GX3500

User Configurable FPGA Board

GXFPGA Software

Includes: GX3501/GX3601, GX3509/GX3609, GX3510/GX3610, GX3540/GX3640, and GX3571/GX3671 Expansion Boards

User's Guide

Last Updated: December 16, 2016

Safety and Handling

Each product shipped by Marvin Test Solutions is carefully inspected and tested prior to shipping. The shipping box provides protection during shipment, and can be used for storage of both the hardware and the software when they are not in use.

The circuit boards are extremely delicate and require care in handling and installation. Do not remove the boards from their protective plastic coverings or from the shipping box until you are ready to install the boards into your computer.

If a board is removed from the computer for any reason, be sure to store it in its original shipping box. Do not store boards on top of workbenches or other areas where they might be susceptible to damage or exposure to strong electromagnetic or electrostatic fields. Store circuit boards in protective anti-electrostatic wrapping and away from electromagnetic fields.

Be sure to make a single copy of the software diskette for installation. Store the original diskette in a safe place away from electromagnetic or electrostatic fields. Return compact disks (CD) to their protective case or sleeve and store in the original shipping box or other suitable location.

Warranty

Marvin Test Solutions products are warranted against defects in materials and workmanship for a period of 12 months. Software products and accessories are warranted for 3 months. Unless covered by software support or maintenance agreement. Marvin Test Solutions shall repair or replace (at its discretion) any defective product during the stated warranty period. The software warranty includes any revisions or new versions released during the warranty period. Revisions and new versions may be covered by a software support agreement. If you need to return a board, please contact Marvin Test Solutions Customer Technical Services department via http://www.marvintest.com/magic the Marvin Test Solutions on-line support system.

If You Need Help

Visit our web site at http://www.marvintest.com for more information about Marvin Test Solutions products, services and support options. Our web site contains sections describing support options and application notes, as well as a download area for downloading patches, example, patches and new or revised instrument drivers. To submit a support issue including suggestion, bug report or question please use the following link: http://www.marvintest.com/magic

You can also use Marvin Test Solutions technical support phone line (949) 263-2222. This service is available between 7:30 AM and 5:30 PM Pacific Standard Time.

Disclaimer

In no event, shall Marvin Test Solutions or any of its representatives be liable for any consequential damages whatsoever (including unlimited damages for loss of business profits, business interruption, loss of business information, or any other losses) arising out of the use of or inability to use this product, even if Marvin Test Solutions has been advised of the possibility for such damages.

Copyright

Trademarks

ATEasy®, CalEasy, DIOEasy®, DtifEasy, WaveEasy	Marvin Test Solutions (prior name is Geotest - Marvin Test Systems Inc.)
Quartus	Altera Corporation
C++ Builder, Delphi	Embarcadero Technologies Inc.
LabView, LabWindows^{tm}/CVI	National Instruments
Microsoft Developer Studio, Microsoft Visual C++, Microsoft Visual Basic, .NET, and Windows	Microsoft Corporation

All other trademarks are the property of their respective owners.

Table of Contents

Safety and Handling ... i
Warranty ... i
If You Need Help ... i
Disclaimer ... i
Copyright ... i
Trademarks ... ii
Table of Contents ... iii
Chapter 1 - Introduction ... 1
Manual Scope and Organization ... 1
Manual Scope ... 1
Manual Organization ... 1
Conventions Used in this Manual ... 1
Chapter 2 - Overview ... 3
Introduction ... 3
Features ... 3
Applications ... 3
Board Description ... 4
Architecture ... 5
FPGA PCI Timing ... 6
Specifications ... 7
Digital I/O Channel ... 7
Expansion Board Interface ... 7
Timing Source ... 7
User FPGA ... 7
Power ... 7
Environmental ... 7
Virtual Panel Description ... 8
Virtual Panel Initialize Dialog ... 9
Virtual Panel Setup Page ... 10
Virtual Panel I/O Page ... 11
Virtual Panel PIO Page ... 12
Virtual Panel About Page ... 13
Chapter 3 - Installation and Connections ... 15
Getting Started ... 15
Interfaces and Accessories ... 15

Packing List 15
Unpacking and Inspection....15
System Requirements....16
Installation of the GXFPGA Software....16
Setup Maintenance Program....16
Overview of the GXFPGA Software....17
Installation Folders....17
Configuring Your PXI System using the PXI/PCI Explorer....18
Board Installation....19
Before you Begin....19
Electric Static Discharge (ESD) Precautions....19
Installing a Board....19
Plug & Play Driver Installation....21
Removing a Board....21
Connectors and Jumpers....22
GX3500 Connectors – J1-J4 Flex I/O Banks (A-D)....25
GX3500 J1 – Flex I/O Bank A Connector....25
GX3500 J3 – Flex I/O Bank B Connector....26
GX3500 J4 – Flex I/O Bank C Connector....27
GX3500 J2 – Flex I/O Bank D Connector....28
Chapter 4 - Programming the Board....29
The GXFPGA Driver....29
Programming Using C/C++ Tools....29
Programming Using Visual Basic and Visual Basic .NET....29
Programming Using Pascal/Delphi....29
Programming GXFPGA Boards Using ATEasy®....30
Programming Using LabVIEW and LabVIEW/Real Time....30
Using and Programming under Linux....30
Using the GXFPGA driver functions....31
Initialization, HW Slot Numbers and VISA Resource....31
Board Handle....32
Reset....32
Error Handling....32
Driver Version....32
Programming Examples....32
Distributing the Driver....32
Chapter 5 - GXFPGA Tutorial and Example....33

Introduction....33
Downloading Altera Design FPGA Design Tools....33
Create New Project....34
Device Selection....34
Pin Assignment Setup....34
Creating Design File....37
Phase 1: Creating the FPGA design - 32 bit Full Adder....39
Components Used....39
Design....40
Phase 2: Creating the FPGA Design - 2 to 1 Clock Mux....48
Components Used....48
Design....48
Phase 3: Creating the FPGA Design - 32 bit Dynamic Digital Pattern Sequencer....49
Components Used....49
Design....49
Configure Project to Output SVF and RPD Files....53
Compile an Example Project and Build RPD and SVF Files....54
Load Gx3500 with SVF File....56
Testing the Design....57
Adder Testing....57
Clock Mux Testing....57
Digital Sequencer Testing....58
Chapter 6 - GX3500 Expansion Boards....59
Expansion Board Design Guide....59
Board Description and Connectors....59
Mechanical Guide....61
Connectors and Electrical Requirements....63
P8 and P9 Host Connectors....64
P10 and P11 Connectors....68
P10 Connector....69
P11 Connector....71
Programming Support for Expansion Boards....73
GX3501, GX3509, GX3510 and GX3540 (PIO) Expansion Boards....75
Internal Connections....75
GX3601 J1-J4 Banks A-D Connectors....83
GX3601 J1 Flex I/O Bank A Connector....83
GX3601 J3 Flex I/O Bank B Connector....84

GX3601 J4 Flex I/O Bank C Connector 85
GX3601 J2 Flex I/O Bank D Connector 86
GX3609 and GX3610 J1-J4 Banks A-D Connectors 87
GX3609, GX3610 J1 Flex I/O Bank A Connector 87
GX3609, GX3610 J3 Flex I/O Bank B Connector 88
GX3609, GX3610 J4 Flex I/O Bank C Connector 89
GX3609, GX3610 J2 Flex I/O Bank D Connector 90
GX3501, GX3509 and GX3510 Programming 90
GX3501 TTL Buffer Expansion Board Specification 90
GX3509 Differential TTL Expansion Board Specification 91
GX3510 MLVDS Expansion Board Specification 91
GX3540 – 40 Channel ECL I/O Expansion Board 92
GX3540 Termination Resistors and Voltage Sources 92
GX3540 Connectors 92
GX3540 J1 – Flex I/O Bank A Inputs Connector 92
GX3540 J2 – Flex I/O Bank D Outputs Connector 93
GX3540 Programming 93
GX3540 Specification 93
GX3571 – Video Generator Board 94
Features 94
Programming 94
Specifications 94
Chapter 7 - Function Reference 95
Introduction 95
GXFPGA Functions 96
GxFpga3571GetAdvancedMode 98
GxFpga3571GetVideoMode 99
GxFpga3571LoadArray 100
GxFpga3571LoadColor 101
GxFpga3571LoadFile 102
GxFpga3571Reset 103
GxFpga3571ResetModeline 104
GxFpga3571SetAdvancedMode 105
GxFpga3571SetModeline 106
GxFpga3571SetVideoMode 107
GxFpgaDiscardEvents 108
GxFpgaGetBoardSummary 109

GxFpgaGetBoardType 110
GxFpgaGetEepromSummary 111
GxFpgaGetExpansionBoardBypass 112
GxFpgaGetExpansionBoardID 113
GxFpgaGetDriverSummary 114
GxFpgaGetErrorString 115
GxFpgaInitialize 117
GxFpgaInitializeVisa 118
GxFpgaLoad 119
GxFpgaLoadFromEeprom 120
GxFpgaLoadStatus 121
GxFpgaLoadStatusMessage 122
GxFpgaPanel 123
GxFpgaPioGetChannel 124
GxFpgaPioGetChannelDirection 125
GxFpgaPioGetGroup 126
GxFpgaPioGetGroupDirection 127
GxFpgaPioReset 128
GxFpgaPioSetChannel 129
GxFpgaPioSetChannelDirection 130
GxFpgaPioSetGroup 131
GxFpgaPioSetGroupDirection 132
GxFpgaReadMemory 133
GxFpgaReadRegister 134
GxFpgaReset 135
GxFpgaSetEvent 136
GxFpgaSetExpansionBoardBypass 137
GxFpgaUpgradeFirmware 138
GxFpgaUpgradeFirmwareStatus 139
GxFpgaWaitOnEvent 140
GxFpgaWriteMemory 141
GxFpgaWriteRegister 142
Index 143

Chapter 1 - Introduction

Manual Scope and Organization

Manual Scope

The purpose of this manual is to provide all the necessary information to install, use, and maintain the GX3500 instrument. This manual assumes the reader has a general knowledge of PC based computers, Windows operating systems, and some understanding of digital I/O.

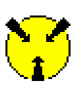
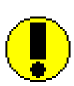
This manual also provides programming information using the GX3500 driver (referred in this manual **GXFPGA**). Therefore, good understanding of programming development tools and languages may be necessary.

Manual Organization

The GX3500 manual is organized in the following manner:

Chapter	**Content**
Chapter 1 - Introduction	Introduces the GX3500 manual. Lists all the supported board and shows warning conventions used in the manual.
Chapter 2 – Overview	Describes the GX3500 features, board description, its architecture, specifications and the panel description and operation.
Chapter 3 –Installation and Connections	Provides instructions on how to install a GX3500board and the GXFPGA software.
Chapter 4 – Programming the Board	Provides a list of the GXFPGA software driver files, general purpose and generic driver functions, and programming methods. Discusses supported application development tools and programming examples.
Chapter 5 – GXFPGA Tutorial and Example	Provides an example of how to use the Quartus II to design and FPGA and then load and test the design using the GXFPGA panel.
Chapter 6 – Expansion Boards	Describes how to design a GX3500 expansion board and describes several standard expansion boards available from Marvin Test Solutions.
Chapter 7 – Functions Reference	Provides a list of the GX3500 driver functions. Each function description provides syntax, parameters, and any special programming comments.

Conventions Used in this Manual

Symbol Convention	**Meaning**
	Static Sensitive Electronic Devices. Handle Carefully.
	Warnings that may pose a personal danger to your health. For example, shock hazard.
	Cautions where computer components may be damaged if not handled carefully.
	Tips that aid you in your work.

Formatting Convention	**Meaning**
`Monospaced Text`	Examples of field syntax and programming samples.
Bold type	Words or characters you type as the manual instructs. For example: function or panel names.
Italic type	Specialized terms. Titles of other reference books. Placeholders for items you must supply, such as function parameters

Chapter 2 - Overview

Introduction

The GX3500 is a user configurable FPGA 3U PXI card which offers 160 digital I/O signals for specific application needs. The card employs the Altera Cyclone III FPGA which can support clock rates up to 150 MHz and features over 55,000 logic elements and 2.34 Mb of memory. The GX3500 can also accept an expansion card assembly which can be used to customize the interface to the UUT – eliminating the need for additional external boards which are cumbersome and physically difficult to integrate into a test system. The design of the FPGA is done by using Altera's free Quartus II Web Edition tool set. Once the user has compiled the FPGA design, the image can be loaded into the FPGA via the PXI bus interface or via an on-board EEROM using the provided function library.

Features

The GX3500's four banks of 40 digital I/O signals can be selectively isolated from the I/O connectors under software control. The signals are 5 volt tolerant and can be configured to support differential or single-ended operation. Logic families supported by the I/O interface include LVTTL and LVCMOS. The FPGA device supports up to four phase lock loops for clock synthesis, clock generation and for support of the I/ O interface. An on-board 80 MHz oscillator is available for use with the FGPA device or alternatively, the PXI 10 MHz clock can be used as a clock reference by the FPGA.

The FPGA has access to all of the PXI bus resources including the PXI 10 MHz clock, the local bus, and the PXI triggers, allowing the user to create a custom instrument which incorporates all of the PXIbus' resources. Control and access to the FPGA is provided via the GX3500's driver (GxFPGA) which includes tools for downloading the compiled FPGA code as well as providing register read and write functionality.

The GX3500 employs an Altera Cyclone III, 484 pin device. Key features for the Altera device include:

- 55,856 logic elements (LEs) and 2.34 Mbits of memory
- Supports up to four phase-locked loops (PLLs) for clock synthesis, clock generation and support of I/O interfaces
- Up to five outputs per PLL can be accessed
- Dynamically reconfigurable logic supports programmable phase shift, frequency multiplication/division, and in-system frequency re-programming without reconfiguring the device
- Support for high-speed external memory interfaces including DDR, DDR2, SDR, SDRAM, and QDRII SRAM at up to 400 megabits per second (Mbps)
- 327 I/O pins arranged in eight I/O banks that support a wide range of industry I/O standards
- Supports up to 875 Mbps receive and 840 Mbps transmit LVDS communications data rates
- Support for Bus LVDS (BLVDS), LVDS, RSDS®, mini-LVDS and PPDS® differential I/O standards
- Supported I/O standards include LVTTL, LVCMOS, SSTL, HSTL, PCI, PCI-X, LVPECL, LVDS, mini-LVDS, RSDS, and PPDS; PCI Express Base

Applications

- Automatic Test Equipment (ATE) and Functional Test
- Data Acquisition
- Process Control
- Factory Automation

Board Description

The GX3500 is a 3U PXI instrument card that consists of 160 TTL I/O Channels divided into groups of 40 channels. Each of these groups is tied to a 68 pin SCSI type connector on the front panel of the instrument (J1-J4). The instrument also has several user configurable jumpers (JP3-JP6) that force the I/O to be routed through the front connectors rather than the expansion board. A short on JP7 will force the user FPGA to be configured automatically on boot up with the contents of the EEPROM. For more information about the connectors and jumpers and their location on the board refer to Chapter 3 – Installation and Connections.

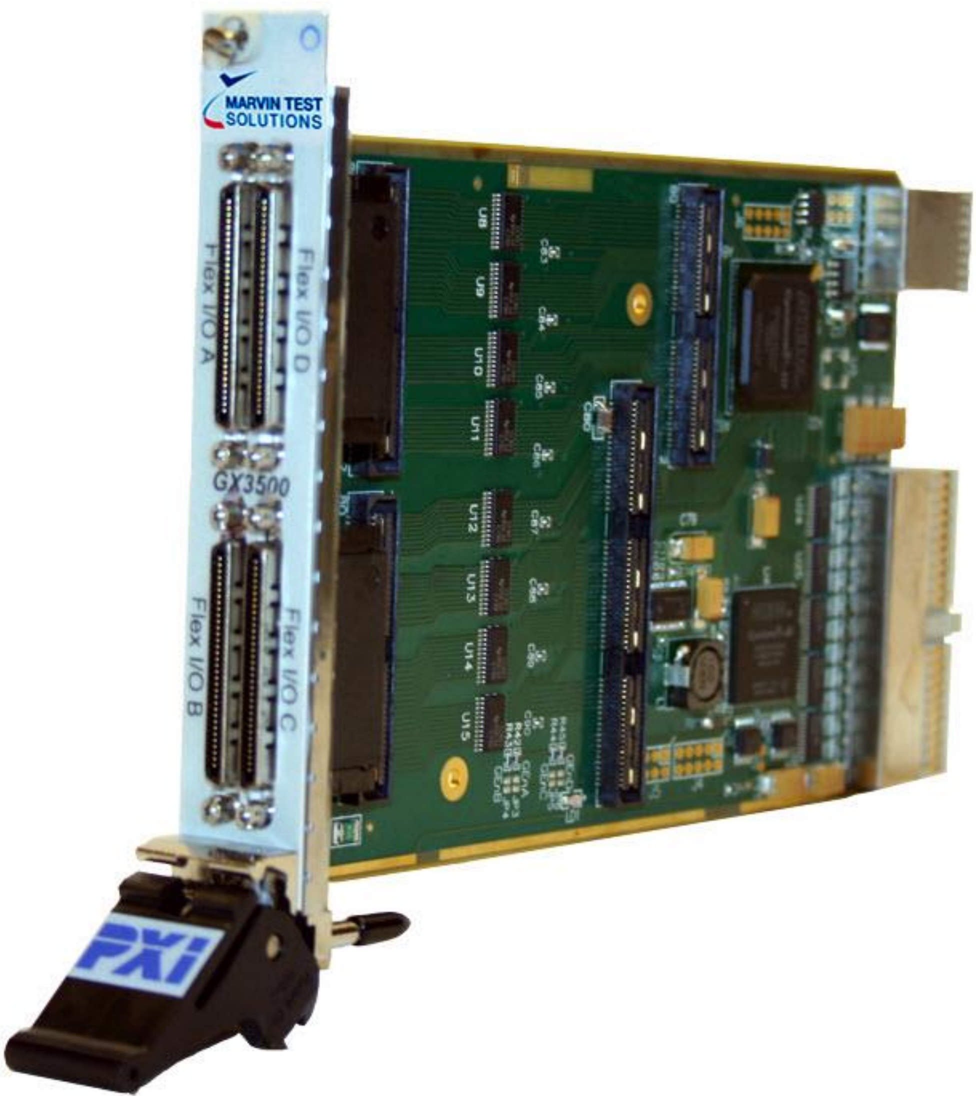
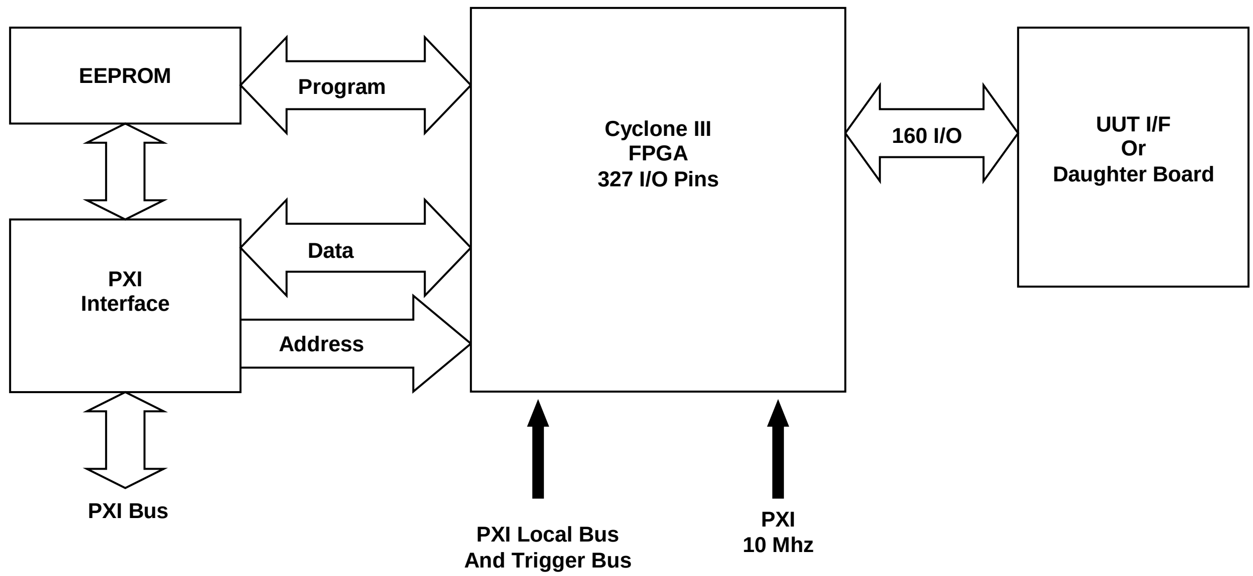


Figure 2-1: GX3500 Board

Architecture

The GX3500 provides 160 I/O Channels that can be connected to the front connectors or the optional expansion daughter board, in groups of 40. To connect the I/O channels to the daughter board, the user disconnects jumpers, JP3 to JP6 for the Bi-Directional Level Shifting Switches, this routs the I/O channels to the daughter board. If the user does not use the optional daughter board and wants to connect the I/O channels directly to the front panel connectors then enabling the Bi-Directional Level Shifting Switches by connecting jumpers JP3, JP4, JP5 and JP6 will route the I/O signals from the user FPGA to the front panel connectors. When connected, each one of these jumpers will permanently select and enable 40 I/O channels to be routed directly to the front panel connectors. For more information see Table 3-1: Connectors and Jumpers and GX3500 Expansion Boards and Chapter 6 - Expansion Board Design Guide.

