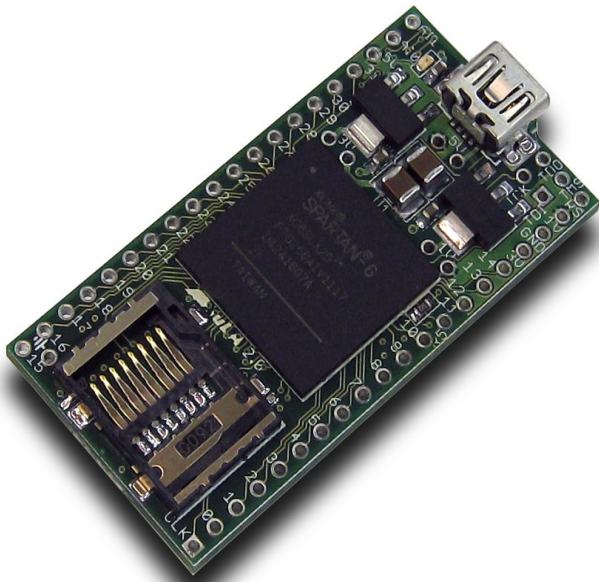


# XuLA2 Manual

*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.

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 <a href="#">Pin Connections appendix</a> .
12/28/2012	1.2	Expanded explanation of how to use a XILINX programming cable with the XuLA2.

<b>C.1 Preliminaries.....</b>	<b>1</b>
Getting Help!.....	1
Take Notice!.....	1
<b>C.2 Installation.....</b>	<b>2</b>
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
<b>C.3 Programming.....</b>	<b>6</b>
Generating Bitstreams for the FPGA.....	6
Downloading Bitstreams into the FPGA.....	9
Downloading Using GXSLD.....	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
<b>C.4 Programmer Models.....</b>	<b>16</b>
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
<b>A.1 Pin Connections.....</b>	<b>22</b>
<b>A.2 Schematic.....</b>	<b>25</b>

## C.1 Preliminaries

---

Here's some helpful information before getting started.

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](mailto: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](#).

- 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*

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](http://www.xilinx.com).

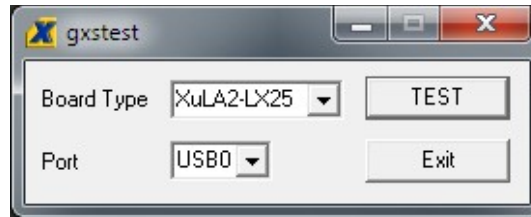
In addition, XESS provides the XSTOOLS utilities for interfacing a PC to your XuLA2. These utilities (along with manuals, design examples and tutorials) are installed automatically when you insert the XSTOOLS CD into your PC. If not, then manually run the SETUP.EXE installation program on the CD. You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

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.

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.

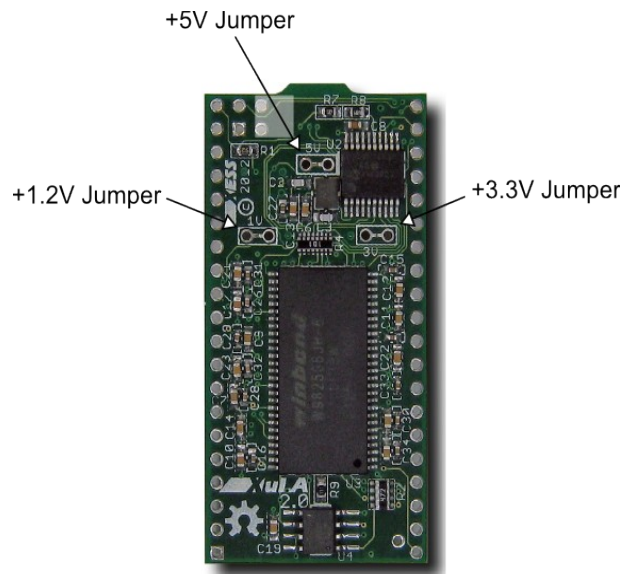


Next, select the type of board you are testing and the port that it's attached to. Then click on the TEST button. GXSTEST will configure the FPGA on your XuLA2 to perform a test procedure. Within a few seconds, a status window will appear informing you of the success or failure of the test.

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.

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.



There are two ways of powering your XuLA2 that can be used alone or in combination:

- receiving power through the USB connector, or
- applying power through the XuLA2 prototyping header.

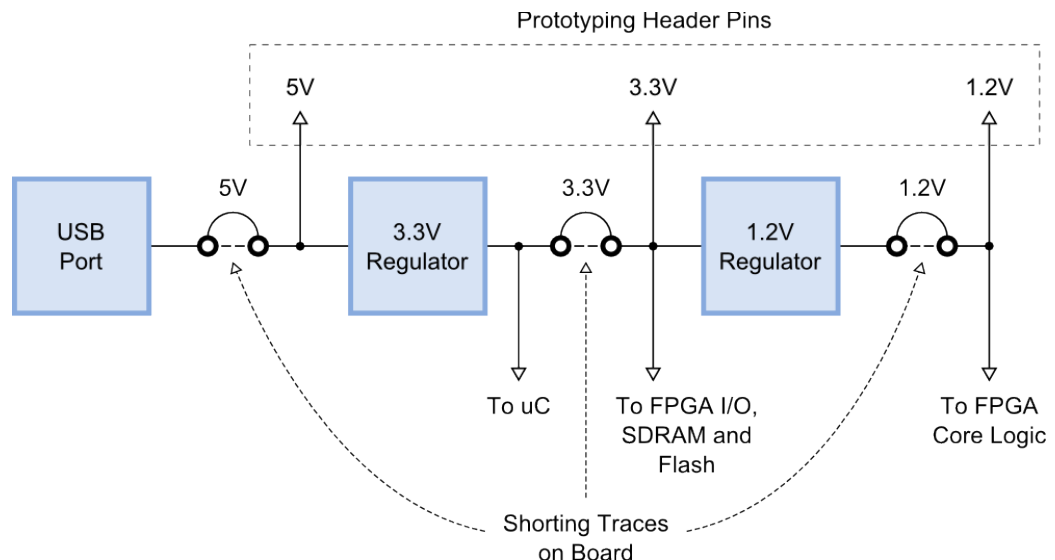
## Applying Power Through the USB Port

Connecting the XuLA2 to a PC USB port provides a 5V supply capable of delivering up to 500 mA of current. This is sufficient for many small to medium-sized FPGA designs running at less than 200 Mhz.

## Applying Power Through the Prototyping Header

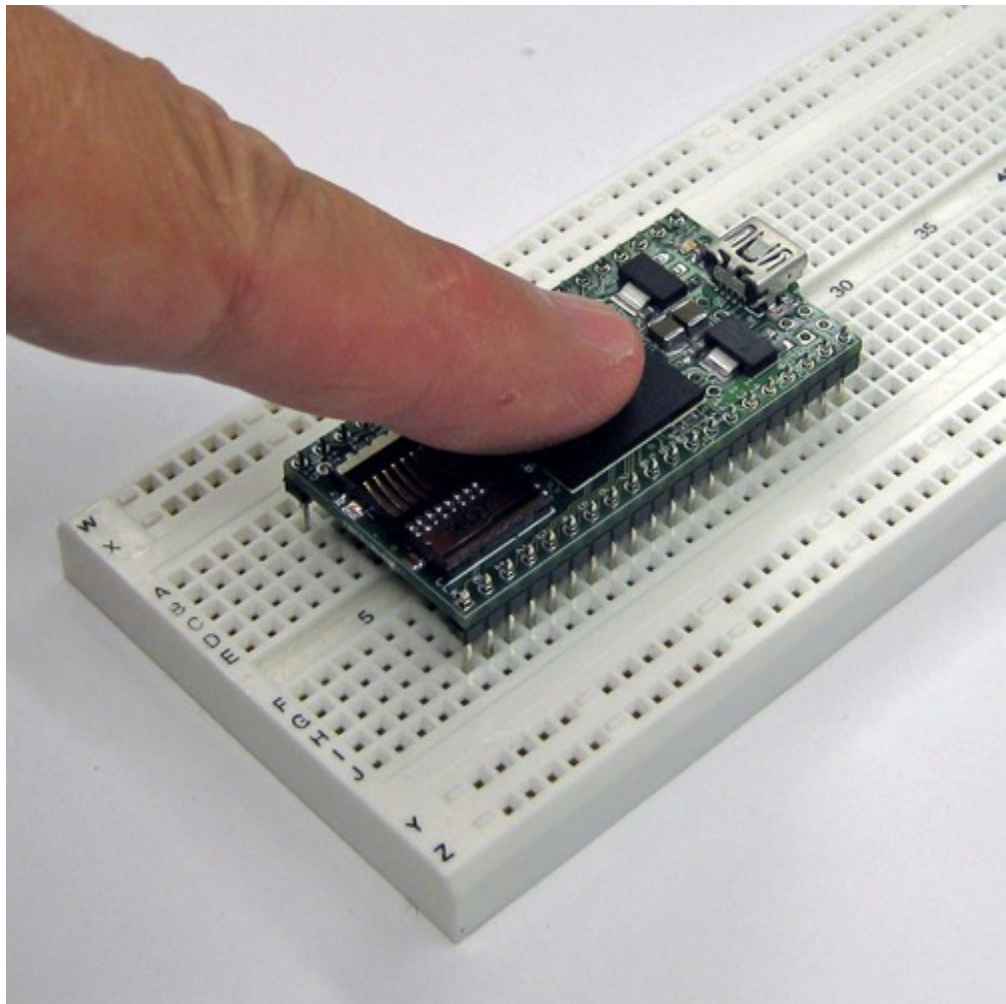
For more power-hungry applications, you can connect one or more voltage supplies directly to the power supply pins of the prototyping header. There are several ways to do this:

- You can attach an 18V-5V supply directly to the 5V pin of the prototyping header. The XuLA2's voltage regulators will generate the required 3.3V and 1.2V supplies needed by the microcontroller, FPGA, SDRAM and Flash (see the figure below). **Do not attach a USB cable unless you have removed the shunt from the 5V jumper** or else you will short the PC USB supply to the external voltage supply and cause possible damage.
- You can attach a 3.3V supply directly to the 3.3V pin of the prototyping header. This supply will directly power the microcontroller, FPGA I/O, SDRAM and Flash while the voltage regulator will generate the 1.2V needed by the FPGA core logic. **Do not attach a USB cable unless you have removed the shunt from the 5V jumper and do not attach a supply to the 5V prototyping pin** or else the output of the 3.3V regulator will drive against the external 3.3V supply and cause possible damage.
- You can power the FPGA core logic by attaching a 1.2V supply directly to the 1.2V pin of the prototyping header and then use one of the previous two methods to power the rest of the XuLA2. **You must remove the shunt on the 1.2V jumper to isolate the output of the 1.2V regulator from the external 1.2V supply.**





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.



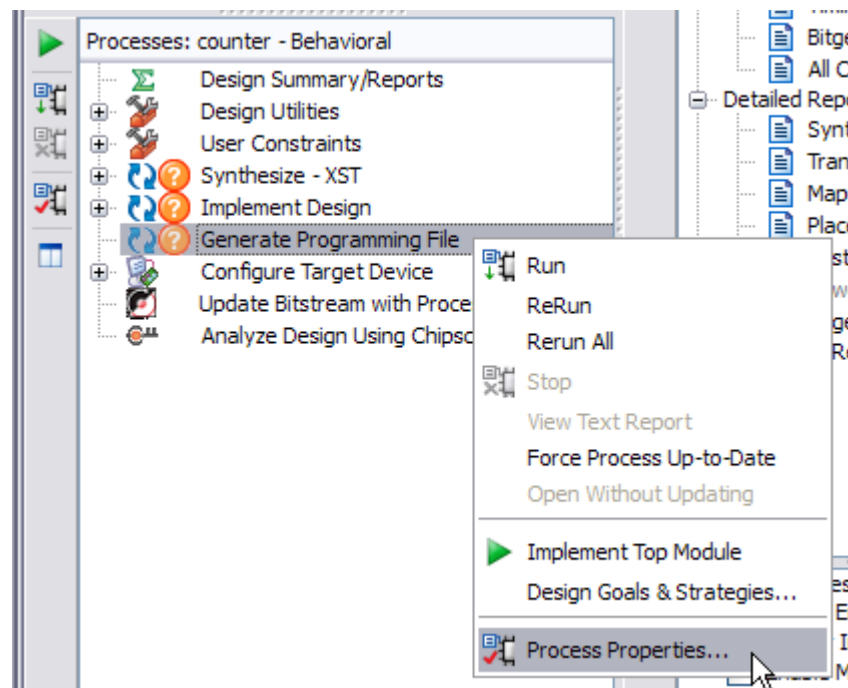
The XuLA2 PCB will accept common headers with 0.025"-thick pins at 0.1" spacing, but you may find it difficult to remove the XuLA2 given how tightly the breadboard grips the pins. A header with thinner pins (such as the Aries 20-0600-20) is a better choice.

## C.3 Programming

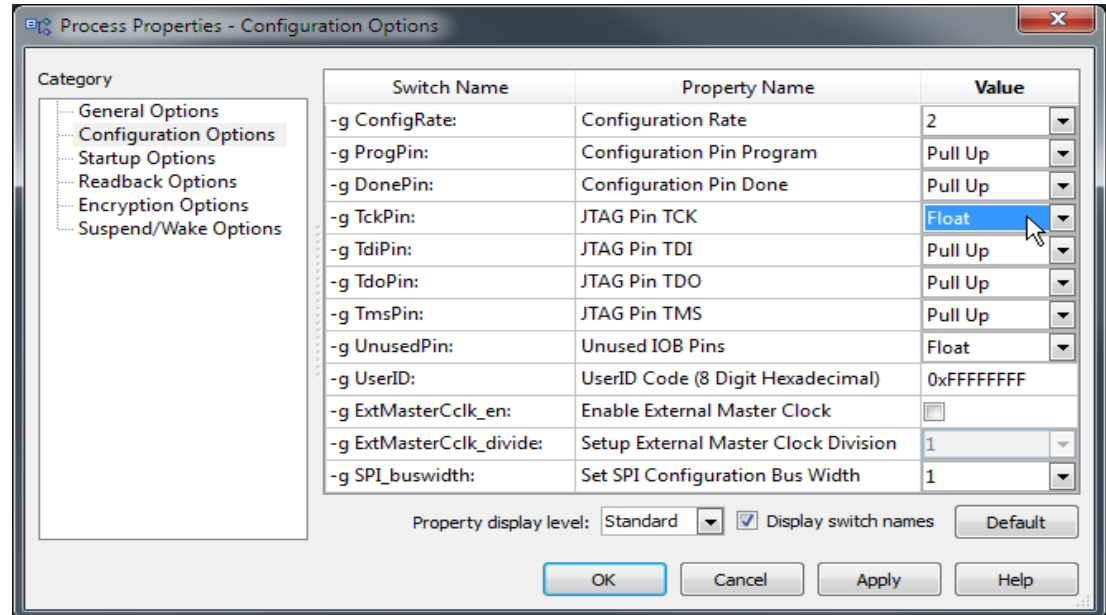
This chapter will show you how to download logic designs into the FPGA of your XuLA2 and how to transfer data between the PC and the SDRAM and Flash memories on the board.

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.



