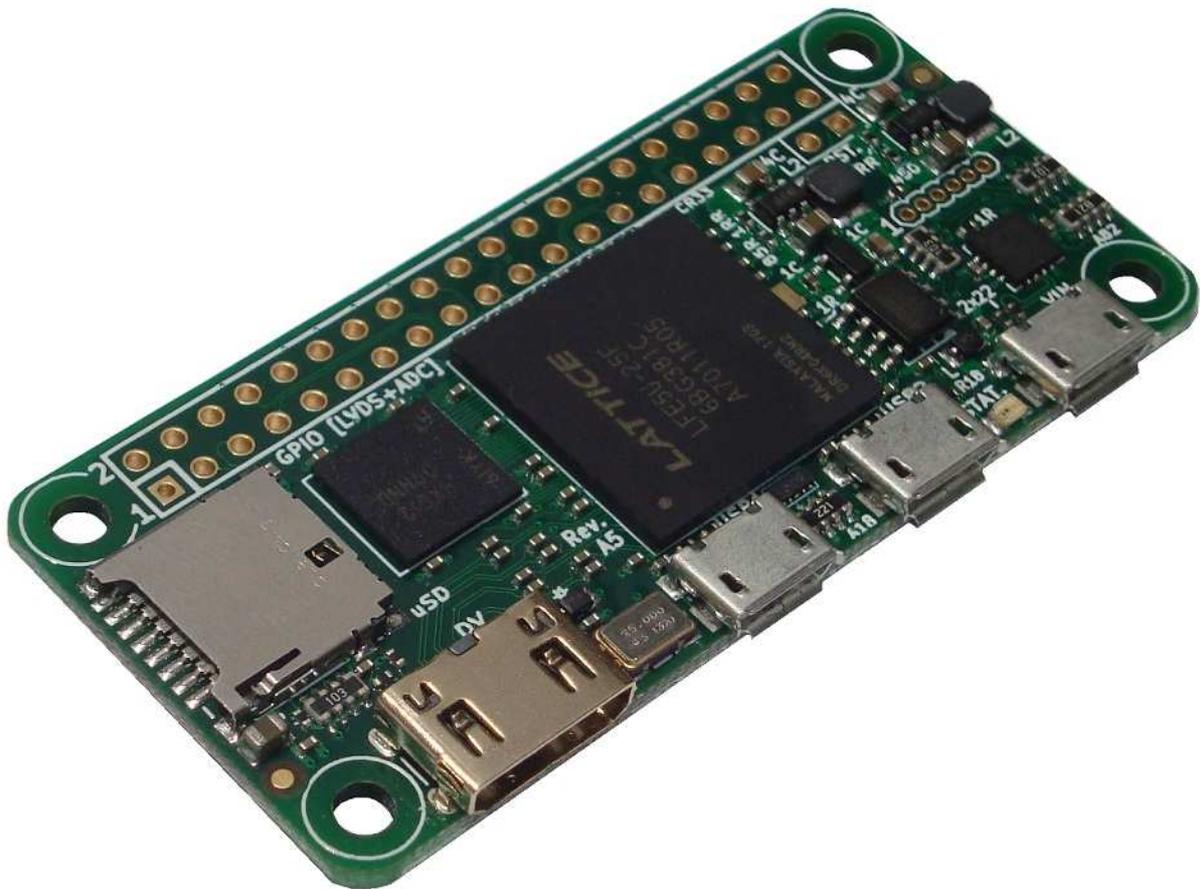


# FleaFPGA Ohm Experimenter Board Quick Startup Guide

Revision 0.12 (Preliminary)



---

[www.fleasystems.com](http://www.fleasystems.com)

# Contents:

SECTION 1 – DISCLAIMER.....	3
SECTION 2 – INTRODUCTION .....	4
SECTION 3 – FLEAFPGA OHM CONNECTIVITY DIAGRAM .....	5
SECTION 3.1 – GPIO HEADER CROSS REFERENCE TABLE .....	7
SECTION 4 – FLEAFPGA OHM HARDWARE OVERVIEW .....	8
SECTION 5 – SOFTWARE INSTALLATION .....	10
SECTION 6 – LOADING BIT-FILES INTO FLEAFPGA OHM.....	16
SECTION 7 – USING LATTICE DIAMOND .....	17
SECTION 8 – WHERE TO FROM HERE? .....	29
SECTION 9 – NOTES.....	31
APPENDIX A – FLEAFPGA OHM (REV.A5) SPECIFICATIONS .....	32
APPENDIX B – FLEAFPGA OHM BOARD SCHEMATICS .....	34
APPENDIX C – CONNECTING FLEAFPGA OHM TO A RASPBERRY PI COMPUTER VIA GPIO HEADER.....	37

# Section 1 – Disclaimer

## **Legal disclaimer:**

“FleaFPGA Ohm Experimenter Board” is intended to be treated as a user-configurable, electronic sub-assembly or module for evaluation purposes only. For users developing commercial products intended for sale that include the “FleaFPGA Ohm Experimenter Board” as a functional element: It is the user’s responsibility to ensure their final system design meets their own intended design specification, as well as any regulatory, compliance and/or Intellectual Property licensing requirements.

For all user applications requiring standalone or unattended operation of the “FleaFPGA Ohm Experimenter Board”: Users must understand the “FleaFPGA Ohm Experimenter Board” **does NOT contain any on-board protection fuses to cover for the unlikely event of an on-board power supply failure resulting in a short-circuit condition on the +5Volt supply rail with respect to ground.** It is therefore up to the user to include a suitably-rated circuit protection fuse (based on the final current demand, including inrush currents etc.) feeding into their “FleaFPGA Ohm Experimenter Board” to help reduce the risk of possible catastrophic failure.

Through the use of “FleaFPGA Ohm Experimenters Board”, you agree to indemnify and hold harmless the author of this document from any consequential damages or claims arising from, or associated with, use of “FleaFPGA Ohm Experimenters Board”.

All HDL examples ported to FleaFPGA Ohm by Valentin Angelovski, are provided on an ‘as-is’ basis in good faith. While these examples appear to work within the constraints set forth in their accompanying README.TXT files, they are purely intended as means of evaluating the FleaFPGA Ohm hardware.

**The “FleaFPGA Ohm Experimenter Board” is NOT RECOMMENDED for use as a solution for applications deemed to be Safety-critical or life-support related. Users shall assume all responsibility for any consequences arising from such use.**

While every effort is made to ensure the information presented is accurate the author makes no such guarantee. I, Valentin Angelovski, reserve the right to modify or update this document without prior notice. This document is the copyright of Valentin Angelovski. All trademarked product names listed herein belong to their respective trademark owners.

**Copyright © 2018 Valentin Angelovski**

## Section 2 – Introduction

Hello! Thank you for choosing the FleaFPGA Ohm Experimenter Board! ☺

As its name suggests, FleaFPGA Ohm is built around the relatively new and exciting world of FPGA (Field-Programmable Gate Array) Technology.

Unlike traditional microprocessor or microcontroller-based boards, FleaFPGA Ohm uses hardware-oriented languages like VHDL or Verilog to realize *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, Communication, Video, Encryption modules and hardware accelerators to name a few!

(\*) = Within the physical limits of the on-board FPGA. FleaFPGA Ohm is built around Lattice's ECP5 FPGA product. Refer to the Lattice ECP5 Datasheet (Document Reference 'FPGA-DS-02012') for further information.

The aims of this startup guide are three-fold:

- 1.) To provide a useful reference for the FleaFPGA Ohm 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 Ohm 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 HDL 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 HDL, basic electrical/electronic and digital logic theory. Further suggestions are provided in Section 8 titled "Where to from here?".

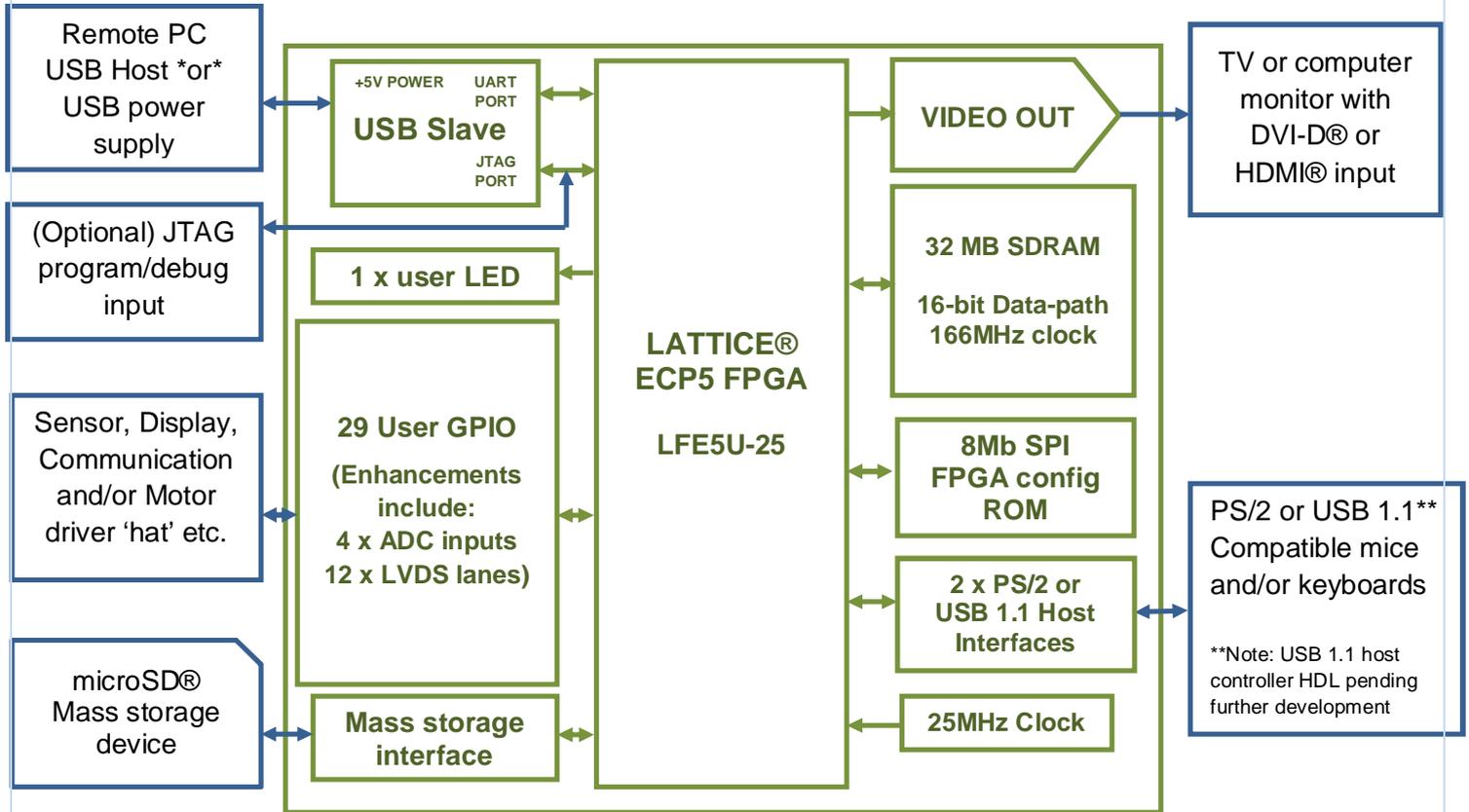
Should you have any questions or feedback about this guide, kindly drop us an email from the contacts section on our site.

