Qrouter 1.3 (Stable) and 1.4 (Development)

Table of Contents

Using Qrouter version 1.2 and higher:

Command-line invocation:

qrouter [*options*] [*basename*]

where *options* may be one of the options below. *basename* is the name of the input DEF file, with or without the "**.def**" extension. File *basename***.def** is assumed to exist and to define cell placement and netlist information. The command-line options are kept for backward compatibility. In general, the preferred method for versions 1.2 and higher is to put all options into the runtime script, which is then either specified on the OS command line as "**-s** *scriptname*", or can be run from the qrouter (Tcl interpreter) command line as "**source** *scriptname*". Qrouter versions 1.2 and higher may also be used completely interactively, with no options given on the command line. A console will appear, with an interpreter prompt, and commands to qrouter may be entered by hand.

If qrouter versions 1.2 and higher are compiled with the configuration option "**--without-tcl**", then it will behave in usage like version 1.1, without the ability to run a script file, a console window, graphics, or interpreter.

Qrouter versions 1.2 and higher are intended to be compiled with Tcl/Tk support and used in conjunction with a script file that tells it what steps to take to read and route a design. See below for the list of qrouter commands that are recognized in the script file or at the interpreter prompt. In addition to the qrouter commands, the interpreter recognizes all valid Tcl and Tk syntax.

Command-line options:

-noc[**onsole**]
-nog[**raphics**]
-s *scriptname*
-c *config_name*
-p *power_bus_name*
-g *ground_bus_name*
-d *delay_file*
-e *effort*
-k *tries*
-i *filename*
-f
-r *value*
-v *level*
-h

Option **-noc** runs qrouter without generating a Tk console, so all output goes to the terminal where qrouter was invoked. This allows qrouter output to be captured into a file.

Option **-nog** disables the graphic display of routing, which is generally preferred for long routing jobs, to optimize for speed. However, the graphic display provides a good visualization of how the routing is progressing as well as providing some debugging options.

Option **-s** immediately executes interpreter commands in the script file **scriptname**, effectively in batch. If the script contains the **quit** command, then it is a batch process; otherwise, after completing all commands in the script, qrouter returns to the Tcl prompt. Using the **-s** option with a script file is the preferred way to run qrouter.

Option **-c** exists for compatibility with the non-Tcl version of qrouter, and replaces the **-s** option. The file *config_name* is assumed to exist and contains basic routing parameters and other information in the older, non-Tcl format (see below). If neither a **-c** or **-s** option is specified, then the default configuration file name of "**route.cfg**" is used, and is expected to be a file in the older (non-Tcl) format. The configuration file should either contain information about the standard cells, or a pointer to a LEF file that does so.

Option **-p** and **-g** supply names for the power and ground buses as *power_bus_name* and *ground_bus_name*, respectively. This allows qrouter to route a net of the same name as a power or ground bus from a signal pin to the nearest available standard cell power or ground terminal, without requiring the net to specify every standard cell as part of the net. This is helpful when power and ground routing is handled separately from the detail router. This option is also available as a command inside the runtime script.

Option **-r** changes the resolution of the output by a factor of *value*. Like most DEF files, output is in units of centimicrons. By default, qrouter will query the LEF file for the manufacturing grid and output values to the required precision. If the LEF file does not declare a manufacturing grid, and the required precision is greater than 10nm, then the **-r** option can be used to specify the required precision. A *value* of 2 will increase the output resolution to 5nm; *value* of 10 will increase the output resolution to 1nm, etc. When the LEF file does specify the manufacturing grid, then *value* is the factor to increase the output resolution beyond the manufacturing grid (generally not recommended). Note that if the LEF file specifies a manufacturing grid, then qrouter will handle the output resolution automatically, and the **-r** option is not needed.

Option **-e** overrides the "**effort**" option passed to the stage2 and stage3 with an absolute minimum required effort *value*. The amount of effort to apply to the specified route stage is the minimum of *value* and the effort value passed to the stage2 or stage3 command. By default, stage2 and stage3 get an effort level of 100, which is considered a high level of effort. See the stage2 command for a complete description of how the effort value operates. Generally, value 10 is a low level of effort, while value 100 is a high level of effort.