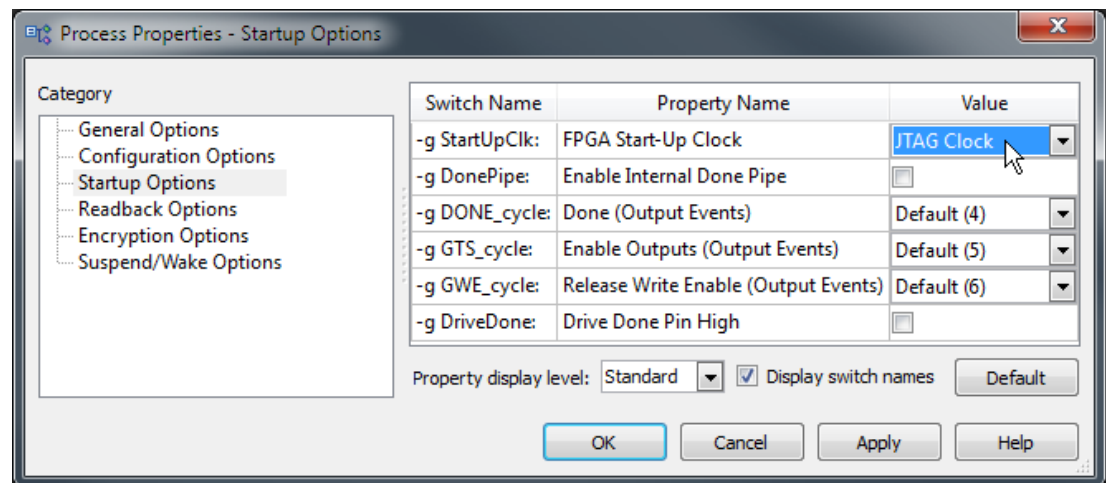
When the **Process Properties** window appears, select the **Configuration Options** category. Make sure the **Configuration Pin Done** property is set to the value **Pull Up**. This is necessary for the microcontroller on the XuLA2 to detect when the FPGA has successfully been configured by the bitstream. 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  $\frac{1}{2}$ -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:

<b>JTAG Clock</b>	Use this setting when your bitstream will be downloaded directly into the XuLA2's FPGA through the USB cable.
<b>CCLK</b>	Use this setting when your bitstream will be downloaded into the XuLA2's serial configuration Flash for eventual transfer to the FPGA.

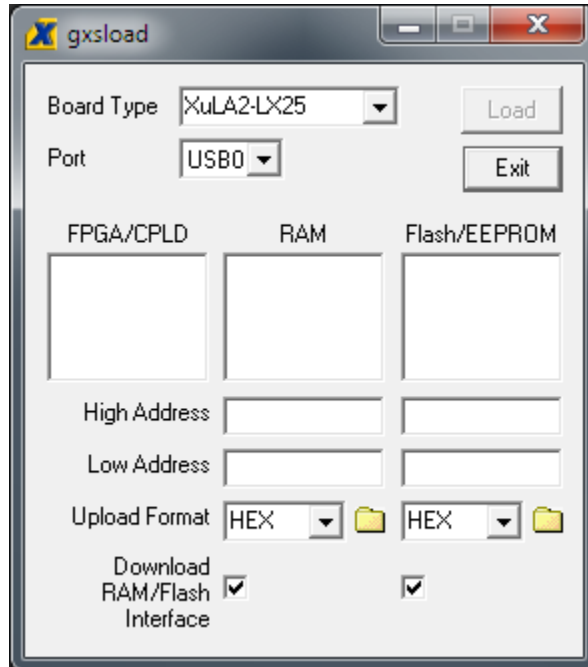


Then click **OK** to finalize your process properties. Now you're ready to generate a bitstream for the XuLA2!

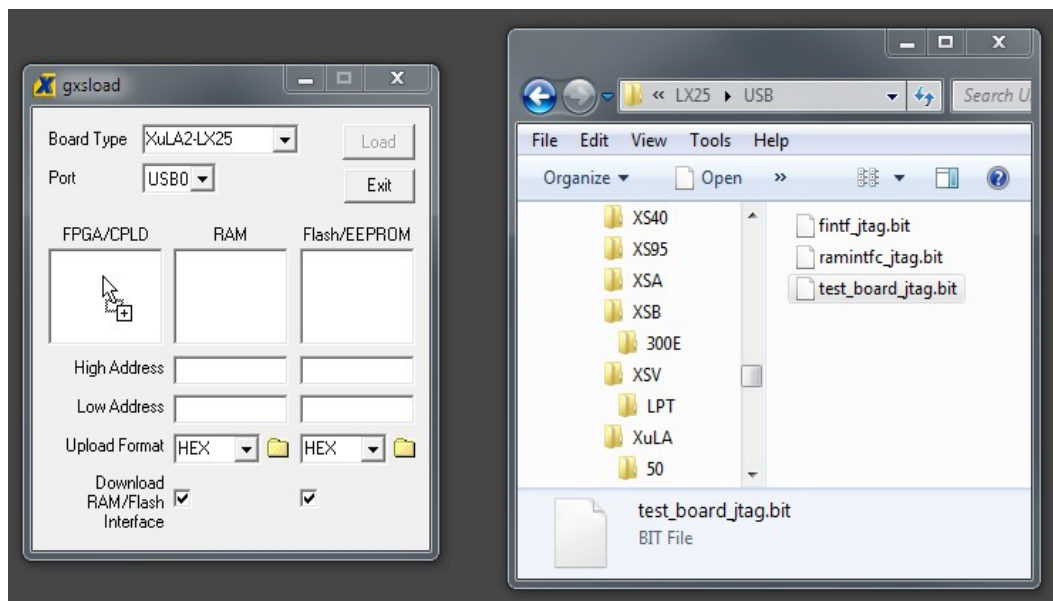
## Downloading Using GXSLOAD

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

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

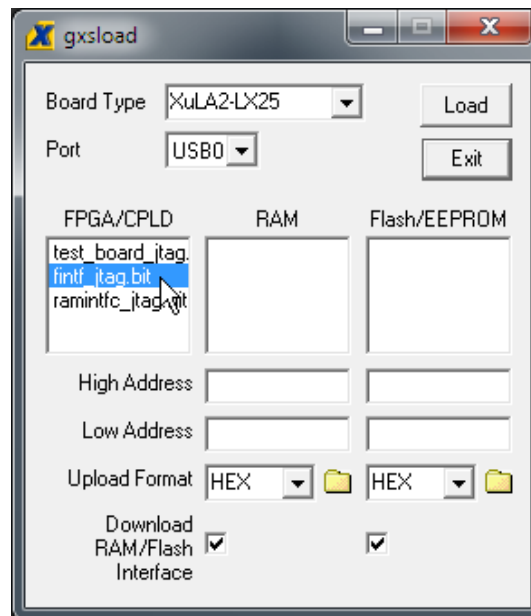


Now you can download bitstream files to the FPGA or CPLD simply by dragging them from their folder and dropping them into the **FPGA/CPLD** pane as shown below.

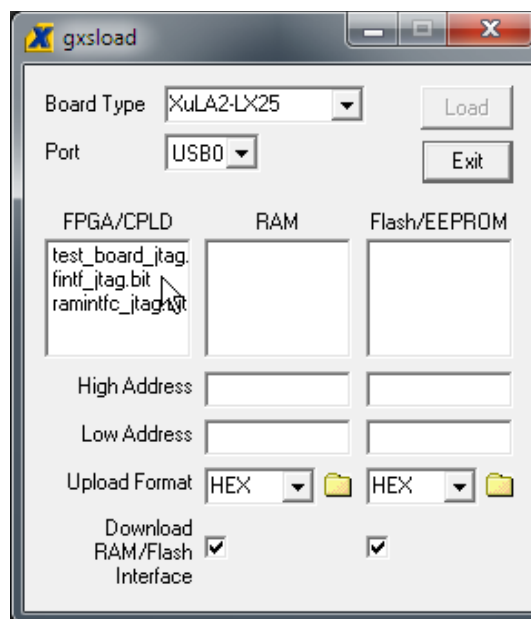


Once you drop the file, the highlighted file name appears in the **FPGA/CPLD** pane and the **Load** button is enabled. Clicking on the **Load** button will begin sending the bitstream to the XuLA2 through the USB cable. GXSLoad will reject any non-downloadable files (ones with a suffix other than .BIT).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

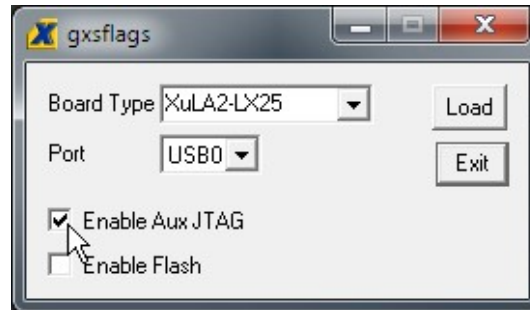


Double-clicking the highlighted file will deselect it so no file will be downloaded. Doing this disables the **Load** button.



