâmetro S para matriz de parâmetro Z
stoz (s[, zref])	Converte matriz de parâmetro S para matriz de parâmetro Y
twoport (m, from, to)	Converts a two-port matrix: from and to are 'Y', 'Z', 'H', 'G', 'A', 'S' and 'T'.
ytos (y[, z0])	Converte matriz de parametro Y para matriz de parametro Z
ytoz (y)	Converte matriz de parametro Z para matriz de parametro S
ztos (z[, z0])	Converte matriz de parametro Z para matriz de parametro Y
ztoy (z)	Converte matriz de parametro Y para matriz de parametro Z

### 8.3.4 Amplificadores

GaCircle (s, Ga[, arcs])	Available power gain Ga circles (source plane)
GpCircle (s, Gp[, arcs])	Operating power gain Gp circles (load plane)
Mu (s)	Fator de estabilidade Mu dos parâmetros S de uma matriz de duas portas
Mu2 (s)	Fator de estabilidade Mu' dos parâmentros S de uma matriz de duas portas
NoiseCircle (Sopt, Fmin, Rn, F[, Arcs])	Noise Figure(s) F circles
PlotVs (data, dep)	Returns data selected from data: dependency dep
Rollet (s)	Círculo de estabilidade no plano de carga
StabCircleL (s[, arcs])	Círculo de estabilidade no plano de fonte
StabCircleS (s[, arcs])	Círculo de estabilidade no plano de fonte
StabFactor (s)	Stability factor of a two-port S-parameter matrix
StabMeasure (s)	Medição de estabilidade B1 dos parâmetros S de uma matriz de duas portas



## 8.4 Nomenclatura

### 8.4.1 Distância

LO:HI	Distância de LO até HI
:HI	No máximo HI
LO:	De LO
:	Sem limitações de distância

### 8.4.2 Matriz completa `M`

M	Matriz completa M
M[2, 3]	Elemento começando na 2a coluna e 3a coluna da matriz M
M[:, 3]	Vetor consistindo da 3a coluna da matriz M

### 8.4.3 Número real

2.5	Número real
1.4+j5.1	Número complexo
[1, 3, 5, 7]	Matriz
[11, 12; 21, 22]	Matriz

### 8.4.4 exa, \* 1e+18

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 Nome dos Valores

$S[1,1]$	Valor do parâmetro $S$
<i>nodename.V</i>	Tensão DC no nó <i>nodename</i>
<i>name.I</i>	Tensão AC no nó <i>nodename</i>
<i>nodename.v</i>	Tensão AC no nó <i>nodename</i>
<i>name.i</i>	Tensão de ruído AC no nó <i>nodename</i>
<i>nodename.vn</i>	Tensão de ruído AC no nó <i>nodename</i>
<i>name.in</i>	Tensão transiente no nó <i>nodename</i>
<i>nodename.Vt</i>	Tensão transiente no nó <i>nodename</i>
<i>name.It</i>	Corrente transiente através do componente <i>name</i>

Nota: Todas as tensões e correntes são valores de pico. Nota: Tensões de ruído são valores RMS em largura de banda de 1 Hz.

## 8.5 Constantes

$i, j$	Unidade imaginária (“raiz quadrada de -1”)
$\pi$	$4 \cdot \arctan(1) = 3.14159\dots$
$e$	Euler = 2.71828...
$k_B$	Carga elementar = 1.6021765e-19 C
$q$	Carga elementar = 1.6021765e-19 C

---

### Lista de Caracteres Especiais

---

É possível utilizar caracteres especiais em textos desenhados e nos rótulos dos eixos dos diagramas. Isso é feito usando as tags LaTeX. A tabela seguinte contém uma lista dos caracteres atualmente disponíveis.

Nota: Os caracteres destes que serão corretamente exibidos dependem da fonte utilizada pelo Qucs!

#### **Letras Gregas pequenas**

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

### Letras Gregas maiúsculas

Tag LaTeX	Unicode	Descrição
<code>\Gamma</code>	0x0393	Gamma
<code>\Delta</code>	0x0394	Delta
<code>\Theta</code>	0x0398	Theta
<code>\Lambda</code>	0x039B	Lambda
<code>\Xi</code>	0x039E	Xi
<code>\Pi</code>	0x03A0	Pi
<code>\Sigma</code>	0x03A3	Sigma
<code>\Upsilon</code>	0x03A5	Upsilon
<code>\Phi</code>	0x03A6	Phi
<code>\Psi</code>	0x03A8	Psi
<code>\Omega</code>	0x03A9	Omega

### Símbolos matemáticos

Tag LaTeX	Unicode	Descrição
<code>\cdot</code>	0x00B7	ponto de multiplicação (ponto centralizado)
<code>\times</code>	0x00D7	cruz de multiplicação
<code>\pm</code>	0x00B1	sinal de mais ou menos
<code>\mp</code>	0x2213	sinal de menos mais
<code>\partial</code>	0x2202	símbolo de diferenciação parcial
<code>\nabla</code>	0x2207	operador nabla
<code>\infty</code>	0x221E	símbolo de infinito
<code>\int</code>	0x222B	símbolo de integral
<code>\approx</code>	0x2248	símbolo de aproximação (sinal de aproximadamente igual)
<code>\neq</code>	0x2260	símbolo de não igual
<code>\in</code>	0x220A	símbolo “contido em”
<code>\leq</code>	0x2264	símbolo de menor ou igual
<code>\geq</code>	0x2265	símbolo de maior ou igual
<code>\sim</code>	0x223C	(centro europeu) sinal de proporcional
<code>\propto</code>	0x221D	(americano) sinal de proporcional
<code>\diameter</code>	0x00F8	sinal de diâmetro (também sinal de média)
<code>\onehalf</code>	0x00BD	metade de um inteiro
<code>\onequarter</code>	0x00BC	um quarto de inteiro
<code>\twosuperior</code>	0x00B2	elevado ao quadrado (potência de dois)
<code>\threesuperior</code>	0x00B3	elevado ao cubo
<code>\ohm</code>	0x03A9	unidade de resistência (omega Maiúscula Grega)



# CAPÍTULO 10

---

## Circuitos de Casamento

---

Criar circuitos de casamento é uma tarefa frequentemente necessária na tecnologia de microondas. Qucs pode fazer isso automaticamente. Estes são os passos necessários:

Faça uma simulação de parâmetro-S para calcular o coeficiente de reflexão.

Coloque um diagrama e exiba o coeficiente de reflexão (ex.  $S[1,1]$  para port 1,  $S[2,2]$  para port 2 etc.)

Crie um marcador no gráfico utilizando a opção do menu Inserir>Inserir Marcador no Gráfico e clique em cima da curva, na frequência desejada.

Clique com o botão direito do mouse no marcador e selecione no menu que aparecer a opção “casamento de potência”.

Uma janela aparecerá onde os valores podem ser ajustados, por exemplo: A impedância de referência pode ser escolhida diferente dos 50 ohms.

Após clicar em “criar” a página retornará para o esquema elétrico e movendo o cursor do mouse, o circuito de casamento poderá ser colocado.

O lado esquerdo do circuito de casamento é a entrada e o lado direito, deve ser conectado ao circuito.

