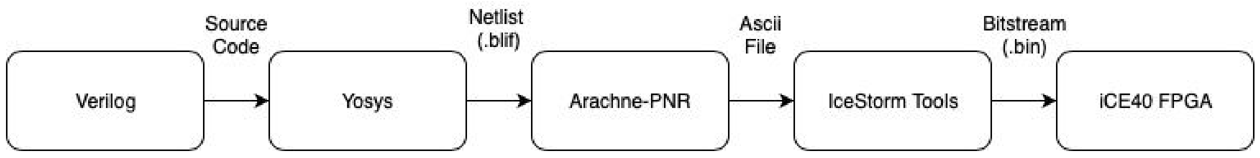
How to use Project IceStorm to program a Lattice iCE40 FPGA on the Upduino Board

Overview:

This guide indicates a workflow to synthesize and upload programs based on open-source tools. The workflow (Using Yosys, Arachne-PNR, and IceStorm Tools) is a fully open source Verilog-to-Bitstream flow for iCE40 FPGAs. This workflow includes synthesizing a verilog file by Yosys, place-and-route by Arachne-PNR, and uploading by Icestorm.

Description:

Project IceStorm is a completely open-source workflow to upload designs developed in verilog onto Lattice iCE40 LP/HX 1K/4K/8K chips as well as iCE40 UltraPlus parts. Project IceStorm is a workflow which uses Yosys to perform logic synthesis, Arachne-PNR to perform place & routing, and IceStorm Tools to perform packing and uploading bitstream files. Yosys is an open-source framework for synthesizing verilog into gate-level netlists in various formats including BLIF, EDIF, BTOR, SMT-LIB, simple RTL Verilog, etc. Arachne-PNR accepts netlists in BLIF format and performs place & routing for the iCE40 family of FPGAs. IceStorm Tools consists of a drivers, packer/unpacker, and some other tools for working with iCE40 bitstream files. Together, we utilize them as:



Purpose:

This guide focuses on the use of the open-source FPGA tools included in Project IceStorm to program the Lattice iCE40 family of FPGAs. In this toolchain, IceStorm is a bitstream generator, Arache-PNR is a place & route tool, and Yosys is a logic synthesis tool. Because Yosys is can only used for Verilog RTL synthesis, Verilog will be our HDL of choice.

Required Tools:

Software: IceStorm, Arachne-PNR, Yosys. We will be using Ubuntu Linux in this guide.
Hardware: Lattice iCE40 FPGA. In this guide we will be using the Gnarly Grey UpDuino v2.0 board.
Demo Code: In this guide we will be using a sample code provided by IceStorm, rgb.v. This can found in the folder in ~/icestorm/examples/up5k_rgb.

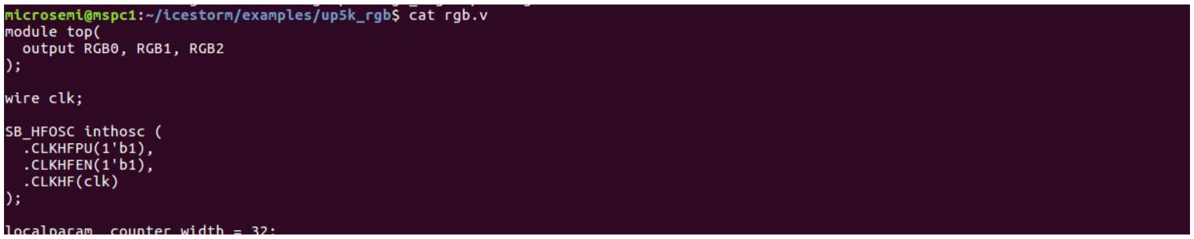
Notes for Linux: Create a file /etc/udev/rules.d/53-lattice-ftdi.rules with the following line in it to allow uploading bit-streams to a Lattice iCEstick and/or a Lattice iCE40-HX8K Breakout Board as unprivileged user:
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6010", MODE="0660", GROUP="plugdev", TAG+="uaccess"

Quick Guide:

0. Make sure to download and install IceStorm and Yosys and that you are in the correct file location.(EX. ***cd ~/icestorm/examples/up5k_rgb***)
1. ***(optional) cat [filename].v*** to view contents of the verilog file
2. ***(optional) cat [filename].pcf*** to view physical constraints file
3. ***yosys -p "read_verilog [filename].v; synth_ice40 -blif [filename].blif"*** create .blif file using yosys. This file contains the logic-level circuit in textual form.
4. ***arachne-pnr -d 5k -p [filename].pcf -o [filename].txt [filename].blif*** create .txt file from the .blif file using arachne. This performs the place and route step of the hardware compilation process for FPGAs.
5. ***(optional) cat [filename].txt*** to view the config bit file
6. ***icepack [filename].txt [filename].bin*** converts ASCII file that contains config bits for the chip into an iCE40 .bin file
7. ***iceprog [filename].bin*** upload bitstream onto the Lattice iCE 40 FPGA (Note: ***sudo iceprog [filename].bin*** required if user does not have appropriate permissions. If such is the case, error will be: "Can't find iCE FTDI USB Device" if this is the case)

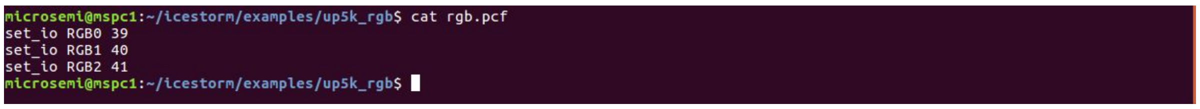
Full Guide:

Step 1 (optional):
Enter "cat [filename].v" to view the contents of the file and ensure it is the correct file you want loaded. You should see the source code of the file you selected.



```
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ cat rgb.v
module top(
  output RGB0, RGB1, RGB2
);

wire clk;

SB_HFOSC inthosc (
  .CLKHFPU(1'b1),
  .CLKHFEN(1'b1),
  .CLKHF(clk)
);

localparam  counter_width = 32;
```

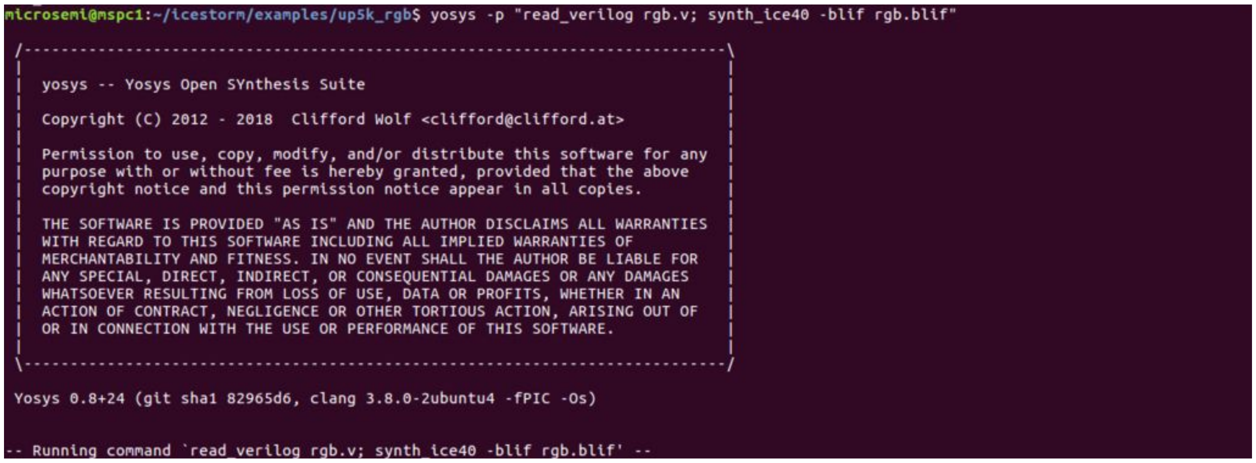
Step 2 (optional):
Enter "cat [filename].pcf" to view the contents of the physical constraint file. You should see something similar to the screenshot below:



```
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ cat rgb.pcf
set_io RGB0 39
set_io RGB1 40
set_io RGB2 41
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ 
```