Happy experimenting! ☺

Valentin Angelovski

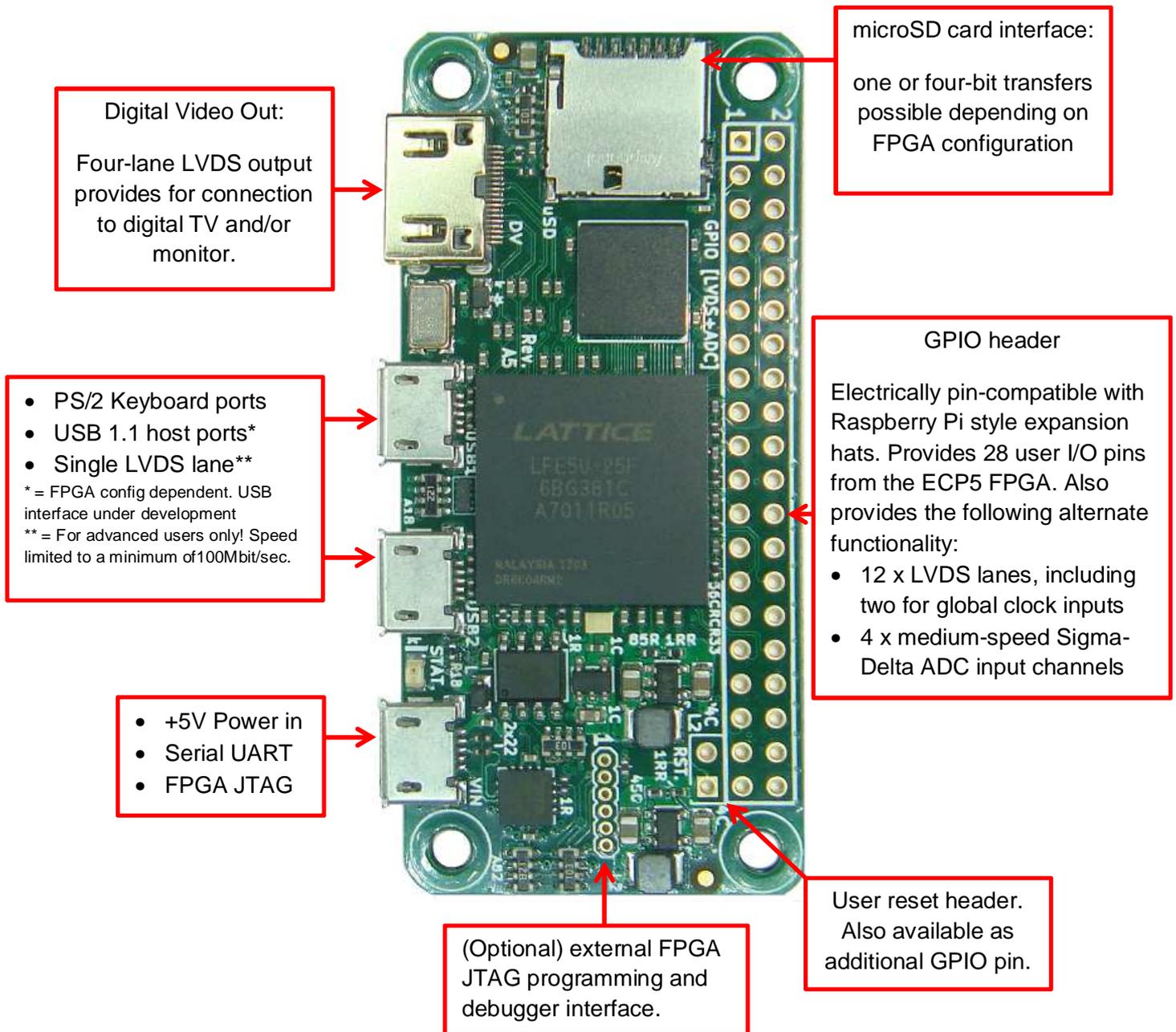
# Section 3 – FleaFPGA Ohm connectivity diagram

FleaFPGA Ohm Rev.A5 PCB SYSTEM BLOCK DIAGRAM:



## Section 3 – FleaFPGA Ohm connectivity diagrams (Continued).

### SYSTEM CONNECTIVITY OPTIONS - OVERVIEW:



## Section 3.1 – GPIO Header Cross Reference Table

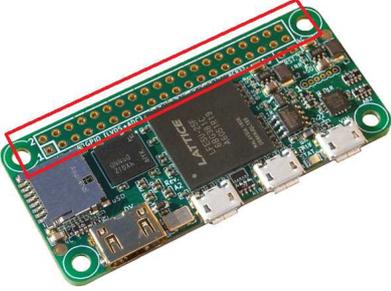
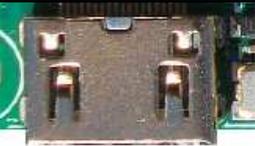
**TABLE#1 FleaFPGA Ohm <-> Raspberry Pi GPIO cross-reference table:**

Following table shows how GPIO electrical signals are physically routed, to the ECP5 FPGA on the FleaFPGA Ohm board from a pin-compatibility perspective (**WARNING: Do NOT** directly plug your FleaFPGA Ohm board into a raspberry pi computer's GPIO header!. Please see Appendix-C for further details):

GPIO Pin #:	RPi Pin definition:	ECP5 FPGA Pin Reference:		GPIO Pin #:	RPi Pin definition:	ECP5 FPGA Pin Reference:
1	+3.3V			2	+5V	
3	GPIO_2	D17		4	+5V	
5	GPIO_3	C18		6	GND	
7	GPIO_4	D19		8	GPIO_14	C20
9	GND			10	GPIO_15	E17
11	GPIO_17	F17		12	GPIO_18	D18
13	GPIO_27	G18		14	GND	
15	GPIO_22	F19		16	GPIO_23	E20
17	+3.3V			18	GPIO_24	D20
19	GPIO_10	G19		20	GND	
21	GPIO_9	J19		22	GPIO_25	L20
23	GPIO_11	K19		24	GPIO_8	M20
25	GND			26	GPIO_7	L19
27	GPIO_IDSD	N19		28	GPIO_IDSC	N20
29	GPIO_5	T20		30	GND	
31	GPIO_6	T19		32	GPIO_12	R18
33	GPIO_13	N16		34	GND	
35	GPIO_19	U18		36	GPIO_16	M17
37	GPIO_26	U17		38	GPIO_20	N17
39	GND			40	GPIO_21	P16

