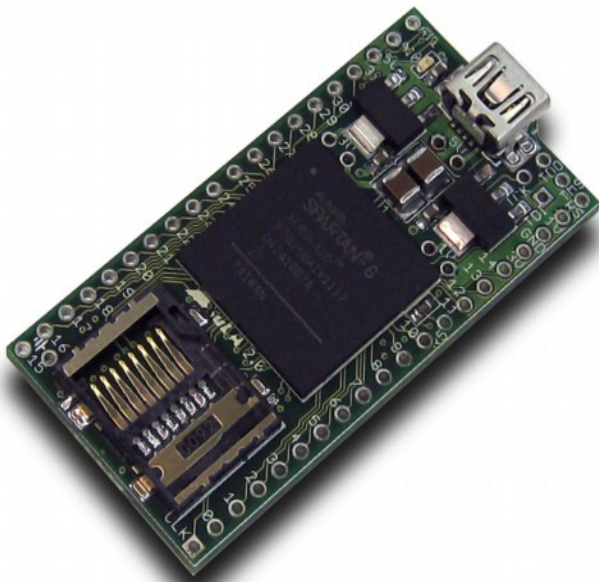


XuLA2 Manual

*How to install, test and use
your new FPGA board*



XESS is disclosing this Document and Intellectual Property (hereinafter “the Design”) to you for use in the development of designs to operate on, or interface with XESS hardware devices. XESS expressly disclaims any liability arising out of the application or use of the Design. XESS reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of XESS. XESS assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. XESS will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED “AS IS” WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XESS, OR ITS AGENTS OR EMPLOYEES. XESS MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XESS BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XESS IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XESS HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XESS WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems (“High-Risk Applications”). XESS specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

© 2012 XESS, Inc. All rights reserved. XESS, the XESS logo, and other designated brands included herein are trademarks of XESS Corporation. All other trademarks are the property of their respective owners.



This document is licensed under the Attribution-ShareAlike 3.0 Unported license, available at <http://creativecommons.org/licenses/by-sa/3.0/>.

XuLA2 Manual MAN007 (V1.2) December 28, 2012

The following table shows the revision history for this document.

Date	Version	Revision
07/25/2012	0.1	Pre-release for the XuLA2.
08/1/2012	0.2	Corrected "Spartan-3A" to "Spartan-6".
08/14/2012	1.0	Changed images for XuLA2 production board. Added instructions for setting FPGA bitstream configuration rate.
08/20/2012	1.1	Added notes about connections from the prototyping header pins to FPGA global clock pins in the Pin Connections appendix .
12/28/2012	1.2	Expanded explanation of how to use a XILINX programming cable with the XuLA2.

Table of Contents

- C.1 Preliminaries..... 1**
 - Getting Help!..... 1
 - Take Notice!..... 1
- C.2 Installation..... 2**
 - Installing the XSTOOLS Utilities and Documentation..... 2
 - Connecting Your XuLA2 to a PC..... 2
 - Testing Your XuLA2..... 3
 - Setting the Jumpers on Your XuLA2..... 3
 - Applying Power to Your XuLA2..... 4
 - Applying Power Through the USB Port..... 4
 - Applying Power Through the Prototyping Header..... 4
 - Inserting the XuLA2 into a Breadboard..... 5
- C.3 Programming..... 6**
 - Generating Bitstreams for the FPGA..... 6
 - Downloading Bitstreams into the FPGA..... 9
 - Downloading Using GXSLOAD..... 9
 - Downloading Using a XILINX or Third-Party JTAG Cable..... 11
 - Storing Non-Volatile Bitstreams in the Serial Configuration Flash..... 12
 - Transferring Data to/from the SDRAM..... 14
- C.4 Programmer Models..... 16**
 - XuLA2 Components..... 16
 - FPGA..... 17
 - Microcontroller..... 17
 - SDRAM..... 18
 - SPI Flash and microSD Card..... 18
 - Prototyping Header..... 19
 - 5V Tolerance Issues..... 20
 - Auxiliary JTAG Header..... 21
- A.1 Pin Connections..... 22**
- A.2 Schematic..... 25**

C.1 Preliminaries

Here's some helpful information before getting started.

Getting Help!

Here are some places to get help if you encounter problems:

- If you can't get the XuLA2 hardware to work, send an e-mail message describing your problem to help@xess.com or submit a problem report at <http://www.xess.com/help.php>. Our web site also has:
 - answers to frequently-asked-questions,
 - example designs, application notes and tutorials for our FPGA boards,
 - a place to sign-up for our email forum where you can post questions to others.
- If you can't get your XILINX ISE *WebPACK* software tools installed properly, check their web site at <http://www.xilinx.com/support/>.
- If you need help using the XILINX ISE *WebPACK* software to create FPGA designs, then check out this [tutorial](#).

Take Notice!

- The XuLA2 is not 5V-tolerant. **Do not connect 5V logic signals to the prototyping header.**
- The XuLA2 printed circuit board (PCB) is manufactured such that the terminals of the jumpers labeled "5V", "3.3V" and "1.2V" are connected on the underside of the PCB by short wiring traces. **You must cut these traces if you want to open the jumper connections.**
- Even if you have experience with the XILINX ISE *WebPACK* software, please read this [section on setting the bitstream generation options for the XuLA2](#).

C.2 Installation

Installing the XSTOOLS Utilities and Documentation

XILINX currently provides the free ISE® *WebPACK*™ software for programming many of their small and mid-size FPGAs and CPLDs. You can download the most current version of ISE *WebPACK* from www.xilinx.com.

In addition, XESS provides the XSTOOLS utilities for interfacing a PC to your XuLA2. These utilities (along with manuals, design examples and tutorials) are installed automatically when you insert the XSTOOLS CD into your PC. If not, then manually run the SETUP.EXE installation program on the CD. You can also download the XSTOOLS installer from www.xess.com.

Connecting Your XuLA2 to a PC

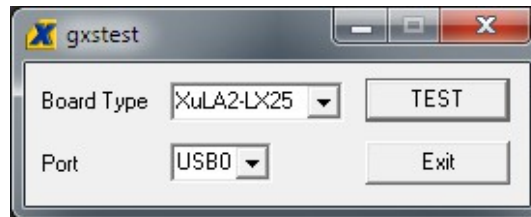
The XuLA2 is a USB peripheral that you can attach to any USB 1.1 or 2.0 port through a cable with a five-pin mini-B connector such as this one:



The LED on your XuLA2 will light up as soon as it establishes a connection with the PC.

Testing Your XuLA2

Once your XuLA2 is connected to a USB port, you can test it by double-clicking the GXSTEST icon placed on your PC desktop during the XSTOOLS installation. This brings up the window shown below.



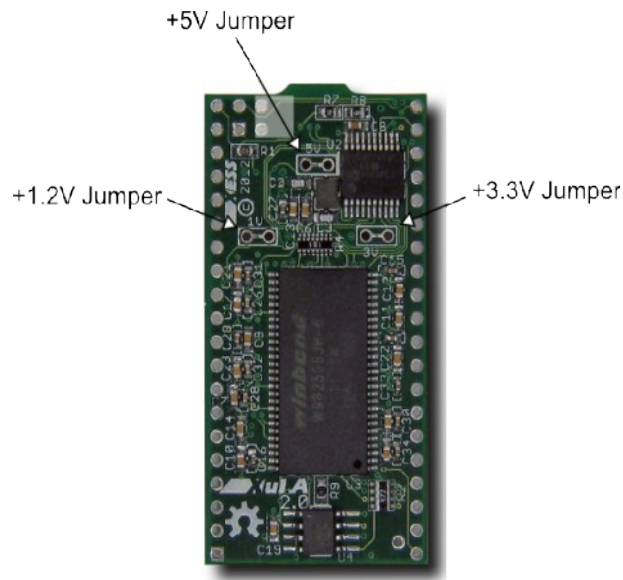
Next, select the type of board you are testing and the port that it's attached to. Then click on the TEST button. GXSTEST will configure the FPGA on your XuLA2 to perform a test procedure. Within a few seconds, a status window will appear informing you of the success or failure of the test.

If your XuLA2 fails the test, you will be shown a checklist of common causes for failure. If none of these applies to your situation, then [contact XESS Corp](#) for further assistance.

