A Beginner's Guide to MAX+plus II

Author: Jacinda Clemenzi - Last Updated: December 20, 2001
Revised: D. E. Troxel - April 8, 2002
Revised: D. E. Troxel - May 1, 2002

The purpose of this guide is to provide a tutorial that will help you to become familiar with how to use the software provided by Altera to edit VHDL, simulate designs, and program the FLEX 10K FPGA devices. It will be assumed that you have a basic understanding of VHDL.

Altera provides a software package for programming their devices. This package is called MAX+plus II and can be accessed from the 6.111 locker from any Sun OS workstation. MAX+Plus II will not run on SGIs or Linux machines. To run MAX+plus II, type:

athena% setup 6.111
athena% max2win &

MAX+Plus II is a fairly complete software package. It provides an editor, compiler, programmer, waveform generator, and simulator. Each of these functions can be chosen from the MAX+plus II menu option. This guide will give a walk-through of the different steps necessary to program a device, but makes no effort to be exhaustive. At the end is a step-by-step tutorial with an example project.

Starting a Project

A project consists of all the files necessary to program one device. MAX+plus II can only work with one project at a time. When you first open MAX+plus II you will have to create a project. The name of the project must be the name of the top level VHDL file. The recommended method for starting a new project is to open an existing top level VHDL file or create a new one through the File menu option:

File > New...

or

File > Open...

Then set the name of the project to the current file by doing the following:

File > Project > Set Project Name to Current File

It is easy to change the name of the project and top level file at any time in the future, so do not worry about it too much.

All of the components you call in your top level VHDL file need only to be in the same directory for the compiler to find them.

When you open MAX+plus II next time, it will automatically load the last project.

Working With VHDL Files

MAX+plus II provides an editor that uses color coding much like Emacs. When you open a VHDL file in MAX+plus II it is the default editor. You can access it directly through the MAX+plus II menu option. You may use any editor to modify files.

Configuring the Device

Once you have defined a project, you will need to configure the device.

Assign > Device...

Now select the FLEX 10K device from the pull-down menu. This will give you a number of selections in the Devices scroll menu. For this tutorial choose EPF10K10LC84-3. (If you later use the right hand chip, choose EPF10K70RC240-2.) From this same window click the **Device Options** button. From the pull-down menu next to the keyword *Configuration* under Device Options select EPC2LC20.

Compiling

Compile the code by selecting the compiler from the MAX+plus II menu option. Before you start the process you should select which version of VHDL syntax you would like the compiler to use. To be consistent with the VHDL code used for programming Cypress CPLDs, you should select the 1993 version:

Interfaces > VHDL Netlist Reader Settings

Now select the VHDL 1993 option.

To compile the file click Start. The process will take some time, especially fitting it to the device. Be patient. Any errors or warnings will appear in a window that pops up during the process.

Viewing the Report

Several files are generated by the fitter after your project is compiled. The two most useful files are the report file and pinout file. The report file will include the pinout close to the bottom.

The report file is *project_name*.rpt
The pinout file is *project_name*.pin

Assigning Pinout

Now that you have compiled the project once and let MAX+plus II assign the first pinout, you can set it so it remains constant in three ways.

First, you can directly edit the configuration file *project_name*.acf. If you edit the configuration file then you will have to set the project name again for the changes to take effect.

Second, you can use the graphical interface provided by MAX+plus II to modify this file:

Assign > Pin/Location/Chip

Enter the name of the node and the corresponding pin assignment. Be sure to select the **Add** button after each assignment.

The last and easiest option to assign the pinout is to let MAX+plus II back assign the pin numbers automatically to the configuration file for the project:

Assign > Back-Annotate Project...

Select the *Chip, Pin & Device* option.

Simulation

MAX+plus II provides a simulator and waveform generator for verifying that your code performs the way you expect it to. To simulate your design, you must first create a waveform file that includes input waveforms and specifies which output waveforms to watch. Do this by using the waveform editor:

MAX+plus II > Waveform Editor

Now select the input and output nodes from your project:

Node > Enter Nodes from SNF...

This will allow you to select multiple nodes at once by highlighting the names and clicking the arrow button to move them into the box on the right. Click **OK** to add these nodes to the Waveform Editor. Now you will want to define the input waveforms by selecting each waveform and using the toolbar along the left. Once you have done this be sure to save as *project_name*.scf.

Now you are ready to run the simulator.

MAX+plus II > Simulator

Deselect all options if any are selected and press **Start**. You can view the results in the Waveform Generator. If you have already closed this window, then click **Open SCF**.

Programming

Set up your lab kit next to the computer with the programmer in the Digital Lab. Be sure to plug the programming cable into the programming port next to the chip you wish to program. Each port programs the chip closest to it. Be sure to align pin 1 by matching the arrows when plugging in this cable. Power on your kit. Now open the programmer from MAX+plus II:

MAX+plus II > Programmer

If this is the first time that you are using the programmer, a hardware configuration window will appear. For reference, the hardware you are using is the BitBlaster. The RS-232 box should read */dev/term/a*, and the baud rate should match the jumper settings on the BitBlaster programming cable. They should be configured for 38400. When you have ensured that this data is correct, click **OK**. You can edit the hardware configuration at any time as follows:

Options > Hardware Setup...

To speed up the programming process, you should turn off the automatic blank-check and verify. Select the following menu option:

Options > Programming Options...

A pop-up window will appear. If any of the options are selected, deselect them. Click **OK**.

Now you have to specify the programming method. We are programming the FPGAs using a JTAG chain so make sure that Multi-Device JTAG Chain is selected from the JTAG menu.

