Qucs Help Documentation

Publicación 0.0.18

Qucs Team (2014)

20 de January de 2015

Contents:

CAPÍTULO 1

Empezando por el principio

Qucs (se pronuncia en inglés: kju:ks) es un simulador de circuitos con un interfaz gráfico para el usuario. Es capaz de llevar a cabo muchos tipos diferentes de simulación (p. ej. con parámetros de continua). Este documento le dará una breve descripción para saber como usar Qucs.

La primera vez que arranca Qucs, crea el directorio ".qucs" dentro de su directorio home. Todos los archivos se graban en este directorio o en alguno de sus subdirectorios. Después de que se haya cargado Qucs, verá la ventana principal con un aspecto similar al de la figura 1. En el lado derecho hay un área de trabajo (6) que contiene los esquemas, visores de datos, etc. Si usa las pestañas (5) sobre ese área, puede cambiar con rapidez entre los documentos que estén abiertos. A la izquierda de la ventana principal de Qucs hay otro área (1), cuyo contenido depende de la selección de las pestañas que se encuentran en su parte superior: "Proyectos" (2), "Contenido" (3) y "Componentes" (4). Después de ejecutar Qucs, la pestaña de "Proyectos" (2) se activa. Si es la primera vez que arranca este programa, este área está vacía porque no tiene ningún proyecto. Pulse el botón "Nuevo" que está encima de este área (1) para que se abra un diálogo. Escriba el nombre de su primer proyecto, p. ej. "primerProyecto" y pulse el botón "Crear". Qucs crea un directorio de proyecto dentro del directorio ~/.qucs, para este ejemplo se llamará "primerProyecto_prj"". Todos los archivos que pertenezcan a este nuevo proyecto se guardarán dentro de este directorio. El proyecto nuevo se abre inmediatamente (como se puede ver en la barra del título de la ventana) y las pestañas cambian a la posición de "Contenido" (3), en la que se muestra en contenido del proyecto abierto. Aún no tiene ningún documento, así que pulse guardar en la barra de herramientas (o use menú principal: Archivo ->Guardar) para guardar el documento sin título que ocupa el área de trabajo (6). Se le pedirá el nombre del nuevo documento. Escriba "primerEsquema" y pulse el botón "Guardar".

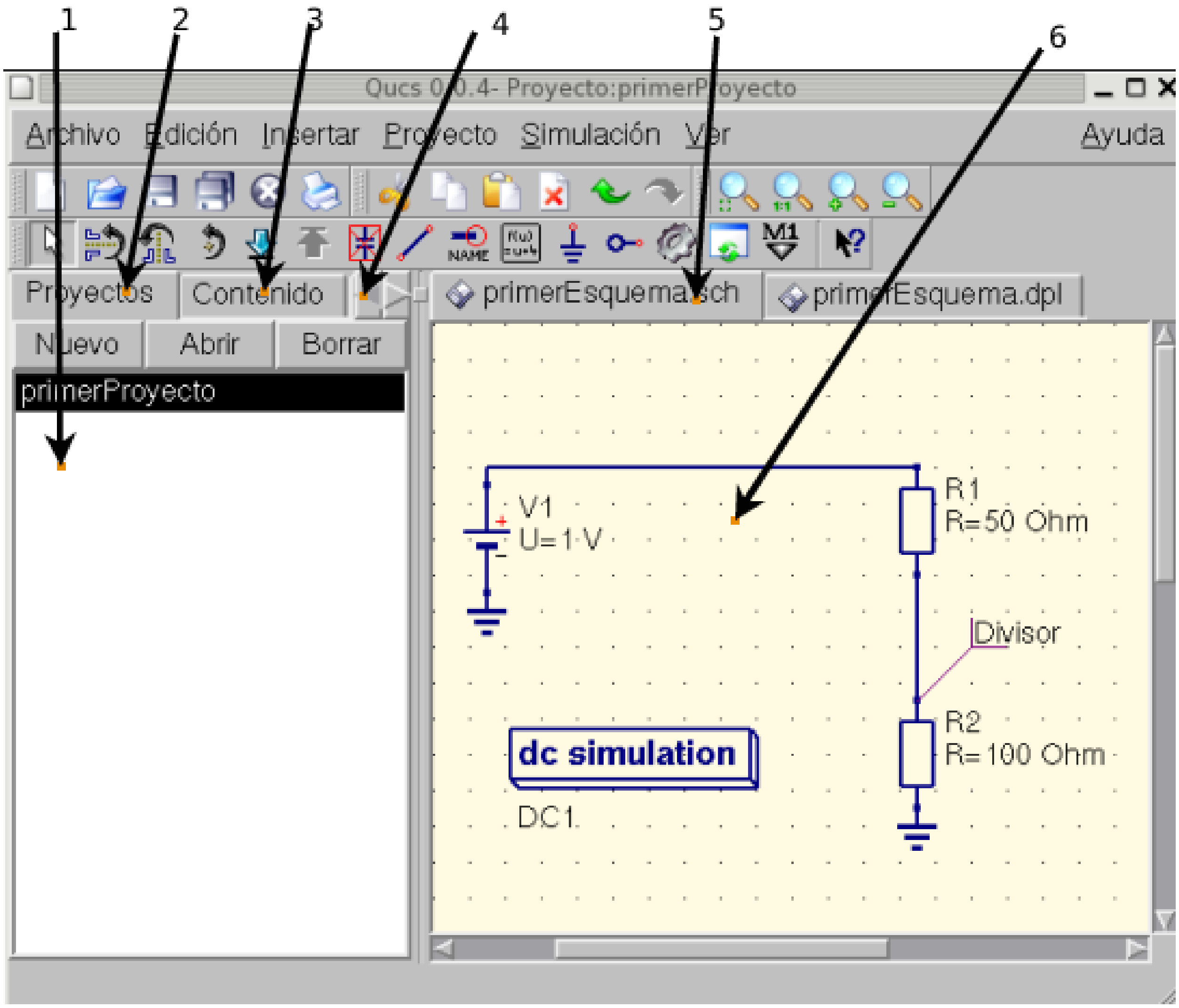


Figura 1 - Ventana principal de Qucs

Ahora queremos hacer una simulación simple de corriente continua, es decir, queremos analizar el circuito de la figura (1). Active la pestaña "Componentes" ((4) en la figura 1). Ahí vera una caja desplegable donde puede escojer un grupo de componentes, y debajo, los componentes del grupo seleccionado. Escoja "componentes sueltos" y pulse en el primer símbolo: "Resistencia". Si mueve el cursor del ratón al área de trabajo (6) llevará el dibujo del símbolo de una resistencia. Al pulsar el botón derecho del ratón rota el símbolo, al pulsar el botón izquierdo coloca el componente en el esquema. Repita este proceso para todos los compnentes tal y como muestra la figura 1. Las fuentes de tensión están el la pestaña de "fuentes", el símbolo de tierra se puede tomar de "componentes sueltos" o desde la barra de herramientas, la simulación que queremos está definida en los bloques grandes de simulación que están en la pestaña "simulaciones". Para editar los parámetros de la segunda resistencia, haga doble clic sobre ella. Aparece un diálogo en el que puede cambiar la resistencia. Escriba "100 Ohm" en el campo de edición que está en el lado derecho y pulse intro.

