Qucs Help Documentation

Release 0.0.19

Qucs Team

nov 05, 2017

Sumário

Conteúdo:

CAPÍTULO 1

Plano de fundo

O ‘Quite universal circuit simulator’ Qucs (pronuncia-se: kju:ks) é um simulador de circuito de código aberto desenvolvido por um grupo de engenheiros, cientistas e matemáticos sob licença pública geral GNU (GPL). Qucs foi criado por engenheiros alemães Michael Margraf e Stefan Jahn. Desde o seu lançamento público inicial em 2003 cerca de vinte colaboradores, de todas as regiões do mundo, investiram seus conhecimentos e tempo para apoiar o desenvolvimento do software. A publicação de executáveis e código fonte ocorrem em intervalos regulares. Diversas versões de Qucs e prévias do código em desenvolvimento podem ser baixados diariamente em (‘< http://qucs.sourceforge.net >’ _). Versões estão disponíveis para Linux (Ubuntu e outras distribuições), Mac OS X © e Windows © 32 bits.

No período desde que foi lançado Qucs evoluiu em uma simulação de circuito avançado e ferramenta de modelagem de dispositivos com uma “interface gráfica” (GUI) amigável para captura de circuito esquemático, para investigar as propriedades do circuito e o dispositivo de DC para RF e além, e para o lançamento de outro software de simulação do circuito, incluindo os simuladores digitaiss FreeHDL (VHDL) e Icarus Verilog. Qucs inclui código interno para processamento e visualização de dados de saída da simulação. Qucs também permite o usuários processar dados de pós-simulação com o popular pacote de análise de dados numéricos Octave. Da mesma forma, otimização de desempenho de circuito é possível usando o pacote SPICE Circuit Optimizer (ASCO) ou código Python ligada à Qucs.

Entre 2003 e janeiro de 2015, as estatísticas de download de Qucs no sourceforge mostram que mais de um milhão de downloads do software foram gravados. Além das extensas capacidades de simulação de circuitos, a Qucs suporta uma gama completa de recursos de modelagem de dispositivos, incluindo modelagem de dispositivos não linear e RF definida por equações e o uso da linguagem de descrição de hardware Verilog-A (HDL) para modelagem de dispositivos compactos e macromodelagem. As recentes extensões ao software visam diversificar as instalações de modelagem da Qucs executando a “Plataforma de Prototipagem de Modelos e Algoritmos” de Berkeley (MAPP) em paralelo com a Qucs, usando Octave lançado a partir da GUI da Qucs. No futuro, à medida que o projeto da Qucs evoluir, o software também fornecerá aos projetistas de circuitos uma opção de mecanismo de simulação selecionado a partir do código interno do Qucs, ngspice e Xyce ©.

Qucs é um grande pacote de software que leva tempo para aprender. Aliás, esta afirmação também é válida para outros simuladores de circuitos GPL. Novos usuários devem perceber que para obter o melhor do software algum esforço é necessário de sua parte. Em particular, uma das melhores maneiras de se familiarizar com Qucs é aprender algumas regras básicas do usuário e como aplicá-las. Uma vez que estes foram dominados os usuários podem avançar com confiança para o próximo nível de compreensão. Eventualmente, um estágio será alcançado que permite que Qucs seja usado produtivamente para modelar dispositivos e investigar o desempenho de circuitos. Qucs é igualmente fácil de usar por iniciantes, como as crianças em idade escolar que aprendem a física dos circuitos elétricos que consistem

em uma bateria e um ou mais resistores, como é por engenheiros de vanguarda que trabalham na modelagem de sub-nano transistor de RF de tamanho com centenas de parâmetros físicos.

O objetivo principal dessas notas é fornecer aos usuários de Qucs uma fonte de referência para a operação e os recursos do software. As informações fornecidas também indicam quaisquer limitações conhecidas e, se disponíveis, fornecem detalhes de quaisquer work-arounds. Qucs é uma ferramenta científica/engenharia de alto nível cuja operação e desempenho exigem que os usuários compreendam os princípios matemáticos, científicos e de engenharia básicos subjacentes à operação de dispositivos eletrônicos e à concepção e análise de circuitos eletrônicos. Assim, as seções individuais do documento Qucs-Help incluem material de natureza técnica misturado com detalhes da operação do software. A maioria das seções apresenta um número de desenhos trabalhados e exemplos de simulação. Estes foram classificados para ajudar os leitores com diferentes níveis de compreensão obter o melhor do simulador de circuito Qucs. Qucs-Help é um documento dinâmico que vai mudar com cada nova versão do software Qucs. Neste momento, Qucs versão 0.0.19, o documento está longe de ser concluída, mas dado o tempo vai melhorar.

CAPÍTULO 2

Começando com a simulação de circuitos analógicos Qucs

Qucs é um pacote de software científico/engenharia para analógico e simulação de circuitos digitais, incluindo análise de DC linear e não linear, análise de circuito pequeno sinal S parâmetro, análise de transiente de domínio de tempo e simulação de circuitos digitais de VHDL/Verilog. Esta seção do documento Qucs-Help apresenta aos leitores as etapas básicas envolvidas na simulação de circuitos analógicos de Qucs. Quando Qucs é iniciado pela primeira vez, ele cria um diretório chamado `.qucs` dentro do diretório home do usuário. Envolvido em simulações Qucs todos os arquivos são salvos no diretório `.qucs` ou em um de sub-diretórios. Após Qucs foi lançado, o software exibe uma janela de Interface de usuário gráfica (GUI) semelhante ou igual, ao mostrado na Figura 1.

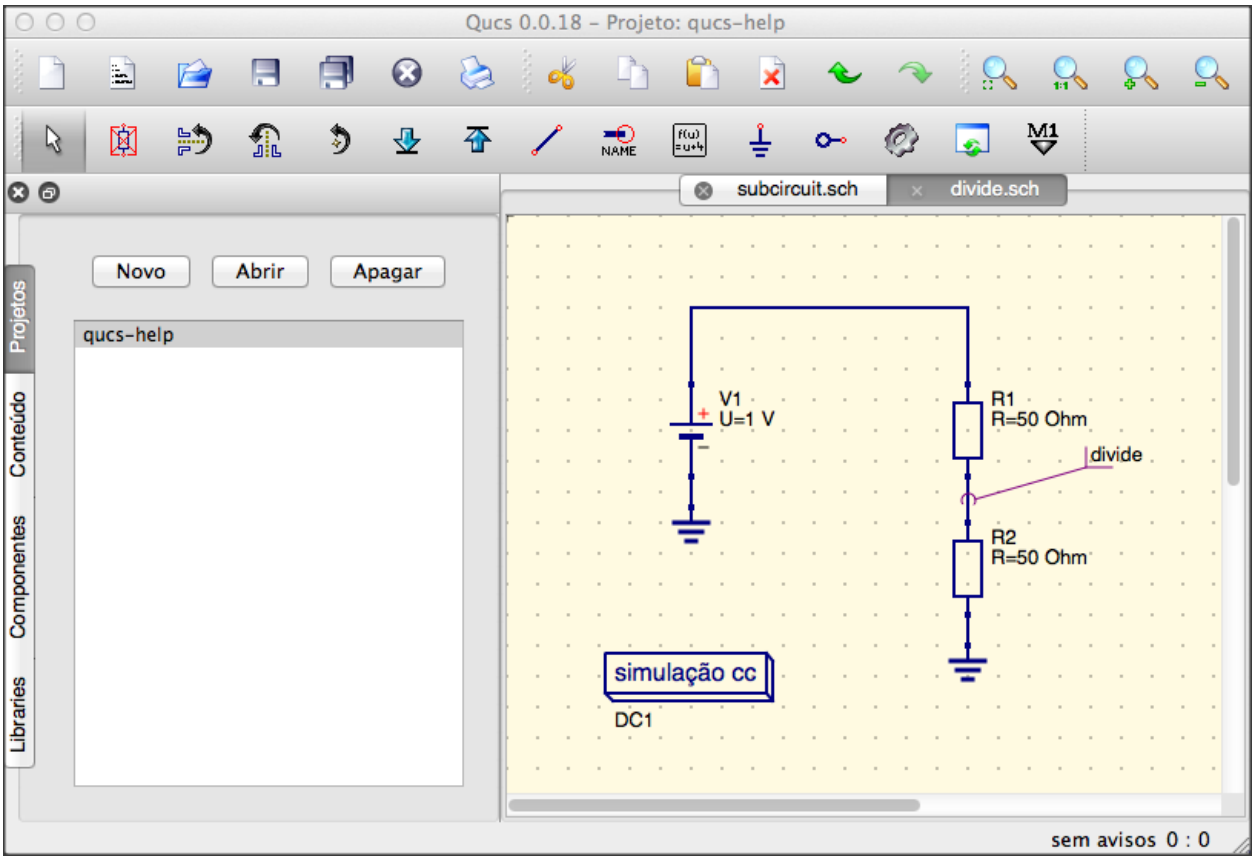


Figura 1 - Janela principal do Qucs

Antes de utilizar Qucs é aconselhável definir o programa de configurações do aplicativo. Isso é feito a partir do **Arquivo** → **Menu de configurações do aplicativo**. Clique em **Aplicativo Configurações** faz com que a janela **EditQucsProperties** seja exibida, consulte a Figura 2. Completo, com as entradas apropriadas para a sua instalação Qucs, as **Configurações**, **Editor de código fonte**, **Typos de arquivos** e **Locais** menus.

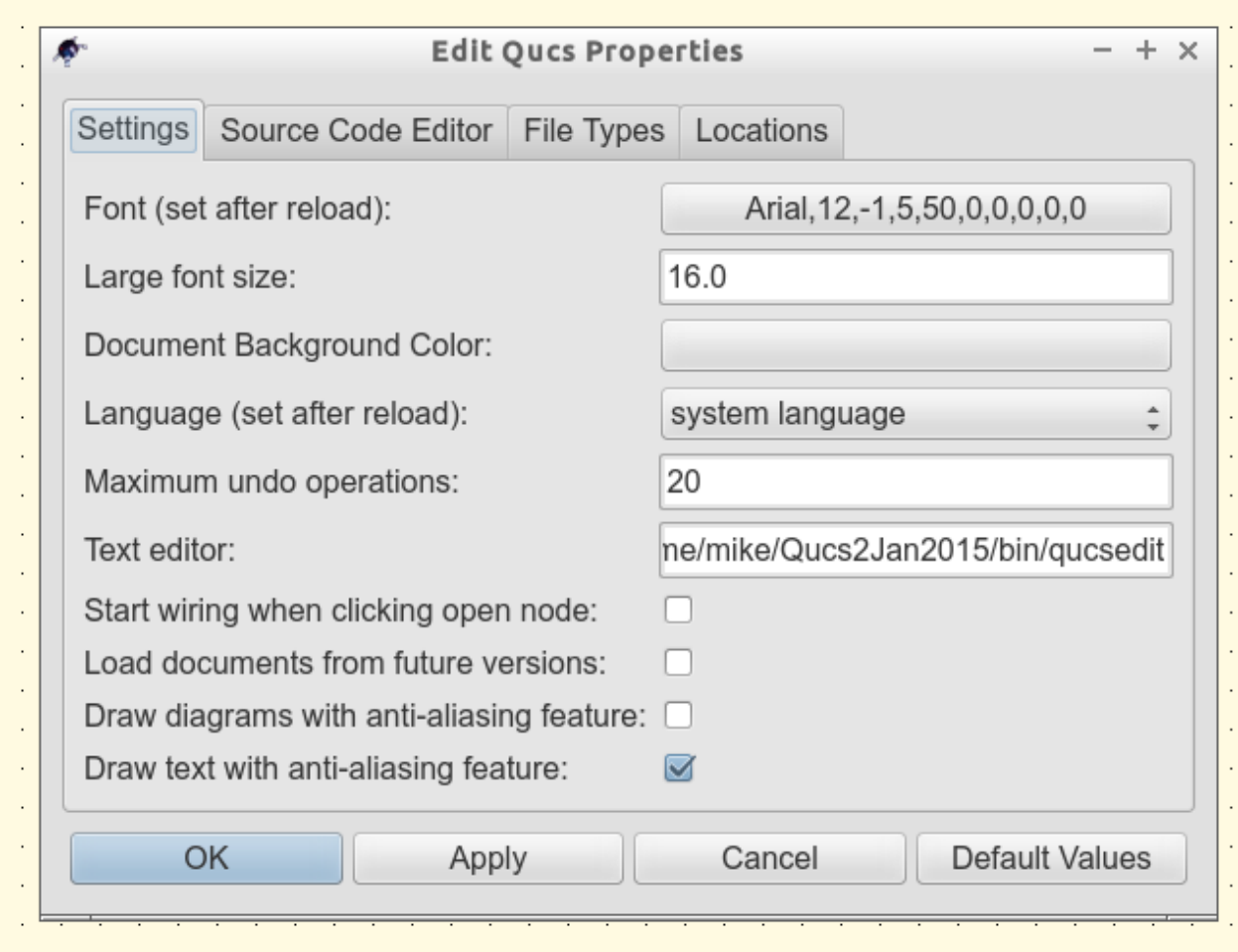


Figura 2 - janela de QucsEditProperties

No lançamento Qucs uma área de trabalho rotulados (6) aparece no centro do GUI. Esta janela é usada para exibir esquemas, numérico e algébrico modelo e projeto de circuito, dados, dados de saída numérica e formas de onda do sinal e numerial visualizado como gráficos, veja a Figura 3. Clicar, com o botão do mouse esquerda em qualquer uma das abas (5) permite aos usuários alternar rapidamente entre os documentos abertos no momento. O lado esquerdo da janela principal do Qucs é uma terceira área rotulada (1) cujo conteúdo depende do estatuto de **Projetos** (2), **Conteúdo** (3), **Componentes* * (4) ou **Bibliotecas**. Depois de executar Qucs, a guia **Projectos** é ativada. Note no entanto, quando Qucs é iniciado pela primeira vez a lista de **Projetos** está vazia.

