ES 4 Getting started with Radiant

1 Getting started

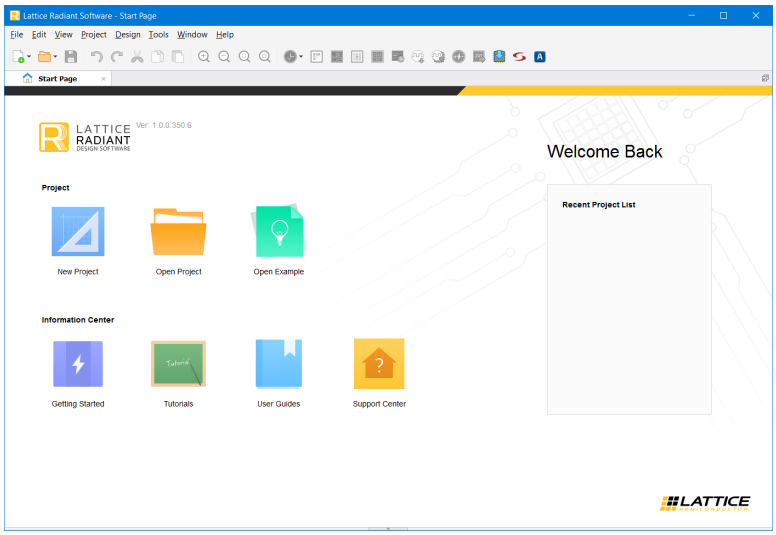
1.1 Tufts labs

If you're using one of the Windows labs (Halligan labs 122, 223, 225, and 229), then browse to `C:\lscc\radiant\1.0\bin\nt64\` and double-click on `pnmain`.

A licensing dialog will pop up. Choose "Specify the License Server" and enter "27001@vm-license1.eecs.tufts.edu".

1.2 Installing on your own computer

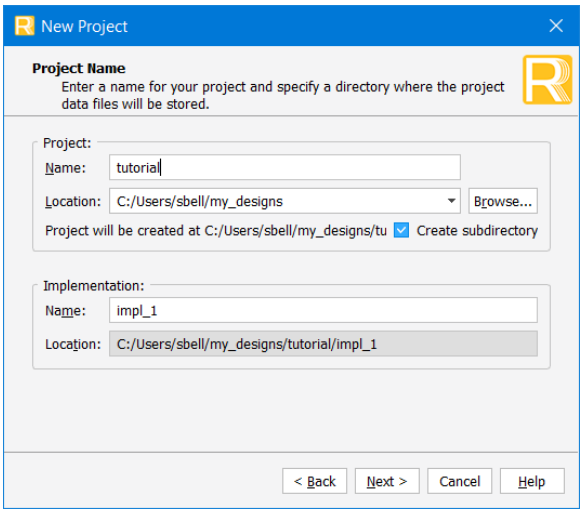
1. Download Radiant from the bottom of this page: http://www.latticesemi.com/Products/DesignSoftwareAndIP/FPGAandLDS/Radiant You will want the basic installer as well as the service pack.
2. Run the installer and follow the prompts
3. Register for a Lattice account. This is necessary to get your license key.
4. Request a licence key here: http://www.latticesemi.com/Support/Licensing/DiamondAndiCEcube2SoftwareLicensing/Radiant You should receive your key by email within a few minutes.
5. Install the license key according to the instructions in the email
6. Try running Radiant — you should see the welcome screen below.



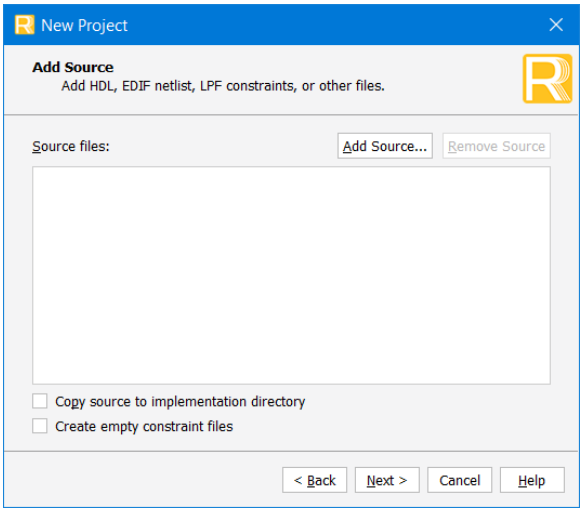
If you want to run Radiant on Linux, talk to Steven — it works, but you may need to implement some hacks to get it running.