## Section 4 – FleaFPGA Ohm hardware overview

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

Peripheral:		Hardware details																																																					
		<p><b>User GPIO:</b> Electrically pin-compatible with Raspberry Pi expansion hats <b>only</b>. Provides up to 28 programmable user digital bidirectional lines based on user FPGA configuration. Please see Appendix B and Section 3.1 for further details. In addition, this GPIO header has a secondary ‘enhanced’ functionality when FleaFPGA Ohm is used in a standalone manner (ie. not connected to a Raspi computer or expansion hat). These include: ADC and LVDS functionality. Please see the following two feature rows for further details. <b>WARNING: I/O pins operate at 3.3V levels only and are <u>not</u> 5V tolerant! Also, users seeking to use FleaFPGA Ohm as an expansion ‘hat’ for their raspberry pi computer should refer to Appendix C.</b></p>																																																					
<p>ADC Pin allocation table:</p> <table border="1"> <thead> <tr> <th>ADC #</th> <th>GPIO header Pin#</th> <th>ECP5 Pin Ref:</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>18</td> <td>D20</td> </tr> <tr> <td>1</td> <td>19</td> <td>G19</td> </tr> <tr> <td>2</td> <td>26</td> <td>L19</td> </tr> <tr> <td>3</td> <td>29</td> <td>T20</td> </tr> </tbody> </table>		ADC #	GPIO header Pin#	ECP5 Pin Ref:	0	18	D20	1	19	G19	2	26	L19	3	29	T20	<p><b>Sigma-Delta ADC inputs:</b> Each analog input in the FleaFPGA Ohm can be thought of as a 1-bit ADC designed for 250 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 <a href="#">Application Note RD1066</a> as well as on <a href="#">David Kessner’s technical blog site</a> Also refer to the default “Simple_ADC.vhd” module as found in the Sweet32 SoC example to see how it is implemented in VHDL.</p>																																						
ADC #	GPIO header Pin#	ECP5 Pin Ref:																																																					
0	18	D20																																																					
1	19	G19																																																					
2	26	L19																																																					
3	29	T20																																																					
<p>GPIO-based LVDS - pin allocation table:</p> <table border="1"> <thead> <tr> <th>LVDS Pair #</th> <th>GPIO header Pin# (+/-):</th> <th>ECP5 FPGA Pin Ref. (+/-):</th> <th>ECP5 Global clock input?</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>5 / 3</td> <td>C18 / D17</td> <td></td> </tr> <tr> <td>1</td> <td>8 / 7</td> <td>C20 / D19</td> <td></td> </tr> <tr> <td>2</td> <td>12 / 10</td> <td>D18 / E17</td> <td></td> </tr> <tr> <td>3</td> <td>11 / 13</td> <td>F17 / G18</td> <td></td> </tr> <tr> <td>4</td> <td>16 / 15</td> <td>E20 / F19</td> <td></td> </tr> <tr> <td>5</td> <td>21 / 23</td> <td>J19 / K19</td> <td>Yes</td> </tr> <tr> <td>6</td> <td>22 / 24</td> <td>L20 / M20</td> <td>Yes</td> </tr> <tr> <td>7</td> <td>27 / 28</td> <td>N19 / N20</td> <td></td> </tr> <tr> <td>8</td> <td>31 / 32</td> <td>T19 / R18</td> <td></td> </tr> <tr> <td>9</td> <td>33 / 36</td> <td>N16 / M17</td> <td></td> </tr> <tr> <td>10</td> <td>35 / 37</td> <td>U18 / U17</td> <td></td> </tr> <tr> <td>11</td> <td>38 / 40</td> <td>N17 / P16</td> <td></td> </tr> </tbody> </table>		LVDS Pair #	GPIO header Pin# (+/-):	ECP5 FPGA Pin Ref. (+/-):	ECP5 Global clock input?	0	5 / 3	C18 / D17		1	8 / 7	C20 / D19		2	12 / 10	D18 / E17		3	11 / 13	F17 / G18		4	16 / 15	E20 / F19		5	21 / 23	J19 / K19	Yes	6	22 / 24	L20 / M20	Yes	7	27 / 28	N19 / N20		8	31 / 32	T19 / R18		9	33 / 36	N16 / M17		10	35 / 37	U18 / U17		11	38 / 40	N17 / P16		<p><b>Low Voltage Differential Signalling (LVDS) lanes:</b>  <b>** RECOMMENDED FOR ADVANCED USERS ONLY **</b>                      Potentially allows high-speed serial transmission (up to a theoretical maximum of 800Mbps per lane) through the use of differential signaling. FleaFPGA Ohm includes 12 such LVDS pairs, including two that may also be used as global clock inputs feeding into the ECP5 FPGA.  <b>WARNING:</b> Assume LVDS33 due to VCCIO voltage of +3.3V Therefore, when configuring the LVDS lanes as outputs, level translators may be required to convert to different LVDS standard ie. LVDS25 or LVDS18 etc.                      Please refer to Appendix B as well as Lattice ECP5 datasheet and application documents: TN1260, TN1263 and TN1265 for further information.</p>	
LVDS Pair #	GPIO header Pin# (+/-):	ECP5 FPGA Pin Ref. (+/-):	ECP5 Global clock input?																																																				
0	5 / 3	C18 / D17																																																					
1	8 / 7	C20 / D19																																																					
2	12 / 10	D18 / E17																																																					
3	11 / 13	F17 / G18																																																					
4	16 / 15	E20 / F19																																																					
5	21 / 23	J19 / K19	Yes																																																				
6	22 / 24	L20 / M20	Yes																																																				
7	27 / 28	N19 / N20																																																					
8	31 / 32	T19 / R18																																																					
9	33 / 36	N16 / M17																																																					
10	35 / 37	U18 / U17																																																					
11	38 / 40	N17 / P16																																																					
		<p><b>Digital Video Out:</b> 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 720p60 or 1080p30, depending on FPGA configuration. Alternate LVDS input functionality also possible (see Appendix B for further details).</p>																																																					
		<p><b>External JTAG Header (optional):</b> For use with an external JTAG debugger for the ECP5 FPGA. Please see Appendix B for pin-out information.</p>																																																					

## Section 4 – FleaFPGA Ohm hardware overview (continued)

Peripheral:	Hardware details
	<p><b>USB Slave port:</b> Performs the following functions:</p> <ol style="list-style-type: none"> <li>1.) Main power supply feed for FleaFPGA Ohm (<b>Warning: NOT fuse protected!</b>). Current consumption by FleaFPGA Ohm with all peripherals connected should not exceed 750mA continuous current.</li> <li>2.) Programming of the on-board FPGA via JTAG</li> <li>3.) Virtual COM port, fed to the ECP5 for console debugging (includes RTS signal connection for potential remote reset of an FPGA core from host PC).</li> </ol> <p><b>Attention!</b> Due to the shared nature of the USB slave design, users must not have the COM port open while performing JTAG operations on FleaFPGA Ohm. Please close any open terminal programs to FleaFPGA's slave COM port before connecting to FleaFPGA Ohm using the JTAG loader utility (and vice versa 😊).</p>
	<p><b>32MByte 166MHz SDRAM:</b> Organised as 16M x 16bit data width, based on Micron Technology's "MT48LC16M16A2F4-6A" SDRAM. Refer to the schematics on Appendix B for pin-out details, or the manufacturer's datasheet for detailed device information.</p>
	<p><b>1MByte SPI ROM:</b> Primarily used for storing ECP5 FPGA configuration data, but does allow at least 64kBytes of free memory at the top-of-ROM for user boot programs etc. Based on Winbond "W25Q80DVSNIG" Flash ROM. Refer to the schematics on Appendix B for pin-out details, or the manufacturer's datasheet for detailed device information.</p>
	<p><b>User 'Activity' LED:</b> Can be user controlled by the ECP5 FPGA for process indication or status. See 'blinky' HDL example as well as Appendix-B for further details.</p>
	<p><b>RESET Header:</b> Normally functions as a system reset jumper (depending on HDL core loaded into the FPGA), but can alternatively be re-configured as a user GPIO pin as well. See Appendix B further details.</p>
	<p><b>Combined PS/2 or (firmware pending) USB 1.1 Host Ports:</b> Provides user input from keyboard or mouse devices with a PS/2 interface. USB 1.1 Host firmware currently under development. Alternatively, ports are also wired for limited but possible single-lane LVDS functionality. It is recommended that drawn from each port should not exceed 150mA.</p> <p><b>Warning: +5Volt rails Not overcurrent protected by FleaFPGA Ohm in any way. Please ensure your 5 Volts supply feed going into FleaFPGA Ohm has fuse and overcurrent protection!</b></p>
	<p><b>MicroSD Card Slot:</b> Supports both SPI (1-bit) and 4-bit data access modes. Refer to Appendix B for further details.</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 Ohm 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 Ohm) that may be downloaded from our fleasystems.com website to be programmed into their FleaFPGA Ohm platform.

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

### **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 Ohm. This is accomplished by installing Lattice Diamond Development and configuring it for use with the FleaFPGA Ohm platform.

**Please Note:** All examples contained in this guide are screenshots from a Windows 7 installation. Windows 8-10 installation processes should be practically identical however.

## Section 5 – Software Installation (Continued)

### **Before you begin:**

You will need the following (minimum) hardware:

- Modern PC with Microsoft Windows 7 (Ideally 64-bit version, but not essential) or newer installed and also web-enabled. Must have at least 4GBytes RAM (8GB or more recommended), as well as 30GBytes of free Hard Disk space minimum and one free USB host port.
- 'FleaFPGA Ohm' Experimenter Board
- USB Type A to Micro B Male adapter cable as needed for power, serial communication and/or JTAG programming tasks)

Additional (recommended, optional or HDL application-dependent) hardware:

- For interfacing with composite USB or PS/2 peripherals via the on-board host ports: One or two USB micro OTG host to Type A USB host adapter cables may be required for communication with composite USB mice and keyboards (Composite USB meaning devices that can also speak PS/2 as well as USB). Additionally, PS/2 to USB adapter dongles may also be needed for interfacing with keyboards and/or mice that have a physical PS/2 connector.
- Mini-HDMI to HDMI adapter cable (must be less than 5 meters in length).
- Suitably formatted microSD card.
- For the case where frequent writes to FPGA configuration ROM are intended, a USB hub is recommended. Allows for JTAG Flash ROM programming time to be reduced from approximately seven minutes down to 35 seconds. For typical FPGA development work, JTAG programming of FPGA configuration SRAM area is relatively quick by comparison (around 12-14 seconds) regardless of whether a USB hub is used or not.
- External JTAG debugger module. Recommended if real-time debugging and faster programming of the ECP5 FPGA are both required (real-time debugging using Lattice Reveal is not covered by this quick-start Guide).

## 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](http://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 Ohm 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 Ohm example projects were built in a 64-bit operating environment, hence all relevant file paths will have a “x64” string added to them. Lattice Diamond version 3.9 is described in this document.**

- 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](http://www.latticesemi.com) to find the relevant support page): <http://www.latticesemi.com/latticediamond>
- 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\3.9\_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. Download the following GitHub repository <https://github.com/Basman74/FleaFPGA-Ohm>
- ii. You will see several files contained within the above repository, where one of these is named “FleaFPGA\_Ohm\_test.vme”. This file is basically a pre-built test firmware image to be uploaded to FleaFPGA Ohm, please save the file in a temporary directory i.e. c:\temp
- iii. Run the **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:

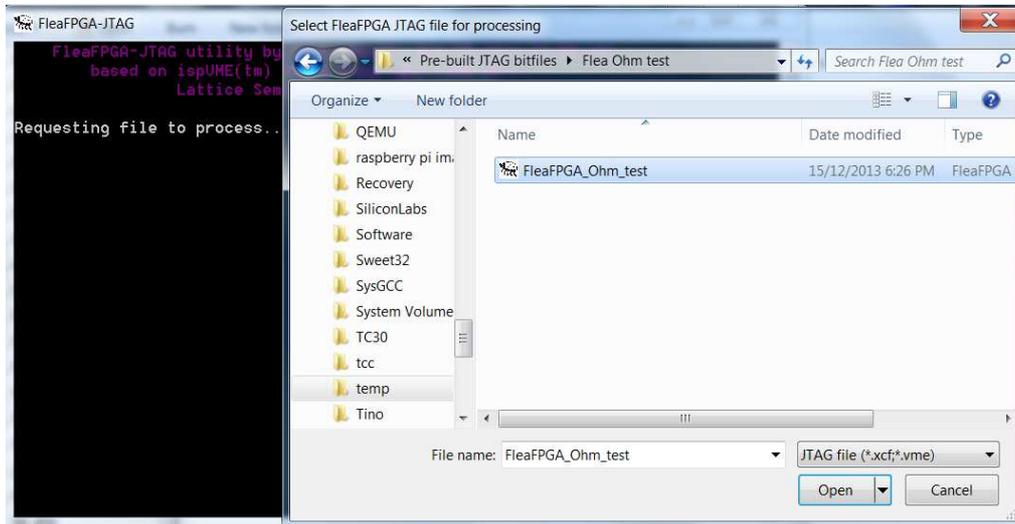


- iv. Make sure the ‘Launch FleaFPGA JTAG Utility’ option is ticked and then click finish. FleaFPGA utility will now run and request the path of “ddtcmd.exe”. This file can typically be found in the following path (assuming you’ve installed the 64-bit Windows version of lattice Diamond v3.9 here): C:\lscd\diamond\3.9\_x64\bin\nt64