Se os pontos do marcador para uma variável chamada “Sopt”, o Menu mostrará a opção “casamento de ruído”. Note que a única diferença para “casamento de potência” é o fato de que o coeficiente de reflexão do complexo conjugado é tomado. Então, se a variável tem outro nome, o casamento de ruído pode ser escolhido re-ajustando os valores na janela.

A janela de casamento pode ser também chamada pelo Menu (Ferramentas->Circuito de Casamento) ou pelo atalho (<CTRL-5>). Mas então todos os valores tem que ser entrados manualmente.

### 10.1 Circuitos de Casamento de 2-Portas

Se o nome da variável no marcador de texto for um parâmetro S, então existe uma opção para casar simultaneamente entrada e saída de um circuito de duas portas. Isto funciona de forma completamente semelhante aos passos mencionados acima. Seus resultados são dois circuitos L: O nó mais a esquerda é para conectar a porta 1, o nó mais a direita é para conectar a porta 2 e os dois nós do meio são para conectar o circuito de duas portas.





---

## Arquivos Instalados

---

O sistema Qucs precisa de vários programas. Eles são instalados durante o processo de instalação. O diretório de instalação do Qucs é determinada durante a instalação (script configure). As seguintes explicações assumem o diretório padrão (/usr/local/).

- /usr/local/bin/qucs - a interface gráfica (GUI)
- /usr/local/bin/qucsator - o simulador (aplicação de console)
- /usr/local/bin/qucsedit - um edito de textos simples
- /usr/local/bin/qucshelp - um pequeno programa exibindo o sistema de ajuda
- /usr/local/bin/qucstrans - um programa para calcular parâmetros de linhas de transmissão
- /usr/local/bin/qucsfilter - um programa que sintetiza circuitos de filtro
- /usr/local/bin/qucsconv - um conversor de formato de arquivo (aplicação de console)

Todos os programas são aplicações separadas e podem ser iniciados de forma independente. O programa principal (GUI)

- chama qucsator quando executa uma simulação,
- chama qucsedit quando mostra arquivos de texto,
- chama qucshelp quando exibe o sistema de ajuda,
- chama qucstrans quando este programa é chamado do Menu “Ferramentas”,
- chama qucsfilter quando este programa é chamado do Menu “Ferramentas”,
- chama qucsconv quando é colocado um componente SPICE e é feita uma simulação com ele.

Além disso, os seguintes diretórios são criados durante a instalação:

- /usr/local/share/qucs/bitmaps - contém todas os bitmaps (ícones, etc.)
- /usr/local/share/qucs/docs - contém documentos HTML para o sistema de ajuda
- /usr/local/share/qucs/lang - contém os arquivos de tradução

## 11.1 Argumentos de linha de comando

```
qucs [arquivo1 [arquivo2 ...]]  
qucsator [-b] -i netlist -o dataset (b = barra de progresso)  
qucsedit [-r] [file] (r = somente leitura)  
qucshelp (sem argumentos)  
qucsconv -if spice -of qucs -i netlist.inp -o netlist.net
```

---

## Formato dos Arquivos de Diagrama Elétrico

---

Este documento descreve o formato dos arquivos de diagrama elétrico do Qucs. Este formato é utilizado para diagramas elétricos (normalmente com o sufixo `.sch`) e para exibir dados (normalmente com o sufixo `.dpl`). O texto abaixo mostra um exemplo curto de um arquivo de diagrama elétrico.

```
<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>
```

O arquivo contém várias seções. Cada uma delas é explicada abaixo. Cada linha consiste em nada mais que um bloco de informação que começa com um sinal de menor `<` e termina com um sinal de maior `>`.

## 12.1 Properties

A primeira seção começa com `<Properties>` e termina com `</Properties>`. Ela contém as propriedades de documento do arquivo. Cada uma das linhas é opcional. As seguintes propriedades são suportadas:

- `<View=x1,y1,x2,y2,scale,xpos,ypos>` contém a posição dos pixels da janela do esquema elétrico nos primeiros quatro números, sua escala atual e a posição atual do canto superior esquerdo (dois últimos números).
- `<Grid=x,y,on>` contém a distância da grade em pixels (primeiros dois números) e se a grade está ligada (último número 1) ou desligada (último número 0).
- `<DataSet=name.dat>` contém o nome do arquivo de dados associado com este esquema elétrico.
- `<DataDisplay=name.dpl>` contém o nome do arquivo de exibição de dados associado com o esquema elétrico (ou nome do arquivo de esquema elétrico se este documento for um arquivos de exibição de dados).
- `<OpenDisplay=yes>` contém 1 se a página de exibição de dados abrir automaticamente após a simulação, caso contrário contém 0.

## 12.2 Symbol

Esta seção começa com `<Symbol>` e termina com `</Symbol>`. Ela contém os elementos desenhados criando um esquema elétrico simbólico para o arquivo. Isto só é normalmente usado para arquivos de esquemas elétricos que signifiquem estar em subcircuitos.

## 12.3 Components

Esta seção começa com `<Components>` e termina com `</Components>`. Ela contém os componentes do circuito do esquema elétrico. O formato da linha é como se segue:

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

- `type` identifica o componente, ex. R para um resistor, C para um capacitor.
- `name` é o identificador único para o componente do esquema elétrico, ex. R1 para o primeiro resistor.
- Um 1 no campo `active` mostra que o componente está ativo, isto é, ele será usado na simulação. Um 0 mostra que está desativado.
- Os próximos dois números são as coordenadas `x` e `y` do centro do componente.
- Os próximos dois números são as coordenadas `x` e `y` do canto superior esquerdo da componente de texto. Elas são relativas ao centro do componente.
- Os próximos dois números indicam o espelhamento sobre o eixo `x` (1 para espelhado, 0 não espelhado) e a rotação no sentido anti-horário (múltiplos de 90 graus, ex. 0...3).
- As próximas entradas são os valores das propriedades do componente (entre aspas duplas) seguidos por um 1 se a propriedade estiver visível no esquema elétrico (caso contrário 0).

## 12.4 Wires

Esta seção começa com `<Wires>` e termina com `</Wires>`. Ela contém os fios (conexão elétrica entre os componentes do circuito) e seus rótulos e nós. O formato da linha é como se segue:

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

- Os primeiros quatro números são as coordenadas do fio em pixels: coordenada x do ponto inicial, coordenada y do ponto inicial, coordenada x do ponto final e coordenada y do ponto final. Todos os fios devem ser horizontais (ambas coordenadas x iguais) ou verticais (ambas coordenadas y iguais).
- A primeira propriedade entre aspas duplas é o rótulo. Ela fica vazia se nenhum rótulo for atribuído a este fio.
- Os próximos dois números são as coordenadas x e y do rótulo ou zero se o rótulo não existir.
- O próximo número é a distância entre o ponto de início do fio e o ponto onde o rótulo for atribuído no fio.
- A última propriedade entre aspas duplas é o nó do fio, ex. a tensão inicial neste nó usada pela simulador para encontrar a solução. Este fica vazio se o usuário não definir um nó para este fio.

