XuLA Board V1.0 User Manual

How to install, test and use your new XuLA Board

XuLA Board V1.0 User Manual
MAN001 (V1.0) December 16, 2010

The following table shows the revision history for this document.

Date	Version	Revision
12/16/10	1.0	Initial release for XuLA Board V1.0.

Table of Contents

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 XuLA Board 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 the XS Boards,

  - a place to sign-up for our email forum where you can post questions to other XS Board users.

- 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 XuLA Board is not 5V-tolerant. **Do not connect 5V logic signals to the prototyping header.**

- The XuLA 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 XuLA Board.

Packing List

Here is what you should have received in your package:

- a XuLA Board.

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 XuLA Board. These utilities (long 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 XuLA Board to a PC

The XuLA Board 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 XuLA Board will light up as soon as it establishes a connection with the PC.

Testing Your XuLA Board

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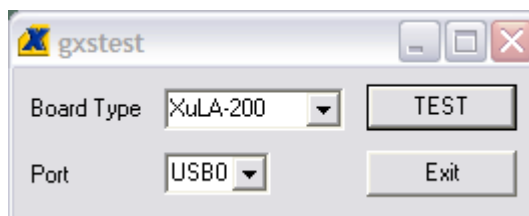
Next, select the type of board you are testing and the port that it's attached to. Then click on the TEST button. GXSTEST will configure the FPGA on your XuLA Board to perform a test procedure. Within a few seconds, a status window will appear informing you of the success or failure of the test.

If your XuLA Board 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 XuLA Board

The XuLA Board 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 XuLA Board through its prototyping header (not through the USB cable) as described here.

The locations of the shorting traces on the underside of the XuLA Board are shown below.
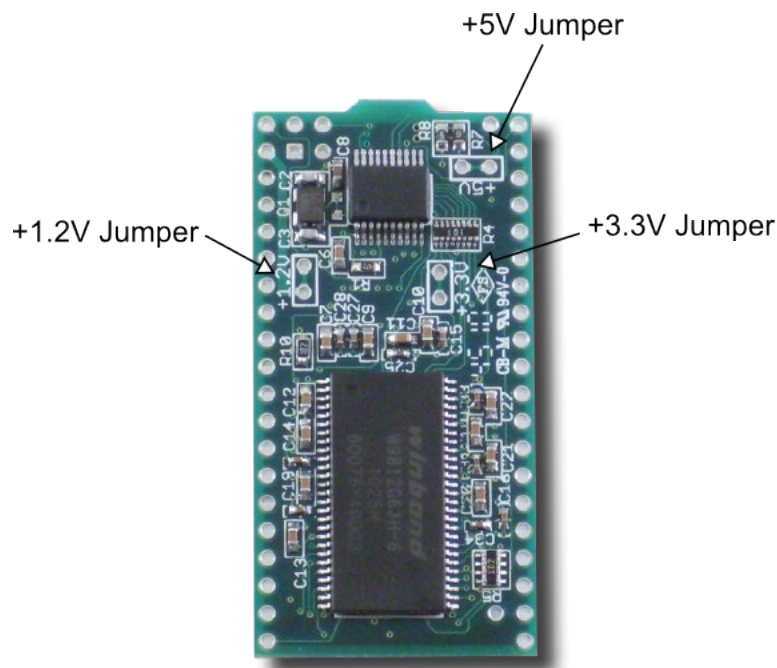
Applying Power to Your XuLA Board

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

■ receiving power through the USB connector, or

■ applying power through the XuLA prototyping header.

Applying Power Through the USB Port

Connecting the XuLA Board 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 XuLA'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 XuLA board. **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.**
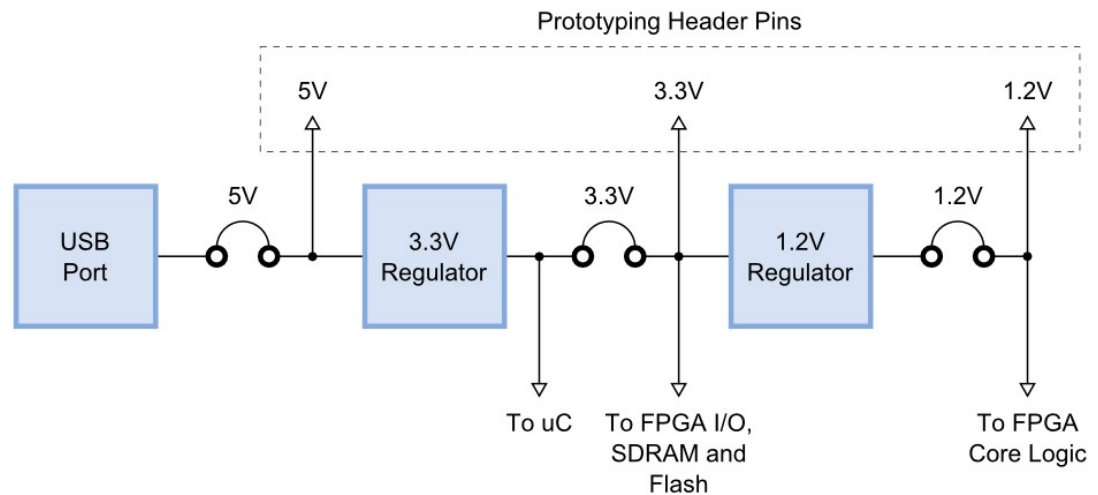
Inserting the XuLA Board into a Breadboard

