OpenFPGA Step by Step Guide

Hao Jun Liu
EECS Department, UC Berkeley
hjliu@eecs.berkeley.edu

July 9, 2014

1 Introduction

This is a step by step guide on compiling and running the OpenFPGA tool flow. If there is any question, please contact the author via email.

2 Checking Out OpenFPGA

```
svn checkout file:///users/hjliu/ProjSVN/open-fpga-complete
```

3 Building VTR

```
cd open-fpga-complete/vtr
make
```

From this point on, we reference all directories relative to the project directory (open-fpga-complete).

4 Create an FPGA Architecture

In this section, we will create a new FPGA architecture that is different from those that come with the project. The FPGA we are going to create is an 8 by 8 tile, 6-input LUT, 4-LUT clustered CLB. It has 96 interconnects in each direction. Let's name the fpga configuration vpr_6_4_10_10_96_8_8.xml. The naming convention is:

```
vpr_{LUT SIZE}_{CLB SIZE}_{IPIN W}_{CHANXY W}_{INTERCONNECTS}_x{X SIZE}_y{Y SIZE}.xml
```

The naming convection has to be kept as tools use it to derive variables. IPIN W and CHANXY W are informational only. The tools do not need them. In order for the existing flow to work, INTERCONNECTS has to be an integer multiple of 8.

```
cd vtr/vtr_flow/arch/timing
cp vpr_6_10_16_16_160_x5_y5.xml vpr_6_4_10_10_96_x8_y8.xml
```

There are a few lines to be modified. Here are them:

```
L10: <layout width="8" height="8"/>
L15: <area grid_logic_tile_area="8000"/>
L85: <input name="I" num_pins="15" equivalent="true"/>
L86: <output name="O" num_pins="4" equivalent="false"/>
L90: <pb_type name="ble" num_pb="4">
```

```
L134: <complete name="crossbar" input="clb.I ble[3:0].out" output="ble[3:0].in">
L135: <delay_constant max="4.000000e-11" in_port="clb.I" out_port="ble[3:0].in" />
L136: <delay_constant max="4.000000e-11" in_port="ble[3:0].out" out_port="ble[3:0].in" />
L138: <complete name="clks" input="clb.clk" output="ble[3:0].clk">
L140: <direct name="clbouts" input="ble[3:0].out" output="clb.O">
```

5 Create a New Test Circuit

We create a new circuit in this case, a decremental counter with a width of 16. Let name this dcounter.v. We also need to create testbench and fpga testbench wrappers.

```
mkdir dcounter
cp counter/counter_tb.v dcounter/dcounter_tb.v
cp counter/counter.v dcounter/dcounter.v
cp counter/fpga_wrapper.v dcounter/fpga_wrapper.v
cp counter/run.sh dcounter/run.sh
cp counter/counter_fpga_tb.v cp dcounter/dcounter_fpga_tb.v
ln -s dcounter/dcounter.v dcounter.v
```

Edit dcounter.v, dcounter_tb.v and run.sh. Now simulate the dcounter with vsim.

```
vlib work
vlog dcounter_tb.v dcounter.v
./run.sh > run.log
```

Check run.log to make sure it is functional correct.
Use the counter circuit as an example; edit dcounter_fpga_tb.v and fpga_wrapper.v

6 Run VTR Flow

Edit the VTR config file and run test circuit.

```
cd vtr/vtr_flow/tasks/func_test_circuit/config
vim config.txt
cd vtr/vtr_flow/scripts
./open_fpga_run_vtr_task.pl func_test_circuit
```

All circuits in different FPGA architectures should run OK.

7 Create Test Dirs

```
cd full-fpga-testing
mkdir custom_ms
cd custom_ms
mkdir dcounter counter ff_en multi_consumer simple_comp single_inv single_inv_reg src
wide_inv wide_inv_reg
cd src
ln -s ../../../archipelago/verilog/tools/cm_sp.v cm_sp.v
```

8 Run OpenFPGA

Create Directory Strcuture and run the flow.

```
cd vtr/vtr_flow/tools
mkdir fpgaGen-vpr_6_4_10_10_96_x8_y8
../create_simlink.sh
```

```
cp ../fpgaGen-vpr_6_10_16_16_160_x5_y5/*sh .
rm -f temp.sh
mv run_5_5_bs.sh run_8_8_bs.sh
mv run_5_5.sh run_8_8.sh
```

Edit shell scripts for the architecture. Teh run_8_8.sh is an important one. You need to put the right parameters and directories as arguments to the Open FPGA tools. The --configs-en argument in the *_bs.sh can be updated after you run the flow. You can get the configuration depth after the flow. After the update, run the vtr flow first and then this flow again.

```
./run_all.sh
vim run_8_8_bs.sh
./run_all.sh
```

9 Test Bitstreams on FPGA

```
full-fpga-testing/custom_ms/dcounter
vlib work; compile.sh
run.sh
```