Para conectar los componentes, pulse el botón cablear de la barra de herramientas (o use el menú principal: Insertar ->Cable). Mueva el cursor sobre una conexión abierta (marcada por pequeños círculos rojos). Haciendo clic en ella comienza el cable. Ahora muévase al punto final y haga clic otra vez. Los componentes están conectados así. SI desea cambiar la dirección de la esquina del cable, haga clic con el botón derecho del ratón antes de fijar el punto final. También se puede finalizar un cable sin pulsar en una conexión abierta o en un cable: Simplemente haga doble clic con el botón izquierdo del ratón.

Finalmente, debe etiquetar el nodo en el que quiera que Qucs calcule la tensión. Pulse en el botón etiqueta de la barra de herramientas (o use el menú: Insertar ->Etiqueta de Cable). Ahora haga clic en el cable deseado. Se abrirá un diálogo para que pueda escribir el nombre del nodo. Escriba "divisor" y haga clic en el botón "Aceptar". Ahora el circuito debería parecerse al de la Figura 1.

Para comenzar la simulación pulse el botón simular de la barra de herramientas (o use el menú: Simulación -> Simular) Se abrirá una ventana que muestra el progreso. Cuando acabe con éxito la simulación se abrirá la pantalla de datos. Normalmente, esto ocurre tan rápido que sólo verá un pequeño parpadeo. Ahora tiene que colocar un diagrama para ver los resultados de la simulación. En la parte izquierda se habrá seleccionado automáticamente la pestaña de "diagramas" en los componentes. Pulse el elemento "Tabular", muévalo al área de trabajo y colóquelo pulsando el botón izquierdo del ratón. Debe abrirse un diálogo en el que puede escojer lo que el nuevo diagrama va a mostrar. En la zona de la izquierda verá el nodo que ha definido: "divisor". Haga doble clic en él y se transferirá a la zona de la derecha. Salga del diálogo pulsando el botón "Aceptar". Ahora debería ver el resultado de la simulación: 0.666667 voltios. Maravilloso, ¡Dése una palmadita en la espalda!

Getting Started with Digital Simulations

Qucs is also a graphical user interface for performing digital simulations. This document should give you a short description on how to use it.

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.

The digital sources *S1* and *S2* are the inputs, the node labeled as *Output* is the output. After performing the simulation, the data display page opens. Place the diagram *truth table* on it and insert the variable *Output*. Now the truth table of a two-port AND cell is shown. Congratulation, the first digital simulation is done!

|image0_ES|

Figura 1 - Ventana principal de Qucs

A truth table is not the only digital simulation that Qucs can perform. It is also possible to apply an arbitrary signal to a circuit and see the output signal in a timing diagram. To do so, the parameter *Type* of the simulation block must be changed to *TimeList* and the duration of the simulation must be entered in the next parameter. The digital sources have now a different meaning: They can output an arbitrary bit sequence by defining the first bit (low or high) and a list that sets the times until the next change of state. Note that this list repeats itself after its end. So, to create a 1GHz clock with pulse ratio 1:1, the list writes: 0.5ns; 0.5ns

To display the result of this simulation type, there is the diagram *timing diagram*. Here, the result of all output nodes can be shown row by row in one diagram. So, let's have fun...

2.1 VHDL File Component

More complex and more universal simulations can be performed using the component "VHDL file". This component can be picked up from the component list view (section "digital components"). Nevertheless the recommended usage is the following: The VHDL file should be a member of the project. Then go to the content list view and click on the file name. After entering the schematic area, the VHDL component can be placed.

The last entity block in the VHDL file defines the interface, that is, all input and output ports must be declared here. These ports are also shown by the schematic symbol and can be connected to the rest of the circuit. During simulation the source code of the VHDL file is placed into the top-level VHDL file. This must be considered as it causes some limitations. For example, the entity names within the VHDL file must differ from names already given to subcircuits. (After a simulation, the complete source code can be seen by pressing F6. Use it to get a feeling for this procedure.)

CAPÍTULO 3

Getting Started with Optimization

For circuit optimization, Qucs uses the ASCO tool (http://asco.sourceforge.net/). A brief description on how to prepare your schematic, execute and interprete the results is given below. Before using this functionality, ASCO must be installed on the computer.

Optimization of a circuit is nothing more than the minimization of a cost function. It can either be the delay or the rise time of a digital circuit, or the power or gain of an analog circuit. Another possibility is defining the optimization problem as a composition of functions, leading in this case to the definition of a figure-of-merit.

To setup a netlist for optimization two things must be added to the already existing netlist: insert equation(s) and the optimization component block. Take the schematic from Figure 1 and change it until you have the resulting schematic given in Figure 2.

|image0_ES|

Figure 1 - Initial schematic.

|image1_ES|

Figure 2 - Prepared schematic.

Now, open the optimization component and select the optimization tab. From the existing parameters, special attention should be paid to 'Maximum number of iterations', 'Constant F' and 'Crossing over factor'. Over- or underestimation can lead to a premature convergence of the optimizer to a local minimum or, a very long optimization time.

|image2_ES|

Figure 3 - Optimization dialog, algorithm options.

In the Variables tab, defining which circuit elements will be chosen from the allowed range, as shown in Figure 4. The variable names correspond to the identifiers placed into properties of components and **not** the components' names.

|image3_ES|

Figure 4 - Optimization dialog, variables options.

Finally, go to Goals where the optimization objective (maximize, minimize) and constraints (less, greater, equal) are defined. ASCO then automatically combines them into a single cost function, that is then minimized.