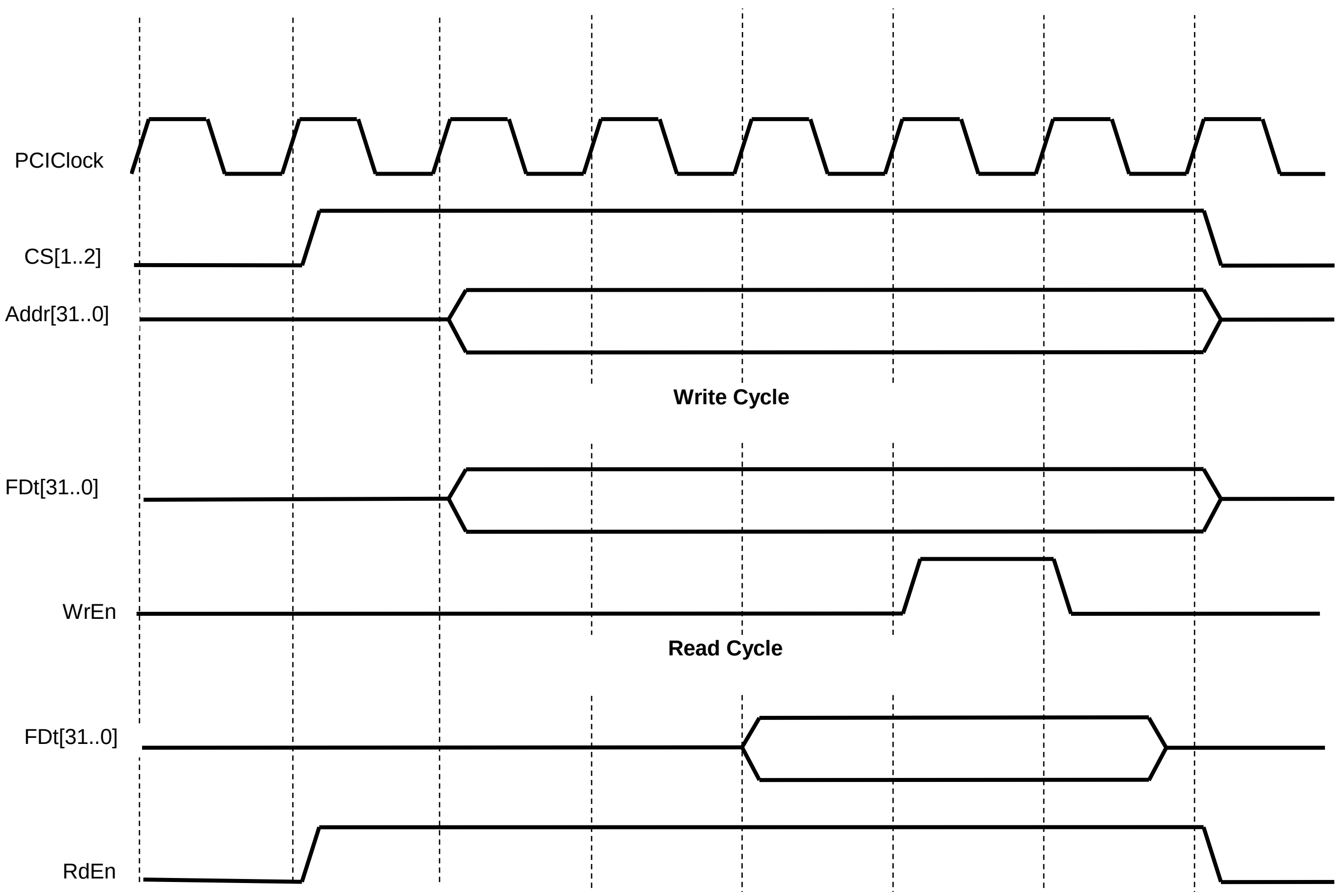


The User FPGA, a Cyclone III, can be configured either through the EEPROM or directly through the PXI Interface. The User FPGA has access to PXI resources such as the local bus, trigger bus, and PXI 10 MHz clock source. The User FPGA is connected to the PXI Interface FPGA to give access to PCI resources and memory. This allows the User FPGA to communicate with the host system's operating system by utilizing the provided GXFPGA software driver.

FPGA PCI Timing

The following graph describes the PCI Signal timing information in relation to the user FPGA (Cyclone III). You can use this information to help in the design of the PCI address decode, read and write circuitry within the user FPGA configuration. Refer to Table 5-1: Pin Assignments Table for information on the pin alias description. The use of these pins is shown in the GXFPGA Tutorial and Example (Phase 1, 2 and 3), where they are used to communicate with the PCI bus and host computer.

Note that the PCIClock signal's period is assumed to be equal to the period of the 33 MHz PCI Clock (30.3 nS).



Legend:

PCIClock – PCI Clock, 33 MHz
CS[1..2] – Chip Select 1 or Chip Select 2
WrEn – Write Enable. When high, rising edge of the clock writes data
RdEn – Read Enable. When high, User FPGA can drive the data bus.
Addr[2..19] – Address bus. Address lines 17..2
FDt[31..0] – Data bus. Data lines 31..0. In a read cycle data must be valid during indicated period.

Specifications

The following table outlines the specifications of the GX3500.

Digital I/O Channel

Logic Families	LVTTL and LVCMOS, 5-volt compatible
Output Current	+/ 4.0 mA
Input Leakage Current	+/- 10 uA
Power On State	Programmable by line, default is disconnected at power on
Number of Channels	4 banks of 40 I/O signals. Direction is configurable on a per pin basis Disconnect on a per bank basis
Protection	Overvoltage: -0.5V to 7.0V (input) Short circuit: up to 8 outputs may be shorted at a time
Connectors	(4) SCSI III, VHDCI type, 68 pin female

Expansion Board Interface

Board ID	4 bits
Digital I/O	160, each bank of 40 can be configured to bypass or access the expansion board
FPGA Flex I/O	4 signals
Master Clear	From PXI interface
Power	+/- 12 volts, +5 volts, +3.3 volts, +2.5 volts, +1.2 volts

Timing Source

PXI 10 MHZ	PXI Bus
Internal	80 MHz oscillator, +/- 20 ppm

User FPGA

FPGA Type	Cyclone III, EP3C55 F484
Number of PLLs	Four
Logic Elements	55,856
Internal Memory	2.34 Mb

Power

3.3 VDC	400 mA (typ.); 1 A (Max.)
5 VDC	300 mA (typ.); 1.2 A (Max.)
12 VDC (For Expansion Board)	

Environmental

Operating Temperature	0 to 50° C
Storage Temperature	-20° C to 70° C
Size	3U PXI
Weight	200 g

Virtual Panel Description

The GX3500 includes a virtual panel program, which enables full utilization of the various configurations and controlling modes. To fully understand the front panel operation, it is best to become familiar with the functionality of the board.

To open the virtual panel application, select **GX3500 Panel** from the **Marvin Test Solutions**, **GXFPGA** menu under the **Start** menu. The GX3500 virtual panel opens as shown here:

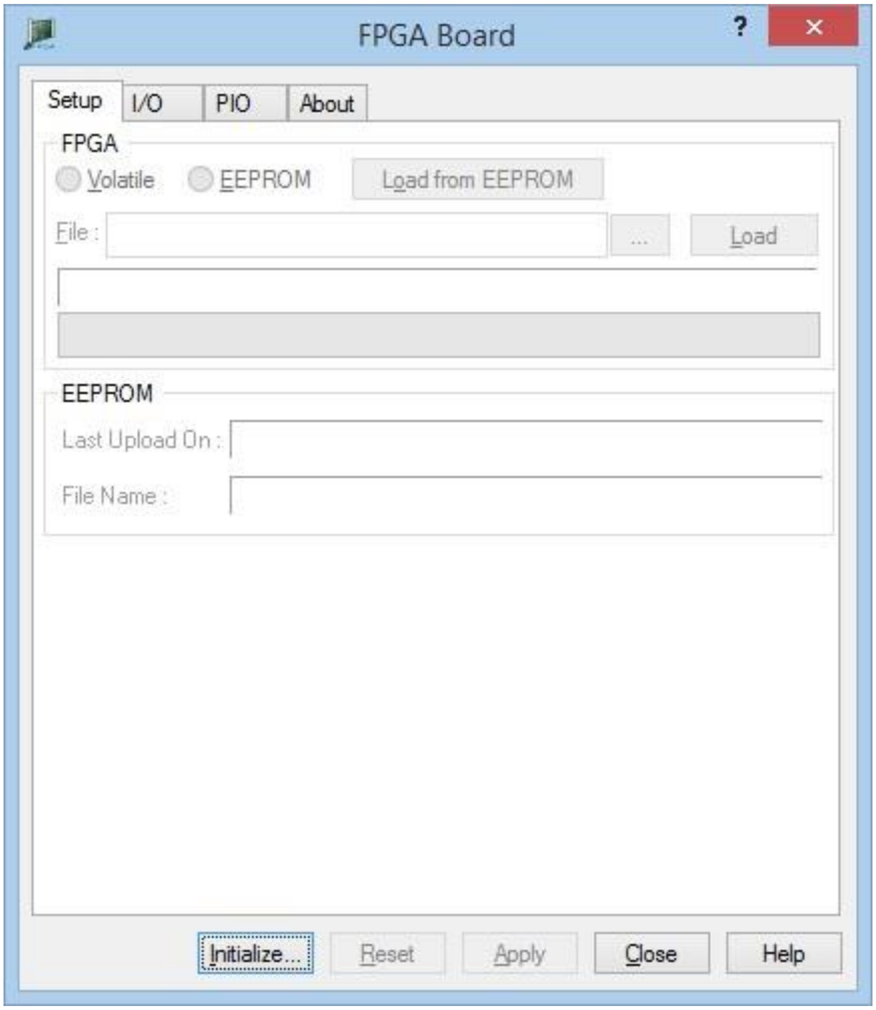


Figure 2-2: GX3500 Virtual Panel

Initialize – Opens the **Initialize Dialog** (see Initialize Dialog paragraph) in order to initialize the board driver. The current settings of the selected board will **not change after calling initialize**. The panel will reflect the current settings of the board after the Initialize dialog closes.

Reset – Resets the PXI board settings to their default state and clears the reading.

Apply – Applies changed settings to the board.

Close – Closes the panel. Closing the panel **does not affect** the board settings.

Help – Opens the on-line help window. In addition to the help menu, the caption shows a **What's This Help** button (?) button. This button can be used to obtain help on any control that is displayed in the panel window. To displays the What's This Help information click on the (?) button and then click on the control – a small window will display the information regarding this control.

Virtual Panel Initialize Dialog

The Initialize dialog initializes the driver for the selected board. The board settings **will not change** after initialize is called. Once initialized, the panel will reflect the current settings of the board.

The Initialize dialog supports two different device drivers that can be used to access and control the board:

Use Marvin Test Solutions' HW – This is the device driver installed by the setup program and is the default driver. When selected, the **Slot Number** list displays the available **GX3500** boards installed in the system and their slots. The chassis, slots, devices and their resources are also displayed by the HW resource manager, **PXI/PCI Explorer** applet that can be opened from the Windows Control Panel. The **PXI/PCI Explorer** can be used to configure the system chassis, controllers, slots and devices. The configuration is saved to PXISYS.INI and PXIeSYS.INI located in the Windows folder. These configuration files are also used by VISA. The following figure shows the slot number 0x109 (chassis 1 Slot 9). This is the slot number argument (*nSlot*) passed by the panel when calling the driver **GxFpgaInitialize** function which is used to initialize the driver for the specified board.

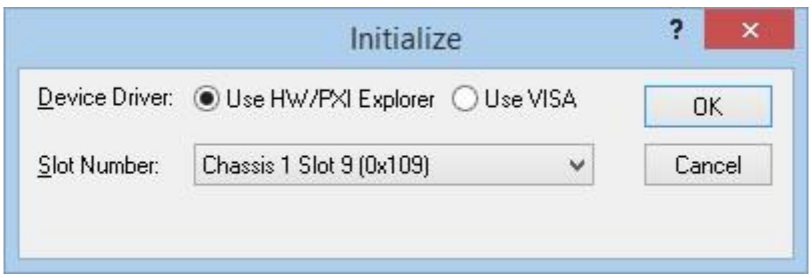


Figure 2-3: Initialize Dialog Box using Marvin Test Solutions' HW driver

Use VISA – This is a third-party device driver usually provided by National Instrument (NI-VISA). When selected, the **Resource** list displays the available boards installed in the system and their VISA resource address. The chassis, slots, devices and their resources are also displayed by the VISA resource manager, **Measurement & Automation** (NI-MAX) and by Marvin Test Solutions **PXI/PCI Explorer**. The following figure shows PXI9::13::INSTR as the VISA resource (PCI bus 9 and Device 13). This is a VISA resource string argument (*szVisaResource*) which is passed by the panel when calling the driver **GxFpgaInitializeVisa** function which initializes the driver for the specified board.

Figure 2-4: Initialize Dialog Box using VISA resources

Virtual Panel Setup Page

After the board is initialized, the panel is enabled and will display the current setting of the board. The following picture shows the **Setup page** settings:

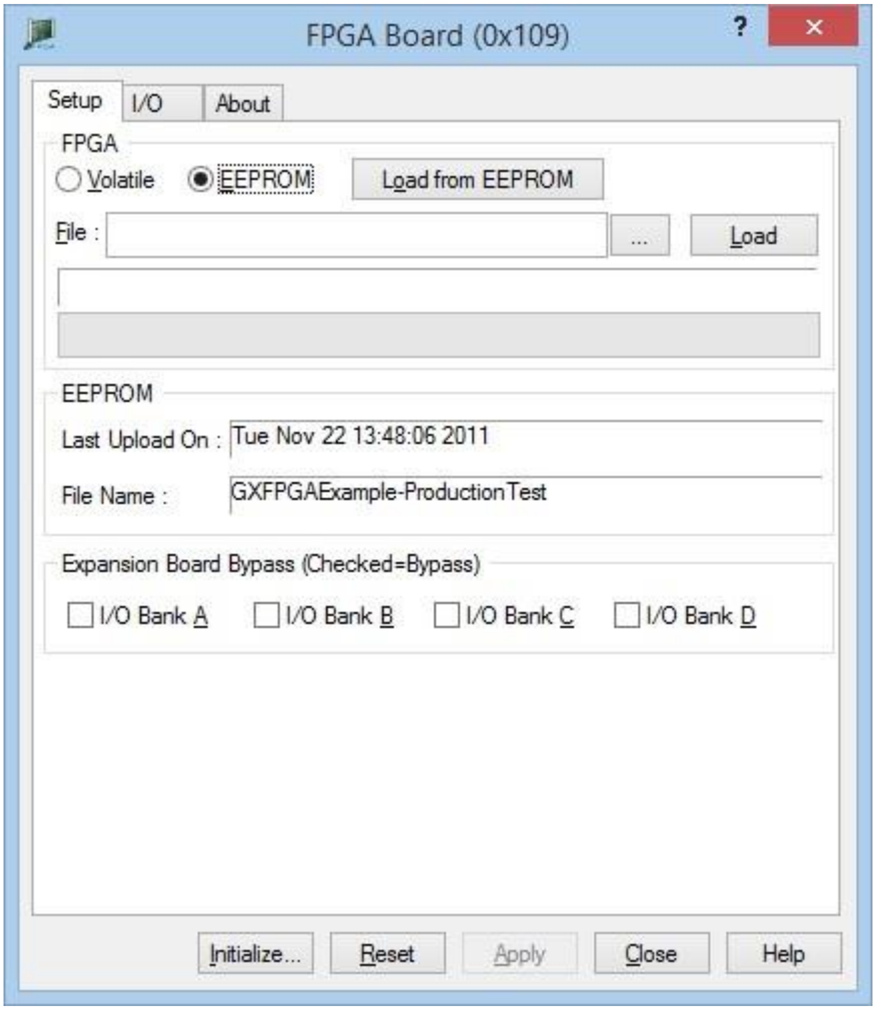


Figure 2-5: GX3500 Virtual Panel – Setup page

The following controls are shown in the Setup page:

Volatile radio button: Select this radio button to load the File to the Volatile (current) FPGA configuration.

EEPROM radio button: Select this radio button to load File to the EEPROM FPGA.

Load From EEPROM button: Loads the volatile (current FPGA) with the FPGA configuration that is stored in the EEPROM

File text box: File path to the programming file intended to load the volatile FPGA or EEPROM. The File type must be Serial Vector File (.svf) for Volatile loading or Raw Programming Data (.RPD) file for EEPROM.