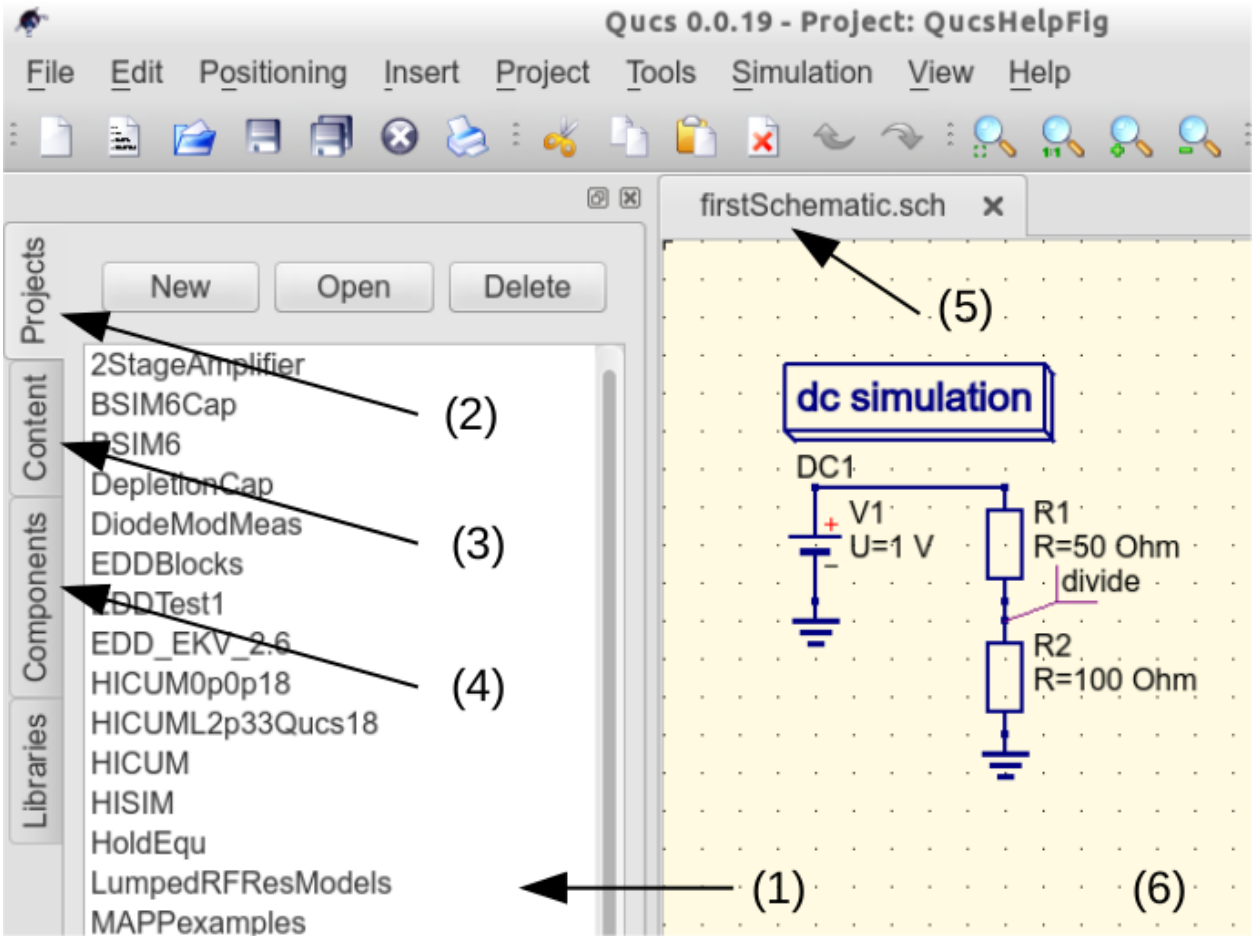


Figura 3 - janela principal Qucs com áreas de trabalho rotulados

Para inserir um novo projeto clique esquerdo sobre o botão **Novo** localizado no lado direito acima da janela (1). Esta ação faz uma caixa de diálogo abrir. Digite o nome de um projeto de Qucs na caixa fornecida, por exemplo, digite *QucsHelpFig* e clique sobre o botão **Ok**. Qucs, em seguida, cria um diretório de projeto no diretório ‘‘ ~/.qucs‘‘. No exemplo mostrado na Figura 3, isso é chamado **QucsHelpFig_prj**. Todos os arquivos pertencentes a este novo projeto é salvo dentro do **QucsHelpFig_prj** diretório. Na criação de um novo projeto é aberto imediatamente e nome é exibido na barra de título janela Qucs. No grupo de abas a esquerda é então ativado **Conteúdo** e o conteúdo do projeto aberto atualmente é exibido. Para salvar um documento aberto, clique sobre o botão **Salvar** (ou use o menu principal: **Arquivo** $\longrightarrow$ **Salvar**). Esta etapa inicia uma sequência que salva o documento exibido na área (6). Para completar a sequência de salvar o programa irá solicitar o nome do seu novo documento. Digite * firstSchematic , *ou outros nome apropriado e clique no botão **Okey** para completar a sequência de salvar.

Como um primeiro exemplo para ajudá-lo a começar com Qucs digite e execute o circuito DC simples mostrado na Figura 3. O circuito ilustrado é uma rede de divisor de tensão de dois resistor conectada a uma fonte de tensão DC valor fixo. Comece clicando sobre a aba **Componentes**. Esta ação faz com que uma caixa de combinação seja exibida, da qual um grupo de componentes pode ser escolhidos e os necessários componentes selecionados. Escolha o grupo de componentes **Lumped componentes** e clique no primeiro símbolo: **Resistor**. Em seguida, mova o cursor do mouse para área (6). Pressionar o botão direito do mouse gira o símbolo do **Resistor**. Da mesma forma, pressionar o botão esquerdo do mouse coloca o componente no esquemático no local em que o cursor do mouse está apontando. Repita este processo para todos os componentes mostrados na Figura 3. A fonte de tensão DC independente situa-se no grupo **Fontes**. O símbolo de terra pode ser encontrado no grupo **Lumped componentes** ou selecionado na barra de ferramentas Qucs. O ícone solicitando simulação DC está listado no grupo **Simulações**. Para editar os parâmetros do segundo resistor, clique duas vezes nele. Um diálogo aberto que permite que o valor do resistor a ser alterado; Digite *100 Ohm* no campo editar no lado direito e pressione enter.

Para conectar os componentes do circuito mostrados na Figura 3, clique no botão de barra de ferramentas de fio (ou use o menu principal: **Inserir** $\longrightarrow$ **Fio**). Mova o cursor para uma porta aberta de componente (indicada por um pequeno círculo vermelho na extremidade de um fio azul). Clicando nele começa o sequência do desenho do fio. Agora, mova o cursor de desenho para o ponto de extremidade de um fio (normalmente este é um círculo vermelho segundo anexado a um componente colocado) e clique novamente. Dois componentes estão agora ligados. Repita a sequência de desenho quantas vezes forem necessárias para conectar o circuito de exemplo. Se você quiser mudar a direção do canto de um fio, clique no botão direito do mouse antes de se mudar para um ponto de extremidade. Você também pode finalizar um fio sem clicar em uma porta aberta ou em um fio, basta clicar duas vezes no botão esquerdo do mouse.

Como um passo final antes da simulação DC rotular o nó, ou nós, os quais a tensão DC é necessária, por exemplo, o fio de ligação entre resistores **R1** e **R2**. Clique no botão de barra de ferramentas de rótulo (ou use o menu: ***Inserir** $\longrightarrow$ **Wire Label**). Agora clique sobre o fio escolhido. Um diálogo aberto permitindo que um nome de nó a ser inserido. Entre *divisão* e clique o **Ok** botão. Se você tiver desenhado o esquema de teste correctamente introduzido esquema deve a mesma aparência, ou ser semelhante a, mostrado na Figura 3.

Para iniciar a simulação DC clique sobre o botão **Simular** na barra de ferramentas (ou use o menu: *Simulação** $\rightarrow$ **Simular**). A janela de simulação abre e uma barra de deslizamento relata o progresso da simulação. Normalmente, tudo isso acontece tão rápido que a janela é vista brevemente na tela de PC (isso depende da velocidade do seu PC). Depois de terminar uma simulação com êxito Qucs abre uma janela de exibição de dados. Isso substitui a janela de entrada esquemática rotulada (6) na Figura 3. A seguir a aba **Componentes** $\longrightarrow$ **diagramas** é aberta. Isso permite que os resultados de simulação ser listado. Clique sobre o item **Tabular* e mova-o para a área de visualização, colocando-o clicando no botão esquerdo do mouse. Uma caixa de diálogo permite a seleção dos sinais que deseja listar, veja a Figura 4. O lado esquerdo do diálogo **Tabular** diálogo (chamado Editar Propriedades do Diagrama) é listado o nome do nó: **divisão.V**. Clique duas vezes nele e ele vai ser transferido para o lado direito do diálogo. Deixe o diálogo clicando o **Ok** botão. Os dados de tensão DC simulação para nó *divisão** agora devem ser listadas em uma caixa na janela de exibição de dados, com um valor de 0,666667 volts.

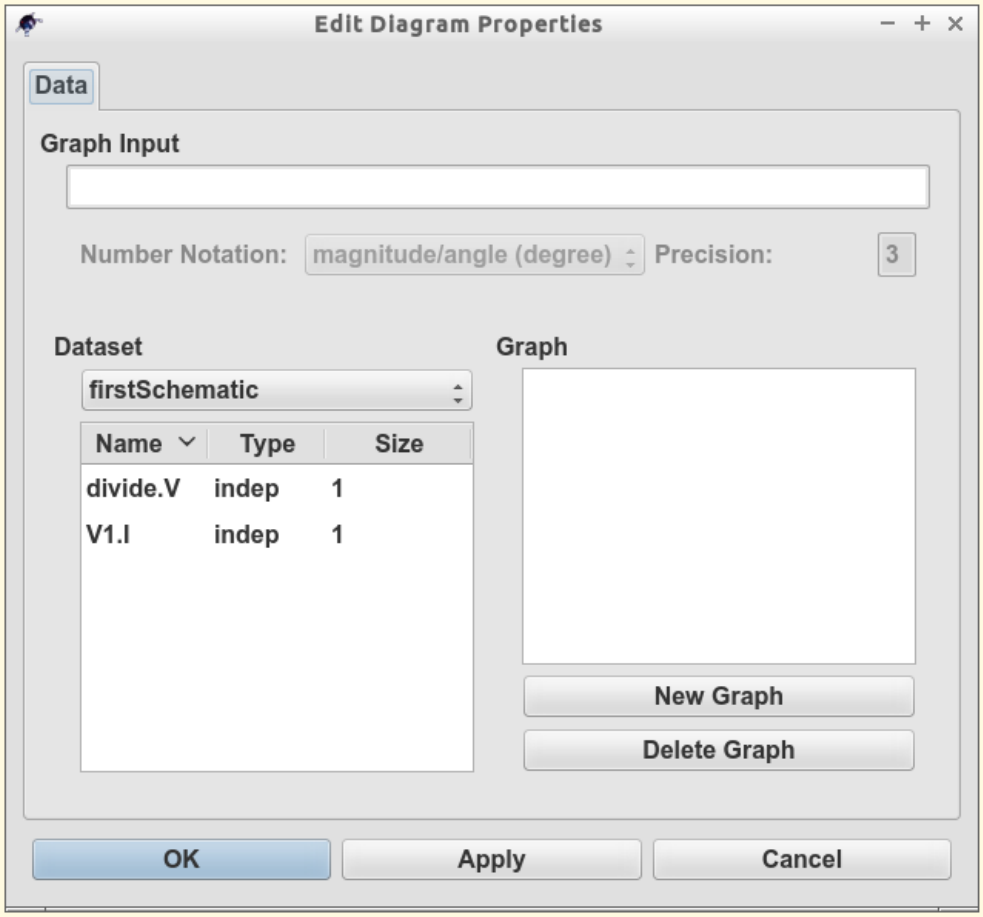


Figura 4 - Qucs dados exibir mostrando janela um caixa de diálogo **Tabular**

CAPÍTULO 3

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 que se obtenha o esquema elétrico exibido na Figura 2.

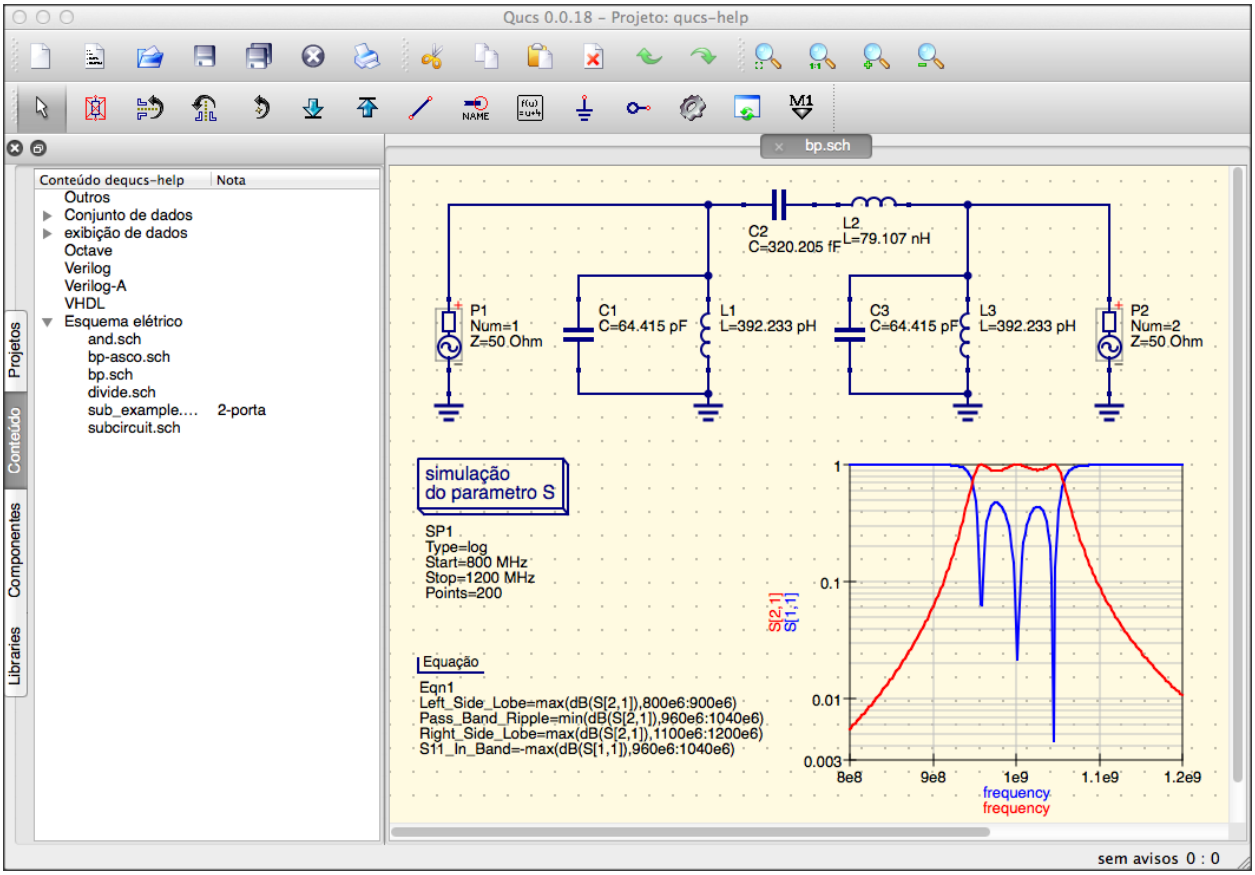


Figura 1 - Esquema elétrico inicial.

