

## Digilent S3 Board Tutorial

Washington University in St. Louis  
Computer Science and Engineering Laboratory

### Purpose

This tutorial will step you through using the Digilent S3 boards in the CSE labs for your projects. It is assumed that you know what VHDL is and how to use the Xilinx tools at some basic level. The complete flow from design entry through simulation and programming file creation will be discussed.

### Getting Started

The first thing you need to do to begin using the hardware (known from now on as the S3 board) is to create a project in the Xilinx tools with appropriate settings so that you will generate proper files for the particular device on the S3 board. Figure 1 shows what the proper settings should be.

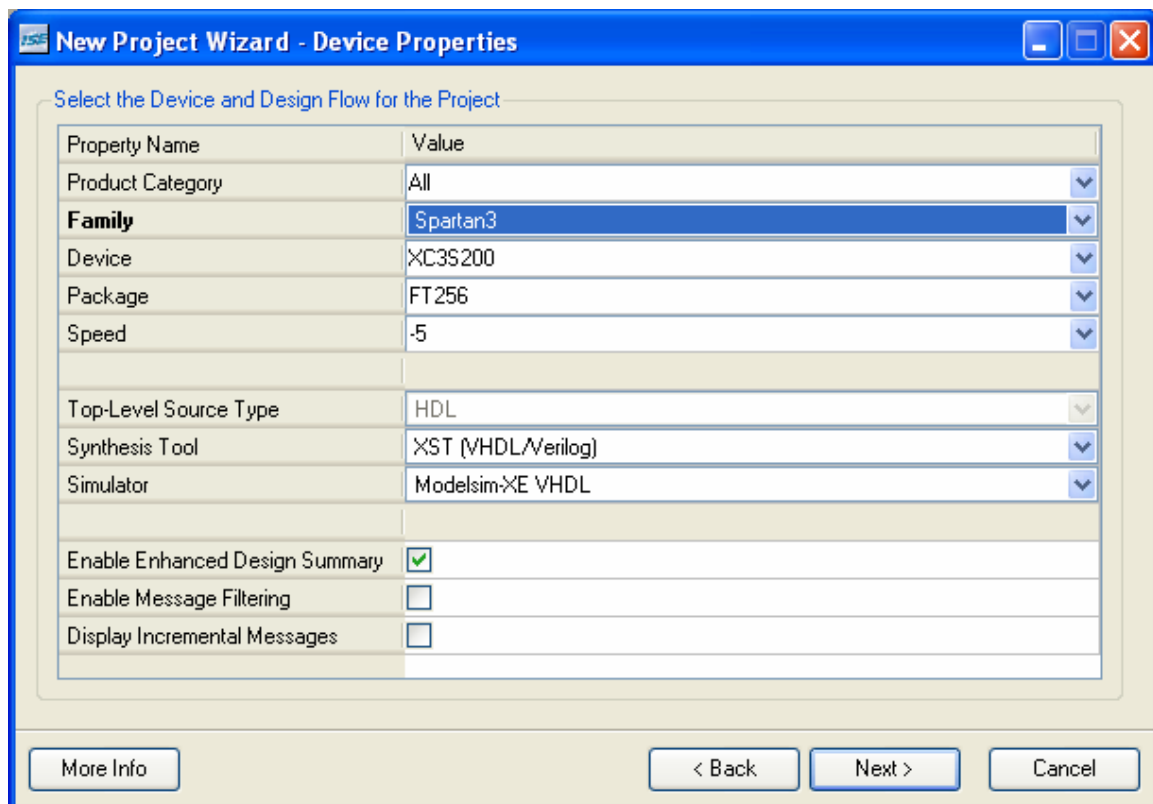


Figure 1: Project properties for the S3 board.

Once you have a proper project created, you can start entering VHDL modules for your design. One of the most important aspects with working with real hardware is to get the top-level ports in your entity correct. Our board has LEDs, displays, switches, buttons and other connectors that are physically connected to the FPGA in a particular way. We must be sure to assign our ports to match what's on the board. This is done in a user constraints file (UCF) that will be supplied to you. At this point, however, what we need to know are the names of the signals that are in the UCF so we can put them in our port list for our entity. For example, here are some signals and what they are connected to:

PORT	DESCRIPTION
an(3 downto 0)	Anodes for the 7-segment displays (enables)
btn(3 downto 0)	Push buttons
led(7 downto 0)	LEDs
srg(7 downto 0)	Cathodes for the 7-segment displays
swt(7 downto 0)	Toggle switches

Table 1: Some port names for the S3 hardware

So for our sample project, we'll invert the states of the toggle switches and display the results on the eight LEDs. That is, when the switch is up (on) the corresponding LED will be off and vice versa. The code, below, does this.