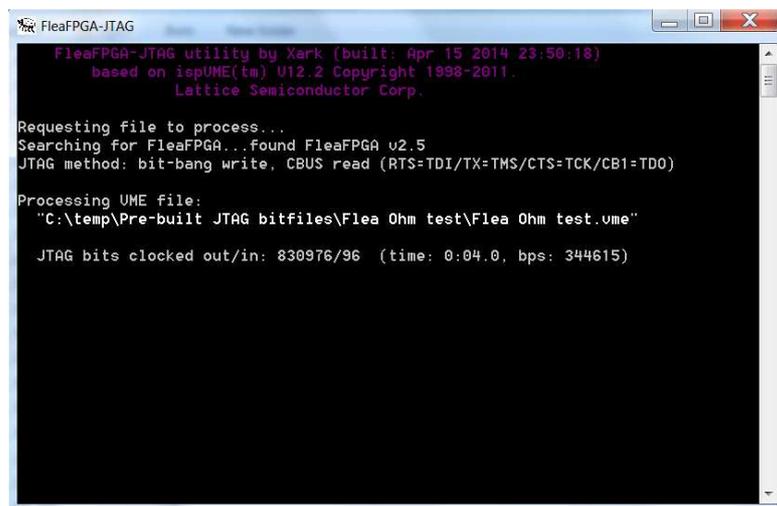
**Please Note:** You only need to select the relevant file and then click “Open” if you have Diamond installed and plan on using it to create your own custom FPGA-based logic designs. If you only wish to play with pre-built FPGA firmware examples and do NOT have Lattice Diamond installed, just click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

## Section 5 – Software installation (Continued)

- v. FleaFPGA JTAG utility will then ask for a programming file (\*.VME extension) to be uploaded to the FleaFPGA board. Please locate the previously-saved “FleaFPGA\_Ohm\_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\_Ohm\_test.vme” bit-file. Program upload screen should appear as follows:



- vii. Once the FleaFPGA Ohm flash program cycle (approximately 40 seconds with, or over 5 minutes without a USB hub!) has completed successfully, you should now see the ‘STAT’ LED blink roughly once per second on your FleaFPGA Ohm board!

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

## Section 6 – Loading bit-files into FleaFPGA Ohm

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

- 1.) Using the FleaFPGA JTAG utility to load suitable firmware files (.VME extension).
- 2.) Loading custom firmware files via Lattice Diamond (which then calls our FleaFPGA JTAG utility to program the ECP5 FPGA).

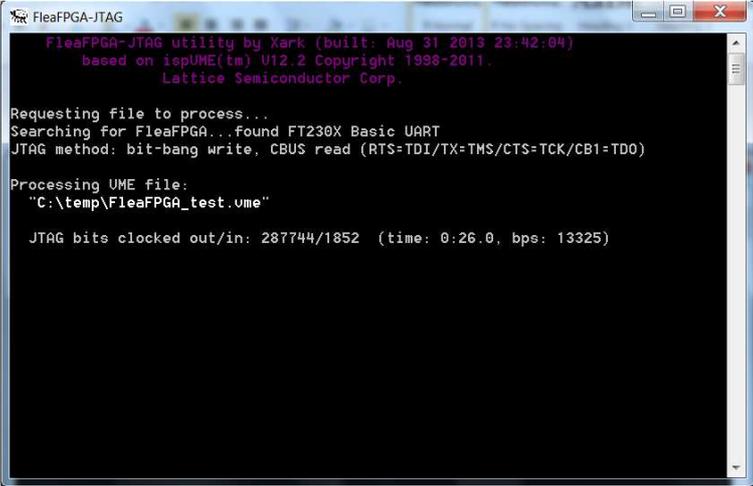
Only method #1 will be discussed in this section. For users interested in custom FPGA experiments, please refer to Section 7 and up. Pre-built .vme example files can be found in the GitHub links provided in section 8.

**Please Note:** Ensure that a suitable USB cable is plugged into your FleaFPGA Ohm's slave USB port. This can be done either directly, or via a hub (if preferred, as recommended in Section 5) from the host PC.

- i. With only the FleaFPGA JTAG utility installed on your system, a 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 Ohm 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 Ohm. Diamond allows the user to configure FleaFPGA Ohm 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 Ohm 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 ECP5 FPGA. Programming the FPGA from within Lattice Diamond is also covered.

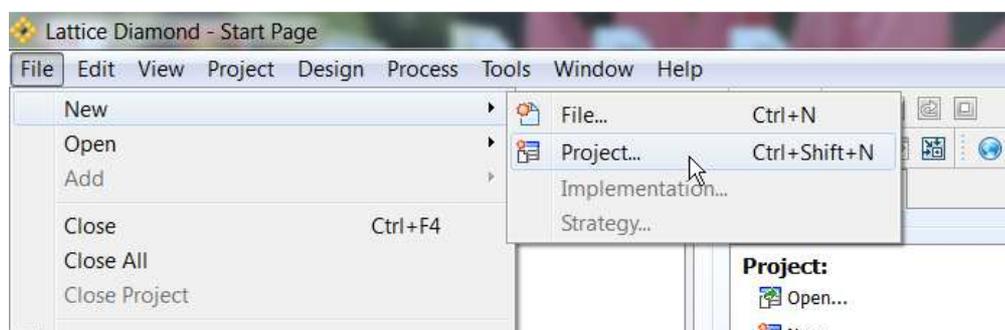
### Section 7.1: Download example project source files

- i. Download 'FleaFPGA\_Ohm\_test.zip' from the following GitHub repository <https://github.com/Basman74/FleaFPGA-Ohm>. Contained in the zip will be the following files:
  - FleaFPGA\_Ohm\_A5\_Top.vhd (Contains FleaFPGA Ohm-specific interface code)
  - FleaFPGA\_Ohm\_A5.lpf (Contains FleaFPGA Ohm-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 the FleaFPGA Ohm Experimenter Board (NOTE: For advanced custom HDL projects requiring the use of high speed signaling via LVDS etc., some modifications to these files may be required. These advanced methods fall beyond the scope of this Quick-start guide).

### Section 7.2: Create a new HDL project

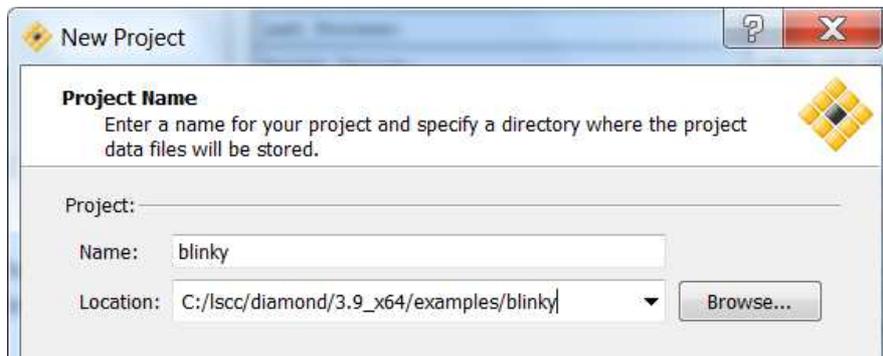
- i. Start Lattice Diamond and create a new project file as shown in the figure below:



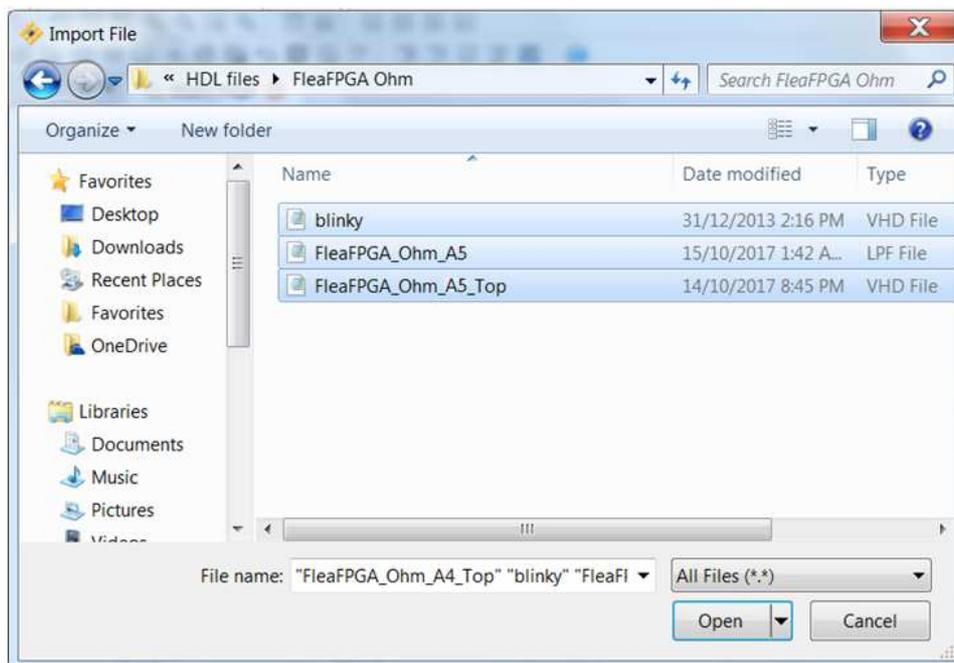
- ii. You should see a 'new project' child window appear. Click 'Next'.

