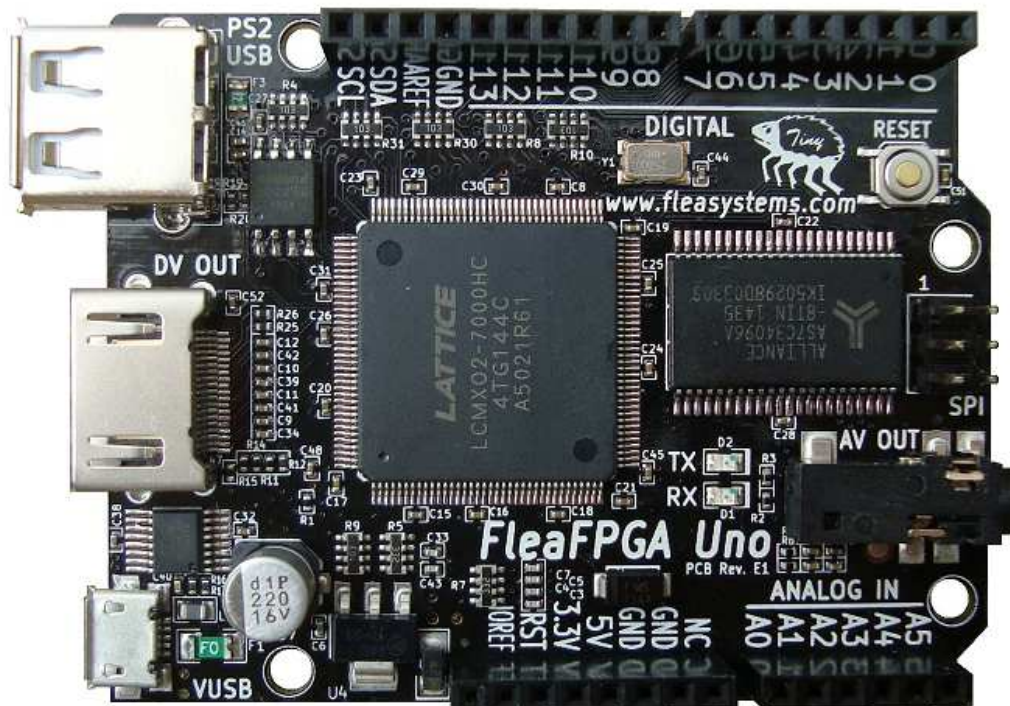


FleaFPGA *Uno*

Starter Board

Quick Startup Guide

Revision 0.27 (Preliminary)



www.fleasystems.com

Contents:

SECTION 1 – WARRANTY AND LEGAL INFORMATION	3
SECTION 2 – INTRODUCTION.....	4
SECTION 3 – FLEAFPGA UNO CONNECTIVITY BLOCK DIAGRAM	5
SECTION 4 – FLEAFPGA UNO HARDWARE OVERVIEW	7
SECTION 5 – SOFTWARE INSTALLATION.....	9
SECTION 6 – LOADING BIT-FILES INTO FLEAFPGA UNO	14
SECTION 7 – USING LATTICE DIAMOND	15
SECTION 8 – WHERE TO FROM HERE?.....	26
SECTION 9 – NOTES	28
APPENDIX A – FLEAFPGA UNO (REV.E1) SPECIFICATIONS	31
APPENDIX B – FLEAFPGA UNO BOARD SCHEMATICS	32

Section 1 – Warranty and legal information

Product Warranty and Liability:

FleaFPGA Uno Starter Board is warranted against defects in materials and workmanship for a period of 90 days from the date-of-purchase. At its discretion, Fleasystems will repair or replace a FleaFPGA Uno Starter Board where it has found defective due to quality control or manufacturing error. In the case of a valid warranty claim, Fleasystems will cover the repair cost of a defective FleaFPGA Uno Starter board and only pay for return shipping via standard mail. In all other instances the user shall pay all shipping and repair costs.

This warranty does not cover product damage or destruction due to improper use i.e. operating FleaFPGA Uno beyond stated electrical and thermal limits or through physical abuse. ***This includes instances where a shield board that only supports 5Volt signalling is installed into FleaFPGA Uno, which only supports 3.3V logic signals and can NOT TOLERATE 5V logic levels (resulting in probable permanent damage to installed shield(s) and FleaFPGA Uno).***

The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies Supplier from all claims arising from the handling or use of the goods.

This warranty also does not cover instances where the user has JTAG-programmed a firmware application into their FleaFPGA Uno Starter Board that does not specifically include the following official project HDL source files “FleaFPGA_Uno_Top.vhd” and “FleaFPGA_Uno_Top.lpf” as downloaded from the www.fleasystems.com/FleaFPGA_Uno support page. It is the responsibility of the user to ensure their custom HDL applications include these two critical source files.

Through the purchase of FleaFPGA Uno Starter Board, you agree to indemnify and hold harmless Fleasystems from any consequential damages or claims arising from, or associated with, use of FleaFPGA Uno Starter Board.

If you do not agree to any of the abovementioned terms, please return your FleaFPGA Uno Starter Board within 90 days of purchase to obtain a full refund.

Legal Notice:

FleaFPGA Uno Starter board is intended to be treated as an electronic module for evaluation purposes only. For commercial products intended for sale that include FleaFPGA Uno, it is the responsibility of the user to ensure their final system meets overall regulatory and compliance requirements.

The FleaFPGA Uno is NOT to be used as a solution for applications deemed to be Safety-critical or life-support related. The user shall assume all responsibility for any consequences arising from such use.

While every effort is made to ensure the information presented is accurate, Fleasystems make no such guarantee. Fleasystems reserve the right to update this document without prior notice. FleaFPGA Uno PCB layout, logo and related documentation are the copyright of Valentin Angelovski. All trademarked product names listed herein belong to their respective trademark owners.

Copyright © 2015 Valentin Angelovski

Section 2 – Introduction

Dear Experimenter,

Thank you for purchasing the FleaFPGA Uno Starter board! As its name suggests, FleaFPGA Uno is built around the relatively new and exciting world of FPGA (Field-Programmable Gate Array) Technology.

Unlike traditional microprocessor or microcontroller-based boards, FleaFPGA Uno uses hardware-oriented languages like VHDL (or Verilog) to describe *custom digital hardware* that the user may potentially wish to create inside the FPGA. Through the power of FPGA, a wide variety (*) of digital circuit functions may be realized i.e. from Counters and simple controllers through to Microprocessors, Digital radio modulators, Video etc.

(*) = Within the physical limits of the on-board FPGA. FleaFPGA Uno is built around Lattice's MachXO2-7000 product. Refer to the Lattice MachXO2 Datasheet for further information.

The aims of this startup guide are three-fold:

- 1.) To provide a useful reference for the FleaFPGA Uno platform hardware. Appendices 'A' and 'B' cover the overall specifications and circuit schematics respectively.
- 2.) Provide the steps needed to load firmware files into FleaFPGA Uno via the on-board USB JTAG interface.
- 3.) Provide insight into how new firmware files are created using FPGA development tools (i.e. Lattice Diamond).

Please Note: This document is NOT a 'how-to' guide on VHDL programming, nor is it an introduction to electronics engineering! It is essential that the reader already has a good understanding of basic electronics (particularly of digital electronics and logic theory). Many tutorials exist (in print form and online) for VHDL, basic electrical/electronic and digital logic theory.

Should you have any questions or feedback about this guide, you may either register on our support forums (<http://www.fleasystems.com/forums>) or drop us an email from the contacts section on our site.