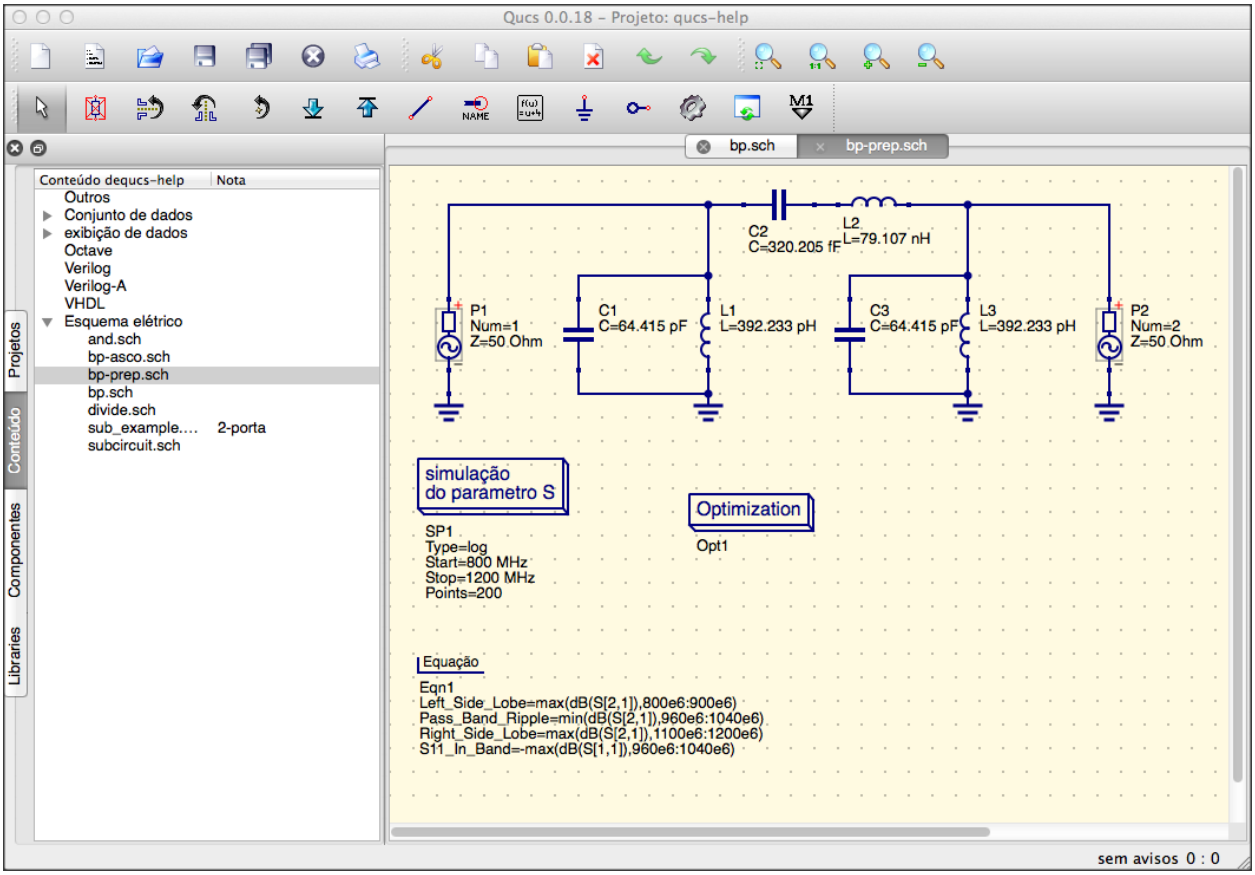


Figura 2 - Esquema elétrico preparado.

Agora, abra o componente otimização e selecione a aba Algorítmo. 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.

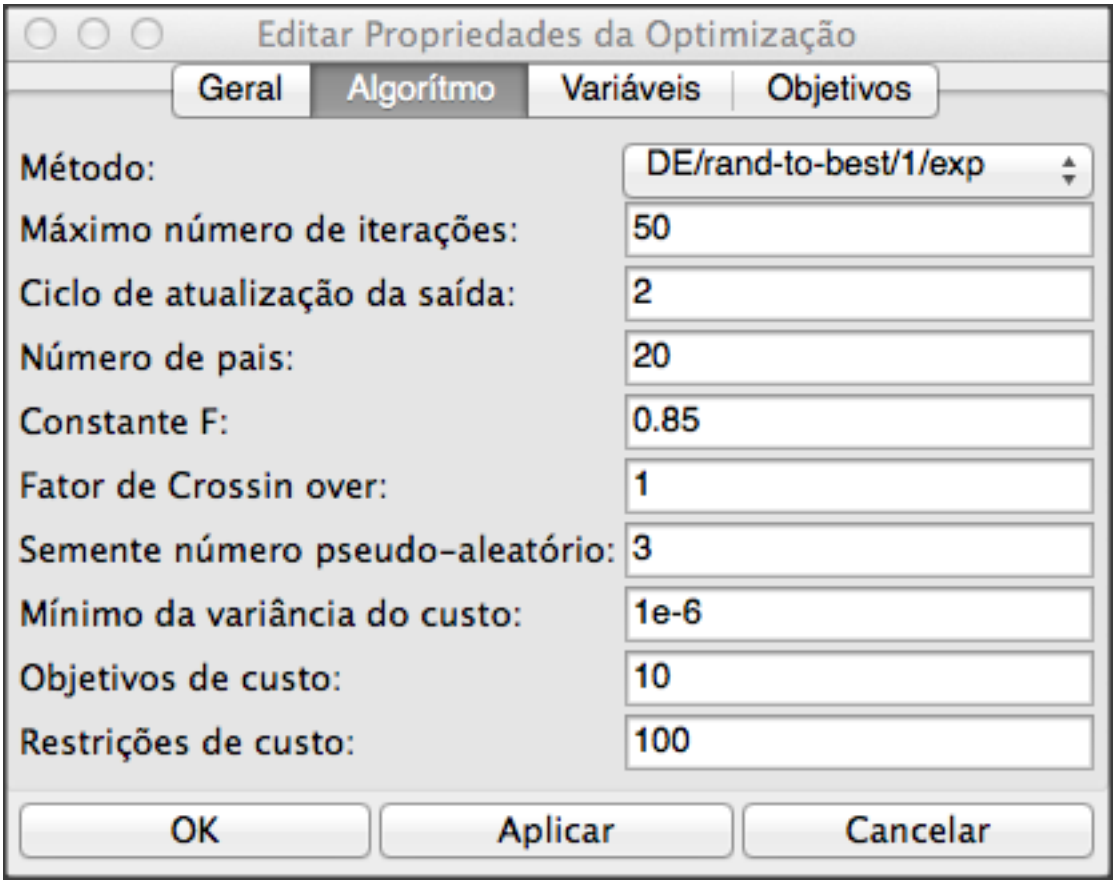


Figura 3 - Janela de otimização, opção Algorítimo.

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.

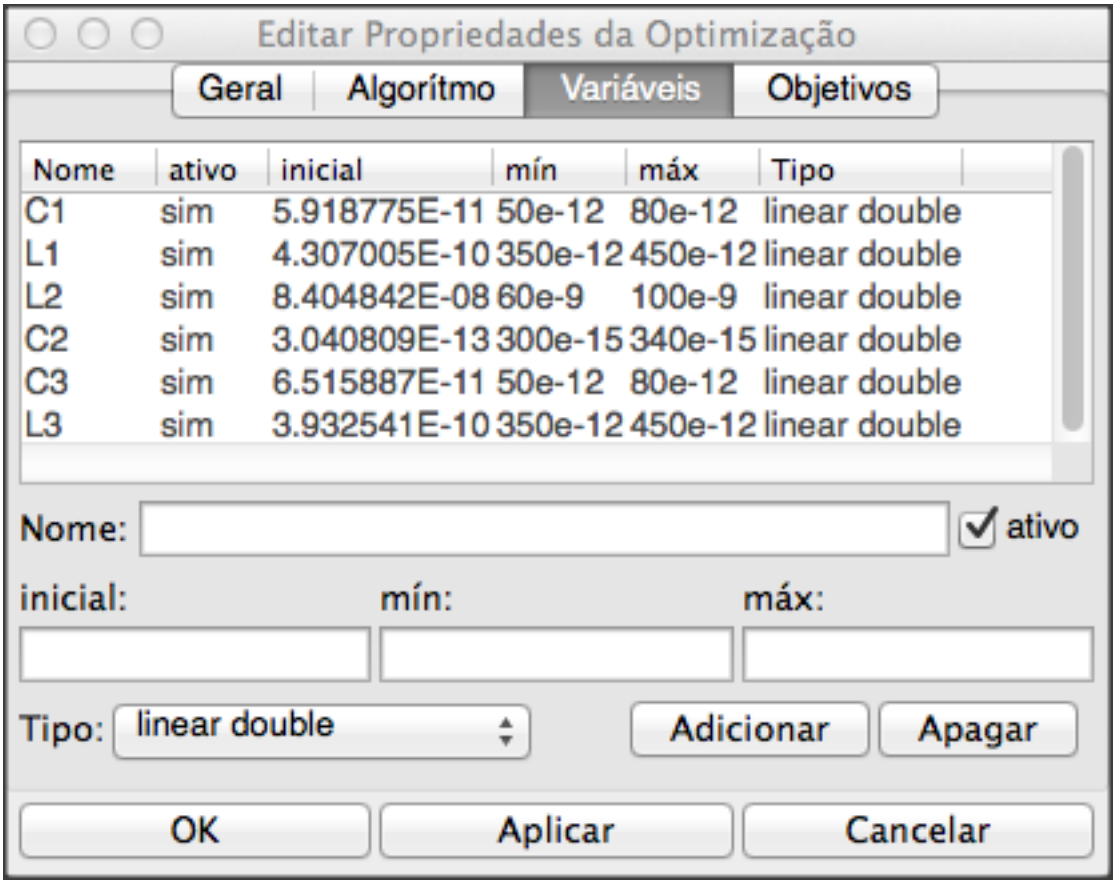


Figura 4 - Janela de otimização, opção Variáveis.

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.

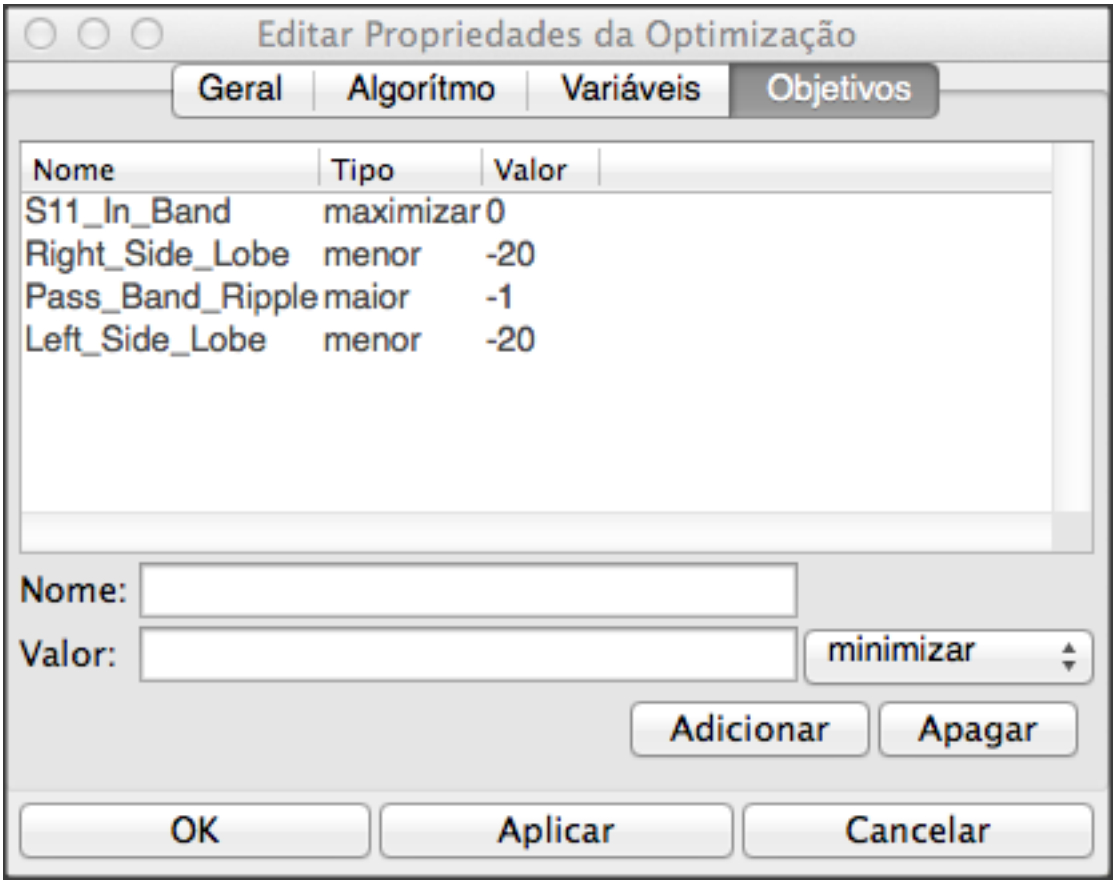


Figura 5 - Janela otimização, opção Metas.

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.