## 12.5 Diagrams

Esta seção começa com `<Diagrams>` e termina com `</Diagrams>`. Ela contém os diagramas com seus gráficos e seus marcadores.

```
<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">
```

- Os primeiros dois números são as coordenadas x e y do canto esquerdo inferior.
- Os dois números seguintes são a largura e altura de limitação do diagrama.
- O quinto número é 1 se a grade estiver ligada e 0 se estiver desligada.
- O próximo é a cor da grade em valor RGB hexadecimal de 24 bits, ex. #FF0000 é vermelho.
- O próximo número determina o estilo da grade.
- O próximo número determina quais eixos tem escala logarítmica.

## 12.6 Paintings

Esta seção começa com `<Paintings>` e termina com `</Paintings>`. Ela contém as pinturas que estão dentro do esquema elétrico.



---

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

---

**Mike Brinson**

Copyright 2014, 2015 Mike Brinson, Centre for Communications Technology, London Metropolitan University, London, UK. (<mailto:mbrin72043@yahoo.co.uk>)

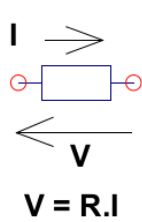
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation.

### 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).

**Model Properties**

<b>R</b>	<b>50</b>	<b>Resistance in Ohms</b>
<b>Temp</b>	<b>26.85</b>	<b>Simulation temperature in Celsius</b>
<b>Tc1</b>	<b>0.0</b>	<b>First order temperature coefficient</b>
<b>Tc2</b>	<b>0.0</b>	<b>Second order temperature coefficient</b>
<b>Tnom</b>	<b>26.85</b>	<b>Temperature at which parameters are extracted</b>

where  $R(\text{Temp}) = R(\text{Tnom}) \cdot (1 + \text{Tc1} \cdot (\text{Temp} - \text{Tnom}) + \text{Tc2} \cdot (\text{Temp} - \text{Tnom})^2)$

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.

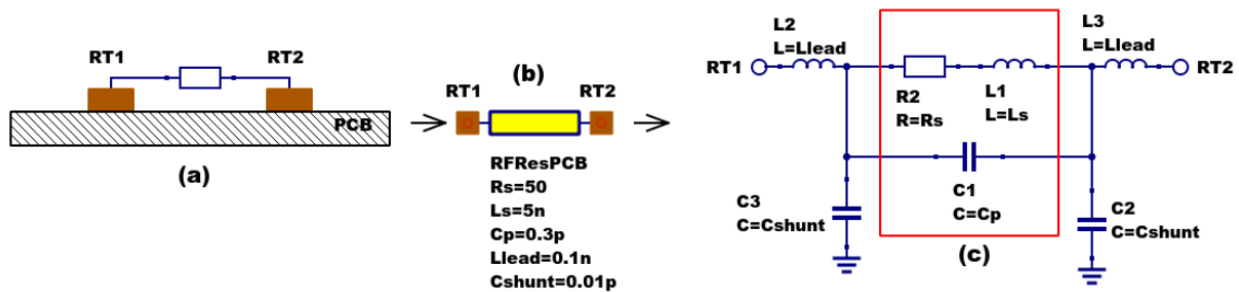


Figure 2 - PCB mounted resistor: (a) axial component mounting, (b) Qucs symbol and (c) equivalent circuit model.

A typical set of model parameters for a 51  $\Omega$  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.



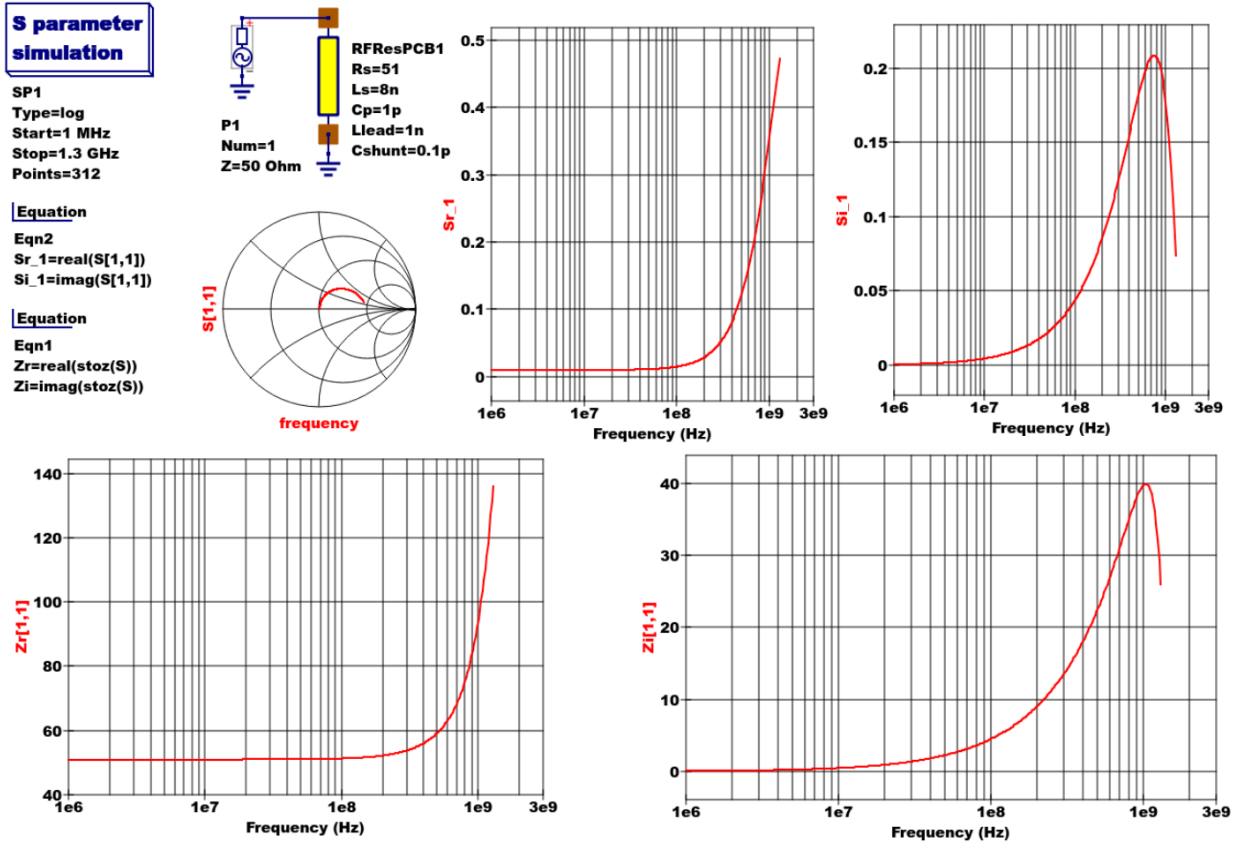


Figure 3 - Qucs S parameter simulation test circuit and plotted output data for a MF axial resistor:  $R_s=51\Omega$ ,  $L_s=8\text{nH}$ ,  $C_p=1\text{pF}$ ,  $L_{lead}=1\text{nH}$  and  $C_{shunt}=0.1\text{pF}$ .