2 Creating a new project

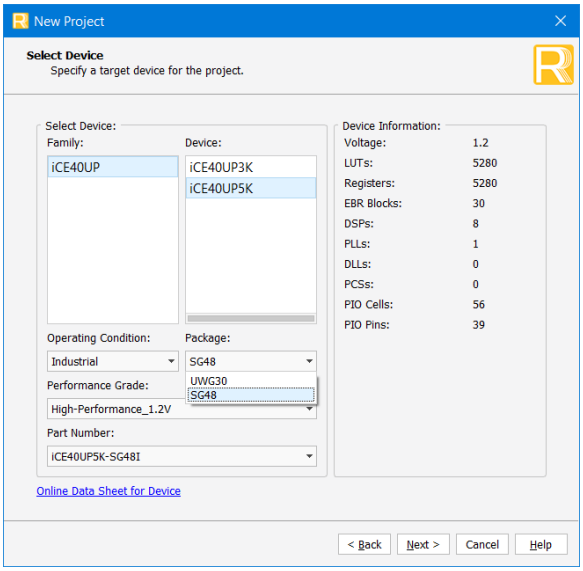
1. Click "New project", either from the welcome screen or the "File→New" menu. Click "Next".
2. Give the new project a name, and change the location if you'd like (Your U: drive is a good choice). Leave the implementation name "`impl_1`". Click "Next".



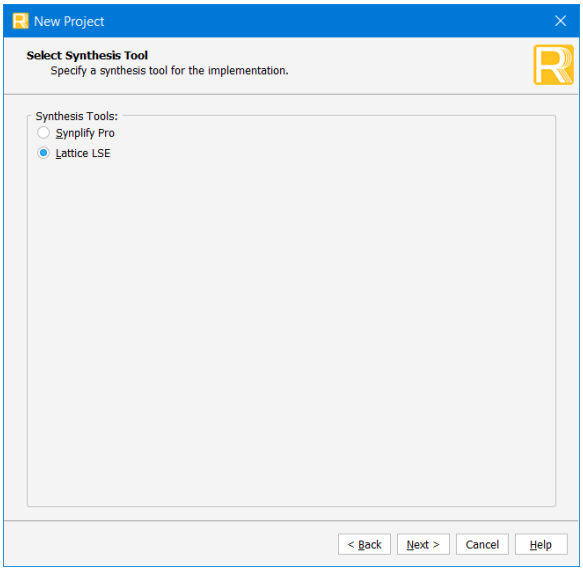
3. You don't need to add any source files right away, so leave this empty and click "Next".



4. Select iCE40UP5K and the SG48 package, and click "Next".



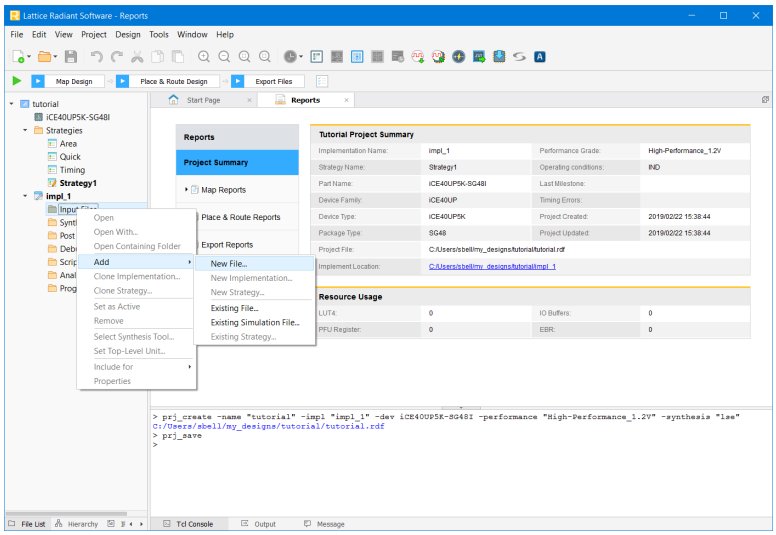
5. Leave the synthesis tool as Lattice LSE, and click "Next".