Load Button: Starts the loading process, either to the volatile FPGA or to the EEPROM, depending on which radio button the user selects.

EEPROM Last Updated On Text: Indicates the last time the EEPROM was loaded.

EEPROM File Name Text: Indicates the last file name that was written to the EEPROM.

Expansion Board Bypass Checkboxes: These checkboxes control the routing of each of the FPGA's I/O Banks. When the box is checked, it indicates that the I/O Bank will be connected directly to the I/O front connectors. If the box is unchecked, it indicates that the I/O Bank will be connected to the expansion board.

Virtual Panel I/O Page

Clicking on the **I/O** tab will show the **I/O page** as shown in Figure 2-6: GX3500 Virtual Panel – I/O page

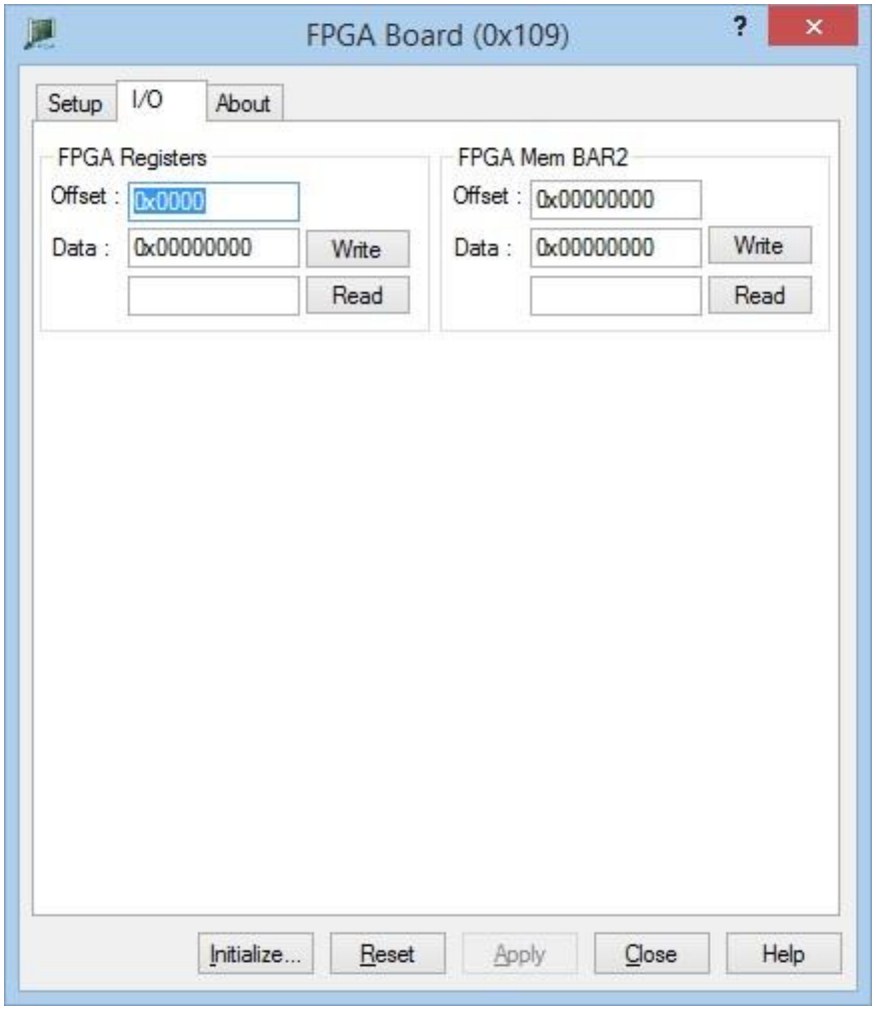


Figure 2-6: GX3500 Virtual Panel – I/O page

The following controls are shown in the I/O page:

Offset Text Field: The offset into the FPGA Register or Memory space in bytes. This field can be used with a decimal or hexadecimal value (prefix the value with 0x). The offset is limited to 0x400 bytes when reading the register space and 0x40000 bytes when reading the memory space. Offset must be specified on a 4-byte alignment.

Write Text Field: The 32-bit data (hexadecimal or decimal) to be written the specified offset in either FPGA Register or Memory space.

Write Button: Write the 32-bit double word to either the FPGA Register or Memory space at the specified offset.

Read Text Field: The 32-bit data that has been read from the specified offset in either FGPA Register or Memory space. Value is specified in hexadecimal.

Read Button: Read the 32-bit double word from either the FPGA Register or Memory space at the specified offset.

Virtual Panel PIO Page

The **PIO** page will only be enabled if the initialized GX3500 has a PIO type expansion board installed (GX3501, GX3509, GX3510, GX3540). Clicking on the **PIO** tab will show the **PIO page** as shown in Figure 2-7: GX3500 Virtual Panel – PIO page

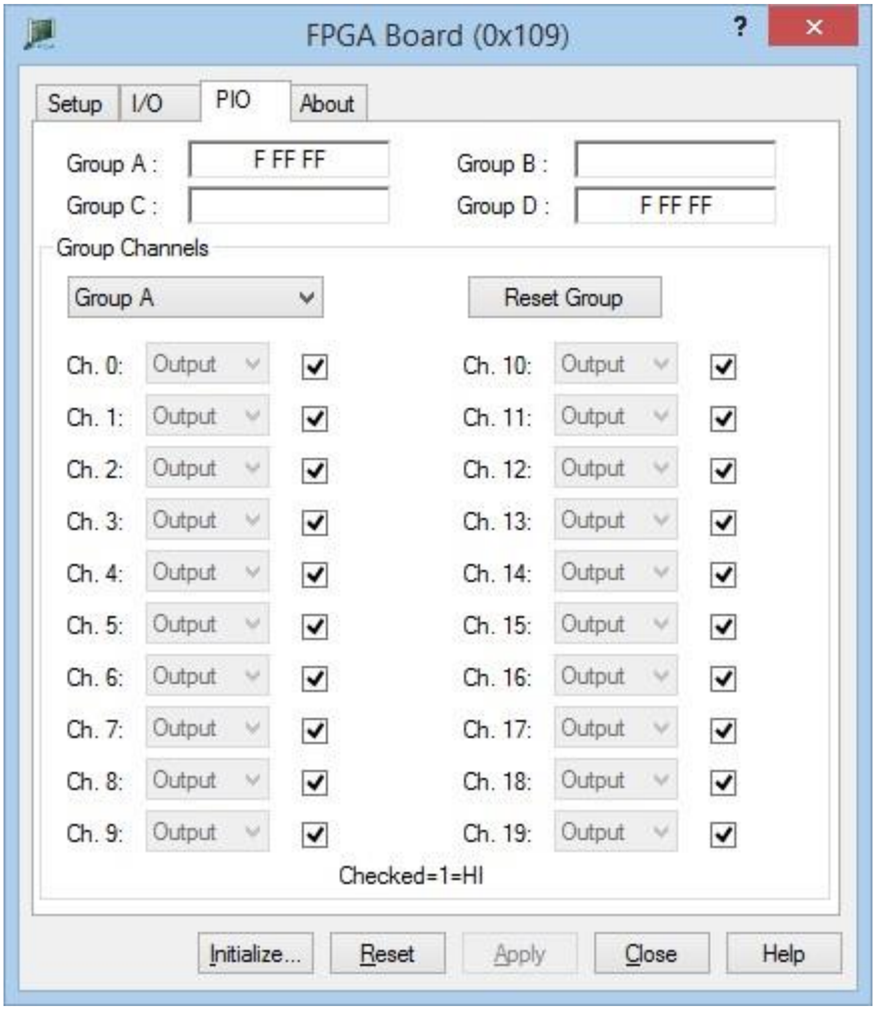


Figure 2-7: GX3500 Virtual Panel – PIO page

The following controls are shown in the I/O page:

Group A/B/C/D Text: Displays the data value of all the channels of the specified group. Each bit represents the logical state of a single channel. The channels are shown with channel 19 as the highest bit and channel 0 as the lowest bit.

Active Group Drop Down List: Identifies the current active group. Upon changing the active group, the channel direction drop down lists and channel data checkboxes will update to reflect the new active group's current state.

Reset Group Button: Clicking this will set the channel direction of all channels in the active group to input. If using the GX3540 expansion board, all output channels will be reset to output a logical 0.

Channel Direction Drop Down Lists: Shows the direction of the specified channel of the active group. If using the GX3540 expansion board, all channel direction drop down lists will be read-only. For all other PIO expansion boards, this control can be used to set the channel direction.

Channel Data Checkboxes: Shows the value of the specified channel of the active group. If checked, the I/O channel is HI (logic 1). If unchecked, the I/O channel is LO (logic 0). If the specified channel direction is set to input, this control will be read-only.

Virtual Panel About Page

Clicking on the **About** tab will show the **About page** as shown in Figure 2-7:

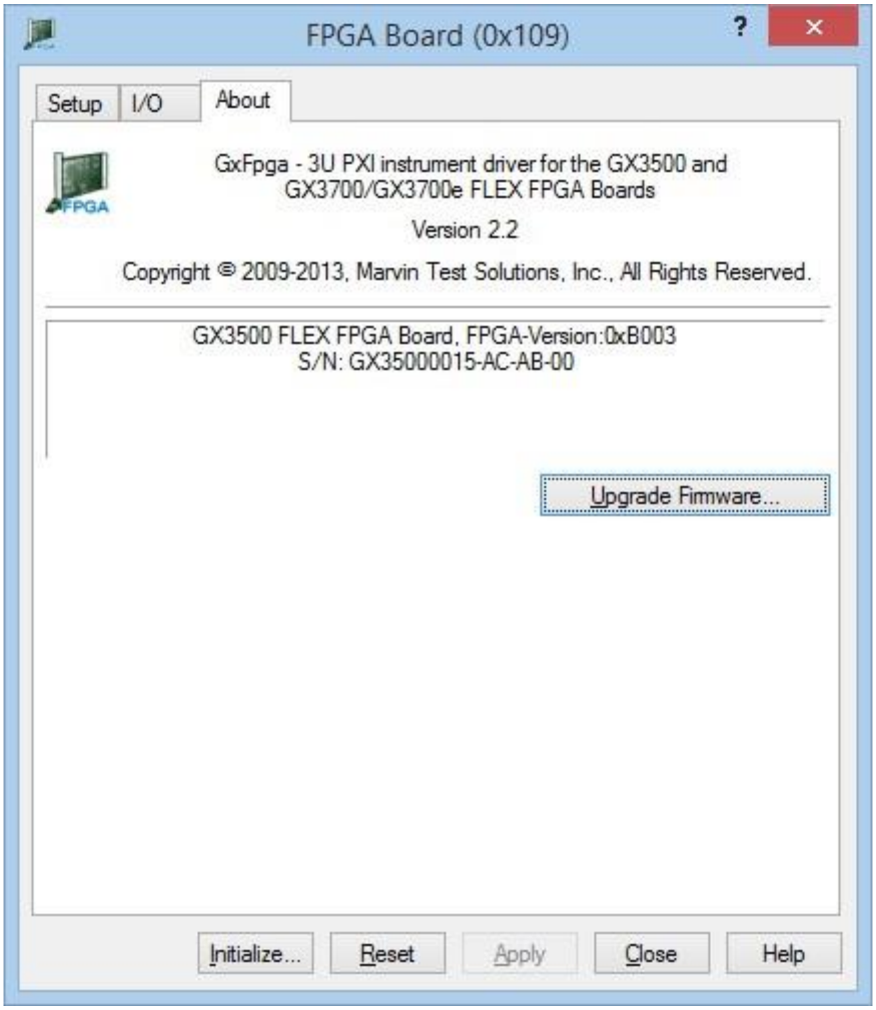


Figure 2-8: GX3500 Virtual Panel – About Page

The top part of the **About** page displays version and copyright of the GX3500 driver. The top part displays the board summary, including the main board FPGA version and each installed I/O Module FPGA version. The **About** page also contains a button **Upgrade Firmware…** used to upgrade the board FPGA. This button may be used only when the board requires upgrade as directed by Marvin Test Solutions support. The upgrade requires a firmware file (.jam) that is written to the board FPGA. After the upgrade is complete you must shut down the computer to recycle power to the board.

Chapter 3 - Installation and Connections

Getting Started

This section includes general hardware installation procedures for the GX3500 board and installation instructions for the GX3500 (GXFPGA) software. Before proceeding, please refer to the appropriate chapter to become familiar with the board being installed.

To Find Information on..	Refer to..
Hardware Installation	This Chapter
GX3500 Driver Installation	This Chapter
Programming	Chapter 4
GXFPGA design tools and tutorial	Chapter 5
Expansion Boards	Chapter 6
GX3500 Function Reference	Chapter 7

Interfaces and Accessories

The following accessories are available from Marvin Test Solutions for GX3500 switching board.

Part / Model Number	Description
GT95015	Connector Interface SCSI to 100 Mil Grid Differential
GT95021	2’ 68-Pin shielded cable
GT95022	3’ 68-Pin shielded cable
GT95028	10’ 68-Pin shielded cable
GT95031	6’ 68-Pin shielded cable

Packing List

All GX3500 boards have the same basic packing list, which includes:

1. GX3500 Board
2. GXFPGA Driver Disk

Unpacking and Inspection

After removing the board from the shipping carton:

Caution - Static sensitive devices are present. Ground yourself to discharge static.

1. Remove the board from the static bag by handling only the metal portions.
2. Be sure to check the contents of the shipping carton to verify that all of the items found in it match the packing list.
3. Inspect the board for possible damage. If there is any sign of damage, return the board immediately. Please refer to the warranty information at the beginning of the manual.

System Requirements

The GX3500 Instrument board is designed to run on PXI compatible computer running Windows 9x, Windows Me, Windows NT, Windows 2000, XP, Vista and above. In addition, Microsoft Windows Explorer version 4.0 or above is required to view the online help.

The board requires one unoccupied 3U PXI bus slot.

Installation of the GXFPGA Software

Before installing the board, it is recommended that you install the GXFPGA software as described in this section. To install the GXFPGA software, follow the instruction described below:

1. Insert the Marvin Test Solutions CD-ROM and locate the **GXFPGA.EXE** setup program. If your computer's Auto Run is configured, when inserting the CD, a browser will show several options. Select the Marvin Test Solutions Files option and then locate the setup file. If Auto Run is not configured, you can open the Windows explorer and locate the setup files (usually located under \Files\Setup folder). You can also download the file from Marvin Test Solutions' web site (www.marvintest.com).
2. Run the GXFPGA setup and follow the instruction on the Setup screen to install the GXFPGA driver.

   Note: When installing under Windows, you may be required to restart the setup logging-in as a user with Administrator privileges. This is required in-order to upgrade your system with newer Windows components and to install kernel-mode device drivers (HW.SYS and HWDEVICE.SYS) which are required by the GXFPGA driver to access resources on your board.

3. The first setup screen to appear is the Welcome screen. Click **Next** to continue.
4. Enter the folder where GXFPGA is to be installed. Either click **Browse** to set up a new folder, or click **Next** to accept the default folder of **C:\Program Files\Marvin Test Solutions\GXFPGA** for 32-bit Windows or **C:\Program Files (x86)\Marvin Test Solutions\GXFPGA** for 64-bit Windows.
5. Select the type of Setup you wish and click **Next.** You can choose between **Typical**, **Run-Time** and **Custom** setups types. The **Typical** setup type installs all files. **Run-Time** setup type will install only the files required for controlling the board either from its driver or from its virtual panel. The **Custom** setup type lets you select from the available components.

The program will now start its installation. During the installation, Setup may upgrade some of the Windows shared components and files. The Setup may ask you to reboot after completion if some of the components it replaced were used by another application during the installation – do so before attempting to use the software.

You can now continue with the installation to install the board. After the board installation is complete you can test your installation by starting a panel program that lets you control the board interactively. The panel program can be started by selecting it from the Start, Programs, GXFPGA menu located in the Windows Taskbar.

Setup Maintenance Program

You can run the Setup again after GXFPGA has been installed from the original disk or from the Windows Control Panel – Add Remove Programs applet. Setup will be in the Maintenance mode when running for the second time. The Maintenance window show below allows you to modify the current GXFPGA installation. The following options are available in Maintenance mode:

Modify. When you want to add, or remove GXFPGA components.

Repair. When you have corrupted files and need to reinstall.

Remove. When you want to completely remove GXFPGA.

Select one of the options and click **Next** and follow the instruction on the screen until Setup is complete.

Overview of the GXFPGA Software

Once the software is installed, the following tools and software components are available:

- **GXFPGA Panel** – Configures and controls the GXFPGA board various features via an interactive user interface.
- **GXFPGA driver** - A DLL based function library (GXFPGA.DLL, located in the Windows System folder) used to program and control the board. The driver uses Marvin Test Solutions' HW driver or VISA supplied by third party vendor to access and control the GXFPGA boards.
- **Programming files and examples** – Interface files and libraries for support of various programming tools. A complete list of files and development tools supported by the driver is included in subsequent sections of this manual.
- **Documentation** – On-Line help and User's Guide for the board, GXFPGA driver and panel.
- **HW driver and PXI/PCI Explorer applet** – HW driver allows the GXFPGA driver to access and program the supported boards. The explorer applet configures the PXI chassis, controllers and devices. This is required for accurate identification of your PXI instruments later on when installed in your system. The applet configuration is saved to PXISYS.ini and PXIeSYS.ini and is used by Marvin Test Solutions instruments HW driver and VISA. The applet can be used to assign chassis numbers, Legacy Slot numbers and instrument alias names. The HW driver is installed and shared with all Marvin Test Solutions products to support accessing the PC resources. Similar to HW driver, VISA provides a standard way for instrument manufacturers and users to write and use instruments drivers. VISA is a standard maintained by the VXI Plug & Play System Alliance and the PXI Systems Alliance organizations (http://www.pxisa.org/). The VISA resource manager such as National Instruments **Measurement & Automation** (NI-MAX) displays and configures instruments and their address (similar to Marvin Test Solutions' PXI/PCI Explorer). The GXFPGA driver can work with either HW or VISA to control an access the supported boards.

Installation Folders

The GX3500 driver files are installed in the default folder **C:\Program Files [(x86)]\Marvin Test Solutions\GXFPGA**. You can change the default GXFPGA folder to one of your choosing at the time of installation.

During the installation, GXFPGA Setup creates and copies files to the following folders:

Name	Purpose / Contents
…\Marvin Test Solutions\GXFPGA	The GXFPGA folder. Contains panel programs, programming libraries, interface files and examples, on-line help files and other documentation.
…\Marvin Test Solutions\HW	HW device driver. Provide access to your board hardware resources such as memory, IO ports and PCI board configuration. See the README.TXT located in this directory for more information.
…\ATEasy\Drivers	ATEasy drivers folder. GXFPGA Driver and example are copied to this directory only if ATEasy is installed to your machine.
…\Windows\System or System32	Windows System directory. Contains the GXFPGA.DLL or GXFPGA64.DLL driver, HW driver shared files and some upgraded system components, such as the HTML help viewer, etc.