### 13.3 Analysis of the RF resistor model

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

$$\begin{aligned}
 Z1 &= j \cdot \omega \cdot L_{lead} \\
 Z2 &= \frac{R_s + j \cdot \omega \cdot L_s \cdot (1 - \omega^2 \cdot C_p \cdot L_s) - j \cdot \omega \cdot C_p \cdot R_s^2}{(1 - \omega^2 \cdot C_p \cdot L_s)^2 + (\omega \cdot C_p \cdot R_s)^2} \\
 Z3 &= \frac{j \cdot \omega \cdot L_{lead}}{(1 - \omega^2 \cdot L_{lead} \cdot C_{shunt})} \\
 Z_{series} &= Z1 + Z2 = R_{series} + j \cdot X_{series} \\
 Zb &= Z_{series} || XC_{shunt} = \frac{Z_{series}}{(1 + j \cdot \omega \cdot C_{shunt} \cdot Z_{series})} = ZBR + j \cdot \omega \cdot ZBI, \\
 Z &= j \cdot \omega \cdot L_{lead} + Zb = ZR + j \cdot \omega \cdot ZI.
 \end{aligned}$$

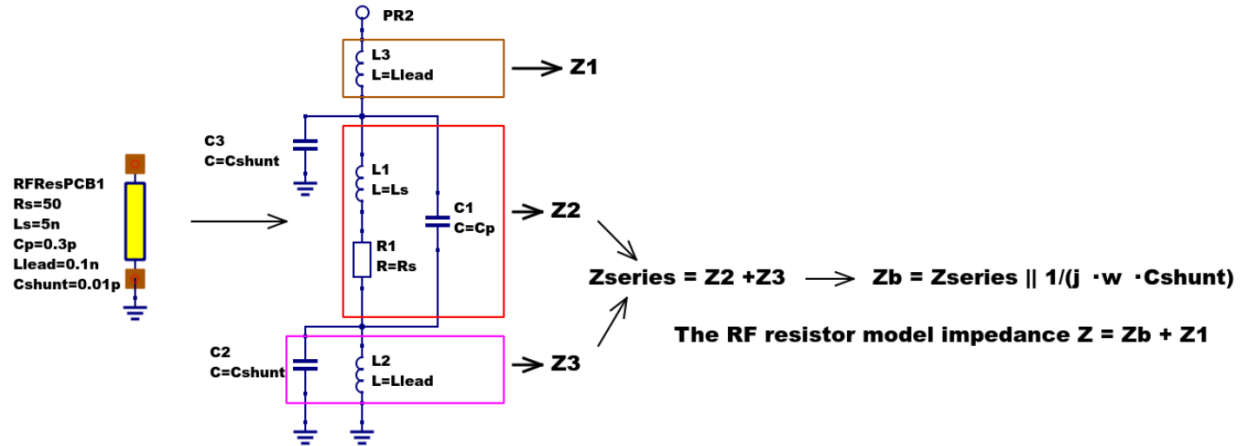


Figure 4 - RF resistor model rotated through 90 degrees and connected with one terminal grounded, similar to the test circuit in Figure. Sections of the model are shown grouped for calculation of the model impedance  $Z$ .

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.

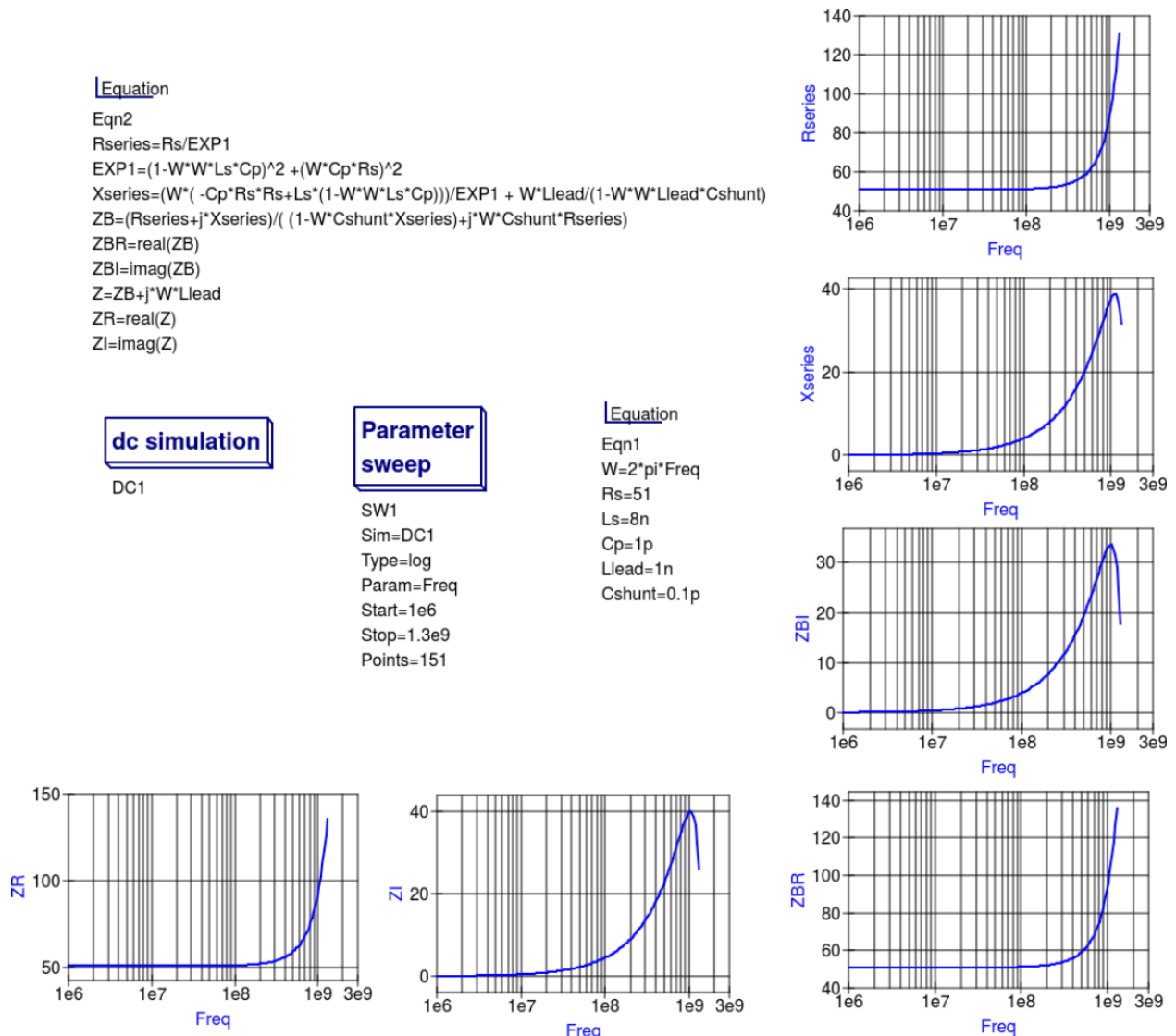


Figure 5- Theoretical analysis of RF resistance impedance Z using Qucs post processing facilities: note a dummy simulation icon, in this example DC simulation, is required to force Qucs to complete the analysis calculations.

## 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  $\Omega$  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.