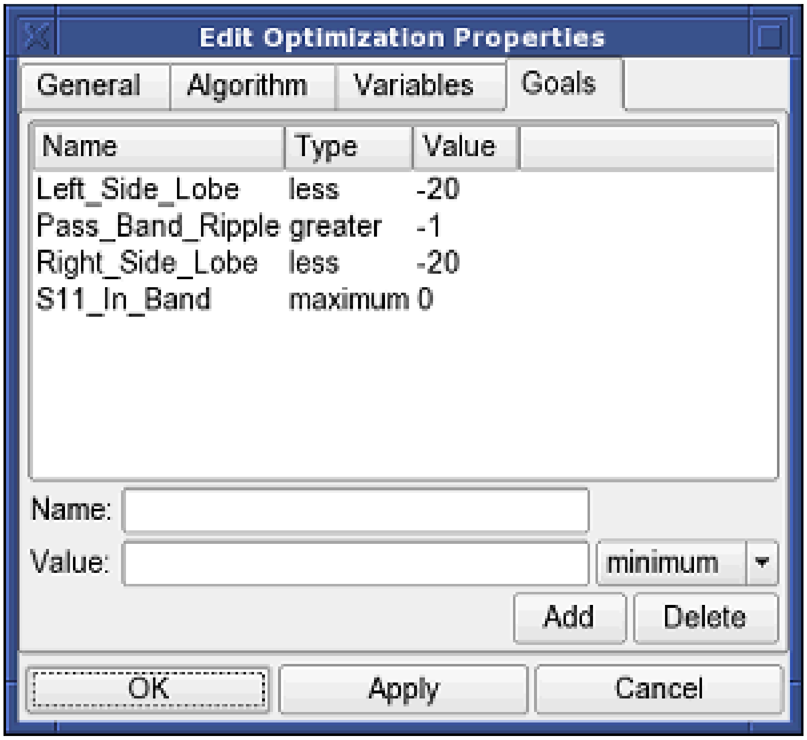


Figure 5 - Optimization dialog, goals options.

The next step is to change the schematic, and define which circuit elements are to be optimized. The resulting schematic is show in Figure 6.

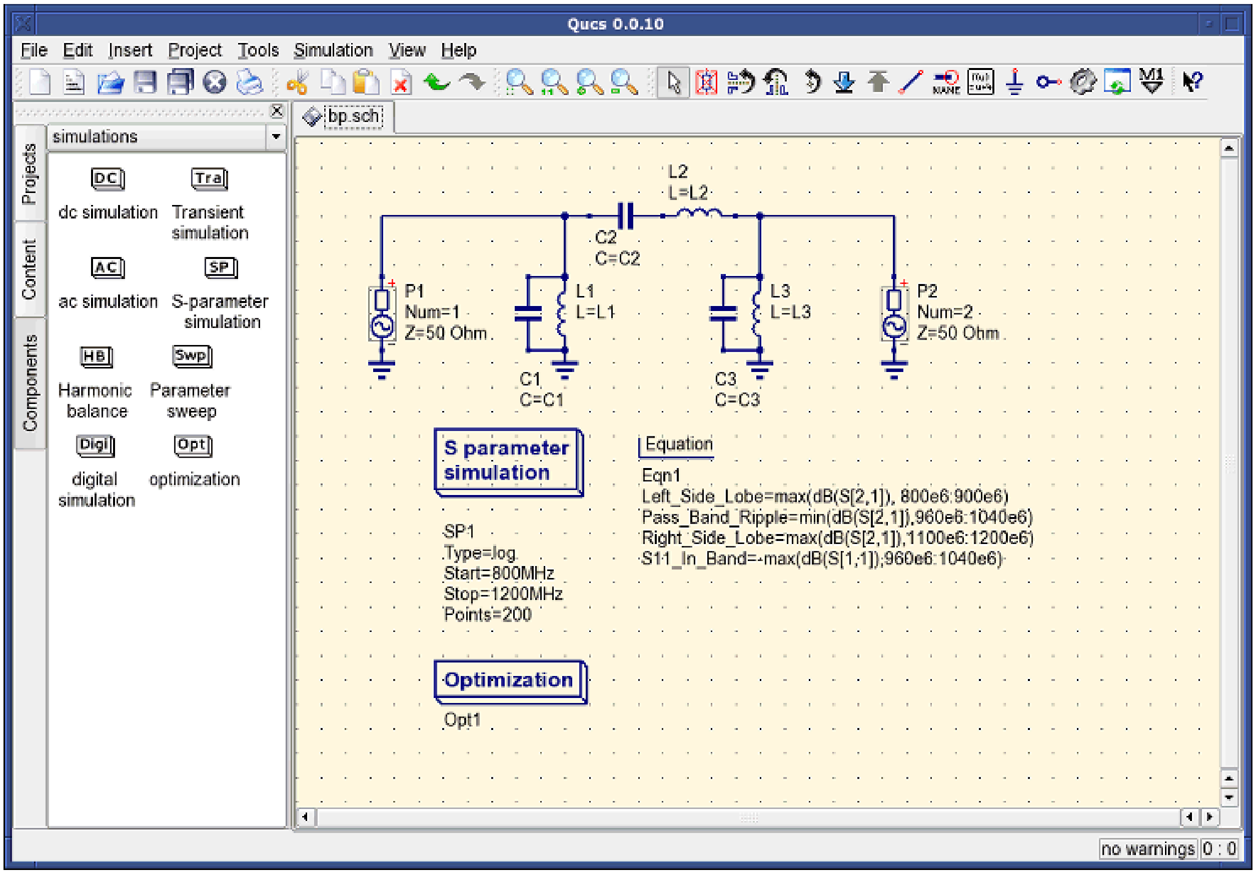


Figure 6 - New Qucs main window.

The last step is to run the optimization, i.e. the simulation by pressing F2. Once finished, which takes a few seconds on a modern computer, the best simulation results is shown in the graphical waveform viewer.