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ledtest is
    Port ( swt : in std_logic_vector(7 downto 0);
          led : out std_logic_vector(7 downto 0);
          an  : out std_logic_vector(3 downto 0));
end ledtest;

architecture Behavioral of ledtest is

begin
    led <= not(swt);      -- invert the switch state
    an <= b"1111";      -- turn off the 7-segment LEDs
end Behavioral;

```

Use *Project*→*New Source* to create a new VHDL module and insert this code. Once you do this, you can attach the UCF that will tell the tools which pins on the FPGA to wire the signals to so that they connect to the physical hardware.

Use *Project*→*Add Source* to add the [S3.ucf](#) file to your project. It will automatically attach it to your VHDL source file for you. You can download this file from the link or from the course [homepage](#).

## Simulation

At this point in the design process, you should have some VHDL code and the UCF ready to compile and test. Even though we will be putting this into real hardware, you should always simulate your designs before doing so. There are many reasons to do this. Two major ones are

1. It takes considerable time to run your design through all the tools necessary to program the physical device. It is much more efficient to simulate some basic functionality before going through all that effort; and
2. Once the device is programmed, it is very difficult to debug the design. If the outputs are not what you expect, finding the problem could be difficult if not impossible. Simulation lets you look *inside* the device and see your internal signals to help determine where the problems are.

You can create a test bench waveform from the Project→New Source menu and it will assist you in setting up your simulation. For this design the simulation is almost trivial, but we will do it so we are ready for timing simulation, later. Simply set up a simulation to see what happens for these two cases: `swt=x"aa"` and `swt=x"55"`. Here is what the functional simulation looks like.

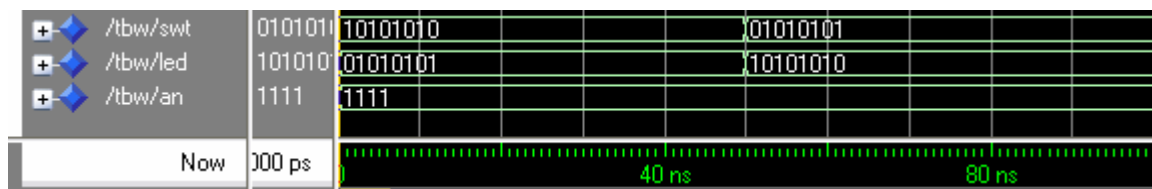


Figure 2: Simulation of ledtest program

Note this is what you expect, but we will use it later for some timing analysis. Once you simulate your design and feel it is function correctly, then you can move on to generating the data needed to actually program the device with your design.