## Section 7 – Using Lattice Diamond (Continued)

- iii. Diamond will now ask for a project name as well as a new working directory. Please enter the name 'Blinky' in the text box, as well as append the same name on to the end of the Location string as shown in the figure below:

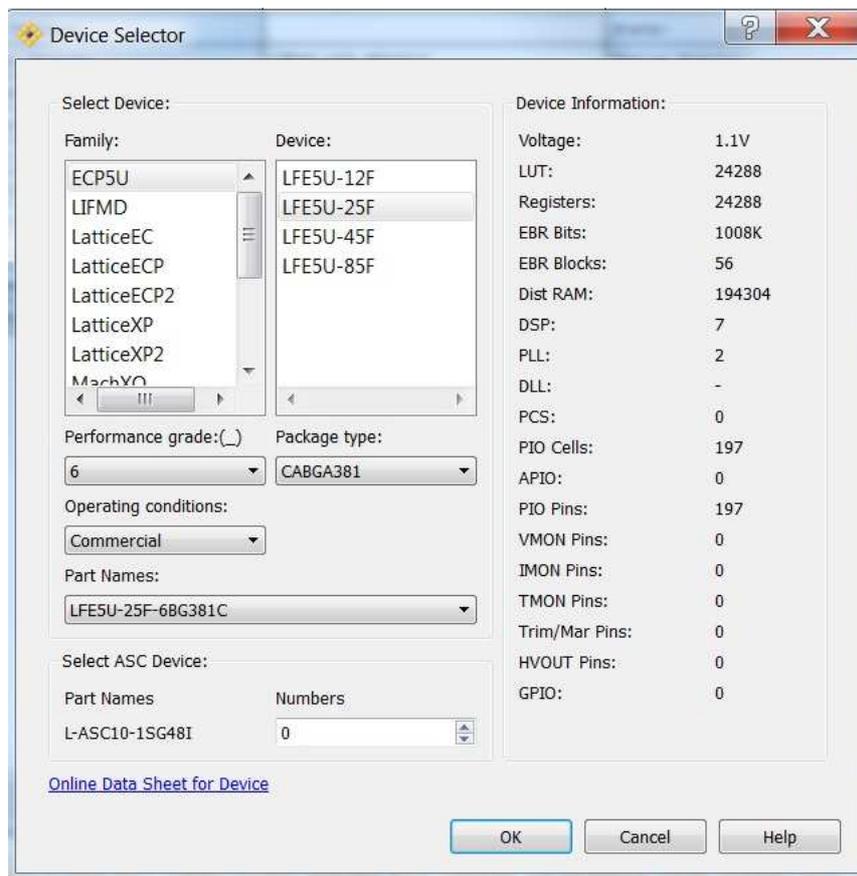


- iv. Next you will be asked to add any VHDL source files to the project. Please tick the “copy source to implementation directory” box. Next, Click “Add source” and then a file import window will appear. Select “All files” in the file extension selection and then select all the files contained within the FleaFPGA\_Ohm\_test.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. FleaFPGA Ohm uses the Lattice ECP5 (LFE5U-25F) FPGA device in the CABGA381 package style. Please select all the dropdown options as shown in the figure below and then click 'Next'.

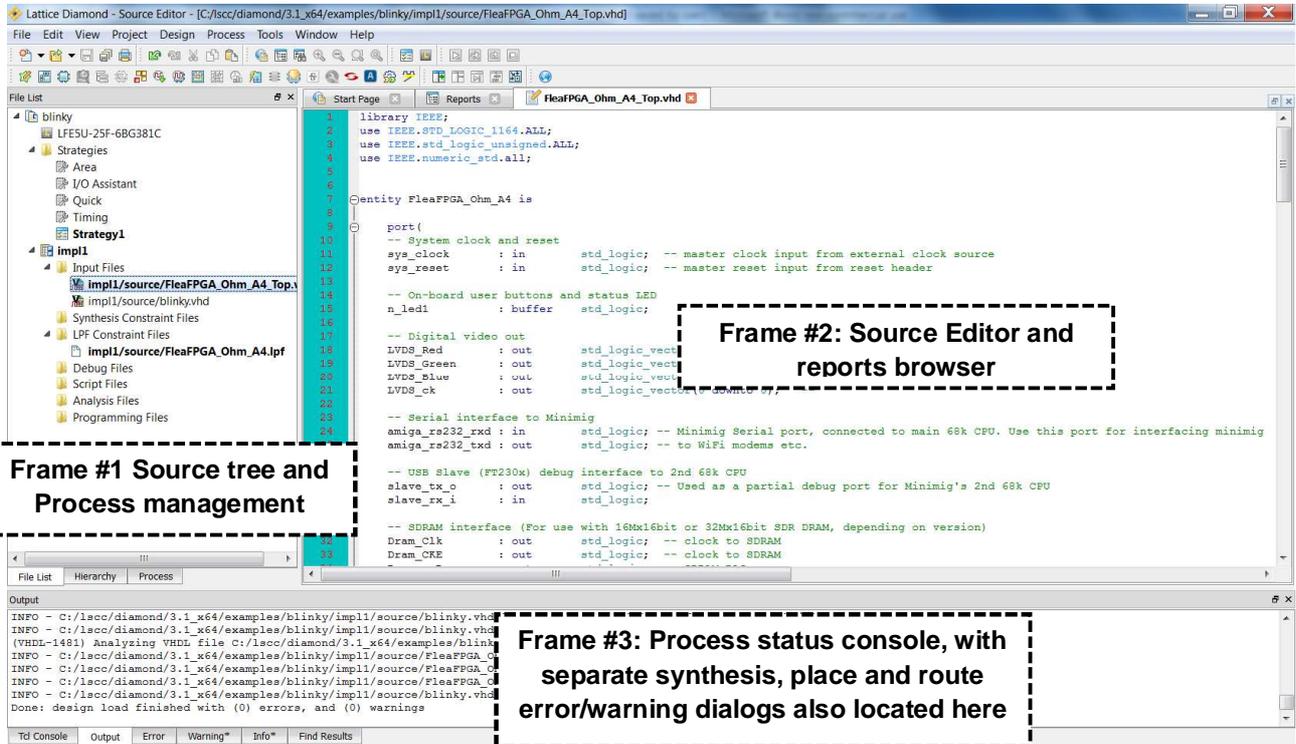


- vi. You will now be asked to select a synthesis tool. Select Synplify Pro and then click 'Next'.
- vii. Diamond will now provide you with a summary of the settings you've made for your new project. Make sure the summary matches this guide and then click 'Finish'.

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:

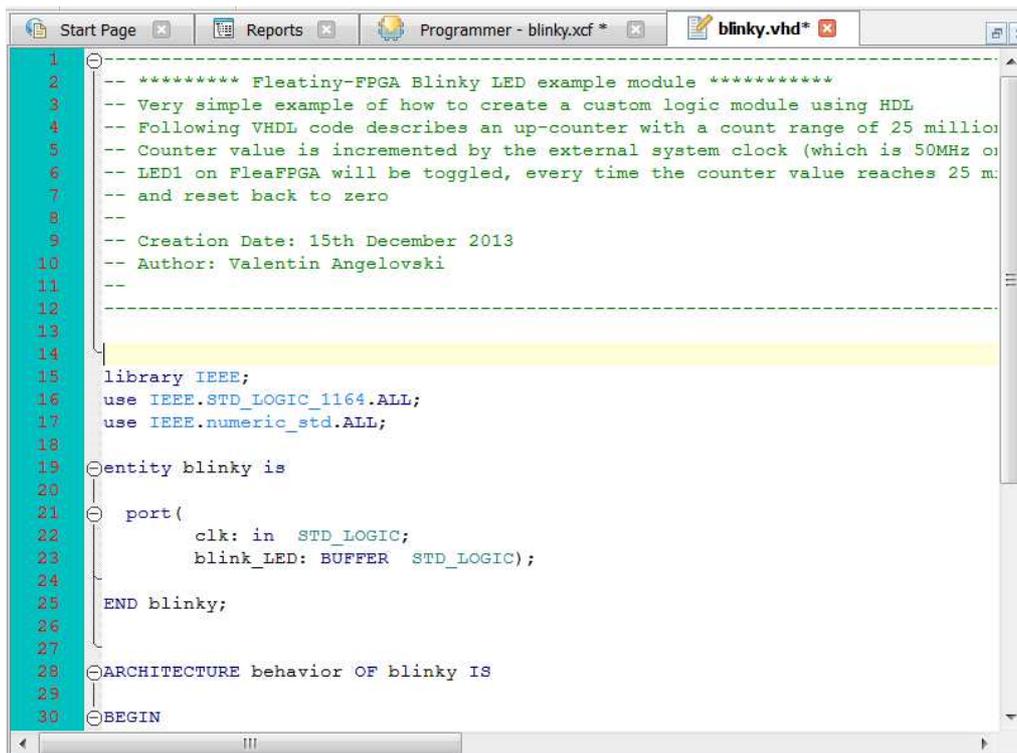


For the remainder of this section, we will be referring to the numbered frames shown above to help emphasize which frame within Lattice Diamond that we're covering.

## 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:

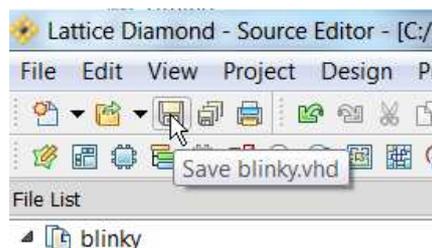


```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
-- ***** Fleatiny-FPGA Blinky LED example module *****
-- Very simple example of how to create a custom logic module using HDL
-- Following VHDL code describes an up-counter with a count range of 25 million
-- Counter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

entity blinky is
port(
    clk: in STD_LOGIC;
    blink_LED: BUFFER STD_LOGIC);
END blinky;

ARCHITECTURE behavior OF blinky IS
BEGIN
```

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



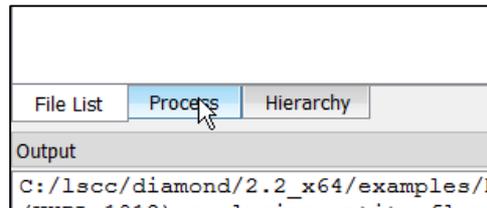
**Please Note:** Any obvious errors in the source file will be reported by Lattice Diamond in the error messaging console (error tab in frame #3). Please refer to the Lattice Diamond User Manual for a detailed explanation of the topics covered in this section.