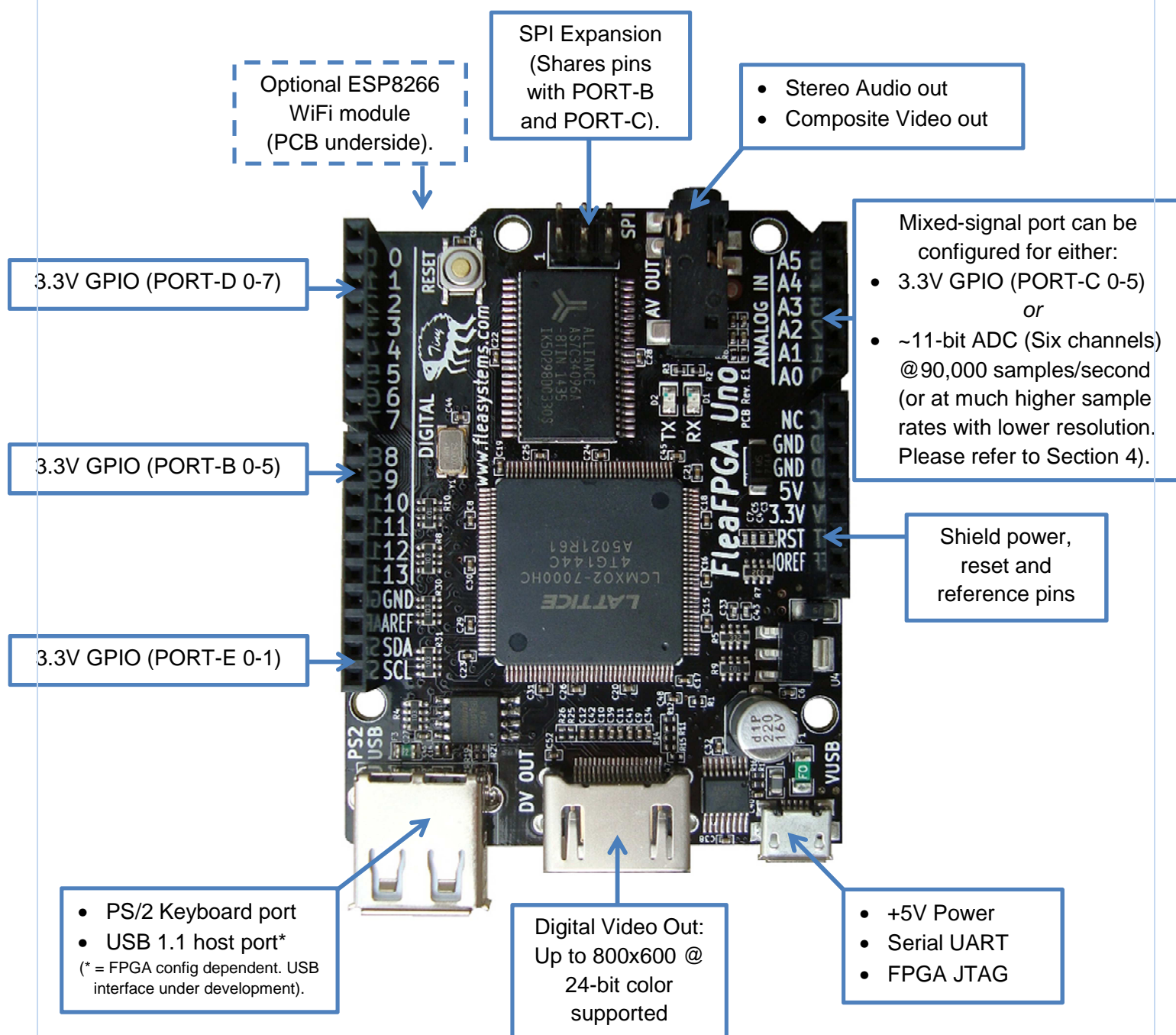
Thanks again and happy experimenting! ☺

Sincerely,

Valentin Angelovski
www.fleasystems.com

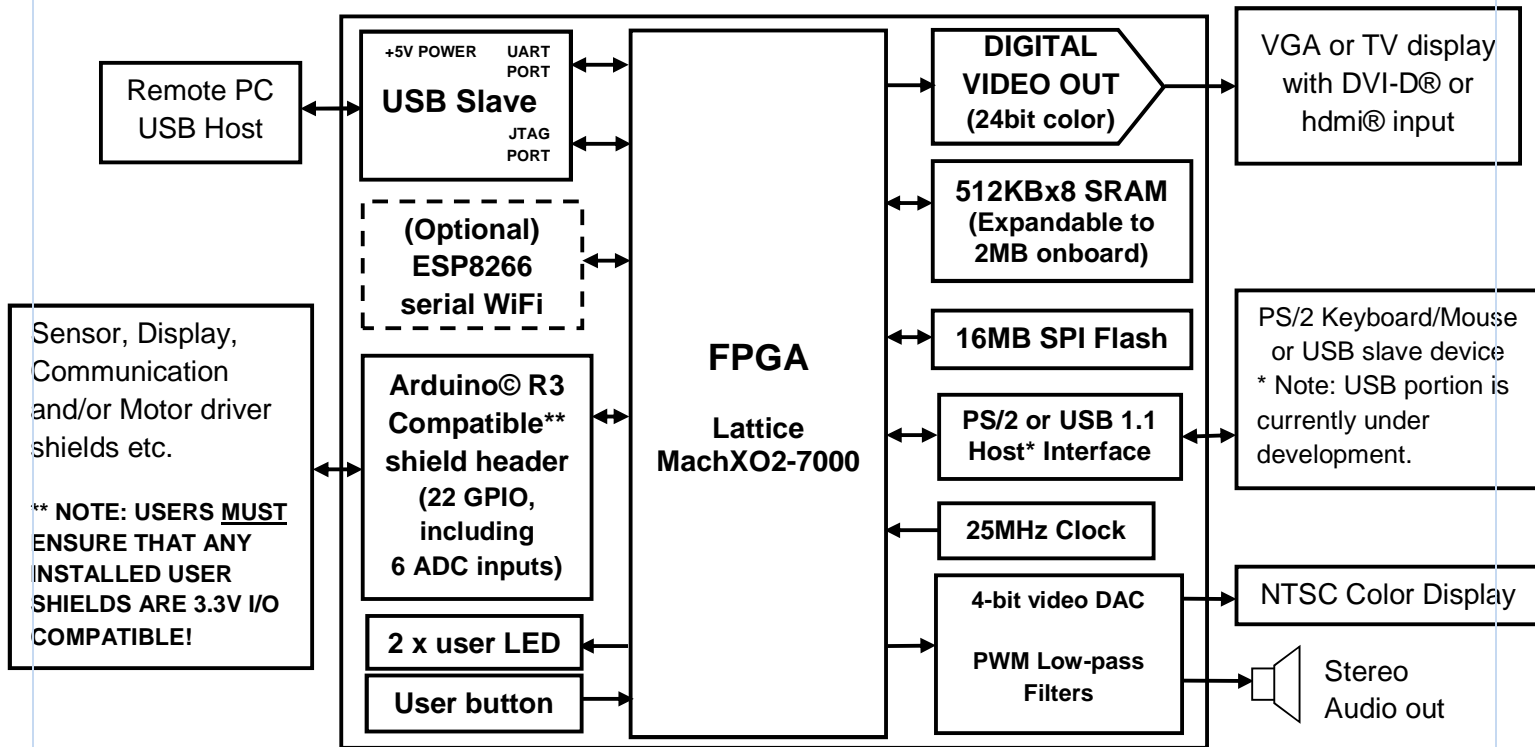
Section 3 – FleaFPGA Uno connectivity block diagram

SYSTEM CONNECTIVITY OPTIONS:






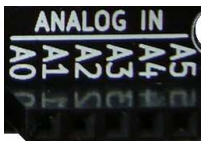
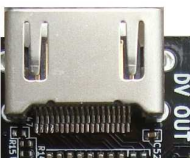

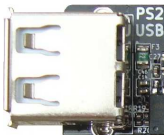
Section 3 – FleaFPGA Uno connectivity block diagram (Continued)

REVISION.E1 PCB SYSTEM BLOCK DIAGRAM:




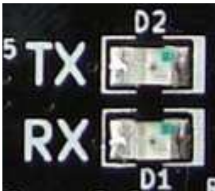
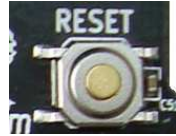



Section 4 – FleaFPGA Uno hardware overview

Following table describes the function of each of the FleaFPGA Uno peripherals in further detail. Please refer to the FleaFPGA Uno schematics in Appendix B for information on electrical connection details.

Peripheral:	Hardware details
  	<p>User Digital GPIO: Normally used for providing digital signal lines to Arduino® R3 compatible shields, Provides up to 22 programmable user digital bidirectional lines depending on how the FPGA is configured. In addition, all digital-only pins feature programmable pull-up resistors. Please read the following warning note and then proceed on to Appendix B for further details.</p> <p>WARNING: ALL GPIO on the FleaFPGA Uno is 3.3V capable only! – DO NOT INSTALL any Arduino-compatible shields that can only drive GPIO pins at 5V levels! Doing so may cause permanent damage to your FleaFPGA Uno and void your warranty. Normally, if an intended shield has the ability to sense the IOREF and AREF (ie. to detect if 3.3V I/O must be used instead of 5V) pins then chances are it can operate correctly with FleaFPGA Uno. If you have any doubts about shield compatibility, please contact us. Thankyou!</p>
	<p>Sigma-Delta ADC inputs: Normally configured for approximately 1980 counts full-scale (ie almost 11bits resolution) @ 90Ksamples/second for use in Arduino-oriented projects. AREF is to +3.3V. For all non-arduino applications, faster sample rates at lower data resolution are possible. This is due to the fact that nearly all of the FleaFPGA Uno's ADC circuits are integrated into the MachXO2 FPGA itself.</p> <p>Each analog input in the FleaFPGA Uno can be thought of as a 1-bit ADC designed for 200 Million samples/second operation. This output is then accumulated and filtered to provide the desired sample rate and resolution. More information on the Sigma-Delta ADC method can be found in Lattice Semiconductor's Application Note RD1066 as well as on David Kessner's technical blog site Also refer to Appendix.B of this document, as well as the default "Simple_ADC.vhd" module as found in either the F32c or Sweet32 SoC examples to see how it is implemented.</p> <p>WARNING: I/O pins operate at 3.3V levels only and are <u>not</u> 5V tolerant.</p>
	<p>Digital Video Out: Provides a 24bit-color, RGB digital video signal out, including horizontal and vertical timing information as per digital standards like DVI-D and/or HDMI. Supported resolutions up to and including 800x600.</p>
	<p>Stereo audio + NTSC video out: Left and Right audio outputs are connected to FPGA pins #97 and #96 respectively via a low-pass filter. Each audio channel is designed to be driven by a PWM (Pulse-Width-Modulated) waveform from within the FPGA. NTSC video out is implemented using a resistive 4bit DAC that provides 0->1.2V output range as required by the NTSC standard for color video generation. Up to 32 colors are possible. Refer to Appendix B further details.</p>
	<p>Combined PS/2 or (firmware pending) USB 1.1 Host: Provides user input from keyboard or mouse devices with a PS/2 interface. USB 1.1 Host firmware currently under development. Maximum current drawn from this port should not exceed 150mA. Data signals are connected to FPGA pins #120 and #119.</p>