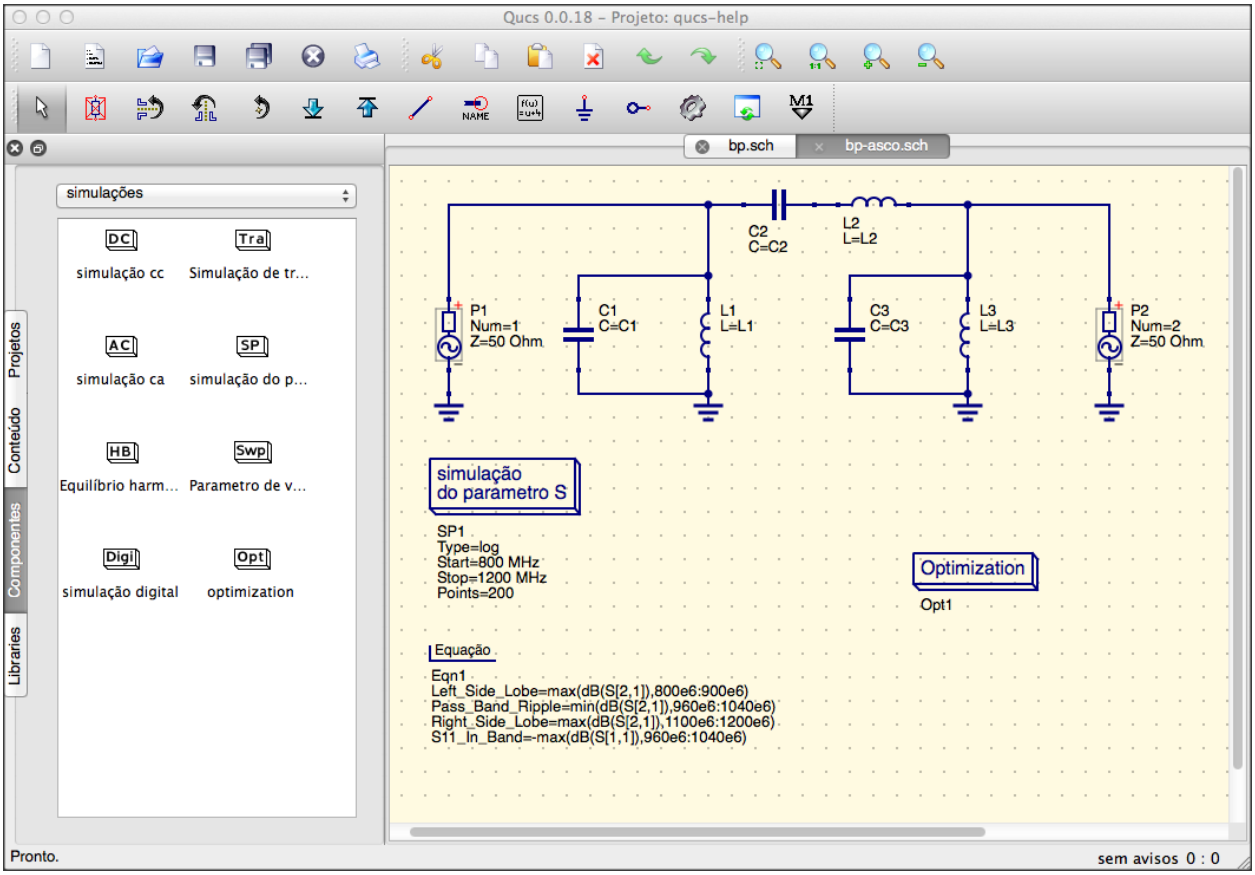


Figura 6 - Nova janela principal do Qucs.

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

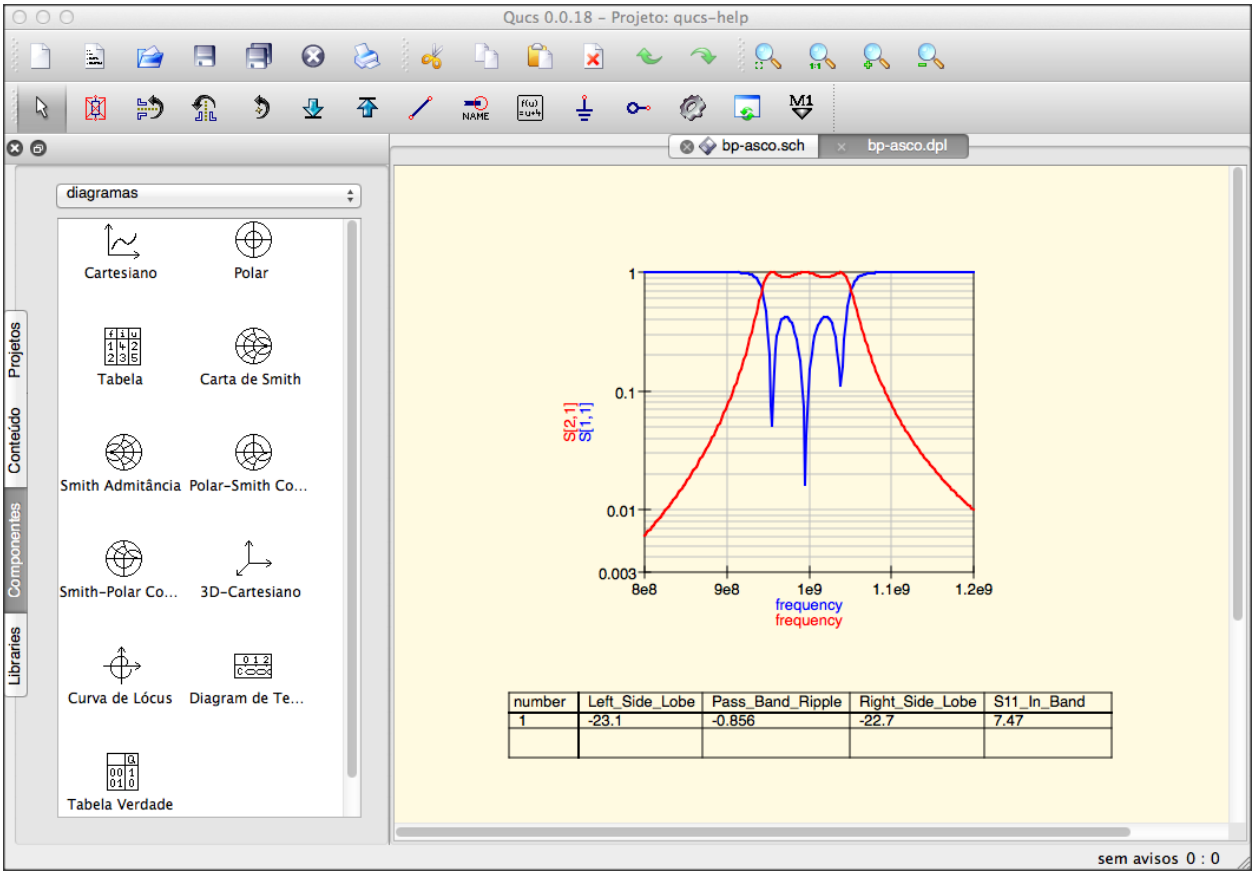


Figura 7 - Janela de resultados do Qucs.

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).

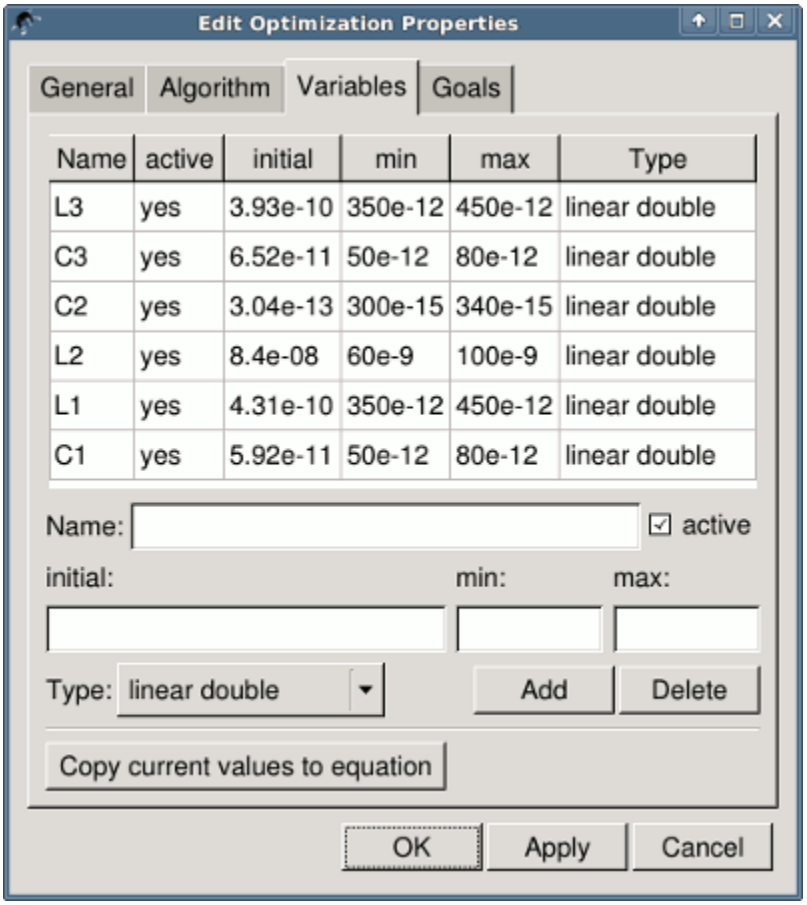


Figura 8 - Os melhores valores encontrados para o circuito.

Clicando no botão “Copiar valores atuais a equação”, um componente de equação definindo todas as variáveis de otimização com os valores da coluna “inicial” será copiado para a área de transferência e pode ser colado no esquemático depois de fechar a caixa de diálogo otimização. O esquema resultante será conforme mostrado na figura a seguir.

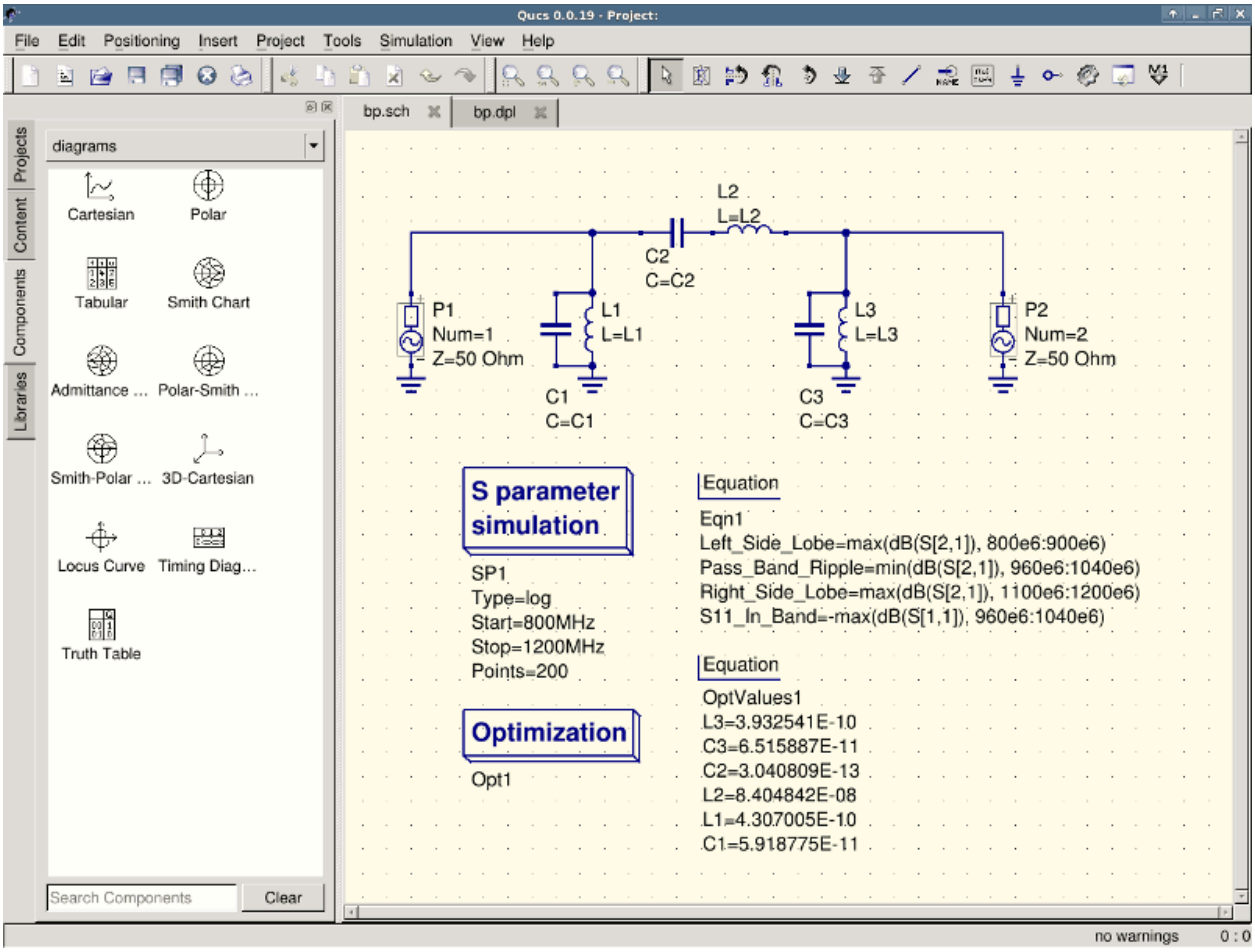


Figura 9 - diagrama esquemático com valores otimizados.

no caso que você precisa fazer mais modificações de esquema, o componente de otimização agora podem ser desabilitadas e os valores otimizados da equação colado serão usados.

Você pode alterar o número de figuras mostradas para os valores otimizados na caixa de diálogo otimização clicando sobre o cabeçalho da tabela “inicial” e selecionando o menu “Definir precisão”, conforme mostrado na figura a seguir.

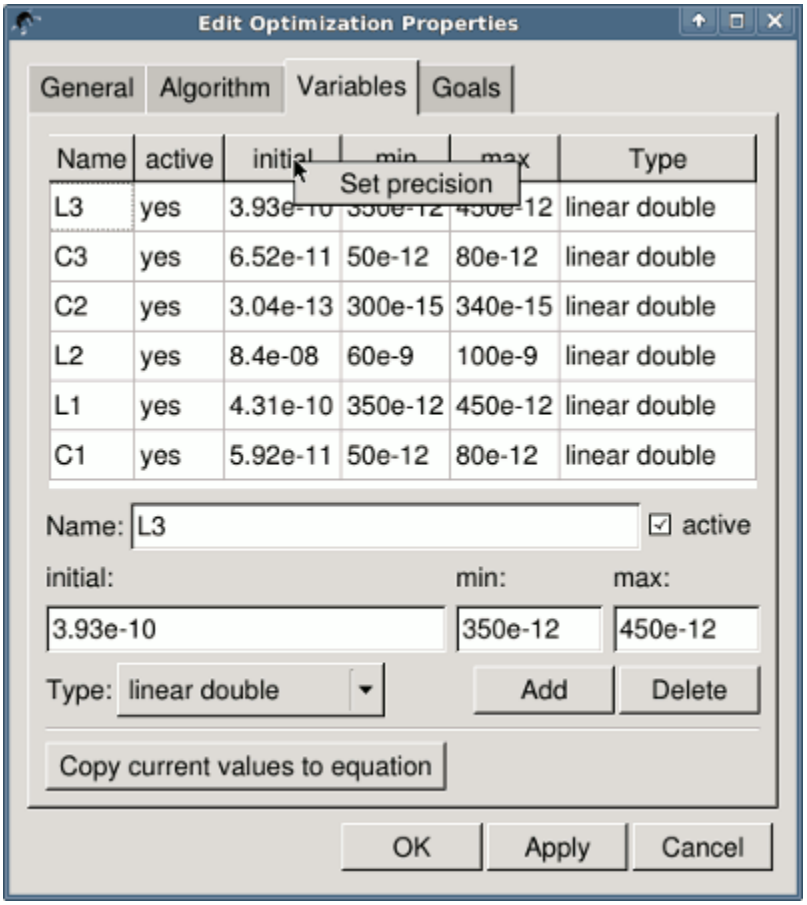


Figura 10 - Alterando a precisão exibida.