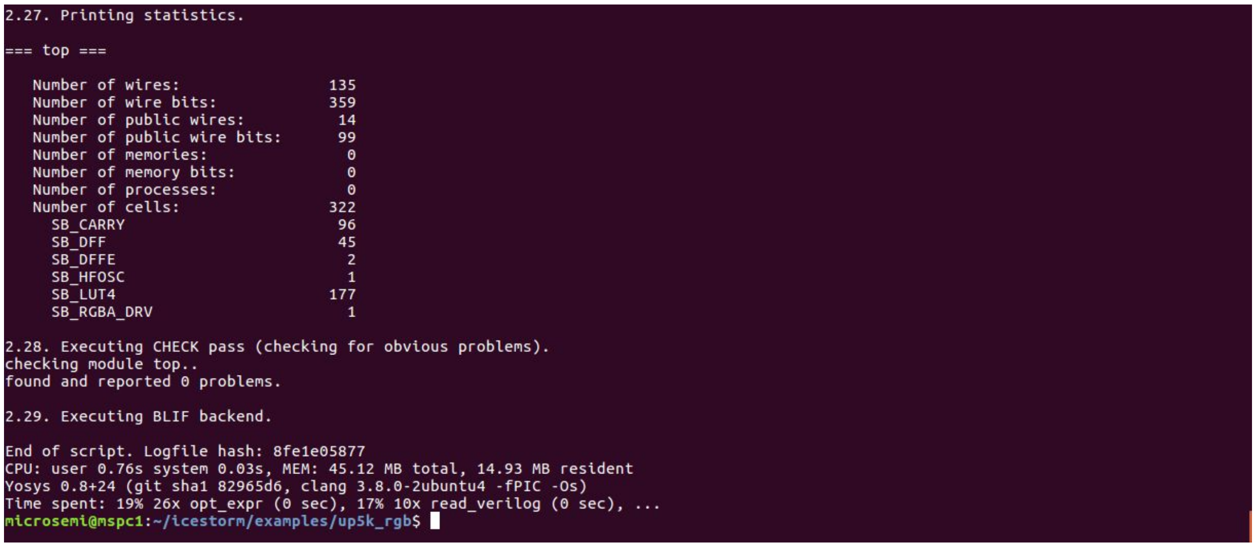
Step 3:

Enter “ yosys -p "read_verilog [filename].v; synth_ice40 -blif [filename].blif" “ to create .blif file using yosys. This file contains the logic-level circuit in textual form. You should see something similar to the screenshot below:



```
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ yosys -p "read_verilog rgb.v; synth_ice40 -blif rgb.blif"

 /----------------------------------------------------------------------------\
 |                                                                            |
 |  yosys -- Yosys Open SYnthesis Suite                                       |
 |                                                                            |
 |  Copyright (C) 2012 - 2018  Clifford Wolf <clifford@clifford.at>           |
 |                                                                            |
 |  Permission to use, copy, modify, and/or distribute this software for any  |
 |  purpose with or without fee is hereby granted, provided that the above    |
 |  copyright notice and this permission notice appear in all copies.         |
 |                                                                            |
 |  THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES  |
 |  WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF          |
 |  MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR   |
 |  ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES    |
 |  WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN     |
 |  ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF   |
 |  OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.            |
 |                                                                            |
 \----------------------------------------------------------------------------/

 Yosys 0.8+24 (git sha1 82965d6, clang 3.8.0-2ubuntu4 -fPIC -Os)


-- Running command `read_verilog rgb.v; synth_ice40 -blif rgb.blif' --
```

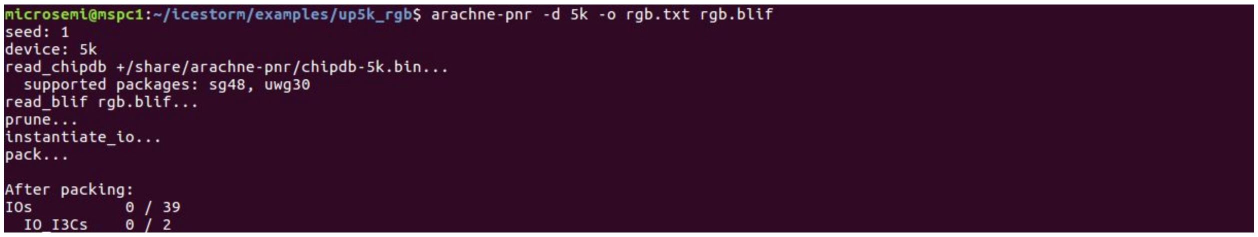
With a similar result as below as the result if successful:



```
2.27. Printing statistics.

=== top ===

   Number of wires:                135
   Number of wire bits:            359
   Number of public wires:          14
   Number of public wire bits:      99
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                322
     SB_CARRY                       96
     SB_DFF                         45
     SB_DFFE                         2
     SB_HFOSC                        1
     SB_LUT4                       177
     SB_RGBA_DRV                     1

2.28. Executing CHECK pass (checking for obvious problems).
checking module top..
found and reported 0 problems.

2.29. Executing BLIF backend.

End of script. Logfile hash: 8fe1e05877
CPU: user 0.76s system 0.03s, MEM: 45.12 MB total, 14.93 MB resident
Yosys 0.8+24 (git sha1 82965d6, clang 3.8.0-2ubuntu4 -fPIC -Os)
Time spent: 19% 26x opt_expr (0 sec), 17% 10x read_verilog (0 sec), ...
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ 
```