Section 4 – FleaFPGA Uno hardware overview (continued)

Peripheral:	Hardware details
	<p>USB Slave port: Enables either of the following functions to be performed:</p> <ol style="list-style-type: none"> 1.) Programming of the on-board FPGA via JTAG 2.) Virtual COM port. Where TX_out and RX_in pins connected to FPGA pins #141 and #140 respectively. <p>In addition to the above, USB slave port also serves as the main power supply feed for FleaFPGA Uno. Current consumption by FleaFPGA Uno with all peripherals connected should not exceed 750mA continuous current.</p> <p>Attention! Due to the shared nature of the USB slave design, users must NOT have the COM port open while performing JTAG operations on FleaFPGA Uno. Please close any open terminal programs to FleaFPGA's slave COM port before connecting to FleaFPGA Uno using the JTAG loader utility (and vice versa ☺).</p>
	<p>512KByte 10 nanosecond SRAM: Organised as 512K x 8bit data width, based on either "ALLIANCE AS7C34096A" or "NETSOL S6R4008W1A" SRAM. Refer to the schematics on Appendix B for pin-out details, or the manufacturer's datasheet for detailed device info. FleaFPGA</p>
	<p>16MByte SPI ROM: Used for storing user programs and/or data. Based on Winbond W25Q128FV Flash ROM. Refer to the schematics on Appendix B for pin-out details, or the manufacturer's datasheet for detailed device info.</p>
	<p>TX / RX or USR#1-#2 LEDs: Normally function as USB UART communication activity indicators, but can be re-assigned for other uses as well. Connected to FPGA pins #1 and #2 respectively. Each LED can be turned on by driving the related FPGA I/O pin high. NOTE: When the ESP8266 WiFi option is installed. LED2 is hardwired as a 'module enable' signal by default (See Appendix B).</p>
	<p>RESET / USR#1 Button: Normally functions as a system reset button, but can be re-configured for other uses as well. Connected directly to FPGA pin #11. When the button is pressed, a logic-low signal is passed on to the FPGA.</p>
	<p>Optional ESP8266 Serial-to-WiFi adapter (on PCB underside): Module TX_out, RX_in, reset, boot_mode and module_enable signals connected to FPGA pins #109, #12, #142, #75 and #2 respectively. Please note in order to enable the WiFi module, pullup resistor tied to Digital#0 (PORTD_0) must be enabled and LED2 must be on. Users are also encouraged to refer to the ESP8266 datasheet, as well as the F32c-based serial console example sketch.</p>

Section 5 – Software Installation

This section covers the basic steps needed to setup your Windows-based PC for use with pre-built FPGA configuration files (i.e. files specifically compiled for FleaFPGA Uno use using the .VME file format) or for end-user FPGA development.

Software drivers Install:

Essentially, this includes all the software drivers and utilities needed to allow the user to upload FPGA files (i.e. VME bit-files which must be specific to FleaFPGA Uno) that may be downloaded from our fleasystems.com website to be programmed into their FleaFPGA Uno platform.

- 1.) Installation of the FTDI D2XX driver as needed by the PC to recognize FleaFPGA Uno's on-board JTAG interface (FT230x interface chip)
- 2.) Installation of the FleaFPGA JTAG utility, allowing suitable bit-files to be uploaded to FleaFPGA Uno

FPGA development tools Install:

Essentially, this includes all the elements of the basic installation, as well as the ability for users to create their own custom logic designs (i.e. user-custom .VME bit-file) on FleaFPGA Uno. This is accomplished by installing Lattice Diamond Development and configuring it for use with the FleaFPGA Uno platform.

Please Note: All examples contained in this guide are screenshots from a Windows7-based installation. Windows 8 installation processes should be almost identical however.

Before you begin:

You will need the following:

- Modern PC with windows7 (Ideally 64-bit version, but not essential) or newer installed and also web-enabled. Must have at least 4GBytes RAM, as well as 30GBytes of free Hard Disk space minimum and one free USB port.
- 'FleaFPGA Uno' Starter board
- USB Type A to Micro B Male adapter cable
- USB Hub is recommended but not essential. Allows for JTAG Flash ROM programming time to be reduced from 150 seconds to 20! Reason it is not essential however, is that JTAG programming of FPGA configuration SRAM is extremely quick (3 seconds) in either case.

Section 5 – Software installation (Continued)

Section 5.1: PC Installation of the FTDI D2XX driver

- i. Download the driver file that matches your Windows OS version from the following link (**Note:** If the link is broken, please navigate from www.ftdichip.com to find the relevant support page): <http://www.ftdichip.com/Drivers/D2XX.htm>
- ii. Once it has downloaded, unzip and run the executable installer. Follow the prompts as given by the installer application.

Section 5 – Software installation (Continued)

Section 5.2: Installation of Lattice Diamond FPGA Development software

Please Note: Users who only wish to load pre-built FleaFPGA Uno firmware files and not create their own custom digital logic designs may skip this process and proceed on to Section 5.5.

Additional Note: Following requires the user to register on the Lattice Semiconductor website in order to be able to download the Lattice Diamond tools, as well as obtain a related freeware license keyfile (LICENSE.DAT file) via email. It is recommended that you do this before installing Lattice Diamond.

Attention! It is strongly advised to download and install the 64-bit version of this software, due to the fact that all FleaFPGA Uno example projects were built in a 64-bit operating environment, hence all relevant file paths will have a “x64” string added to them.

- i. Once you've received an email from Lattice containing a freeware LICENSE.DAT file. You may now proceed to download the installer that matches your Windows OS version from the following link (If the link is broken, please navigate from www.latticesemi.com to find the relevant support page):

<http://www.latticesemi.com/Products/DesignSoftwareAndIP/FPGAandLDS/LatticeDiamond.aspx>

- ii. Run the Lattice Diamond installation wizard. Follow the installation steps as per the wizard recommended or default settings.
- iii. Once the installation process is complete, copy the LICENSE.DAT file you received from Lattice Semiconductor and save it in the following directory on your computer:
C:\lsc\diamond\3.3_x64\License

Your Lattice Diamond software is now installed and ready to use! All that remains now is installation of the FleaFPGA JTAG programming utility.

Section 5 – Software installation (Continued)

Section 5.3: Installation of the FleaFPGA JTAG programming utility

- i. In the shipping confirmation email of your FleaFPGA order, you will have received the following attachment “FleaFPGA-JTAG-Setup.7z”.
- ii. You will see several files contained within the 7z file, where one of these is named “FleaFPGA_Uno_test.vme”. This file is basically a pre-built test firmware image to be uploaded to FleaFPGA Uno, please save the file in a temporary directory i.e. c:\temp
- iii. Unzip and run **FleaFPGA-JTAG-Setup.exe** installation wizard. Follow the installation steps as per the wizard recommended or default settings until you arrive at the following window:

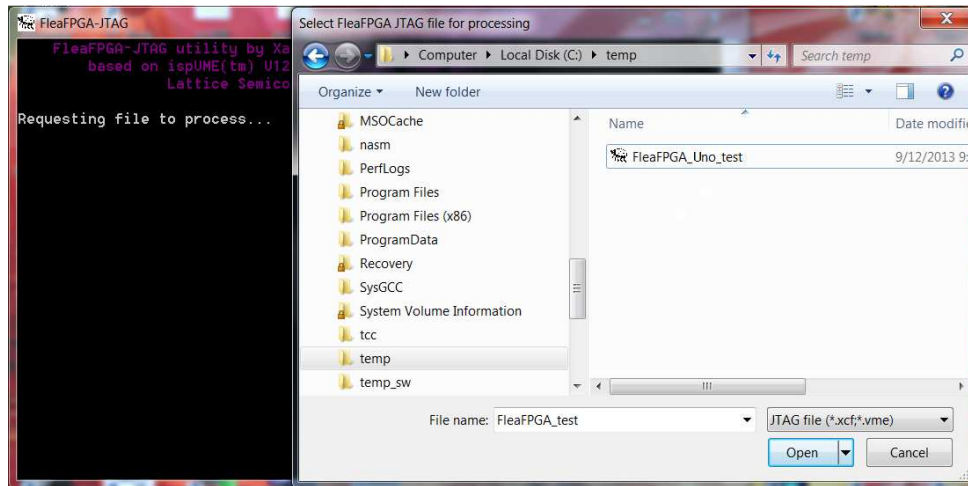


- iv. Make sure the ‘Launch FleaFPGA JTAG Utility’ option is ticked and then click finish. FleaFPGA utility will now run and request the path of “ddtcmd.exe”. This file can typically be found in the following path (assuming you’ve installed the 64-bit version of lattice Diamond here): C:\lsc\lattice\diamond\3.3_x64\bin\nt64