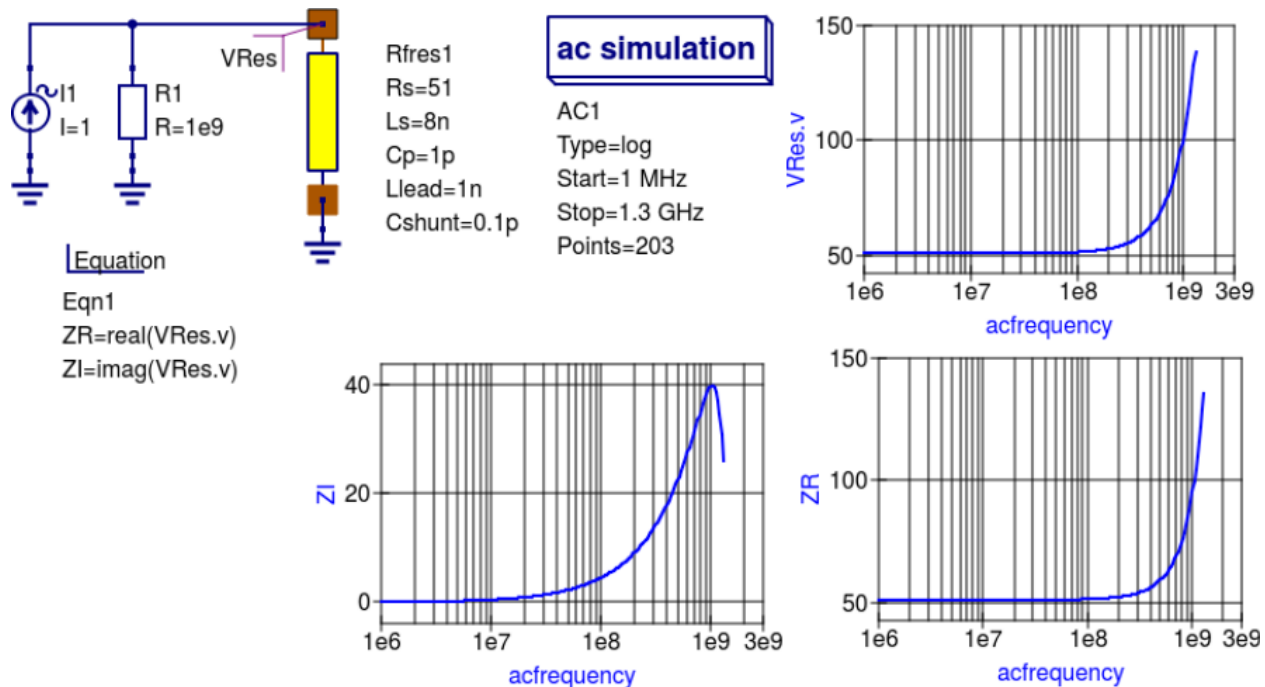


Figure 6 -A simple Qucs test circuit for demonstrating the use of an AC constant current source to measure electrical network impedance.

## 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 (VNA)<sup>1</sup>. 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 VNA 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  $\Omega$  resistor<sup>2</sup>. As well as displaying, and printing, measured data the DGSAQ Vector Network Analyser software can output data tabulated in Touchstone<sup>3</sup> “SnP” 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 VNA 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 VNA 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  $\Omega$** . 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

<sup>1</sup> DG8SAQ VNA 3 & 3E- Vector Network Analysers, SDR Kits Limited, Grangeside Business Centre, 129 Devizes Road, Trowbridge, Wilts, BA14-7sZ, United Kingdom, 2014.

<sup>2</sup> See DG8SAQ VNA 3 & 3E- Vector Network Analysers- Getting Started Manual for Windows 7, Vista and Windows XP.

<sup>3</sup> ([http://www.vhdl.org/ibis/connector/touchstone\\_spec11.pdf](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  $\Omega$  resistor the post-minimization RF resistor model parameters are **Rs = 47.3  $\Omega$** , **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.

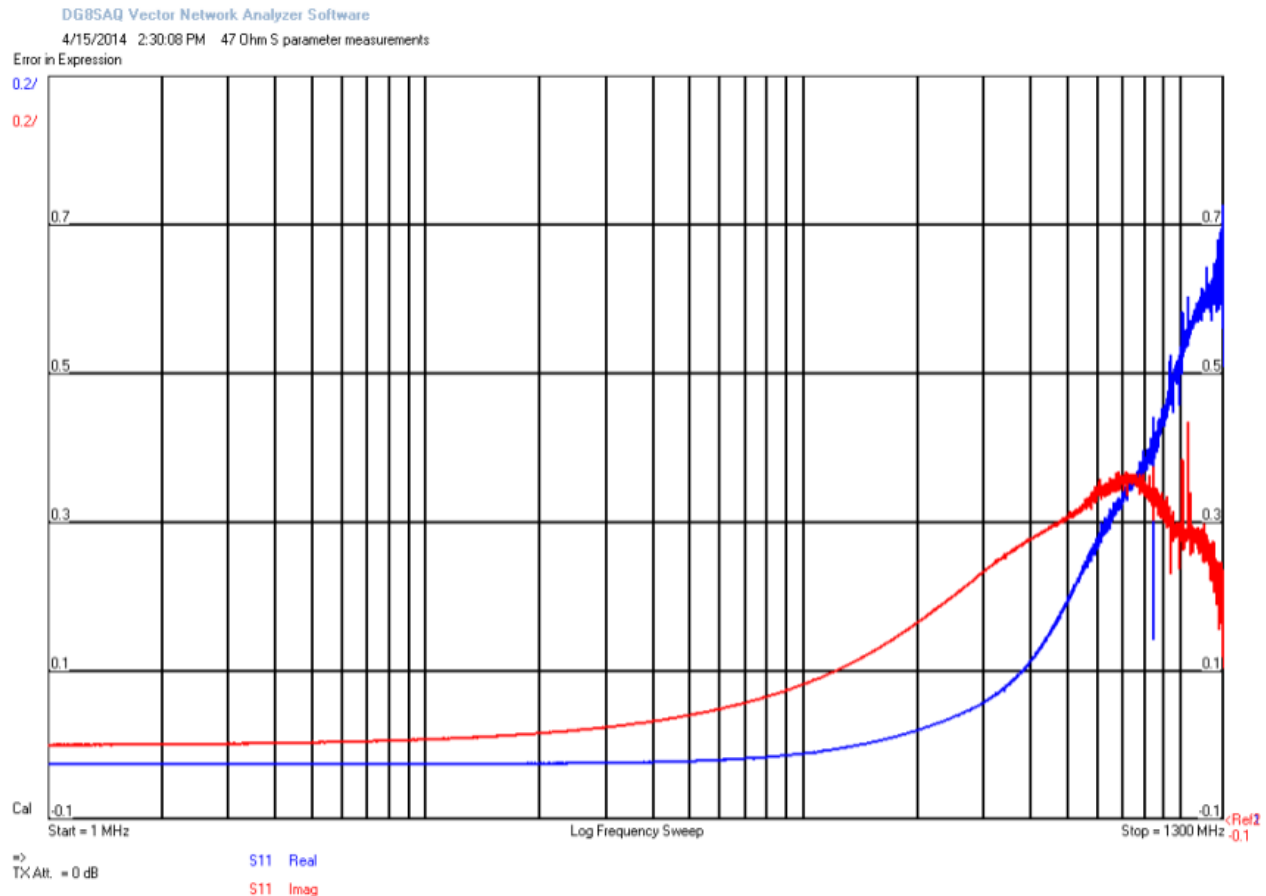


Figure 7 - DGSAQ Vector Network Analyser S parameter measurements for a 47  $\Omega$  axial RF resistor.