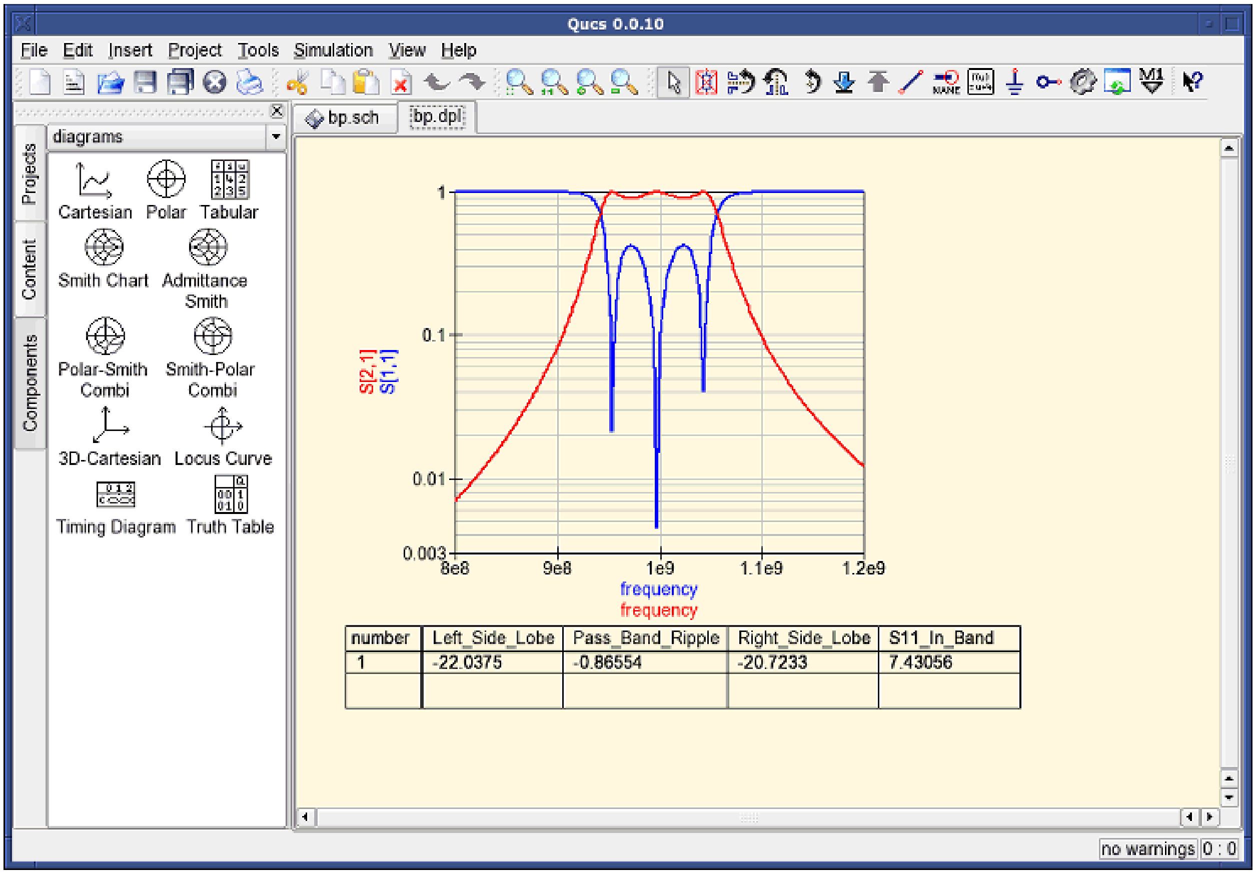


Figure 7 - Qucs results window.

The best found circuit sizes can be found in the optimization dialog, in the Variables tab. They are now the initial values for each one of introduced variables (Figure 8).

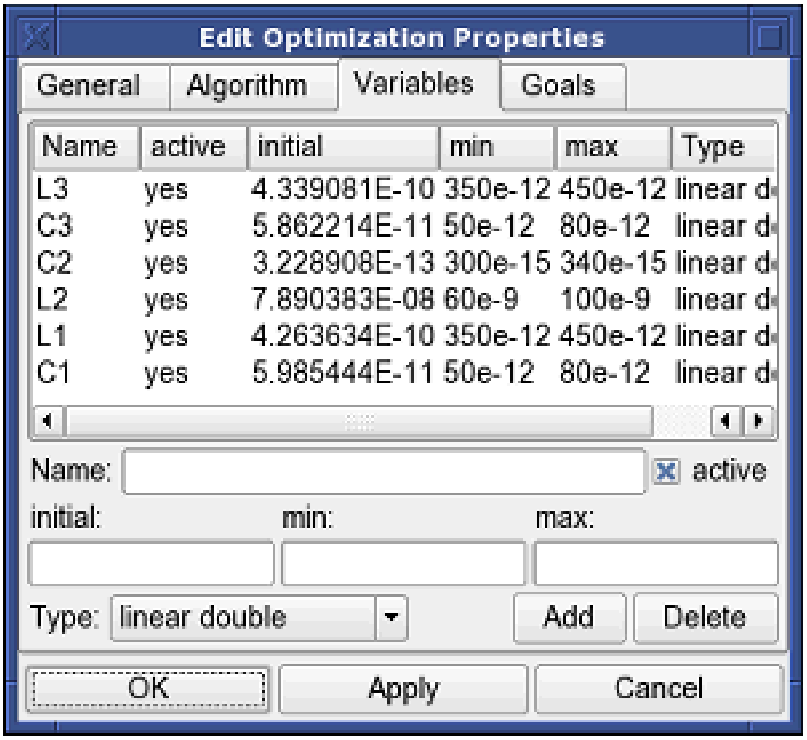


Figure 8 - The best found circuit sizes.

Getting Started with Octave Scripts

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

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

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

4.1 Postprocessing

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

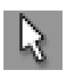
Descripción breve de las Acciones

5.1 Acciones generales

(válidas para todos los modos)

rueda del ratón	esplaza verticalmente el área de dibujo. También se puede desplazar más allá del tamaño actual.
rueda del ratón + Botón Shift	Desplaza horizontalmente el área de dibujo. También se puede desplazar más allá del tamaño actual.
rueda del ratón + Botón Ctrl	Amplia o reduce el área de dibujo.
drag'n'drop file into document area	Tries to open file as Qucs schematic or data display.

5.2 Modo "Seleccionar"



(Menu: Edición->Seleccionar)

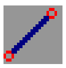
botón izquierdo del ratón	Selecciona el elemento que se encuentra debajo del cursor del ratón. Si hay varios componentes ahí, se puede hacer clic ahí varias veces hasta obtener el deseado.Si se mantiene pulsado el botón del ratón, se puede mover el componente que está debajo del cursor del ratón y todos los que estén seleccionados.Si se mantiene el botón del ratón pulsado y nohay ningún elemento debajo, abre un rectángulo. Cuando se suelte el botón del ratón, se seleccionan todos los elementos que estén dentro del rectángulo.Se puede cambiar el tamaño de un diagrama seleccionado o un dibujo presionando el botón izquierdo del ratón sobre una de sus esquinas y moviéndo el ratón sin soltar el botón
botón derecho del ratón	Permite seleccionar más de un elemento, es decir, que seleccionar un elemento no quite la selección a los demás. Si se hace clic en un elemento seleccionado, lo deselecciona. Este modo también funciona seleccionando mediante un rectángulo, como en el apartado anterior.
Modo "Cablear"	Haciendo clic sobre un cable selecciona una línea recta en lugar de la línea completa.
Modo "Insertar Componente"	Abre un diálogo que edita las propiedades del elemento (Las etiquetas de los cables, los parámetros de los componentes, etc.).