## 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 GXSFflags utility. Connect the XuLA2 to your PC with a USB cable and then double-click the GXSFflags icon so the following window appears.



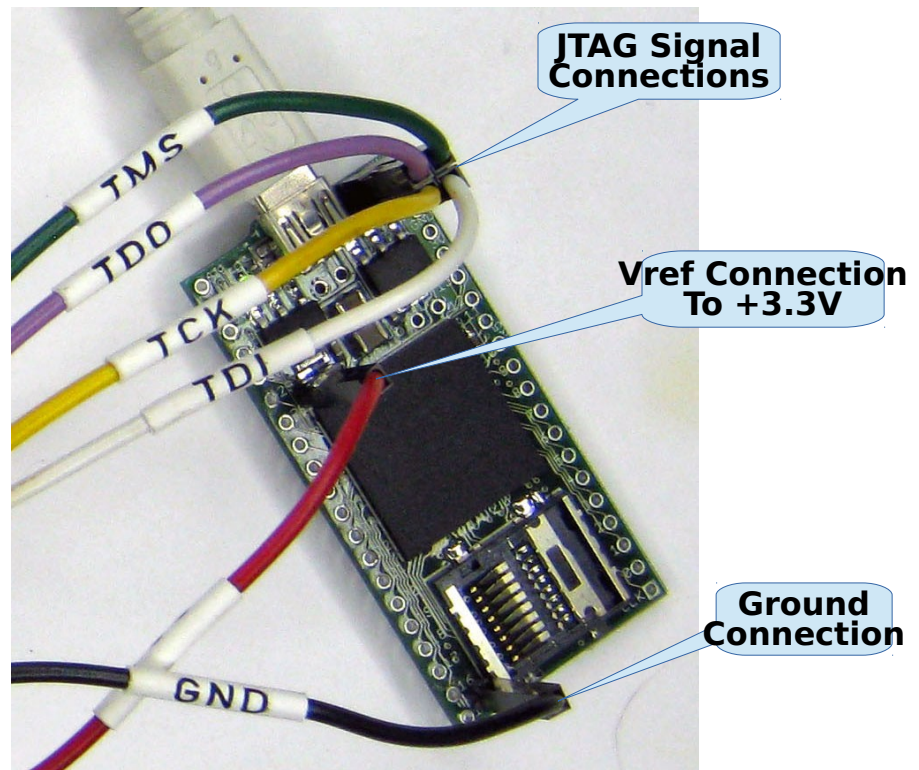
Click on the **Enable Aux JTAG** checkbox and then click on the **Load** button to set the flag. Note that the flag only needs to be set once to support the JTAG cable because it retains its value even when power is removed from the XuLA2. (Be aware that enabling the auxiliary JTAG header disables the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

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 GXSLLOAD 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.)*

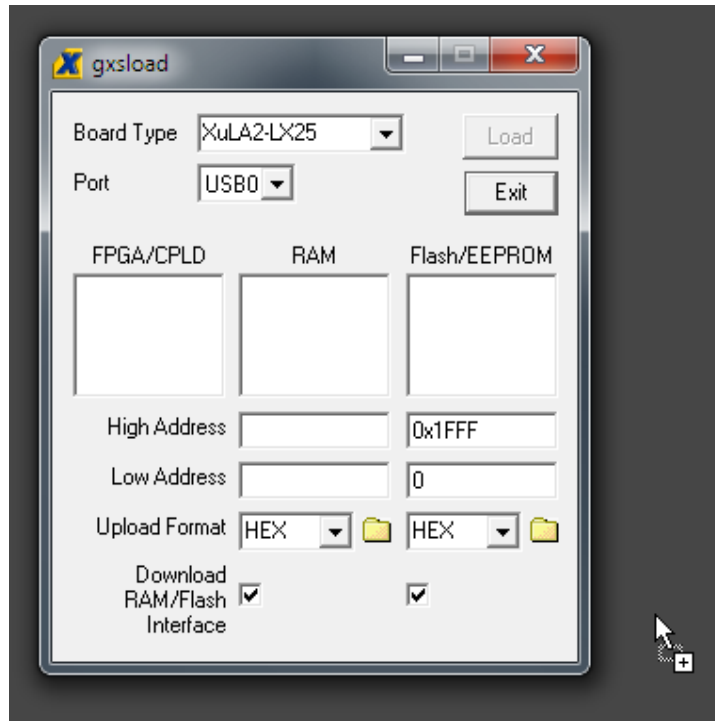
You can also use the XILINX iMPACT software to convert FPGA bitstreams into .MCS or .EXO data files. These file types can be programmed into the XuLA2's Flash using GXSLLOAD in the same way as with a .BIT file.

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.

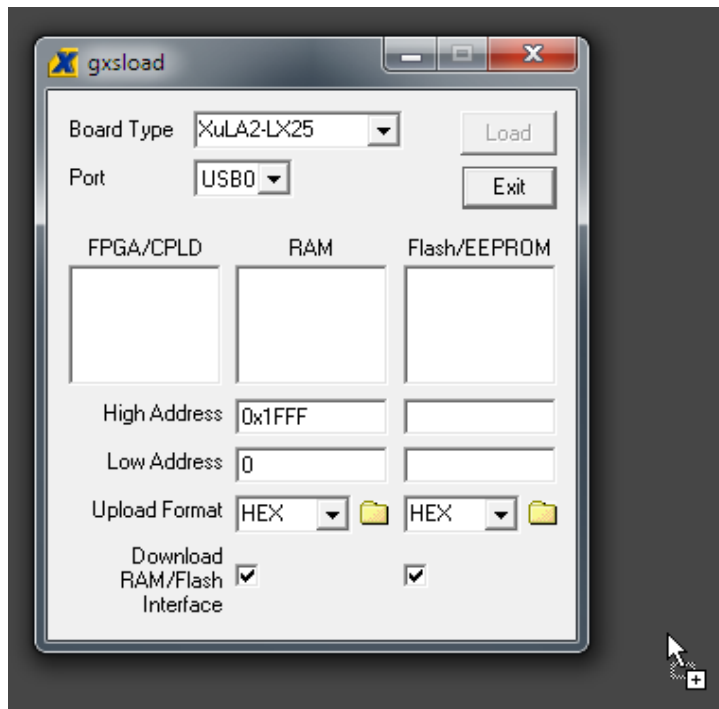
The uploaded data can be stored in the following formats:

- 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.

The XuLA2 contains a synchronous DRAM (SDRAM) whose contents can be downloaded and uploaded by GXSLD. 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 **gxslod** 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*

