FPGA lab

Andreas Ehliar

June 30, 2010

1 Lab environment

If you have an account at ISY, just run the following command on a Linux computer to setup the paths required to access Xilinx ISE 11.

```
module add xilinx/11.3i
```

If you don't have an account at ISY, the Webpack version of ISE (which you can download for free from Xilinx) is hopefully enough to run the labs, but you still need to have access to Modelsim.

1.1 Lab code

The lab code is available for download on the course homepage. The design module is an FFT processor which is based on the architecture shown at the end of this lab manual. (For more information about this kind of FFT processor, see *"A New Approach to Pipeline FFT Processor"* by Shousheng He and Mats Torkelsson (published at IPPS'96).

1.2 Target FPGA

For task 1-3 you need to use the xc4vlx15-12 device (Virtex 4 LX 15, speed-grade 12). For the remaining tasks, feel free to select another device if you want to. (The makefile should work for other devices...)

1.3 Important makefile targets

The Makefile included in the lab contains the following targets:

- **clean** Removes all generated files, backup files, etc.
- **lab.sim** Simulates the design in Modelsim
- **lab.simc** Runs the testbench in modelsim in console mode. Useful if you just want to check that the design (still) works. It is probably a good idea to run this command often during development to make sure that you haven't inadvertedly changed the functionality of the design.
- **lab.synth** Synthesizes the design using XST
- **lab.route** Place and route the design
- **lab.timing** Generate static timing report
- **lab.fmax** Try to route the design using a few different timing constraints based on TIMESPEC in the Makefile. 7 attempts are made, each attempt lowers the timing constraint with 0.1 ns.
- **lab.synthsim** Simulates the post synthesis simulation model
- **lab.synthsimc** Same as above but in console mode
- **lab.parsim** Simulates the post place and route simulation model
- **lab.parsimc** Same as above but in console mode
- **lab.bitgen** Generate a programming file (not needed for the labs)
- **lab.powersim** Run a power simulation
- **lab.xdl** Create an XDL (ascii based netlist format) version of your routed design.

Note that all makefile targets (should) have dependency tracking. That is, if you run for example make lab.timing, it will also synthesize the design and place and route the design if necessary.

1.4 Important files

- **Makefile** If you add any files to the design you will need to add them in this file as well.
- **lab.ucf** This file contains the timing constraints. It can also contain other kinds of constraints (floorplanning, etc), but no such constraints are included in the default file.
- **lab-synthdir/synth/design.syr** The synthesis report file
- **lab-synthdir/layoutdefault/design.par** The place and route report
- **lab-synthdir/layoutdefault/design_map.mrp** The mapping report file
- **lab-synthdir/layoutdefault/design.twr** Static timing analysis
- **lab-synthdir/layoutdefault/design.xdl** An XDL netlist

2 Task 1 - Setting up the flow

Familiarize yourself with the design flow. The design is using a makefile based flow, but if you already know and are comfortable with how to do all of the tasks mentioned above using the ISE gui, feel free to do that instead.

Run all parts of the flow. Notice that the post synthesis and post place and route simulation does not seem to work correctly. Sometimes, this is caused by a bug in the testbench, for example by violating setup or hold constraints. However, in this case it is caused by a nasty mistake in the VHDL source code which causes a synthesis/simulation mismatch. You should fix this bug. Take a look at the synthesis report file to see if you can figure out what the problem is. (If you can't figure it out, take a look at the end of this document for a hint towards where to look in the design to find the bug.)

3 Task 2 - Determine operating parameters

Determine the maximum operating frequency, area, latency, and power requirements of the unoptimized design. Familiarize yourself with the timing

report file.

4 Task 3 - Performance optimizations

Improve the performance of the design. I suggest aiming for 250 MHz in a xc4vlx15-12 (the PART selected in the Makefile). If this turns out to be easy for you, reaching 300 MHz should still be quite a bit of a challenge. However, if it turns out that you cannot reach 250 MHz, you should at least aim for 200 MHz, and you will also need to spend a bit more effort on the remaining parts of this lab.

