Automatic Routing with Trout

30th September 2019

Making all connections by hand using the editor is a nasty and error-sensitive job. *Trout*, your automatic router, can do this job for you much quicker and without errors. One of the most interesting features of *trout* is that it allows any amount of pre-routed nets and obstacles. For instance, you can route some critical part of your circuit by hand, and then let the router complete the remaining parts. You can also ask the router to make only a part of the design. Many creative combinations of automatic and manual design are possible.

Another special property of *trout* is that it routes *through* the cells, rather than around them. This approach is especially suitable for Sea-of-Gates layout.

1 What is required before I can click Trout?

The router is a very complicated tool, which requires all kinds of data before it can start its job. If any of this data is missing, it will fail miserably. Basically it needs the same as *madonna*: a proper netlist which specifies the desired connectivity of the modules. But apart from this, the router expects a placement layout, in which all instances are placed. This placement can be made by hand or using *madonna* (or both). It should contain all instances which were mentioned in the netlist description. It is not necessary to place the terminals, because the router will do it itself if they are missing. It is also not necessary to give the instances the proper instance names.

You can expect an error- or warning-window in the following cases:

- One or more instances are missing in the placement. In this case the circuit will be routed without the terminals on the missing instances. This will most likely lead to an incorrect layout.
- There are unknown instances in the placement, that is, the layout contains instances which do not appear in the circuit description. These instances will be treated as dummies: apart from the power rails they will not be connected. They will only be treated as obstacles. To prevent unknown instances, use the set instance name in the instances menu.
- The placement contains a terminal which is not in the net list.
- If the router fails to route certain nets.

2 Running *trout*

3 Calling *trout* from *seadali*

After pressing Trout, *seadali* will ask for the name of the circuit. Then you'll see a small menu. Just press DO IT ! to start the router. After a short while the routed circuit should appear on your screen. Optionally you can set the following parameters in the menu:

no power If selected, the horizontal power rails of the circuit will not be connected. By default all vdd (and vss) rails will be connected to each other by a vertical wires in ins. *Trout* will do this even if the power nets are not present in the netlist.

erase wires Erase all wires (boxes) and terminals which are present in the layout before routing. By default the router will only add wires to the existing layout.

border terminals Place all unspecified terminals on the boundary the cell. In this way it is ensured that all terminals can be reached on a higher hierarchical level. This button is enabled by default.

route box only After pressing this button you can drag a box over the layout. *Trout* will only route the nets of which the terminal patterns are in this box. All wires which are generated by the router (except power) will only be inside the box. In this way you can locally re-route a large design. On large empty designs routing in a number of smaller boxes can be considerably faster than routing in one piece.

Notice that the connectivity verifier (which is run automatically after *trout*) is very likely to report errors for unconnected nets because some nets cannot be routed inside the box.

single pass routing Prevent the trout from retrying in case the routing was not complete.

Option menu Opens a second menu with additional options for the router. See section 11 for more details.

DO IT! Start the router. Depending on the size of the circuit and the load of the system, this will take 5 seconds up to 20 minutes. The routing time increases quadratically with the size of the layout and linearly with the number of nets.

3.1 Stopping the router

There are two ways to stop *trout* while it is executing from within *seadali*:

1. By clicking KILL in the menu. This is the brute-force way. It will immediately kill *trout* and leave the layout unchanged.
2. By clicking STOP in the menu. This will request *trout* to interrupt the routing as soon as possible and return the current layout. This will most likely be an incompletely routed circuit. This button can be useful if routing takes too long, but you are still interested in the (partial) solution.

4 Running *trout* from the command line

The non-interactive way to call *trout* is similar to the way *madonna* is run:

```
[op5u9/myproject]  sea -r name
```

in which *name* is the name of the circuit cell. It is assumed that a layout cell *name* with all instances already exists in the layout view. See section **??** for more details on *sea*.

5 What does *trout* do for me?

Apart from just routing the nets according to the netlist description, the router performs a number of other tasks. Most of these features were added just to make the 'default behavior' of *trout* is as intuitive and useful as possible. *Trout* performs the following automatically:

Orphan instances The router tries to assign unknown instances ('orphans') to missing instances if they belong to the same cell type. For instance: if an instance 'x' of cell 'nand2' was missing in the placement, but an unknown instance 'y' of cell 'nand2' was found (that is, 'y' is not in the circuit description) then instance 'y' will be renamed to 'x'. Notice that this does not necessarily result in a good placement for the instances. This feature was added for your convenience during manual placement.