Setting the Jumpers on Your XuLA2

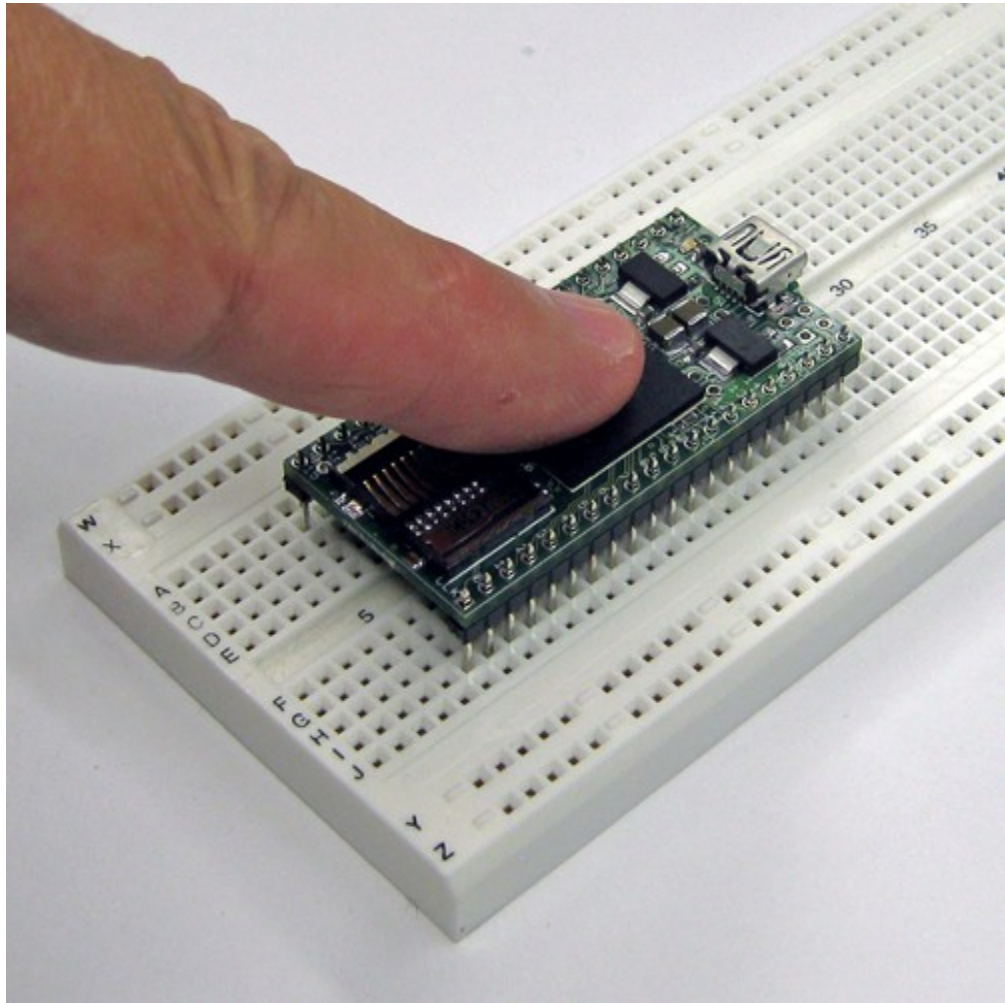
The XuLA2 has three jumpers labeled "5V", "3.3V" and "1.2V" that are used to configure how the board receives power. In their factory-original configuration, the jumpers are unpopulated but the terminals of each jumper are connected on the underside of the PCB by short wiring traces. **You must cut these traces if you wish to open the jumper connections.** You only need to do this if you are powering your XuLA2 through its prototyping header (not through the USB cable) as described [here](#).

The locations of the shorting traces on the underside of the XuLA2 PCB are shown below.



Inserting the XuLA2 into a Breadboard

In its factory-original configuration, the XuLA2's prototyping header is empty. If desired, you can solder in a pair of twenty-pin headers and then insert the XuLA2 into a standard solderless breadboard as shown below.



The XuLA2 PCB will accept common headers with 0.025"-thick pins at 0.1" spacing, but you may find it difficult to remove the XuLA2 given how tightly the breadboard grips the pins. A header with thinner pins (such as the Aries 20-0600-20) is a better choice.

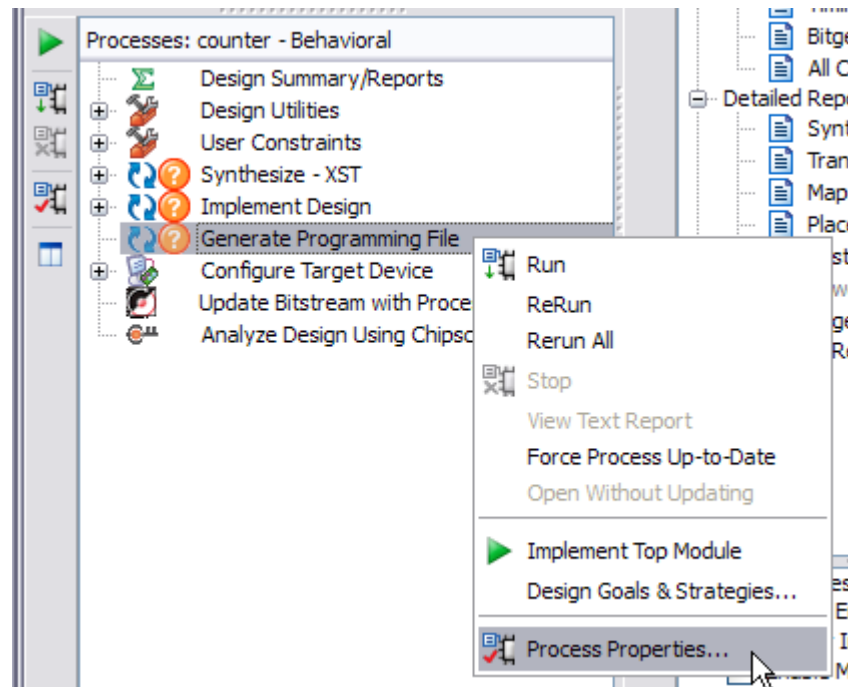
C.3 Programming

This chapter will show you how to download logic designs into the FPGA of your XuLA2 and how to transfer data between the PC and the SDRAM and Flash memories on the board.

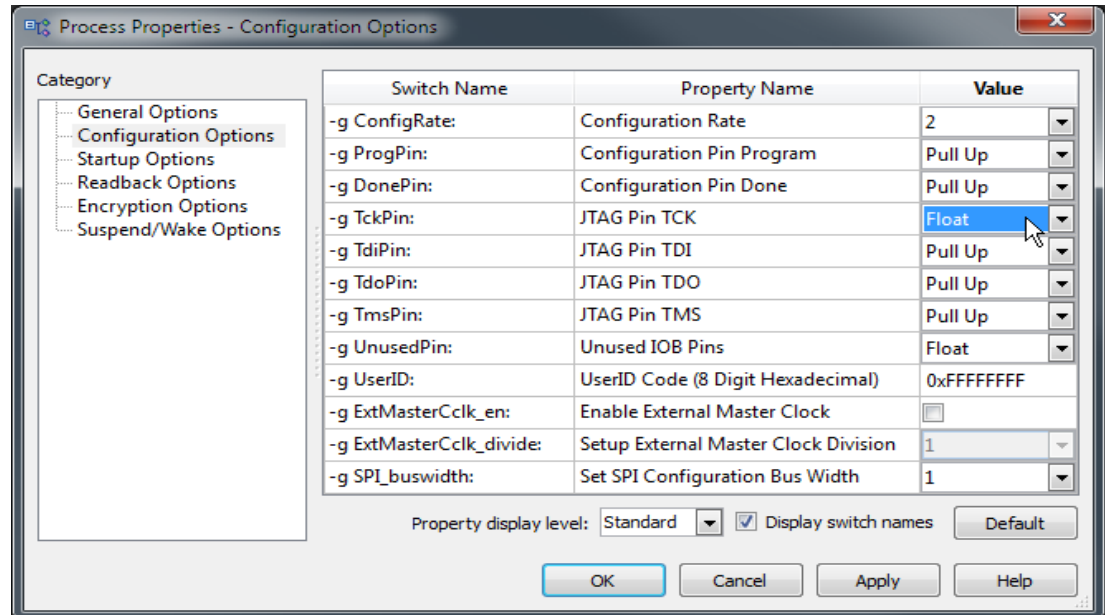
Generating Bitstreams for the FPGA

Before you can download a logic design to the FPGA on your XuLA2, you need to generate a bitstream (i.e., a .BIT file) with XILINX ISE *WebPACK*. Steps for doing this are given in the XILINX documentation and this [XESS tutorial](#), but there are several details that you have to be aware of when generating the bitstream.

After creating your logic design in ISE *WebPACK*, right-click on the **Generate Programming File** item in the **Process** window and select **Process Properties...** from the context menu as shown below.



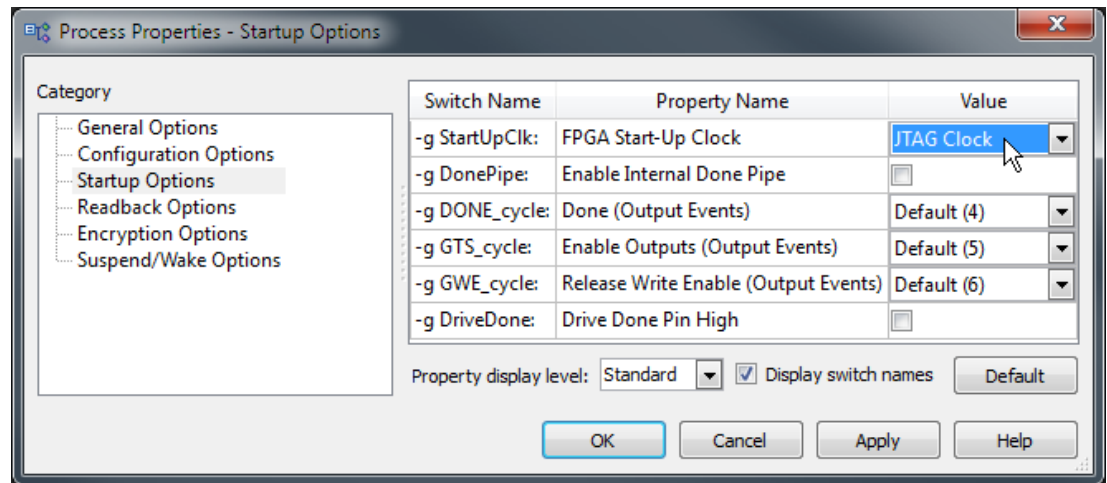
When the **Process Properties** window appears, select the **Configuration Options** category. Make sure the **Configuration Pin Done** property is set to the value **Pull Up**. This is necessary for the microcontroller on the XuLA2 to detect when the FPGA has successfully been configured by the bitstream. Also, set the **JTAG Pin TCK** property to either **Float** or **Pull Down** so it does not conflict with the pull-down resistor on the XuLA2. Finally, set the **Unused IOB Pins** property **Float** or **Pull Up**.



If you're planning on storing the bitstream into the serial configuration flash, you will also want to set the **Configuration Rate** to 10 MHz or more. That will ensure the 5 Mbit bitstream will load into the FPGA in 1/2-second or less. But if you're only going to download the bitstream to the FPGA via the USB link, then the **Configuration Rate** setting can be ignored.

Finally, select the **Startup Options** category and set the **FPGA Start-Up Clock** to one of the following values:

FPGA Start-Up Clock Settings	
JTAG Clock	Use this setting when your bitstream will be downloaded directly into the XuLA2's FPGA through the USB cable.
CCLK	Use this setting when your bitstream will be downloaded into the XuLA2's serial configuration Flash for eventual transfer to the FPGA.