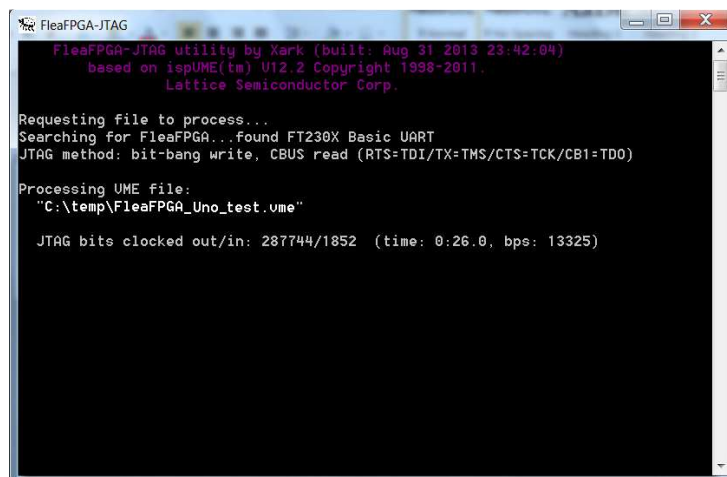
Please Note: You only need to select the relevant file and then click “Open” if you have Diamond installed and plan on using it to create your own custom FPGA-based logic designs. If you only wish to play with pre-built FPGA firmware examples and do NOT have Lattice Diamond installed, just click ‘Cancel’.

Section 5 – Software installation (Continued)

- v. FleaFPGA JTAG utility will then ask for a programming file (*.VME file) to be uploaded to the FleaFPGA board. Please locate the previously-saved “FleaFPGA_Uno_test.vme”, select it and then click “Open”.



- vi. FleaFPGA JTAG will now proceed to program the on-board FPGA with the ready-made “FleaFPGA_Uno_test.vme” bit-file. Program upload screen should appear as follows:



- vii. Once the FleaFPGA Uno flash program cycle (approximately 20 seconds with or over 2 minutes without a USB hub!) has completed successfully, you should now see ‘TX’ LED blink roughly once per second on your FleaFPGA Uno board! If a HDMI / DVI-D TV or monitor is plugged into the video output port at this point you should also see a nice color test pattern.

If you got this far, then all JTAG support and (if Section 5.2 was not skipped) suitable development software for FleaFPGA should be now installed and ready for use! ☺

Section 6 – Loading bit-files into FleaFPGA Uno

Currently, two methods are available to program FleaFPGA Uno from MS-Windows:

- 1.) Using the FleaFPGA JTAG utility to load suitable firmware files (.VME extension).
- 2.) Loading custom firmware files using Lattice Diamond (which then calls our FleaFPGA JTAG utility to program the FPGA).

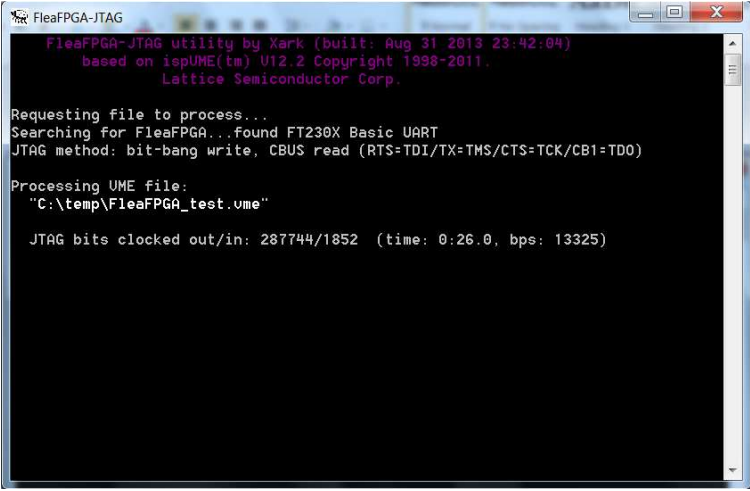
Only method #1 will be discussed in this section. For users interested in custom FPGA experiments, please refer to Section 7 and up. Pre-built .vme example files can be found on the FleaFPGA projects page of our site.

Please Note: Ensure that a suitable USB cable is plugged into your FleaFPGA Uno's slave USB port from the host PC. Also recommended (but not essential) is that connection be made via a USB hub, as it cuts the Flash ROM programming time from 2 minutes to 20 seconds or so.

- i. With FleaFPGA JTAG installed on your system, firmware image may be loaded by:
 - 1.) Double-clicking on a desired .vme file to load from Windows Explorer
 - 2.) Starting FleaFPGA JTAG utility and then selecting a desired .vme file to load

Either method will work, though method #1 is quicker. ☺

- ii. FleaFPGA JTAG will now proceed to program the on-board FPGA with your selected "prebuilt_xyz.vme" bit-file. Program upload status screen should appear as per the following:



```
FleaFPGA-JTAG
FleaFPGA-JTAG utility by Xark (built: Aug 31 2013 23:42:04)
based on ispVME(tm) V12.2 Copyright 1998-2011.
Lattice Semiconductor Corp.

Requesting file to process...
Searching for FleaFPGA...found FT230X Basic UART
JTAG method: bit-bang write, CBUS read (RTS=TDI/TX=TMS/CTS=TCK/CB1=TD0)

Processing VME file:
"C:\temp\FleaFPGA_test.vme"

JTAG bits clocked out/in: 287744/1852 (time: 0:26.0, bps: 13325)
```

- iii. Once the FPGA flash program cycle is complete (and a successful status message appear in the JTAG console window), you should see your FleaFPGA Uno board running the desired firmware. Nothing to it really ☺

Section 7 – Using Lattice Diamond

In this section, you will be introduced to Lattice semiconductor's Diamond FPGA Development Environment for use with FleaFPGA Uno. Diamond allows the user to configure FleaFPGA Uno with custom logic functions. All FPGA-based logic designs are created by using a Hardware Description Language (HDL), of which two (VHDL and Verilog) are most relevant and popular. Programs written using this language tell the FPGA exactly how to behave when power is applied to it. Examples that are provided for use on FleaFPGA Uno are written in VHDL.

Please Note: Following is NOT a substitute for Lattice's own Diamond User Guide. This guide only provides the reader with an insight into the steps involved in building a HDL project using Lattice software tools. Interested readers are also encouraged to read through the Lattice Diamond User Guide as suggested in Section 8.

By the end of this section, you will hopefully gain some idea around how to create a new VHDL project within Lattice Diamond and then synthesizing (i.e. compiling) the project source into a suitable binary file to be loaded into the FPGA. Programming the FPGA from within Lattice Diamond is also covered.

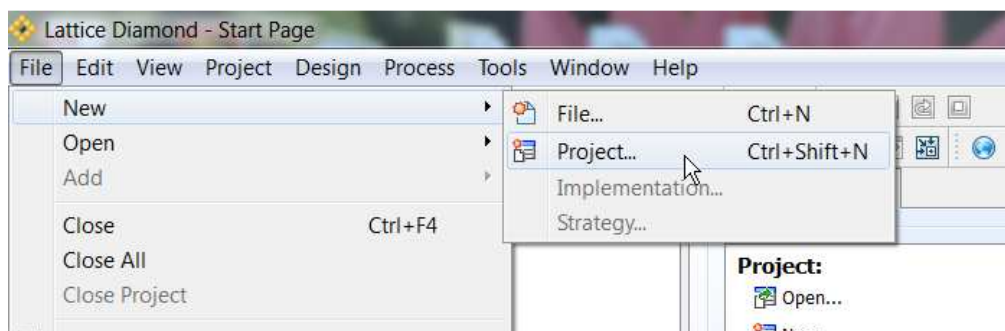
Section 7.1: Download example project source files

- i. Download the 'blinky' example project (Uno_blinky.zip) from the FleaFPGA Uno project page of www.fleasystems.com. Contained in the zip will be the following files:
 - FleaFPGA_Uno_top.vhd (Contains FleaFPGA Uno-specific interface code)
 - FleaFPGA_Uno_top.lpf (Contains FleaFPGA Uno-specific FPGA pinout info.)
 - Blinky.vhd (Contains user-level VHDL project source)

Attention! When creating your own projects, you must **always** include the first two files, as they are needed to help tell Lattice Diamond what physical hardware is connected to the FPGA on FleaFPGA Uno..