Option **-k** overrides qrouter's assessment of its progress, by which it determines that a route solution cannot be found. Usually this algorithm suffices to distinguish between routable and unroutable designs. Occasionally, however, qrouter will prematurely halt a route-in-progress that is close to a solution. In these rare cases, the **-k** option can force qrouter to completion. The value *tries* sets a limit on the additional attempts. As of qrouter version 1.3.75, the level of effort (see option "**-e**" above) is used to determine progress and option **-k** is deprecated; if used, it is converted to a level of effort by multiplying by 100.

Option **-f** forces qrouter to define at least one tap point for every pin connected to a network, even if qrouter has determined that the pin is not available due to obstructions. This option can work around two particular cases: (1) in which pins are arranged in tight spacing and satisfy euclidean-measure DRC rules (but not manhattan-measure DRC rules, which qrouter uses), and (2) in which pin geometry in the LEF macro definition file is coincident with obstruction geometry. In other cases, the **-f** option is likely to cause DRC errors in the layout, but can be useful for generating routed layout so that one can analyze other possible errors like incorrect track offsets. This option is also available as the argument "**force**" to the "**stage1**" and "**stage2**" commands.

Option **-d** generates a file named *delay_file* containing parasitic resistance and capacitance values for directed graphs of each route, appropriate for calculating the delay on each wire. This is used in qflow to back-annotate routing delays into static timing analysis.

Option **-i** generates a file named *filename* that contains information about routing layers pulled from the LEF technology file. This file is used by qflow for managing the setup of the routing process.

Option **-v** sets the level of verbosity to *level*. Level 0 provides no output other than critical error information. Level 1 provides a normal output with general information about the routing progress. Levels 2 and above give increasing detail about the individual routes and produce large amounts of output. They are meant largely for debugging purposes.

Option **-h** provides a brief summary of qrouter usage. In conjunction with option **-v 0**, it returns just the version of qrouter. The version string ends with "**T**" if qrouter has been compiled with scripting (Tcl/Tk) support.

Qrouter version 1.2 and higher interpreter commands:

Qrouter versions 1.2 and higher, when compiled with Tcl/Tk support, run as an extension of the Tcl/Tk interpreter. This means that qrouter shows a Tcl interpreter command-line prompt, and all valid Tcl and Tk syntax may be entered at this prompt. In addition to the standard Tcl and Tk commands, qrouter registers a set of its own commands with the Tcl interpreter. These commands are specific to the router, and comprise the information needed by the router to find the input files it needs and perform the detail route. The commands are as follows, with a one-line summary. Click on each command name for a full description of the command.

Input/Output commands

read_lef
: Read a LEF file (either technology or standard cell)

read_def [-abort]
: Read a DEF file to be routed. If the **-abort** option is given (from qrouter-1.4.32) then the **read_def** command will throw an error on certain errors encountered in the DEF file such as an unidentified component or an incorrect specification of route tracks. Normally, most such errors are deemed "recoverable" by qrouter and ignored.

write_def
: Write an annotated DEF file with detail routes

read_config
: Read a configuration file in the format used by qrouter version 1.1.

write_delays
: Write a file of delays in the format used by vesta, in qflow (see qflow for scripts to convert delay output to standard delay formats).

write_failed
: Output a text file listing nets that failed to route.

Basic routing setup commands

obstruction
: Specify additional routing obstructions

via
: Declare the number of vias allowed to stack, and how to pattern.

vdd
: Specify the name of the net used for power routing

gnd
: Specify the name of the net used for ground routing

Additional routing setup commands

pitchx
: Modify the base pitch of vertically-oriented route tracks.

pitchy
: Modify the base pitch of horizontally-oriented route tracks.

unblock
: Set the policy for handling possible false-positive blockages of pins.

