Projects
Mabuchilab Project Blogs Metablog

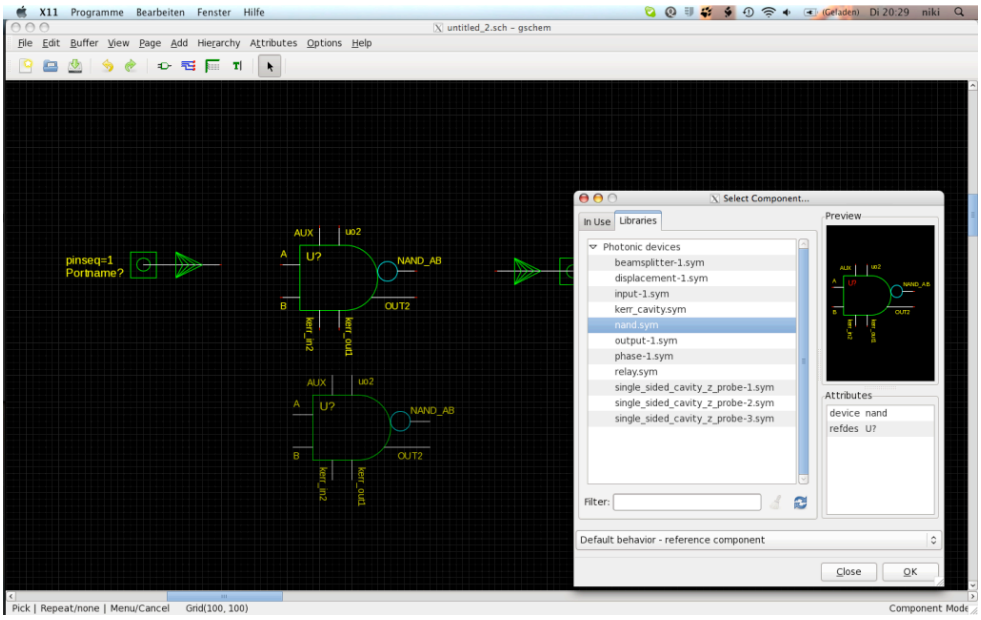
Circuit Design Intro 2: Visual Circuit Design with gschem

Posted on June 21, 2011 by nikolas

In this post, I will explain how to quickly create a schematic from some basic components. Assuming you have logged into mldc1 with X11-forwarding enabled, launch gschem

```
gschem &
```

You will now see a blank grid, waiting to be filled with your circuit. To get started, go to the menu “Add”->”Component” or alternatively press the ‘i’-key of your keyboard. In the pop-up dialog, expand the “Photonic devices” library. Once you have selected a component, move your mouse outside the dialog window over the grid. You should now see a preview of where you would place an instance of that component. In this way you can place multiple instances of the same or different components before closing the dialog window.



You should also add at least one input port object and one output port object. Instances of these special components serve as the global inputs and outputs to your circuit. They are also the only components where the number of inputs and outputs is not equal. Once you have placed some of these instances via the dialog, you can later simply duplicate them without having to reopen the dialog window.

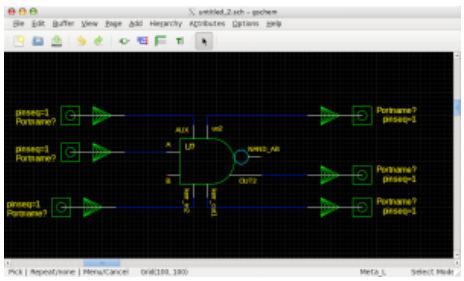
You can select an already placed instance by clicking on it. To select a group of objects, drag a rectangle with the mouse around the full group. Only objects completely within the rectangle are selected. You can rotate a selection of objects in 90° steps by typing ‘er’ on your keyboard. You can duplicate a selection of objects by typing ‘ec’ and clicking where you want to place the copy.