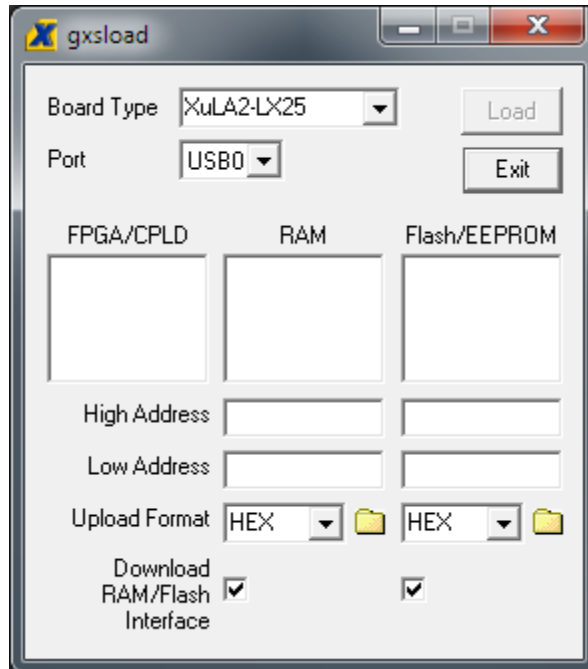
Then click **OK** to finalize your process properties. Now you're ready to generate a bitstream for the XuLA2!

Downloading Bitstreams into the FPGA

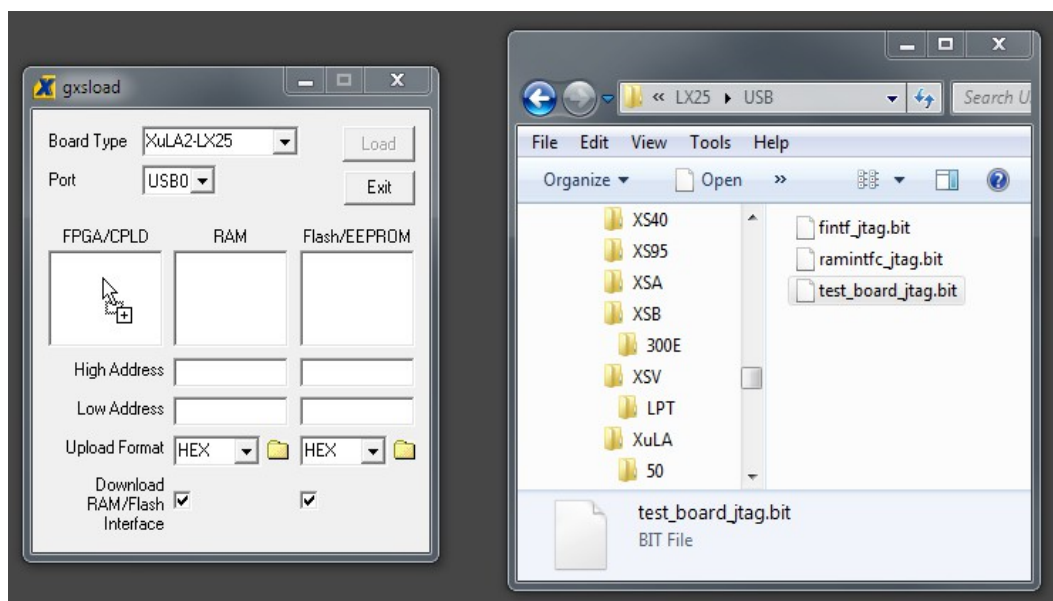
Downloading Using GXSLD

As you work on a logic design, you will usually download the bitstream from the PC to the XuLA2 to test your modifications. The GXSLD utility is used for downloading bitstreams over the USB link.

Start GXSLD by double-clicking the icon placed on the desktop during the XSTOOLS installation. This brings up the window shown below.

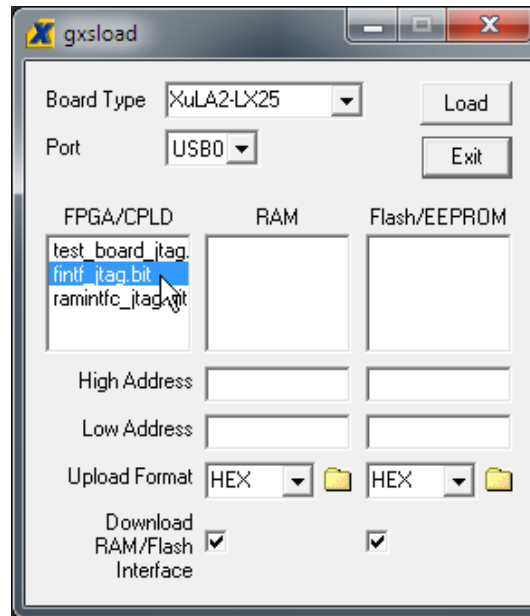


Now you can download bitstream files to the FPGA or CPLD simply by dragging them from their folder and dropping them into the **FPGA/CPLD** pane as shown below.

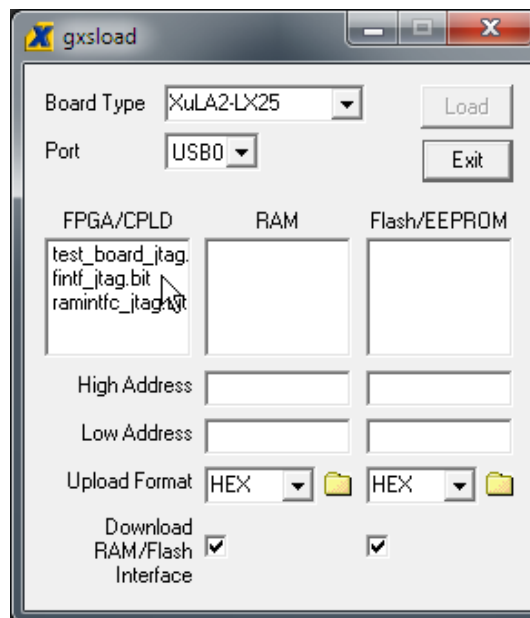


Once you drop the file, the highlighted file name appears in the **FPGA/CPLD** pane and the **Load** button is enabled. Clicking on the **Load** button will begin sending the bitstream to the XuLA2 through the USB cable. GXSLD will reject any non-downloadable files (ones with a suffix other than .BIT). During the downloading process, GXSLD will display the name of the bitstream file and the progress of the current download. The LED on the XuLA2 will blink as the bitstream is transferred.

You can drag & drop multiple files into the **FPGA/CPLD** pane. Clicking your mouse on a file name will highlight the name and select it for downloading. Only one file at a time can be selected for downloading.

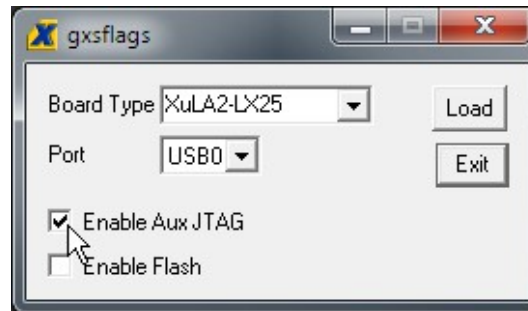


Double-clicking the highlighted file will deselect it so no file will be downloaded. Doing this disables the **Load** button.



Downloading Using a XILINX or Third-Party JTAG Cable

As an alternative to using GXSLD and a simple USB cable, you can use a XILINX or third-party JTAG cable to configure the FPGA on the XuLA2. But first, you have to set a non-volatile flag to give priority to the auxiliary JTAG header using the GXSFLAGS utility. Connect the XuLA to your PC with a USB cable and then double-click the GXSFLAGS icon so the following window appears.

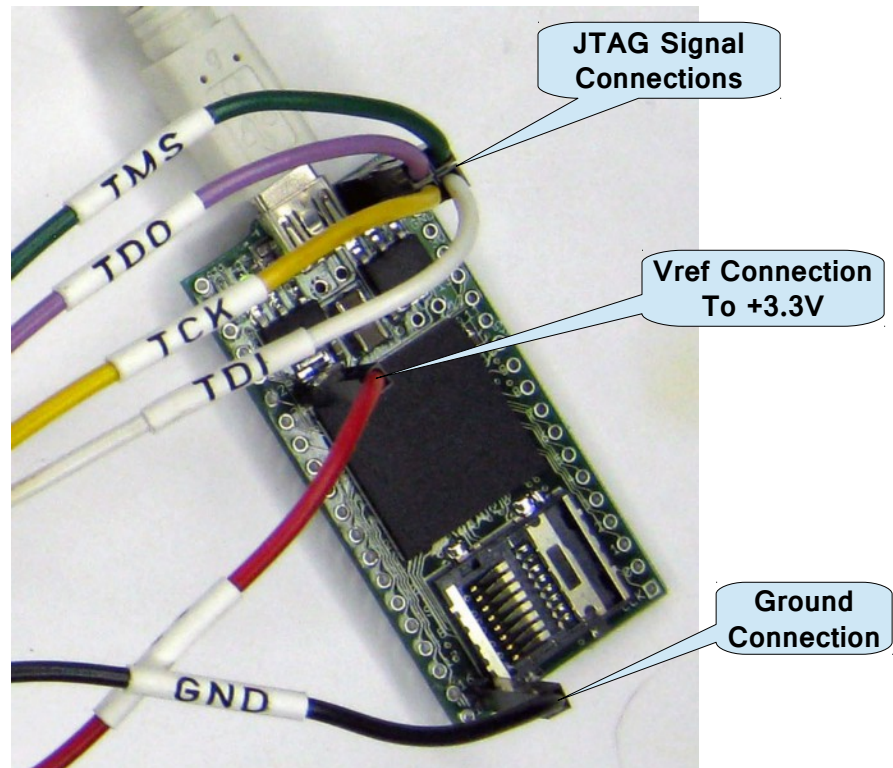


Click on the **Enable Aux JTAG** checkbox and then click on the **Load** button to set the flag. Note that the flag only needs to be set once to support the JTAG cable because it retains its value even when power is removed from the XuLA2. (Be aware that enabling the auxiliary JTAG header disables the functions of the other XSTOOLS utilities such as GXSTEST and GXSLD. To restore their functions, you'll have to use GXSFLAGS again to disable the auxiliary JTAG header.)