Section 7.2: Create a new HDL project

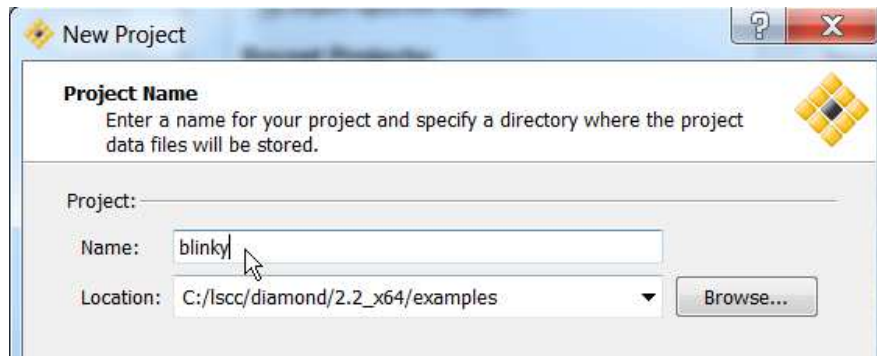
- i. Start Lattice Diamond and create a new project file as shown in the figure below:



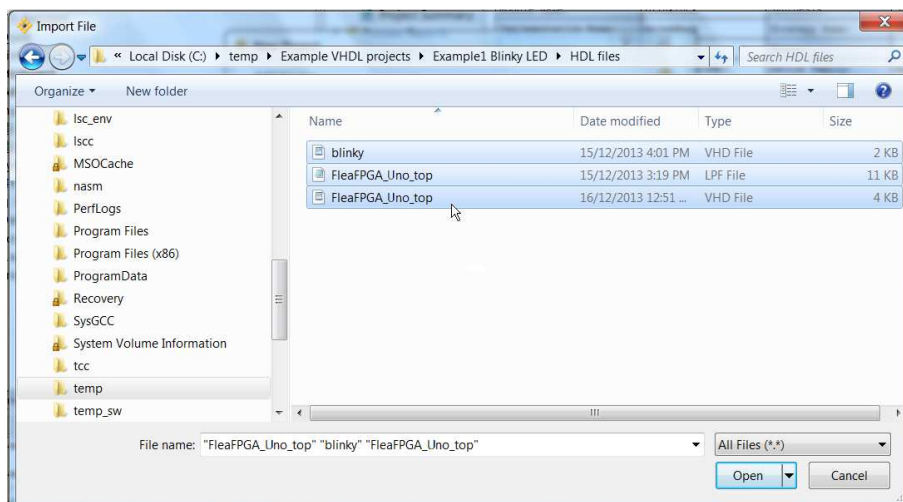
- ii. You should see a 'new project' child window appear. Click 'Next'.

Section 7 – Using Lattice Diamond (Continued)

- iii. Diamond will now ask for a project name. Please enter the name 'Blinky' in the text box as shown in the figure below and then click 'Next'.

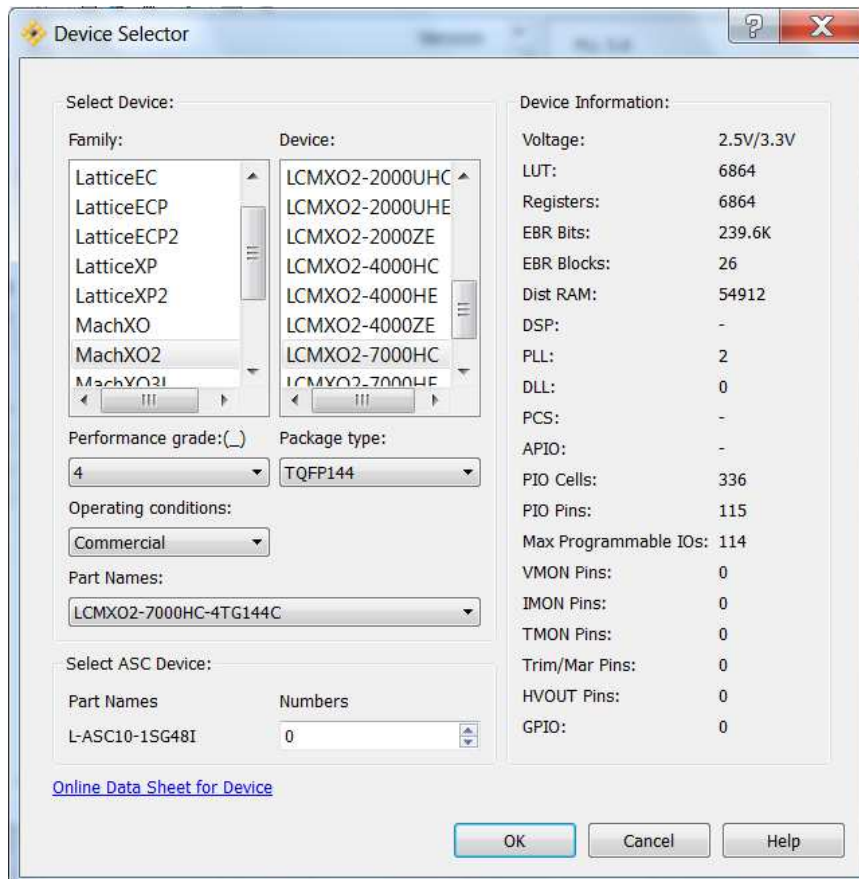


- iv. Next you will be asked to add any VHDL source files to the project. Please tick the “copy source to implementation directory” box. Next, Click “Add source” and then a file import window will appear. Select “All files” in the file extension selection and then select all the files contained within the example blinky.zip file. Once all the files are selected as shown below click ‘Open’.



Section 7 – Using Lattice Diamond (Continued)

- v. Next you will be asked to enter in the actual Lattice FPGA you will be using in this project. As FleaFPGA uses the Lattice MachXO2-7000HC-4 FPGA part in the TQFP-144 IC package style, please select the options as shown in the figure below and then click 'Next'.

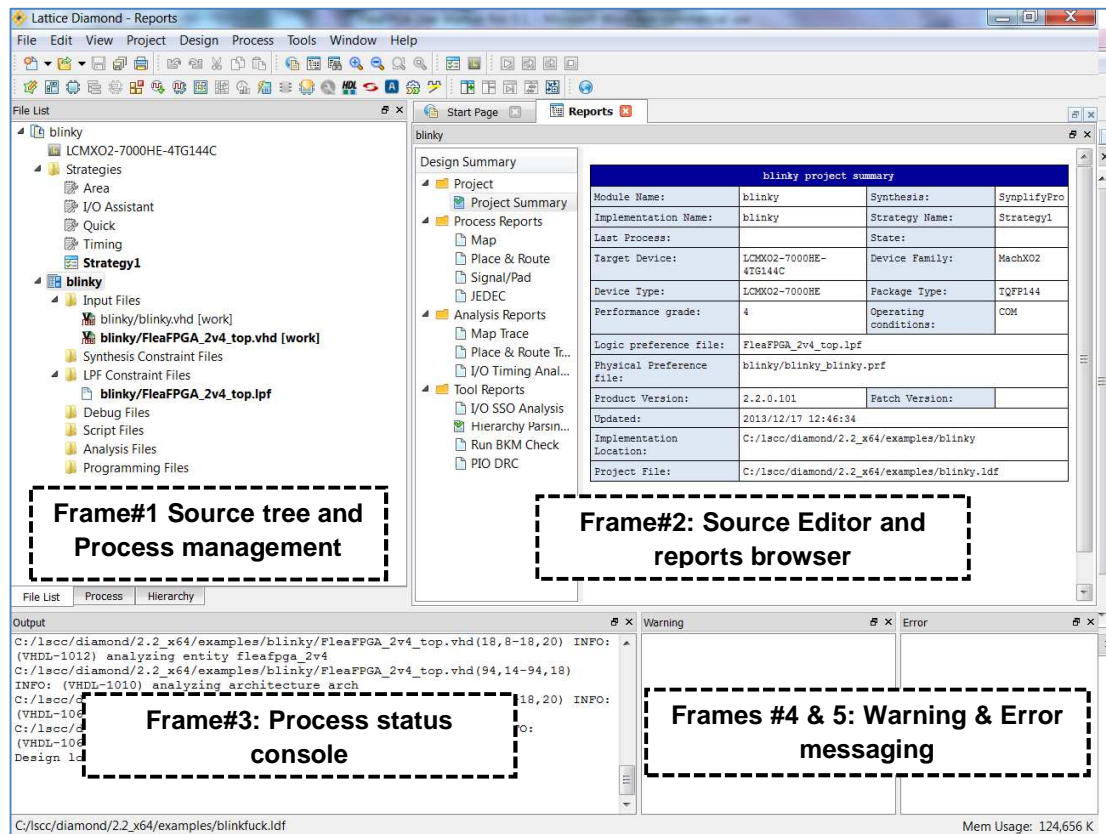


- vi. You will now be asked to select a synthesis tool. Select Synplify Pro and then click 'Next'.
- vii. Diamond will now provide you with a summary of the settings you've made for your new project. Make sure the summary matches this guide and then click 'Finish'.