In theory, it is probably possible to reach a frequency of 500 MHz which is the limit of the BlockRAMs and DSP48 blocks, but doing so will require quite a lot of effort. *However, if you can reach a performance of over 450 MHz, you may skip task 6 if you want to.*

4.1 Hints for task 3

- Take a look at the static timing analysis report to figure out what the worst path is.
- Pipelining the design is relatively easy as the control vector "ctrl" is routed through each layer. Adding one pipeline layer is merely a matter of pipelining the ctrl and the Data_out vectors in BF_I_FIFO and BF_II_FIFO. Also, the testbench is written in such a way that the length of the pipeline should not matter.
- You may also have to add a register to the output of the FIFO module. This can also be done easily for the LARGEFIFO part by modifying when fifo_rden is activated.
- To reach a high clock frequency you will probably need to add more than one pipeline register per BF_x_FIFO module.
- The easiest way to get a working design after adding a register or two to the coefficient memory modules is to change the "Comparator_value" generics in the BF_I_FIFO module.

5 Task 4 - Memory optimizations

You only need to do one of these two tasks:

5.1 Task 4a - Optimize the usage of the memories

There is a constant in the FIFO module that determines the threshold for when a memory is used and when a shift register is used. Try to determine the optimum setting for this variable, both in terms of area, and in terms of power. Hint: An area efficient shift register based on SRL16 is not allowed to have a reset signal...

5.2 Task 4b - Minimizing the coefficient memories

There are three coefficient memories, each 24 bits wide. After optimizing your design it is likely that these coefficient memories will consume one blockram each. All in all, these three memories need to deliver 72 bits every clock cycle. This means that it should be possible to fit all the coefficient memory into one blockram (assuming both ports are used). Your task is to perform this area optimization.

6 Task 5 - Power simulation

Compare the power of your design when running an FFT and when it is idle (that is, rst is asserted). Hint: You will need to modify the testbench here so that you can simulate the same time period with and without rst asserted. You will also need to modify your testbench so that you run it at an appropriate clock frequency...

(Note that there seems to be some sort of problem so that we don't get full signal coverage in power simulation. I don't know why this occurs, but if you get around 90% coverage it should be ok.)

7 Task 6 - Power optimization

You only need to do one of these tasks:

7.1 Task 6a - Add a clock mux or a clock enable to your design

The idea is to reduce the idle power consumption of the design. You are not allowed to change the interface of the design (but you are allowed to change the latency). You must also handle clock domain crossings safely! Hint: Take a look at the documentation of the BUFGMUX and BUFGCE primitives in the library documentation guide for information on how to instantiate a clock mux in an FPGA.

7.2 Task 6b - Optimizing memories for power

Add enable signals to your BlockRAM and SRL16 based memories so that they are only used when rst is deasserted. You will (probably) need to manually instantiate BlockRAM or SRL16 primitives in order to do this.

8 Resources

In the Xilinx ISE 11 directory there is a variety of documentation files which could be of interest. For example:

```
$XILINX/doc/usenglish/isehelp/xst.pdf
$XILINX/doc/usenglish/isehelp/virtex4_hdl.pdf
$XILINX/doc/usenglish/isehelp/devref.pdf
$XILINX/doc/usenglish/isehelp/cgd.pdf
```

You may also want to look into various user guides, especially the parts on how to instantiate blockrams and possibly DSP48 blocks. Search for ug070.pdf and ug073.pdf.

9 Running PlanAhead

To use the PlanAhead tool you first need to have a routed design and a static timing report. (That is, you need to run make lab.timing.) Unfortunately this is not yet integrated into the makefile.

To start the planehad tool, use the following command:

```
$XILINX/../PlanAhead/bin/planAhead
```

To analyze a design:

- Select File->New project
- Specify some project name and some project location (the defaults should be fine)
- Import the ISE Place and Route results
- Select the netlist (lab-synthdir/synth/design.ngc)
- Choose a part and a floorplan name (The defaults should be ok)
- Add the lab.ucf file (not required)
- Import ISE Implementation results
  - For placement: select lab-synthdir/layoutdefault/design.ncd
  - For timing, select lab-synthdir/layoutdefault/design.twx)

When you have started PlanAhead you will see a graphical view of the FPGA and at the bottom you will get the timing results. Clicking on a path in the timing results will show this path

10 The lab report

A high quality lab report should be sent to `andeh78.liu@analys.urkund.se` when you are done. You should also send your modified source code to `ehliar@isy.liu.se`.

11 Hint for task 1

The hint is reversed, so you don't read it by mistake:

.elif troper sisehtnys eht ni strap esoht ta gnikool no etartnecnoc os ,dhv.lortnoc ni stsixe gub ehT

Architecture of the FFT