Now attach your JTAG cable to the auxiliary JTAG header as shown below. (The USB cable remains attached only to provide the XuLA2 with power. You can detach it if the board is getting power from another source, such as through its prototyping header.)



Here's a close-up of the connections from the XILINX cable to the XuLA2:



Now start the ISE *WebPACK* iMPACT tool and follow the XILINX instructions to download bitstreams to the FPGA in boundary-scan mode.

Storing Non-Volatile Bitstreams in the Serial Configuration Flash

The FPGA on the XuLA2 stores its configuration in an on-chip SRAM which is erased whenever power is removed. Once you complete a design, you may want to store the bitstream in the serial configuration Flash on the XuLA2. After that, the FPGA will configure itself from the Flash each time power is applied.

Loading a bitstream into the Flash is easy: just drag the .BIT file into the **Flash/EEPROM** pane of GXSLD and click on the **Load** button. This activates the following sequence of events:


1. The FPGA on the XuLA2 is configured with an interface to transfer data from the USB port to the Flash.
2. The entire Flash is erased.
3. The contents of the .BIT file are downloaded into the Flash through the USB port.

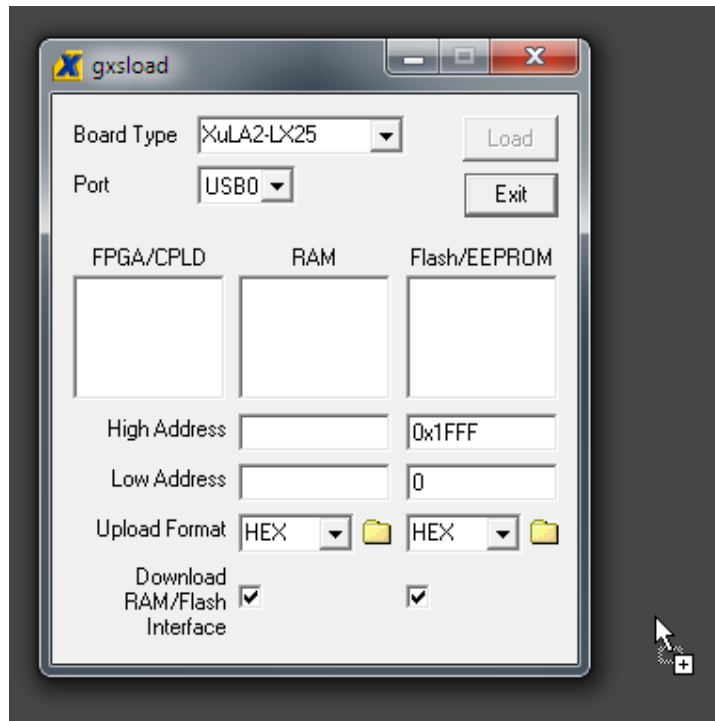
Once the Flash download is completed, the FPGA will be configured with the stored bitstream whenever power is applied to the XuLA2. *(Make sure that you selected [CCLK](#) as the start-up clock and set the [configuration rate to 10 MHz or more](#) when you generated the bitstream or else the FPGA will fail to configure from the Flash.)*

You can also use the XILINX iMPACT software to convert FPGA bitstreams into .MCS or .EXO data files. These file types can be programmed into the XuLA2's Flash using GXSLD in the same way as with a .BIT file.

Multiple files can be stored in the Flash device just by dragging them into the

Flash/EEPROM area, highlighting the files to be downloaded and clicking the **Load** button. (Note that anything previously stored in the Flash will be erased by each new download.) This is useful if you need to store information in the Flash in addition to the FPGA bitstream. Files are selected and de-selected for downloading just by clicking on their names in the Flash/EEPROM area. *The address ranges of the data in each file should not overlap or this will corrupt the data stored in the Flash device!*

You can also examine the contents of the Flash by uploading it to the PC. To upload data from an address range in the Flash, type the upper and lower bounds of the range into the **High Address** and **Low Address** fields located below the **Flash/EEPROM** pane, and select the format for the uploaded data from the **Upload Format** pulldown list. Then click on the file icon () and drag & drop it into any folder.



This activates the following sequence of steps:

1. The FPGA on the XuLA2 is configured with an interface between the Flash device and the PC USB port.
2. The Flash data between the high and low addresses (inclusive) is uploaded through the USB port.
3. The uploaded data is stored in a file named FLSHUPLD with an extension that reflects the selected upload file format.

The uploaded data can be stored in the following formats:


- MCS:** Intel hexadecimal file format.
- HEX:** Identical to MCS format.
- EXO-16:** Motorola S-record format with 16-bit addresses (suitable for 64 KByte uploads only).
- EXO-24:** Motorola S-record format with 24-bit addresses.
- EXO-32:** Motorola S-record format with 32-bit addresses.
- XESS-16:** XESS hexadecimal format with 16-bit addresses. (This is a simplified file format that does not use checksums.)

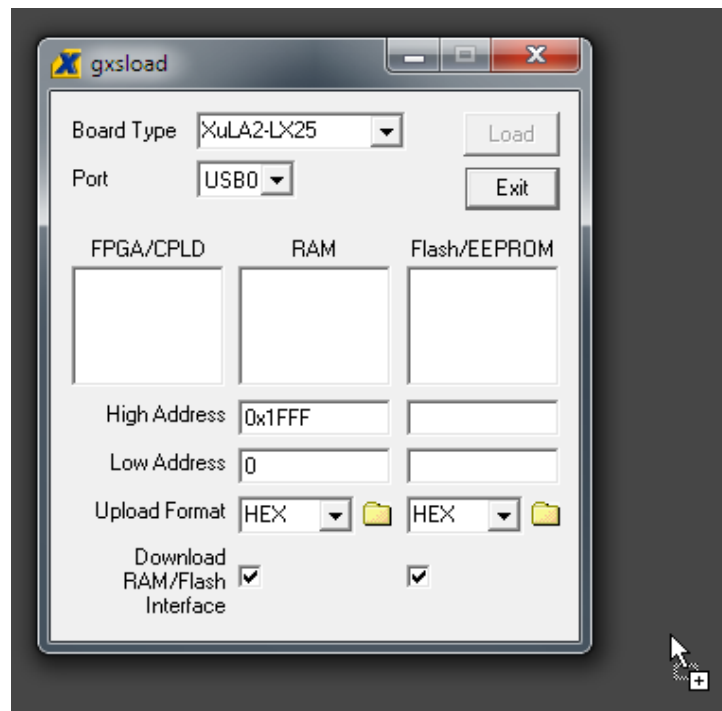
- XESS-24:** XESS hexadecimal format with 24-bit addresses.
XESS-32: XESS hexadecimal format with 32-bit addresses.

Transferring Data to/from the SDRAM

The XuLA2 contains a synchronous DRAM (SDRAM) whose contents can be downloaded and uploaded by GXLOAD. This is useful for initializing the SDRAM with data for use by the FPGA and then reading the SDRAM contents after the FPGA has operated upon it. The SDRAM is loaded with data by dragging & dropping one or more .EXO, .MCS, .HEX, and/or .XES files into the **RAM** pane of the **gxload** window and then clicking on the **Load** button. This activates the following sequence of steps:

1. The FPGA is configured with an interface to transfer data between the SDRAM device and the USB port.
2. The contents of the .EXO, .MCS, .HEX or .XES file are downloaded into the SDRAM through the USB port.

You can also examine the contents of the SDRAM device by uploading it to the PC. To upload data from an address range in the SDRAM, type the upper and lower bounds of the range into the **High Address** and **Low Address** fields below the **RAM** pane, and select the format for the uploaded data from the **Upload Format** pulldown list. Then click on the file icon () and drag & drop it into any folder.



This activates the following sequence of steps:

1. The FPGA is configured with an interface between the SDRAM device and the PC USB port.
2. The SDRAM data between the high and low addresses (inclusive) is uploaded through the USB port.
3. The uploaded data is stored in a file named RAMUPLD with an extension that reflects the selected upload file format.

The 16-bit data words in the SDRAM are mapped into the eight-bit data format of the .HEX, .MCS, .EXO and .XES files using a Big Endian style. That is, the 16-bit word at location N in the SDRAM is stored in the eight-bit file with the upper eight bits at address $2N$ and the lower eight bits at address $2N+1$. This byte-ordering applies for both RAM uploads and downloads.