Routing commands

stage1
: Attempt to route all nets and generate a list of failures

stage2
: Route nets on failure list with rip-up and re-route

stage3
: Remove and re-route each net in turn to clean up

failing
: View and manipulate the list of route failures

remove
: Remove a routed net, individually or collectively

antenna
: Analyze or fix antenna violations.

cleanup
: Clean up net routes by removing adjacent vias.

Additional routing manipulation commands

ignore
: Specify one or more nets to be ignored by the router

priority
: Specify one or more nets to be given priority by the router

cost
: View and change the weights of router costing functions

layers
: Change the number of routing layers used

passes
: Change the number of passes per routing attempt (default 10)

resolution
: Change the resolution of the output units (default is 1)

Graphics visualization commands

redraw
: Redraw the graphics display

map
: Specify what to draw in the background of the router display

General-purpose commands

verbose
: Set the level of printed output (default 3)

layer_info
: Query information about the routing layers defined by the technology and used by qrouter.

congested
: Return a list of instances in order of predicted congestion, along with the estimated severity of the congestion. To be used as feedback to placement tools to suggest the amount of fill padding needed to make the layout routable by qrouter.

quit
: Quit qrouter

Debugging commands

query
: Query information about a grid position, instance, node, or net.

DEF File Formats

The DEF file is an open standard file format for describing circuit layout. For purposes of input to **qrouter**, the DEF file describing the circuit to be routed should contain a **COMPONENTS** section describing the placement of standard cells, an optional **PINS** section describing the placement of (labeled) pins, and a **NETS** section declaring the netlist to be routed. It should also contain **TRACKS** statements for each metal routing layer, declaring where the routing tracks should be placed.

qrouter produces as its output the same DEF file that was given as input, annotated with the phyical routes by adding **+ ROUTED** records to the **NETS** section for each net. The output file name is *basename*_**route.def**.

Nets in the input DEF file may be pre-routed. Such routes will be assumed to follow the defined route tracks in the same DEF file. Qrouter will maintain these routes as part of its database. If not marked as FIXED or COVER, qrouter may rip up and reroute such predefined routes as necessary to find a solution.

LEF File Format

The LEF file is an open standard file format for describing circuit technology and parameters, and defining subcircuit geometry, especially for standard cells. For purposes of input to **qrouter**, the LEF file should contain **LAYER** sections to describe each routing layer, **VIA** sections to describe the geometry of vias to connect routing layers, and **MACRO** sections to describe the geometry of each subcell in the layout. Generally, this LEF file will be provided by the standard cell vendor to match a specific geometry, and complements a GDS file containing the manufacturing layer geometry of the same standard cells. **qrouter** should parse the unmodified vendor LEF file.

Common Problems and How to Solve Them

But first, a quick question-and-answer session:

> **Q:** Qrouter can't route my 100K gate superscalar processor!
>
> **A:** Qrouter does an amazing job for something that I wrote in my spare time using algorithms that came off the top of my head. Don't expect it to be industrial-strength. The solution to routing problems as a function of scale is often to force the placement stage to add additional padding with filler cells, increasing the area of the layout but also increasing the likelihood of routing success. However, qrouter does not alter component placement, so where density reaches routing capacity, qrouter is likely to fail.
>
> **Q:** Qrouter can't route my design in a 65nm technology!
>
> **A:** Standard cell routing becomes somewhat difficult at 180nm due to minimum metal area requirements; it becomes a pain at 90nm, and it becomes a royal pain at 65nm and below. Antenna rules, parallel run-length requirements, and increasingly complicated DRC rules become difficult to satisfy. Generally speaking, if you can afford the mask cost of a 90nm or smaller feature-size chip, you can probably afford Synopsys.

Qrouter can have an easy or hard time with the routing, depending on the technology setup. Although Qrouter is not a fully-featured router with bells and whistles for partitioning large designs and applying measures for congestion control, it is quite capable for designs of moderate size and complexity.

Like most routers, it will perform best when all pins have a port geometry that is centered on the routing grid, and when all cell pin and obstruction geometry is correctly specified, and when all route layer information is correctly specified. Qrouter will make an attempt to accomodate improperly specified technology and standard cell information, but the usual "garbage in, garbage out" caveats apply.