Configuring Your PXI System using the PXI/PCI Explorer

To configure your PXI/PCI system using the **PXI/PCI Explorer** applet follow these steps:

1. **Start the PXI/PCI Explorer applet**. The applet can be start from the Windows Control Panel or from the Windows Start Menu, **Marvin Test Solutions**, **HW**, **PXI/PCI Explorer**.
2. **Identify Chassis and Controllers**. After the PXI/PCI Explorer is started, it will scan your system for changes and will display the current configuration. The PXI/PCI Explorer automatically detects systems that have Marvin Test Solutions controllers and chassis. In addition, the applet detects PXI-MXI-3/4 extenders in your system (manufactured by National Instruments). If your chassis is not shown in the explorer main window, use the Identify Chassis/Controller commands to identify your system. Chassis and Controller manufacturers should provide INI and driver files for their chassis and controllers which are used by these commands.
3. **Change chassis numbers, PXI devices Legacy Slot numbering and PXI devices Alias names.** These are optional steps and can be performed if you would like your chassis to have different numbers. Legacy slots numbers are used by older Marvin Test Solutions or VISA drivers. Alias names can provide a way to address a PXI device using a logical name (e.g. "FPGA1"). For more information regarding slot numbers and alias names, see the **GX3500Initialize** and **GxFpgaInitializeVisa** functions.
4. **Save your work**. PXI Explorer saves the configuration to the following files located in the Windows folder: PXISYS.ini, PXIeSYS.ini and GxPxiSys.ini. Click on the **Save** button to save your changes. The PXI/Explorer will prompt you to save the changes if changes were made or detected (an asterisk sign ' *' in the caption indicated changes).

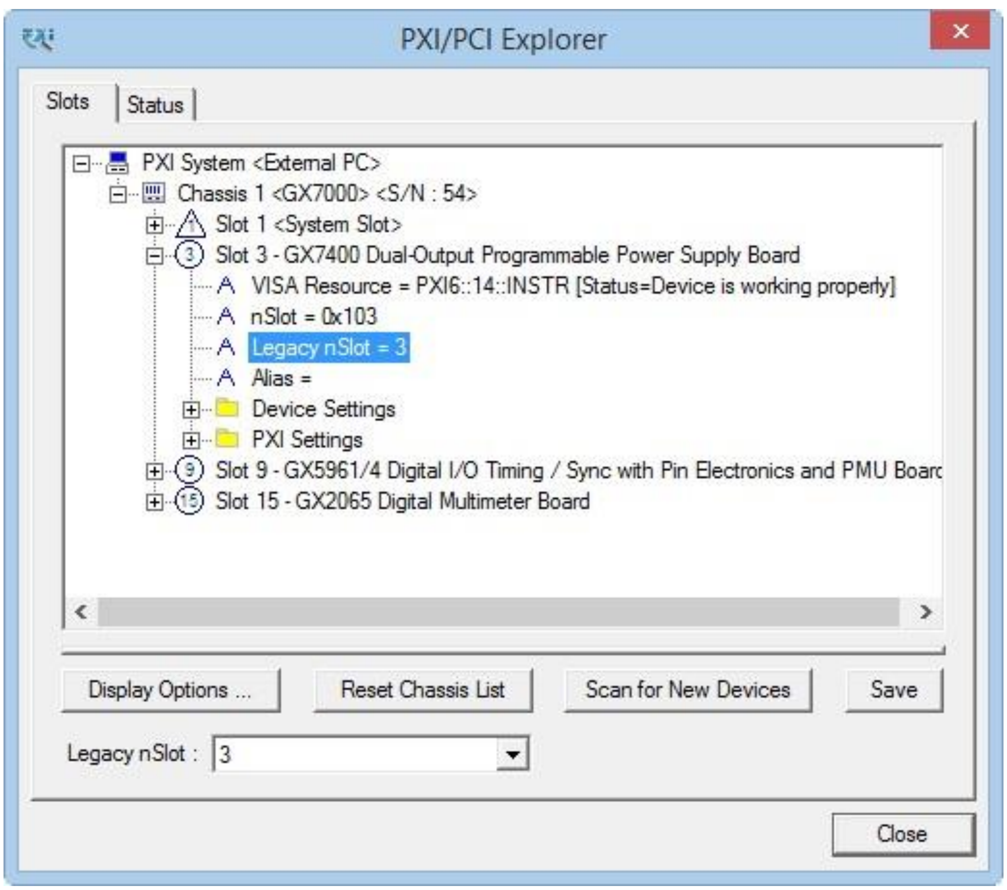


Figure 3-1: PXI/PCI Explorer

Board Installation

Before you Begin

- Install the GXFPGA driver as described in the prior section.
- Configure your PXI/PC system using **PXI/PCI Explorer** as described in the prior section.
- Verify that all the components listed in the packing list (see previous section in this chapter) are present.

Electric Static Discharge (ESD) Precautions

To reduce the risk of damage to the GX3500 board, the following precautions should be observed:

Leave the board in the anti-static bags until installation requires removal. The anti-static bag protects the board from harmful static electricity.

Save the anti-static bag in case the board is removed from the computer in the future.

Carefully unpack and install the board. Do not drop or handle the board roughly.

Handle the board by the edges. Avoid contact with any components on the circuit board.

Caution – Do not insert or remove any board while the computer is on. Turn off the power from the PXI chassis before installation.

Installing a Board

Install the board as follows:

1. Install first the GXFPGA Driver as described in the next section.
2. Turn off the PXI chassis and unplug the power cord.
3. Locate a PXI empty slot on the PXI chassis.
4. Place the module edges into the PXI chassis rails (top and bottom).
5. Carefully slide the PXI board to the rear of the chassis, make sure that the ejector handles are pushed **out** (as shown in Figure 3-2).

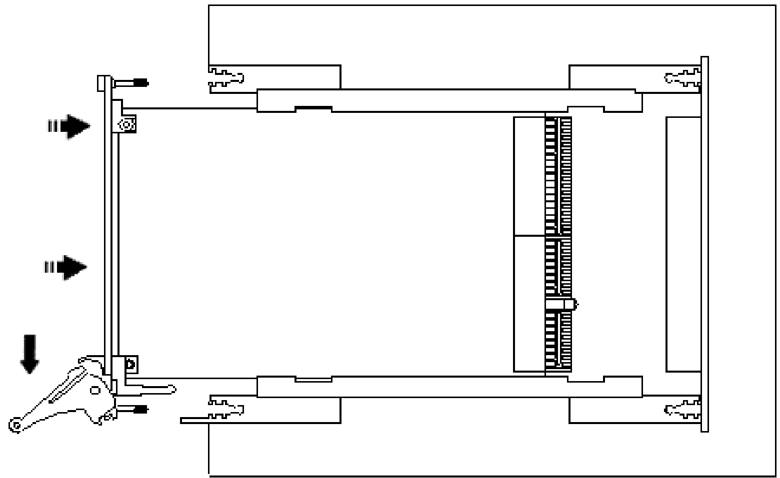


Figure 3-2: Ejector handles position during module insertion

6. After you feel resistance, push in the ejector handles as shown in Figure 3-3 to secure the module into the frame.

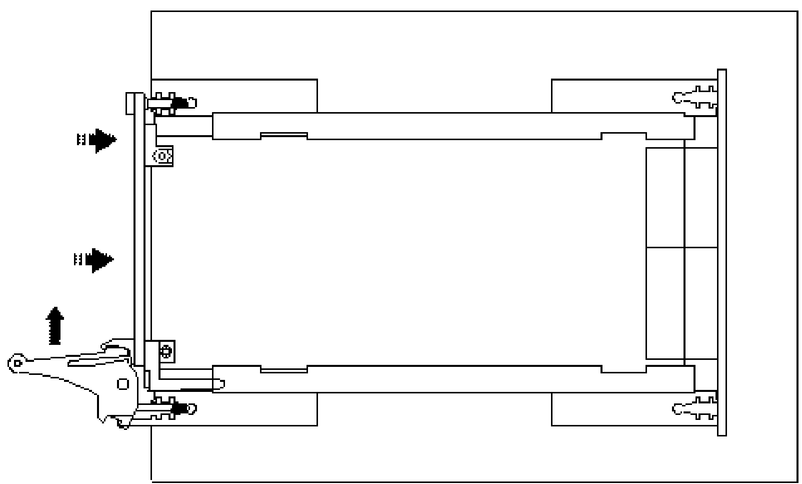


Figure 3-3: Ejector handles position after module insertion

7. Tighten the module's front panel to the chassis to secure the module in.
8. Connect any necessary cables to the board.
9. Plug the power cord in and turn on the PXI chassis.

Plug & Play Driver Installation

Plug & Play operating systems such as Windows notifies the user that a new board was found using the **New Hardware Found** wizard after restarting the system with the new board.

If another Marvin Test Solutions board software package was already installed, Windows will suggest using the driver information file: HW.INF. The file is located in your Program Files\Marvin Test Solutions\HW folder. Click **Next** to confirm and follow the instructions on the screen to complete the driver installation.

If the operating system was unable to find the driver (since the GXFPGA driver was not installed prior to the board installation), you may install the GXFPGA driver as described in the prior section, then click on the **Have Disk** button and browse to select the HW.INF file located in C:\Program File\Marvin Test Solutions\HW.

If you are unable to locate the driver click **Cancel** to the found New Hardware wizard and exit the New Hardware Found Wizard, install the GXFPGA driver, reboot your computer and repeat this procedure.

The Windows Device Manager (open from the System applet from the Windows Control Panel) must display the proper board name before continuing to use the board software (no Yellow warning icon shown next to device). If the device is displayed with an error, you can select it and press delete and then press F5 to rescan the system again and to start the New Hardware Found wizard.

Removing a Board

Remove the board as follows:

1. Turn off the PXI chassis and unplug the power cord.
2. Locate a PXI slot on the PXI chassis.
3. Disconnect and remove any cables/connectors connected to the board.
4. Un-tighten the module’s front panel screws to the chassis.
5. Push out the ejector handles and slide the PXI board away from the chassis.
6. Optionally – uninstall the GXFPGA driver.

Connectors and Jumpers

The following table and figures describes the GX3500 connectors and jumpers.

Connector/Jumpers	Description
J1	Flex I/O Bank A (I/O channels1-40)
J2	Flex I/O Bank D (I/O channels 121-160)
J3	Flex I/O Bank B (I/O channels 41- 80)
J4	Flex I/O Bank C (I/O channels 81-120)
JP3	Force I/O channels 1-40 to be routed directly to the J1 connector. See Figure 3-5.
JP4	Force I/O channels 41-80 to be routed directly to the J3 connector. See Figure 3-5.
JP5	Force I/O channels 81-120 to be routed directly to the J4 connector. See Figure 3-5.
JP6	Force I/O channels 121-160 to be routed directly to the J2 connector. See Figure 3-5.
JP7	Force user FPGA to be configured automatically by the EEPROM contents when power is first applied to the instrument (upon power up of the host system). See Figure 3-6.

Table 3-1: GX3500 Connectors and Jumpers

Figure 3-4 shows the available GX3500 board connectors and jumpers followed by their description:

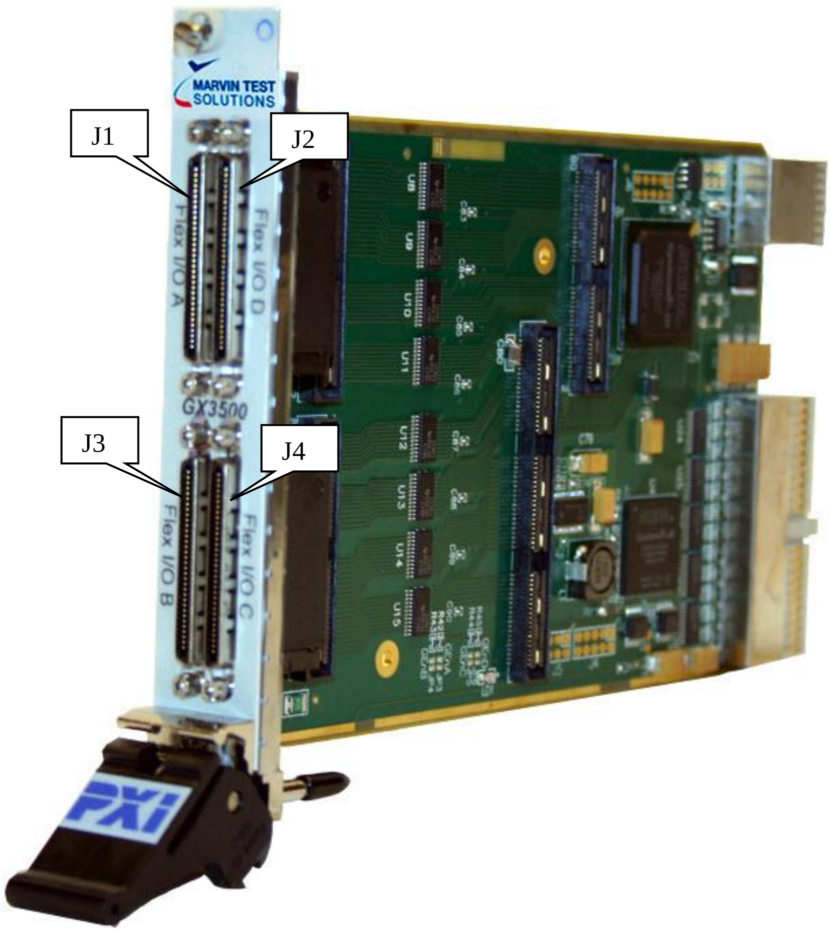


Figure 3-4: GX3500 Connectors (J1-J4)

Figure 3-5 shows GX3500 board JP3, JP4, JP5 and JP6 jumpers:

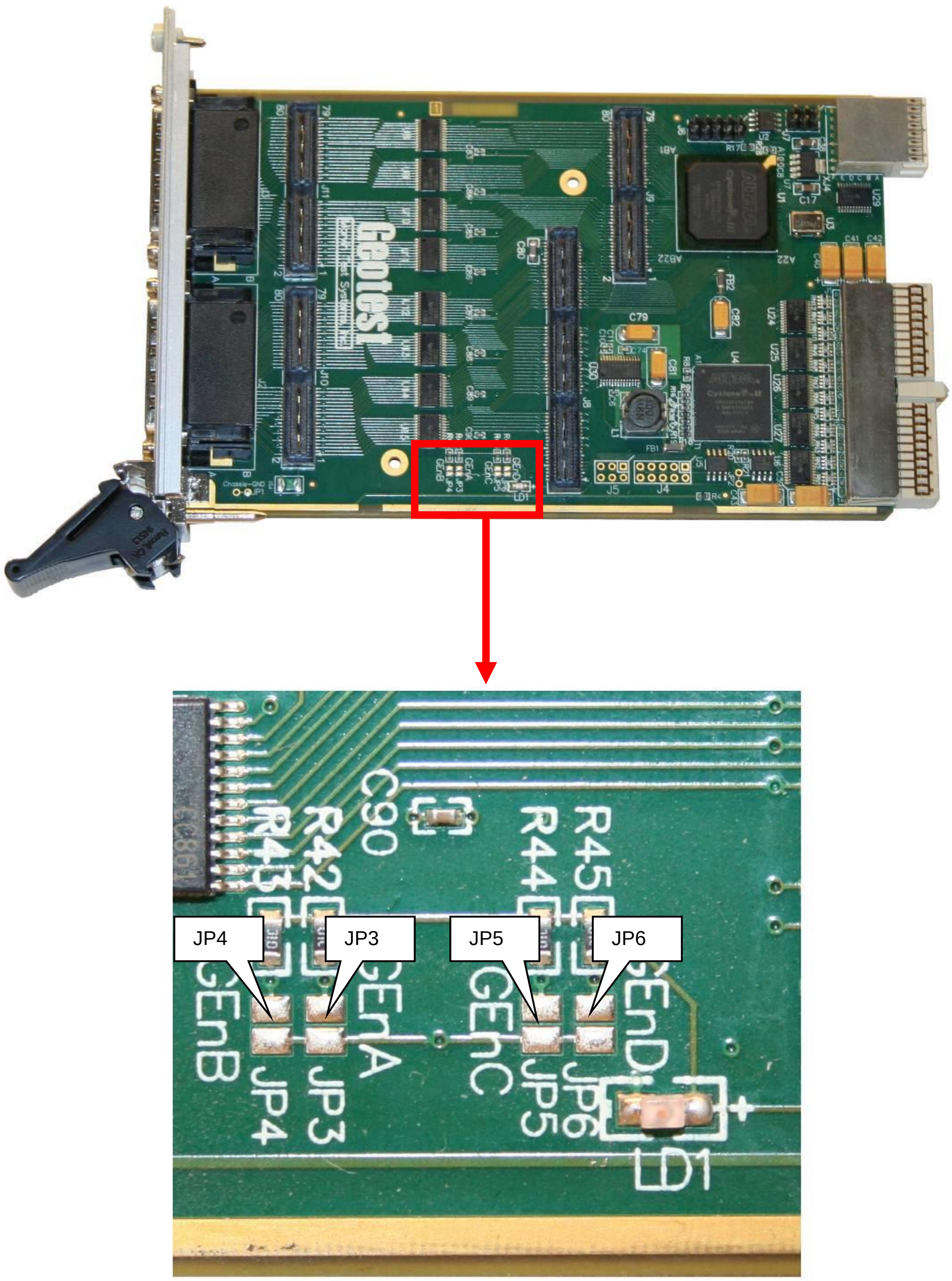


Figure 3-5: GX3500 – Front View Jumpers (JP3-JP6)

Figure 3-6 shows GX3500 board JP7 Jumper:

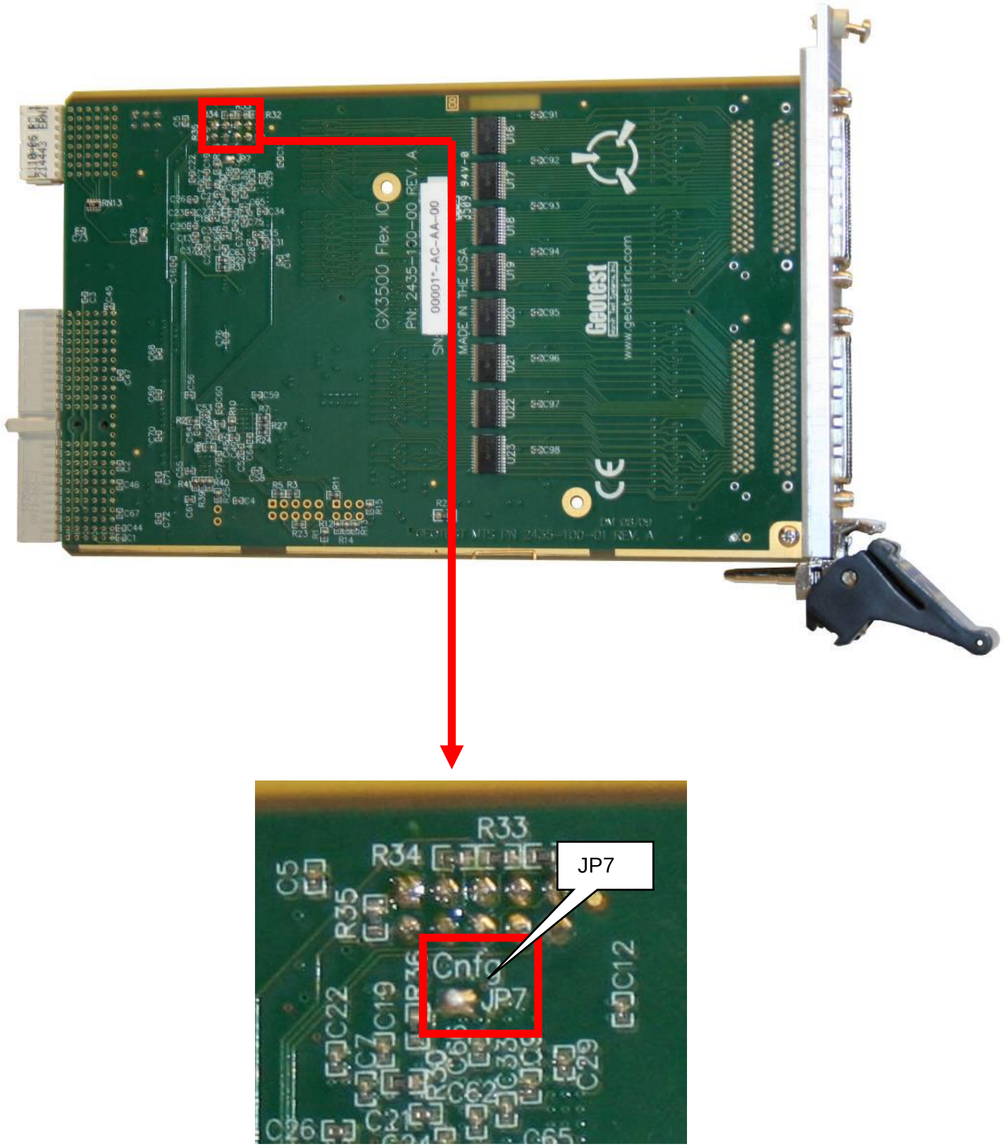


Figure 3-6: GX3500 – Back View Jumper (JP7)

GX3500 Connectors – J1-J4 Flex I/O Banks (A-D)

Connections to the GX3500 may be made with 68-pin VHDCI male plug connector. Shielded cables with matching connectors are available from Marvin Test Solutions.