Congratulations! You have just created your first Lattice Diamond project. 😊

Section 7 – Using Lattice Diamond (Continued)

If your new project was created successfully, you should now see the following source tree appear in your Lattice Diamond project:

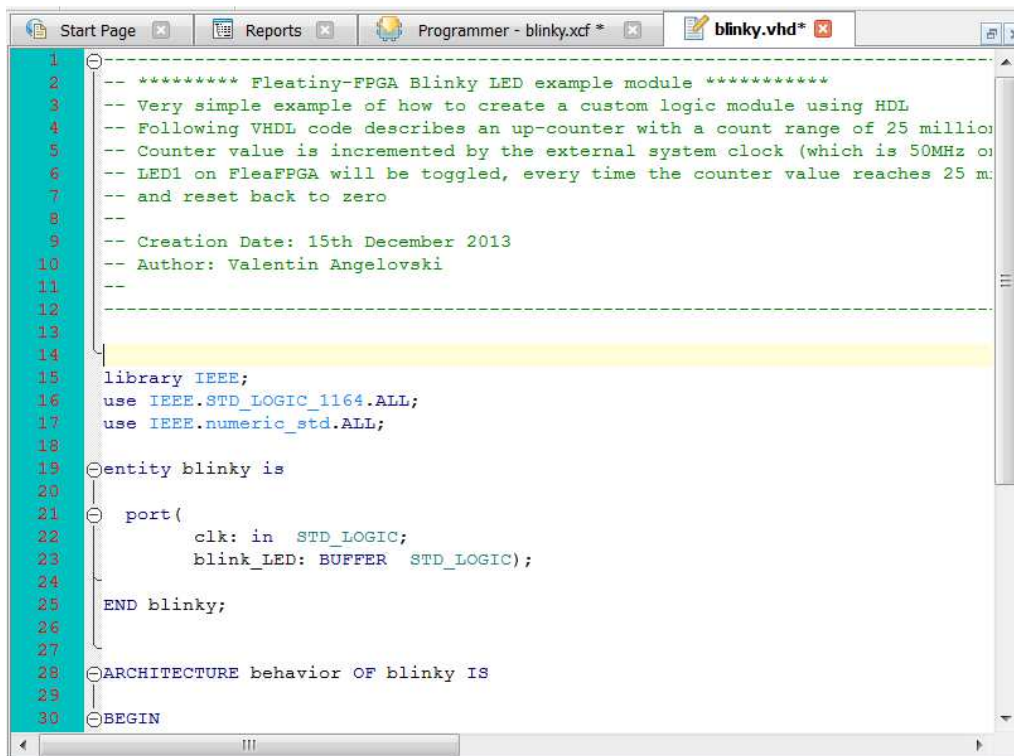


For the remainder of this section, we will be referring to the numbered frames shown above to help emphasize which frame within Lattice Diamond that we're covering.

Section 7 – Using Lattice Diamond (Continued)

Section 7.3: Editing HDL source files within Lattice Diamond.

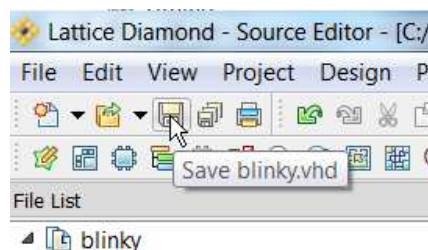
- i. From the source tree (Frame #1), double-click on a source file of interest i.e. “blinky.vhd”. You should see the source contained within that file appear in the source editor (Frame #2) as shown:



The screenshot shows the Lattice Diamond Source Editor with the 'blinky.vhd' file open. The editor has a menu bar (File, Edit, View, Project, Design, Pr) and a toolbar. The source code is displayed in a text area with line numbers 1 through 30. The code is a VHDL file for a Blinking LED example. It includes a library declaration for IEEE, an entity declaration for 'blinky', and an architecture declaration for 'behavior'. The code is as follows:

```
1  -- ***** Fleatiny-FPGA Blinky LED example module *****
2  -- Very simple example of how to create a custom logic module using HDL
3  -- Following VHDL code describes an up-counter with a count range of 25 million
4  -- Counter value is incremented by the external system clock (which is 50MHz on
5  -- LED1 on FleaFPGA will be toggled, every time the counter value reaches 25 m
6  -- and reset back to zero
7  --
8  --
9  -- Creation Date: 15th December 2013
10 -- Author: Valentin Angelovski
11 --
12 -----
13
14
15 library IEEE;
16 use IEEE.STD_LOGIC_1164.ALL;
17 use IEEE.numeric_std.ALL;
18
19 entity blinky is
20
21     port(
22         clk: in  STD_LOGIC;
23         blink_LED: BUFFER STD_LOGIC);
24
25     END blinky;
26
27
28 ARCHITECTURE behavior OF blinky IS
29
30 BEGIN
```

- ii. Once you've modified the source file in the source editor frame. Save your work as shown in the following figure.



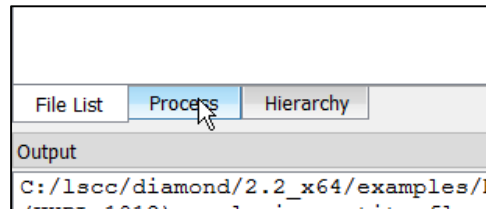
Please Note: Any obvious errors in the source file will be reported by Lattice Diamond in the error messaging console (frames #4 & 5). Please refer to the Lattice Diamond User Manual for a detailed explanation of the topics covered in this section.

Next, we will compile the project and then upload the resultant bit-file to FleaFPGA Uno.

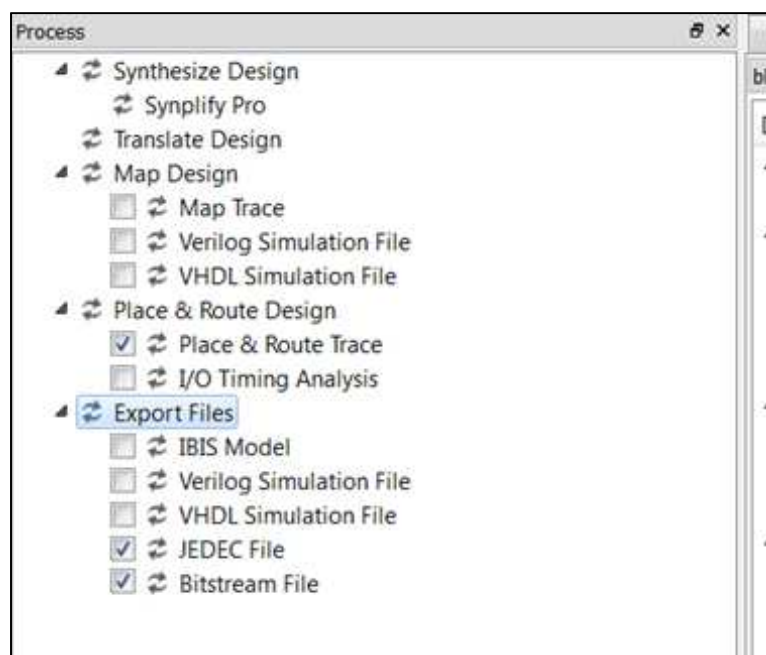
Section 7 – Using Lattice Diamond (Continued)

Section 7.4: Compiling a HDL project.

- i. Select the 'Process' tab located at the bottom of source tree (frame#1).



- ii. You will then see new information appear in Frame#1 (Process Tree Information). Please tick the option boxes as they appear in the following figure.

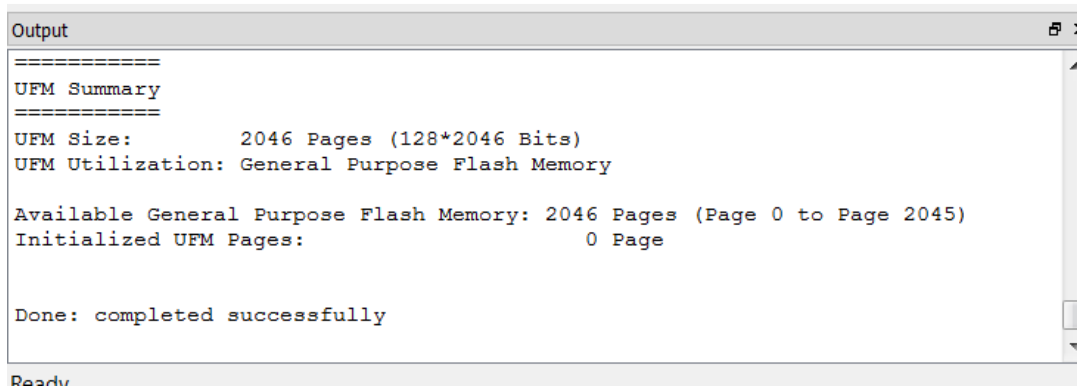


Please Note: While “Place & Route Trace” option may not be needed for this project, it is critical this report be generated, because it will be needed for more complex projects.