CAPÍTULO 4

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 apartir 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.

CAPÍTULO 5

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 rectângulo 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 valido para seleção feita com retângulo (veja ítem 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)

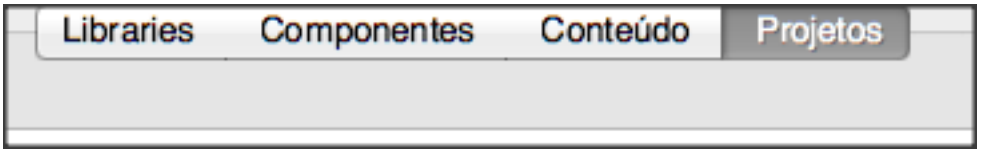
botão esquerdo do mouse	Define o ponto de início/fim do fio.
botão direito do mouse	Muda a direção do canto do fio (primeiro esquerdo/direito ou primeiro cima/baixo).
clique duplo botão direito do mouse	Termina um fio sem que seja feita uma conexão com outro fio ou porta.

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”	• abre arquivo selecionado
	renomear	• muda o nome do arquivo selecionado
	excluir	• apaga o arquivo selecionado
	“copy file”	• copy schematic file. Only file operations are performed. Dataset and dispaly settings in schematic properties are kept untouched.

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.
Cursores esquerda/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.
Cursores 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).

CAPÍTULO 6

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 todos 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. Entao 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.

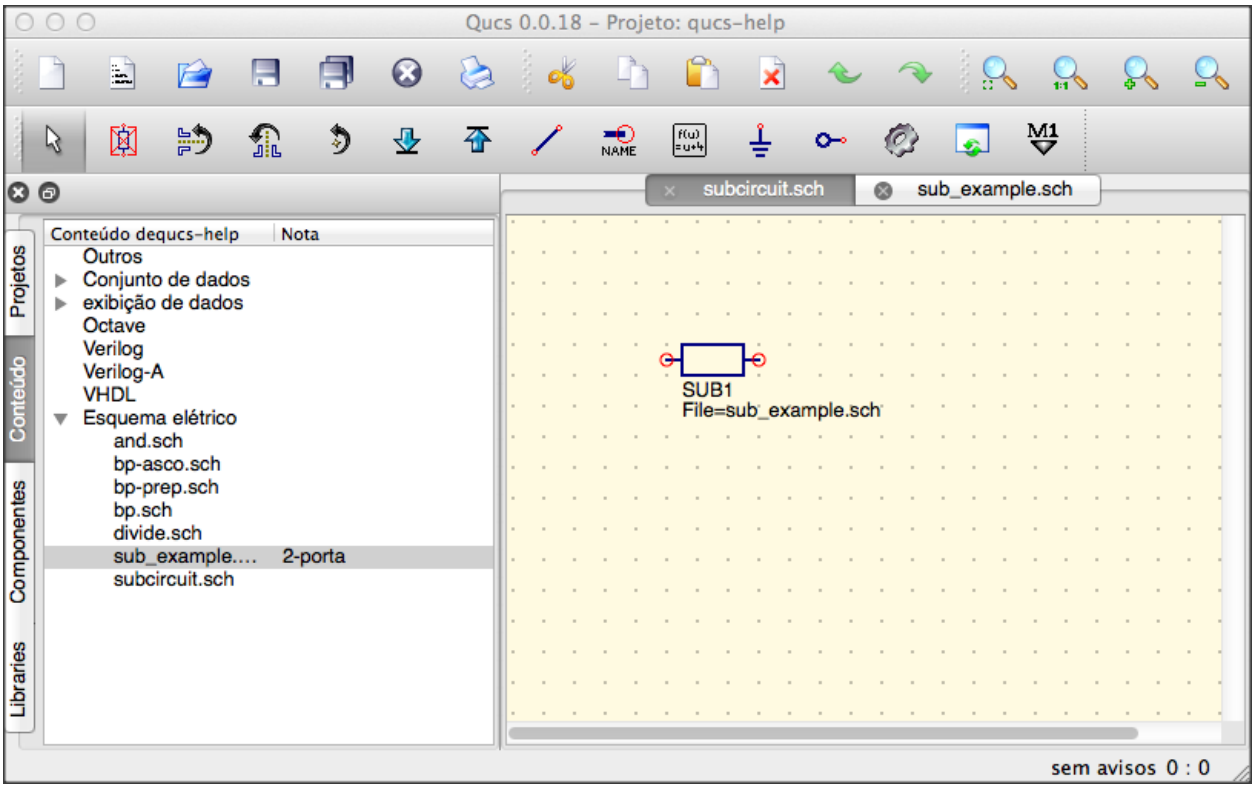


Figure 1 - Accesing a subcircuit

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 simbolo 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 desenhado 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 ediotr 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 parâmentros 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 parâmentros 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.

CAPÍTULO 7

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.

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

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

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!

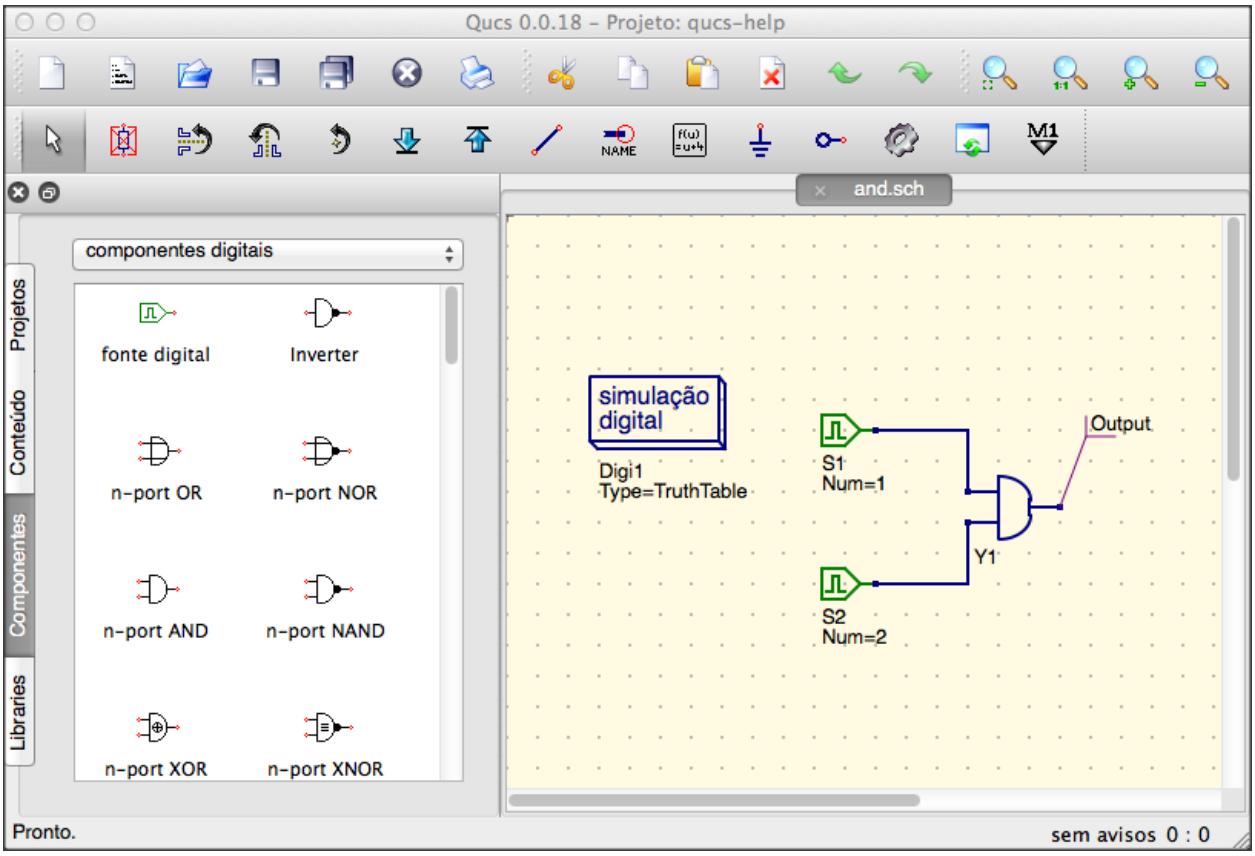


Figura 1 - Janela principal do Qucs

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 examplo, 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.)

CAPÍTULO 8

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

`!x`	Negação
`x&&y`	E
`x||y`	OU
`x^^y`	OU Exclusivo
`x?y:z`	Abreviação para expressão condicional `SE x ENTÃO y CASO CONTRÁRIO z`
`x==y`	Igual
`x!=y`	Não igual
`x<y`	Menor que
`x<=y`	Menor ou igual a
`x>y`	Maior que
`x>=y`	Maior ou igual a

8.2 Funções Matemáticas

8.2.1 Vetores e Matrizes: Criação

`eye(n)`	Cria uma matriz identidade n x n
`length(y)`	Retorna o comprimento do vetor `y`
`linspace(from,to,n)`	Vetor real com `n` componentes linearmente espaçados entre `` `from `` e `` ``to ``
`logspace(from,to,n)`	Vetor real com `n` componentes logarítmicamente espaçados entre `from` e `to`

8.2.2 Vetores e Matrizes: Funções Matriciais Básicas

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

8.2.3 Funções Matemáticas Elementares: Funções Básicas Real e Complexa

`abs(x)`	Valor absoluto, magnitude do número complexo
`angle(x)`	Fase angular em radianos de um número complexo. Sinônimo para arg()
`arg(x)`	Fase angular em radianos de um número complexo
`conj(x)`	Conjugado de um número complexo
`deg2rad(x)`	Converte fase em graus para radianos
`hypot(x,y)`	Função distância Euclidiana
`imag(x)`	Parte imaginária de um número complexo
`mag(x)`	Magnitude de um número complexo
`norm(x)`	Quadrado do valor absoluto de um vetor
`phase(x)`	Fase angular em graus de um número complexo
`polar(m,p)`	Transforma de coordenadas polares `m` e `p` em número complexo
`rad2deg(x)`	Converte fase em radianos para graus
`real(x)`	Parte real de um número complexo
`sign(x)`	Função sinal
`sqr(x)`	Quadrado (potência de dois) de um número
`sqrt(x)`	Raíz quadrada
`unwrap(p[,tol[,step]])`	Descobrir o ângulo `p` (radianos) – padrão `step` = 2pi, `tol` =pi

8.2.4 Funções Matemáticas Elementares: Funções Exponencial e Logarítmica

`exp(x)`	Função exponencial para a base e
`limexp(x)`	Função exponencial limitada
`log10(x)`	Logarítmo na base 10
`log2(x)`	Logarítmo na base 2
`ln(x)`	Logarítmo natural (base e)

8.2.5 Funções Matemáticas Elementares: Funções Trigonométricas

`cos(x)`	Função cosseno
`cosec(x)`	Função cossecante
`cot(x)`	Função cotangente
`sec(x)`	Funçao secante
`sin(x)`	Função seno
`tan(x)`	Função tangente

8.2.6 Funções Matemáticas Elementares: Funções Trigonométricas Inversas

`arccos(x)`	Arco cosseno (também conhecida como "cosseno inverso")
`arccosec(x)`	Arco cossecante
`arccot(x)`	Arco cotangente
`arcsec(x)`	Arco secante
`arcsin(x)`	Arco seno (também conhecida como "seno inverso")
`arctan(x[,y])`	Arco tangente (também conhecida como "tangente inversa")

8.2.7 Funções Matemáticas Elementares: Funções Hiperbólicas

`cosh(x)`	Cosseno hiperbólico
`cosech(x)`	Cossecante hiperbólico
`coth(x)`	Cotangente hiperbólico
`sech(x)`	Secante hiperbólico
`sinh(x)`	Seno hiperbólico
`tanh(x)`	Tangente hiperbólico

8.2.8 Funções Matemáticas Elementares: Funções Hiperbólicas Inversas

`arcosh(x)`	Arco cosseno hiperbólico
`arcosech(x)`	Arco cossecante hiperbólico
`arcoth(x)`	Arco cotangente hiperbólico
`arsech(x)`	Arco secante hiperbólico
`arsinh(x)`	Arco seno hiperbólico
`artanh(x)`	Arco tangente hiperbólico

8.2.9 Funções Matemáticas Elementares: Arredondamento

`ceil(x)`	Arredondar para o próximo maior inteiro
`fix(x)`	Truncar casas decimais de um número real
`floor(x)`	Arredondar para o próximo menor inteiro
`round(x)`	Arredondar para o inteiro mais próximo

8.2.10 Funções Matemáticas Elementares: Funções Matemáticas Especiais

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

8.2.11 Análise de Dados: Estatística Básica

`avg(x[,range])`	Média do vetor `x`. Se intervalo dado `x` deve ter uma dependência de dados único
`cumavg(x)`	Média acumulativa dos elementos de um vetor
`max(x,y)`	Retorna o maior dos valores entre x e y
`max(x[,range])`	Máximo do vetor `x`. Se intervalo dado `x` deve ter uma dependência de dados único
`min(x,y)`	Retorna o menor de todos os valores entre x e y
`min(x[,range])`	Mínimo do vetor `x`. Se intervalo dado `x` deve ter uma dependência de dados único
`rms(x)`	Raiz quadrada média de um vetor de elementos
`runavg(x)`	Corrida média dos elementos de um vetor
`stddev(x)`	Desvio padrão dos elementos de um vetor
`variance(x)`	Variância dos elementos de um vetor
`random()`	Número aleatório entre 0.0 e 1.0
`srandom(x)`	Semente aleatória dada