GX3500 J1 – Flex I/O Bank A Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	Flex I/O 1	52	Flex I/O 18
2	GND	19	GND	36	Flex I/O 2	53	Flex I/O 19
3	GND	20	GND	37	Flex I/O 3	54	Flex I/O 20
4	GND	21	GND	38	Flex I/O 4	55	Flex I/O 21
5	GND	22	GND	39	Flex I/O 5	56	Flex I/O 22
6	GND	23	GND	40	Flex I/O 6	57	Flex I/O 23
7	GND	24	GND	41	Flex I/O 7	58	Flex I/O 24
8	GND	25	GND	42	Flex I/O 8	59	Flex I/O 25
9	GND	26	GND	43	Flex I/O 9	60	Flex I/O 26
10	GND	27		44	Flex I/O 10	61	Flex I/O 27
11	GND	28		45	Flex I/O 11	62	Flex I/O 28
12	GND	29	Flex I/O 29	46	Flex I/O 12	63	Flex I/O 30
13	GND	30	Flex I/O 31	47	Flex I/O 13	64	Flex I/O 32
14	GND	31	Flex I/O 33	48	Flex I/O 14	65	Flex I/O 34
15	GND	32	Flex I/O 35	49	Flex I/O 15	66	Flex I/O 36
16	GND	33	Flex I/O 37	50	Flex I/O 16	67	Flex I/O 38
17	GND	34	Flex I/O 39	51	Flex I/O 17	68	Flex I/O 40

Table 3-2: J1 Flex IO Bank A Pin Out

GX3500 J3 – Flex I/O Bank B Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	Flex I/O 41	52	Flex I/O 58
2	GND	19	GND	36	Flex I/O 42	53	Flex I/O 59
3	GND	20	GND	37	Flex I/O 43	54	Flex I/O 60
4	GND	21	GND	38	Flex I/O 44	55	Flex I/O 61
5	GND	22	GND	39	Flex I/O 45	56	Flex I/O 62
6	GND	23	GND	40	Flex I/O 46	57	Flex I/O 63
7	GND	24	GND	41	Flex I/O 47	58	Flex I/O 64
8	GND	25	GND	42	Flex I/O 48	59	Flex I/O 65
9	GND	26	GND	43	Flex I/O 49	60	Flex I/O 66
10	GND	27		44	Flex I/O 50	61	Flex I/O 67
11	GND	28		45	Flex I/O 51	62	Flex I/O 68
12	GND	29	Flex I/O 69	46	Flex I/O 52	63	Flex I/O 70
13	GND	30	Flex I/O 71	47	Flex I/O 53	64	Flex I/O 72
14	GND	31	Flex I/O 73	48	Flex I/O 54	65	Flex I/O 74
15	GND	32	Flex I/O 75	49	Flex I/O 55	66	Flex I/O 76
16	GND	33	Flex I/O 77	50	Flex I/O 56	67	Flex I/O 78
17	GND	34	Flex I/O 79	51	Flex I/O 57	68	Flex I/O 80

Table 3-3: J3 Flex IO Bank B Pin Out

GX3500 J4 – Flex I/O Bank C Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	Flex I/O 81	52	Flex I/O 98
2	GND	19	GND	36	Flex I/O 82	53	Flex I/O 99
3	GND	20	GND	37	Flex I/O 83	54	Flex I/O 100
4	GND	21	GND	38	Flex I/O 84	55	Flex I/O 101
5	GND	22	GND	39	Flex I/O 85	56	Flex I/O 102
6	GND	23	GND	40	Flex I/O 86	57	Flex I/O 103
7	GND	24	GND	41	Flex I/O 87	58	Flex I/O 104
8	GND	25	GND	42	Flex I/O 88	59	Flex I/O 105
9	GND	26	GND	43	Flex I/O 89	60	Flex I/O 106
10	GND	27		44	Flex I/O 90	61	Flex I/O 107
11	GND	28		45	Flex I/O 91	62	Flex I/O 108
12	GND	29	Flex I/O 109	46	Flex I/O 92	63	Flex I/O 110
13	GND	30	Flex I/O 111	47	Flex I/O 93	64	Flex I/O 112
14	GND	31	Flex I/O 113	48	Flex I/O 94	65	Flex I/O 114
15	GND	32	Flex I/O 115	49	Flex I/O 95	66	Flex I/O 116
16	GND	33	Flex I/O 117	50	Flex I/O 96	67	Flex I/O 118
17	GND	34	Flex I/O 119	51	Flex I/O 97	68	Flex I/O 120

Table 3-4: J4 Flex IO Bank C Pin Out

GX3500 J2 – Flex I/O Bank D Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	Flex I/O 121	52	Flex I/O 138
2	GND	19	GND	36	Flex I/O 122	53	Flex I/O 139
3	GND	20	GND	37	Flex I/O 123	54	Flex I/O 140
4	GND	21	GND	38	Flex I/O 124	55	Flex I/O 141
5	GND	22	GND	39	Flex I/O 125	56	Flex I/O 142
6	GND	23	GND	40	Flex I/O 126	57	Flex I/O 143
7	GND	24	GND	41	Flex I/O 127	58	Flex I/O 144
8	GND	25	GND	42	Flex I/O 128	59	Flex I/O 145
9	GND	26	GND	43	Flex I/O 129	60	Flex I/O 146
10	GND	27		44	Flex I/O 130	61	Flex I/O 147
11	GND	28		45	Flex I/O 131	62	Flex I/O 148
12	GND	29	Flex I/O 149	46	Flex I/O 132	63	Flex I/O 150
13	GND	30	Flex I/O 151	47	Flex I/O 133	64	Flex I/O 152
14	GND	31	Flex I/O 153	48	Flex I/O 134	65	Flex I/O 154
15	GND	32	Flex I/O 155	49	Flex I/O 135	66	Flex I/O 156
16	GND	33	Flex I/O 157	50	Flex I/O 136	67	Flex I/O 158
17	GND	34	Flex I/O 159	51	Flex I/O 137	68	Flex I/O 160

Table 3-5: J2 Flex IO Bank D Pin Out

I/O: Input/ Output, **R**: Reserved, **DNU**: Do Not Use, **P**: Power/GND

Chapter 4 - Programming the Board

This chapter contains information about how to program the GX3500 board using the GXFPGA driver.

The GXFPGA driver contains functions to initialize, reset, and control the board. A brief description of the functions, as well as how and when to use them, is included in this chapter.

The GXFPGA driver supports many development tools. Using these tools with the driver is described in this chapter. In addition, the GXFPGA directory contains examples written for these development tools.

The GXFPGA Driver

The GXFPGA DLL driver is provided with support for 32 bit Windows (GXFPGA.DLL) and 64 bit Windows (GXFPGA64.DLL). Additional drivers are provided for other operating systems such as Linux and LabView Real-Time, see the readme file for more information regarding these drivers. The 32-bit DLL is used with 32 bit applications running under Windows 32 or 64 bit editions and the 64-bit DLL on Windows 64-bit editions. The DLLs uses device driver (HW provided by Marvin Test Solutions or VISA provided by a third-party vendor) to access the board resources. The device driver HW includes HW.SYS and HW64.SYS is installed by the GXFPGA setup program and is shared by other Marvin Test Solutions products (ATEasy, GTDIO, etc.).

The DLLs can be used with various development tools such as Microsoft Visual C++, Borland C++ Builder, Microsoft Visual Basic, Borland Pascal or Delphi, ATEasy and more. The following paragraphs describe how to create an application that uses the driver with various development tools. Refer to the paragraph describing the specific development tool for more information.

Programming Using C/C++ Tools

The following steps are required to use the GXFPGA driver with C/C++ development tools:

- Include the GXFPGA.h header file in the C/C++ source file that uses the GXFPGA function. This header file is used for all driver types. The file contains function prototypes and constant declarations to be used by the compiler for the application.
- Add the required .LIB file to the projects. This can be import library GXFPGA.lib and GXFPGA64.lib (for 64 bit applications) for Microsoft Visual C++ and GXFPGABC.lib for Borland C++. Windows based applications that explicitly load the DLL by calling the Windows LoadLibrary() API should not include the .LIB file in the project.
- Add code to call the GXFPGA as required by the application.
- Build the project.
- Run, test, and debug the application.

Programming Using Visual Basic and Visual Basic .NET

To use the driver with Visual Basic 4.0 or above (for 32-bit applications), the user must include the GXFPGA.bas to the project. The file can be loaded using *Add File* from the Visual Basic *File menu*. The GXFPGA.bas contains function declarations for the DLL driver. If you are using Visual Basic .NET – use the GXFPGA.vb.

Programming Using Pascal/Delphi

To use the driver with Borland Pascal or Delphi, the user must include the GXFPGA.pas to the project. The GXFPGA.pas file contains a **unit** with function prototypes for the DLL functions. Include the GXFPGA unit in the **uses** statement before making calls to the GXFPGA functions.

Programming GXFPGA Boards Using ATEasy®

The GXFPGA package is supplied with a separate ATEasy driver for each board types. For example, the GX3500 is supplied with GXFPGA.drv ATEasy driver. The ATEasy driver uses the GXFPGA.dll to program the board. In addition, each driver is supplied with an example that contains a program and a system file pre-configured with the ATEasy driver. Use the driver shortcut property page from the System Drivers sub-module to change the PXI HW slot number or VISA resource string before attempting to run the example.

Using commands declared in the ATEasy driver are easier to use than using the DLL functions directly. The driver commands will also generate exceptions that allow the ATEasy application to trap errors without checking the status code returned by the DLL function after each function call.

The ATEasy driver contains commands that are similar to the DLL functions in name and parameters, with the following exceptions:

The *nHandle* parameter is omitted. The driver handles this parameter automatically. ATEasy uses driver logical names instead i.e. FPGA1 for GX3500.

The *nStatus* parameter was omitted. Use the Get Status commands instead of checking the status. After calling a DLL function the ATEasy driver will check the returned status and will call the error statement (in case of an error status) to generate exception that can be easily trapped by the application using the **OnError** module event or using the **try-catch** statement.

Some ATEasy drivers contain additional commands to permit easier access to the board features. For example, parameters for a function may be omitted by using a command item instead of typing the parameter value. The commands are self-documented. Their syntax is similar to English. In addition, you may generate the commands from the code editor context menu or by using the ATEasy's code completion feature instead of typing them directly.

Programming Using LabVIEW and LabVIEW/Real Time

To use the driver with LabVIEW use the provided lab view library GXFPGA.llb. The library is located in the GXFPGA folder. An example for LabVIEW is also provided in the Examples folder. A DLL located in the LabViewRT folder can be used for deployment with LabVIEW/Real-Time.

Using and Programming under Linux

Marvin Test Solutions provides a separate software package **GtLinux** with Linux driver (Marvin Test Solutions Drivers Pack for Linux). The software package can be downloaded from the Marvin Test Solutions website. See the ReadMe.txt in that package for more information regarding using and programming the driver under Linux.

Using the GXFPGA driver functions

The following paragraphs describe the steps required to program the boards.

Initialization, HW Slot Numbers and VISA Resource

The GXFPGA driver supports two device drivers HW and VISA which are used to initialize, identify and control the board. The user can use the **GxFpgaInitialize** to initialize the board 's driver using HW and **GxFpgaInitializeVisa** to initialize using VISA. The following describes the two different methods used to initialize:

1. **Marvin Test Solutions' HW** – This is the default device driver that is installed by the GXFPGA driver. To initialize and control the board using the HW use the **GxFpgaInitialize**(*nSlot*, *pnHandle*, *pnStatus*) function. The function initializes the driver for the board at the specified PXI slot number (*nSlot*) and returns boards handle. The **PXI/PCI Explorer** applet in the Windows Control Panel displays the PXI slot assignments. You can specify the *nSlot* parameter in the following way:

   - A combination of chassis number (chassis # x 256) with the chassis slot number, e.g. 0x105 for chassis 1 and slot 5. The chassis number can be set by the **PXI/PCI Explorer** applet.

   - Legacy nSlot is used by earlier versions of HW/VISA. The slot number contains no chassis number and can be changed using the **PXI/PCI Explorer** applet: 23 in this example.

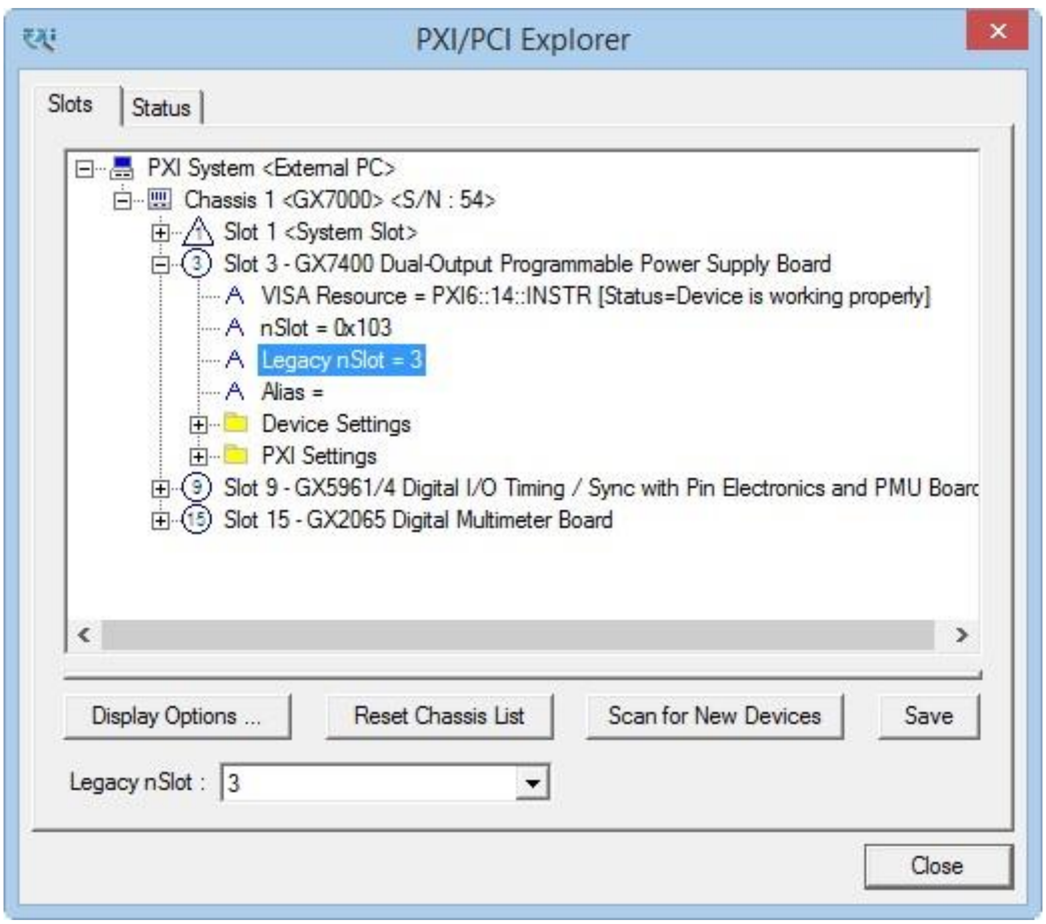


Figure 4-1: PXI/PCI Explorer

2. **VISA** – This is a third-party library usually supplied by National Instruments (NI-VISA). You must ensure that the VISA installed supports PXI and PCI devices (not all VISA providers supports PXI/PCI). GXFPGA setup installs a VISA compatible driver for the GXFPGA board in-order to be recognized by the VISA provider. Use the GXFPGA function **GxFpgaInitializeVisa** (*szVisaResource, pnHandle, pnStatus*) to initialize the driver's board using VISA. The first argument *szVisaResource* is a string that is displayed by the VISA resource manager such as NI **Measurement and Automation** (NI_MAX). It is also displayed by Marvin Test Solutions **PXI/PCI Explorer** as shown in the prior figure. The VISA resource string can be specified in several ways as the following examples demonstrate:

   - Using chassis, slot: "PXI0::CHASSIS1::SLOT5"
   - Using the PCI Bus/Device combination: "PXI9::13::INSTR" (bus 9, device 9).
   - Using the alias: for example, "COUNTER1". Use the PXI/PCI Explorer to set the device alias.

   Information about VISA is available at http://www.pxisa.org.

Board Handle

The **GxFpgaInitialize** and the **GxFpgaInitializeVisa** functions return a handle that is required by other driver functions in order to program the board. This handle is usually saved in the program as a global variable for later use when calling other functions. The initialize functions do not change the state of the board or its settings.

Reset

The Reset function sets the board to a known default state. A reset is usually performed after the board is initialized. See the Function Reference for more information regarding the reset function.

Error Handling

All the GXFPGA functions returns status - *pnStatus* - in the last parameter. This parameter can be later used for error handling. The status is zero for success status or less than zero for errors. When the status is error, the program can call the **GxFpgaGetErrorString** function to return a string representing the error. The **GxFpgaGetErrorString** reference contains possible error numbers and their associated error strings.