C.4 Programmer Models

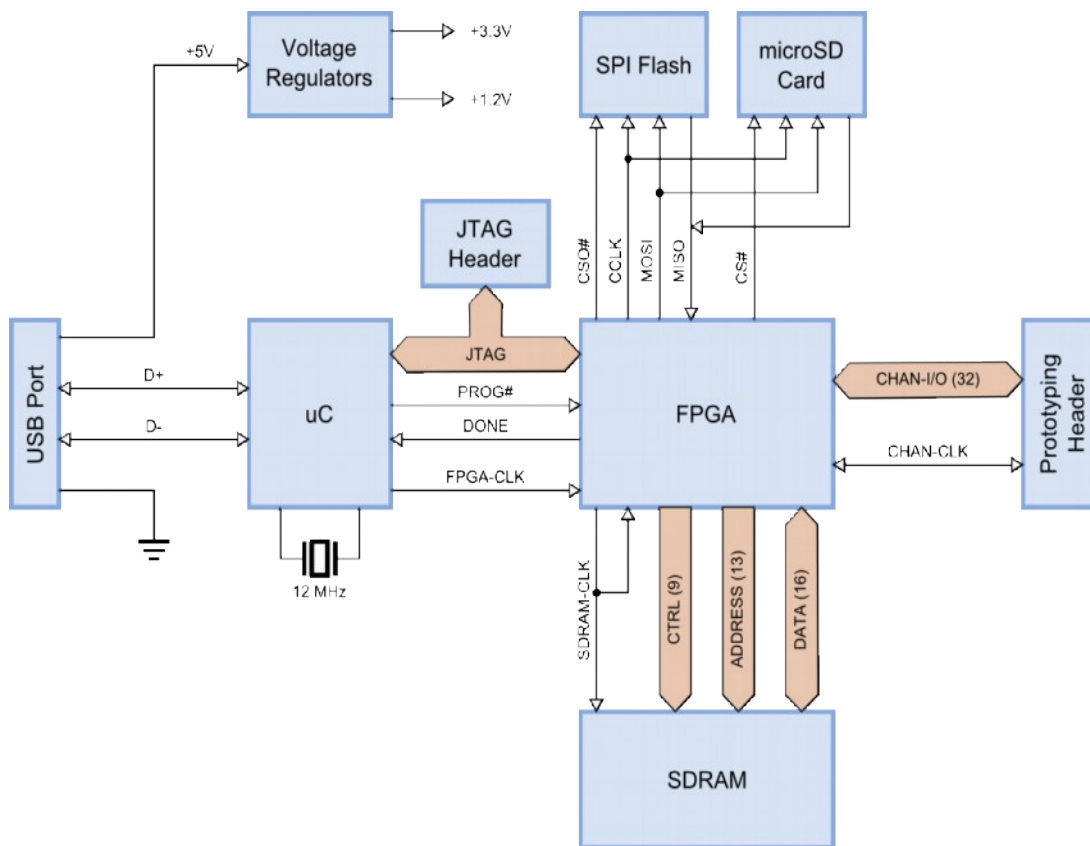
This section describes the various sections of the XuLA2 and shows how the FPGA I/O is connected to the rest of the circuitry. The schematics which follow are less detailed so as to simplify the descriptions. For more information, you can find a table of [pin connections](#) and detailed [schematics](#) at the end of this manual.

XuLA2 Components

The XuLA2 contains the following major components:

- FPGA:** This is the field programmable gate array.
- Microcontroller:** The microcontroller (uC) handles initialization, clock generation and USB-to-JTAG communications.
- SDRAM:** A 256-Mbit SDRAM provides volatile data storage accessible by the FPGA.
- SPI Flash:** An 8-Mbit serial Flash device provides non-volatile storage for FPGA configuration bitstreams and user data.
- MicroSD Card:** A Secure Digital memory card provides non-volatile storage for user data.
- Prototyping Header:** FPGA I/O pins, several uC pins and power/GND pins are connected to this 40-pin header that is meant to mate with solderless breadboards or other devices with 0.1"-spacing sockets.
- Aux. JTAG Header:** This header provides direct access to the FPGA's JTAG pins.

The interconnection of these components is shown in the following block diagram.



FPGA

The programmable logic device on the XuLA2 is a [XILINX XC6SLX25 Spartan 6 FPGA](#) in a 256-ball BGA package (FT256).

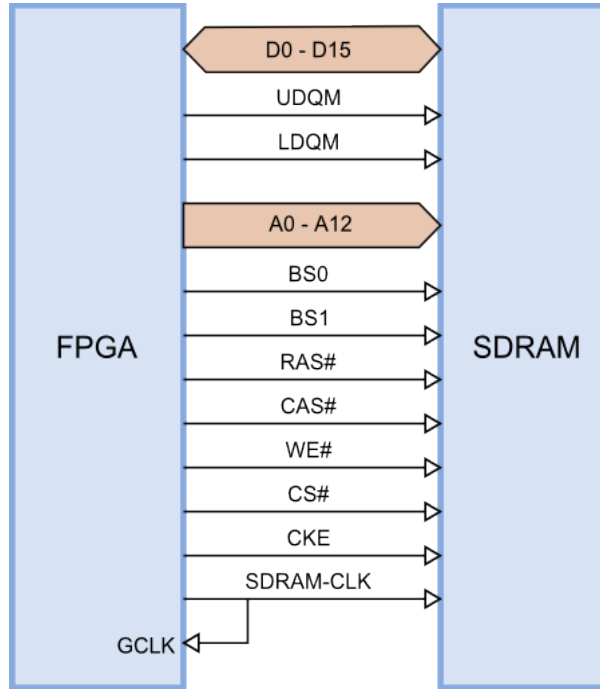
Microcontroller

The XuLA2 uses a [Microchip 18F14K50 PIC](#) to perform the following functions:

- Reset & initialization:** Upon power-up or assertion of the reset, the microcontroller initiates the configuration of the FPGA from the SPI Flash and holds the FPGA in its cleared state if the configuration fails. It instantiates its USB endpoints and participates in the USB enumeration process.
- Clock generation:** The microcontroller uses its pulse-width modulation (PWM) circuitry to generate a 12 MHz square-wave that enters one of the FPGA's global clock inputs.
- USB-to-JTAG communication:** The microcontroller accepts configuration bitstreams and data as packets over the USB link and transforms these into a sequence of transitions upon the FPGA's JTAG pins. It also receives data from the FPGA through the JTAG port which it bundles into packets for return through the USB link.

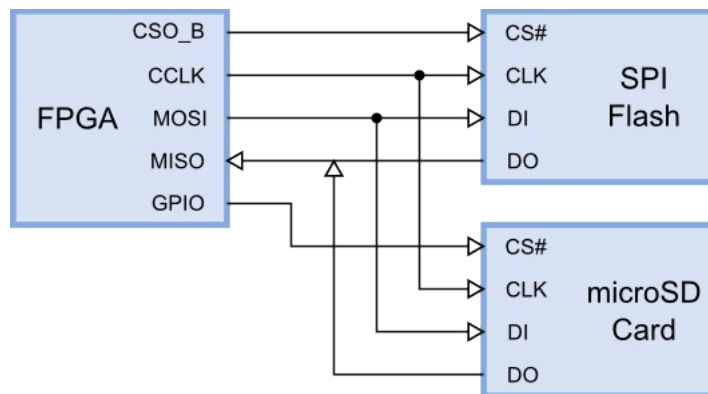
SDRAM

The XuLA2 incorporates a 16M x 16 SDRAM ([Winbond W9825G6JH](#)) that connects to the FPGA as shown below. To compensate for circuit delays, the clock signal to the SDRAM is re-routed back to a global clock input so the FPGA can synchronize itself with the SDRAM.



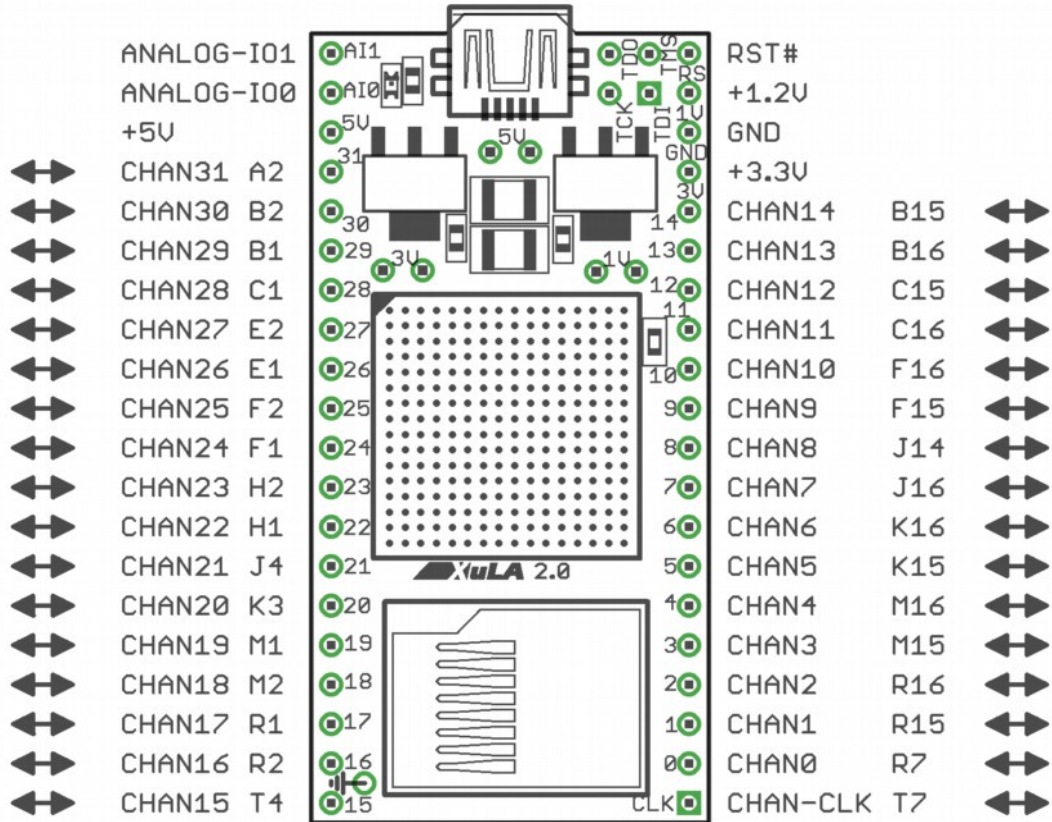
SPI Flash and microSD Card

The XuLA2 has an 8-Mbit SPI Flash ([Winbond W25Q80BV](#)) and a microSD card socket that connect to the FPGA as shown below. During FPGA configuration, the bitstream is read from the SPI flash while the microSD card is disabled. After configuration, either the SPI Flash or the microSD card can be accessed by lowering their respective chip-selects and performing SPI read/write operations.



Prototyping Header

The prototyping header connects the FPGA I/O, microcontroller I/O and the power/GND planes to external circuitry. The signals attached to the header pins are shown below.