8.2.12 Análise de Dados: Operações Básicas

`cumprod(x)`	Produto acumulativo dos elementos de um vetor
`cumsum(x)`	Soma acumulativa dos elementos de um vetor
`interpolate(f,x[, n])`	Interpolação spline de vetor `f` usando `n``pontos equidistantes de ``x`
`prod(x)`	Produto dos elementos de um vetor
`sum(x)`	Soma dos elementos de um vetor
`xvalue(f,yval)`	Retorna x-valor mais próximo de `yval` no vetor de dependência única `f`
`yvalue(f,xval)`	Retorna o valor y mais próximo `xval` no vetor de dependência única `f`

8.2.13 Análise de Dados: Diferenciação e Integração

`ddx(expr,var)`	Derivada matemática da expressão expr com relação a variável var
`diff(y,x[,n])`	Diferencia o vetor `y` em relação ao vetor `x` `n` vezes. Padrão `n` = 1
`integrate(x,h)`	Vetor integração x assumindo numericamente uma constante de tamanho passo h

8.2.14 Análise de Dados: Processamento de Sinais

`dft(x)`	Transformada Discreta de Fourier do vetor x
`fft(x)`	Transformada Rápida de Fourier do vetor x
`fftshift(x)`	Embaralha os valores FFT de vetor `x` para mover a frequência 0 ao centro do vetor
`Freq2Time(V,f)`	Transformada Discreta Inversa de Fourier da função V(f) interpretada fisicamente
`idft(x)`	Transformada Discreta Inversa de Fourier de um vetor x
`ifft(x)`	Transformada Rápida Inversa de Fourier do vetor x
`kbd(x[,n])`	Janela de derivada Kaiser-Bessel
`Time2Freq(v,t)`	Transformada Discreta de Fourier da função v(t) 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)
`rtoy(x[,zref])`	Converte coeficiente de reflexão para admitância; a referência padrão `zref` = 50 ohms
`rtoz(x[,zref])`	Converte coeficiente de reflexão para impedância; a referência padrão `zref` = 50 ohms
`ytor(x[,zref])`	Converte admitância para coeficiente de reflexão; a referência padrão `zref` = 50 ohms
`ztor(x[,zref])`	Converte impedância para coeficiente de reflexão; a referência padrão `zref` = 50 ohms

8.3.3 Conversões de Matrizes de N-Portas

`stos(s,zref[,z0])`	Converte matriz de parâmetro S para matriz de parâmetro S com diferente 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)`	Converte uma matriz de duas portas: `from` e `to` são 'Y', 'Z', 'H', 'G', 'A', 'S' e '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])`	Círculo(s) com ganho de potência disponível constante Ga no plano da fonte
`GpCircle(s,Gp[,arcs])`	Círculos(s) com operação de ganho de potência constante Gp no plano de carga
`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])`	Figura(s) de Ruído `F` círculos
`PlotVs(data,dep)`	Retorna os dados selecionados a partir de `data` : a dependência `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)`	Fator de estabilidade dos parâmetros S de uma matriz de duas portas
`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 <code>M</code>

`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
nodename.V	Tensão DC no nó *nodename*
name.I	Tensão AC no nó *nodename*
nodename.v	Tensão AC no nó *nodename*
name.i	Tensão de ruído AC no nó *nodename*
nodename.vn	Tensão de ruído AC no nó *nodename*
name.in	Tensão transiente no nó *nodename*
nodename.Vt	Tensão transiente no nó *nodename*
name.It	Corrente transiente através do componente *name*

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*arctan(1) = 3.14159...
e	Euler = 2.71828...
kB	Carga elementar = 1.6021765e-19 C
q	Carga elementar = 1.6021765e-19 C

CAPÍTULO 9

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
\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

Letras Gregas maiúsculas

Tag LaTeX	Unicode	Descrição
\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

Símbolos matemáticos

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

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 simultâneamente entrada e saída de um circuito de duas portas. Isto funciona de forma completamente semelhante aos passos mencionandos 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.

CAPÍTULO 11

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`

CAPÍTULO 12

Qucs File Formats

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

12.1 Schematic file format

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

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

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 >.

12.1.1 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.
- `<Script=name.m>` contains the file name of the octave script associated with this schematic.
- `<RunScript=0>` contains 1 if the octave script is executed after the simulation.
- `<showFrame=0>` specify if a frame is drawn and if so which size it is. valid values are 0 (do not show a frame), 1 (A5 landscape), 2 (A5 portrait), 3 (A4 landscape), 4 (A4 portrait), 5 (A3 landscape), 6 (A3 portrait), 7 (letter landscape) and 8 (letter portrait).
- `<FrameText0=NE555 sub-circuit model>`, `FrameText1=Draw by: anonymous`, `FrameText2=Date: 1984`, and `<FrameText3=Revision: 42>` specifiy the texts to be placed into the frame text boxes.

12.1.2 Symbol

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.

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

12.1.3 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 componete 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 (multiplos 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`).

12.1.4 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 inícial, 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.
- The next number is the distance between the wire starting point and the point where the label is set on the wire.
- 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.

12.1.5 Diagrams

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

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

Diagram line format:

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

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

Here is a list of known diagram types:

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

Graph line format:

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

Marker line format:

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

12.1.6 Paintings

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

Refers to “Shared file format” section below.

12.2 Library file format

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

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

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

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

12.2.1 Default symbol

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

Refers to “Shared file format” section below.

12.2.2 Component

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

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

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

12.3 Shared file format

12.3.1 Painting elements

A painting line can be found in:

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

A painting line has one of the following format:

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

12.3.2 Symbol definition

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

The `.ID` line has the following format:

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

Where:

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

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

The `.PortSym` line has the following format:

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

Where:

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

CAPÍTULO 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).

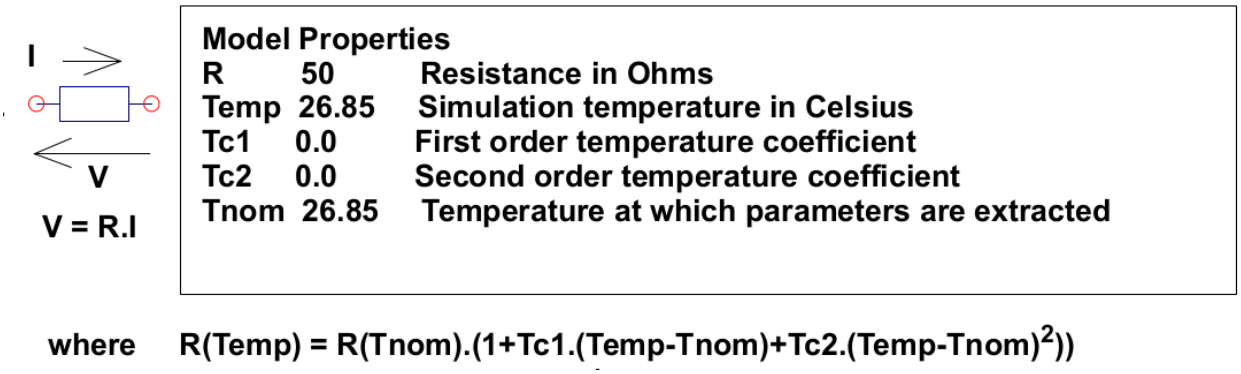


Figure 1 - Qucs built-in resistor model.

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.

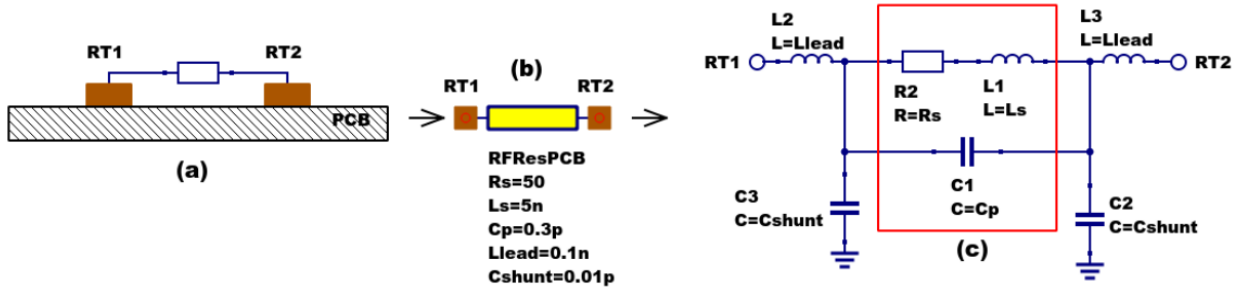


Figure 2 - PCB mounted resistor: (a) axial component mounting, (b) Qucs symbol and (c) equivalent circuit model.

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.

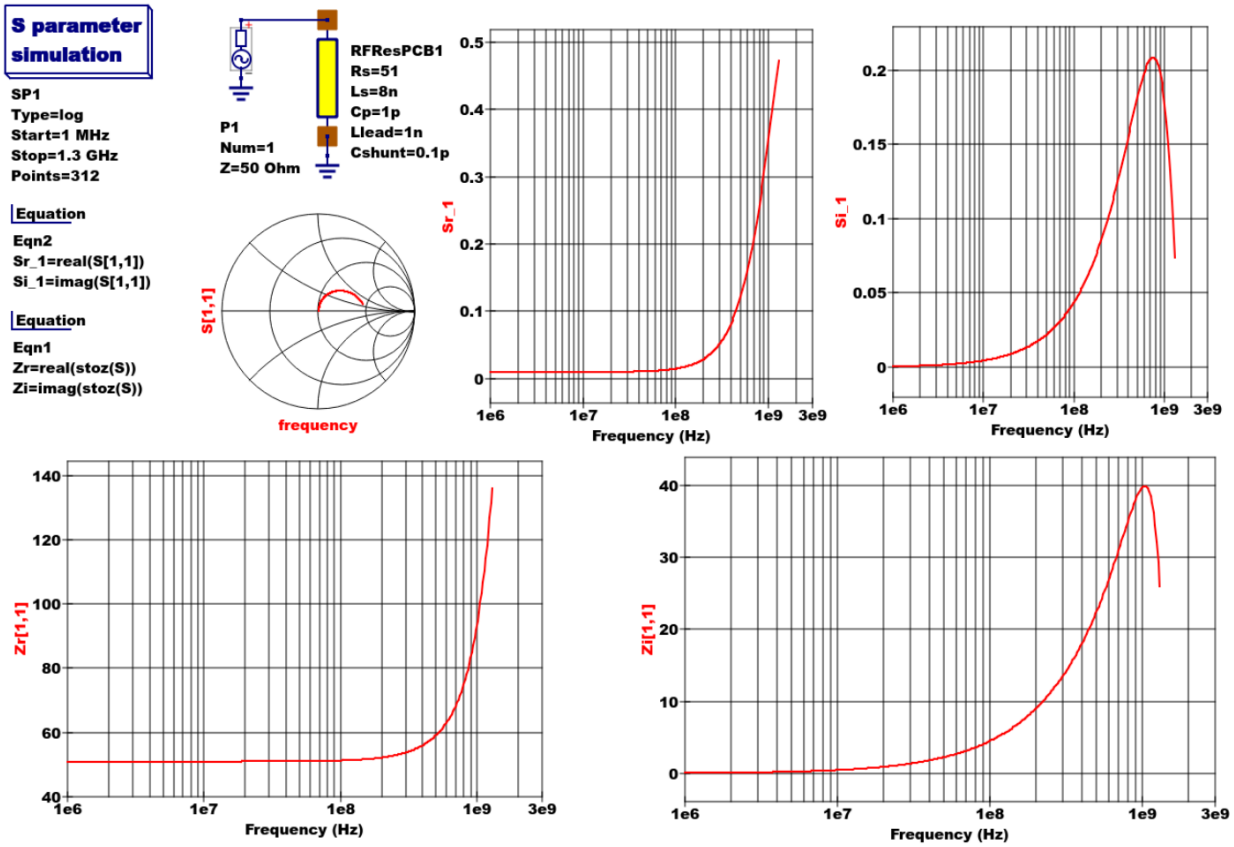


Figure 3 - Qucs S parameter simulation test circuit and plotted output data for a MF axial resistor: *Rs*=51Ω, *Ls*=8nH, *Cp*=1pF, *Llead*=1nH and *Cshunt*=0.1pF.

13.3 Analysis of the RF resistor model

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

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

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

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

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

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

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

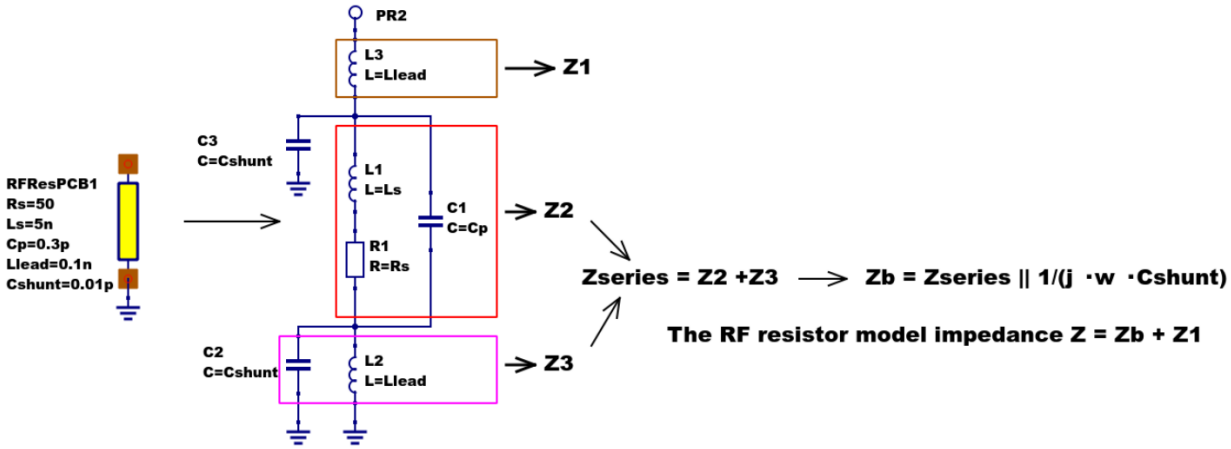


Figure 4 - RF resistor model rotated through 90 degrees and connected with one terminal grounded, similar to the test circuit in Figure. Sections of the model are shown grouped for calculation of the model impedance Z.

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.