5.3 botón izquierdo del ratón C

oloca una nueva instancia del componente en el esquema.

botón izquierdo del ratón	ota el componente. (No tiene efecto en los diagramas.)
Modo "Cablear"	Rota el componente. (No tiene efecto en los diagramas.)

5.4 Modo "Cablear"



(Menu: Insertar->Cable)

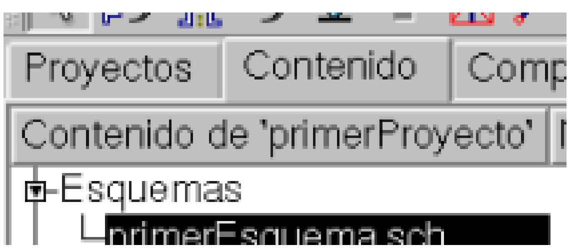
botón izquierdo del ratón	Pone el punto de inicio/fin del cable.
Modo "Cablear"	ermina un cable sin hacerlo en un cable o en una conexión
Modo "Insertar Componente"	Termina un cable sin hacerlo en un cable o en una conexión

5.5 Modo "Pegar"

(Menu: Edición->Pegar)

botón izquierdo del ratón	Coloca los elementos en el esquema (desde el portapapeles).
Modo "Cablear"	clic izquierdo |

5.6 Selecciona archivo.



doble-clic clic derecho |	Abre archivo. Abre archivo.	
clic derecho	Muestra un menú con: "abrir"	■ abre el archivo seleccionado
	"renombrar"	■ cambia el nombre del archivo seleccionado
	"borrar"	■ borra el archivo seleccionado
	"borrar grupo"	■ borra el archivo seleccionado y todos los relativos a él (esquema, datos mostrados, conjunto de datos)

5.7 Teclado

Hay muchas acciones que se pueden activar/hacer con pulsaciones de teclado. Se pueden ver en el menú principal a la derecha del comando. Algunos comandos muy usados están en esta lista:

"Suprimir" o "Backspace"	Borra los elementos seleccionados o entra en modo borrar si no hay ningún elemento seleccionado.
Cursor izquierdo/derecho	Cambia la posición de los marcadores seleccionados en sus gráficos.Si no hay ningún marcador seleccionado, mueve los elementos seleccionados.Si no hay ningún elemento seleccionado, desplaza el área del documento.
Cursor arriba/abajo	Cambia la posición de los marcadores seleccionados en gráficos de más dimensiones.Si no hay ningún marcador seleccionado, mueve los elementos seleccionados.Si no hay ningún elemento seleccionado, desplaza el área del documento.
Tabulador	Cambia al siguiente documento abierto.

Trabajar con Subcircuitos

Los Subcircuitos se usan para dar más claridad a un esquema. Es muy útil en circuitos muy grandes o en circuitos en los que aparece un mismo bloque de componentes varias veces.

En Qucs cada esquema que contenga una conexión de subcircuito es un subcircuito. Para conseguir una conexión de subcircuitos use la barra de herramientas, la vista de componentes (en componentes sueltos) o el menú (Insertar -> Insertar conexión). Después de colocar todas las conexiones del subcircuito (por ejemplo, dos) tiene que guardar el esquema (pulse CTRL-S). Si echa un vistazo a la lista de contenidos (figura 1) verá que ahora aparecerá un "2-port" a la derecha del nombre del esquema (columna "Nota"). Esta nota marca todos los documentos que son subcircuitos. Ahora cmbie a un esquema en el que quiera usar el subcircuito. Después pulse en el nombre del subcircuito (vista de contenido). Volviendo a entrar en el área del documento verá que ahora puede colocar el subcircuito dentro del circuito principal. Hágalo y complete el esquema. Ahora puede realizar una simulación. El resultado es el mismo que si todos los componentes del subcircuito se hubieran colocado directamente en el circuito.

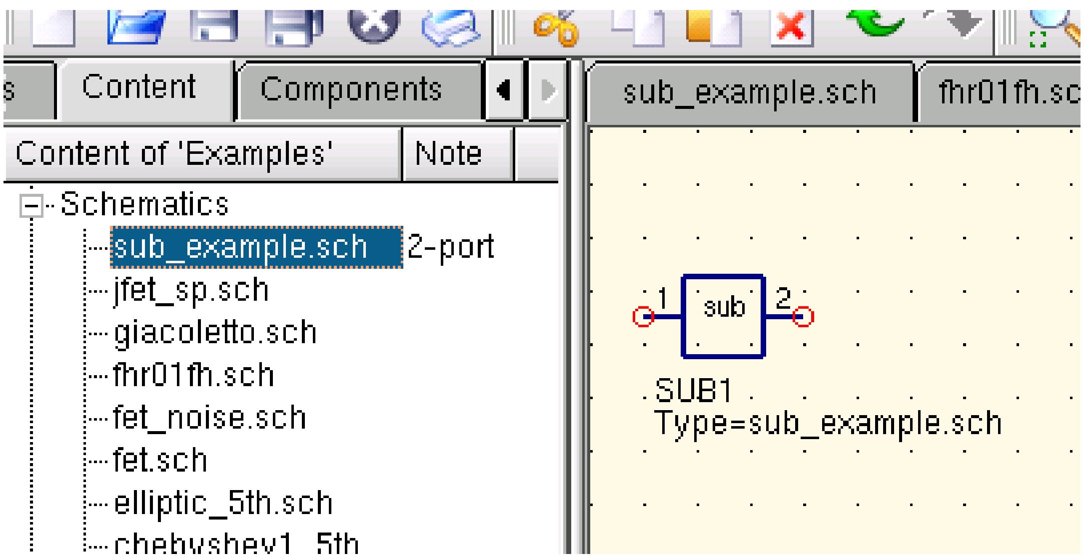


Figura 1 - Acceder a un subcircuito

Si selecciona un componente subcircuito (clic en su símbolo en el esquema) puede entrar en el esquema del subcircuito pulsando CTRL-I (por supuesto, esta función está accesible también via barra de herramientas y menú). Puede salir del subcircuito pulsando CTRL-H.

Si no le gusta el símbolo dado al componente de un subcircuito, puede dibujar su propio símbolo. Simplemente haga el esquema del subcircuito y vaya al menú: Archivo ->Editar símbolo de circuito. Si no ha dibujado aún un símbolo para este circuito se creará un dibujo simple para este. Ahora puede editar este símbolo pintando líneas y arcos. Cuando termine guárdelo. Ahora colóquelo en otro esquema y, voilá, tiene un nuevo símbolo.