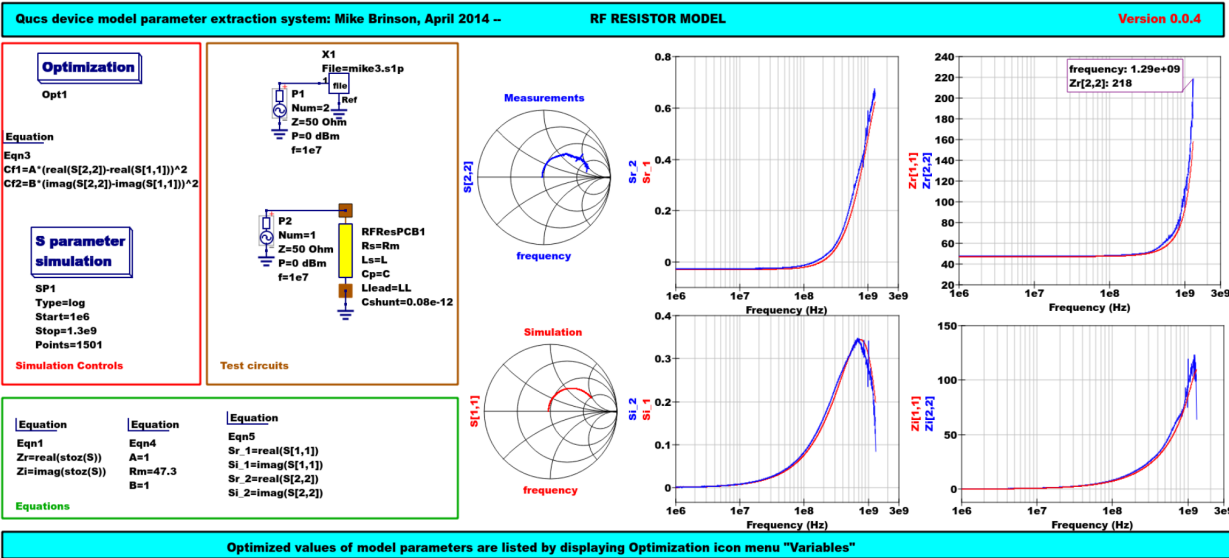


Figure 8 - Qucs device model parameter extraction system applied to a nominal 47  $\Omega$  resistor represented by the subcircuit model illustrated in Figure 2 (c). Fixed model parameter values:  $R_s = R_m = 47.3 \Omega$ ,  $C_{Shunt} = 0.08 \text{ pF}$ ; Optimised values:  $L_s = L = 10.43 \text{ nH}$ ,  $L_{lead} = LL = 1.47 \text{ nH}$ ,  $C_p = C = 0.69 \text{ pF}$ . To reduce simulation time the ASCO cost variance was set to 1e-3. The ASCO method was set to DE/best/1/exp.

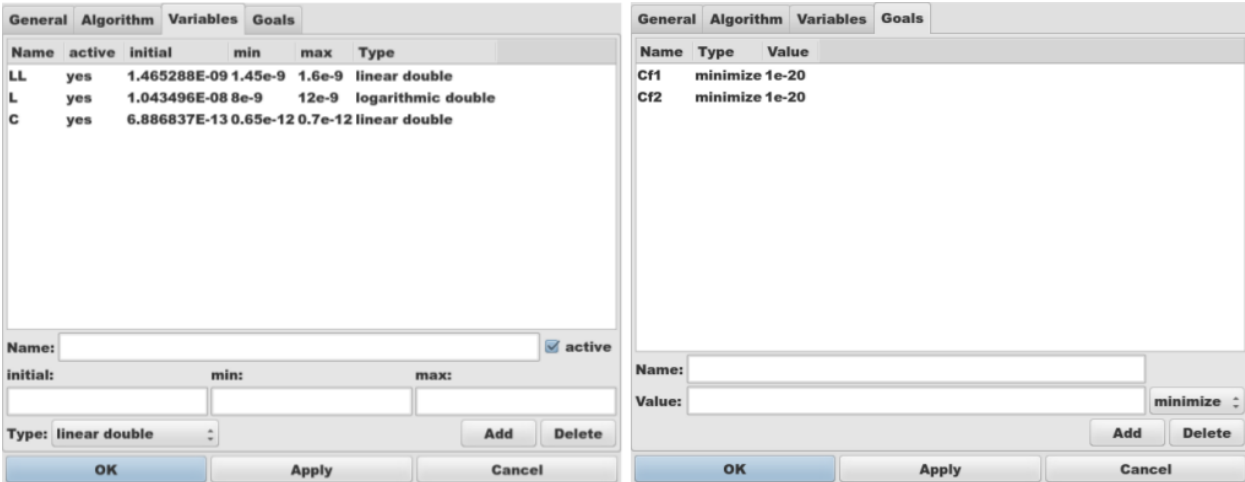


Figure 9 - Qucs Minimization Icon drop down menus: left "Variables" and right "Goals".

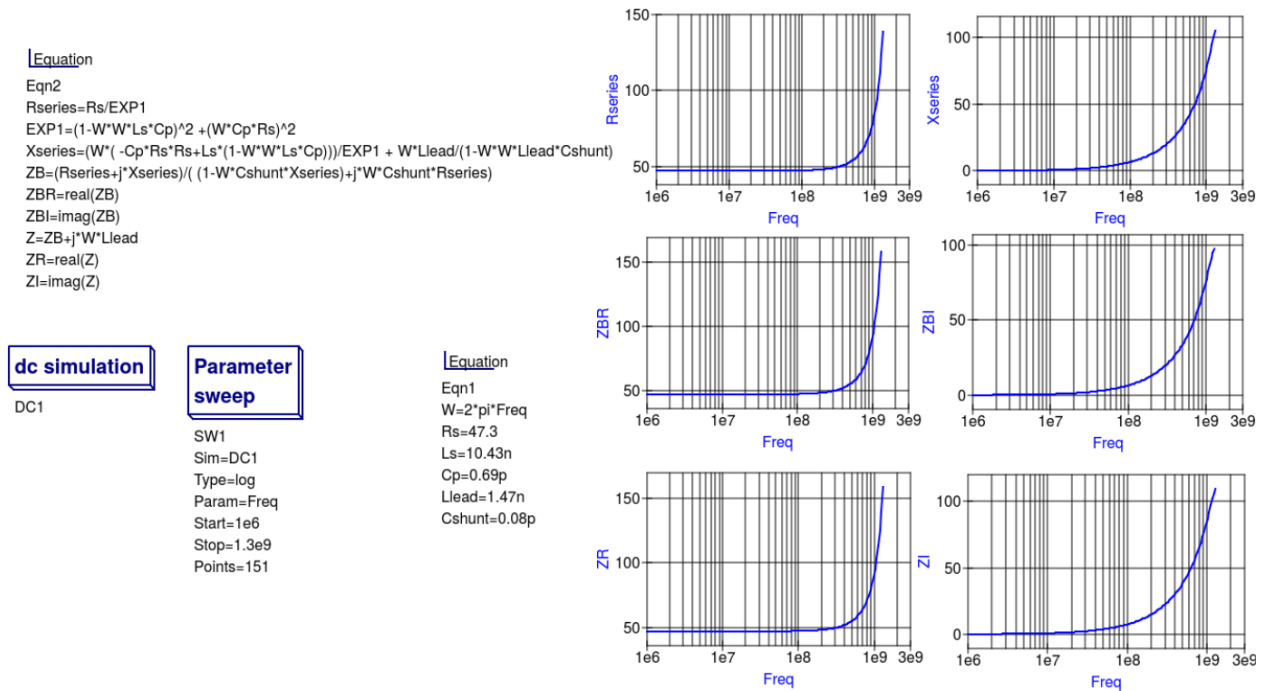


Figure 10 - Qucs simulation of nominal 47  $\Omega$  resistor based on theoretical analysis.