Driver Version

The **GxFpgaGetDriverSummary** function can be used to return the current GXFPGA driver version. It can be used to differentiate between the driver versions. See the Function Reference for more information.

Programming Examples

The README.txt located on the GXFPGA folder contains a list of the GXFPGA programming examples provided with the GXFPGA software. Examples are provided for various programming languages including C, VB.NET, VB (6.0). ATEasy and more.

Distributing the Driver

Once the application is developed, the driver files (GXFPGA.dll, GXFPGA64.dll and the HW device driver files) can be shipped with the application. Typically, the GXFPGA.dll should be copied to the Windows System directory. The HW device driver files should be installed using a special setup program HWSETUP.EXE that is provided with GXFPGA driver files (see Marvin Test Solutions\HW folder) or a standalone setup HW.exe. Alternatively, you can provide the GXFPGA.exe setup to be installed along with the board.

Chapter 5 - GXFPGA Tutorial and Example

Introduction

This tutorial will go over the basic workflow to start designing and loading a FPGA configuration for the Gx3500. The contents will entail:

- Downloading and installing the FPGA design tool
- Creating a new FPGA Design project with the Cyclone III as the target device
- Setup the pin assignment to work with the GX3500 and Cyclone III FPGA
- Use the graphical design tool to create an example FPGA configuration
- Compile the project and generate the SVF and RPD programming files
- Loading the board with the generated programming files
- Testing the design using the Gx3500 Front Panel software and ATEasy

The example configuration is broken down into three phases, each with a distinct function:

- **Phase 1:** Take two values located in PCI Registers and generate a Sum (Adder) which can then be read through a third PCI Register.
- **Phase 2:** 2 to 1 multiplexer to choose between the 10 MHz Clock and the PCI Clock which will be output on one of the FlexIO pins. The clock will be selected through a PCI Register.
- **Phase 3:** A simple dynamic digital sequencer with a memory depth of 32 double words (written to through the PCI bus) driven by a PLL that continuously outputs digital patterns to the first 32 FlexIO pins.

The source code for the examples in this chapter is provided in the Examples\Quartus folder.

Downloading Altera Design FPGA Design Tools

The Marvin Test Solutions Gx3500 User programmable FPGA board can be designed using the free Altera Quartus II Web Edition or Subscription Edition design tool. This FPGA design tool allows end users to generate fully featured FPGA designs that can be downloaded to the Gx3500 board using the Marvin Test Solutions GXFPGA software API or software front panel. Other 3rd party tools can also be used to design the FPGA. Before proceeding with this tutorial, you must have Altera Quartus II installed on your PC. More information about this tool and how to download it can be found at http://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html.

Create New Project

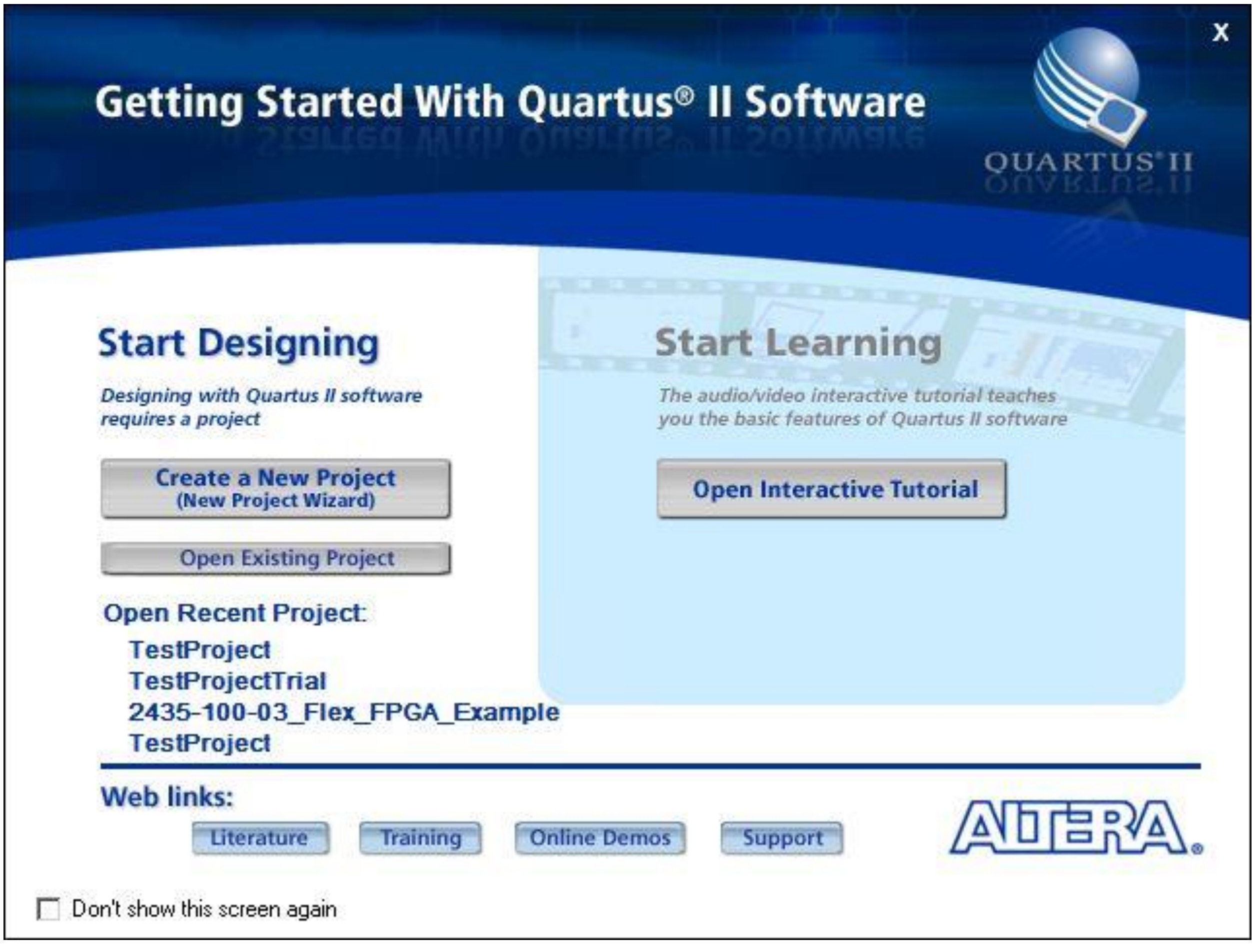


Figure 5-1: Quartus II Start Dialog

After installing Quartus II Web Edition, start the application and select C**reate a new Project** to start the New Project Wizard or select **File, New**, **New Quartus Project**.

Click on **Next** and then select the Project Folder and enter **FPGATutorial** as the project name.

Click on **Next** twice (skip the adding files window).

Device Selection

The next window will allow you to select the FPGA target device. Select **Cyclone III** as the Family and **EP3C55F484C8** as the Available Devices selection.

Click on **Next** twice (skip the Specify Tools window).

A window summarizing all the choices made for the creation of this project is shown. Click on **Finish**.

Pin Assignment Setup

You should now have an empty skeleton project loaded in Quartus II. Before you can get started on the FPGA design, you must assign the FPGA pins distinct names so that you can reference them in your design. This can be accomplished by running a **TCL script** which contains all the information necessary to configure the pin assignments. These pin assignments are unique to this the Cyclone III FPGA and the GX3500 in particular. The following table lists all the pin assignments and their respective descriptions. The Pin Alias's listed in the table are the pin names you will be using in your design to reference the actual hardware pins on the FPGA.

Pin Alias (Node Name)	**Description**
Clocks	
10Mhz	10 MHz Reference Clock Signal from the PXI Backplane
PCIClock	33 MHz PCI Bus clock
RefClk	80 Mhz Reference Clock onboard the GX3500
RefClka	80 Mhz Reference Clock onboard the GX3500
PCI Bus	
Addr[2..19]	The PCI Address lines from the PCI bus
FDt[0..31]	PCI Data lines from the PCI bus
CS[1..2]	Chip Select lines from the PCI bus. CS[1] is for PCI BAR1 and CS[2] is for PCI BAR2.
RdEn	PCI Read Enable line from the PCI bus
WrEn	PCI Write Enable line from the PCI bus
PXI Bus	
PxiTrig[0..7]	PXI Bus trigger signals
StarTrig	PXI Star Trigger signal
I/O	
FlexIO[1..160]	The physical IO Channels
Misc	
Spare[0..2]	Do Not Use
IRQ	Interrupt input pin, Pulse width >20nSecRising edge (transition from 0 to 1) activates the IRQ, falling edge (transition from 1 to 0) clears the IRQ.
Spare[4..6]	Do Not Use
FSpr[0..5]	Spare Signals from the User FPGA and Expansion Board
MClr	FPGA Master Clear, Active Low
TP[0..5]	Test Points located on the GX3500 PCB

Table 5-1: Pin Assignments Table

To configure the pin assignment first the .TCL configuration script should be added to the project. To add the script to the project, right click on the **Files** icon in the **Project Navigator** window and click on **Add/Remove Files in Project…** In the dialog box, click on the **…** button and browse for GX3500Pins.tcl file in the C:\Program Files\Marvin Test Solutions\GXFPGA folder. Click Open and then the **Add** button.

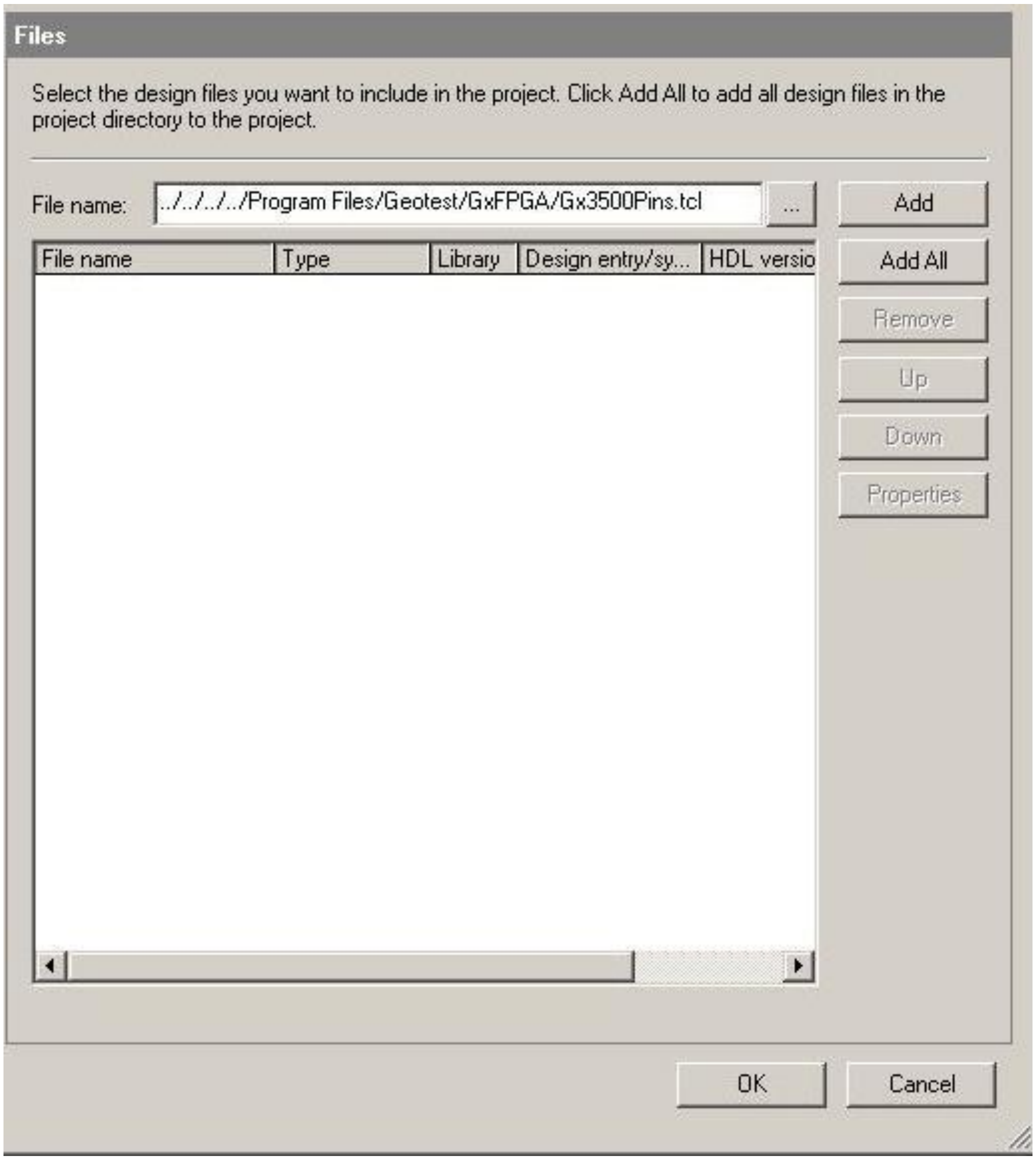


Figure 5-2: Add Tcl Script to Project

Then click on Tools | TCL Scripts … Select the configuration script file, **GX3500Pins.tcl** and click on **Run**. This will configure your FPGA pin assignments.

You can view the pin assignments by running the Pin Planner application which is found in the Tasks list as highlighted below:

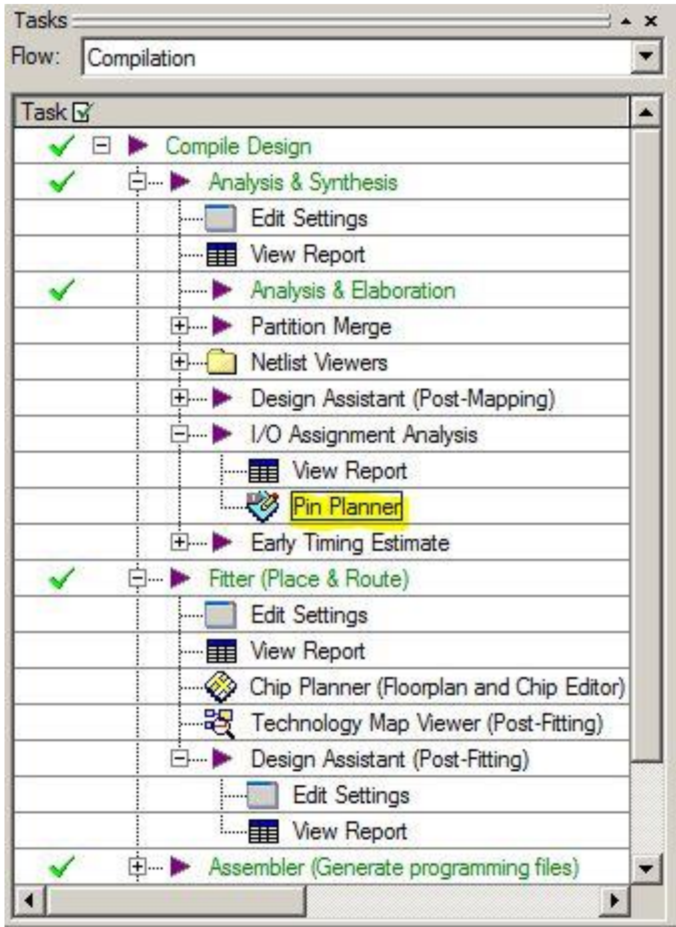


Figure 5-3: Task Flow

The Pin Planner will display a matrix of the physical FPGA pins and their mapped names as well as the I/O standard supported by the pin. These mapped names are used in the FPGA design, as wire names and I/O pins, to connect to the physical connections of the FPGA.

Creating Design File

You must now create a design file as part of the project. Click on **File**, **New**, **Block Diagram/Schematic File** as shown in Figure 5-4**.** Go to **File** | **Save As** and name the file **TutorialDesign** and click **OK**. The new design file, **TutorialDesign.bdf** will be added to your project. Right click on the file and select **Set As Top-Level Entity**. Double click on it to open the file. You will be presented with a blank schematic view.

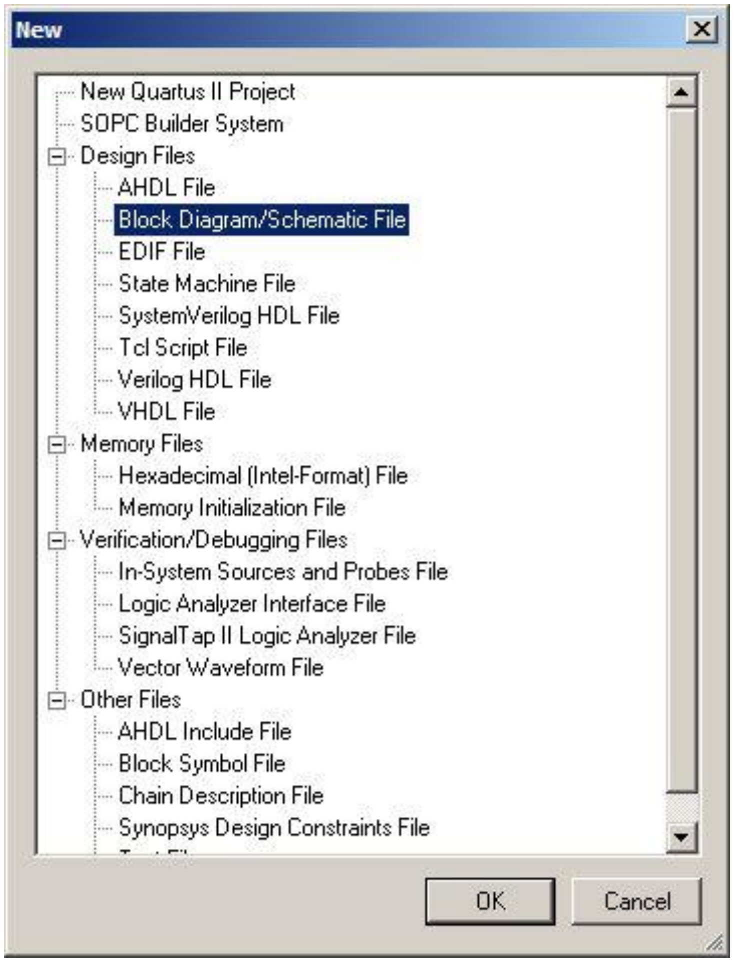


Figure 5-4: New Block Diagram

You can now begin schematic entry.

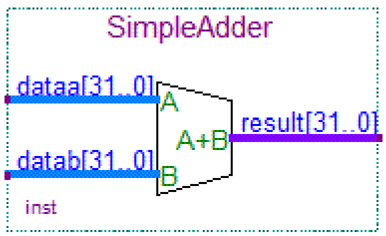
Note: There is more than one way to accomplish the following designs.