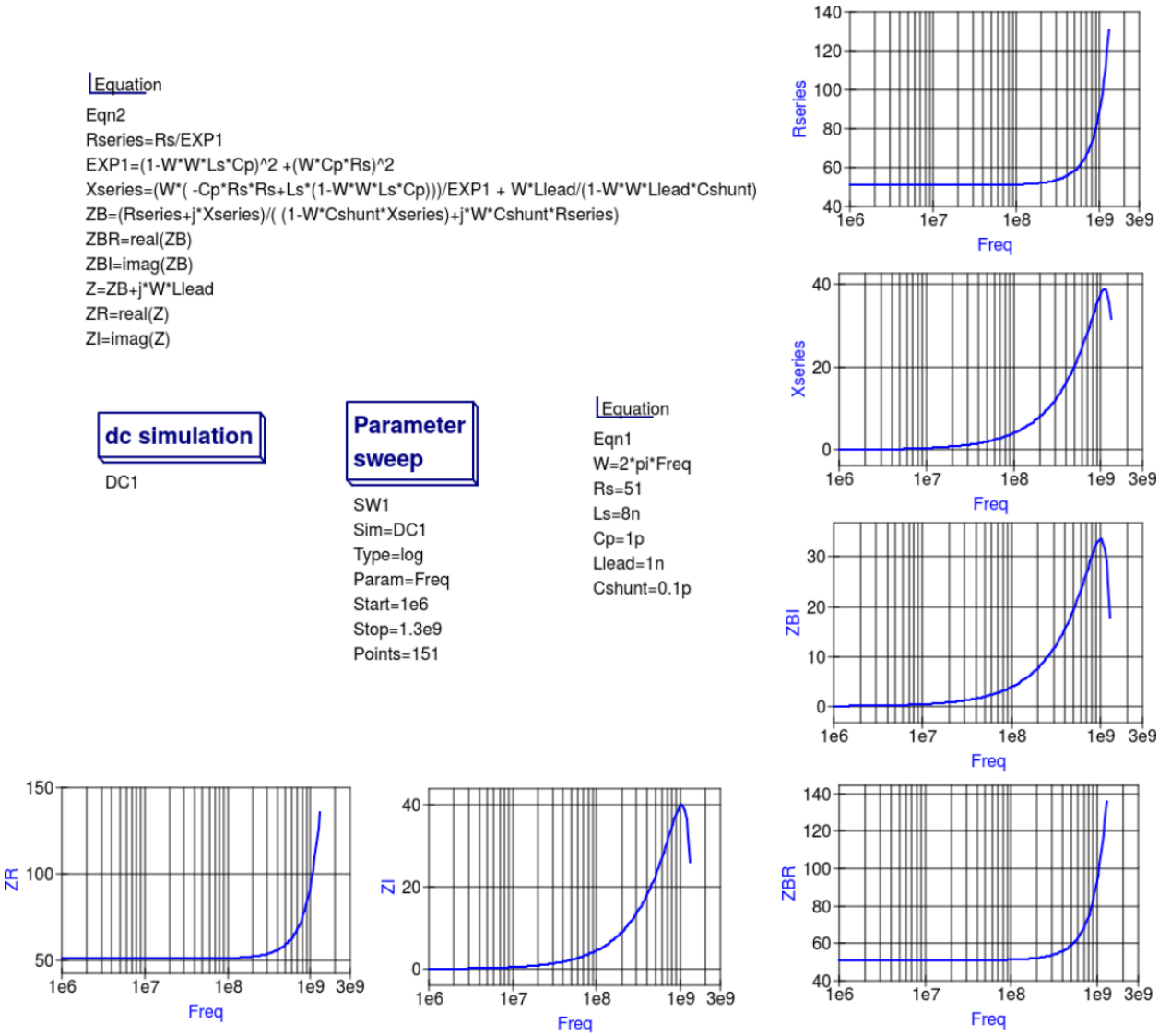


Figure 5- Theoretical analysis of RF resistance impedance Z using Qucs post processing facilities: note a dummy simulation icon, in this example DC simulation, is required to force Qucs to complete the analysis calculations.

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.

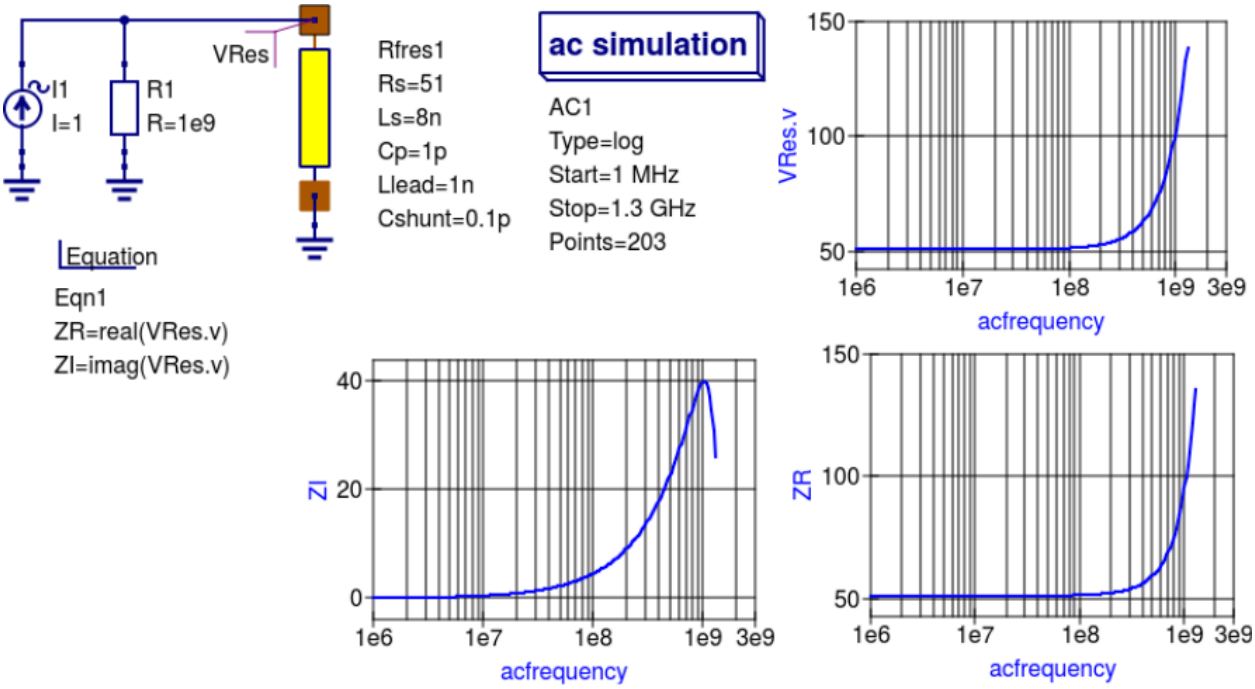


Figure 6 -A simple Qucs test circuit for demonstrating the use of an AC constant current source to measure electrical network impedance.

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.

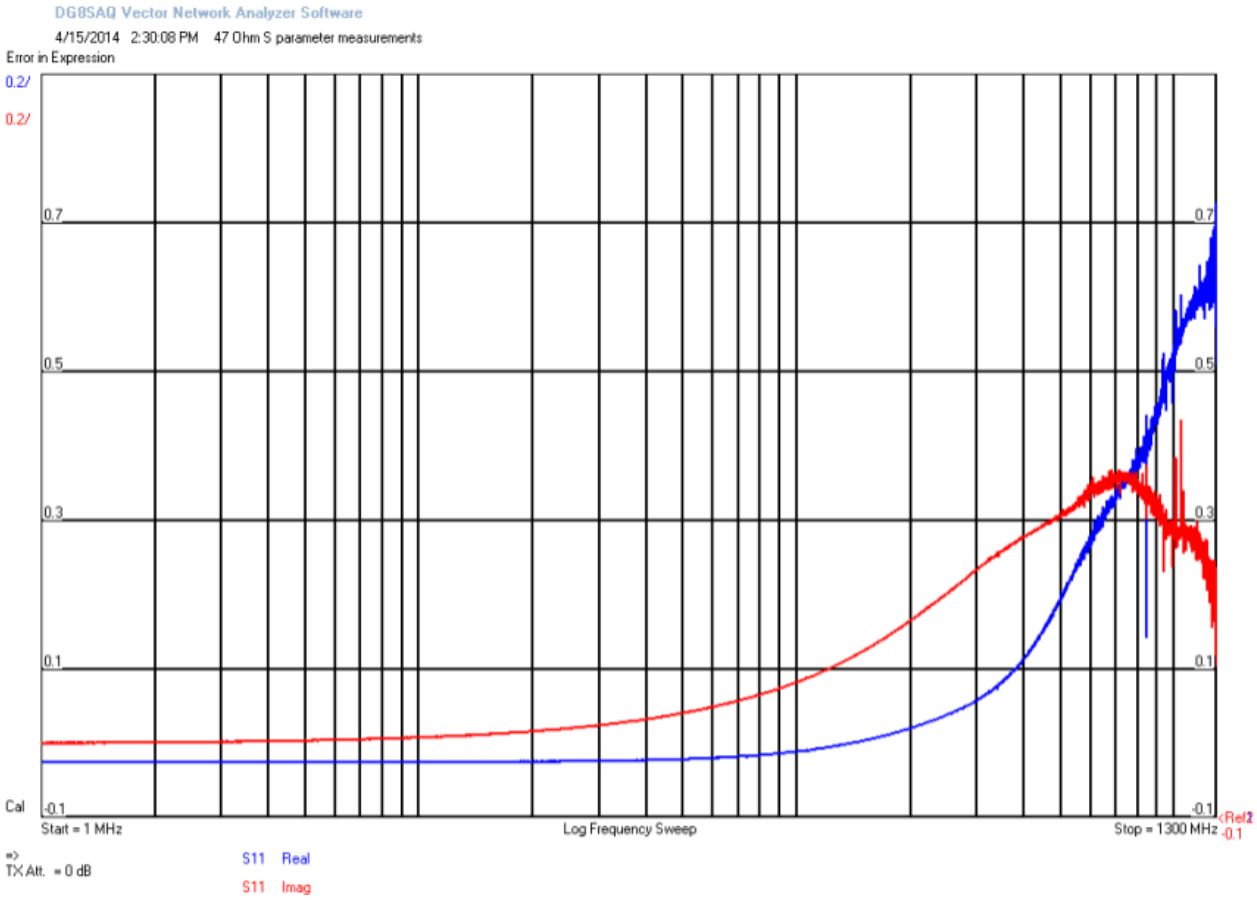


Figure 7 - DGSAQ Vector Network Analyser S parameter measurements for a 47 Ω axial RF resistor.

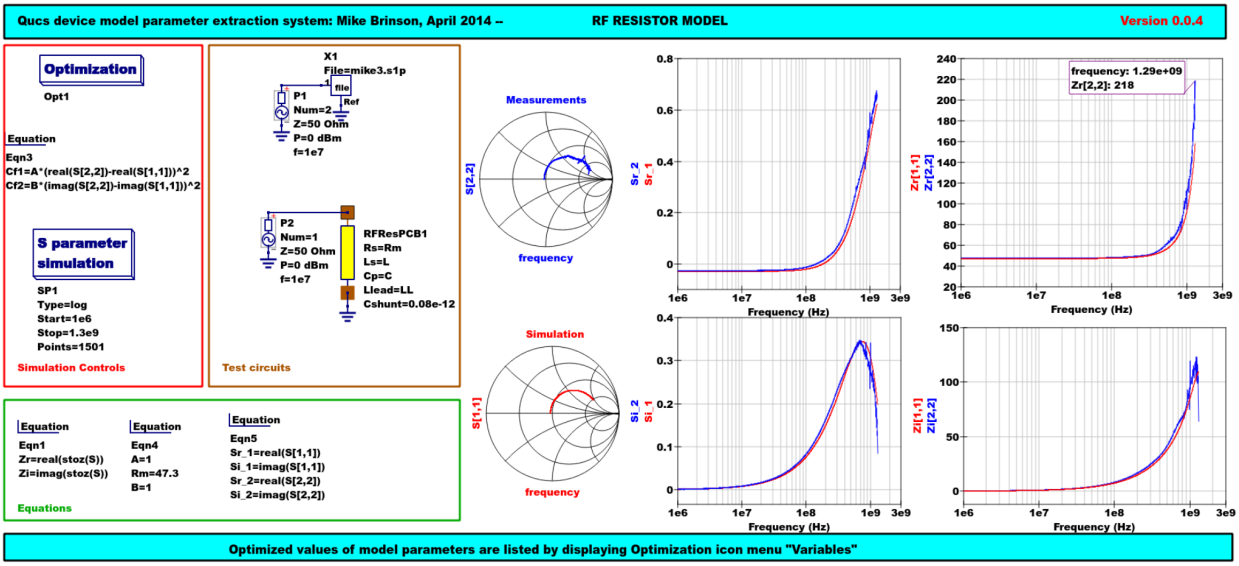


Figure 8 - Qucs device model parameter extraction system applied to a nominal 47 Ω resistor represented by the subcircuit model illustrated in Figure 2 (c). Fixed model parameter values: **Rs = Rm = 47.3** Ω, **CShunt = 0.08pF**; Optimised values: **Ls = L = 10.43nH**, **Llead = LL = 1.47nH**, **Cp =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

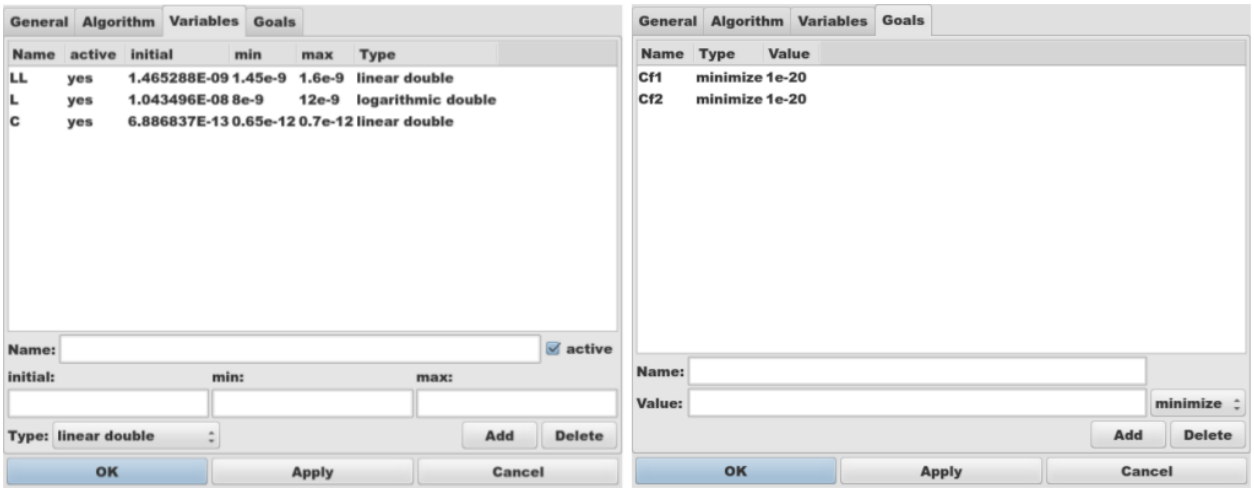


Figure 9 - Qucs Minimization Icon drop down menus: left ”Variables“ and right ”Goals‘‘.