Connecting ports is simple: Hover your mouse over the place where you would like to start a wire (i.e. over some

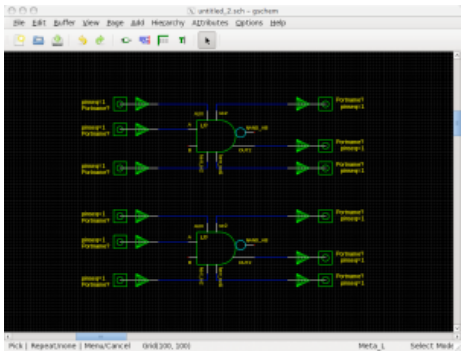
device pin, usually marked red) and press the ‘n’-key. You can then place the wire by clicking your mouse. The device pins are kind of magnetic to the wire-connecting function, so take care not to place your instances too close together. Once you have finished a single connection, right-click your mouse and left-click where you wish to start a new wire.
Now there are some straight-forward rules that any photonic circuit must obey:

1. Your circuit must have an equal number of global inputs and outputs
2. Every port of every instance must be connected to exactly one other port of the same (self-feedback) or another instance.
3. Every global input (output) port must be connected to an instance input (output) port.
4. Every instance output port must be connected to an instance input port or a global output port.

Note that gschem does not verify these conditions, but if you violate them, things are bound to go wrong somewhere down the line.



In this example we create a [NS-NR-latch] out of two NANDs in a feedback configuration.
So far, we have one half of the circuit. Now we select everything and duplicate it.



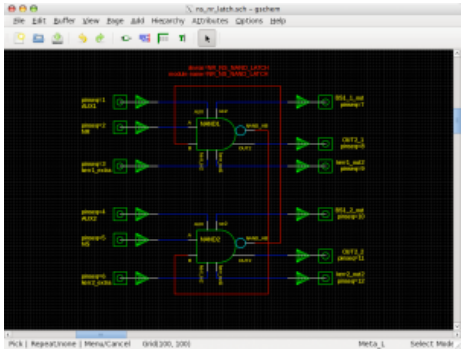
After adding the feedback lines, you may want to highlight these special wires by giving them a different color. To do this, select the feedback connections and type ‘eo’ . A color-dialog window will open and you can select a different color and click on the “apply” button.

Finally, you need to name all your component instances and define an explicit order for your ports. Every embedded instance includes a certain number of attributes. Of these, the 'refdes' attribute specifies the instance identifier. Valid instance specifiers consist of letters, numbers and underscores, much like variables in most modern programming languages. Edit these attributes by double-clicking them. Alternatively, you can double-click the instance itself, whereupon you get a nice dialog to edit all attributes of the instance simultaneously.

The port-sequence is defined by assigning each port a unique 'pinseq' attribute.
Here, it is absolutely necessary, that you obey the convention that all global incoming ports appear above all outgoing ports. Also, if your circuit has a double-digit number of ports, you will need to specify the single-digit portseq entries as "01", "02" (I will try to fix this, but it has to do with the fact that the LISP routines that are used to create the QHDL file read this attribute as a string and not a number).

Furthermore, you can already simplify your final network expression by taking into consideration how your subcircuits are connected with each other. If you already know that the lines emanating from the first n global inputs will not be scattered into any of the remaining photon lines, then it makes sense to also group the corresponding output ports together and make them the first n output lines. This way, you give my algebra-code a chance of successfully factoring the overall network into a concatenation $Q_{1\ldots n} \boxplus Q_{\text{rest}}$.

Finally, you need to add to further attributes to your overall schematic. To do this deselect any component that might be selected by clicking on to the empty grid. Now type 'aa'. You need to add the attributes "device" and "module-name" and set both of them to the same value (this should also be a valid identifier obeying the same conventions as above). This will later become the identifier of your QHDL-entity, so do not choose a name that has already been assigned to something else in the component library.



Once you are finished, save your result to a '.sch' file. You can find this example in my home folder under '/home/nikolas/gschem_examples/ns_nr_latch.sch'.

To generate a qhdl file from this schematic run

```
gnetlist -g qhdl -o my_new.qhdl ns_nr_latch.sch
```

and examine your result:

```
less my_new.qhdl
```

Update:
A common issue of some of the first users has been that not all (unused) ports of the circuit were properly connected to input or output ports. gnetlist actually does not complain about this, but will rather assign such an output port to “OPEN” in the QHDL file. In the future I might add some heuristic to the network parser that will result in a default port ordering for such cases, but until then, it is necessary that you specify all ports in form of the input/output circuit symbols. Alternatively, you can still specify these ports by hand-editing the resulting QHDL file before passing it to the parser.

This entry was posted in computing, Quantum Circuit Algebra, Simulation and tagged Circuit Software Tutorial, quantum network, simulation. Bookmark the permalink.

Projects
Proudly powered by WordPress.