## Synthesis and Programming Flow

The flow for creating the files necessary for implementation is, in reality, quite complicated. We will only focus on a couple of steps, however, as the details are not going to be important for small designs like we are implementing.

The first step is synthesis. This is the process of implementing your VHDL into logic gates. Once that is done, the next few steps are device and device vendor specific steps of translating the gates into physical implementations of the gate functionality. For example, your VHDL might synthesize a 2-input AND gate, but the technology only has 2-input NOR gates. These steps will implement the AND gate as two NAND gates, for example. Once this technology mapping is done the design can be placed into your target device and routed. Finally, a *bitfile* is created that will actually program the device with your design.

While most of the details along the way can be ignored, there is one thing we need to set up so that this process will succeed. By default, the tools will issue errors if you have things in the UCF that are not in your VHDL source file. This is not good for us since all of the signals in the UCF are not used in our current test design. Hence, we need to turn off this “feature.” You do this by right-clicking on the *Implement Design* section in the process window in the Xilinx tools. Then, in the *Translate Properties* tab, find “Allow Unmatched LOC Constraints” and check it. It should look like this when you are done:

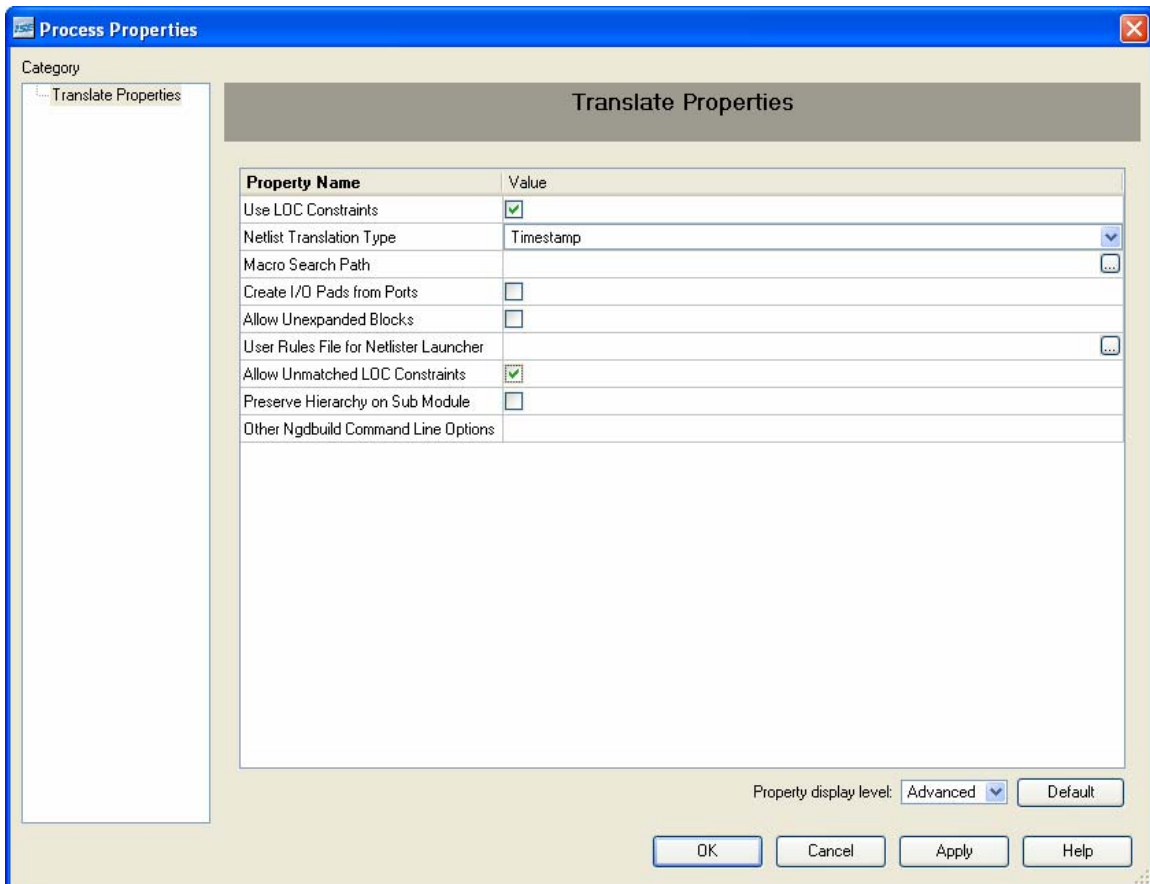


Figure 3: Setting LOC Constraint property in Process Properties dialog box.

With that done, you can go through the process of synthesis, mapping, place and route and programming file creation. You can do each step one-by-one or you can simply double-click on “Generate Programming File” as seen in Figure 3 and the entire process will run, assuming no errors. If you do that, now, you should see green checks by each step indicating success. If you see red Xs and/or errors, you need to fix them before moving on.

## Generate PROM File

Once the programming file is generated, you need to create one more file that is used to program the PROM that is used on the S3 board. You do this by double-clicking on the “Generate PROM, ACE, or JTAG File” line as seen in Figure 3. Once in the tool, follow these steps to prepare the file:

1. Select “Prepare a PROM File” from the dialog that asks what you want to create and click Next.
2. Change the PROM File Name in the next dialog box if you wish, but note the name for use in the next step. I chose ledtest and is the name that will be assumed for the rest of these instructions. Do not change anything else in this dialog and click Next.
3. On the next dialog box, select “Auto Select PROM” and click Next.
4. You will need to confirm the settings. Do so and click Next, again, then click “OK” to the small dialog box that pops up.
5. Now you should see the “Add Device” dialog. Select your file, e.g., “ledtest.bit,” to add the bit file that you generated, earlier and click “Open.”
6. When prompted if you want to add another design file, click “No” and then “OK” to the small dialog box that pops up.
7. Double click on the “Generate File” selection in the “iMPACT Processes” sub-window to generate the PROM file.
8. Assuming this worked, you can close the window that shows you that you were successful. **DO NOT SAVE THE SETTINGS!**

## Programming the Hardware

Now you are ready to program the PROM on the S3 board. All you really need is the MCS file you created, above. So, if you did all the previous work on your own computer or outside the lab, simply bring the MCS file with you to the lab or ftp it over.

The S3 board should be plugged into the programming cable and the green LED should be on. If the LED is not on, the board cannot be programmed and you should use another machine or seek assistance.

Now you are ready to program the board. If you are running the Xilinx tools on the PC attached to the cable, you can double click on “Configure Device (iMPACT)” as shown in Figure 4.

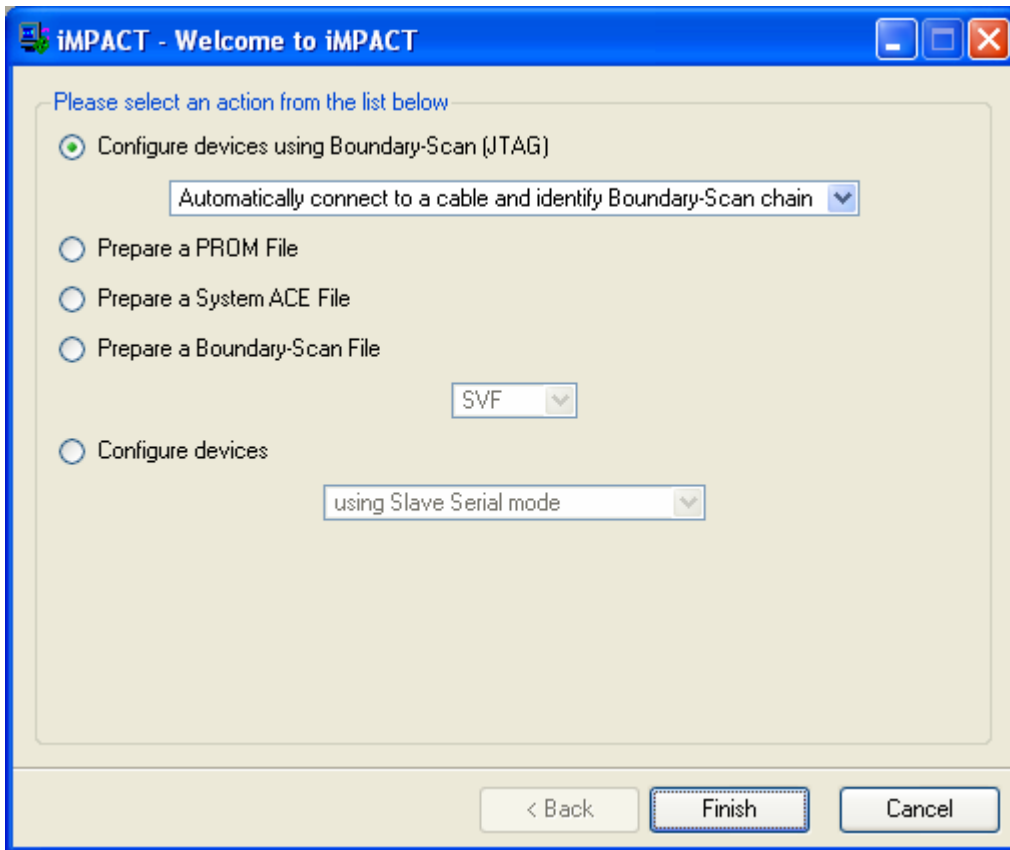


Figure 4. iMPACT is used to configure your FPGA via the JTAG port.

If you brought the MCS file with you, however, you can run iMPACT standalone by going to Start→All Programs→Xilinx→Accessories→ iMPACT.

1. Click “Finish” in the window shown in Figure 4.
2. You will be prompted for a file for the FPGA device, first. Click “Cancel.”
3. You will next be prompted for a file for the SPROM. Select the MCS file you created and click “Open.”