Just like all other components, subcircuits can have parameters. To create your own parameters, go back to the editor where you edited the subcircuit symbol and double-click on the subcircuit parameter text (SUB1 in the Figure 1). A dialog apperas and you now can fill in parameterswith default values and descriptions. When you are ready, close the dialog and save the subcircuit. In every schematic where the subcircuit is placed, it owns the new parameters which can be edited as in all other components.

6.1 Subcircuits with Parameters

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

Create a subcircuit:

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

Insert subcircuit and define parameters:

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

- The parameters of the RLC subcircuit can be changed on the top schematic.

Descripción breve de las Funciones matemáticas

The following operations and functions can be applied in Qucs equations. For detailed description please refer to the "Measurement Expressions Reference Manual". Parameters in brackets "[]" are optional.

7.1 Operators

7.1.1 Arithmetic Operators

+x	Unary plus
-x	Unary minus
x+y	Addition
x-y	Subtraction
x*y	Multiplication
x/y	Division
x%y	Modulo (remainder of division)
x^y	Power

7.1.2 Logical Operators

!x	Negation
x&&y	And
x||y	Or
x^^y	Exclusive or
x?y:z	Abbreviation for conditional expression "if x then y else z
x==y	Equal
x!=y	Not equal
x<y	Less than
x<=y	Less than or equal
x>y	Larger than
x>=y	Larger than or equal

7.2 Math Functions

7.2.1 Vectors and Matrices: Creation

`eye(n)`	Creates `n` x `n` identity matrix
`length(y)`	Returns the length of the given vector
`linspace(from,to,n)`	Creates a real vector with `n` linearly spaced components between `from` and `to`, both inclusively
`logspace(from,to,n)`	Creates a real vector with `n` logarithmically spaced components between `from` and `to`, both inclusively

7.2.2 Vectors and Matrices: Basic Matrix Functions

`adjoint(x)`	Adjoint matrix of `x` (transposed and conjugate complex)
`det(x)`	Determinant of a matrix `x`
`inverse(x)`	Inverse matrix of `x`
`transpose(x)`	Transposed matrix of `x` (rows and columns exchanged)

7.2.3 Elementary Mathematical Functions: Basic Real and Complex Functions

`abs(x)`	Absolute value, magnitude of complex number
`angle(x)`	Phase angle in radians of a complex number. Synonym for `arg()`
`arg(x)`	Phase angle in radians of a complex number
`conj(x)`	Conjugate of a complex number
`deg2rad(x)`	Converts phase from degrees into radians
`hypot(x,y)`	Euclidean distance function
`imag(x)`	Imaginary part of a complex number
`mag(x)`	Magnitude of a complex number
`norm(x)`	Square of the absolute value of a vector
`phase(x)`	Phase angle in degrees of a complex number
`polar(m,p)`	Transform from polar coordinates (magnitude `m`, phase `p`) into complex number
`rad2deg(x)`	Converts phase from radians into degrees
`real(x)`	Real part of a complex number
`sign(x)`	Signum function
`sqr(x)`	Square (power of two) of a number
`sqrt(x)`	Square root
`unwrap(p[,tol[,step]])`	Unwraps the angle `p` (in radians – default `step` is 2*pi) using the optional tolerance value `tol` (default is pi)

7.2.4 Elementary Mathematical Functions: Exponential and Logarithmic Functions

`exp(x)`	Exponential function to basis e
`limexp(x)`	Limited exponential function
`log10(x)`	Decimal logarithm
`log2(x)`	Binary logarithm
`ln(x)`	Natural logarithm (base e)

7.2.5 Elementary Mathematical Functions: Trigonometry

`cos(x)`	Cosine function
`cosec(x)`	Cosecant
`cot(x)`	Cotangent function
`sec(x)`	Secant
`sin(x)`	Sine function
`tan(x)`	Tangent function

7.2.6 Elementary Mathematical Functions: Inverse Trigonometric Functions

`arccos(x)`	Arc cosine (also known as "inverse cosine")
`arccosec(x)`	Arc cosecant
`arccot(x)`	Arc cotangent
`arcsec(x)`	Arc secant
`arcsin(x)`	Arc sine (also known as "inverse sine")
`arctan(x[,y])`	Arc tangent (also known as "inverse tangent")

7.2.7 Elementary Mathematical Functions: Hyperbolic Functions

`cosh(x)`	Hyperbolic cosine
`cosech(x)`	Hyperbolic cosecant
`coth(x)`	Hyperbolic cotangent
`sech(x)`	Hyperbolic secant
`sinh(x)`	Hyperbolic sine
`tanh(x)`	Hyperbolic tangent

7.2.8 Elementary Mathematical Functions: Inverse Hyperbolic Functions

`arcosh(x)`	Hyperbolic area cosine
`arcosech(x)`	Hyperbolic area cosecant
`arcoth(x)`	Hyperbolic area cotangent
`arsech(x)`	Hyperbolic area secant
`arsinh(x)`	Hyperbolic area sine
`artanh(x)`	Hyperbolic area tangent

7.2.9 Elementary Mathematical Functions: Rounding

`ceil(x)`	Round to the next higher integer
`fix(x)`	Truncate decimal places from real number
`floor(x)`	Round to the next lower integer
`round(x)`	Round to nearest integer

7.2.10 Elementary Mathematical Functions: Special Mathematical Functions

`besseli0(x)`	Modified Bessel function of order zero
`besselj(n,x)`	Bessel function of first kind and `n`-th order
`bessely(n,x)`	Bessel function of second kind and `n`-th order
`erf(x)`	Error function
`erfc(x)`	Complementary error function
`erfinv(x)`	Inverse error function
`erfcinv(x)`	Inverse complementary error function
`sinc(x)`	Sinc function (sin(`x`)/`x` or 1 at `x` = 0)
`step(x)`	Step function

7.2.11 Data Analysis: Basic Statistics

`avg(x[,range])`	Arithmetic average of vector elements; if a range is given then `x` must have a single data dependency
`cumavg(x)`	Cumulative average of vector elements
`max(x,y)`	Returns the greater of the values `x` and `y`
`max(x[,range])`	Maximum value in vector; if a range is given then `x` must have a single data dependency
`min(x,y)`	Returns the lesser of the values `x` and `y`
`min(x[,range])`	Minimum value in vector; if a range is given then `x` must have a single data dependency
`rms(x)`	Root Mean Square of vector elements
`runavg(x)`	Running average of vector elements
`stddev(x)`	Standard deviation of vector elements
`variance(x)`	Variance of vector elements
`random()`	Random number between 0.0 and 1.0
`srandom(x)`	Give random seed