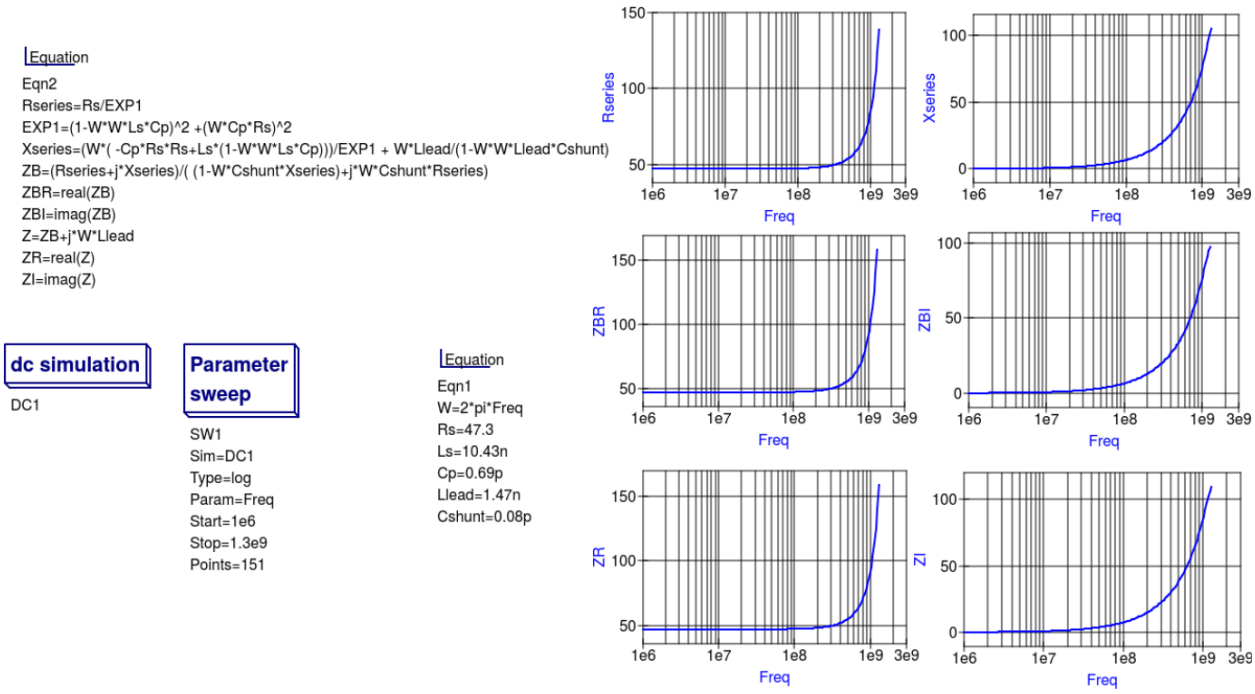


Figure 10 - Qucs simulation of nominal 47 Ω resistor based on theoretical analysis.l

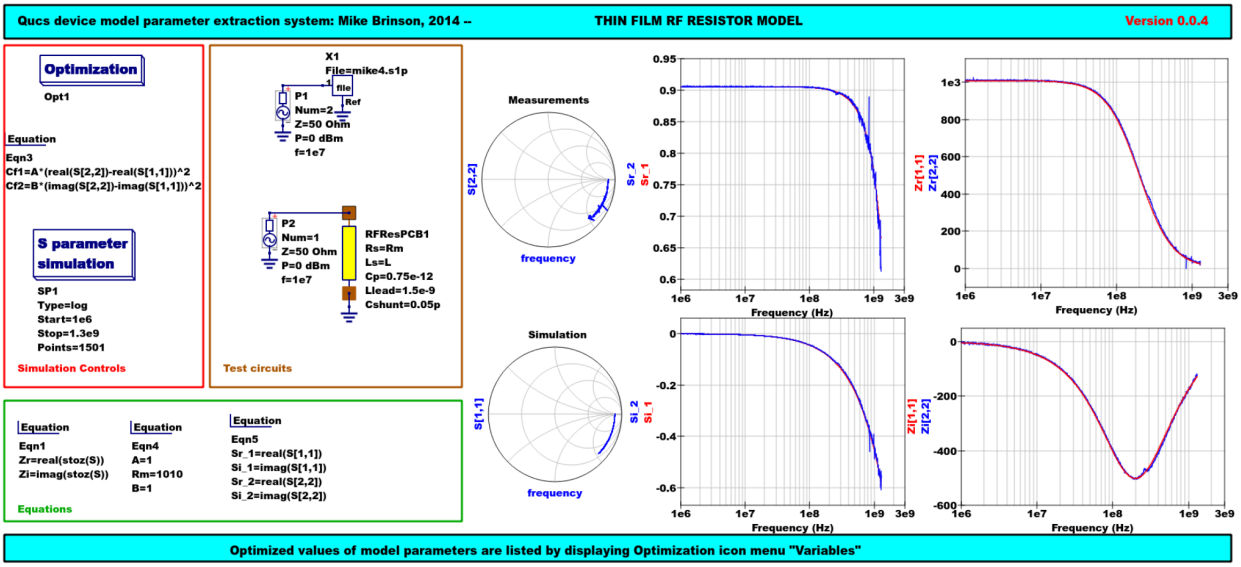


Figure 11 - Qucs device model parameter extraction system applied to a nominal 1000 Ω resistor represented by the subcircuit model illustrated in Figure 2(c).

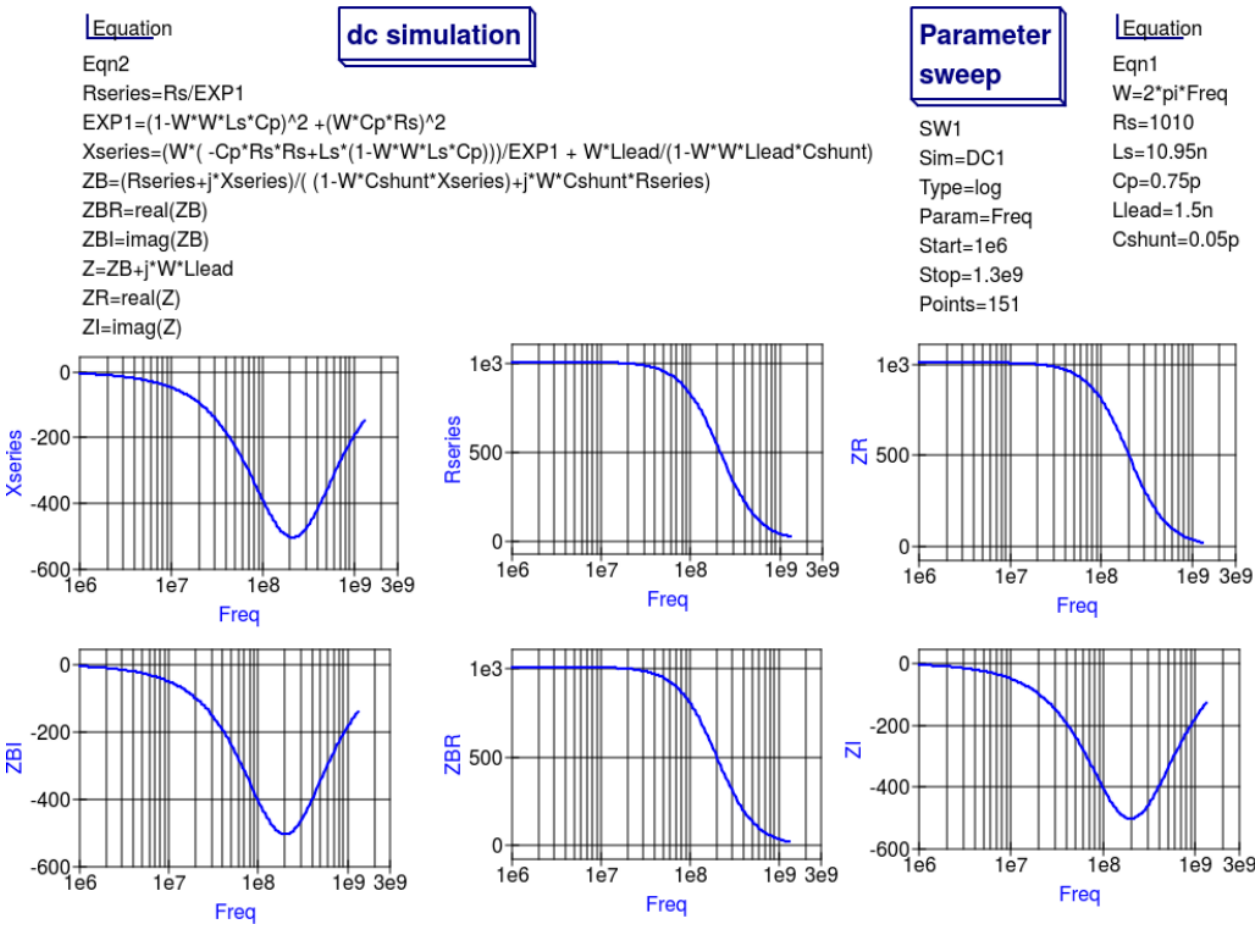


Figure 12 - Qucs simulation of nominal 1000 Ω resistor based on theoretical analysis.

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.

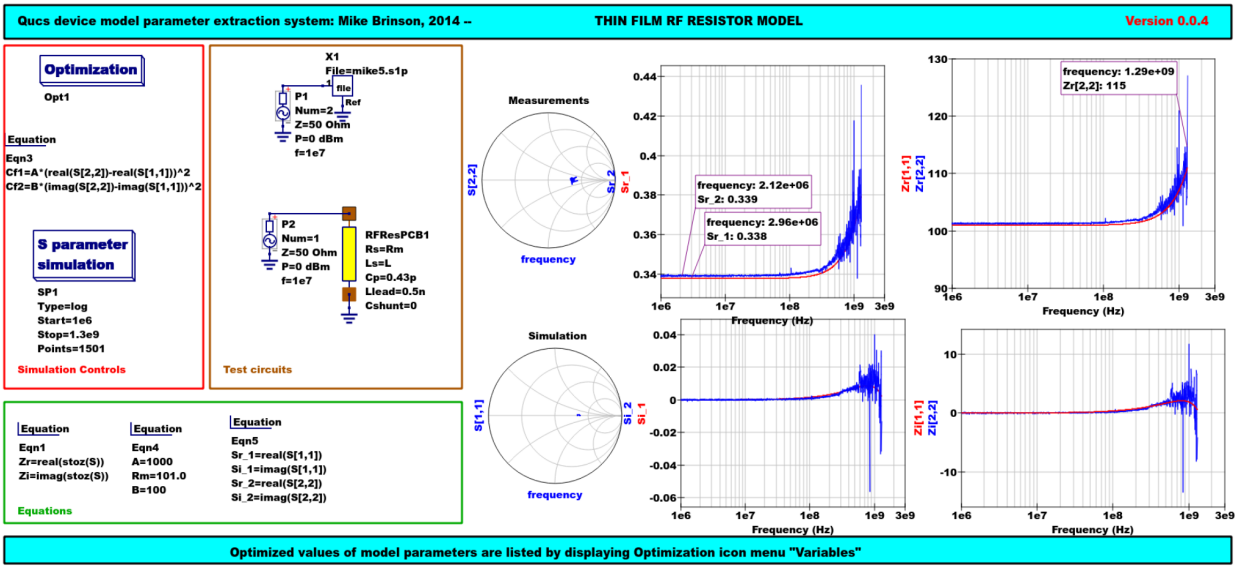


Figure 13 - Qucs device model parameter extraction system applied to a nominal 100 Ω SMD resistor represented by the subcircuit model illustrated in Figure 2 (c).

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");
parameter real Llead = 0.1e-9  from [1e-20 : inf)
```

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

```
 `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.

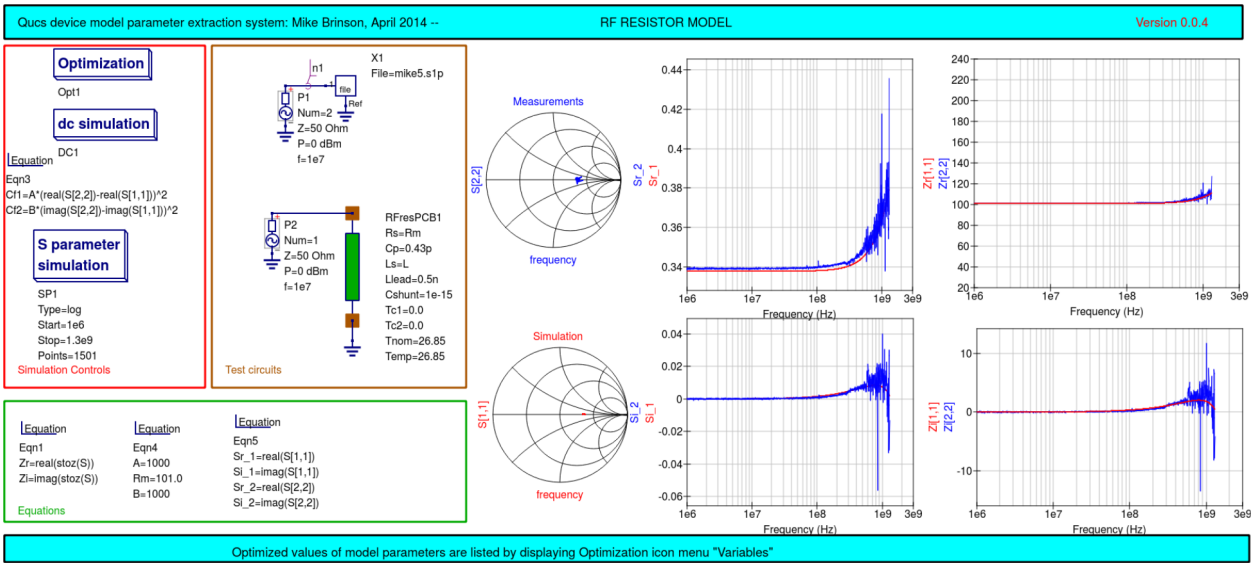


Figure 15 - Verilog-A models parameter data extraction for a 100 Ω axial thin film resistor. Fixed model parameter values: **Rs = Rm =101** Ω, **CShunt = 1e-15 F**, **Llead = LL = 0.5nH**, **Cp = C = 0.43pF**; Optimised values: **Ls = L = 3.99nH**. To reduce simulation time the ASCO cost variance was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

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

[5] Qucs release 0.0.18, or greater.