4. Right-click on the PROM device with your filename listed below it and choose “Program” from the menu.
5. OK the dialog box to program and verify your bitfile. This will take about 20 seconds.
6. You should see “Programming Succeeded.” If it failed, simply go back to step 4 and try, again. If it still fails after two or three tries, seek assistance.

## Running your Design

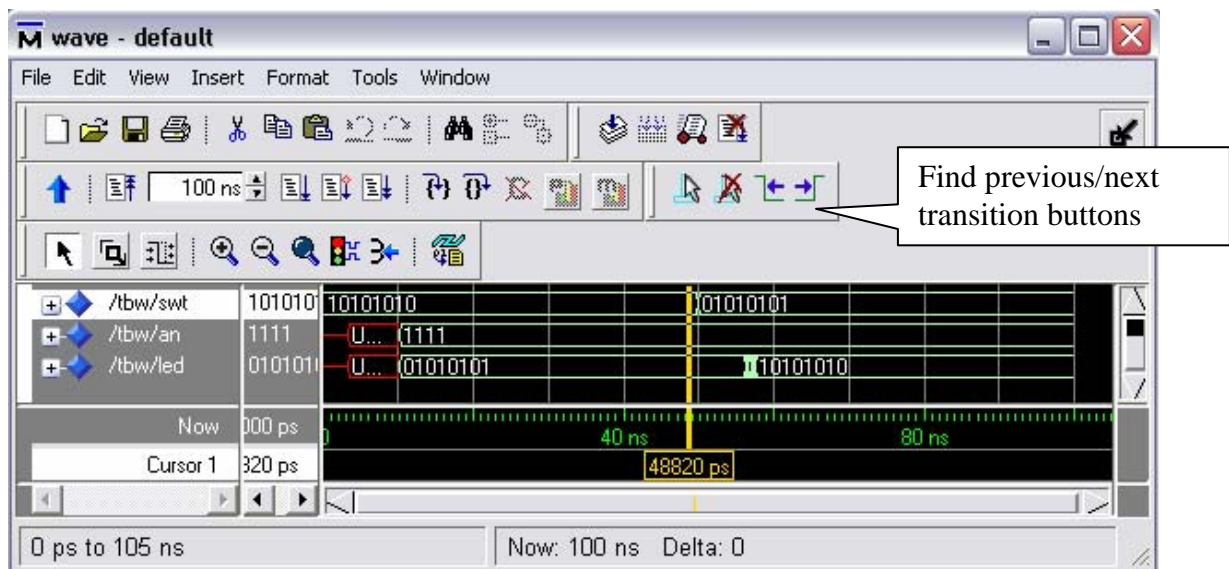
Once you program the PROM, you need to reset the FPGA so that it loads your program. Press the “PROG” button on the S3 board. This is located near the power cable and has a green LED next to it. Once you do this, your design should be loaded and you can test it to see that it works as expected.

## Timing Simulation

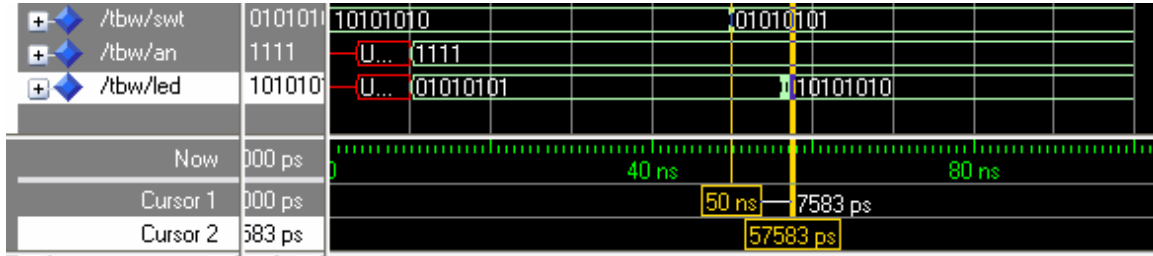
Now that you have a complete design running, you might notice some issues, or just be curious as to what the timing looks like. You can run the simulator with accurate timing based on the logic used in the device and the wire delays due to routing. To do this, go back to your Xilinx project and select your test bench waveform, again. Instead of selecting to simulate the behavioral model, double-click on “Simulate Post-Place & Route VHDL Model.”

The tools will create the necessary files and then start the simulator for you. Now when you look at the results you will see that the outputs do not change with the inputs. To see what these delays are you can use cursors to help you. Here is an example using the test bench for this design.

1. Click near the edge where the switch input changes from AA to 55. The cursor should move there. To get the cursor on the edge, click on the “swt” signal name to select it in the wave window.
2. Then click on *find next transition* or *find previous transition* buttons to move in the direction you wish.



3. Now add a second cursor via the Insert→Cursor menu.
4. Drag the second cursor over to where the output (led) is stable after the input change. Use the find previous transition button to get it on the transition after the output is stable. Note there is now a delta time between the cursors indicating a 5.7583 ns delay.



You can add more cursors or move these around as you see fit to find delays and see how your circuit performs.