In its factory-original configuration, the XuLA's prototyping header is empty. If desired, you can solder in a pair of twenty-pin headers and then insert the XuLA into a standard solderless breadboard as shown below.
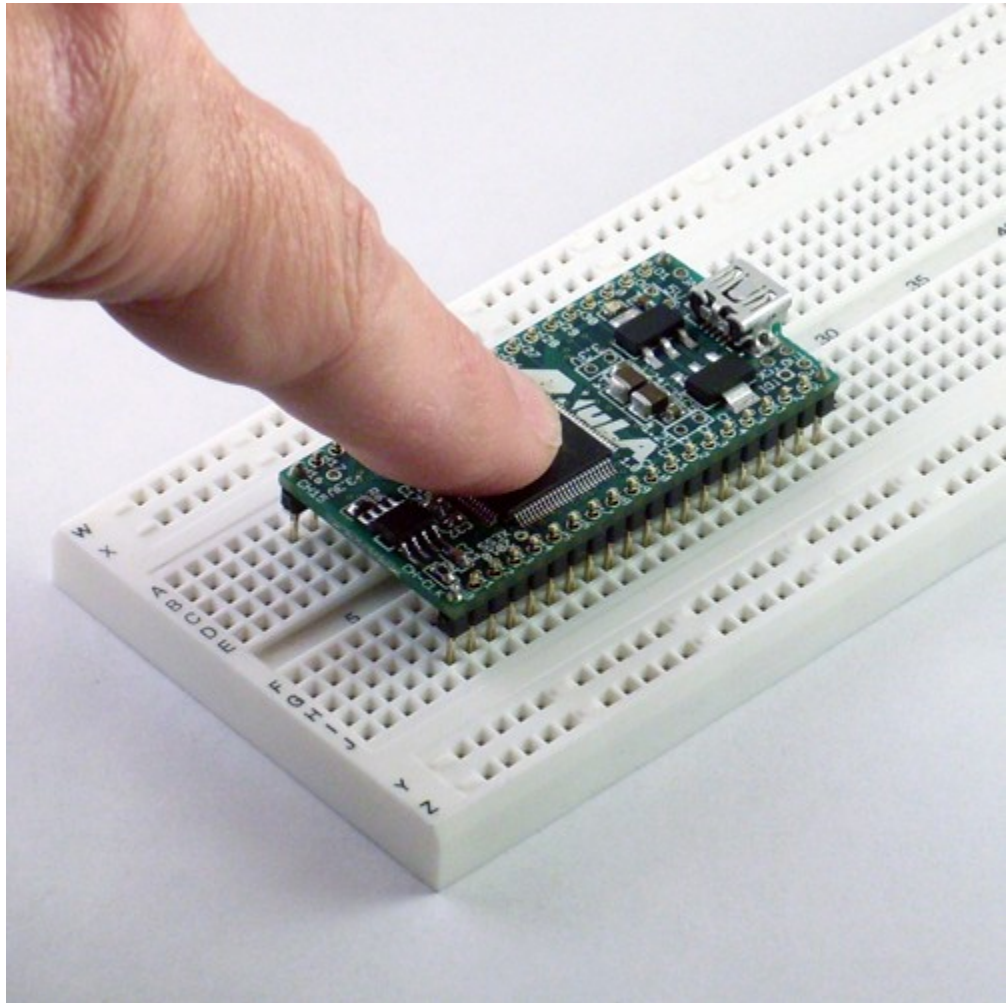
The XuLA Board will accept common headers with 0.025"-thick pins at 0.1" spacing, but you may find it difficult to remove the XuLA Board given how tightly the breadboard grips the pins. A header with thinner pins (such as the Aries 20-0600-20) is a better choice.

C.3 | *Programming*

This chapter will show you how to download logic designs into the FPGA of your XuLA Board 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 XuLA Board, you need to use the XILINX ISE *WebPACK* software to generate a bitstream (i.e., a .BIT file). 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 the **Process Properties...** from the context menu as shown below.
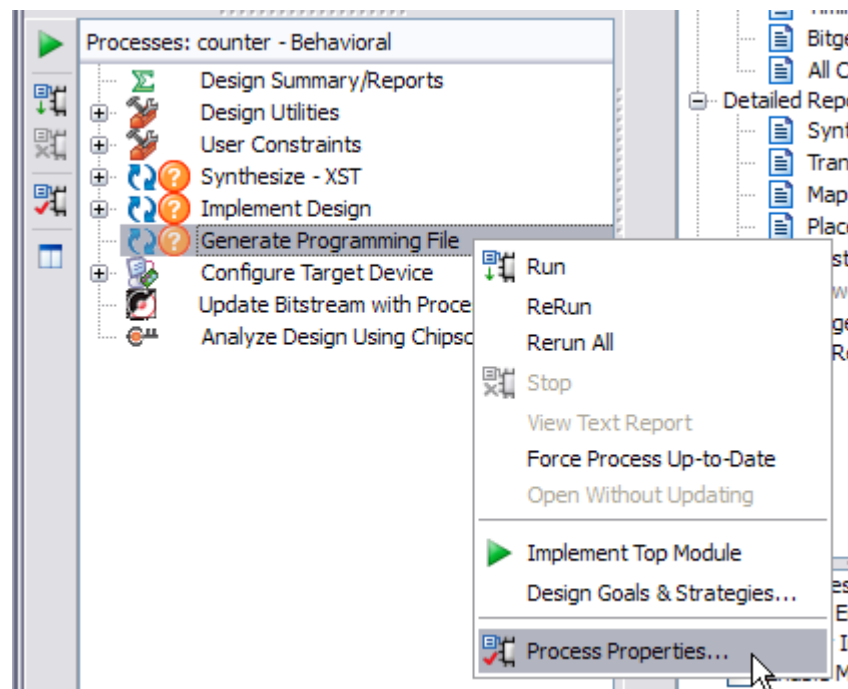
When the **Process Properties** window appears, select the **Configuration Options** category. Make sure the **Configuration Pin Done** property is set to the value **Pull Up**. This is necessary for the microcontroller on the XuLA to detect when the FPGA has successfully been configured by the bitstream.
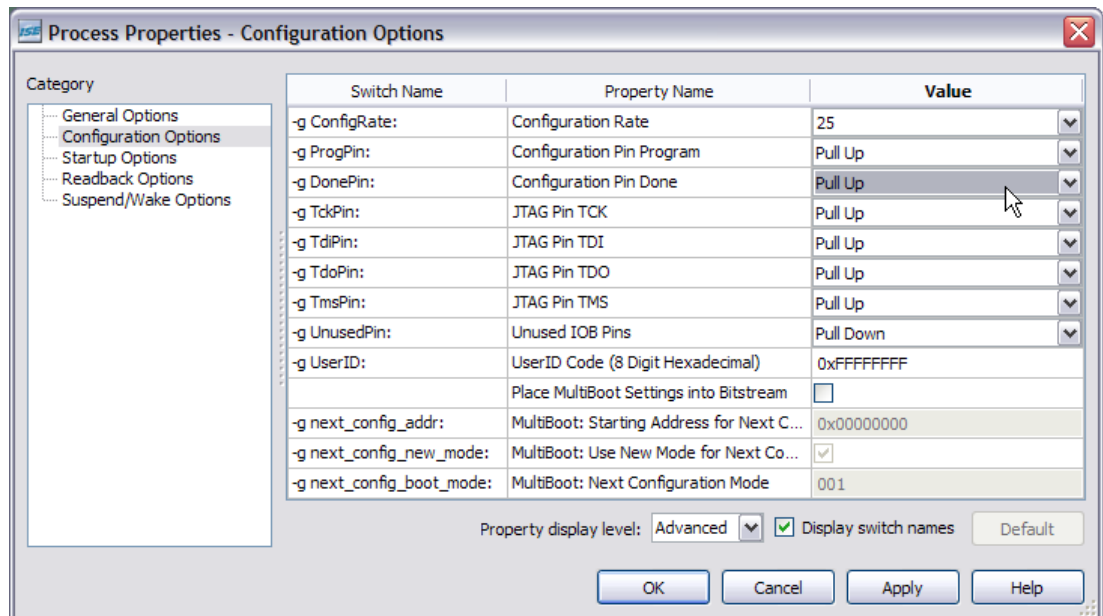
Next, select the **Startup Options** category and set the **FPGA Start-Up Clock** to one of the following values:

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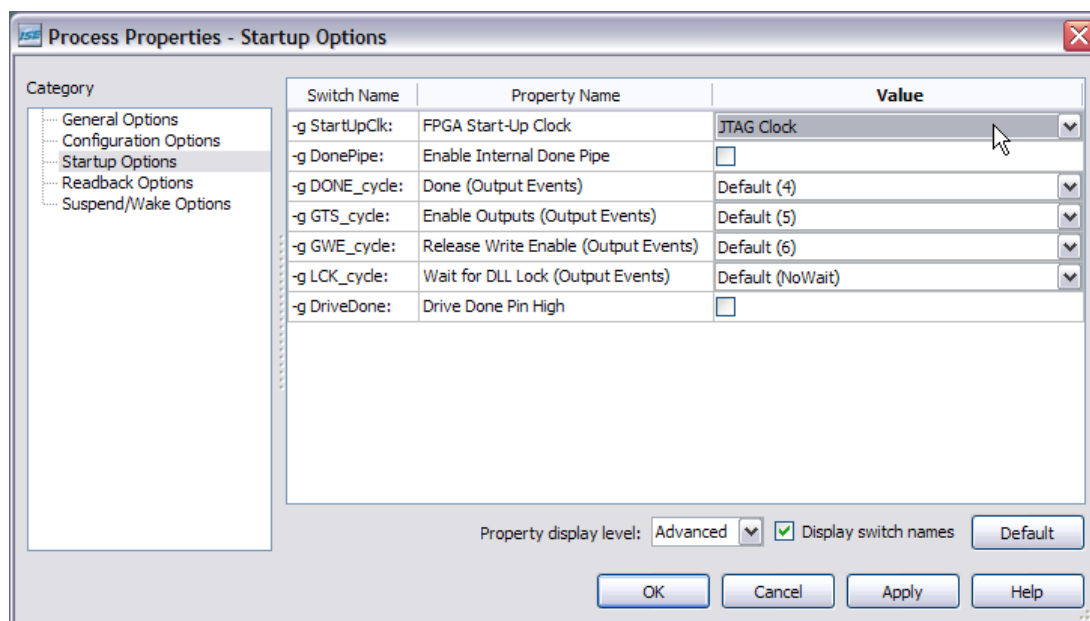
Downloading Bitstreams into the FPGA

Downloading Using GXSLOAD

As you develop and test a logic design, you will usually connect the XuLA Board to your PC and download the configuration bitstream to the FPGA each time you make changes. You can download a bitstream to the FPGA using the GXSLOAD utility.

Start GXSLOAD by double-clicking the icon placed on the desktop during the XSTOOLs installation. This brings up the window shown below.
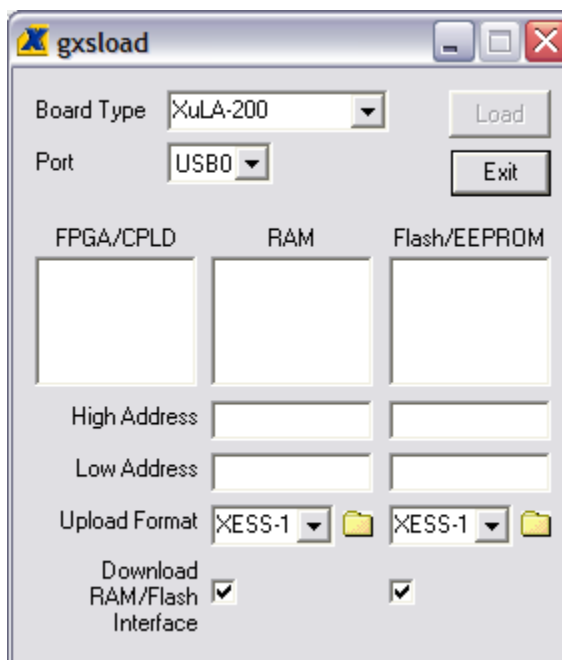
Now you can download bitstream files to the FPGA or CPLD simply by dragging them from their folder and dropping them into the **FPGA/CPLD** pane as shown below.
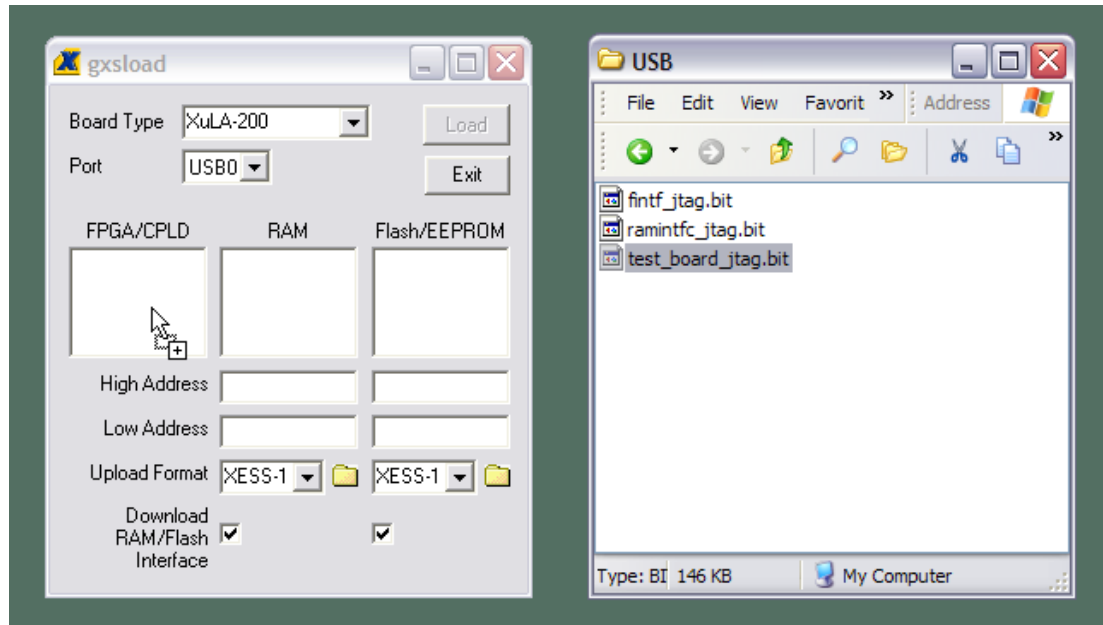
Once you drop the file, the highlighted file name appears in the **FPGA/CPLD** pane and the **Load** button is enabled. Clicking on the **Load** button will begin sending the bitstream to the XuLA Board through the USB cable. GXSLOAD will reject any non-downloadable files (ones with a suffix other than .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 XuLA Board 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.
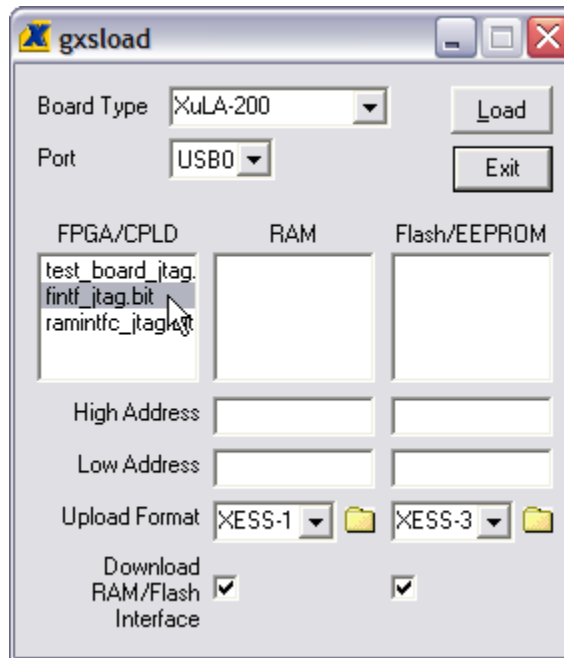
Double-clicking the highlighted file will deselect it so no file will be downloaded. Doing this disables the **Load** button.
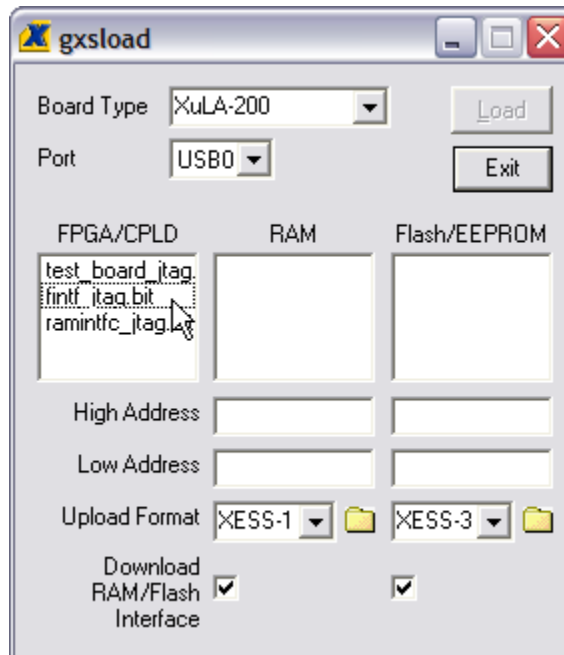
Downloading Using a XILINX or Third-Party JTAG Cable