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

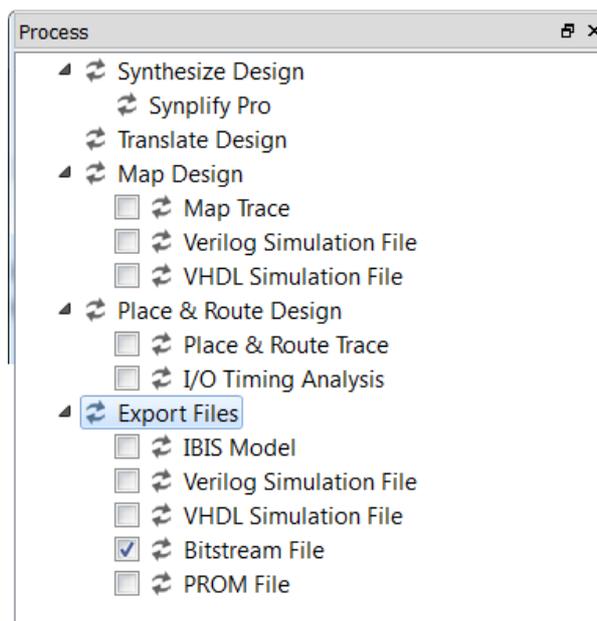
## Section 7 – Using Lattice Diamond (Continued)

### 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.

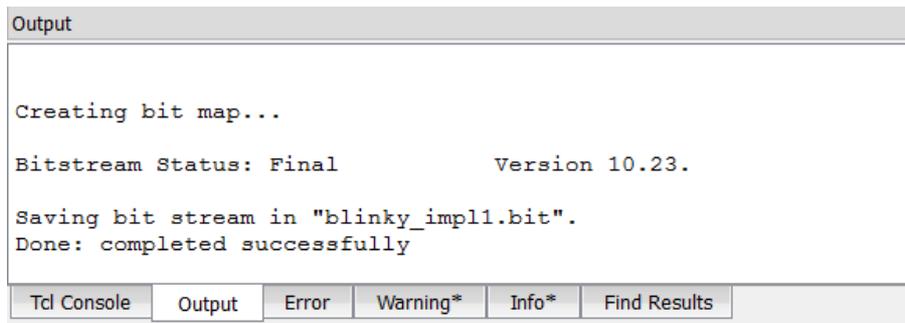


**Please Note:** While “Place & Route Trace” option may not be needed for this project, it is critical this report be generated, because it will be needed for more complex projects.

- 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

Creating bit map...

Bitstream Status: Final           Version 10.23.