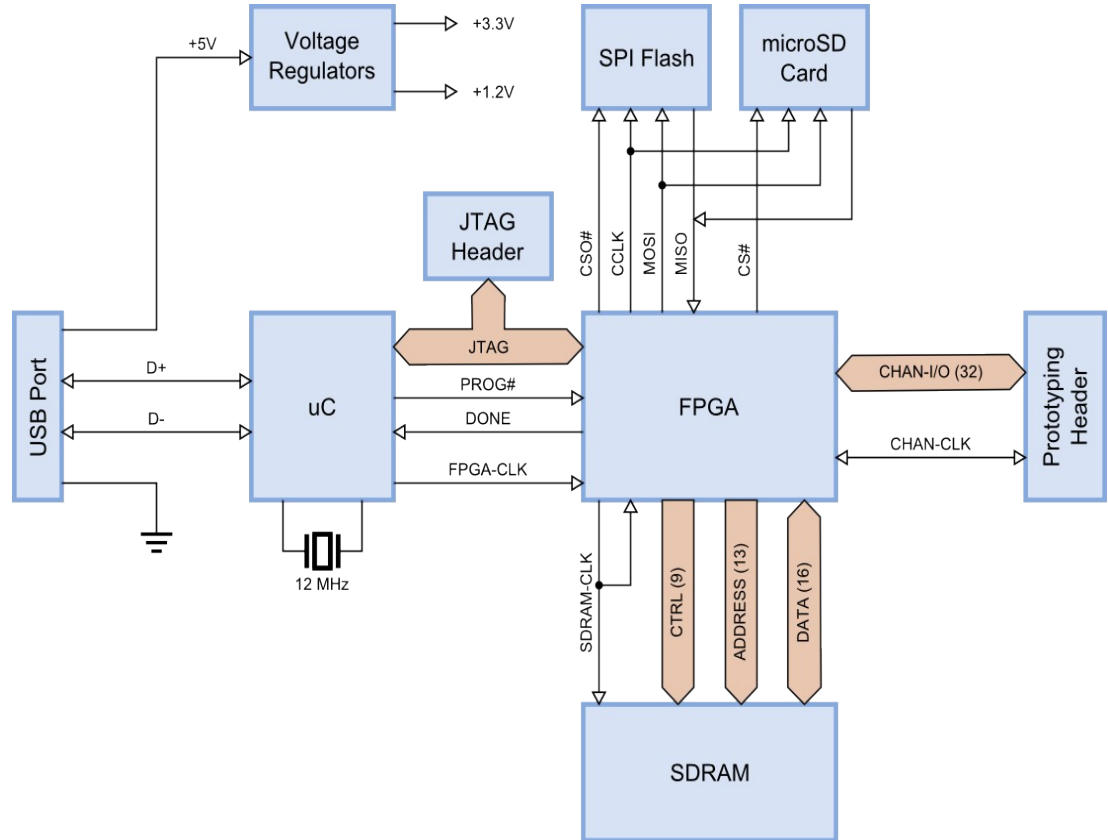
---

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.

The XuLA2 contains the following major components:

<b>FPGA:</b>	This is the field programmable gate array.
<b>Microcontroller:</b>	The microcontroller (uC) handles initialization, clock generation and USB-to-JTAG communications.
<b>SDRAM:</b>	A 256-Mbit SDRAM provides volatile data storage accessible by the FPGA.
<b>SPI Flash:</b>	An 8-Mbit serial Flash device provides non-volatile storage for FPGA configuration bitstreams and user data.
<b>MicroSD Card:</b>	A Secure Digital memory card provides non-volatile storage for user data.
<b>Prototyping Header:</b>	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.
<b>Aux. JTAG Header:</b>	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).

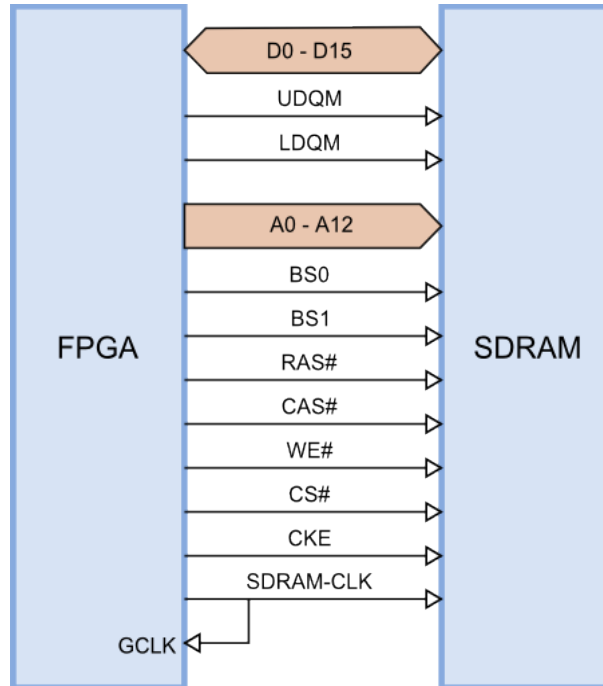
## Microcontroller

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.

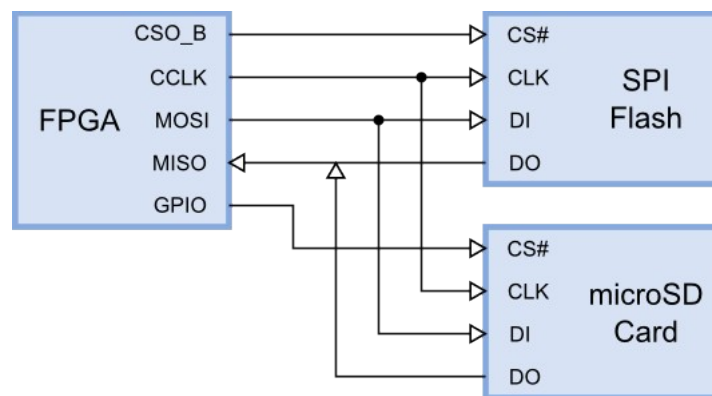
## SDRAM

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.



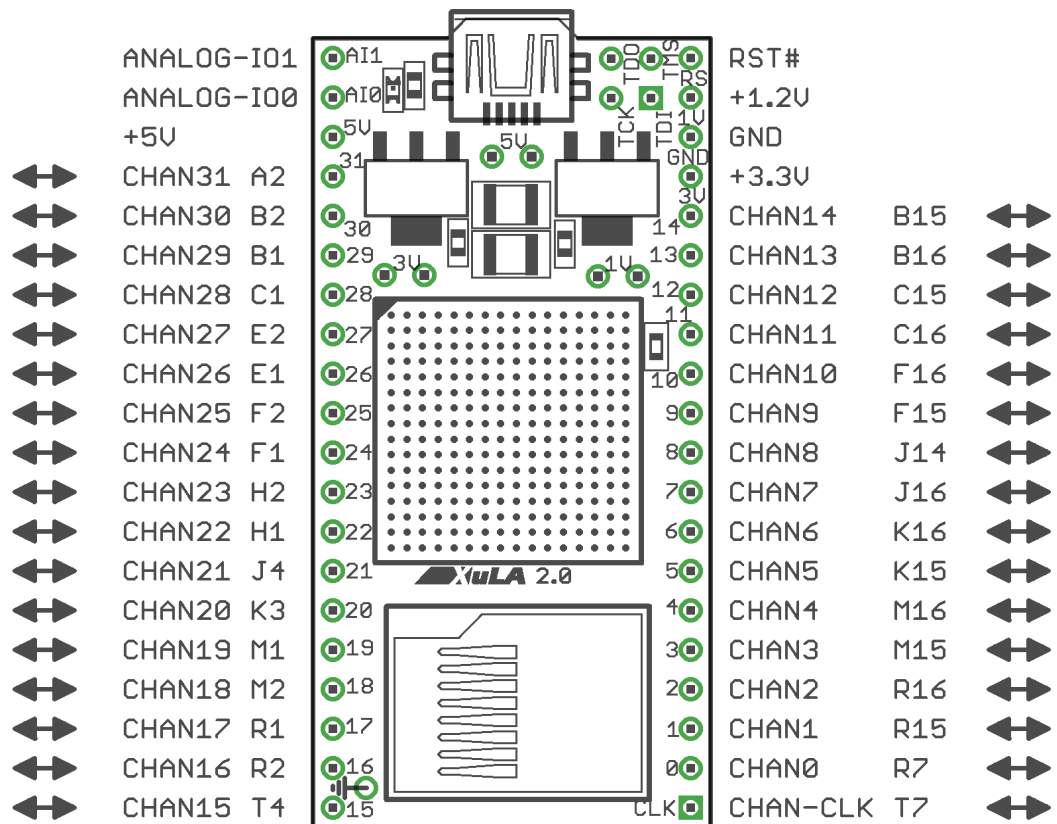
## SPI Flash and microSD Card

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.