You can use a XILINX or third-party JTAG cable to configure the FPGA on the XuLA Board by setting a non-volatile flag to give priority to the auxiliary JTAG header. First, connect the XuLA Board to your PC with a USB cable. Then double-click the GXSFLAGS icon so the following window appears.
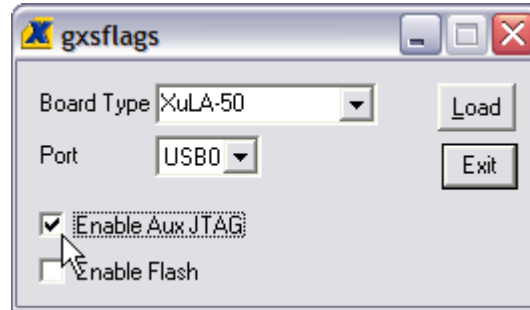
Click on the **Enable Aux JTAG** checkbox and then click on the **Load** button to set the flag. Then connect the JTAG cable to the auxiliary JTAG header. Now start the ISE *WebPACK* iMPACT tool to download bitstreams to the FPGA in boundary-scan mode. (Be aware that enabling the auxiliary JTAG header disables the functions of the other XSTOOLs utilities such as GXSTEST and GXSLOAD. You'll have to disable the auxiliary JTAG header to restore their functions.)

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 XuLA Board.

Storing Non-Volatile Bitstreams in the Flash

The FPGA on the XuLA Board stores its configuration in an on-chip SRAM which is erased whenever power is removed. Once your design is finished, you may want to store the bitstream in the serial Flash device on the XuLA Board. 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 GXSLOAD and click on the **Load** button. This activates the following sequence of events:

1. The FPGA on the XuLA Board is configured with an interface to transfer data from the USB port to the Flash device.

2. The entire Flash device is erased.

3. The contents of the .BIT file are downloaded into the Flash through the USB port.

Once the Flash download is complete, the FPGA will be configured with the stored bitstream whenever power is applied to the XuLA board. *(Make sure that you selected CCLK as the start-up clock 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 XuLA's Flash using GXSLOAD 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!*

Because of the FPGA's limited number of pins, only the SDRAM or Flash (not both) is accessible by the FPGA. In order to access the data that follows the bitstream in the Flash, you have to set a non-volatile flag to allow Flash access. First, connect the XuLA Board to your PC with a USB cable. Then double-click the GXSFLAGS icon so the following window appears.
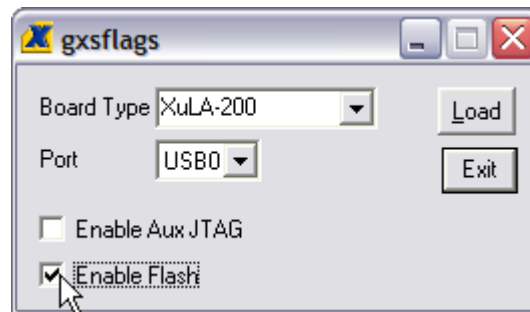
Click on the **Enable Flash** radio button and then click on the **Load** button to set the flag in the microcontroller. Now, any FPGA logic design you download to the XuLA Board can access the Flash (but not the SDRAM).

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.
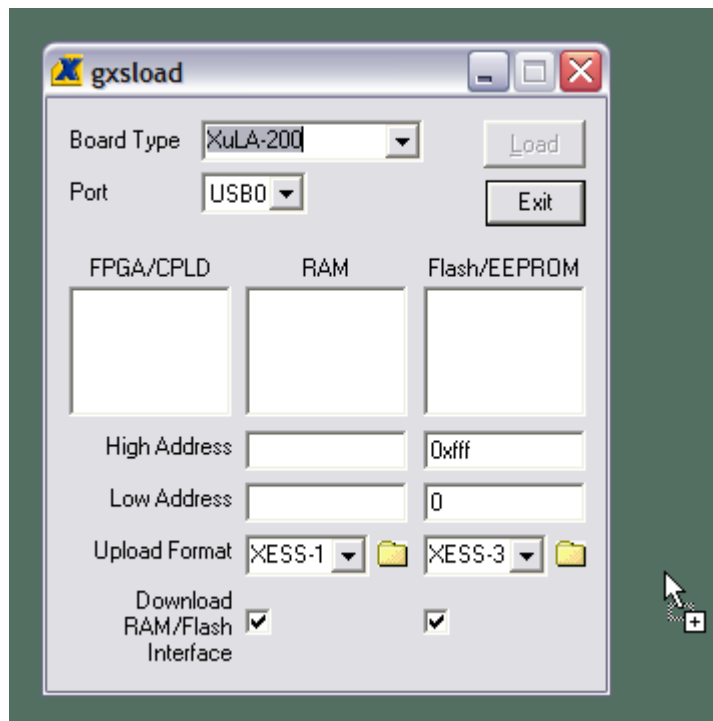
This activates the following sequence of steps:

1. The FPGA on the XuLA Board 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 XuLA Board contains a synchronous DRAM (SDRAM) whose contents can be downloaded and uploaded by GXSLOAD. 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 **gxsload** 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.
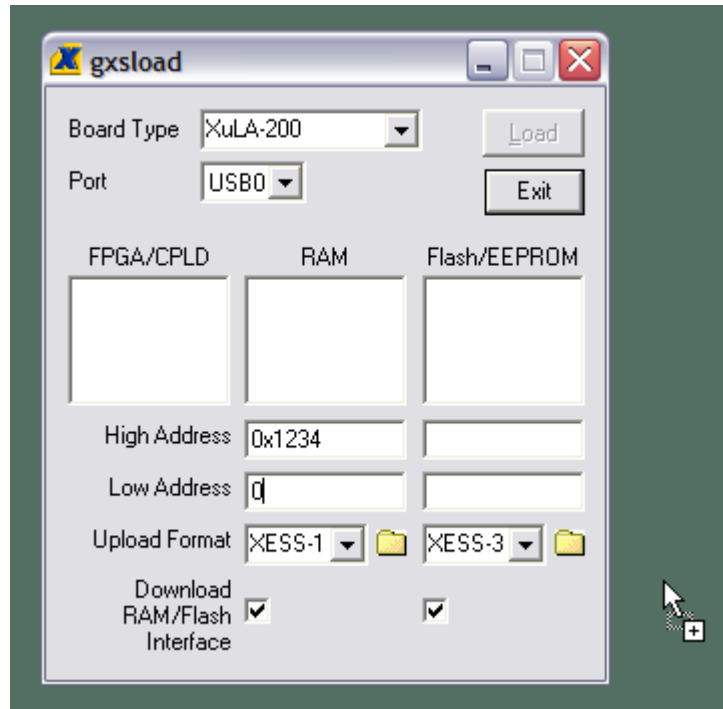
This activates the following sequence of steps:

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 XuLA Board 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.

XuLA Board Components

The XuLA Board contains the following major components:

FPGA: This is the programmable logic device.

Microcontroller: The microcontroller (uC) handles initialization, clock generation and USB-to-JTAG communications.

SDRAM: A 128-Mbit SDRAM provides volatile data storage accessible by the FPGA.

Flash: A 2-Mbit serial Flash device provides non-volatile storage for FPGA configuration bitstreams and 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 arrangement of these components is shown in the following block diagram.
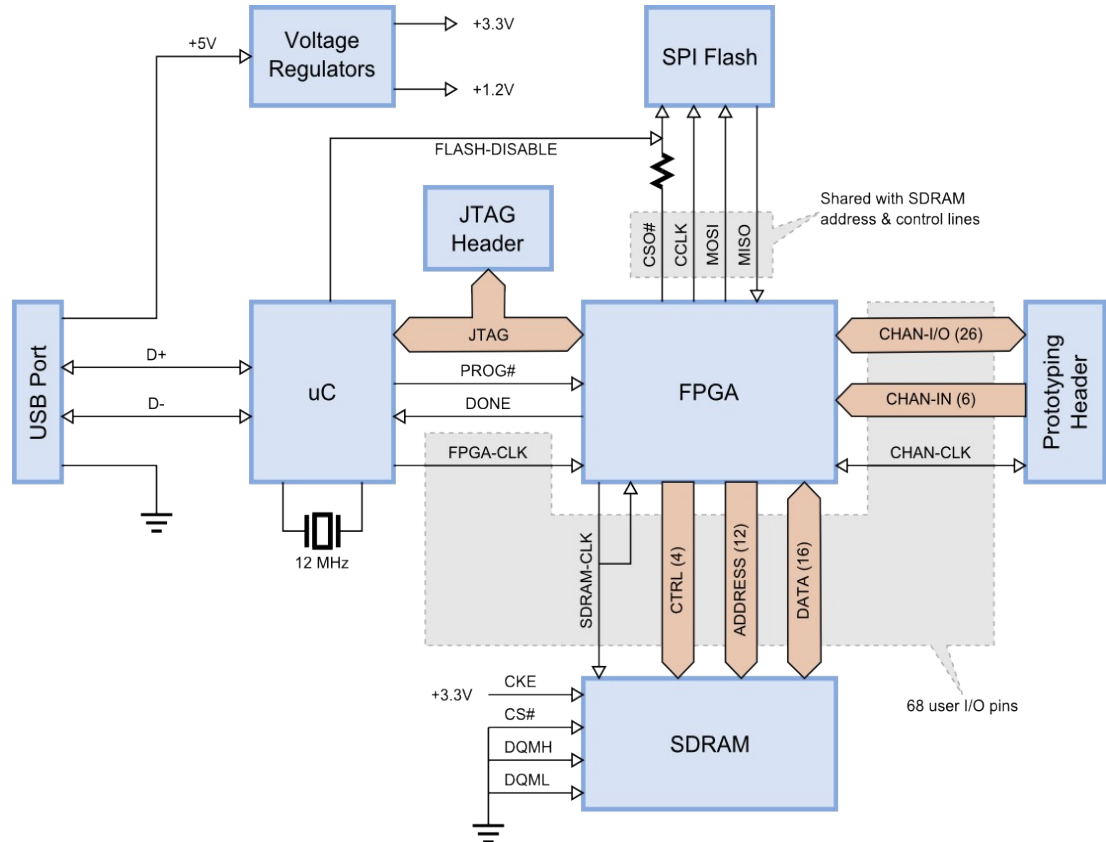
FPGA

The programmable logic device on the XuLA Board is either a XILINX 200,000-gate XC3S200A Spartan 3A FPGA in a 100-pin VQFP, or a 50,000-gate XC3S50A.

Microcontroller

The XuLA Board 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 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 XuLA Board incorporates an 8M x 16 SDRAM (Winbond W9812G6JH) that connects to the FPGA as shown below.
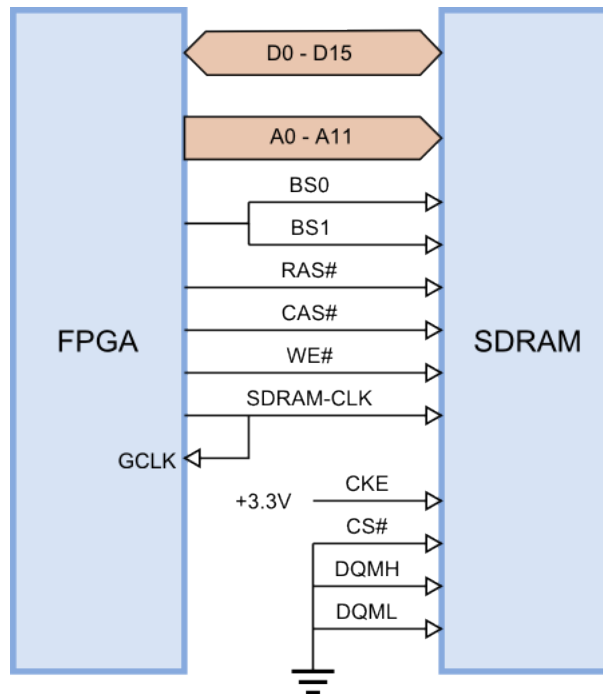
The FPGA applies the 12 MHz clock it receives from the microcontroller to one of its Digital Clock Managers (DCMs) to generate a higher frequency clock for the SDRAM. To compensate for delays to the SDRAM, the clock signal is re-routed back to a global clock input so the FPGA can synchronize its operations with those of the SDRAM.

Because of the limited number of pins provided by the FPGA's package, the following modifications were made to the SDRAM interface:

■ The chip-select (CS#) is tied low so the SDRAM is always enabled.

■ The clock-enable (CKE) is tied high so the SDRAM always requires a clock signal in order to keep its data refreshed.

■ Both data qualifier mask enables (DQMH and DQML) are tied low to disable byte-wide access to the SDRAM, so all operations involve the entire sixteen-bit data width.