Father terminals The terminals of the father cell can be placed automatically by *trout*. The router automatically places any missing terminal of the father circuit on the border of the cell or on an optimal position. No message will be printed in this case. It is better not to place father terminals unless you really want to force them on a specific position. See section 6 for more details.

Power terminals The router automatically adds the 'vdd' and 'vss' terminals to your layout, in case they are not yet present yet. *Trout* will do this even if the circuit does not contain power nets.

Power rails The horizontal vdd- and vss power rails (in metal1) are connected by vertical wires in metal2. In this way all power terminals are connected to the vdd or vss terminals, even if the design contains many rows of transistors. Click no power to disable power rails connection.

Connectivity check *Trout* runs the connectivity verifier automatically after routing. If there is any short or unconnect a warning window will pop up and markers will be placed in the layout. This is the same verifier as under the button Check nets (see **??** on page **??**).

6 Guidelines for successful routing

Trout is doing its best to route all nets for you. It can happen, however, that *trout* is unable to route them all. Some of the problems may be cause by a design error at an earlier phase. To ensure the routability of the nets, you must design de layout of your instances in a certain way. Since the cell which you are constructing now may also be used as a leaf cell on a higher lever, it is wise to comply with the following guidelines:

- Make sure that all terminals can be reached by a wire from the 'outside' of cell. A terminal which cannot be connected is not useful. The best way to ensure the 'accessibility' is by placing the terminals on the border of the cell. The router helps you with that.
- Let *trout* take care of the placement of the terminals (the button border terminals). If you do not specify the terminal positions (e.g. if you pressed erase wires), *trout* will place them on the border of the cell. The router will automatically find the shortest (and most clever) route to the border. In special cases you can always enforce a terminal placement by placing terminals manually.
- Try to make the cell as transparent as possible. Avoid using horizontal wires in metal2. You will notice that you router itself does not like to generate this kind of wires. If *trout* does generate them, that is a sure indication that area is very congested (close to its maximum routing capacity).
- Leave some space in critical areas by shifting the modules apart. Manually update your *madonna* placement: she is not too clever.
- *Trout* does not make short-circuits by itself, it can only make existing short-circuit bigger. If the automatic checker detects a short-circuit after routing, this generally indicates the presence of an error (short-circuit) in one of the son-cells. In many of the cases a terminal in a sub-cell makes a short circuit with the terminal of another net in that sub-cell. Since *trout* expands the entire pattern of the terminal, including the entire short-circuit, connecting more terminals of the net will just make the short bigger. Unfortunately finding the cause of these short is not too easy, since the exact position of the cause of the problem cannot be indicated in the layout.

7 What should I do in the case of incomplete routing?

Despite all precautions, incomplete routing may still occur. Basically we can distinguish two situations for this case:

1. Many nets are unconnected. With 'many' we mean more than 5 or more than a few percent.
2. A few shattered nets are unconnected.

The strategies for these situations will be discussed in the following sections.

8 Many nets are unconnected

In this case the conclusion is simple: the cell must be enlarged. We have to live with the fact that the routing capacity on our Sea-of-Gates chip is limited. In almost all cases we will have to sacrifice some transistors to create space for the wires. You can do this by:

1. Running *madonna* again in a larger box.
2. Running *madonna* again in a box with a different width/height ratio. In 'flat' horizontal cells the capacity for horizontal wires is easily too small. In narrow vertical cells there might not be enough space for vertical wires. Have a look at the density of the wires in each of the layers so see which is the case. Use X-expand or Y-expand to guide *madonna*.
3. Manually replacing some blocks. Try to distribute the wiring density over the entire cell area.

Prevent spending too much time squeezing the size of your placement. Always try the easiest and quickest way first: use *madonna*.

9 A small number of nets is unconnected

At this point it is useful to get a little feeling for way in which the router works. *Trout* routes the nets one at a time in a 'greedy' way. For each net he tries to find the best wires which connect the terminals. The router is very clever in finding this path. He uses all tricks in the image (such as using the polysilicon gates for wires). You can be sure that if the router reports that it cannot connect a net, you will also not be able to do it manually. Unfortunately the router does not have much overview over the entire set of nets and wires. Therefore it is possible that the wires of a net block a terminal of a still unrouted net. This was the cause of the incomplete routing in the case of a few shattered unconnects. Notice that this is very dependent on the order in which the nets were routed.

