Working with the GODIL

Author: Ruud Baltissen
Credits: Michael Randelzhofer, Ed Spittles

Date: August 2010

What is it?
This document describes a way to get familiar with the Xilinx FPGAs on OHO's Godil, especially the XC3S500E and the software to program it: ISE Webpack.

Why?
I was always interested in upgrading my Commodore computers in way or another. On http:/forum.6502.org I asked if it would be possible to design an FPGA replacement for the 6502 processor. BigEd pointed me to the GODIL.
But being a newbie to programming FPGAs and only having a tiny bit of experience working with VHDL this meant I had a lot to learn. I looked for a kind of "GODIL for dummies" but that was not to be found. So I decided to write one using my own experiences.

What do you need?
Beside the GODIL you need the ISE Webpack from Xilinx. It is free but you need to register. Xilinx can also provide you with various tutorials. Again you need to be registered to get them. The tutorial I used as base for this document is ***ise_tutorial_ug695.pdf*** which goes with version 12.1 of the software.

Let's go!
After starting the software we have to create a new project using the New Project Wizard. This is done by:

1. From Project Navigator, select **File > New Project**. The New Project Wizard appears.

2. In the Location field, browse to the directory where you want your projects. You have to know that Webpack will create a directory with the name of your project.

3. In the Name field, enter the name of your project.

4. Verify that HDL is selected as the Top-Level Source Type, and click Next. The New Project Wizard—Device Properties page appears.

5. Select the following values in the New Project Wizard—Device Properties page:
    - Product Category: All
    - Family: Spartan3E
    - Device: XC3S500E (or whatever is on the board)
    - Package: VQ100
    - Speed: -4
    - Synthesis Tool: XST (VHDL/Verilog)

- Simulator: ISim (VHDL/Verilog)
- Preferred Language: VHDL-93

Other properties can be left at their default values.

6. Click **Next**, then **Finish** to complete the project creation.

Next step is to add the source files to the project. I used the VHDL files provided by OHO-Elektronik as base. To make sure that I indeed reprogrammed the GODIL at the end, I changed the original demo program a bit: I replaced the shown numbers 1, 2 and 3 by 9, 7 and 5. If you don't have a LED display, simply switch the two LEDs.

You now need to add the three source files to the project as follows:

1. Select **Project > Add Source**.

2. Select the following files from the project directory, and click **Open**.
   - ***Demo_Dy1.vhd***
   - ***Oho_Dy1.vhd***
   - ***OhoPack.vhd***

3. In the Adding Source Files dialog box, verify that the files are associated with **All**, that the associated library is **work**, and click **OK**.

4. In the Processes pane, open **Synthesize – XST** by clicking the boxed **+** in front of it.

5. Now double-click **Check Syntax**. Verify that the syntax check passes successfully: a green circle with a white **V** in it should appear as sign that things went well.

We now enter the synthesis options:

1. In the Hierarchy pane of the Project Navigator Design panel, select your VHD file, in my case I kept the original name: **demo_dy1.vhd**

2. In the Processes pane, right-click **Synthesize - XST**, and select **Process Properties**.

3. Under the **Synthesis Options** tab, set the **Netlist Hierarchy** property to a value of **Rebuilt**.

4. Click **OK**.

5 In the Processes pane, double-click **Synthesize - XST**.

At this moment I don't know if this part is needed but the PDF says at the end that a NGC file has been created and I don't know if that one is needed in future. But it only takes a few seconds so no problem IMHO.

1. Double-click **View RTL Schematic.**

2. If the Set RTL/Technology Viewer Startup Mode dialog appears, select **Start with the Explorer Wizard**.

3. In the Create RTL Schematic start page, select **demo_dy1** from the Available Elements list, and then click the **Add** button to move the selected items to the Selected Elements list.

4. Click **Create Schematic** (bottom-right).

Next step is to add **Demo_Dy1.ucf** to the project in the same way you added the three VHDL files previously.

Top-left you see **View: O Implementation O Simulation**. Make sure **Implementation** is selected.

Next we have to do some more settings. I went to these steps several times and in all cases the settings were already set as described in the PDF. But we go through it just to be sure.

1. Right-click the **Implement Design** process, and select **Process Properties**.

2. Ensure that you have set the **Property display level** (bottom-middle) to **Advanced**.

3. Click the **Place & Route Properties** category.

4. Change the **Place & Route Effort Level (Overall)** to **High**.

5. Click **OK**.

The next part is **Creating Timing Constraints**. Here you have to expand **User Constraints** and to double-click **Create Timing Constraints**. Here I was asked if an UCF had to be inserted and I clicked **Yes**. Then I was told that I had to do something about the “m49” clock signal. Double-clicking the one at top-middle and just clicking **OK** in the screen that then popped up, did the trick (but I don’t know why yet).
Now close the pane and answer **Yes** to the question if the altered **UCF** file has to be changed.