■ A single FPGA pin drives both bank-select inputs (BS0 and BS1). **This reduces the accessible region of the SDRAM by half to 4M x 16.**

Flash

The XuLA Board has a 2-Mbit SPI Flash (Winbond W25X20BV) that connects to the FPGA as shown below.
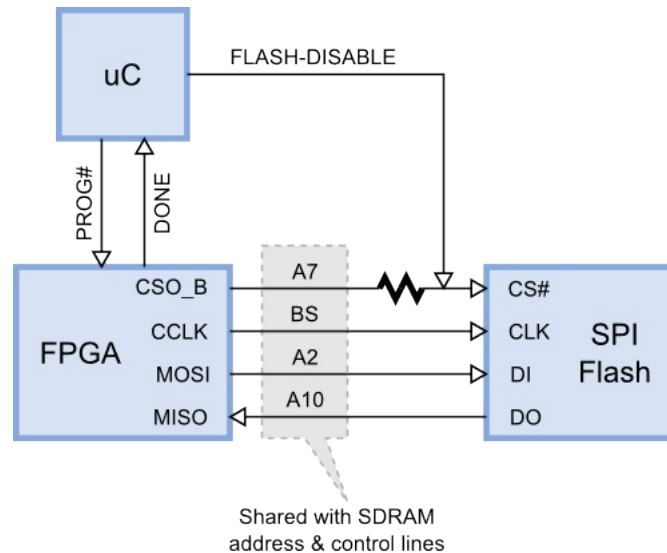
Due to the limited I/O of the FPGA package, the Flash and SDRAM chips share some of the FPGA pins. In combination with the SDRAM being permanently enabled (its chip-select is always asserted), this means that the Flash and SDRAM cannot be used simultaneously. Therefore, the following events occur when the XuLA Board comes on:

1. The microcontroller puts its FLASH-DISABLE output into a high impedance state and pulses the FPGA's PROG# pin to initiate the configuration from the Flash.

2. The FPGA lowers its CSO_B pin to enable the Flash, and then uses its CCLK, MOSI and MISO pins to read the stored bitstream. Because the SDRAM is not active during the FPGA's configuration (SDRAM-CLK is inactive since the FPGA is not configured), it doesn't matter if the SDRAM address and control pins are toggling.

3. Once the Flash delivers its bitstream, the FPGA's DONE pin is asserted. The microcontroller senses this and drives its output pin high to de-assert the FLASH-DISABLE signal, thus turning the Flash off. The FPGA can't access the Flash any more, but it can access the SDRAM without affecting the Flash.

It is possible to set a flag that prevents the microcontroller from disabling the Flash after the FPGA is configured (as described here). Then the logic design loaded into the FPGA can access the Flash, but it can no longer use the SDRAM (the toggling of the SDRAM address and control pins might inadvertently corrupt the Flash contents).

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.
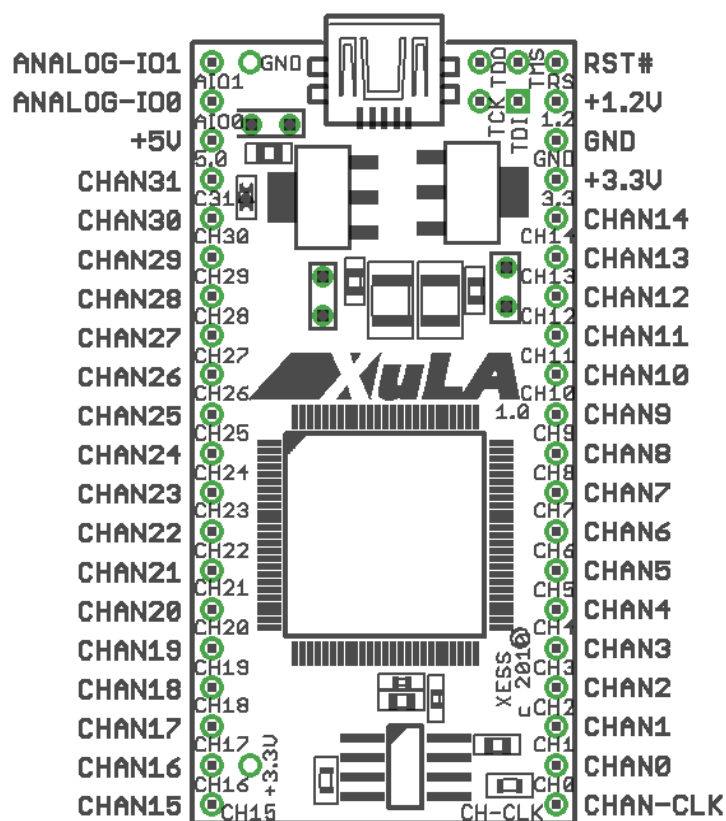
XuLA Header Pin Functions	
CHAN0 – CHAN31	These pins connect directly to the FPGA I/O pins. Channels 2, 9, 12, 19, 24 and 27 are input-only to the FPGA.
CHAN-CLK	This is a direct connection to a global clock pin of the FPGA. It can also be used as a general-purpose I/O pin.
+5V, +1.2V, +3.3V, GND	These pins connect to the voltage supply planes of the XuLA Board as depicted in this figure.
RST#	This pin connects to the microcontroller reset pin. Pulling it to ground and releasing it will cause the board 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.

Auxiliary JTAG Header

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

JTAG pins of the FPGA.
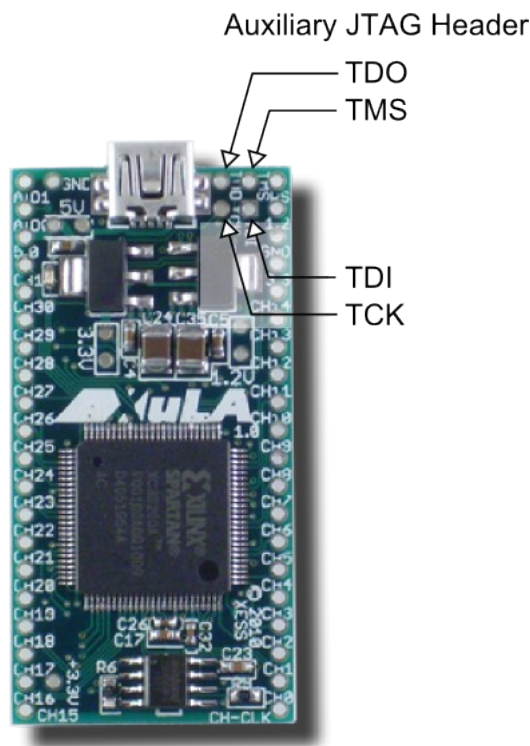
The connections of the JTAG header to the rest of the XuLA circuitry are as follows.
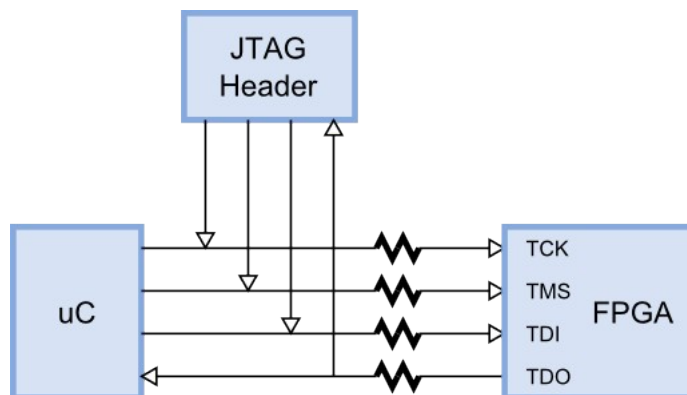
The current limiting resistors prevent the microcontroller or JTAG cable from damaging the FPGA if they are using a higher supply voltage than the FPGA.