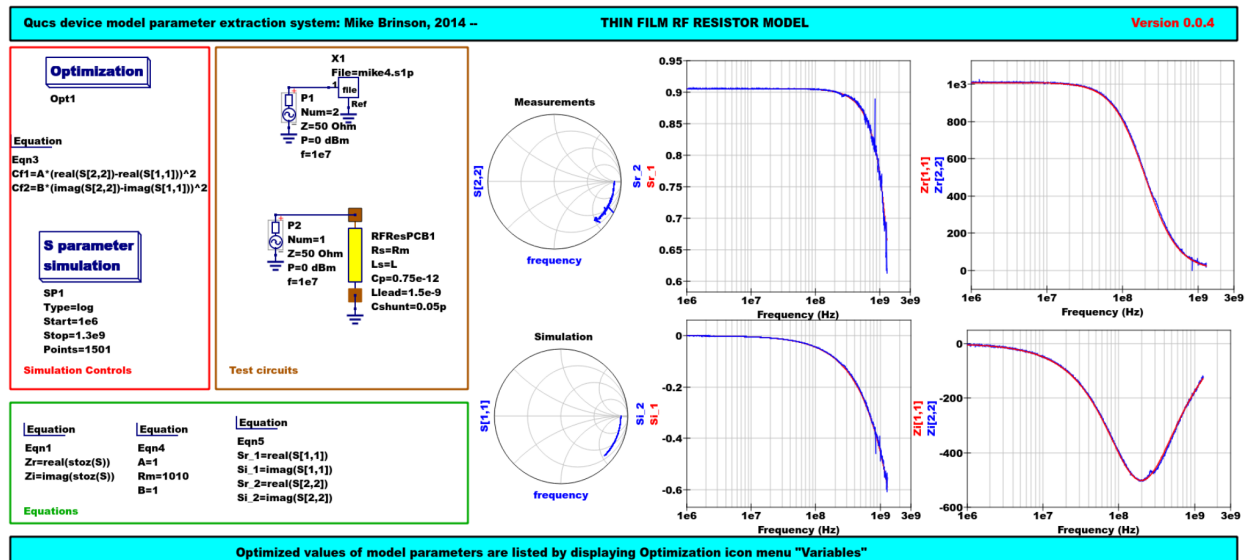
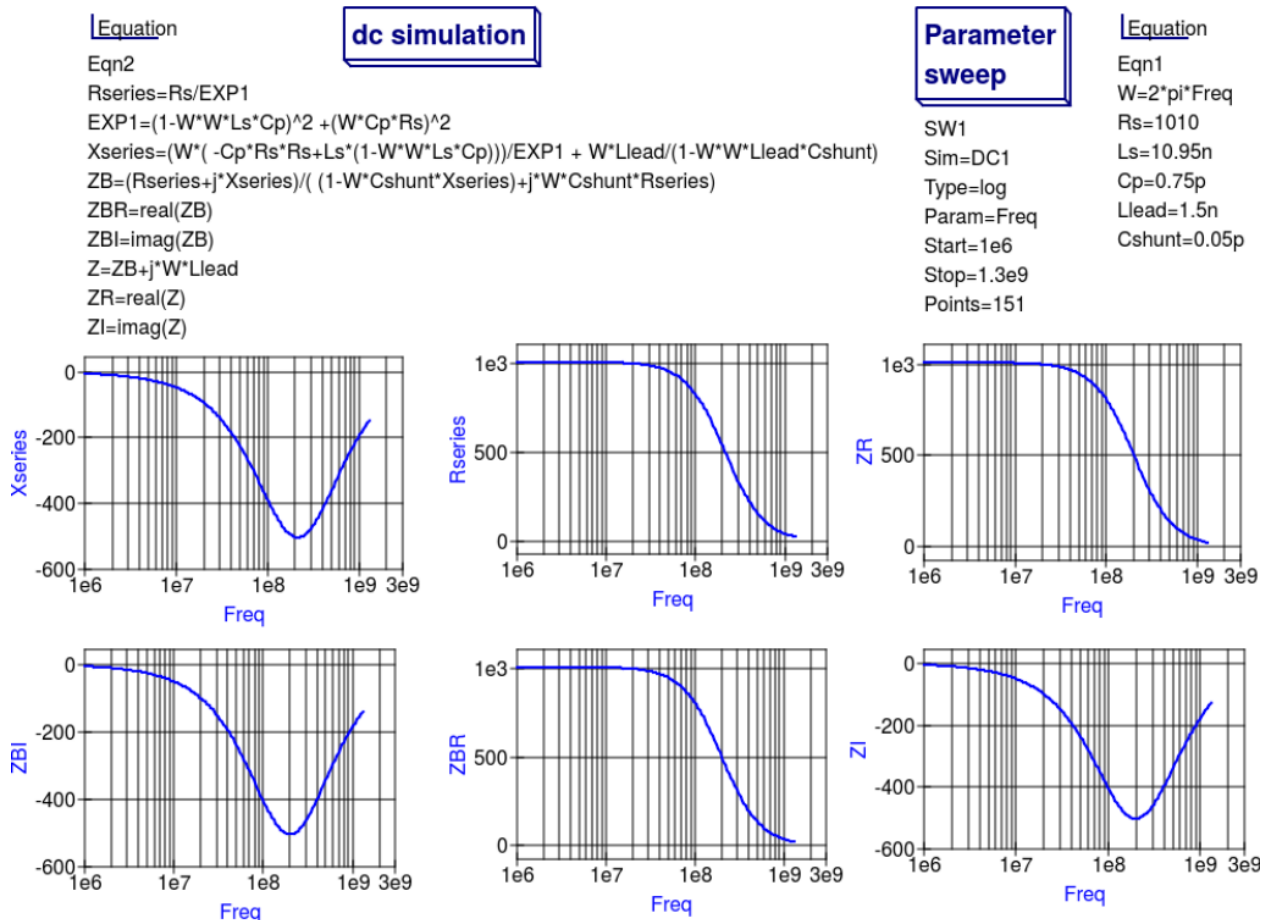


Figure 11 - Qucs device model parameter extraction system applied to a nominal 1000  $\Omega$  resistor represented by the subcircuit model illustrated in Figure 2(c).

Figure 12 - Qucs simulation of nominal 1000  $\Omega$  resistor based on theoretical analysis.