Phase 1: Creating the FPGA design - 32 bit Full Adder

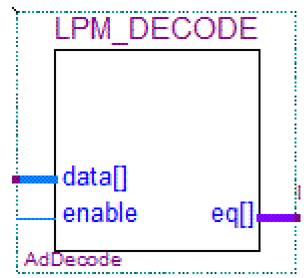
This design will take two double word (32 bit) values, located in the first two double words in the Register space (byte offset 0x0 and 0x4), and add them together. The sum of the two values will be immediately output to the third double word in the Register space (byte offset 0x8).

Components Used

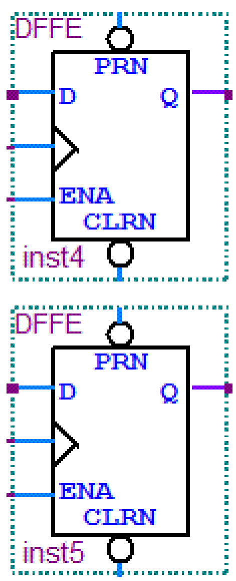
1x 32 bit Full Adder



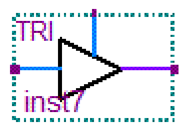
1x Decoder



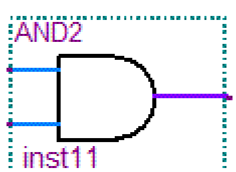
2x D Flip Flops



1x Tristate Buffer



2x AND Gate



1x Constant

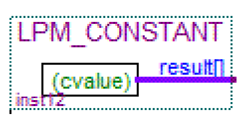


Figure 5-5: Phase 1 Adder Components

Design

First start with creating the circuitry required to decode the PCI Address when data is to be written from the PC to the FPGA. This circuit will be used in all three functions of this example project. The signals required for PCI Write access will be the **PCI Clock**, **Write Enable**, **Chip Select 1**, and some **PCI Address lines**. The PCI Address lines 5 to 2 will be fed to a decoder which will generate a 32-bit value, and the result will be ANDed with the Chip Select 1 bit. Each Chip Select bit represents a certain PCI BAR access (GX3500 has two bars, memory and register memories). Bit 1 represents BAR1 of the PCI memory space (bit 2 for BAR2). BAR1 is the general-purpose Control Register BAR for the GX3500. The results of the AND operation will be once again ANDed to the Write Enable PCI signal.

Double click on the blank space in the schematic view and select **lpm_decode** from the **Megafunction, Gates** directory.

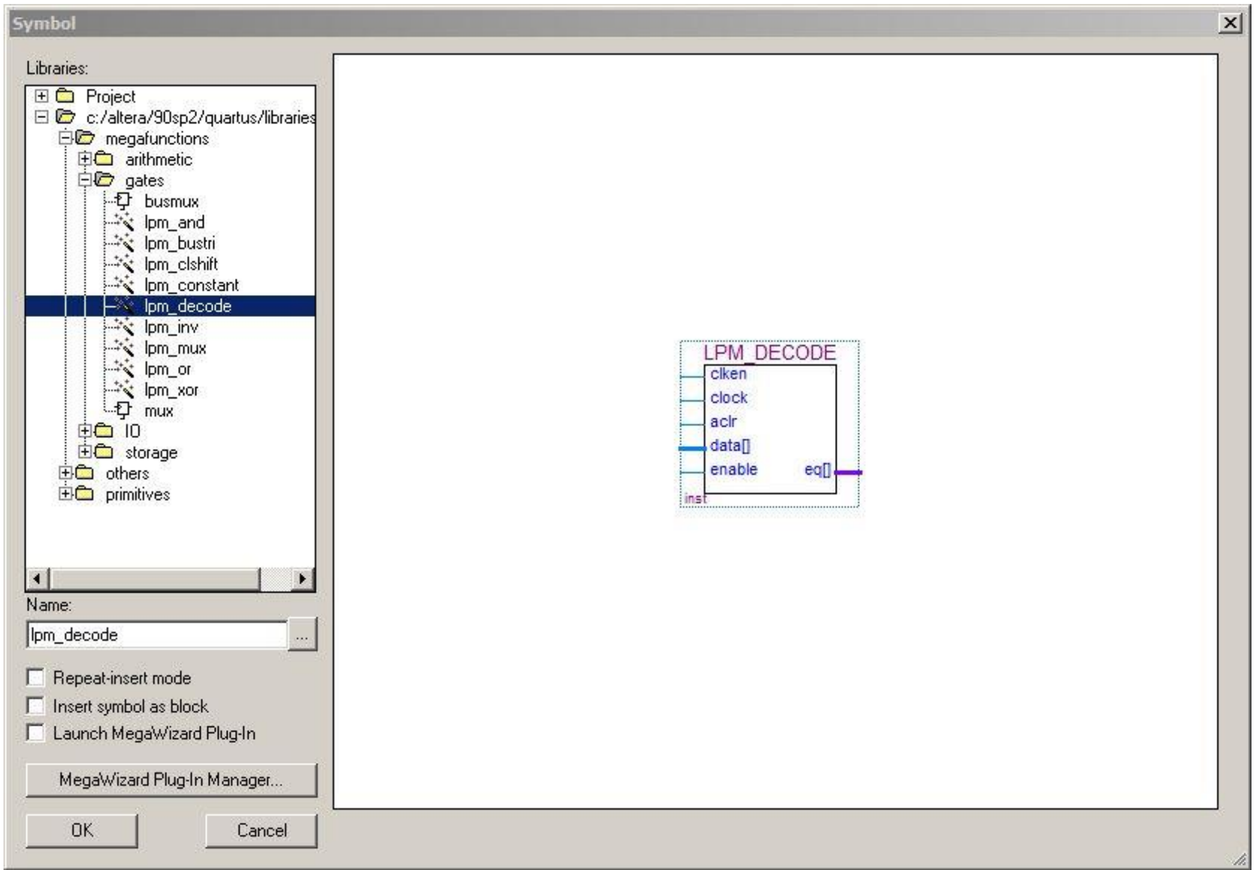


Figure 5-6: Symbol Insert Dialog Box

Make sure the Launch **MegaWizard Plug-In** checkbox is unchecked.

Click **OK** and place the symbol on the blank design document.

Now that the Decoder has been placed, some of its parameters have to be set. Right click on the Decoder symbol and select **Properties**. Click on the **Parameters** Tab. Set the **Width** and **Decodes** properties as shown below:

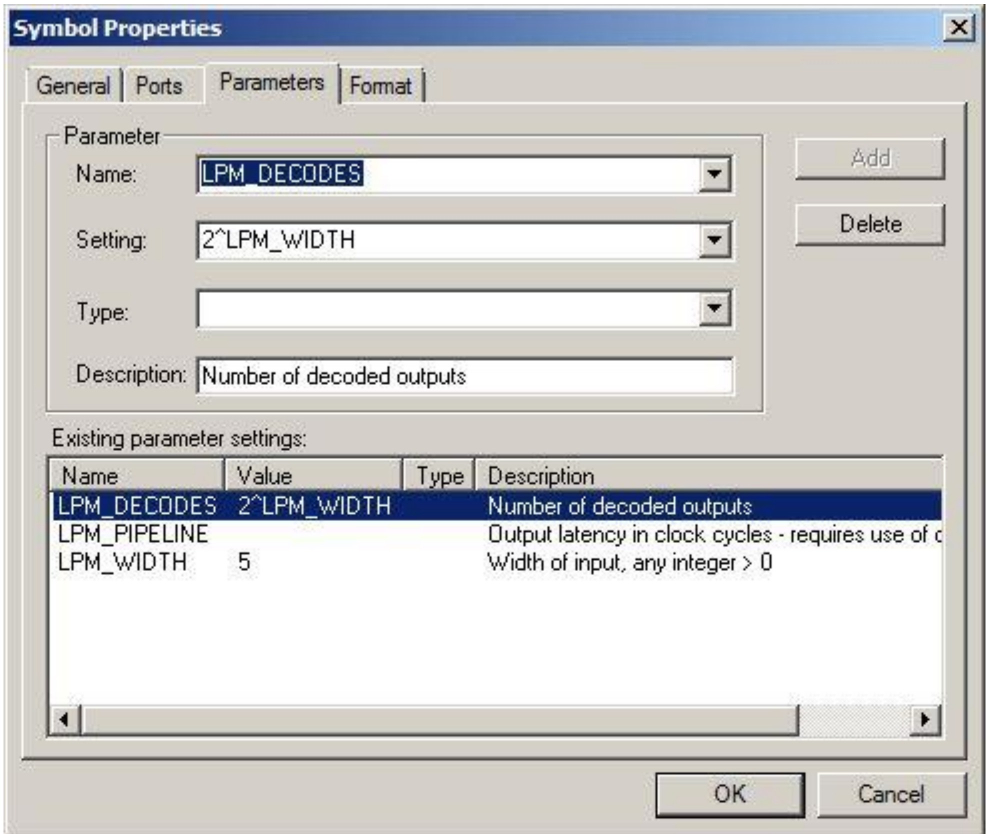


Figure 5-7: Decoder Properties

Click **OK** when done. Place another symbol on the design by double clicking on the design document, and selecting **Input Pin** from **Primitives**, **Pin**, **Input**. After placing the input pin symbol, rename it to **Addr[6..2]**. The symbol will now represent 5 PCI address lines that will be used to communicate with the PC.

Also, place 2 AND gates after the Decoder and a few more input pins with the appropriate names **DecAddr, Sel and WE** as the following figure shows:

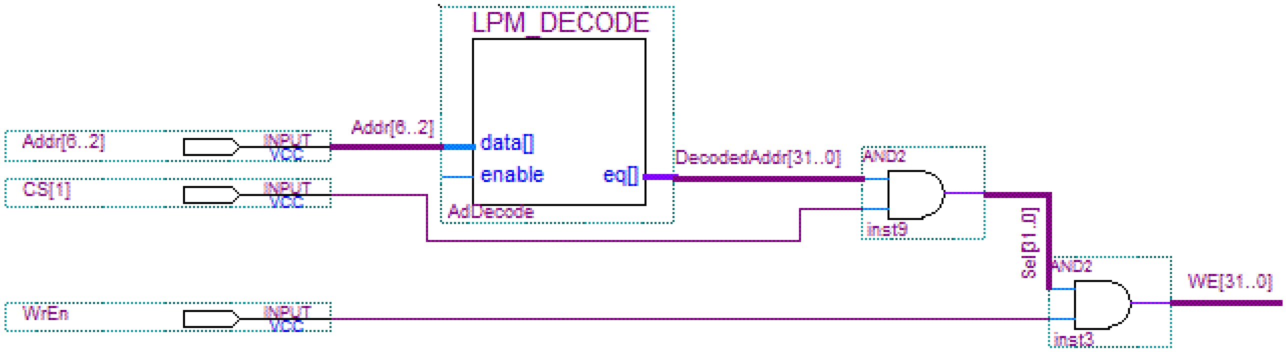


Figure 5-8: PCI Address Decoder Circuit

Note: To wire several signals together (as a bus), such as Addr[6..2] or Sel[31..0], use the **Bus Wiring Tool** highlighted in red below.

Figure 5-9: Bus Wiring Tool

Now that the PCI address decoder circuit is complete, we can feed the appropriate bits from the WE bus to D Flip Flops that will store data clocked in from the PCI data lines. For example, the first double word in PCI memory (representing the first number to be summed) will be written to a D Flip Flop with its enable line tied to WE[0] (the first bit in the WE bus). The second double word to be added will be written to another D Flip Flop with its enable line tied to WE[1]. Finally, the PCI Clock signal (33Mhz) will be used as the clock source of the D Flip Flops. Note that each bit of the Sel and WE busses represent a consecutive double word address (bit 0 corresponds with byte 0, bit 1 corresponds with byte 4, bit 2 corresponds with byte 8 etc.)

Place two D Flips Flops (located at **primitives**, **storage**, **dffe**) and an input pin named **PCIClock**. We will leave the D Flip Flops input lines (D) disconnected for now. Eventually the PCI data lines will drive these inputs.

Wire the output of the AND gate to D Flips Flops as shown below.

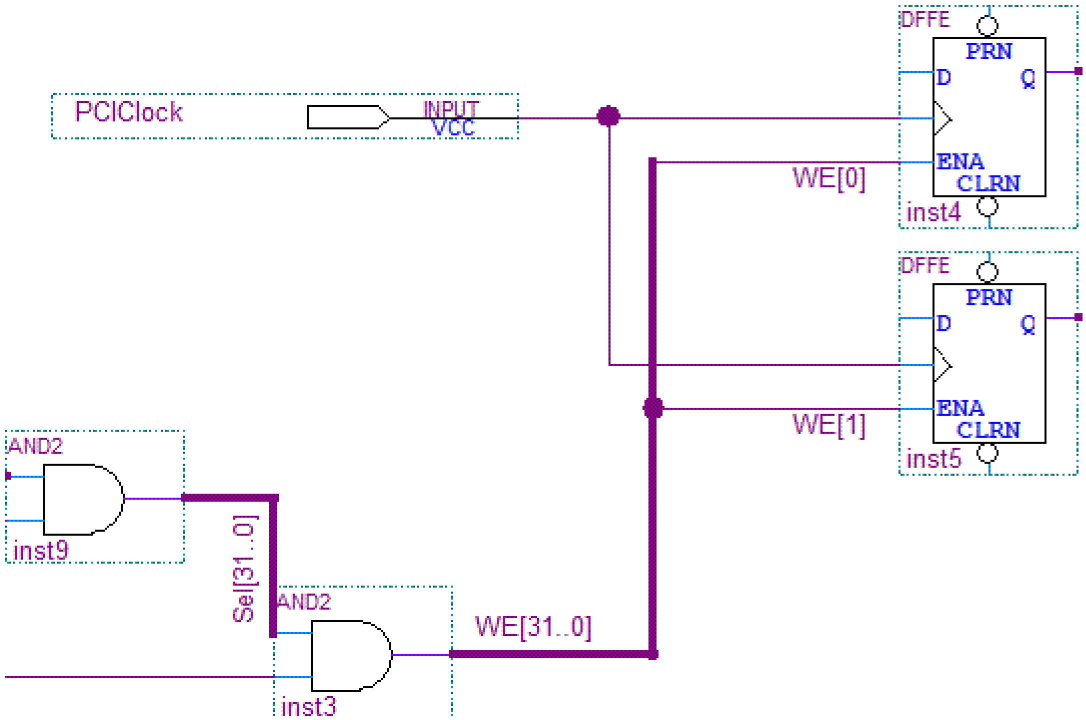


Figure 5-10: D Flip Flops

The D Flips Flops will feed a 32-bit adder and the resulting summation will be wired to the PCI data lines so that the PC can read the result.

The 32-bit adder will be placed onto the design using the MegaFunction wizard tool. This tool will customize a component by allowing you to make selections through a wizard.

Double click on the design window and navigate to **megafunctions, arithmetic, lpm_add_sub**. Make sure the **Launch Megafunctions Wizard** checkbox is selected and click **OK**. You will see a dialog box like the following:

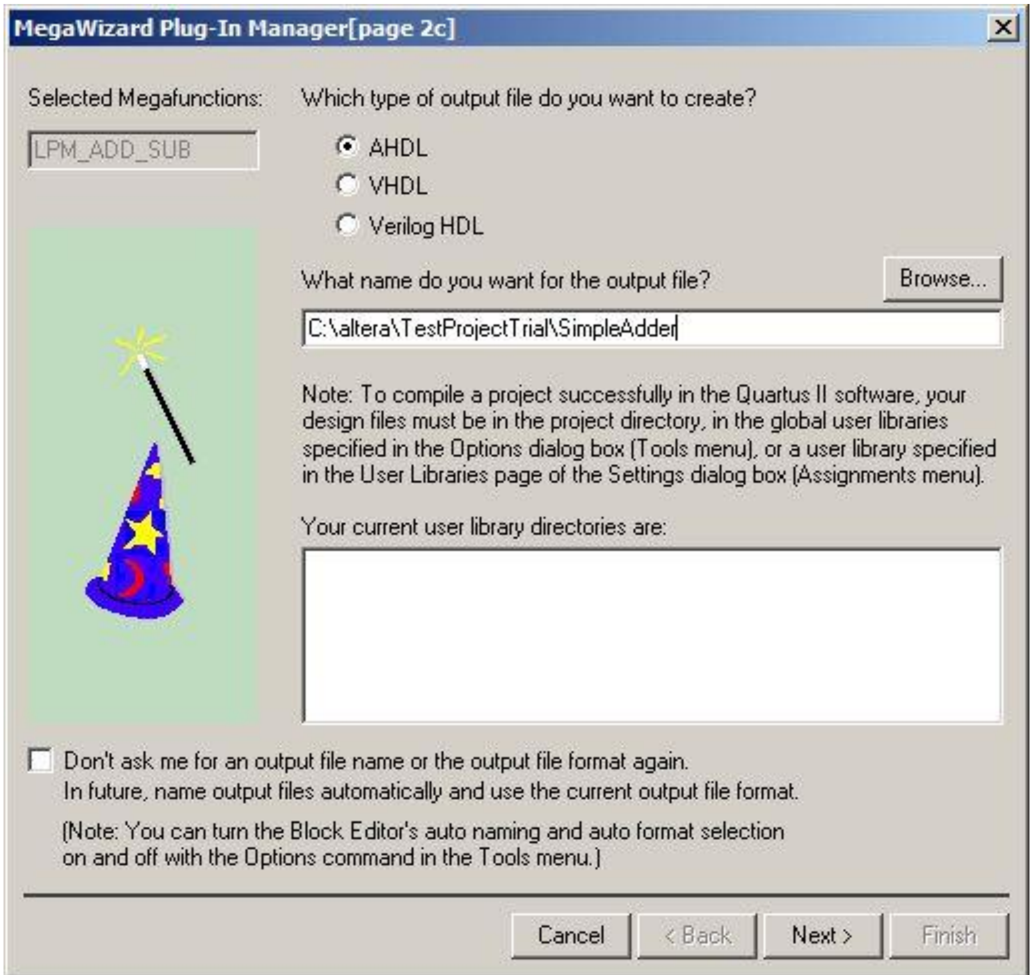


Figure 5-11: Adder Wizard

Name the output file **SimpleAdder** and make sure the path is the same as your project. Click **Next** and enter **32** as the data width.