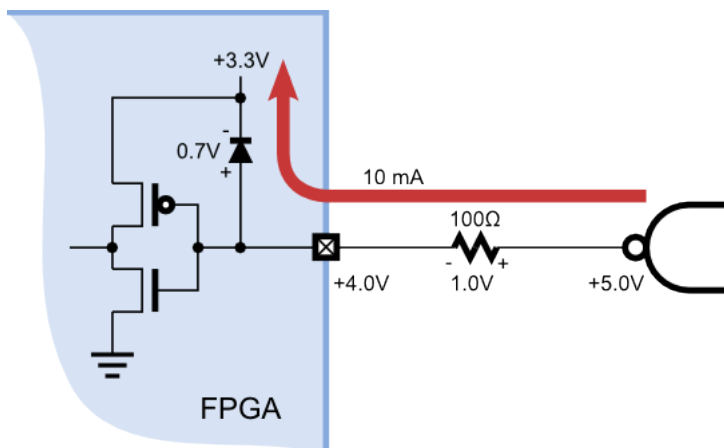


XuLA2 Header Pin Functions	
CHAN0 – CHAN31	These pins connect directly to the FPGA I/O pins. These pins are not 5V-tolerant (see below).
CHAN-CLK	This is a direct connection to a global clock pin of the FPGA. It can also be used as a general-purpose I/O pin. This pin is not 5V-tolerant (see below).
+5V, +1.2V, +3.3V, GND	These pins connect to the voltage supply planes of the XuLA2 as depicted in this figure .
RST#	This pin connects to the microcontroller reset pin. Pulling it to ground and releasing it will cause the XuLA2 to reconfigure itself.
ANALOG-IO0, ANALOG-IO1	These pins connect to the analog-to-digital converter in the microcontroller. ANALOG-IO0 can also output an analog voltage with a limited range and precision. Both pins are also useable as standard digital I/O pins.

5V Tolerance Issues

The CHAN* and CHAN-CLK pins connect directly to the pins of the FPGA which typically use an I/O voltage of 3.3V. When driving the inputs of external 5V logic, you should check that their V_{IH} threshold is less than the V_{OH} of the XuLA2 outputs. (This is true for most 5V logic families.)

You must also take care not to exceed the input voltage rating of the FPGA pins when they are driven by external 5V logic outputs. A common technique for protecting the Xilinx FPGA pins is shown below. When presented with a voltage greater than $3.3V + 0.7V = 4.0V$, the protection diode built into the FPGA pin's circuitry conducts current and the excess voltage is dropped across the 100Ω resistor. This keeps the voltage directly on the FPGA pin from ever exceeding 4V, which is within tolerated limits. The resistor value should be set so the current through the protection diode does not exceed 10 mA.

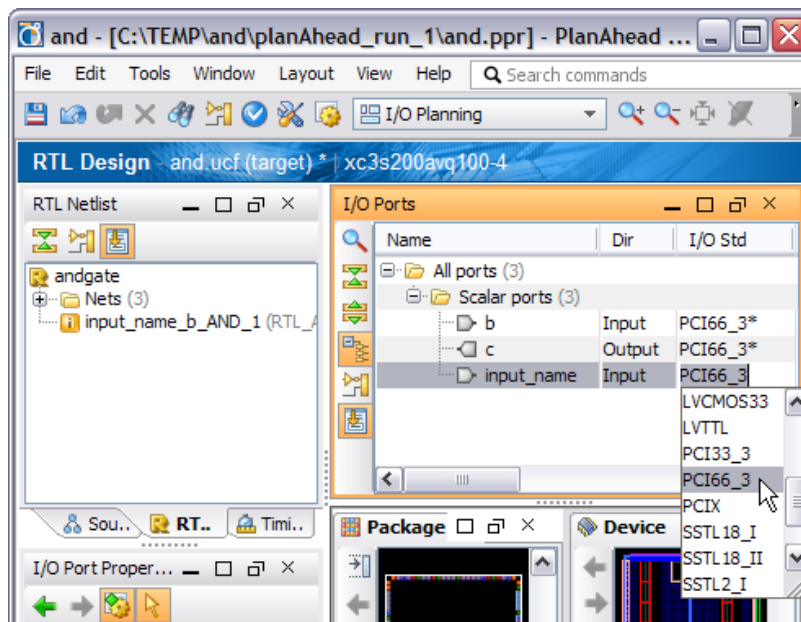


However, by default, the Xilinx Spartan-6 FPGA disables the protection diodes.

In order to enable these diodes, you can place the following text into your Xilinx ISE project constraint file for every I/O pin that will be connected to 5V logic:

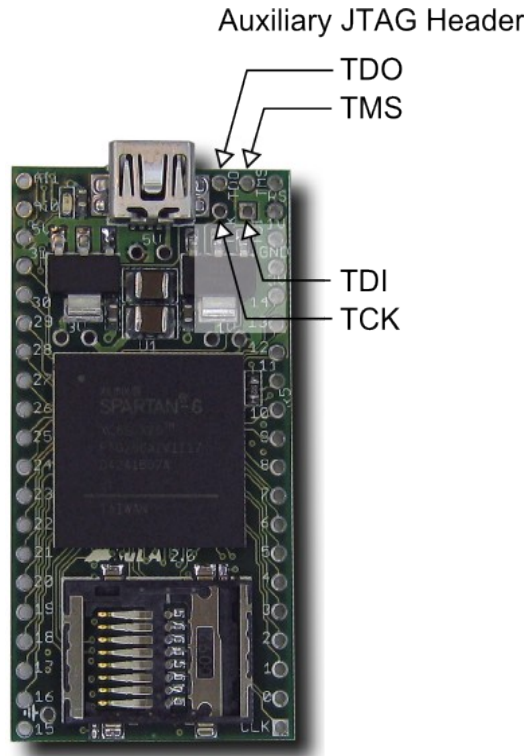
```
NET "input_name" IOSTANDARD = PCI66_3;
```

Or you can use the Xilinx PlanAhead tool to set the I/O standard for the pins like so:

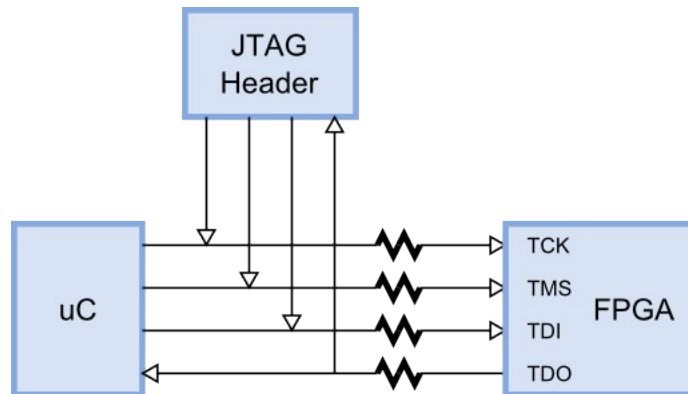


Auxiliary JTAG Header

This four-pin header (shown below) provides third-party JTAG cables with access to the JTAG pins of the FPGA.



The connections of the JTAG header to the rest of the XuLA2 circuitry are as follows.



The current limiting resistors prevent the microcontroller or JTAG cable from damaging the FPGA if they are using a higher supply voltage than the FPGA.

When using a third-party JTAG cable, the associated I/O pins of the microcontroller must be placed in a high-impedance state. The procedure for doing this is described [here](#).

