# Implementing a Xilinx Project

Tom Kelliher, CS 240

Apr. 4, 2008

So, you've synthesized your VHDL design and now it's time to implement it, generate the bitstream file, and download the bitstream into the FPGA to configure it according to your design. This document will take you through that process using the Xilinx and XSTOOLS software:

- Required VHDL code for clock signals. **Do this after synthesizing and simulating.**
- Adding pin constraints to your design.
- Implementing your design and generating a bitstream file.
- Simulating your design after adding the VHDL clock signal code.
- Testing the FPGA board (XSTOOLS).
- Downloading a bitstream into the FPGA board (XSTOOLS).

I assume you have a synthesized design upon which you've performed behavioral simulation.

## 1 Required VHDL Code for Clock Signals

You can hold off on adding the buffers below until you're ready to download your design into a Xilinx board. **However, I recommend you use the clock naming convention described below.**

First off, you'll need to verify that you're using the correct device. In the *Sources for* pane, verify that the device you're using is `xc2s50-5tq144-XST VHDL`. If not, right-click on the line, choose *Properties*, and set the properties as follows:

- Device Family: `Spartan2`
- Device: `xc2s50`
- Package: `tq144`
- Speed Grade: `-6`

Following this, open the *Project* menu and select *Cleanup Project Files* to clear out any old, erroneous information. (Cleaning up the project files will often correct "weird" errors you may be getting.)

Due to their importance, clock signals require special handling. You will need to add the following to your VHDL code. I will assume that you named your external clock signal (the name given in the port list of the entity declaration) `clk_ext` and that you are using the signal `clk` internally to clock your flip-flops. (Otherwise, modify the following for your naming convention.) Initially, you should tie `clk` to `clk_ext` like so:

```
signal clk: std_logic;

...

begin

clk <= clk_ext;
```

When it's time to add the buffers, comment-out your assignment to `clk` and add the following to the declarations section of your architecture body:

```
  component ibuf port (
    i : in  std_logic;
    o : out std_logic);
  end component;

  component bufg port (
    i : in  std_logic;
    o : out std_logic);
  end component;

  -- clk is the internal clock name.  Modify as necessary.
  signal clk_pad, clk : std_logic;
```

Add this to the body of your architecture:

```
  u1 : ibuf port map (
    i => clk_ext,   -- External clock name.  Modify as necessary.
    o => clk_pad);

  u2 : bufg port map (
    i => clk_pad,
    o => clk);   -- Internal clock name.  Modify as necessary.
```

The clock signal that you will now use to control your flip-flops is `clk`.

## 2 Adding Pin Constraints

1. Add pin constraints *after* implementing your project once (see the next Section).
2. With your project open, create a new source file. For the type of source file, choose `Implementation Constraints File`. Once you've created the new file, highlight it in the *Sources for* window and double-click `Assign Package Pins` in the *Processes for Current Source* window.
3. The Xilinx PACE tool will run.
4. Within the *Design Object List — I/O Pins* window, you will see the names of all your project's port signals listed. To constrain these I/O signals to specific FPGA pins, you will enter values for the *Location* field. The values to be used for the problems will be given to you separately.

**Be careful when filling in this information. Doing this incorrectly can result in damage to the FPGA board, the PC, or both. Not to mention what it will do to your wallet!**

5. Save the constraints and exit the tool.

6. Re-implement your design.

**Note:** Any time you modify the constraints, you will need to re-implement the design and generate a new bitstream file.

# 3 Implementing a Design and Generating a Bitstream File

1. In the *Sources for* window, highlight your VHDL file. Then, in the *Processes for Current Source* window, double-click `Implement Design`. Your design will be implemented. Correct any errors. (If you neglect to add the VHDL for clock signals, you will receive several fatal errors at this point.)

2. Expand the *Implement Design* tree by clicking on the `+` symbol before it. Find and open the `Pad Report` and confirm that your I/O signals were indeed constrained to the correct FPGA pins. If they were not, go back and correct your constraints.

   Do not proceed until the Pad Report is correct.

3. Contract the *Implement Design* tree.

4. Once your design has been successfully implemented, double-click `Generate Programming File` to generate your design's bitstream file.

# 4 Simulating Your Design After Adding the VHDL Clock Signal Code

Once you've added the VHDL for the clock signal, the behavioral VHDL simulation will no longer work properly. However, the Post-Translate VHDL simulation will, so use that instead if you need to perform further simulation.

# 5 Testing the FPGA Board

You can do this to verify that the board is functioning correctly.

- From the Start menu, go to Programs, then XSTOOLS, then choose GXSTEST.
- Set Board Type to XSA-50 and Port to LPT1.
- Press the TEST button. GXSTEST will report the result of the test.

## 6 Downloading a Bitstream

1. Open your design folder and locate your bitstream file. It will have a `.bit` extension.
2. From the Start menu, go to Programs, then XSTOOLS, then choose GXSLOAD.
3. Make sure that the Board Type is set to XSA-50 and that Port is set to LPT1.
4. Drag your bitstream file into the FPGA/CPLD area of GXSLOAD. GXSLOAD will download the highlighted bitstream file into the FPGA when you click the `Load` button.

   If the bitstream file was successfully downloaded, the decimal point on the seven-segment display will be lit. **Anytime** the decimal point is not lit, the FPGA is **not** correctly configured.
5. You're now ready to use the FPGA.