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	Flash	uC	Notes
A0	49	A0				
A1	48	A1				
A2	46	A2		DI		
A3	31	A3				
A4	30	A4				
A5	29	A5				
A6	28	A6				
A7	27	A7		CS#		
A8	23	A8				
A9	24	A9				
A10	51	A10		DO		
A11	25	A11				
BS	53	BS0, BS1		CLK		SDRAM bank-selects connected
CAS#	60	CAS#				
CHAN-CLK	44		CHAN-CLK			
CHAN0	36		CHAN0			
CHAN1	37		CHAN1			
CHAN2	39		CHAN2			Input-only to FPGA
CHAN3	50		CHAN3			
CHAN4	52		CHAN4			
CHAN5	56		CHAN5			
CHAN6	57		CHAN6			
CHAN7	61		CHAN7			
CHAN8	62		CHAN8			
CHAN9	68		CHAN9			Input-only to FPGA
CHAN10	72		CHAN10			
CHAN11	73		CHAN11			
CHAN12	82		CHAN12			Input-only to FPGA
CHAN13	83		CHAN13			
CHAN14	84		CHAN14			
CHAN15	35		CHAN15			

Net Name	FPGA	SDRAM	Prototyping Header	Flash	uC	Notes
CHAN16	34		CHAN16			
CHAN17	33		CHAN17			
CHAN18	32		CHAN18			
CHAN19	21		CHAN19			Input-only to FPGA
CHAN20	20		CHAN20			
CHAN21	19		CHAN21			
CHAN22	13		CHAN22			
CHAN23	12		CHAN23			
CHAN24	7		CHAN24			Input-only to FPGA
CHAN25	4		CHAN25			
CHAN26	3		CHAN26			
CHAN27	97		CHAN27			Input-only to FPGA
CHAN28	94		CHAN28			
CHAN29	93		CHAN29			
CHAN30	89		CHAN30			
CHAN31	88		CHAN31			
D0	90	D0				
D1	77	D1				
D2	78	D2				
D3	85	D3				
D4	86	D4				
D5	71	D5				
D6	70	D6				
D7	65	D7				
D8	16	D8				
D9	15	D9				
D10	10	D10				
D11	9	D11				
D12	6	D12				
D13	5	D13				
D14	99	D14				
D15	98	D15				
DONE	54				I/O	
FPGA-CLK	43				I/O	12MHz from uC
PROG#	100				I/O	
RAS#	59	RAS#				
SDRAM-CLK	41	SDRAM-CLK				
SDRAM-CLKFB	40					SDRAM clock feedback
TCK	76				I/O	
TDI	2				I/O	
TDO	75				I/O	
TMS	1				I/O	
WE#	64	WE#				

A.2 Schematic

U2 — PIC 18F14K50 uC

Left pins:
- 1 VDD — +3.3V
- 2 RA5/IOCA5/OSC1/CLKIN — OSC1
- 3 RA4/AN3/IOCA3/OSC2/CLKOUT — OSC2
- 4 RA3/IOCA3/MCLR#/UPP — RST#
- 5 RC5/CCP1/P1A/T0CKI — LED
- 6 RC4/P1B/C12OUT/SRQ — FPGA-CLK (12 → 5)
- 7 RC3/AN7/P1C/C12IN3-/PGM — PROG#
- 8 RC6/AN8/SS/T13CKI/T1OSCI — TMS
- 9 RC7/AN9/SDO/T1OSCO — TDI
- 10 RB7/IOCB7/TX/CK — FLSH-DSBL

Right pins:
- 20 VSS — GND
- 19 RA0/IOCA0/D+/PGD — USB_D+
- 18 RA1/IOCA1/D-/PGC — USB_D-
- 17 VUSB — VUSB
- 16 RC0/AN4/C12IN+/INT0/VREF+ — DONE
- 15 RC1/AN5/C12IN1-/INT1/VREF- — SENSE-USB
- 14 RC2/AN6/P1D/C12IN2-/CUREF/INT2 — ANALOG-IO0
- 13 RB4/AN10/IOCB4/SDI/SDA — TDO
- 12 RB5/AN11/IOCB5/RX/DT — ANALOG-IO1
- 11 RB6/IOCB6/SCK/SCL — TCK

Oscillator
- OSC2, OSC1
- R2 0, Q1 12 MHz
- C2 10pF, C3 10pF
- GND

Reset / LED
- +3.3V, R1 5.1K, RST#, LED
- R4D 100, LED1, GND

JP3
- TDI 1 — 2 TCK
- TMS 3 — 4 TDO

FLSH-DSBL
- R6 47K, +3.3V

U5 — W25X20BV-SNIG/SSIG
- R5C 1.0K, A7 3 — 6
- A10 — R9 110
- 1 CS# 8 VCC +3.3V
- 2 DO   7 HOLD# +3.3V
- 3 WP# 6 CLK BS
- 4 GND 5 DI A2
- +3.3V, GND

JP1
Pin	Signal
1	CHAN-CLK
2	CHAN0
3	CHAN1
4	CHAN2
5	CHAN3
6	CHAN4
7	CHAN5
8	CHAN6
9	CHAN7
10	CHAN8
11	CHAN9
12	CHAN10
13	CHAN11
14	CHAN12
15	CHAN13
16	CHAN14
17	+3.3V
18	GND
19	+1.2V
20	RST#

JP2
Pin	Signal
1	CHAN15
2	CHAN16
3	CHAN17
4	CHAN18
5	CHAN19
6	CHAN20
7	CHAN21
8	CHAN22
9	CHAN23
10	CHAN24
11	CHAN25
12	CHAN26
13	CHAN27
14	CHAN28
15	CHAN29
16	CHAN30
17	CHAN31
18	+5V
19	ANALOG-IO0
20	ANALOG-IO1

JTAG resistor network
- PROG# R4F 100 — 6/11 — 100 PROG_B
- DONE R4C 100 — 3/14 — 51 DONE
- TDI R4H 100 — 8/9 — 2 TDI
- TDO R4B 100 — 12/15 — 75 TDO
- TMS R4G 100 — 7/10 — 1 TMS
- TCK R4A 100 — 4/16 — 76 TCK
- R5D 1.0K, R10 1.0K, GND GND
- GND 8 — 1.0K R5A — 1 A8
- GND 7 — 1.0K R5B — 2 A9