JTAG > Multi-Device JTAG Chain

If this is the first time that you have used this option, the configuration window will pop up automatically. From the Device Name pull-down menu select EPC2. Click **Select Programming File...** and choose the file called *project_name*.pof. Click **Add**. Click **OK**.

Now you are ready to program the device. To do this, click **Program** from the main programming window. The process will take a few minutes.

You will have to turn the power to your kit off and on to get the new program to load into the FPGA.

A Simple Example

This section will walk you through all the steps above with a short tutorial to create a simple 8-bit counter. The 8-bit counter has two modes: user mode and clock mode. In the first mode, the counter will increment each time you push a trigger switch. In the second mode, the counter will increment on each rising edge of the clock signal. A reset signal will allow the counter to be reset to zero.

This section has been divided into two parts because there are two implementations in VHDL of the 8-bit counter. The first implementation should be straight forward and understandable from the notes in the VHDL file. After completing this part you may find that the counter output is unreadable when in clock mode because the clock is too fast. The second implementation divides the clock signal by a power of two to produce a slow_clk signal, which is counted instead of the clock signal.

Part I

The steps are listed here with a little explanation. Please see the previous sections for more information.

- The first step is to create a directory for your work and move the tutorial files to this directory. The files you will need are counter8.vhd and counter8.acf. The file counter8.vhd is the VHDL file, and counter8.acf is the configuration file that sets the device, configuration device, and pinout.

athena% setup 6.111
athena% mkdir 6.111.example
athena% cd 6.111.example

Copy the example files from the web or the 6.111 locker to your directory:
6.111% cp /mit/6.111/altera/beginner/counter8.* ./

Then change the mode so you can write to these files.
6.111% chmod 644 *

Then start MAX+plus II:
6.111% max2win &

- Now open counter8.vhd.
- Click the *File* menu and select Open. Now choose counter8.vhd.
- Now Set your project to the current file.
- Click the *File* menu. Click *Project* and select Set Project to Current File. You have now started your project.
- Now you have to configure the device. Select *Device...* from the *Assign* menu. A small window will pop up that will allow you to choose the chip you are programming. Select FLEX 10K from the device pull-down menu. Select EPF10K10LC84-3 from the Devices scroll menu. Now click on Device Options. From the pull-down menu next to *Configuration* select EPC2LC20. Click OK. Click OK again to get rid of the configuration window.
- Save the project at this time.
- You should now compile the project. Select the Compiler from the MAX+plus II menu. Press Start. This will take a few minutes. Be patient.
- You may optionally simulate the counter at this point.
- The pinout has already been assigned for you in the counter8.acf file that you copied from the 6.111 locker. You can view this file or counter8.pin to see this pinout so you can wire your kit.
- The first time you use a new (to you) FPGA board program both FPGAs (EPC2s) This is to ensure that all pins are tristated before you try to use them. The files to use are /mit/6.111/altera/beginner/blank10K*0.pof. Do not cd to this directory, rather, copy the fi;es to a local directory.
- You are now ready to program. Set up your kit. Plug in the programming cable and turn on the power. Make sure that the programming cable is not plugged in backwards. The red stripe should be on the left with the front of the kit towards you.
- Open the programmer by selecting the Programmer from the MAX+plus II menu. Two menus pop up.
- Under *Hardware Setup* choose *BitBlaster*.
- Now select *Multi-Device JTAG Chain* from the *JTAG* menu. If a pop-up window does not appear then select *Multi-Device JTAG Chain Setup...*JTAG menu.

li>In the pop-up window, select EPC2 from the *Device Name* pull-down menu and the click *Select Programming File...* Choose the file called counter8.pof. Click Add. Click OK.

- Click Program from the programmer window. Programming will take a few minutes. Be patient.
- Now you have to prepare your kit for testing. Wire pushbutton switches to the signals *reset* (AD2 on the Nubus) and *trigger* (AD3 on the Nubus). Wire a toggle switch to the

mode input (AD1 on the Nubus). Make a clock with a crystal oscillator and wire this to the *clk* input (AD0 on the Nubus). The counter will display on the second and third HEX LED display units (AD4 through AD11 on the Nubus).

- You will have to turn the power to your kit off and on to get the new program to load into the FPGA.

Part II

The second part of this example implements a slightly more complicated version of the 8-bit counter from Part I.

You should already have a directory for your work from Part I. The files you will need for Part II are slow8.vhd and slow8.acf. The file slow8.vhd is the VHDL file, and slow8.acf is the configuration file that sets the device, configuration device, and pinout.

Follow the same steps that you followed in Part I to program and test the device. The functionality on the 6.111 kit should be the same as the example in Part I except that the counter will visibly count when in clock mode.

Appendix: Differences between MAX+plus II and Warp

MAX+plus II is the software provided by Altera for writing VHDL, compiling and programming their FPGAs. Cypress provides a similar tool for their CPLDs called Warp. If you already have experience with Warp, you may be interested in the specific differences between the two programs.

VHDL Libraries

Unlike Warp, Altera uses the standard IEEE libraries for arithmetic operations with the std_logic_vector type. In Warp, the necessary library is work.std_arith. In MAX+plus II, the standard libraries are ieee.std_logic_arith, ieee.std_logic_unsigned, and ieee.std_logic_signed. You probably want the latter, especially if you want to subtract.

Working With Projects

In MAX+plus II projects are handled a bit differently. The project is defined and set to the same name as the top level file. Each VHDL file must have the same name as that of a component defined in them. Furthermore, all components referenced by a top level VHDL file must be in the same directory as the top level file.

Assigning Pinout

Unlike Warp, the pinout can not be specified in the VHDL file. The *.acf* file holds all the device configuration information including pin assignments, device, configuration device, and logic block assignments. This file can be edited directly, but Altera also provides a way of editing each of these options through the graphical user interface.