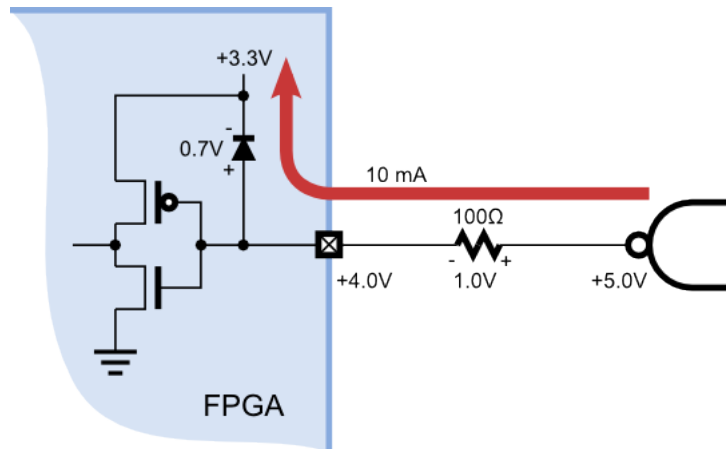
<b>CHAN0 - CHAN31</b>	These pins connect directly to the FPGA I/O pins. <b>These pins are not 5V-tolerant</b> (see below).
<b>CHAN-CLK</b>	This is a direct connection to a global clock pin of the FPGA. It can also be used as a general-purpose I/O pin. <b>This pin is not 5V-tolerant</b> (see below).
<b>+5V, +1.2V, +3.3V, GND</b>	These pins connect to the voltage supply planes of the XuLA2 as depicted in <a href="#">this figure</a> .
<b>RST#</b>	This pin connects to the microcontroller reset pin. Pulling it to ground and releasing it will cause the XuLA2 to reconfigure itself.
<b>ANALOG-IO0, ANALOG-IO1</b>	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  $100\Omega$  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.