The problem can be solved in the following ways:

1. Trout itself tries to solve the problem by ripping up wires and re-routing. It will only do this with a small number of unconnected nets and if the CPU-consumption is not too high.
2. Make a new placement by running *madonna* again in the same or a slightly larger box. This will create a new one which - with a little bit of luck - will be routable. Any time you press Madonna, you'll get a different placement.
3. Delete a part of the congested area and run *trout* again. You can specifically delete some pieces of wire which you suspect to be the cause of the problem. The router will try to glue the nets back together again, in a different order. You can use the route box only-button to focus the router. Have a look at which spot the unconnects are located.
4. Route the cell in a few overlapping boxes using the route box only-button.
5. Modify the layout manually. At any case you can call the router again to finish the result automatically.

10 The run-time of *trout*

The router will automatically keep you informed about its progress. If it takes a longer time, *trout* will also print a rough estimate of the time it needs to finish. The cpu-time consumption of *trout* depends on the following factors:

- The dominating factor is the **size** of your placement (in square grid points). The run time increases quadratically with the size: if a layout is twice as large, it takes four times as much time to route it.
- The run time increases approximately linear with the number of nets.
- The router is at is best for large complicated layouts with many wires. It is quite slow, however, if there is a lot of empty space in your placement. This is because in the latter case there are many different ways of making a wire. *Trout*'s algorithm evaluates many routes for each wire.
- For small placements, the overhead time for database IO is determining the time you have to wait.
- If *trout* fails to route a wire, it may spend some time trying to complete the routing in a different way. It might also spend some time trying to solve problems by rip-up and re-routing.

The worst-case CPU-consumption should be in the order of an hour for 200 nets spanning the entire 200,000 transistor sea-of-gates chip. The typical cases are much better: generally it is much less than a minute.

11 Additional options of *trout*

Clicking Option menu opens a menu with a few more optional features of the router. They are only useful for the final assembly of your layout, and they should (in general) not be used to make leaf cells.

make capacitors This is the most powerful option of all. It causes trout to convert all unused transistors into capacitors between the power lines. In this way the noise on the power lines decreases. This option creates many contacts and therefore makes your layout very big. Generally you should only use it at the highest level of hierarchy.

dangling transistors Connect all unused (dangling) transistors to the appropriate power wire. In this way the state of all transistors is known and the capacity of the power nets increases.

substrate contacts Create substrate contacts under the horizontal power rails. The substrate contacts are required by the design rules of the process.

make fat power lines Make the horizontal power rails as big as possible. In this way the resistance of the power wires decreases and the capacitance increases. *Trout* will also attempt to make as many as possible vertical connections between the horizontal power rails. The latter can

be turned off by clicking [no power]. In a large hierarchical design we advise you to click [flood mesh] as well.

[use borders] Normally the router doesn't use the upper- and lower-most row, which is on top of a power rail. In this way it prevents short-circuits with adjacent cells. In some cases you might wish to switch off this feature by pressing this button.

[no routing] No routing of the signal nets. Use this option to add the special features to the layout (fat power, substrate contacts, etc.). This option turns off the automatic verification of the connectivity.

[flood mesh] In a hierarchical design some fat wires may contain some holes. Especially after using [make fat power lines] this could be the case. Pressing [flood mesh] causes the router to fill all holes in your design. Please note that pressing [Fish] again will undo the result of this option, because *fish* does not have a hierarchical view of your design.

[overlap wires] On the Sea-of-Gates chip a metal2 wire (ins) has a higher probability to crack if it is running in parallel with a metal1 (in) wire. These 'co-incident edges', however, are hard to avoid. To reduce the strain the button [overlap wires] will makes the in wires wider at the spots where they overlap with ins wires. In this way the edges are less co-incident. Please note that pressing [Fish] again will undo the result of this option.

[options] Set any other (unofficial) option which should be propagated to *trout*.

Warning: The options [make capacitors], [make fat power lines], [substrate contacts] and [dangling transistors] will reduce the transparency of the cell considerably. This makes it almost impossible to route new wires through the cell. These options should therefore only be used on the highest level of hierarchy, or for critical cells. A secondary effect of these options is that the router is likely to generate a huge amount of boxes, which makes *seadali* slower and your database very big.

Warning: Pressing [Fish] or [Check nets] will undo the results of the options [flood mesh] and [overlap wires]. Therefore you should not use *fish* anymore after you've used these options.