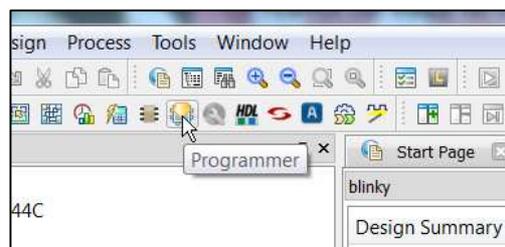
Saving bit stream in "blinky_impl1.bit".
Done: completed successfully
```

Tcl Console   Output   Error   Warning\*   Info\*   Find Results

### Section 7.5: Setting up Diamond programmer for use with FleaFPGA Ohm

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

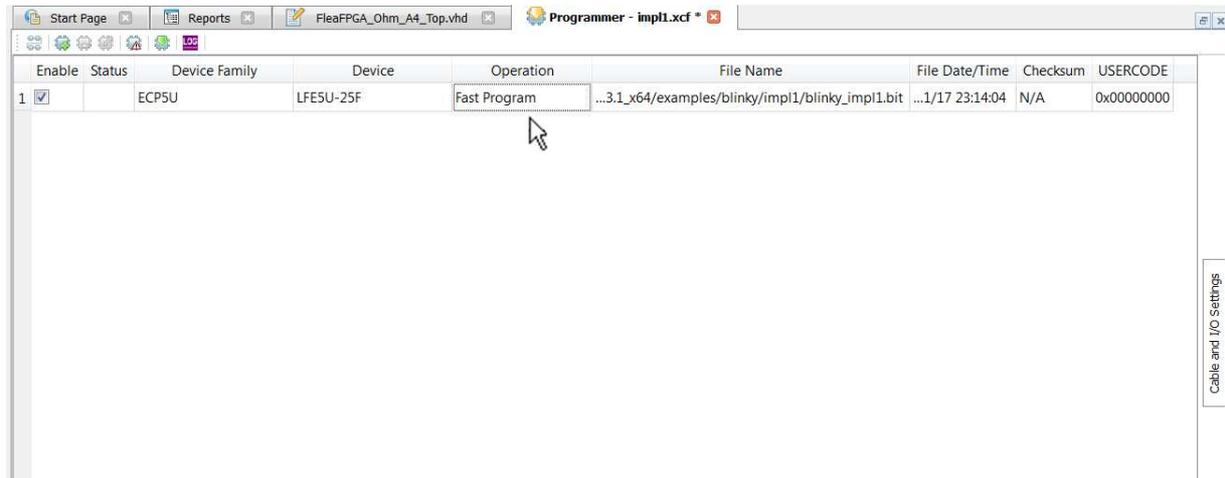
- i. Make Sure your FleaFPGA Ohm's USB slave port is connected to the PC (ideally via a hub, but not essential).
- ii. From the small icons list in the top menu, Click on the programmer icon as shown:



- 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:



Enable	Status	Device Family	Device	Operation	File Name	File Date/Time	Checksum	USERCODE
1	<input checked="" type="checkbox"/>	ECP5U	LFESU-25F	Fast Program	...3.1_x64/examples/blinky/impl1/blinky_impl1.bit	...1/17 23:14:04	N/A	0x00000000

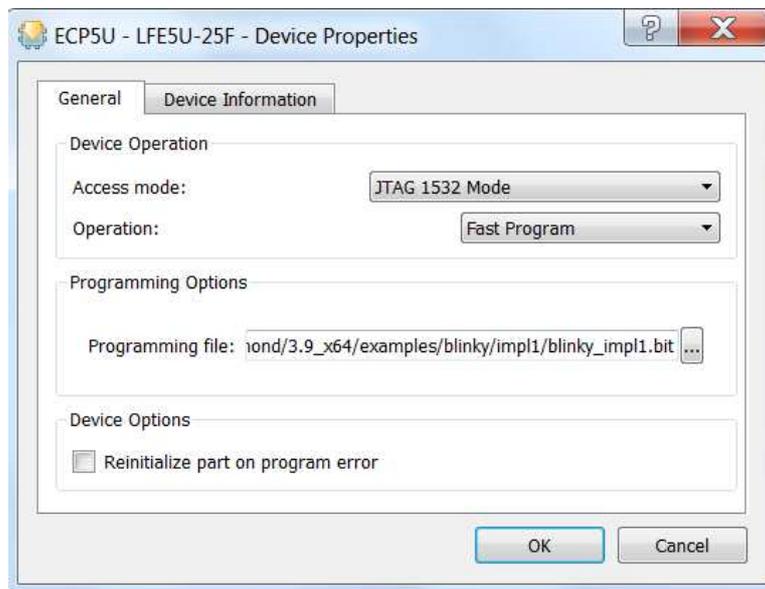
**NOTE:** Following pages cover the two main programming *modes* for FleaFPGA Ohm:

- 1.) **“Fast Program” - Uploading firmware into the FPGA configuration SRAM:**  
User firmware can be updated an infinite number of times with a relatively quick upload time of 15 seconds. However, user firmware uploaded in this mode is NOT permanently stored and will be lost upon power-off
- 2.) **“Flash Erase, Program” - Loading firmware into the FPGA configuration Flash:**  
User firmware may be permanently committed to the FPGA using this mode. However, programming time is considerably longer i.e. 40 seconds to over 5 minutes depending whether a USB hub is used or not (hence why a USB hub is recommended). 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)

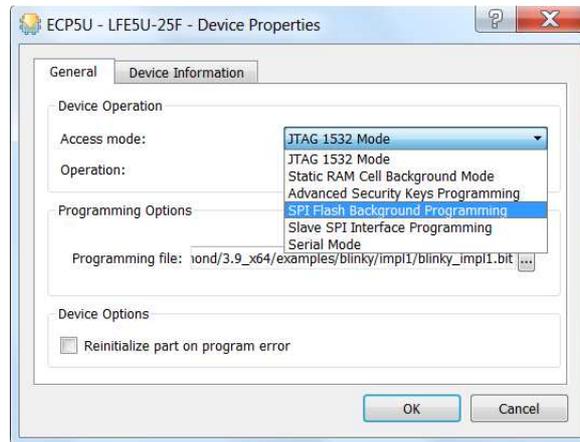
- v. **FPGA SRAM programming mode:** In the following child window, please select the correct 'Access Mode', 'Operation' and 'Programming File' options as shown below. Note that for either SRAM or Flash ROM programming, Lattice Diamond uses the \*.BIT extension for all FPGA firmware bit-files for uploading to the ECP5 FPGA.

**Warning!** Before you can successfully access the fast SRAM programming mode, you **MUST** first program the FleaFPGA Ohm with a valid bit-file to Flash ROM. Any attempt to program FleaFPGA Ohm's configuration SRAM will simply not work otherwise. If you have not yet programmed your FleaFPGA Ohm'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.

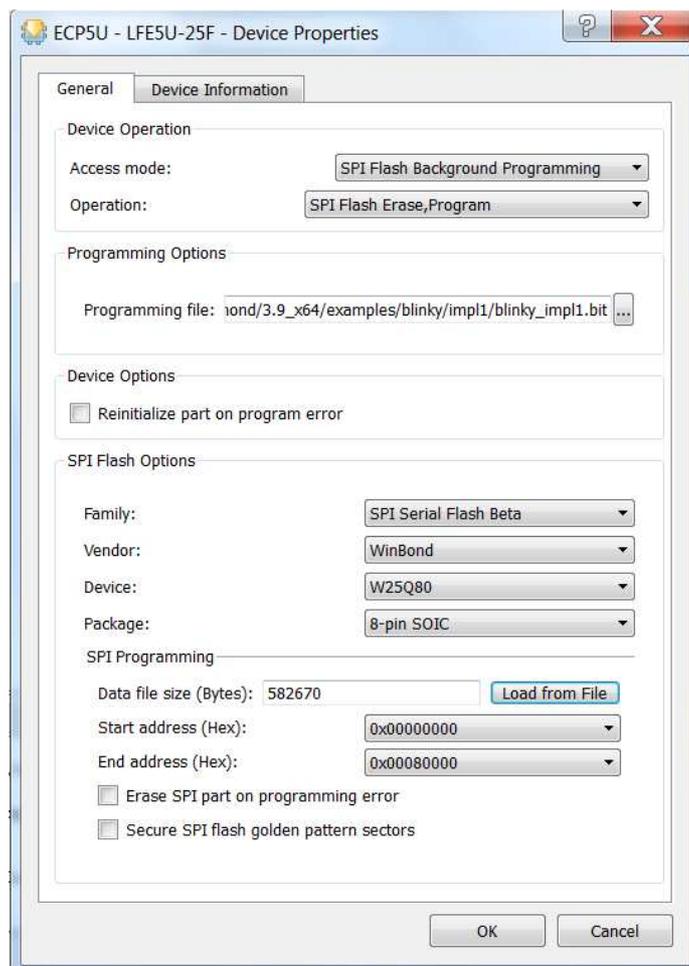


## Section 7 – Using Lattice Diamond (Continued)

- vi. **FPGA Flash ROM programming mode:** In the following child window, please select the 'SPI Flash Background Programming' in the Dropdown menu below...

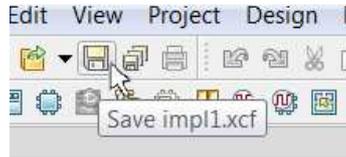


...and now the child window is expanded to include more options. Please select all options as shown in the following figure and then click 'Ok' to continue to the next step.

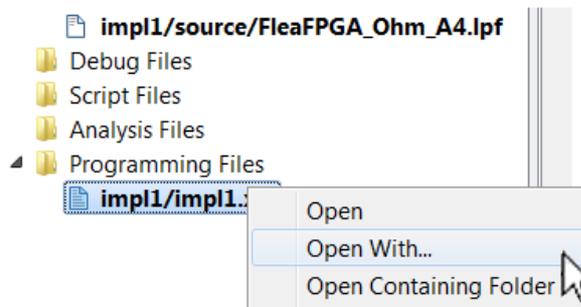


## Section 7 – Using Lattice Diamond (Continued)

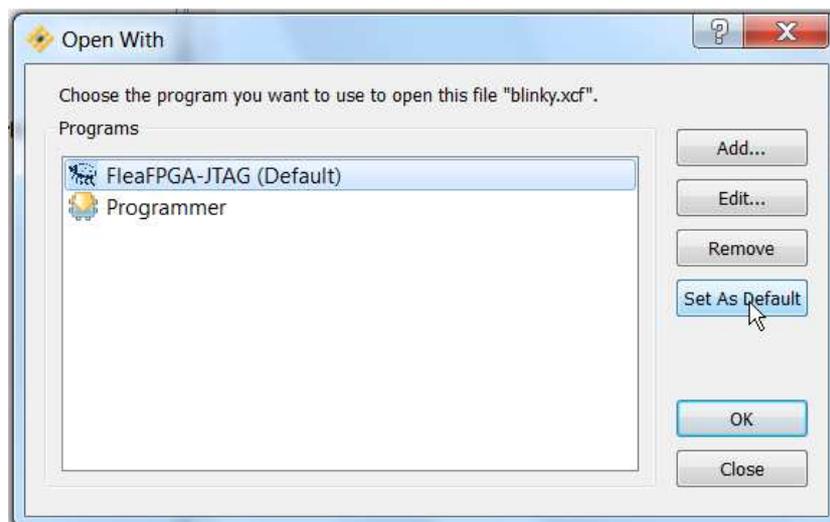
- vii. Save the current JTAG configuration by clicking on the related icon in the top menu as shown.



- 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 "impl1.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 Ohm 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 ‘STAT’ LED on the FleaFPGA Ohm 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 Ohm! 😊

**Please Note:** To initiate successive programming cycles to FleaFPGA Ohm via Lattice Diamond, simply double-click on “impl1.xcf” in the source tree to trigger a custom firmware file upload to FleaFPGA Ohm.

## 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 Ohm than blink an LED... ☺

### Exploring further HDL examples created for FleaFPGA Ohm:

Further HDL project examples are available for FleaFPGA Ohm. These can be found on [http://www.fleasystems.com/fleaFPGA\\_Ohm.html](http://www.fleasystems.com/fleaFPGA_Ohm.html). Example projects include:

- Simple Digital Oscilloscope [https://github.com/Basman74/Flea\\_scope](https://github.com/Basman74/Flea_scope)
- Sweet32 MRISC System <https://github.com/Basman74/Sweet32-CPU>
- Next186 System <https://github.com/Basman74/Next186>
- Minimig ECS System [https://github.com/Basman74/Minimig\\_ECS](https://github.com/Basman74/Minimig_ECS)
- More to follow! ☺

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:\lscd\diamond\3.9\_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.

### Developing your own HDL projects from scratch:

For those who are interested in developing their own HDL projects using FleaFPGA Ohm, 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 one or the other (or optionally, both ☺). 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](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 **FleaFPGA\_Ohm\_A5\_Top.vhd** and **FleaFPGA\_Ohm\_A5.Ipf** files. These files may be found on [www.fleasystems.com](http://www.fleasystems.com) in the FleaFPGA projects page.
- 4.) Better understanding of the Lattice ECP5 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 ECP5 User Guides available from [www.latticesemi.com](http://www.latticesemi.com).
- 5.) You may also find what others have previously done online interesting and useful. Some websites worth visiting are, but not limited to the following:
  - i. <http://www.fpga4fun.com/> Something for the beginners here ☺
  - ii. [http://hamsterworks.co.nz/mediawiki/index.php/Main\\_Page](http://hamsterworks.co.nz/mediawiki/index.php/Main_Page) Cool VHDL projects!
  - iii. <http://zipcpu.com/> includes an educational FPGA-related tech-blog and more

## Section 8 – Where to from here? (Continued)

**Attention!** Though you're free to view, modify and upload those HDL examples to your FleaFPGA Ohm, it is important that you try and understand the underlying theory as explained on the previous page.

### “What if I get stuck?”

Users with any questions relating to FleaFPGA Ohm may:

- 1.) Read the FAQ on the FleaFPGA Ohm:  
[http://www.fleasystems.com/fleaFPGA\\_Ohm.html](http://www.fleasystems.com/fleaFPGA_Ohm.html)
- 2.) Post a question to [fleasystems community forums](#) on Google groups.