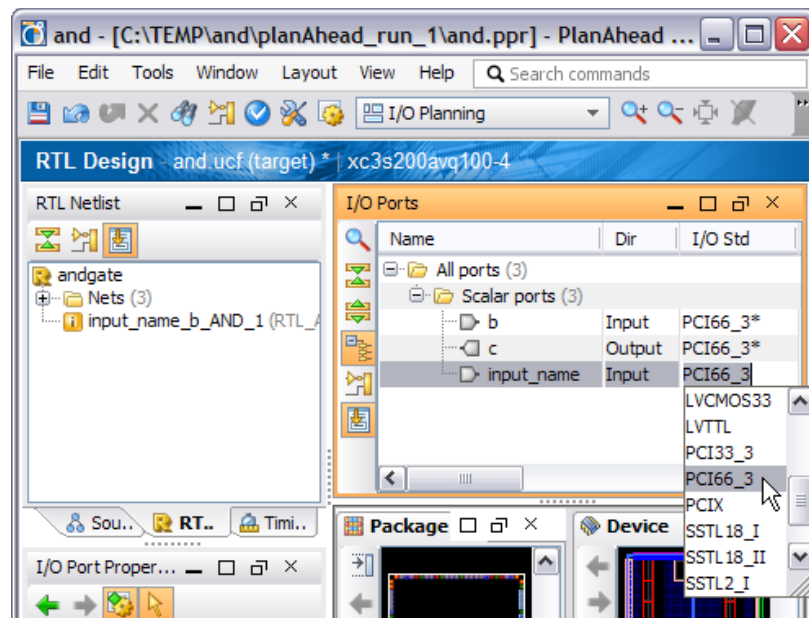


**However, by default, the Xilinx Spartan-6 FPGA disables the protection diodes.**

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 = PCI166_3;
```

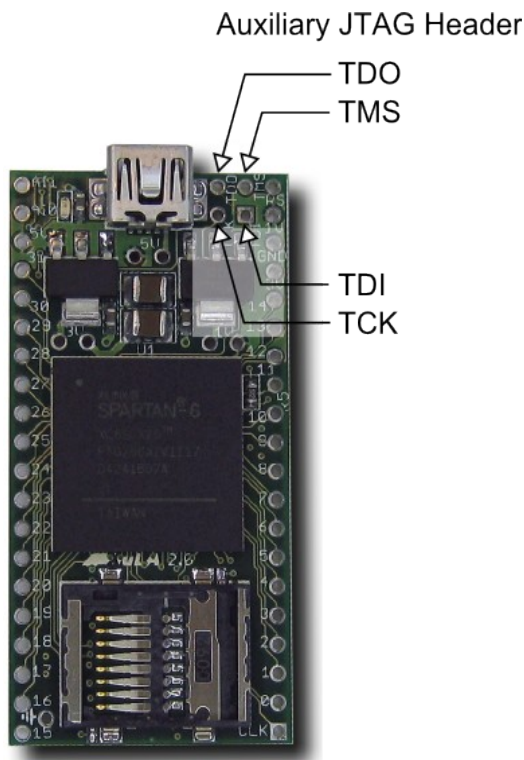
Or you can use the Xilinx PlanAhead tool to set the I/O standard for the pins like so:



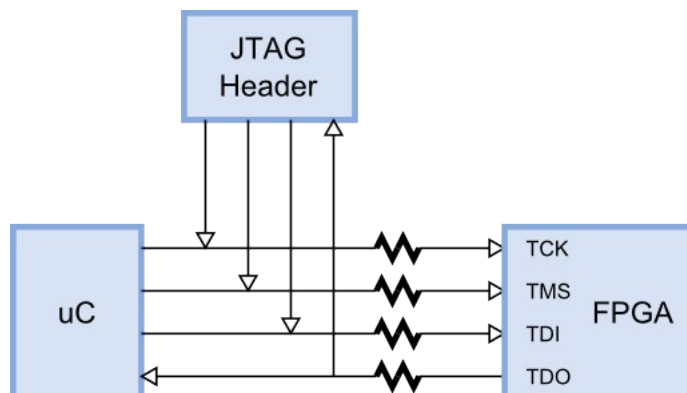


## Auxiliary JTAG Header

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

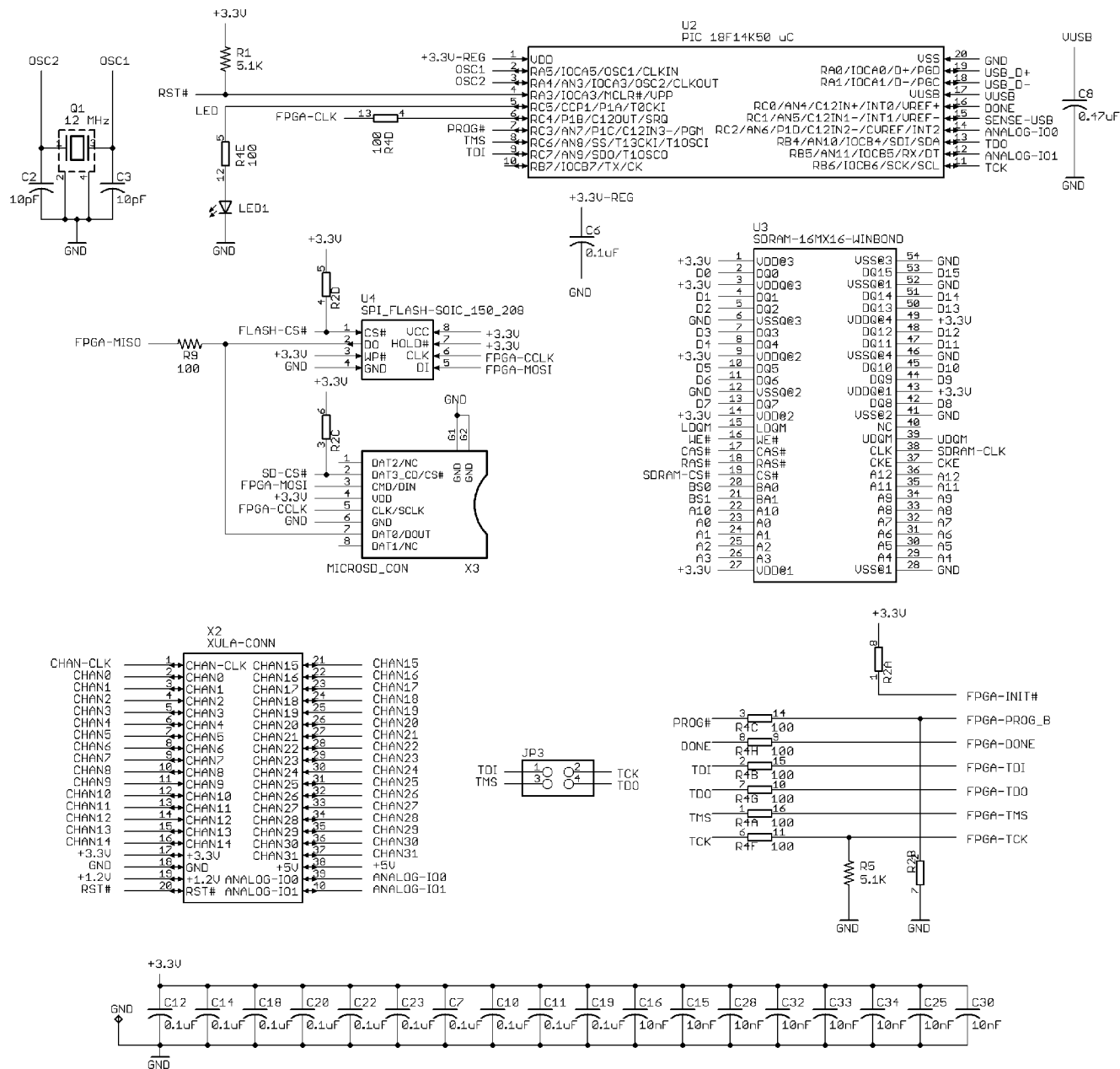
Net Name	FPGA	SDRAM	Prototyping Header	SPI Flash	microSD	uC	Notes
A0	E4	A0					
A1	E3	A1					
A2	D3	A2					
A3	C3	A3					
A4	B12	A4					
A5	A12	A5					
A6	D12	A6					
A7	E12	A7					
A8	G16	A8					
A9	G12	A9					
A10	F4	A10					
A11	G11	A11					
A12	H13	A12					
BS0	H3	BS0					
BS1	G3	BS1					
CAS#	L3	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*

---



2012, XESS Corp. Some rights reserved.



Licensed under a Creative Commons Attribution-ShareAlike Unported License. See [creativecommons.org/licenses/by-sa/3.0/](http://creativecommons.org/licenses/by-sa/3.0/)

## Schematic

PROJECT: XULA2

REV: 2.0

Date: not saved!

AUTHOR: Dave Vandenbout / XESS Corp.

Sheet: 2/3

U1  
XC6SLX25-FT256

## CONFIG

FPGA-PROG\_B I2 → PROGRAM\_B\_2  
FPGA-INIT# P13 → IO\_L65P\_INIT\_B\_2  
FPGA-DONE T11 → DONE\_2  
+3.3V N14 → IO\_L1N\_M0\_CMPMISO\_2  
GND C14 → IO\_L13P\_M1\_2  
GND C4 → IO\_L1P\_FSWAPEN\_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 I7 → IO\_L32N\_GCLK28\_2  
CHAN0 R7 → IO\_L32P\_GCLK29\_2  
CHAN1 R15 → IO\_L49P\_M1DQ10\_1  
CHAN2 R16 → IO\_L49N\_M1DQ11\_1  
CHAN3 M15 → IO\_L46P\_FCS\_B\_M1DQ2\_1  
CHAN4 K15 → IO\_L46N\_FOE\_B\_M1DQ3\_1  
CHAN5 K16 → IO\_L44P\_A3\_M1DQ6\_1  
CHAN6 K16 → IO\_L44N\_A2\_M1DQ7\_1  
CHAN7 J16 → IO\_L43N\_GCLK4\_M1DQ5\_1  
CHAN8 J14 → IO\_L43P\_GCLK5\_M1DQ4\_1  
CHAN9 F15 → IO\_L35P\_A11\_M1A7\_1  
CHAN10 F16 → IO\_L35N\_A10\_M1A2\_1  
CHAN11 C16 → IO\_L33N\_A14\_M1A4\_1  
CHAN12 B14 → IO\_L33P\_A15\_M1A10\_1  
CHAN13 B15 → IO\_L29N\_A22\_M1A14\_1  
CHAN14 B15 → IO\_L29P\_A23\_M1A13\_1  
CHAN15 T4 → IO\_L63N\_2  
CHAN16 R2 → IO\_L32P\_M3DQ14\_3  
CHAN17 R1 → IO\_L32N\_M3DQ15\_3  
CHAN18 H2 → IO\_L35P\_M3DQ10\_3  
CHAN19 H1 → IO\_L35N\_M3DQ11\_3  
CHAN20 K3 → IO\_L42P\_GCLK25\_TRDY2\_M3UDM\_3  
CHAN21 H1 → IO\_L42N\_GCLK24\_M3LDM\_3  
CHAN22 H2 → IO\_L39N\_M3LDQSN\_3  
CHAN23 H2 → IO\_L39P\_M3LDQ5\_3  
CHAN24 F1 → IO\_L41N\_GCLK26\_M3DQ5\_3  
CHAN25 F2 → IO\_L41P\_GCLK27\_M3DQ4\_3  
CHAN26 F1 → IO\_L46N\_M3CLKN\_3  
CHAN27 F2 → IO\_L46P\_M3CLK\_3  
CHAN28 B1 → IO\_L50P\_M3J4E\_3  
CHAN29 B2 → IO\_L50N\_M3BA2\_3  
CHAN30 B2 → IO\_L32P\_M3A8\_3  
CHAN31 A2 → IO\_L52N\_M3A9\_3

## FLASH

FPGA-CLK A8 → IO\_L34N\_GCLK18\_0

SD-CS# T8 → IO\_L30N\_GCLK0\_USERCCCLK\_2  
FLASH-CS# T9 → IO\_L65N\_CS0\_B\_2  
FPGA-CCLK R11 → IO\_L1P\_CCLK\_2  
FPGA-MOSI T10 → IO\_L3N\_MOSI\_CSI\_B\_MISO0\_2  
FPGA-MISO P10 → IO\_L3P\_D0\_DIN\_MISO\_MISO1\_2

CKE J12 → IO\_L40N\_GCLK10\_M1A6\_1  
SDRAM-CLK K11 → IO\_L42N\_GCLK6\_TRDY1\_M1LDM\_1  
SDRAM-CS# K12 → IO\_L42P\_GCLK7\_M1UDM\_1  
SDRAM-CLK H4 → IO\_L41P\_GCLK21\_M3A5\_3  
RAS# L3 → IO\_L45P\_M3A3\_3  
CAS# H3 → IO\_L36P\_M3DQ0\_3  
WE# E4 → IO\_L54P\_M3RESET\_3  
A0 E3 → IO\_L54N\_M3A11\_3  
A1 E3 → IO\_L54N\_M3A11\_3  
A2 E3 → IO\_L49P\_M3A7\_3  
A3 C3 → IO\_L48P\_M3BA0\_3  
A4 B12 → IO\_L62P\_0  
A5 A12 → IO\_L62N\_UREF\_0  
A6 D12 → IO\_L65N\_SCP0\_0  
A7 E12 → IO\_L1N\_A24\_UREF\_1  
A8 G12 → IO\_L36N\_A8\_M1BA1\_1  
A9 F12 → IO\_L38P\_A5\_M1CLK\_1  
A10 F4 → IO\_L53P\_M3CKE\_3  
A11 G14 → IO\_L30N\_A20\_M1A11\_1  
A12 H13 → IO\_L39P\_M1A3\_1  
BS1 G3 → IO\_L40P\_M3DQ6\_3  
BS0 H3 → IO\_L44N\_GCLK20\_M3A6\_3  
D0 P6 → IO\_L47P\_2  
D1 T6 → IO\_L47N\_2  
D2 P6 → IO\_L48N\_RDWR\_B\_UREF\_2  
D3 R6 → IO\_L49N\_D1\_2  
D4 P4 → IO\_L48P\_D7\_2  
D5 N6 → IO\_L49P\_D3\_2  
D6 P4 → IO\_L63P\_2  
D7 N4 → IO\_L2N\_3  
D8 R12 → IO\_L12N\_D2\_MISO3\_2  
D9 T12 → IO\_L52P\_M1DQ14\_1  
D10 T14 → IO\_L51N\_M1DQ13\_1  
D11 R14 → IO\_L51P\_M1DQ12\_1  
D12 T15 → IO\_L50P\_M1UDQ5\_1  
D13 T15 → IO\_L50N\_M1UDQ5N\_1  
D14 T12 → IO\_L52N\_M1DQ15\_1  
D15 P14 → IO\_L13N\_D10\_2  
LDQM H4 → IO\_L1P\_3  
UDQM L13 → IO\_L53N\_UREF\_1