7.2.12 Data Analysis: Basic Operation

`cumprod(x)`	Cumulative product of vector elements
`cumsum(x)`	Cumulative sum of vector elements
`interpolate(f,x[,n])`	Equidistant spline interpolation of real function vector `f(x)` using `n` equidistant datapoints; the latter can be omitted and defaults to a reasonable value
`prod(x)`	Product of vector elements
`sum(x)`	Sum of vector elements
`xvalue(f,yval)`	Returns the x-value which is associated with the y-value nearest to a specified y-value `yval` in a given vector `f`; therefore the vector `f` must have a single data dependency
`yvalue(f,xval)`	Returns the y-value of the given vector `f` which is located nearest to the x-value `xval`; therefore the vector `f` must have a single data dependency

7.2.13 Data Analysis: Differentiation and Integration

`ddx(expr,var)`	Derives mathematical expression `expr` with respect to the variable `var`
`diff(y,x[,n])`	Differentiate vector `y` with respect to vector `x` `n` times. If `n` is omitted it defaults to `n` = 1
`integrate(x,h)`	Integrate vector `x` numerically assuming a constant step-size `h`

7.2.14 Data Analysis: Signal Processing

dft(x)	Discrete Fourier Transform of vector x
fft(x)	Fast Fourier Transform of vector x
fftshift(x)	Shuffles the FFT values of vector x in order to move the frequency 0 to the center of the vector
Freq2Time(V,f)	Inverse Discrete Fourier Transform of function V(f) interpreting it physically
idft(x)	Inverse Discrete Fourier Transform of vector x
ifft(x)	Inverse Fast Fourier Transform of vector x
kbd(x[,n])	Kaiser-Bessel derived window
Time2Freq(v,t)	Discrete Fourier Transform of function v(t) interpreting it physically

7.3 Electronics Functions

7.3.1 Unit Conversion

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

7.3.2 Reflection Coefficients and VSWR

rtoswr(x)	Converts reflection coefficient to voltage standing wave ratio (VSWR)
rtoy(x[,zref])	Converts reflection coefficient to admittance; by default reference zref is 50 ohms
rtoz(x[,zref])	Converts reflection coefficient to impedance; by default reference zref is 50 ohms
ytor(x[,zref])	Converts admittance to reflection coefficient; by default reference zref is 50 ohms
ztor(x[,zref])	Converts impedance to reflection coefficient; by default reference zref is 50 ohms

7.3.3 N-Port Matrix Conversions

stos(s,zref[,z0])	Converts S-parameter matrix to S-parameter matrix with different reference impedance(s)
stoy(s[,zref])	Converts S-parameter matrix to Y-parameter matrix
stoz(s[,zref])	Converts S-parameter matrix to Z-parameter matrix
twoport(m,from,to)	Converts a two-port matrix from one representation into another, possible values for from and to are 'Y', 'Z', 'H', 'G', 'A', 'S' and 'T'.
ytos(y[,z0])	Converts Y-parameter matrix to S-parameter matrix
ytoz(y)	Converts Y-parameter matrix to Z-parameter matrix
ztos(z[,z0])	Converts Z-parameter matrix to S-parameter matrix
ztoy(z)	Converts Z-parameter matrix to Y-parameter matrix

7.3.4 Amplifiers

`GaCircle(s,Ga[,arcs])`	Circle(s) with constant available power gain `Ga` in the source plane
`GpCircle(s,Gp[,arcs])`	Circle(s) with constant operating power gain `Gp` in the load plane
`Mu(s)`	Mu stability factor of a two-port S-parameter matrix
`Mu2(s)`	Mu' stability factor of a two-port S-parameter matrix
`NoiseCircle(Sopt,Fmin,Rn,F[,Arcs])`	Generates circle(s) with constant Noise Figure(s) `F`. `Arcs` specifies the angles in degrees created by e.g. `linspace(0,360,100)`. If Arcs is a number it specifies the number of equally spaced circle segments, if it is omitted this number defaults to a reasonable value
`PlotVs(data,dep)`	Returns a data item based upon vector or matrix vector `data` with dependency on a given vector `dep`, e.g. `PlotVs(Gain,frequency/1e9)`
`Rollet(s)`	Rollet stability factor of a two-port S-parameter matrix
`StabCircleL(s[,arcs])`	Stability circle in the load plane
`StabCircleS(s[,arcs])`	Stability circle in the source plane
`StabFactor(s)`	Stability factor of a two-port S-parameter matrix. Synonym for `Rollet()`
`StabMeasure(s)`	Stability measure B1 of a two-port S-parameter matrix

7.4 Nomenclature

7.4.1 Ranges

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

7.4.2 Matrices and Matrix Elements

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

7.4.3 Immediate

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

7.4.4 Number suffixes

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

7.4.5 Name of Values

S[1,1]	S-parameter value
nodename.V	DC voltage at node *nodename*
name.I	DC current through component *name*
nodename.v	AC voltage at node *nodename*
name.i	AC current through component *name*
nodename.vn	AC noise voltage at node *nodename*
name.in	AC noise current through component *name*
nodename.Vt	Transient voltage at node *nodename*
name.It	Transient current through component *name*

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

7.5 Constants

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

List of Special Characters

It is possible to use special characters in the text painting and in the text of the diagram axis labels. This is done by using LaTeX tags. The following table contains a list of currently available characters.

Note: Which of those characters are correctly displayed depends on the font used by Qucs!

Small Greek letters

LaTeX tag	Unicode	Descripción
`\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

Capital Greek letters

LaTeX tag	Unicode	Descripción
\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

Mathematical symbols

LaTeX tag	Unicode	Descripción
\cdot	0x00B7	multiplication dot (centered dot)
\times	0x00D7	multiplication cross
\pm	0x00B1	plus minus sign
\mp	0x2213	minus plus sign
\partial	0x2202	partial differentiation symbol
\nabla	0x2207	nabla operator
\infty	0x221E	infinity symbol
\int	0x222B	integral symbol
\approx	0x2248	approximation symbol (waved equal sign)
\neq	0x2260	not equal sign
\in	0x220A	"contained in" symbol
\leq	0x2264	less-equal sign
\geq	0x2265	greater-equal sign
\sim	0x223C	(central european) proportional sign
\propto	0x221D	(american) proportional sign
\diameter	0x00F8	diameter sign (also sign for average)
\onehalf	0x00BD	one half
\onequarter	0x00BC	one quarter
\twosuperior	0x00B2	square (power of two)
\threesuperior	0x00B3	power of three
\ohm	0x03A9	unit for resistance (capital Greek omega)

Matching Circuits

Creating matching circuits is an often needed task in microwave technology. Qucs can do this automatically. These are the neccessary steps:

Perform an S-parameter simulation in order to calculate the reflexion coefficient.