U3 — 8M×16 SDRAM
Left pins:
- 1 VDD@3 +3.3V
- 2 DQ0 D0
- 3 VDDQ@3 +3.3V
- 4 DQ1 D1
- 5 DQ2 D2
- 6 VSSQ@3 GND
- 7 DQ3 D3
- 8 DQ4 D4
- 9 VDDQ@2 +3.3V
- 10 DQ5 D5
- 11 DQ6 D6
- 12 VSSQ@2 GND
- 13 DQ7 D7
- 14 VDD@1 +3.3V
- 15 LDQM GND
- 16 WE# WE#
- 17 CAS# CAS#
- 18 RAS# RAS#
- 19 CS# GND
- 20 BS BS
- 21 BS1 BS
- 22 A10/AP A10
- 23 A0 A0
- 24 A1 A1
- 25 A2 A2
- 26 A3 A3
- 27 VDD@2 +3.3V

Right pins:
- 54 VSS@1 GND
- 53 DQ15 D15
- 52 VSSQ@1 GND
- 51 DQ14 D14
- 50 DQ13 D13
- 49 VDDQ@4 +3.3V
- 48 DQ12 D12
- 47 DQ11 D11
- 46 VSSQ@4 GND
- 45 DQ10 D10
- 44 DQ9 D9
- 43 VDDQ@1 +3.3V
- 42 DQ8 D8
- 41 VSS@2 GND
- 40 NC@2
- 39 UDQM GND
- 38 CLK SDRAM-CLK
- 37 CKE +3.3V
- 36 NC@1
- 35 A11 A11
- 34 A9 A9
- 33 A8 A8
- 32 A7 A7
- 31 A6 A6
- 30 A5 A5
- 29 A4 A4
- 28 VSS@3 GND

U1 — Spartan-3A FPGA

Power pins:
- +3.3V 96 VCCO_0@1, 79 VCCO_0@2, 67 VCCO_2@1, 45 VCCO_2@2, 26 VCCO_2@2, 11 VCCO_3
- +3.3V 22 VCCAUX@1, 92 VCCAUX@2, 55 VCCAUX@3
- +1.2V 38 VCCINT@1, 17 VCCINT@2, 66 VCCINT@3, 81 VCCINT@4
- GND 8 GND@1, 47 GND@2, 42 GND@3, 69 GND@4, 14 GND@5, 58 GND@6, 87 GND@7, 91 GND@8, 95 GND@9, 74 GND@10, 18 GND@11, 80 GND@12, 63 GND@13

Bank 3 / I/O:
- CHAN26 3 IO_L01P_3
- CHAN25 4 IO_L01N_3
- D13 5 IO_L02P_3
- D12 6 IO_L02N_3
- CHAN24 7 IP_3/VREF_3
- D11 9 IO_L03P_3/LHCLK0
- D10 10 IO_L03N_3/LHCLK1
- CHAN23 12 IO_L04P_3/LHCLK2
- CHAN22 13 IO_L04N_3/IRDY2/LHCLK3
- D9 15 IO_L05P_3/TRDY2/LHCLK6
- D8 16 IO_L05N_3/LHCLK7
- CHAN21 19 IO_L06P_3
- CHAN20 20 IO_L06N_3
- CHAN19 21 IP_3
- A8 23 IO_L01P_2/M1
- A9 24 IO_L02P_2/M2
- A11 25 IO_L01N_2/M0

Bank 2:
- A7 27 IO_L02N_2/CSO_B
- A6 28 IO_L03P_2/RDWR_B
- A5 29 IO_L03N_2/VS2
- A4 30 IO_L04P_2/VS1
- A3 31 IO_L04N_2/VS0
- CHAN18 32 IO_L05P_2
- CHAN17 33 IO_L05N_2
- CHAN16 34 IO_L06P_2/D7
- CHAN15 35 IO_L06N_2/D6
- CHAN0 36 IO_L07P_2/D5
- CHAN1 37 IO_L07N_2/D4
- CHAN2 39 IP_2/VREF_2
- SDRAM-CLK 40 IO_L08P_2/GCLK14
- SDRAM-CLK 41 IO_L08N_2/GCLK15
- FPGA-CLK 13 IO_L09P_2/GCLK0
- CHAN-CLK 11 IO_L09N_2/GCLK1
- A2 16 IO_2/MOSI/CSI_B
- A1 48 IO_L10P_2/INIT_B
- A0 49 IO_L10N_2/D3
- CHAN3 50 IO_L11P_2/D2

Bank 2 / 1:
- A10 51 IO_L12P_2/D0/DIN/MISO
- CHAN4 52 IO_L11N_2/D1
- BS 53 IO_L12N_2/CCLK
- CHAN5 56 IO_L01P_1
- CHAN6 57 IO_L01N_1
- RAS# 59 IO_L02P_1/RHCLK0
- CAS# 60 IO_L02N_1/RHCLK1
- CHAN7 61 IO_L03P_1/RHCLK2
- CHAN8 62 IO_L03N_1/TRDY1/RHCLK3
- WE# 64 IO_L04P_1/IRDY1/RHCLK6
- D7 65 IO_L04N_1/RHCLK7
- CHAN9 68 IP_1/VREF_1
- D6 70 IO_L05P_1
- D5 71 IO_L05N_1
- CHAN10 72 IO_L06P_1
- CHAN11 73 IO_L06N_1

Bank 0:
- D1 77 IO_L01P_0/VREF_0
- D2 78 IO_L01N_0
- CHAN12 82 IP_0/VREF_0
- CHAN13 83 IO_L02P_0/GCLK4
- CHAN14 84 IO_L02N_0/GCLK5
- D3 85 IO_L03P_0/GCLK6
- D4 86 IO_L03N_0/GCLK7
- CHAN31 88 IO_L04P_0/GCLK8
- CHAN30 89 IO_L04N_0/GCLK9
- D0 90 IO_0/GCLK11
- CHAN29 93 IO_L05P_0
- CHAN28 94 IO_L05N_0
- CHAN27 97 IP_0
- D15 98 IO_L06P_0/VREF_0
- D14 99 IO_L06N_0/PUDC_B

Decoupling (top row)
+3.3V / +3.3V
- C12 0.1uF, C6 0.1uF, C14 0.1uF, C18 0.1uF, C20 0.1uF, C22 0.1uF, C23 0.1uF, C7 0.1uF, C10 0.1uF, C11 0.1uF, C19 0.1uF, C16 10nF, C15 10nF, C28 10nF, C32 10nF, C33 10nF, C34 10nF, C25 10nF, C30 10nF
- GND

Power supply
- SENSE-USB, R8 680K, R7 330K, GND
- 5V, X1 USB connector: 1 VCC, 2 D-, 3 D+, 4 ID, 5 GND
- USB_D-, USB_D+
- +5V
- C1 4.7uF
- U6 +3.3V Reg.: IN OUT, GND HS — 3.3V — +3.3V
- C4 4.7uF, C24 22uF
- U7 +1.2V Reg.: IN OUT, GND HS — 1.2V — +1.2V
- C5 4.7uF, C35 22uF
- GND

Decoupling (+1.2V)
- +1.2V
- C9 0.1uF, C13 0.1uF, C17 0.1uF, C21 0.1uF, C27 10nF, C29 10nF, C31 10nF, C26 10nF
- GND

Schematic
PROJECT: XuLA
REV: 1.0
Date: 12/2/2010 8:43:24 AM
AUTHOR: Dave Vandenbout / XESS Corp.
Sheet: 2/2