If Qrouter is failing on a large number of nets, chances are good that the input has problems. The first thing to do is to let Qrouter run to completion, then load the resulting DEF file into a layout editor to look at what happened.

In particular, look at where the route tracks and ports are relative to the standard cell pins. If they do not line up, there is a track offset specification problem. This can happen if the standard cell set places cell boundaries at an offset from track positions, but does not specify an offset for the cell position. If this happens, then the technology LEF file that describes the route layers may need to have OFFSET statements for each route layer.

If some cell pins seem to be on-grid and some off-grid, then there may be a problem with the way the standard cell bounding boxes are described, or the standard cells may have been designed with a pitch that is not the same as the pitch described by the technology LEF file for the routing layers.

If the routing is just too dense, check if it is possible to work with a larger number of route layers, and change Num_layers in the configuration file accordingly. Another solution to dense routing is to run the standard cell placement with a higher percentage of fill (or increased spacing between rows). This is a tradeoff between routing congestion and overall design area.

Using Qrouter version 1.1 and non-Tcl-based qrouter:

Note that Qrouter version 1.1 is considered deprecated; instructions for use are given here for historical reference. For the same reason, it is generally *not* recommended to compile qrouter without Tcl/Tk support.

Command-line invocation:

qrouter [*options*] *basename*

Using Qrouter version 1.1:

options is one of the options given in the usage for versions 1.2 and higher, with the exception of the **-m** option.

For versions 1.2 and earlier, the **-k** option used with a truly unroutable circuit may cause qrouter to run indefinitely.

Note: Prior to version 1.3.42, qrouter would ignore the manufacturing grid specified in the LEF file or files, and generate centimicron output by default, requiring the use of the **-r** option for any technology having a higher base resolution than 10nm.

basename is the name of the project file to route, where the project file should be a DEF format file. *basename* may be given without an extension.

qrouter writes an annotated DEF file *basename*_**route.def**, which is the original DEF file with the physical routes added.

Configuration File Format for version 1.1

The configuration file is kept for backwards compatibility with qrouter version 1.1 and later versions of qrouter that are compiled without Tcl/Tk support. For versions 1.2 and higher compiled with Tcl/Tk support, the preferred method is to write a script file invoking qrouter interpreter commands in sequence (see list of interpreter commands, above).

The configuration file contains all information not present in the LEF file. It allows some simple definitions for macros where a LEF file is not provided; however, these definitions are not as complete as those parsed from a LEF file.

The qrouter source distribution comes with an example **route.cfg** file in the `lib/` subdirectory.

Other than filenames, the file contents are not case-sensitive. Comment lines begin with the character "**#**".

Generally speaking, the configuration file will need only two entries:

Num_layers *layers*
: where *layers* is the number of layers to route. The LEF file will generally define *all* routing layers, and the end-user will normally want to restrict the physical routing to a subset of these.

lef *filename*
: where *filename* is a relative or absolute path to the technology LEF file defining the standard cell geometry and routing layer parameters. If the technology specifications and the standard cell macros are in separate LEF files, the **lef** statement may be used more than once.

In addition to the required entries, a set of parameters controls the basic routing algorithm. The tool sets sensible default values. All route costs are integer and have no absolute meaning; only their values relative to one another are meaningful.

Route Segment Cost *value*
: Cost of a route of the length of one track pitch on any metal layer in the preferred direction of the route layer. This should be the lowest of all the costs. Default value 2.

Route Via Cost *value*
: Cost to switch routing layers up or down using a via. Default value 10.

Route Jog Cost *value*
: Cost of a route of the length of one track pitch on any metal layer against the preferred direction of the route layer. This cost should be relatively high; however, it should be less than (Route Segment Cost + 2 * Route Via Cost) to allow a jog of a single track pitch to be preferred over switching to another layer. Default value 20.

Route Crossover Cost *value*
: Cost of routing directly over or under an unrouted pin connection to a cell. This helps prevent pins from getting boxed in before they are routed, making them unroutable. Default value 8.