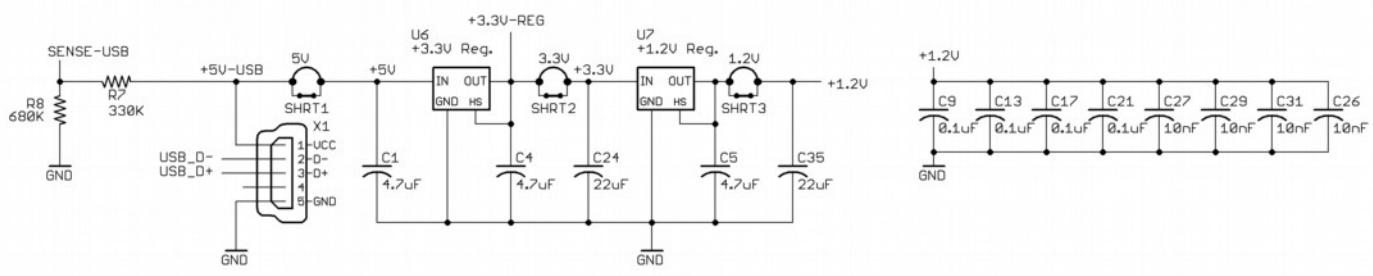
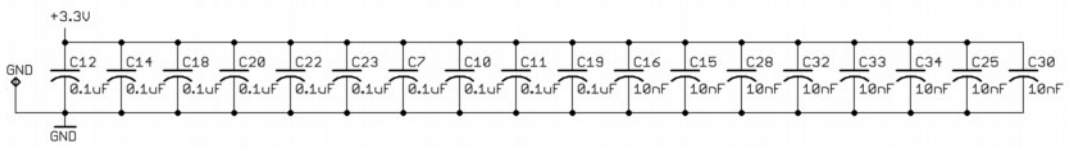
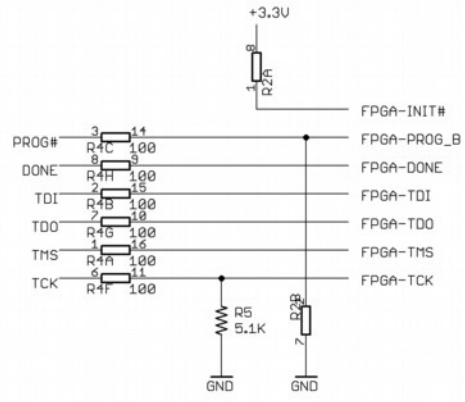
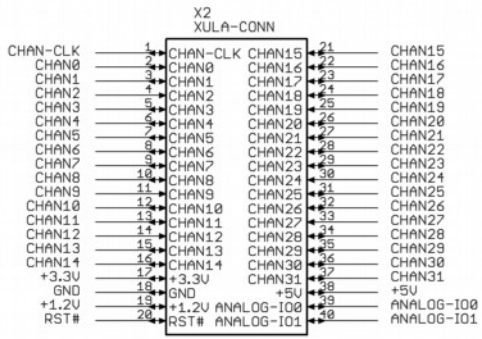
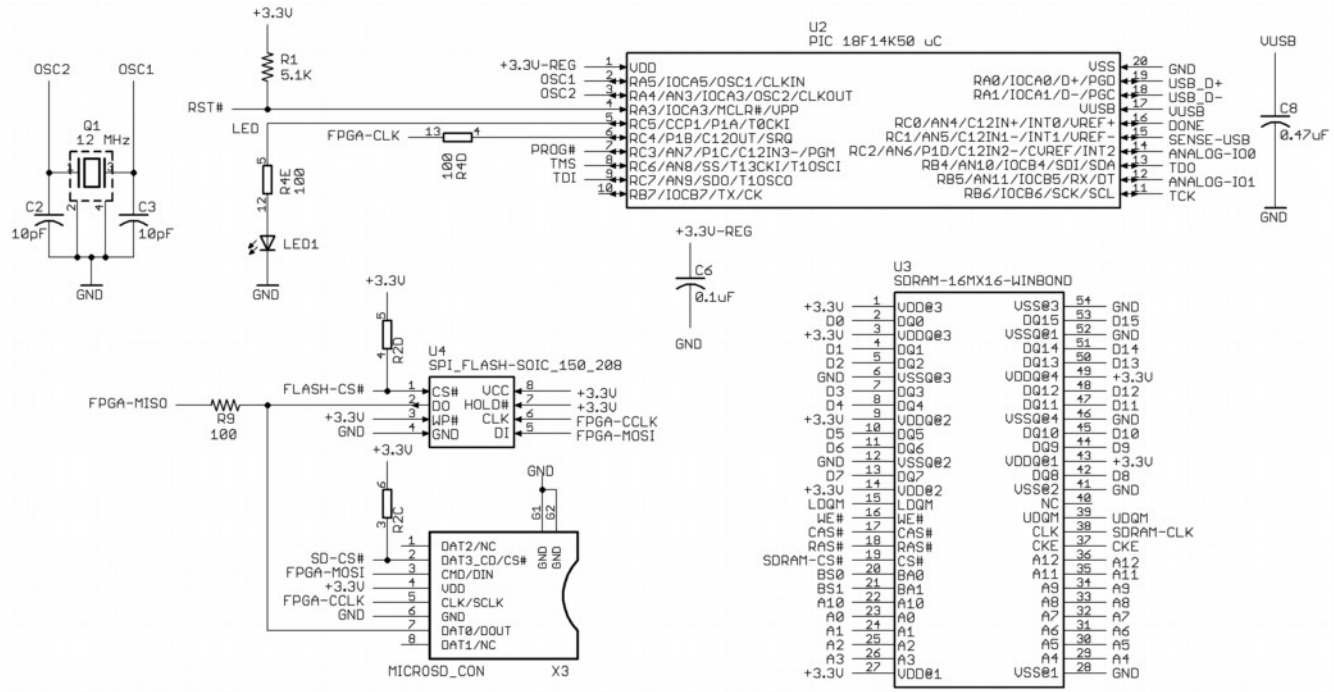
A.1 Pin Connections

Net Name	FPGA	SDRAM	Prototyping Header	SPI Flash	microSD	uC	Notes
A0	E4	A0					
A1	E3	A1					
A2	D3	A2					
A3	C3	A3					
A4	B12	A4					
A5	A12	A5					
A6	D12	A6					
A7	E12	A7					
A8	G16	A8					
A9	G12	A9					
A10	F4	A10					
A11	G11	A11					
A12	H13	A12					
BS0	H3	BS0					
BS1	G3	BS1					
CAS#	L3	CAS#					
CHAN-CLK	T7		CHAN-CLK				Goes to FPGA GCLK pin.
CHAN0	R7		CHAN0				Goes to FPGA GCLK pin.
CHAN1	R15		CHAN1				
CHAN2	R16		CHAN2				
CHAN3	M15		CHAN3				
CHAN4	M16		CHAN4				
CHAN5	K15		CHAN5				
CHAN6	K16		CHAN6				
CHAN7	J16		CHAN7				Goes to FPGA GCLK pin.
CHAN8	J14		CHAN8				Goes to FPGA GCLK pin.
CHAN9	F15		CHAN9				
CHAN10	F16		CHAN10				
CHAN11	C16		CHAN11				
CHAN12	C15		CHAN12				
CHAN13	B16		CHAN13				

Net Name	FPGA	SDRAM	Prototyping Header	SPI Flash	microSD	uC	Notes
CHAN14	B15		CHAN14				
CHAN15	T4		CHAN15				
CHAN16	R2		CHAN16				
CHAN17	R1		CHAN17				
CHAN18	M2		CHAN18				
CHAN19	M1		CHAN19				
CHAN20	K3		CHAN20				Goes to FPGA GCLK pin.
CHAN21	J4		CHAN21				Goes to FPGA GCLK pin.
CHAN22	H1		CHAN22				
CHAN23	H2		CHAN23				
CHAN24	F1		CHAN24				Goes to FPGA GCLK pin.
CHAN25	F2		CHAN25				Goes to FPGA GCLK pin.
CHAN26	E1		CHAN26				
CHAN27	E2		CHAN27				
CHAN28	C1		CHAN28				
CHAN29	B1		CHAN29				
CHAN30	B2		CHAN30				
CHAN31	A2		CHAN31				
CKE	J12	CKE					
D0	P6	D0					
D1	T6	D1					
D2	T5	D2					
D3	P5	D3					
D4	R5	D4					
D5	N5	D5					
D6	P4	D6					
D7	N4	D7					
D8	P12	D8					
D9	R12	D9					
D10	T13	D10					
D11	T14	D11					
D12	R14	D12					
D13	T15	D13					
D14	T12	D14					
D15	P11	D15					
FLASH-CS#	T3			CS#			SPI Flash chip-select.
FPGA-CCLK	R11			SCLK	SCLK		
FPGA-CLK	A9					I/O	12 MHz from uC.
FPGA-DONE	P13					I/O	
FPGA-MISO	P10			DO	DO		
FPGA-MOSI	T10			DI	DI		
FPGA-PROG_B	T2					I/O	
FPGA-TCK	C14					I/O	
FPGA-TDI	C12					I/O	

Net Name	FPGA	SDRAM	Prototyping Header	SPI Flash	microSD	uC	Notes
FPGA-TDO	E14					I/O	
FPGA-TMS	A15					I/O	
LDQM	M4	LDQM					
RAS#	L4	RAS#					
SD-CS#	T8				CS#		MicroSD chip-select.
SDRAM-CLK	K12	CLK					
SDRAM_CLKFB	K11						SDRAM clock feedback.
SDRAM-CS#	H4	CS#					
UDQM	L13	UDQM					
WE#	M3	WE#					

A.2 *Schematic*



U1
XC6SLX25-FT256

CONFIG

FPGA-PROG_B T2 → PROGRAM_B_2
 FPGA-INIT# R3 → IO_L65P_INIT_B_2
 FPGA-DONE P13 → DONE_2
 +3.3V T41 → IO_L1N_N0_CMPMISO_2
 GND W1 → IO_L13P_M1_2
 GND C4 → IO_L1P_HSWAPEN_0
 GND P14 → SUSPEND

JTAG

FPGA-TCK C14 → TCK
 FPGA-TMS A15 → TMS
 FPGA-TDI C12 → TDI
 FPGA-TDO E14 → TDO

PROTOTYPING HEADER

CHAN-CLK T7 → IO_L32N_GCLK28_2
 CHAN0 R2 → IO_L32P_GCLK29_2
 CHAN1 R15 → IO_L49P_M1D010_1
 CHAN2 R16 → IO_L49N_M1D011_1
 CHAN3 M15 → IO_L46P_FCS_B_M1D02_1
 CHAN4 M16 → IO_L46N_FOE_B_M1D03_1
 CHAN5 K15 → IO_L41P_A3_M1D06_1
 CHAN6 K16 → IO_L41N_A2_M1D07_1
 CHAN7 J16 → IO_L43N_GCLK4_M1D05_1
 CHAN8 J14 → IO_L43P_GCLK5_M1D04_1
 CHAN9 F15 → IO_L35P_A11_M1A7_1
 CHAN10 F16 → IO_L35N_A10_M1A2_1
 CHAN11 C16 → IO_L33N_A14_M1A4_1
 CHAN12 C15 → IO_L33P_A15_M1A10_1
 CHAN13 B16 → IO_L29N_A22_M1A14_1
 CHAN14 B15 → IO_L29P_A23_M1A13_1
 CHAN15 T4 → IO_L63N_2
 CHAN16 B2 → IO_L32P_M3D014_3
 CHAN17 R1 → IO_L32N_M3D015_3
 CHAN18 M2 → IO_L35P_M3D010_3
 CHAN19 M1 → IO_L35N_M3D011_3
 CHAN20 K3 → IO_L42P_GCLK25_TROY2_M3UDM_3
 CHAN21 J4 → IO_L42N_GCLK24_M3LDM_3
 CHAN22 H1 → IO_L39N_M3LDQSN_3
 CHAN23 H2 → IO_L39P_M3LDQS_3
 CHAN24 F2 → IO_L41N_GCLK26_M3DQ5_3
 CHAN25 F1 → IO_L41P_GCLK27_M3DQ4_3
 CHAN26 E1 → IO_L46N_M3CLKN_3
 CHAN27 E2 → IO_L46P_M3CLK_3
 CHAN28 C1 → IO_L50P_M3UE_3
 CHAN29 B1 → IO_L50N_M3BA2_3
 CHAN30 B2 → IO_L52P_M3A8_3
 CHAN31 A2 → IO_L52N_M3A9_3

FLASH

FPGA-CLK A8 → IO_L34N_GCLK18_0
 SD-CS# T8 → IO_L30N_GCLK0_USERCLK_2
 FLASH-CS# R4 → IO_L65N_CS0_B_2
 FPGA-CCLK F14 → IO_L1P_CCLK_2
 FPGA-MOSI T10 → IO_L3N_MOSI_CSI_B_MISO0_2
 FPGA-MISO P10 → IO_L3P_00_DIN_MISO_MISO1_2