### 13.6 Extraction of RF resistor parameters from measured S data for a nominal 1000 $\Omega$ 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  $L_s$  contribution dominates any reactance effects by  $C_p$ ,  $L_{lead}$  and  $C_{shunt}$ . However, as  $R_s$  is increased above a few hundred Ohm's the reverse becomes true with reactive effects dominated by contributions from  $C_p$ . Figures 11 and 12 demonstrate the dominance of  $C_p$  reactive effects at low to mid-range frequencies.

### 13.7 One more example: extraction of RF resistor parameters from measured S data for a nominal 100 $\Omega$ 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 capacitive parasitics which in turn ensures that the high frequency performance of SMD resistors is much improved.



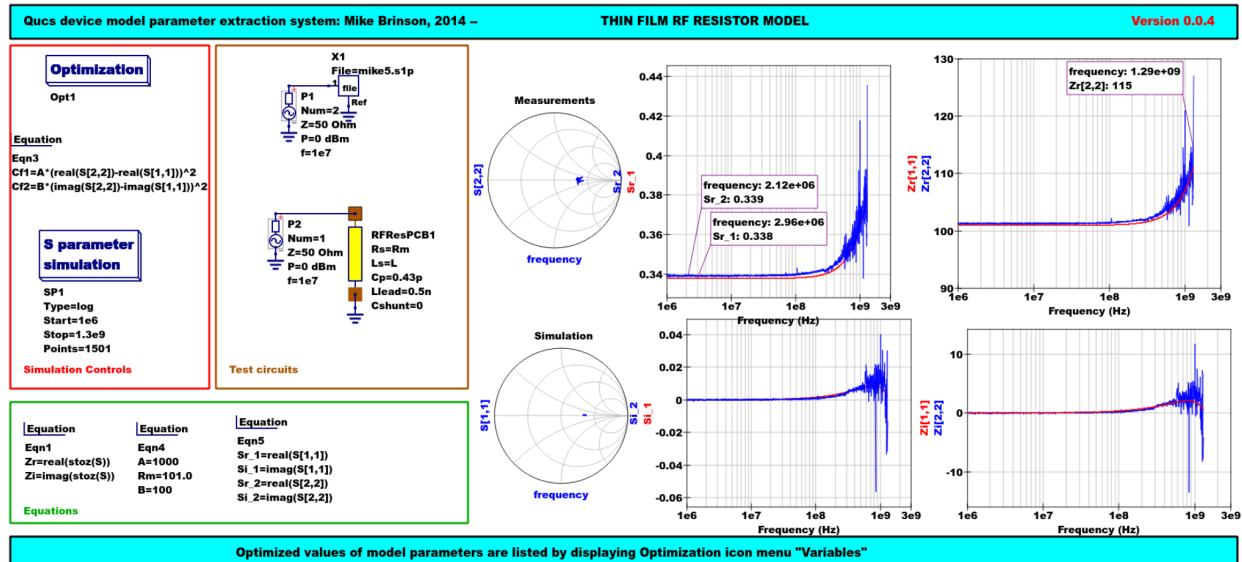


Figure 13 - Qucs device model parameter extraction system applied to a nominal 100  $\Omega$  SMD resistor represented by the subcircuit model illustrated in Figure 2 (c).

## 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)<sup>4</sup> inductors **Ls** and **Lead** are modelled by gyrators and capacitors with values identical to **Ls** or **Lead**.

```
// 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 inductance" unit="H");
```

(continues on next page)

<sup>4</sup> (<http://sourceforge.net/projects/mot-adms/>).

(continuação da página anterior)

```

parameter real Llead = 0.1e-9  from [1e-20 : inf)
`attr(info="Parasitic lead inductance" 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
Is	Series inductance	8.5e-9
Llead	Parasitic lead inductance	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

$T_{diff} = (Temp - Tnom);$

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

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

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

----- end of Initial Model

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

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

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

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

$I(RT1, RT1) <+ (-V(nx, nx));$

$I(nx, nx) <+ V(RT1, RT1);$

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

$I(n2, n2) <+ (-V(ny, ny));$

$I(ny, ny) <+ V(n2, n2);$

$I(ny, ny) <+ ddt((Is \cdot V(ny, ny)));$

$I(n3, n3) <+ (-V(nz, nz));$

$I(nz, nz) <+ V(n3, n3);$

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

$I(n1, n1) <+ white\_noise(Rn, "thermal");$

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 $\Omega$ 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.

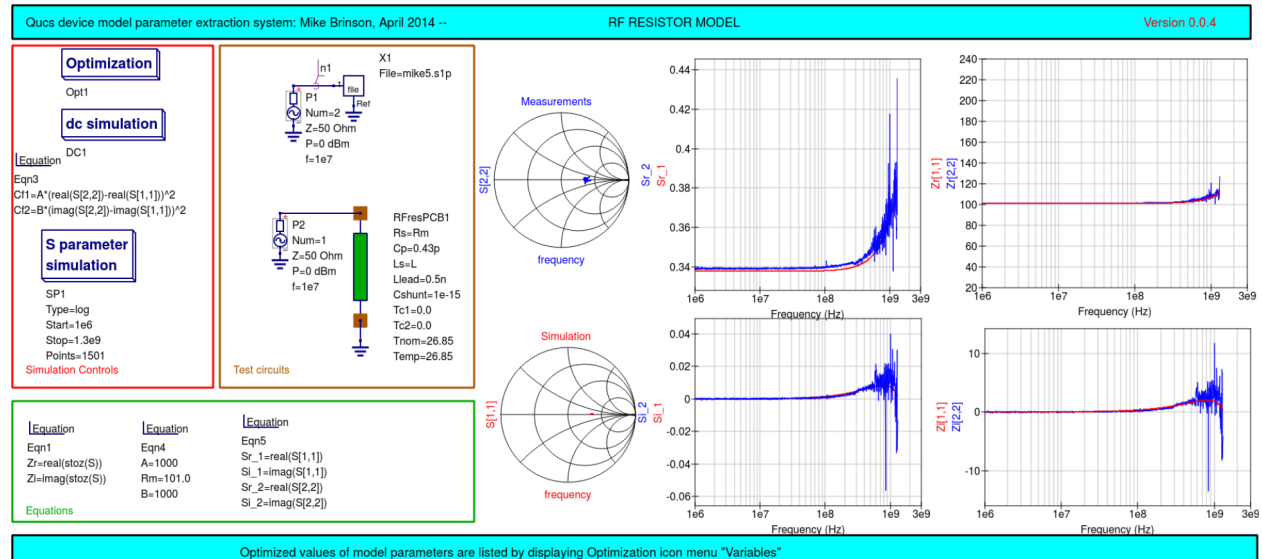


Figure 15 - Verilog-A models parameter data extraction for a 100  $\Omega$  axial thin film resistor. Fixed model parameter values:  $R_s = R_m = 101 \Omega$ ,  $C_{shunt} = 1e-15 F$ ,  $L_{lead} = L_L = 0.5nH$ ,  $C_p = C = 0.43pF$ ; Optimised values:  $L_s = L = 3.99nH$ . To reduce simulation time 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 subcircuit 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<sup>5</sup>.

Descrição técnica a respeito do simulador

Disponível em <http://qucs.sourceforge.net/tech/technical.html>

Circuitos de Exemplo

Disponíveis em <http://qucs.sourceforge.net/download.html#example>

<sup>5</sup> Qucs release 0.0.18, or greater.