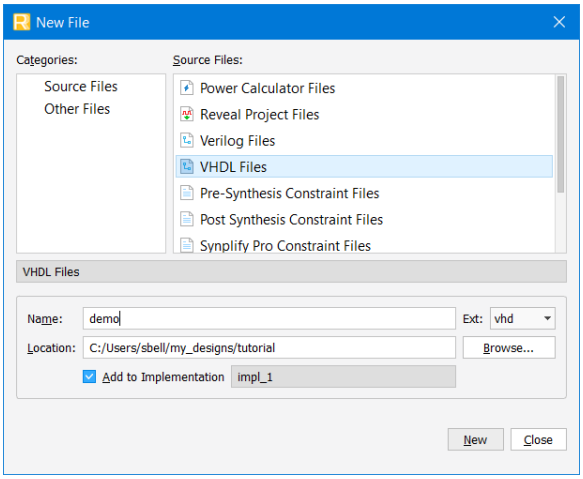
6. Click "Finish" to create the project.

3 Building the design

1. Add a new file: Right-click on "Input files", and select "Add → New File".



2. Select “VHDL files” and give the file a name.



3. Click “New”. The file will be opened in the editor.

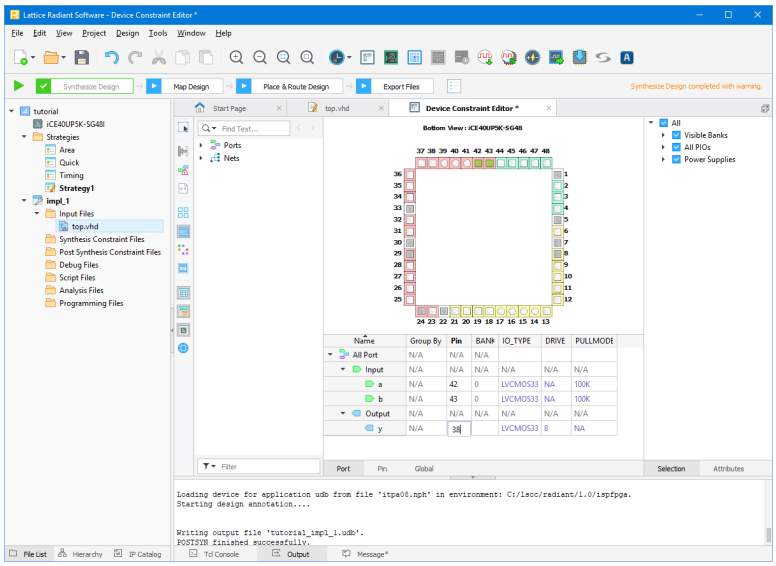
4. For this tutorial, copy the simple AND gate ipmlementation below:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity tutorial is
  port(
          a : in std_logic;
          b : in std_logic;
          y : out std_logic
  );
end tutorial;