- iii. Double-click on the “Export Files” option as highlighted in the above figure. This will commence the HDL compilation process.

Section 7 – Using Lattice Diamond (Continued)

- iv. Lattice Diamond will now compile the project. Depending on the complexity of the HDL source (as well as available PC resources), this may take from 20 seconds or so to well over a few minutes in very-complex projects. If the project completed with no errors, you should see the following in the process status console (Frame#3):



```
Output
=====
UFM Summary
=====
UFM Size:          2046 Pages (128*2046 Bits)
UFM Utilization:  General Purpose Flash Memory

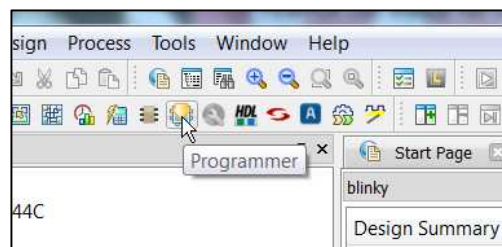
Available General Purpose Flash Memory: 2046 Pages (Page 0 to Page 2045)
Initialized UFM Pages:                    0 Page

Done: completed successfully
```

Section 7.5: Setting up Diamond programmer for use with FleaFPGA Uno

Please Note: This section only needs to be completed once at project creation time.

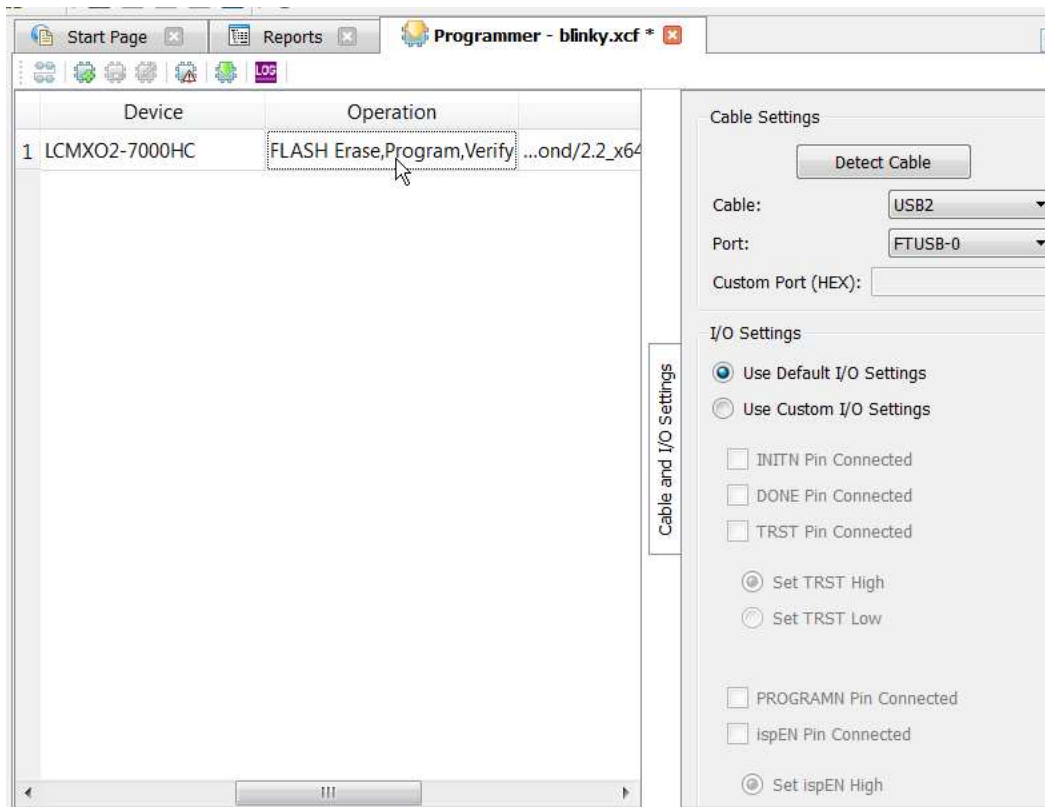
- i. Make Sure your FleaFPGA Uno's USB slave port is connected to the PC (ideally via a hub, but not essential).
- ii. From the small icons list in the top menu, Click on the programmer icon as shown:



- iii. You now see a child window titled "Programmer: Getting Started" appear. Select the "Create a New Blank Project" option and Click 'Ok'.

Section 7 – Using Lattice Diamond (Continued)

- iv. Now, in the source editor frame you will see some FPGA programming-related information (see below). Please scroll across until you see a column marked “Operation”. Double-click on the cell immediately below it as shown:



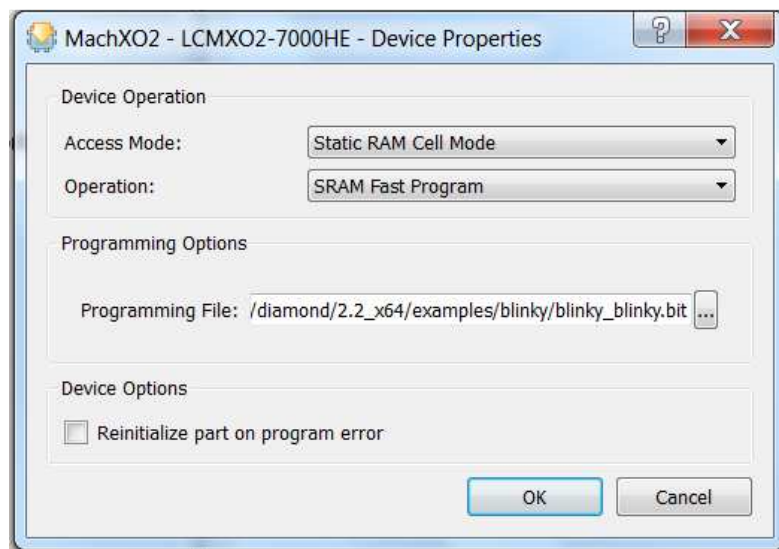
Please Note: Following page covers the two main programming *modes* for FleaFPGA Uno:

- 1.) **Loading user firmware into the FPGA configuration SRAM.** User firmware can be updated an infinite number of times with a very quick upload time of 3 seconds. However, user firmware uploaded in this mode is NOT permanently stored and will be lost upon power-off
- 2.) **Loading user firmware into the FPGA configuration Flash ROM.** User firmware may be permanently committed to the FPGA using this mode. However, programming time is considerably longer i.e. 20 seconds to over 2 minutes depending whether a USB hub is used or not. In addition, programming cycles are limited to 100,000 times or so before the Flash ROM begins to wear out.

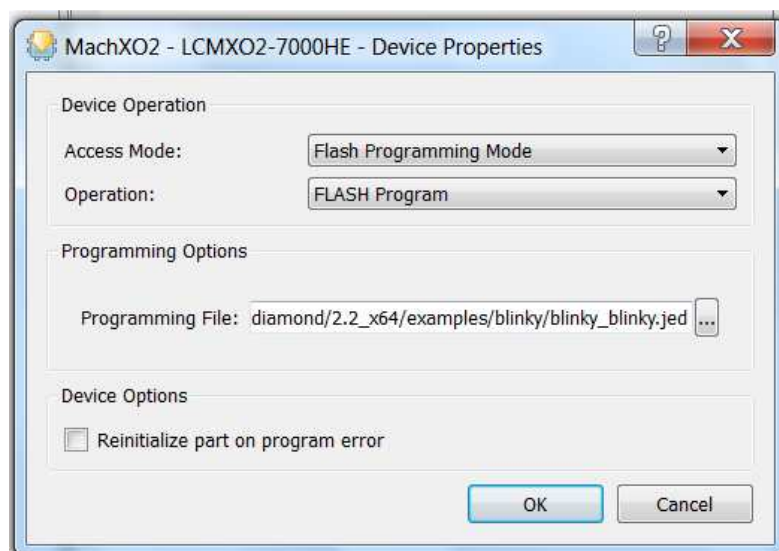
Section 7 – Using Lattice Diamond (Continued)

- v. **FPGA SRAM programming mode:** In the following child window, please select the Correct 'Access Mode', 'Operation' and 'Programming File' options as shown below. Note that for SRAM programming, Lattice Diamond uses the *.bit extension.

Warning! Before you can successfully access the fast SRAM programming mode, you *MUST* first program the FleaFPGA Uno with a valid bit-file to Flash ROM. Any attempt to program FleaFPGA Uno's configuration SRAM will simply not work otherwise. If you have not yet programmed your FleaFPGA Uno's Flash ROM, please skip this step and only come back when you can confirm that you have successfully load a bit-file into Flash ROM.

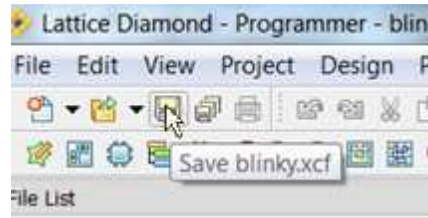


- vi. **FPGA Flash ROM programming mode:** In the following child window, please select the Correct 'Access Mode', 'Operation' and 'Programming File' options as shown below. Note that for Flash ROM programming, Lattice Diamond uses the *.jed extension.