SDRAM

CKE J12 → IO_L40N_GCLK10_M1A6_1
 SDRAM-CLK K15 → IO_L42N_GCLK6_TROY1_M1LDM_1
 SDRAM-CLK K14 → IO_L42P_GCLK7_M1UDM_1
 SDRAM-CS# H4 → IO_L40P_GCLK21_M3A5_3
 RAS# L4 → IO_L45P_M3A3_3
 CAS# L3 → IO_L36P_M3D08_3
 WE# M3 → IO_L1N_UREF_3
 A0 E4 → IO_L54P_M3RESET_3
 A1 D3 → IO_L54N_M3A11_3
 A2 C3 → IO_L49P_M3A7_3
 A3 B12 → IO_L48P_M3BA0_3
 A4 A12 → IO_L62P_0
 A5 D12 → IO_L62N_UREF_0
 A6 D12 → IO_L66N_SCP0_0
 A7 E12 → IO_L1N_A24_UREF_1
 A8 G16 → IO_L36N_A8_M1BA1_1
 A9 G12 → IO_L38P_A5_M1CLK_1
 A10 G14 → IO_L53P_M3CKE_3
 A11 H13 → IO_L38N_A20_M1A11_1
 A12 H13 → IO_L39P_M1A3_1
 BS1 G3 → IO_L48P_M3DQ6_3
 BS0 H3 → IO_L44N_GCLK20_M3A6_3
 D0 P6 → IO_L47P_2
 D1 T6 → IO_L47N_2
 D2 T5 → IO_L48N_RDWR_B_UREF_2
 D3 R5 → IO_L49N_D4_2
 D4 N5 → IO_L48P_D7_2
 D5 P4 → IO_L49P_D3_2
 D6 P4 → IO_L63P_2
 D7 N4 → IO_L2N_3
 D8 P12 → IO_L12N_D2_MISO3_2
 D9 H12 → IO_L52P_M1DQ14_1
 D10 T14 → IO_L51N_M1DQ13_1
 D11 T14 → IO_L51P_M1DQ12_1
 D12 R14 → IO_L50P_M1DQ05_1
 D13 T12 → IO_L50N_M1UDQSN_1
 D14 P11 → IO_L52N_M1DQ15_1
 D15 P11 → IO_L13N_D10_2
 LDQM M4 → IO_L1P_3
 UDQM L13 → IO_L53N_UREF_1

POWER & GROUND

GND A1 → GND019 +3.3V D2 → UCC0_303
 GND T1 → GND022 +3.3V J7 → UCC0_304
 GND G2 → GND018 +3.3V N2 → UCC0_305
 GND L2 → GND020 +3.3V B4 → UCC0_002
 GND P3 → GND024 +3.3V G4 → UCC0_302
 GND I4 → GND012 +3.3V K4 → UCC0_301
 GND J5 → GND016 +3.3V R4 → UCC0_203
 GND R6 → GND013 +3.3V F5 → UCCAUX01
 GND H7 → GND05 +3.3V L6 → UCCAUX07
 GND K7 → GND014 +3.3V D7 → UCCAUX08
 GND G8 → GND03 +3.3V U → UCC0_005
 GND J8 → GND015 +3.3V N7 → UCC0_201
 GND J8 → GND09 +3.3V F8 → UCCAUX05
 GND H8 → GND025 +3.3V R8 → UCC0_204
 GND E9 → GND011 +3.3V B9 → UCC0_001
 GND H9 → GND023 +3.3V L9 → UCCAUX04
 GND K9 → GND017 +3.3V D10 → UCC0_003
 GND R10 → GND06 +3.3V G10 → UCCAUX06
 GND B11 → GND08 +3.3V U → UCCAUX03
 GND H12 → GND07 +3.3V U → UCC0_202
 GND D13 → GND026 +3.3V F14 → UCCAUX02
 GND N13 → GND021 +3.3V B13 → UCC0_004
 GND G15 → GND01 +3.3V G13 → UCC0_102
 GND L15 → GND02 +3.3V K13 → UCC0_101
 GND A16 → GND010 +3.3V D13 → UCC0_104
 GND T16 → GND014 +3.3V J15 → UCC0_104
 +3.3V N15 → UCC0_105
 +3.3V N15 → UCC0_103
 +1.2V G7 → UCCINT05
 +1.2V J7 → UCCINT01
 +1.2V H8 → UCCINT04
 +1.2V K8 → UCCINT02
 +1.2V J9 → UCCINT03
 +1.2V H10 → UCCINT08
 +1.2V K10 → UCCINT06
 +1.2V K10 → UCCINT07

UNUSED

D1 → IO_L49N_M3A2_3
 G1 → IO_L40N_M3DQ7_3
 J1 → IO_L38N_M3DQ3_3
 K1 → IO_L37N_M3DQ1_3
 L1 → IO_L36N_M3DQ9_3
 M1 → IO_L34N_M3UDQSN_3
 P1 → IO_L33N_M3DQ13_3
 R1 → IO_L1N_UREF_0
 T1 → IO_L48N_M3BA1_3
 U1 → IO_L37P_M3DQ0_3
 V1 → IO_L33P_M3DQ12_3
 W1 → IO_L83N_UREF_3
 X1 → IO_L83P_3
 Y1 → IO_L53N_M3A12_3
 Z1 → IO_L38P_M3DQ2_3
 AA1 → IO_L34P_M3UDQ5_3
 AB1 → IO_L2N_0
 AC1 → IO_L2P_0
 AD1 → IO_L3N_0
 AE1 → IO_L3P_0
 AF1 → IO_L55N_M3A14_3
 AG1 → IO_L51N_M3A4_3
 AH1 → IO_L51P_M3A10_3
 AI1 → IO_L43N_GCLK22_IRDY2_M3CASN_3
 AJ1 → IO_L47P_M3A0_3
 AK1 → IO_L45N_M3DQ1_3
 AL1 → IO_L2P_3
 AM1 → IO_L4N_0
 AN1 → IO_L4P_0
 AO1 → IO_L7N_0
 AP1 → IO_L7P_0
 AQ1 → IO_L5N_0
 AR1 → IO_L55P_M3A13_3
 AS1 → IO_L51P_M3A10_3
 AT1 → IO_L43P_GCLK23_M3RASN_3
 AU1 → IO_L47N_M3A1_3
 AV1 → IO_L64P_D8_2
 AW1 → IO_L64N_D9_2
 AX1 → IO_L6N_0
 AY1 → IO_L6P_0
 AZ1 → IO_L36P_GCLK15_0
 BA1 → IO_L5P_0
 BB1 → IO_L62N_D6_2
 BC1 → IO_L31N_GCLK30_D15_2
 BD1 → IO_L31P_GCLK31_D14_2
 BE1 → IO_L33N_0
 BF1 → IO_L33P_0
 BG1 → IO_L38N_UREF_0
 BH1 → IO_L38P_0
 BI1 → IO_L36N_GCLK14_0
 BJ1 → IO_L62P_D5_2
 BK1 → IO_L29N_GCLK2_2
 BL1 → IO_L30P_GCLK1_D13_2
 BM1 → IO_L34P_GCLK19_0
 BN1 → IO_L40N_0
 BO1 → IO_L40P_0
 BP1 → IO_L29P_GCLK3_2
 BQ1 → IO_L14P_D11_2
 BR1 → IO_L14N_D12_2
 BS1 → IO_L23P_2
 BT1 → IO_L23N_2
 BU1 → IO_L35N_GCLK16_0
 BV1 → IO_L35P_GCLK17_0
 BW1 → IO_L37N_GCLK12_0
 BX1 → IO_L37P_GCLK13_0
 BY1 → IO_L64P_SCP5_0
 BZ1 → IO_L14P_2
 CA1 → IO_L66N_UREF_2
 CB1 → IO_L39N_0
 CC1 → IO_L39P_0
 CD1 → IO_L66P_SCP1_0
 CE1 → IO_L64N_SCP4_0
 CF1 → IO_L38N_A4_M1CLK_1
 CG1 → IO_L48P_GCLK11_M1A5_1
 CH1 → CHPCS_B_2
 CI1 → IO_L2N_CMPMISO_2
 CJ1 → IO_L38P_A21_M1RESET_1
 CK1 → IO_L53P_1
 CL1 → IO_L2P_CMPCLK_2
 CM1 → IO_L12P_D1_MISO2_2
 CN1 → IO_L63N_SCP6_0
 CO1 → IO_L63P_SCP7_0
 CP1 → IO_L1P_A25_1
 CQ1 → IO_L32P_A17_M1A8_1
 CR1 → IO_L41P_GCLK9_IRDY1_M1RASN_1
 CS1 → IO_L74P_AWAKE_1
 CT1 → IO_L65N_SCP2_0
 CU1 → IO_L65P_SCP3_0
 CV1 → IO_L31P_A19_M1CKE_1
 CW1 → IO_L32N_A16_M1A9_1
 CX1 → IO_L36P_A9_M1BA0_1
 CY1 → IO_L39N_M1DQ1_1
 CZ1 → IO_L41N_GCLK8_M1CASN_1
 DA1 → IO_L47P_FHE_B_M1DQ0_1
 DB1 → IO_L74N_DOUT_BUSY_1
 DC1 → IO_L45P_A1_M1DQ05_1
 DD1 → IO_L34P_A13_M1WE_1
 DE1 → IO_L37P_A7_M1A0_1
 DF1 → IO_L48P_HDC_M1DQ8_1
 DG1 → IO_L31N_A18_M1A12_1
 DH1 → IO_L34N_A12_M1BA2_1
 DI1 → IO_L37N_A6_M1A1_1
 DJ1 → IO_L47N_LDC_M1DQ1_1
 DK1 → IO_L45N_A0_M1LDQSN_1
 DL1 → IO_L48N_M1DQ9_1

FPGA Connections			
PROJECT: XuLA2	REV: 2.0	Date: not saved!	
AUTHOR: Dave Vandenbout / XESS Corp.			Sheet: 3/3