## SDRAM

## POWER & GROUND

GND A1 → GND0819  
GND T1 → GND0822  
GND G2 → GND0818  
GND L2 → GND0820  
GND R3 → GND0821  
GND F4 → GND0812  
GND R6 → GND0816  
GND B7 → GND0813  
GND H7 → GND0814  
GND K7 → GND083  
GND G8 → GND0815  
GND J8 → GND089  
GND H9 → GND0825  
GND L9 → GND0811  
GND K9 → GND0823  
GND R10 → GND0817  
GND G10 → GND086  
GND B11 → GND088  
GND H12 → GND087  
GND D13 → GND0826  
GND N13 → GND0821  
GND G14 → GND081  
GND L15 → GND082  
GND A16 → GND0810  
GND T16 → GND084  
+3.3V D2 → UCC0\_303  
+3.3V J2 → UCC0\_304  
+3.3V N2 → UCC0\_305  
+3.3V B4 → UCC0\_002  
+3.3V G4 → UCC0\_302  
+3.3V K4 → UCC0\_301  
+3.3V E5 → UCC0\_203  
+3.3V H6 → UCCAUX01  
+3.3V H6 → UCCAUX07  
+3.3V L6 → UCCAUX05  
+3.3V K7 → UCC0\_009  
+3.3V N7 → UCC0\_201  
+3.3V F8 → UCCAUX05  
+3.3V R8 → UCC0\_204  
+3.3V B9 → UCC0\_001  
+3.3V L9 → UCCAUX04  
+3.3V D10 → UCC0\_003  
+3.3V G10 → UCCAUX06  
+3.3V J10 → UCCAUX03  
+3.3V N10 → UCC0\_202  
+3.3V F11 → UCCAUX02  
+3.3V G13 → UCC0\_004  
+3.3V K13 → UCC0\_102  
+3.3V R13 → UCC0\_101  
+3.3V D15 → UCC0\_106  
+3.3V J15 → UCC0\_104  
+3.3V J15 → UCC0\_105  
+3.3V N15 → UCC0\_103  
+1.2V G2 → UCCINT05  
+1.2V J2 → UCCINT01  
+1.2V H9 → UCCINT04  
+1.2V G9 → UCCINT02  
+1.2V J9 → UCCINT03  
+1.2V H10 → UCCINT06  
+1.2V K10 → UCCINT07

## UNUSED

D4 → IO\_L49N\_M3A2\_3  
G4 → IO\_L40N\_M3DQ7\_3  
J4 → IO\_L38N\_M3DQ3\_3  
K4 → IO\_L37N\_M3DQ1\_3  
L4 → IO\_L36N\_M3DQ9\_3  
N4 → IO\_L34N\_M3UDQ5N\_3  
P4 → IO\_L33N\_M3DQ13\_3  
A1 → IO\_L1N\_UREF\_0  
A2 → IO\_L48N\_M3BA1\_3  
A3 → IO\_L37P\_M3DQ0\_3  
A4 → IO\_L33P\_M3DQ12\_3  
A5 → IO\_L83N\_UREF\_3  
B3 → IO\_L83P\_3  
F3 → IO\_L83N\_M3A12\_3  
J3 → IO\_L38P\_M3DQ2\_3  
N3 → IO\_L34P\_M3UDQ5\_3  
A6 → IO\_L2N\_0  
B6 → IO\_L2P\_0  
C6 → IO\_L3N\_0  
D6 → IO\_L3P\_0  
F6 → IO\_L55N\_M3A14\_3  
G6 → IO\_L51N\_M3A1\_3  
H6 → IO\_L43N\_GCLK22\_TRDY2\_M3CASN\_3  
K6 → IO\_L47P\_M3A0\_3  
L6 → IO\_L45N\_M3DQ1\_3  
A7 → IO\_L2P\_3  
A8 → IO\_L4N\_0  
A9 → IO\_L4P\_0  
B7 → IO\_L7N\_0  
C7 → IO\_L7P\_0  
D7 → IO\_L5N\_0  
F7 → IO\_L85P\_M3A13\_3  
G7 → IO\_L51P\_M3A10\_3  
J7 → IO\_L43P\_GCLK23\_M3RASN\_3  
K7 → IO\_L47N\_M3A1\_3  
N7 → IO\_L64P\_D8\_2  
A8 → IO\_L64N\_D9\_2  
B8 → IO\_L6N\_0  
C8 → IO\_L6P\_0  
D8 → IO\_L36P\_GCLK15\_0  
E8 → IO\_L5P\_0  
F8 → IO\_L62N\_D6\_2  
G8 → IO\_L31N\_GCLK30\_D15\_2  
H8 → IO\_L31P\_GCLK31\_D14\_2  
J8 → IO\_L33N\_0  
K8 → IO\_L33P\_0  
L8 → IO\_L38N\_UREF\_0  
N8 → IO\_L38P\_0  
P8 → IO\_L36N\_GCLK14\_0  
A9 → IO\_L62P\_D5\_2  
B9 → IO\_L29N\_GCLK2\_2  
C9 → IO\_L30P\_GCLK1\_D13\_2  
D9 → IO\_L34P\_GCLK18\_0  
E9 → IO\_L40N\_0  
F9 → IO\_L40P\_0  
G9 → IO\_L29P\_GCLK3\_2  
H9 → IO\_L14P\_D11\_2  
J9 → IO\_L14N\_D12\_2  
K9 → IO\_L23P\_2  
L9 → IO\_L23N\_2  
N9 → IO\_L35N\_GCLK16\_0  
P9 → IO\_L35P\_GCLK17\_0  
A10 → IO\_L37N\_GCLK12\_0  
B10 → IO\_L37P\_GCLK12\_0  
C10 → IO\_L37P\_GCLK12\_0  
D10 → IO\_L61P\_SCP9\_0  
E10 → IO\_L6P\_2  
F10 → IO\_L16N\_UREF\_2  
G10 → IO\_L39N\_0  
H10 → IO\_L39P\_0  
J10 → IO\_L66P\_SCP1\_0  
K10 → IO\_L64N\_SCP4\_0  
L10 → IO\_L39N\_A1\_M1CLKN\_1  
N10 → IO\_L40P\_GCLK11\_M1A5\_1  
P10 → CHPCS\_B\_2  
A11 → IO\_L2N\_CHPMOSI\_2  
B11 → IO\_L30P\_A21\_M1RESET\_1  
C11 → IO\_L53P\_1  
D11 → IO\_L2P\_CHPCLK\_2  
E11 → IO\_L12P\_D1\_MISO2\_2  
F11 → IO\_L63N\_SCP6\_0  
G11 → IO\_L63P\_SCP7\_0  
H11 → IO\_L1P\_A25\_1  
J11 → IO\_L32P\_A17\_M1A8\_1  
K11 → IO\_L41P\_GCLK9\_TRDY1\_M1RASN\_1  
L11 → IO\_L74P\_AWAKE\_1  
N11 → IO\_L65N\_SCP2\_0  
P11 → IO\_L65P\_SCP3\_0  
A12 → IO\_L31P\_A19\_M1CKE\_1  
B12 → IO\_L32N\_A16\_M1A9\_1  
C12 → IO\_L36P\_A9\_M1BA0\_1  
D12 → IO\_L39N\_M1DQ1\_1  
E12 → IO\_L41N\_GCLK8\_M1CASN\_1  
F12 → IO\_L47P\_FWE\_B\_M1DQ0\_1  
G12 → IO\_L74N\_DOUT\_BUSY\_1  
H12 → IO\_L45P\_A1\_M1DQ5\_1  
J12 → IO\_L34P\_A13\_M1WE\_1  
K12 → IO\_L37P\_A7\_M1A0\_1  
L12 → IO\_L48P\_HDC\_M1DQ8\_1  
N12 → IO\_L31N\_A18\_M1A12\_1  
P12 → IO\_L34N\_A12\_M1BA2\_1  
A13 → IO\_L37N\_A6\_M1A1\_1  
B13 → IO\_L47N\_LOC\_M1DQ1\_1  
C13 → IO\_L45N\_A0\_M1UDQ5N\_1  
D13 → IO\_L48N\_M1DQ9\_1

### FPGA Connections

PROJECT: XuLA2

REV: 2.0

Date: not saved!

AUTHOR: Dave Vandenbout / XESS Corp.

Sheet: 3/3