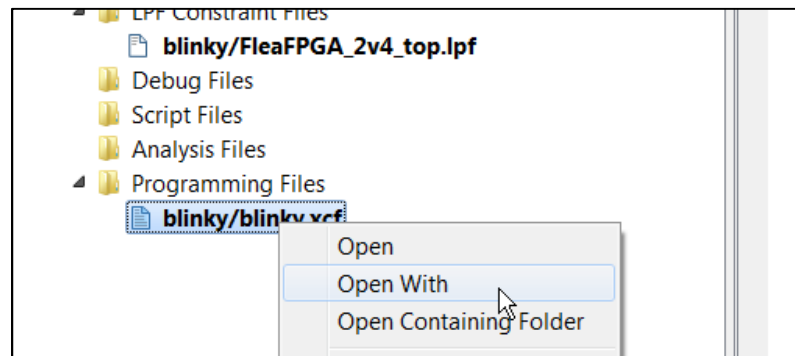


Section 7 – Using Lattice Diamond (Continued)

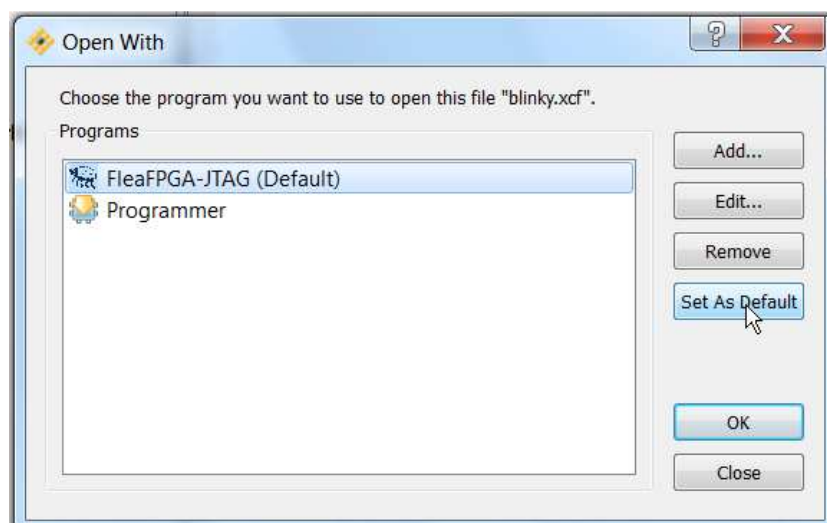
- vii. Save the current JTAG configuration by clicking on the related icon in the top menu as shown.



- viii. Now we will go back to our Source tree (Frame#1) and enable use of the FleaFPGA JTAG utility for use within Lattice Diamond. With the mouse cursor over the file "blinky.xcf" in the source tree hit right-click and then select "Open With" from the right-click menu as shown below:



- ix. Select the FleaFPGA JTAG utility from the following list as shown. Click "Set as Default" and then Click "Ok".



Section 7 – Using Lattice Diamond (Continued)

- x. You should now see the FleaFPGA JTAG utility console appear and begin to load FleaFPGA Uno with the blinky LED example.
- xi. Once the programming cycle has completed (and FleaFPGA JTAG has reported no errors), you should now see a slow-blinking #TX LED on the FleaFPGA Uno board of around once per second! 😊

If you got to this point without any problems, your Lattice Diamond software is now ready to create digital designs with FleaFPGA Uno! 😊

Please Note: To initiate successive programming cycles to FleaFPGA Uno via Lattice Diamond, simply double-click on “blinky.xcf” (or myproject.xcf etc.) in the source tree to trigger a custom firmware file upload to FleaFPGA Uno.

Section 8 – Where to from here?

Once you've followed the previous sections of this guide (i.e. installing software tools, loading firmware via JTAG, using lattice Diamond etc.), you may want to do more with your FleaFPGA Uno than blink an LED.. ☺

Exploring further HDL examples:

Further HDL project examples are available for FleaFPGA Uno. These can be found on http://www.fleasystems.com/fleaFPGA_Uno.html. Example projects include:

- F32c-based System-on-Chip <https://github.com/f32c/f32c>
- Sweet32-based System-on-Chip <https://github.com/Basman74/Sweet32-CPU>
- Apple-2 Home computer <http://www.cs.columbia.edu/~sedwards/apple2fpga/>
- More to follow! ☺

Developing your own HDL projects from scratch:

For those who are interested in developing their own HDL projects using FleaFPGA Uno, you will require the following (minimum):

- 1.) Some background in digital electronics i.e. Boolean Algebra, state machines, understanding timing diagrams as well as a good understanding of logic elements (i.e. gates, flip-flops tri-state buffers etc.) are required.
- 2.) If you are not familiar with VHDL (or Verilog) then you will need to learn it! Several reference guides are included for further study:
 - i. <http://www.ashenden.com.au/vhdl-book/SG2E.html>
 - ii. http://www.doulos.com/content/products/golden_reference_guides.php#Anchor-Th-61209
- 3.) It is essential that your custom HDL project 'knows' what external hardware is connected to the FPGA. Therefore, your project must include both **FleaFPGA_Uno_Toplevel.vhd** and **FleaFPGA_Uno_Toplevel.lpf** files. These can be found on www.fleasystems.com in the FleaFPGA projects page.
- 4.) Better understanding of the Lattice MachXO2 FPGA and Diamond tools, including the in-built HDL simulator (i.e. where you can test out your project before programming the FPGA). It is recommended the reader read the Lattice Diamond and MachXO2 User Guides available from www.latticesemi.com.

Attention! Though you're free to view, modify and upload these files to your FleaFPGA Uno, you need to make sure that you understand the underlying theory as outlined above.

It is recommended that interested readers download, unzip and then copy each of the above examples into the following path on your local machine i.e. C:\lsc\lsc\diamond\3.3_x64\examples

Once you've copied the project files, start Lattice Diamond. Open the desired project by selecting 'open -> project' from the file menu from the abovementioned path. Lattice project files use the *.ldf extension.

Section 8 – Where to from here? (Continued)

“What if I get stuck?”

Users with any questions relating to FleaFPGA Uno may:

- 1.) Read the FAQ on the FleaFPGA Uno: http://www.fleasystems.com/fleaFPGA_Uno.html
- 2.) Register on our forum <http://www.fleasystems.com/forums/> and ask a question there
- 3.) Send an email using the address provided in the contacts section of our site.

Section 9 – Notes

Section 10 – Notes

Section 11 – Notes

Appendix A – FleaFPGA Uno (Rev.E1) Specifications

Peripheral-set:

- Lattice MachXO2-7000HC FPGA
- 512KByte (512Kx8-bit) 10nsec. Parallel SRAM (expandable to 2MBytes on-board)
- 16MByte (128Mx1-bit) SPI user Flash ROM.
- Digital Video output interface - 640x480 and 800x600 @ 24bit color supported
- “Combo” USB type-A port, enabling the following options (dependent on FPGA config.):
 1. PS/2 Keyboard or Mouse port (but not both)
 2. (To be implemented) USB 1.1 host port.
- 3.5mm stereo audio via 1bit PWM + NTSC composite video out via 4bit DAC.
- Separate USB slave serial port via FTDI interface. Also serves as a JTAG interface for easy programming of the FPGA.
- 22 x 3.3V-level GPIO lines via Arduino® compatible shield header – including six capable of Analog to Digital conversion (ADC). ADC resolution is ~1900 counts Full-scale. On FleaFPGA Uno, IOREF and AREF are both tied to 3.3V (See Appendix B).

WARNING: DO NOT install a user shield that requires 5V I/O to operate, doing so may risk damaging both FleaFPGA Uno and/or the installed user shield and voiding the warranty!

- One push-button (Marked as RESET on the board)
- Two user LEDs (Marked as TX and RX on the board)
- 25MHz onboard oscillator.
- Optional ESP8266 WiFi to serial interface module.

Electrical Ratings:

- Typical current draw: Depending on the uploaded FPGA user-configuration as well as active peripherals (if any). Generally around 50mA - 250mA range. This estimate does not include power drawn from the USB host port or GPIO header.
- GPIO Header pins: 3.3V CMOS-compatible with 8mA current source/sink capability. (Please refer to the Lattice MachXO2 datasheet for more detailed information).
- **Maximum current draw from FleaFPGA Uno’s USB host port is 150mA**
- **Maximum recommended current draw from FleaFPGA Uno’s Shield header is 250mA**
- **FleaFPGA Uno’s Input polyfuse rated for a maximum continuous current of 750mA**

Mechanical:

- PCB Dimensions: 69 x 63.5mm
- Overall dimensions: 74 x 63.5 x 15mm (Typically, with WiFi option installed).
- Four 3.1mm diameter mounting holes arranged in Arduino-compatible pattern.
- System rated for 0-50deg.c temperature ambient.

Appendix B – FleaFPGA Uno Board Schematics