architecture synth of tutorial is
begin
  y <= a and b;
end;
```

5. Click “Synthesize design”, and correct any VHDL errors it reports under the ”Message” tab. After synthesizing successfully, you can look at the schematic to see what logic was created by clicking on “Netlist Analyzer” 

6. Next we need to assign the ports on the entity to physical pins on the FPGA board. To do this, open the “Device Constraint Editor”. 

7. You should see the ports from your VHDL entity listed. Simply type in the pin number that you want to correspond to each port.



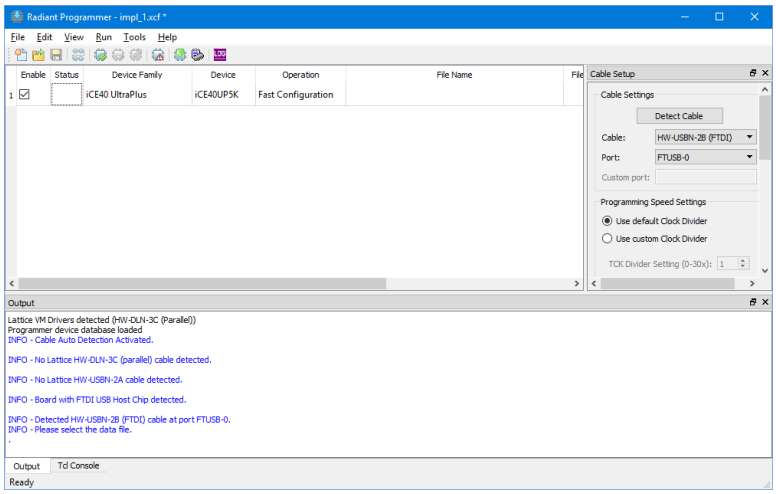
8. After assigning all of the ports to pins, click “Export Files” to run the rest of the build process (map, place & route, and export).

4 Flashing the FPGA

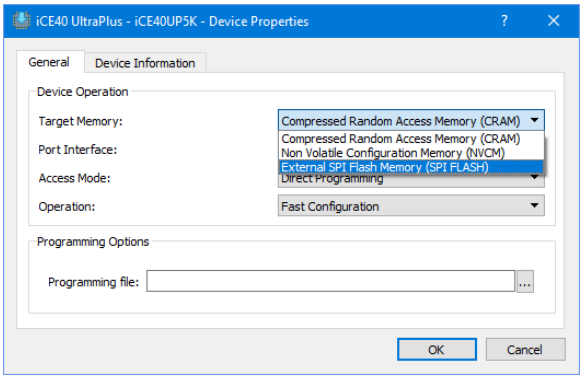
Once your design is built, you’ll need to “flash” it to the FPGA.

1. Plug your UPDuino into the computer via the Micro USB port

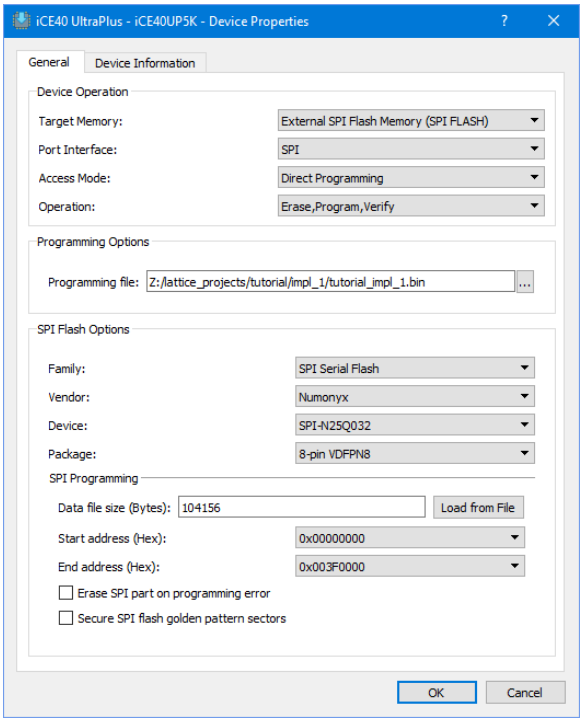
2. In Radiant, click “Programmer”  A new window will pop up:



3. Click “Detect Cable”
4. If “Device family” shows “Generic JTAG Device”, click it and select “iCE40 UltraPlus”.
5. If “Device” shows iCE40UP**3**K, change it to iCE40UP**5**K.
6. Under “Operation” double-click the item to bring up the configuration dialog.



7. Change “Target memory” to “External SPI Flash”



8. Select your `*.bin` programming file under 'Programming file'. This will be in the `impl_1` directory, and is usually called `PROJECTNAME_impl_1.bin`.
9. Configure the following under "SPI Flash Options"
   (a) Vendor: Numonyx
   (b) Device: SPI-N25Q032
   (c) Package: 8-pin VDFPN8
10. Click "OK" to return to the programmer.
11. Click "Program Device" Afer a few seconds, you should see the message `INFO - Operation successful` and the design should start running on the FPGA. You may get a Windows firewall warning; this should not affect the operation of the programmer.

That's it! Take a moment to celebrate your first FPGA design!