A tip: rename the original **UCF** file and use this new one as base for future use.

According Xilinx’s PDF, now you have to expand **User Constraints**, and double-click **I/O Pin Planning (PlanAhead) - Post-Synthesis**. This starts up the PlanAhead software. This can take a few moments. Next I closed both screens.

Next step is to double-click **Implement designs**.

Next step is to create the Configuration data.

1. In the Processes pane, right-click **Generate Programming File**, and select **Process Properties**.

2. In the dialog box, click the **Startup Options** category.

3. Change the **FPGA Start-Up Clock** property from **CCLK** to **JTAG Clock**.

4. Click **OK**.

5. Double-click **Generate Programming File**.

Next step is to generate the PROM file.

1. Expand **Configure Target Device**, and double-click **Generate Target PROM/ACE File**. A screen appears, just click **OK**.

2. In iMPACT, double-click on **Create PROM File (PROM File Formatter)** in the iMPACT Flows window.

3. In the PROM File Formatter window, select **SPI Flash/Configure Single FPGA** in the **Select Storage Target** section.

4. Click the green arrow to activate the next section.

5. In the Add Storage Device(s) section, select **32M** for **Storage Device (bits)**.

6. Click **Add Storage Devic**e.

7. Click the green arrow to activate the next section.

8. In the **Enter Data** section, enter an **Output File Name**: demo_dy1.

9. Verify that the **Checksum Fill Value** is set to **FF** and the **File Format** is **MCS**.

10. Check the **Output file location**. If you have started a new project, it still points to the directory of the previous project.

11. Click **OK** to close the PROM File Formatter.

11. In the **Add Device** dialog box, click **OK**.

12. Select the **demo_dy1.bit** file.

13. Click **No** when you are asked if you would like to add another design file to **Revision: 0**.

14. Click **OK** to complete the process.

13. Select the Xilinx device graphic in the workspace area, then in the **iMPACT Processes** view, double-click **Generate File**. The **Generate Succeeded** message should appear.

14. Close iMPACT by selecting **File > Exit**.

15. If prompted to save the file, click **Yes**.

16. Enter a file name: **demo_dy1**.

Next step: using **iMPACT**. This involves connecting the GODIL to your PC. I used an USB-JTAG cable. Make sure you connect the flatcable in the right way because it took me valuable time to find out why things didn't work as they should! Simply have a good look at the picture in chapter 3.2.
FYI: I placed the GODIL on two headers on a left over piece of experiment board and powered it up by soldering a surplus USB cable to the board and using my PC as power supply.
I started iMPACT up as said in the PDF, opened project **demo_dy1.ipf** and clicked the gray area under "iMPACT Flows" at the top left. This pops up the little menu you see before you opened the project. The window mentioned in the PDF does not show up (as the PDF mentions as a possibility) but the menu doesn't show up either; I just found out by pure luck. Double-click Boundary Scan and right-click the big field (as it says itself).

1. Choose **Cable Auto Select** to let iMPACT find out what cable you have (worked fine for me).

2. Now select **Initialize Chain**. In the graphic part a Xilinx IC will appear.

3. And a screen will appear, asking if you want to continue and assign configuration file(s). Answer **Yes**.

4, Choose **demo_dy1.bit**.

5. A screen will appear saying that this device supports attached Flash PROMs. Do you want to attach an SPI or BPI PROM to this device? Answer **Yes**.

6. Choose **demo_dy1.mcs**.

7. Choose **SPI PROM** and **M25P32**. Choose **OK**.

8. Just click **OK** in the next screen.

9. Make sure the FPGA is selected.

10. Now double-click **Program** at middle-left.

The programming will only take a moment and the GODIL now behaves according the new program. But this program is not stored permanently in the EEPROM. Power the GODIL off and on again and you will see that the old program is back again.
To store it permanently, make sure that you select the EEPROM at step 9. In my case the programming took 23 seconds.

I found out about the above behaviour by accident and AFAIK it isn't mentioned in the PDF. Programming only the FPGA has two advantages:
1) You save about 21 seconds waiting time.

2) Although you can reprogram the EEPROM over 10000 times, it is still a limit. In this way you can enjoy your GODIL a very long time.

I hope this document is the start of a long time of fun with the GODIL. If you have comment about this document, don't hesitate to email me: Ruud@Baltissen.org.

I have the intention to expand this document with experiences of users and links to working projects where other users can benefit from. So please share your experiences and, if possible of course, your work. Many thanks in advance!

Kind regards, Ruud Baltissen