- 3.) Send an email using the address provided in the contacts section of fleasystems.com.

## Section 9 – Notes

# Appendix A – FleaFPGA Ohm (Rev.A5) Specifications

## Hardware Summary:

- Lattice ECP5 24k LUT FPGA (Part# LFE5U-25F-6BG381C).
- 32MByte (16Mx16-bit) 166MHz SDRAM.
- 1MByte (8Mx1-bit) SPI configuration ROM.
- Digital Video output interface. Up to 720p or 1080p30 @ 24bit color supported
- Two “Combo” USB host ports, enabling the following options, depending on config:
  1. PS/2 Keyboard or Mouse port.
  2. USB 1.1 host port (Work in progress, to be implemented).
- USB ‘VIN’ port. Provides the following functions:
  1. Power feed for FleaFPGA Ohm via USB power from a suitable USB power adapter or PC host port.
  2. Slave serial port, including UART RTS handshake pin connection.
  3. JTAG interface for user programming of the ECP5 FPGA's configuration.
- MicroSD card slot
- 28 x 3.3V-level GPIO lines, organized as pin-compatible with a Raspberry Pi expansion hats **only** (See Appendix-C for more info regarding connection with a Raspberry Pi computer's GPIO header). In ‘standalone mode’, FleaFPGA Ohm provides the following enhanced functionality from the GPIO header:
  1. Four pins capable of Analog to Digital conversion (ADC). Sigma-delta method with variable bit-depth and/or sample rate. ADC Vref = 3.3V
  2. Twelve LVDS33 lane pairs, each capable of being clocked at up to 800Mbit/sec. per lane (including two configurable as global clock inputs).
- One RESET header (Marked as ‘RST’ on the board). Also possible to use as an additional user GPIO pin.
- One Green user LED (Marked as ‘STAT’ on the board).
- 25MHz onboard oscillator.

**Electrical Ratings:** Located on the next page.

# Appendix A – FleaFPGA Ohm (Rev.A5) Specifications (Continued)

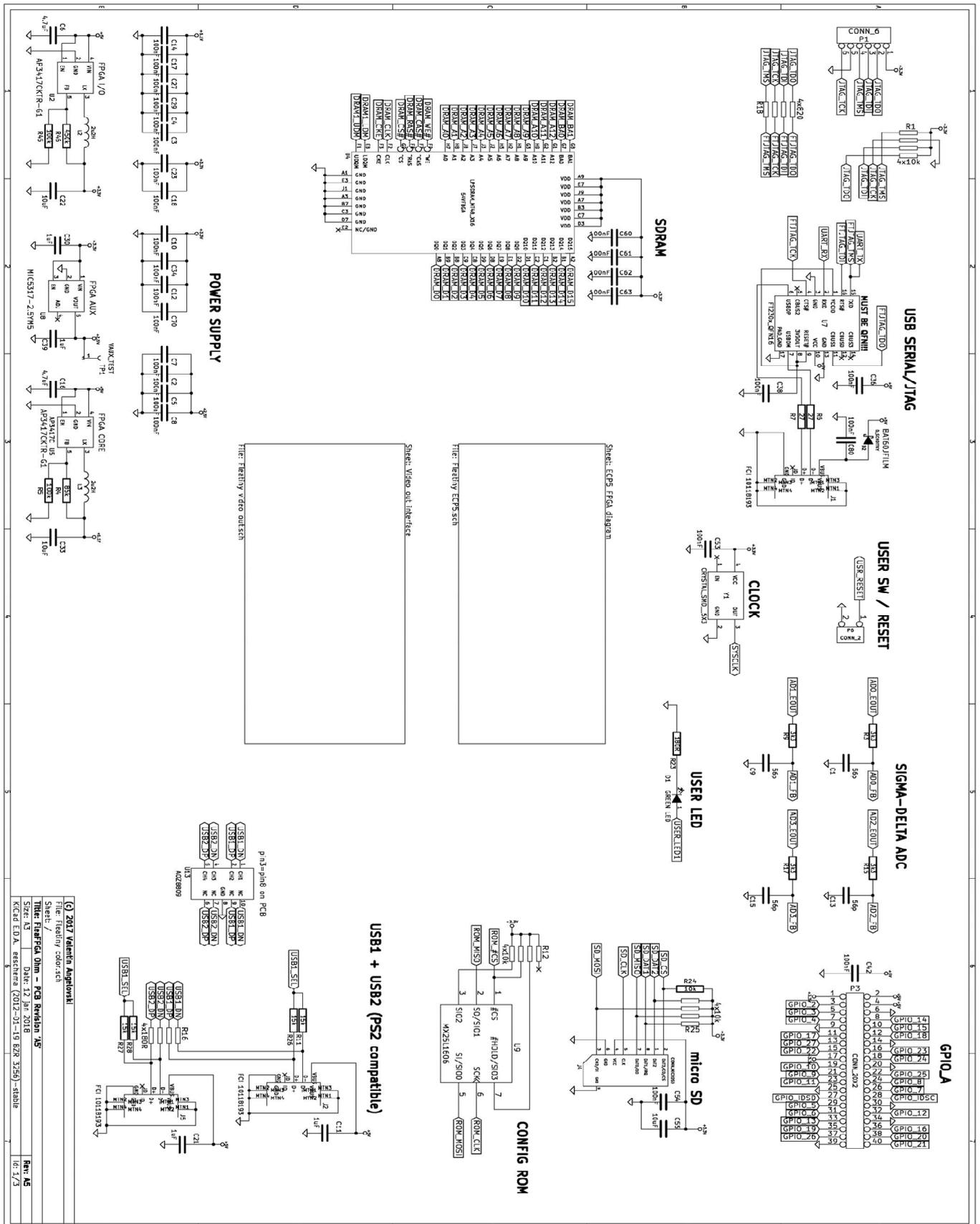
## Electrical Ratings:

- FleaFPGA Ohm board - typical current draw (assuming +5V supply): Dependent on the uploaded FPGA user-configuration as well as external peripherals (where connected). Typical current consumption figures shown below:
  - FPGA Config ROM erased and no input devices or SDcard: ~20mA
  - With Minimig ECS core loaded, SDcard and all ports connected: ~100mA
  - With Next186 core loaded, SDcard and all ports connected: ~150mA
- GPIO Header pins: 3.3V CMOS-compatible with adjustable current sink capability (typically set to 4mA for sink/source). Please refer to the Lattice ECP5 datasheet (Lattice doc reference: FPGA-DS-02012) for more detailed information about ECP5 GPIO pins).
- **Suggested maximum current draw from USB host ports: 150mA per port**
- **Suggested maximum current draw by an expansion hat via +3.3V rail: 250mA** (Stated rail has overcurrent protection and can output up to 1 Amp, but that is intended to cater for *all* onboard circuits and interfaces that need +3.3V including expansion hats).
- **Recommended maximum current draw by an expansion hat via the +5V rail: 500mA Do not exceed this figure!** (NOTE: If more than 500mA is needed by a user's custom expansion hat, users may alternatively feed a separate +5V supply into the FleaFPGA Ohm via the GPIO header. FleaFPGA Ohm would be powered by the expansion hat in that scenario).
- **FleaFPGA Ohm On-board power supply feed is NOT fuse protected!**

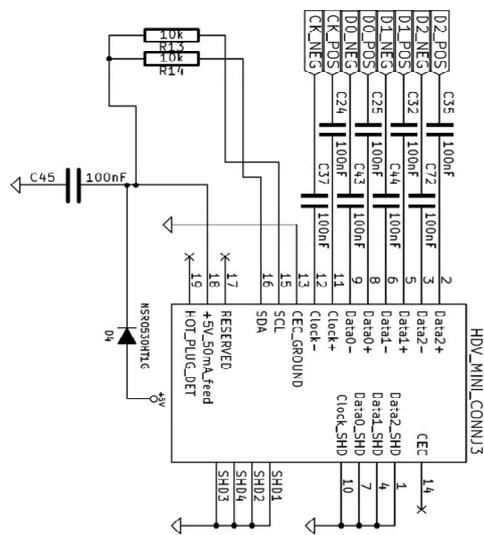
## Mechanical:

- PCB Dimensions: 69 x 63.5mm.
- Four 3.1mm diameter mounting holes, with hole-centers compatible to raspberry pi zero.
- System rated for 0 to +50deg.C temperature ambient.

# Appendix B – FleaFPGA Ohm Board Schematics

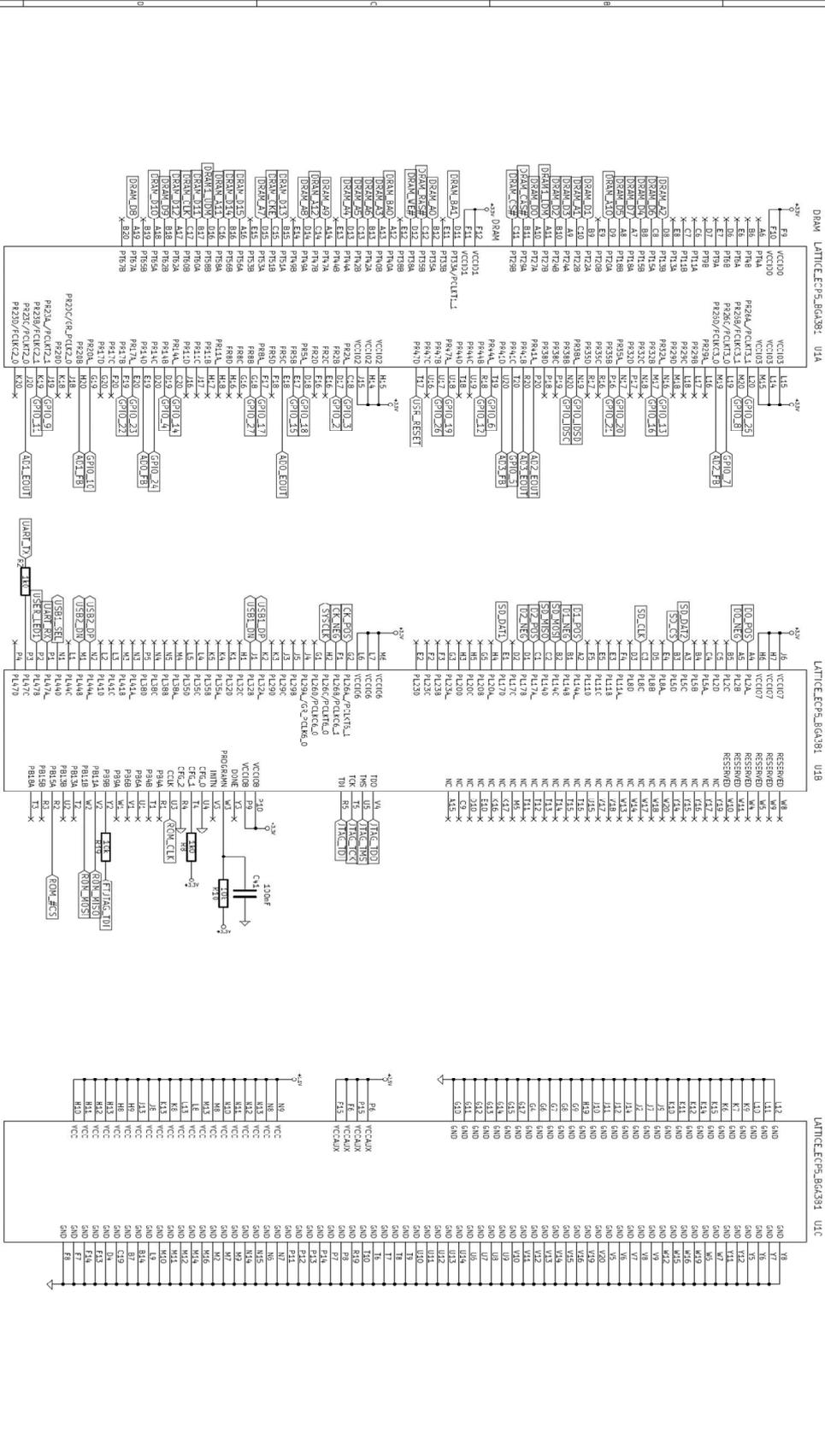


## Appendix B – Schematics (Continued)



© 2017 Valentin Angelovski	
File: Fleatiny video out.sch	
Sheet: /Video out Interface/	
Title: FleaFPGA Ohm – PCB Revision 'A5'	
Size: A4	Date: 12 an 2018
KICad E.D.A. eschema (2012-01-19 BZR 3256)–stable	
	Rev: A5 Id: 2/3

# Appendix B – Schematics (Continued)



© 2017 Valentin Anghelosi  
 File: Factory\_EC35.sch  
 Sheet: FPGAs FPGA diagram/  
 Title: FleaFPGA Ohm - PCB Revision 'A5'  
 Size: A3 Date: 12 Jan 2018  
 KICAD EDA, version: (2017-01-19 BZH 3250)-stable  
 Rev: A5  
 Id: 3/3

## Appendix C – Connecting FleaFPGA Ohm to a Raspberry Pi computer via GPIO header

Although not specifically designed to do so, it *is* possible to connect your FleaFPGA Ohm board to a Raspberry Pi computer via the GPIO header. **However**, users **must** consider the following precautions or limitations before they attempt using their FleaFPGA Ohm board in this manner:

- 1.) Users **MUST** ensure that the +3.3V supply pins (i.e. pins #1 and #17 on the GPIO header) are **ELECTRICALLY ISOLATED (i.e. not connected at all)** between your FleaFPGA Ohm and Raspberry Pi boards via the GPIO header. **Failure to do so may result in the +3.3V supplies on both boards coming into conflict, forcing up current draw and increasing risk of damage to one or both boards considerably!**
- 2.) For any custom applications where FleaFPGA Ohm is being used to interact directly with a Raspberry Pi computer, users must ensure they use a robust or recommended method for the transfer of data between the Raspberry Pi computer and FleaFPGA Ohm. **Failure to do so may result GPIO data bus contention between boards, once again forcing up current draw and increasing risk of damage to one or both boards considerably!**
- 3.) It is recommended that users power their Raspberry Pi computer from its micro-USB power port, in cases where FleaFPGA Ohm is intended for use as an expansion hat with Raspberry Pi as its master. It is not recommended to power your Raspberry Pi computer purely from the 'USB Vin' port on FleaFPGA Ohm while both are connected together via GPIO (Users risk exceeding the specifications on either board with potentially negative and/or unpredictable consequences).
- 4.) Custom applications of this kind are fairly non-trivial in nature and therefore fall outside the scope of this Quick start guide. Purpose of this appendix section merely aims to provide some clarity around the potential pitfalls and/or restrictions around using FleaFPGA Ohm in this manner.