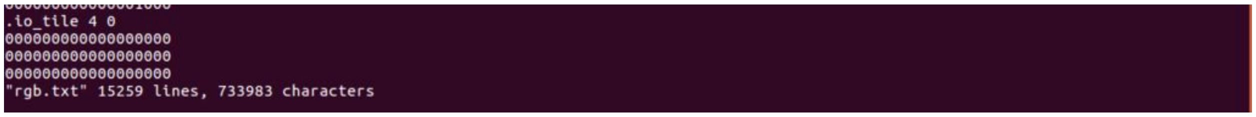
Step 4:

Enter “***arachne-pnr -d 5k -p [filename].pcf -o [filename].txt [filename].blif*** ” to create .txt file containing configuration bits using arachne. This performs the place and route step of the hardware compilation process for FPGAs.



```
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ arachne-pnr -d 5k -o rgb.txt rgb.blif
seed: 1
device: 5k
read_chipdb +/share/arachne-pnr/chipdb-5k.bin...
  supported packages: sg48, uwg30
read_blif rgb.blif...
prune...
instantiate_io...
pack...

After packing:
IOs          0 / 39
  IO_I3Cs    0 / 2
```

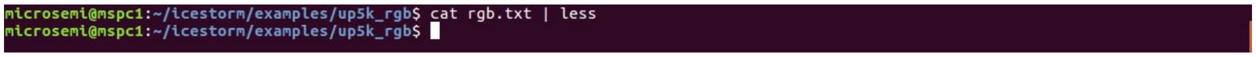
You should receive a similar result as below if the result is successful:



```
.io_tile 4 0
000000000000000000
000000000000000000
000000000000000000
"rgb.txt" 15259 lines, 733983 characters
```

Step5 (Optional):

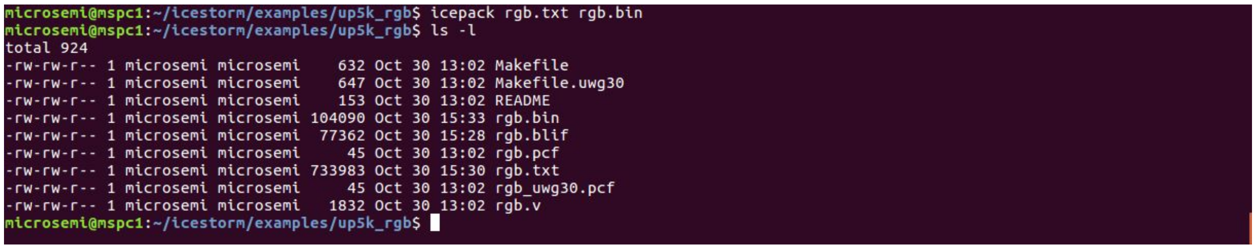
Enter "cat [filename].txt" if you wish to see the contents of the .txt file. (Output not shown because it is very long)



```
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ cat rgb.txt | less
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ 
```

Step 6:

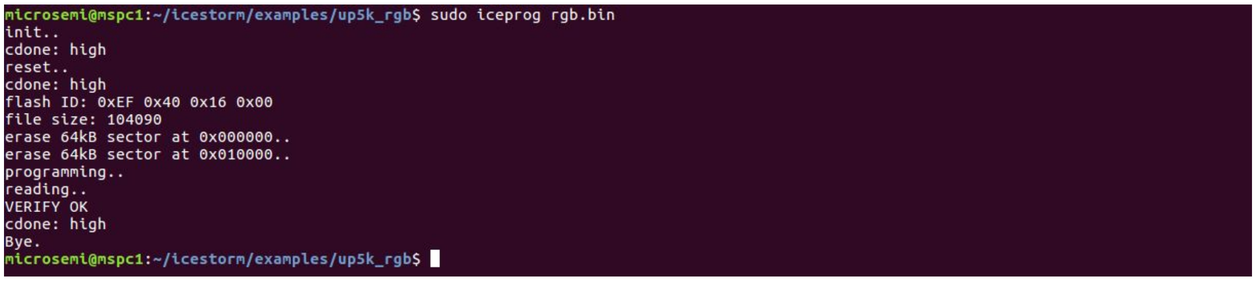
Enter "icepack [filename].txt [filename].bin" to convert the ASCII .txt file that contains config bits for the chip into an iCE40 .bin file.



```
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ icepack rgb.txt rgb.bin
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ ls -l
total 924
-rw-rw-r-- 1 microsemi microsemi    632 Oct 30 13:02 Makefile
-rw-rw-r-- 1 microsemi microsemi    647 Oct 30 13:02 Makefile.uwg30
-rw-rw-r-- 1 microsemi microsemi    153 Oct 30 13:02 README
-rw-rw-r-- 1 microsemi microsemi 104090 Oct 30 15:33 rgb.bin
-rw-rw-r-- 1 microsemi microsemi  77362 Oct 30 15:28 rgb.blif
-rw-rw-r-- 1 microsemi microsemi     45 Oct 30 13:02 rgb.pcf
-rw-rw-r-- 1 microsemi microsemi 733983 Oct 30 15:30 rgb.txt
-rw-rw-r-- 1 microsemi microsemi     45 Oct 30 13:02 rgb_uwg30.pcf
-rw-rw-r-- 1 microsemi microsemi   1832 Oct 30 13:02 rgb.v
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ 
```

Step 7:

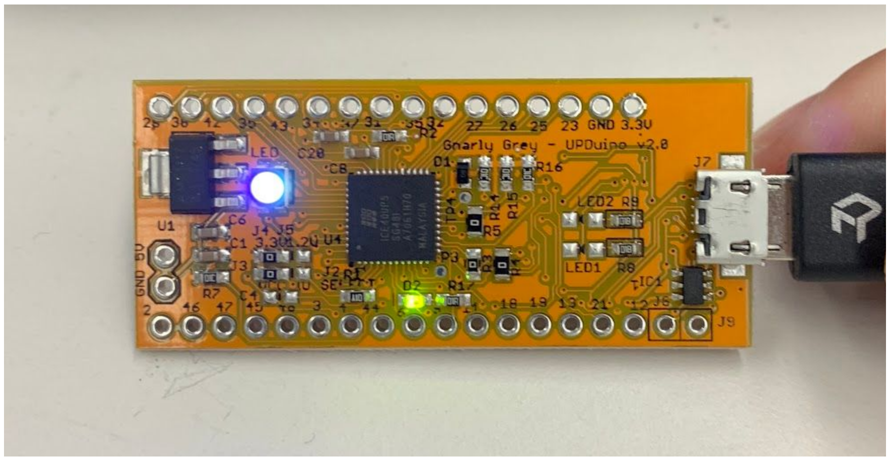
Enter "iceprog [filename].bin" to upload bitstream onto the Lattice iCE 40 FPGA (Note: sudo iceprog [filename].bin required if user does not have appropriate permissions. If such is the case, error will be: "Can't find iCE FTDI USB Device" if this is the case)



```
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ sudo iceprog rgb.bin
init..
cdone: high
reset..
cdone: high
flash ID: 0xEF 0x40 0x16 0x00
file size: 104090
erase 64kB sector at 0x000000..
erase 64kB sector at 0x010000..
programming..
reading..
VERIFY OK
cdone: high
Bye.
microsemi@mspc1:~/icestorm/examples/up5k_rgb$ 
```

Final result:

Your FPGA should have the program uploaded at this point. If you used the same source design as shown in this guide, the rgb should be slowly changing color. Shown here is when the rgb was blue.


Gnarly Grey - UPDuino v2.0