Place a diagram and display the reflexion coefficient (i.e. S[1,1] for port 1, S[2,2] for port 2 etc.)

Set a marker on the graph and step to the desired frequency.

Click with the right mouse button on the marker and select "power matching" in the appearing menu.

A dialog appears where the values can be adjusted, for example the reference impedance can be chosen different from 50 ohms.

After clicking "create" the page switches back to the schematic and by moving the mouse cursor the matching circuit can be placed.

The left-hand side of the matching circuit is the input and the right-hand side must be connected to the circuit.

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

The matching dialog can also be called by menu (Tools->matching circuit) or by short-cut (<CTRL-5>). But then all values has to be entered manually.

9.1 2-Port Matching Circuits

If the variable name in the marker text is an S-parameter, then an option exists for concurrently matching input and output of a two-port circuit. This works quite alike the above-mentioned steps. It results in two L-circuits: The very left node is for connecting port 1, the very right node is for connectiong port 2 and the two node in the middle are for connecting the two-port circuit.

CAPÍTULO 10

Archivos instalados

El sistema Qucs necesita varios programas que se instalan durante el proceso de instalación.

- /usr[/local]/bin/qucs - el interfaz gráfico
- /usr[/local]/bin/qucsator - el simulador (aplicación de consola)
- /usr[/local]/bin/qucsedit - un simple editor de texto
- /usr[/local]/bin/qucshelp - un programita para mostrar el sistema de ayuda
- `/usr/local/bin/qucstrans` - a program for calculation transmission line parameters
- `/usr/local/bin/qucsfilter` - a program synthesizing filter circuits
- `/usr/local/bin/qucsconv` - a file format converter (console application)

Todos los programas son aplicaciones independienes y pueden arrancarse por separado. El programa principal o interfaz gráfico

- llama a qucsator para realizar una simulación,
- llama a qucsedit para mostrar archivos de texto y
- llama a qucshelp para mostrar el sistema de ayuda.
- calls `qucstrans` when calling this program from menu "Tools",
- calls `qucsfilter` when calling this program from menu "Tools",
- calls `qucsconv` when placing the SPICE component and when performing a simulation with the SPICE component.

Además, se crean en la instalación los siguientes directorios:

- /usr[/local]/share/qucs/bitmaps - contiene todos los gráficos (iconos, etc.)
- /usr[/local]/share/qucs/docs - contiene documentos HTML para el sistema de ayuda
- /usr[/local]/share/qucs/lang - contiene los archivos con las traducciones

10.1 Command line arguments

`qucs [file1 [file2 ...]]`

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

`qucsedit [-r] [file]` (r = read-only)

`qucshelp` (no arguments)

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

Schematic File Format

This document describes the schematic file format of Qucs. 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>
```

The file contains several section. Each of it is explained below. Every line consists of not more than one information block that starts with a less-sign < and ends with a greater-sign >.

11.1 Propiedades

The first section starts with `<Properties>` and ends with `</Properties>`. It contains the document properties of the file. Each line is optional. The following properties are supported:

- `<View=x1,y1,x2,y2,scale,xpos,ypos>` contains pixel position of the schematic window in the first four numbers, its current scale and the current position of the upper left corner (last two numbers).
- `<Grid=x,y,on>` contains the distance of the grid in pixel (first two numbers) and whether grid is on (last number 1) or off (last number 0).

- `<DataSet=name.dat>` contains the file name of the data set associated with this schematic.
- `<DataDisplay=name.dpl>` contains the file name of the data display page associated with this schematic (or the file name of the schematic if this document is a data display).
- `<OpenDisplay=yes>` contains 1 if the data display page opens automatically after simulation, otherwise contains 0.

11.2 Symbol

This section starts with `<Symbol>` and ends with `</Symbol>`. It contains painting elements creating a schematic symbol for the file. This is usually only used for schematic files that meant to be a subcircuit.

11.3 Componentes

This section starts with `<Components>` and ends with `</Components>`. It contains the circuit components of the schematic. The line format is as follows:

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

- The `type` identifies the component, e.g. `R` for a resistor, `C` for a capacitor.
- The `name` is the unique component identifier of the schematic, e.g. `R1` for the first resistor.
- A `1` in the `active` field shows that the component is active, i.e it is used in the simulation. A `0` shows it is inactive.
- The next two numbers are the x and y coordinates of the component center.
- The next two numbers are the x and y coordinates of the upper left corner of the component text. They are relative to the component center.
- The next two numbers indicate the mirroring about the x axis (`1` for mirrored, `0` for not mirrored) and the counter-clockwise rotation (multiple of 90 degree, i.e. 0...3).
- The next entries are the values of the component properties (in quotation marks) followed by an `1` if the property is visible on the schematic (otherwise `0`).

11.4 Wires

This section starts with `<Wires>` and ends with `</Wires>`. It contains the wires (electrical connection between circuit components) and their labels and node sets. The line format is as follows:

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

- The first four numbers are the coordinates of the wire in pixels: x coordinate of starting point, y coordinate of starting point, x coordinate of end point and y coordinate of end point. All wires must be either horizontal (both x coordinates equal) or vertical (both y coordinates equal).
- The first string in quotation marks is the label name. It is empty if the user has not set a label on this wire.
- The next two numbers are the x and y coordinates of the label or zero if no label exists.
- The next number is the distance between the wire starting point and and the point where the label is set on the wire.

- The last string in quotation marks is the node set of the wire, i.e. the initial voltage at this node used by the simulation engine to find the solution. This is empty if the user has not set a node set for this wire.

11.5 Diagrams

This section starts with `<Diagrams>` and ends with `</Diagrams>`. It contains the diagrams with their graphs and their markers. The line format is as follows (line break not allowed):

```
<x y width height grid gridcolor gridstyle log xAutoscale xmin xstep
 xmax yAutoscale ymin ystep ymax zAutoscale zmin zstep zmax xrotate
 yrotate zrotate "xlabel" "ylabel" "zlabel">
```

- The first two numbers are x and y coordinate of lower left corner.
- The next two numbers are width and height of diagram boundings.
- The fifth number is 1 if grid is on and 0 if grid is off.
- The next is grid color in 24 bit hexadecimal RGB value, e.g. #FF0000 is red.
- The next number determines the style of the grid.
- The next number determines which axes have logarithmical scale.

11.6 Paintings

This section starts with `<Paintings>` and ends with `</Paintings>`. It contains the paintings that are within the schematic.

Descripción técnica completa del simulador

Disponible sólo en internet en http://qucs.sourceforge.net/tech/technical.html

Circuitos de ejemplo

Disponible sólo en internet en http://qucs.sourceforge.net/download.html#example