Route Block Cost *value*
: Cost of routing directly over or under an unrouted pin connection to a cell, when that connection is the only available vertical connection to the pin. Default value 25.

Route Collision Cost *value*
: Cost of shorting another net. This happens during the second routing stage. All nets that are shorted by this net will have to be ripped up and rerouted. The value should be large to avoid a large number of existing routes needing to be ripped up. Default value 50.

Another set of entries can be used to affect the routing of the networks by declaring routes that must be made first, declaring routes that should not be made, and declaring special obstruction areas.

no stacked vias
: (Qrouter version 1.1 and later only) Avoid creating stacked contacts in routes.

stack *n*
: (Qrouter version 1.1.5 and later only) Avoid creating stacked contacts having more than *n* contacts at the same route point. For example, "**stack 2**" could connect metals 1, 2,

and 3 (two contacts at the same route point), but would not connect metals 1, 2, 3, and 4 without adjusting the route to reduce the via stack height. "**stack 1**" is the same as "**no stacked vias**". "**stack 0**" is also treated the same as "**no stacked vias**".

via pattern *type*

Specify a type of via patterning across the grid. Values for *type* are "**none**" (the default), "**normal**", and "**invert**". If "**normal**" or "**invert**" are used, then the technology LEF file should define two vias per layer pair, one oriented in the horizontal direction, and one oriented in the vertical direction. These will be used alternately across the route track grid in a checkerboard pattern, allowing an increased density of tracks. "**normal**" and "**invert**" differ only in the track position of the horizontal or vertical vias.

do not route node *name*

Do not route the net named *name*. Note that this can also be accomplished by removing the net from the input DEF file.

route priority *name*

Route the net named *name* first, before other routes. This will ensure the most efficient route solution for the network, as it will not have to route around existing networks. Otherwise, networks are ordered by the number of nodes connected to the net.

obstruction *llx lly urx ury layer*

While the technology LEF file declares all obstructions occuring in a cell, there are many reasons why one may want to declare other areas to be obstructions to routing, for example, to prevent placement of vias on top of pins. Argument *layer* is the name of a metal route layer that is obstructed by the declared area, and the bounds of the obstruction rectangle are declared by the remaining values, which are in units of microns.

Finally, there is a set of entries that are shadowed by parameters declared in the LEF and DEF files. If these entries are supplied in addition to the LEF file, then values in the LEF file will take precedence. These entries can be used to replace the LEF file entirely; however, the geometry information is limited and cannot describe obstructions in the routing path.

Layer_*n*_name name

The name of the routing layer number *n* used in the DEF file. The value *n* should start at one for the first (lowest) routing layer.

Layer_*n*_width value

The width of the metal routing layer wires for layer number *n*. The value is given in units of microns.

Layer *n* **wire pitch value**

The track pitch of the metal routing layer for layer number *n*. The value is given in units of microns.

Layer *n* **horizontal|vertical**

Declares the metal routing layer number *n* to have a preferred route direction which is either horizontal or vertical. This will determine which direction is given the segment cost and which is given the jog cost (see above).

X upper bound *value*

Indicates the right side limit of the routing. If not declared, the upper bound will be determined by the extent of the cell and pin layout. The value is in units of microns and can be used to allow the router to route outside of the immediate cell layout area.

X lower bound *value*

Indicates the left side limit of the routing. If not declared, the upper bound will be determined by the extent of the cell and pin layout. The value is in units of microns and can be used to allow the router to route outside of the immediate cell layout area.

Y upper bound *value*

Indicates the top side limit of the routing. If not declared, the upper bound will be determined by the extent of the cell and pin layout. The value is in units of microns and can be used to allow the router to route outside of the immediate cell layout area.

Y lower bound *value*

Indicates the bottom side limit of the routing. If not declared, the upper bound will be determined by the extent of the cell and pin layout. The value is in units of microns and can be used to allow the router to route outside of the immediate cell layout area.

---

email: tim@opencircuitdesign.com

Last updated: September 17, 2019 at 10:40pm