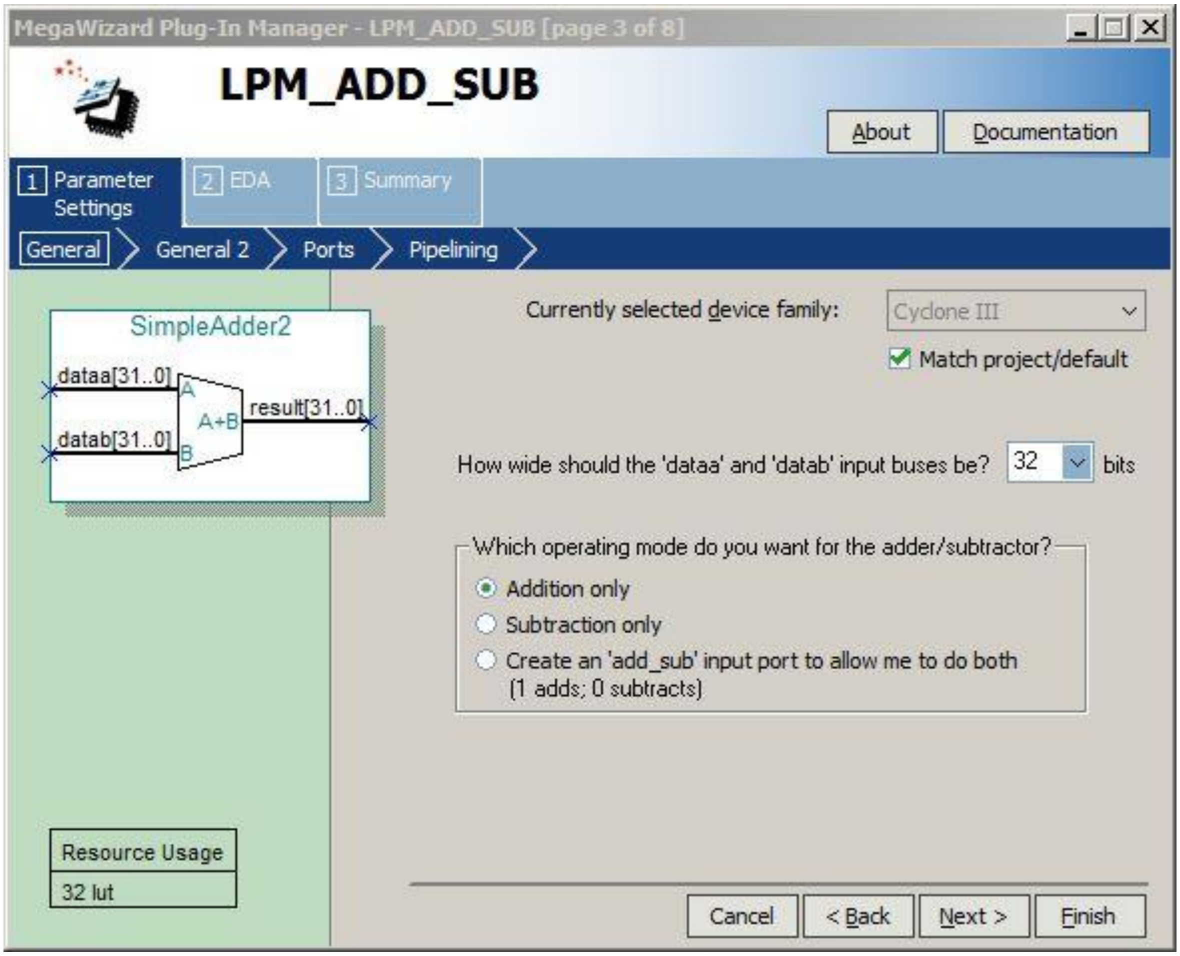


Figure 5-12: Adder Wizard 2

Click **Next** through the rest of the wizard and keep the default choices. Finally, the dialog box will show the newly created design files that will be included in your project. Click **Finish** and place the newly created Adder in your design. Wire the adder to the flip flops and add an AND gate, Read Enable pin, and tristate buffer as the following shows:

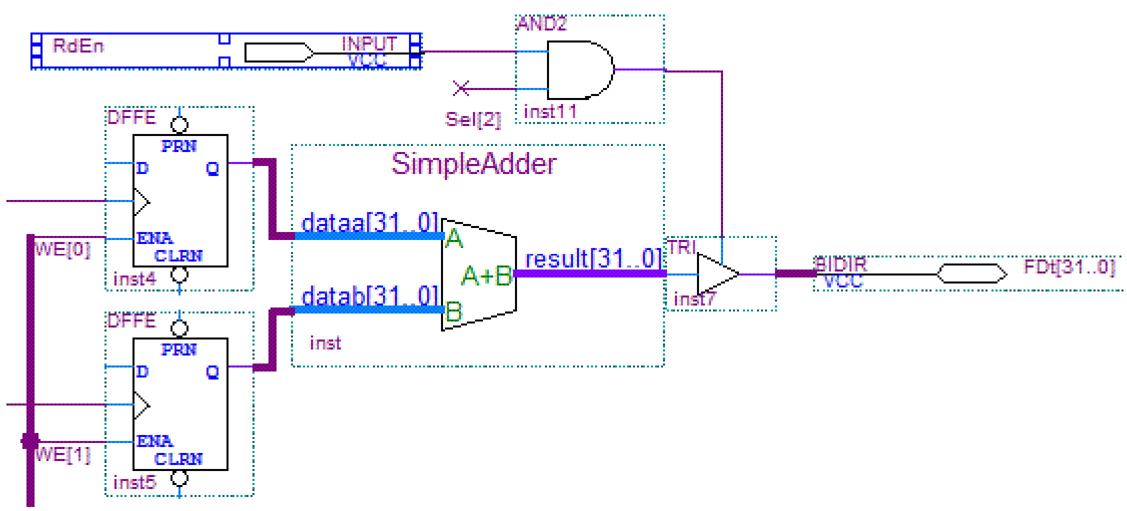


Figure 5-13: Adder Circuit

Note that we are using the FDt[31..0] PCI data lines as bidirectional pins since we will be reading and writing to the PCI bus. The Tristate buffer is used to select whether the Adder will be driving the PCI Data lines or not. The Tristate buffer is controlled by the 3rd bit of the decoded PCI Address ANDed with the Read Enable line. When both signals are high (Sel[2] and RdEn) it indicates that the PCI Bus is expecting the 3rd double word to be written to the PCI bus. In our case, this means the 32-bit result from the Adder.

The inputs to the D Flips Flops can now be wired to the PCI data lines (FDt) as follows:

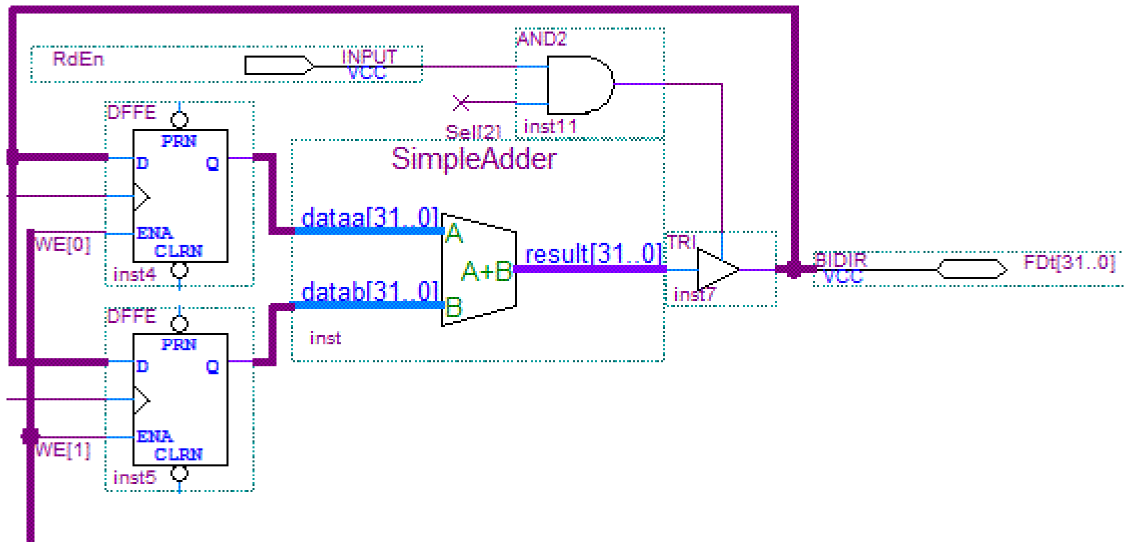


Figure 5-14: Adder Circuit with PCI Bus Connection

Now that the design has been completed, a revision number should be added so that the end user can read it back from the PCI bus at the 32nd register double word location (byte address 0x7C).

Including a revision number constant to the design is a Marvin Test Solutions standard practice that we recommend end users to follow. The revision constant is 32 bits long and is read as a hexadecimal number such as 0x3564A000. The first two digits of the hexadecimal number represent the company, in this case 35 is for Marvin Test Solutions designs. The next two digits are the design specific code, 64 in this case. And the last 4 digits, A000, is the revision of the design.

A constant component needs to be placed in the design (LPM_CONSTANT). When placing this component make sure that the "Launch MegaWizard Plug-In" selection is unchecked. After placing the component, right click on it and select properties to set the value and width of the constant as the following figures show:

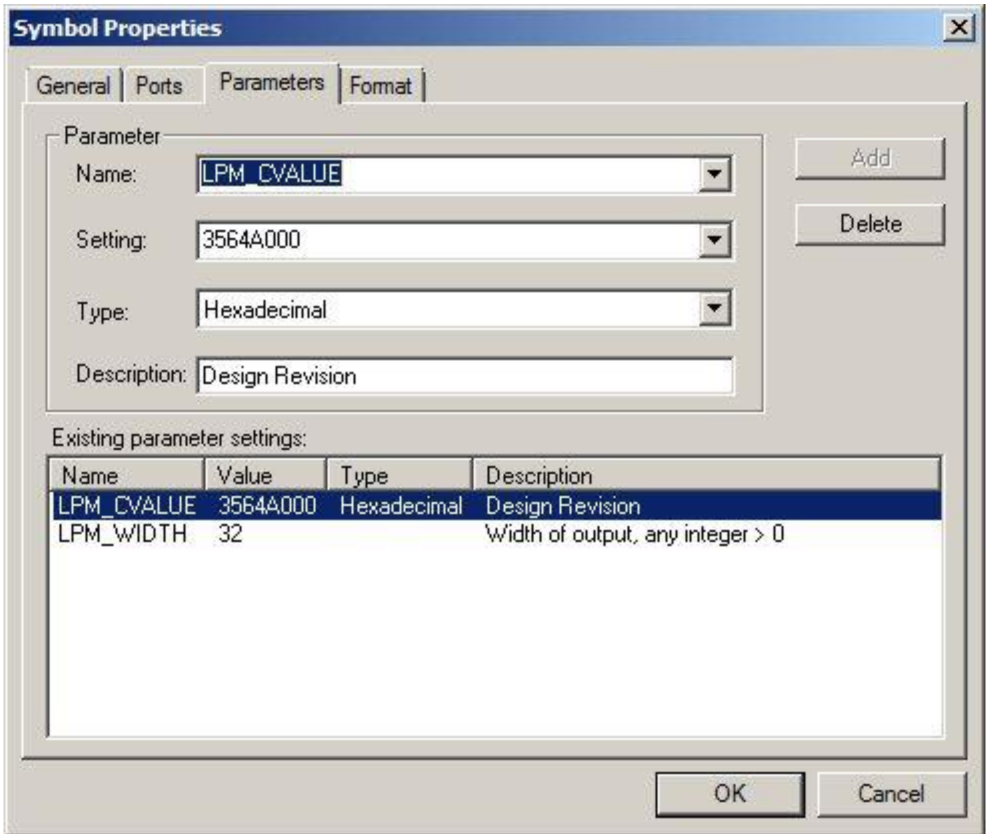


Figure 5-15: Symbol Properties

Now place the 2 port AND gate and the tristate buffer. You can rotate it, as shown in Figure 5-16, by right clicking on the symbol (after placing it) and select "Rotate By Degrees | 90".

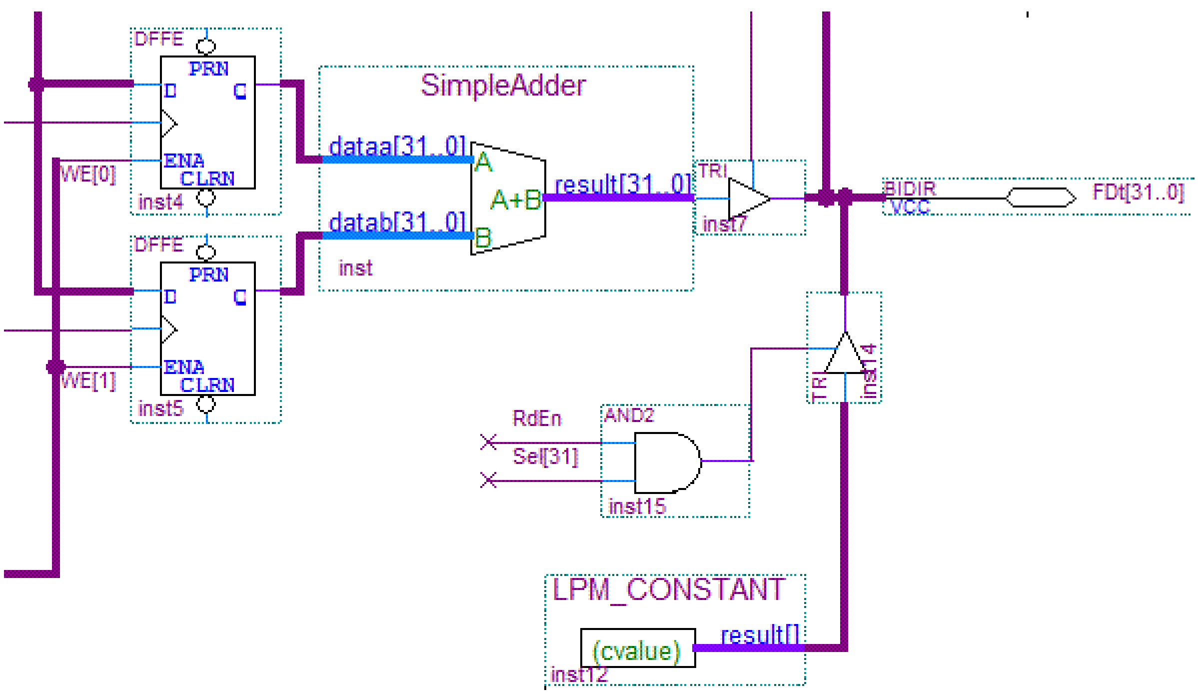


Figure 5-16: Adder Circuit with Revision Constant

Phase 2: Creating the FPGA Design - 2 to 1 Clock Mux

This design will output either the PCI Clock (33Mhz) or the 10Mhz clock to IO Channel 33 (**Pin 31** on Flex I/O A connector) depending on what was written to the 4th double word in the PCI register space (byte offset 0xC). A 1 will select the 10Mhz clock signal, and a 0 will select the PCI clock signal.

Components Used

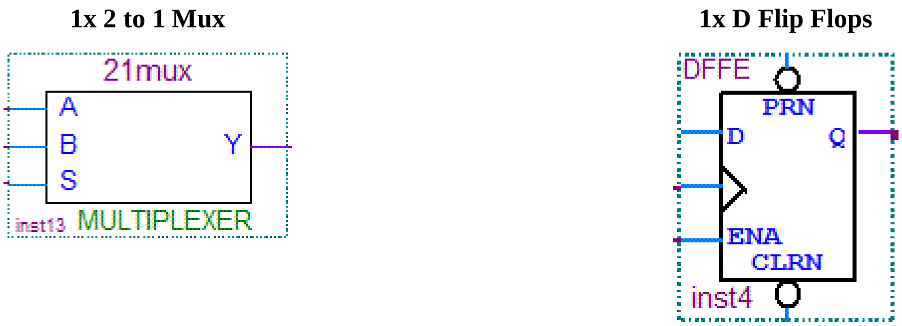


Figure 5-17: Phase 2 Mux Components

Design

You will now build upon the tutorial project to add the functionality of a 2 to 1 Clock Mux. The 10Mhz clock will be brought into the design by an input pin. The PCI Clock signal input pin is already present in the Phase 1 circuit, so this will be reused. FlexIO[33] (IO Channel 33) will be used to output the selected clock to the outside world.

Place the 2 to 1 Mux symbol by double clicking on the design area and selecting mega functions **others, maxplus2, mux21.**

Create three wires attached to the D, ENA(enable) and B inputs of the D Flip Flop. Name the wires **FDt[0], Sel[3]**, and **PCIClock** respectively. Note that you did not have to place new input pins to access these signals. This is due to the fact that input pins were already created for these signals in the Phase 1 design. Therefore, you can just use named wires to tap into the same input pins.

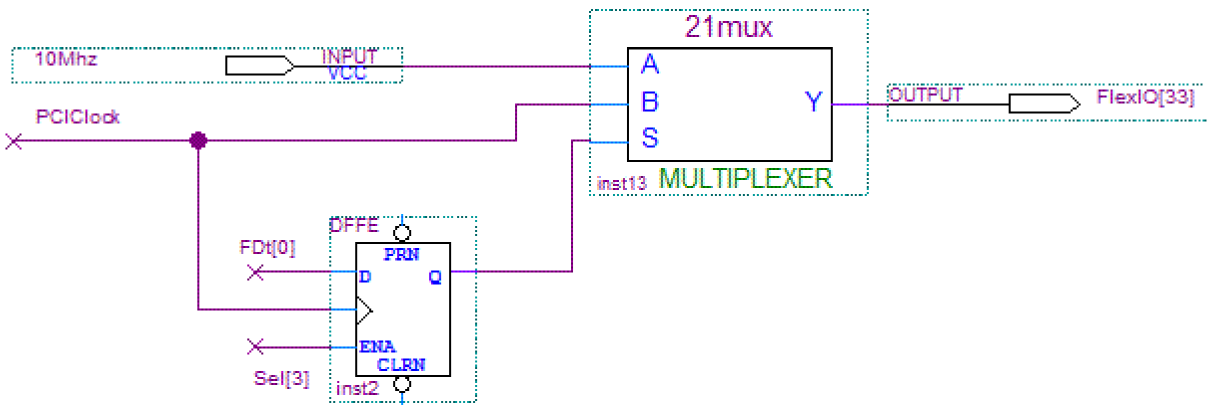


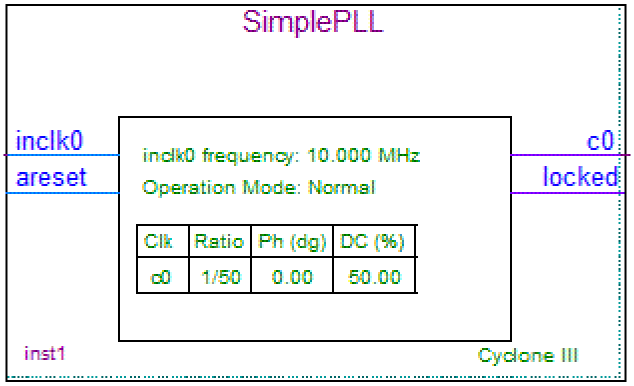
Figure 5-18: Clock Mux Circuit

FDt[0] is the first bit of the PCI data bus. This bit can either be 0 or 1, to indicate which clock source to choose. Sel[3] is the 4th bit from the decoded PCI Address. When this bit is high, it indicates that the PCI Bus is addressing the 4th double word (byte offset 0xC) of the Register space for the GX3500. In our case, the value of this double word is used to select which clock is selected by our Mux.

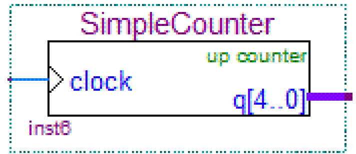
Phase 3: Creating the FPGA Design - 32 bit Dynamic Digital Pattern Sequencer

Components Used

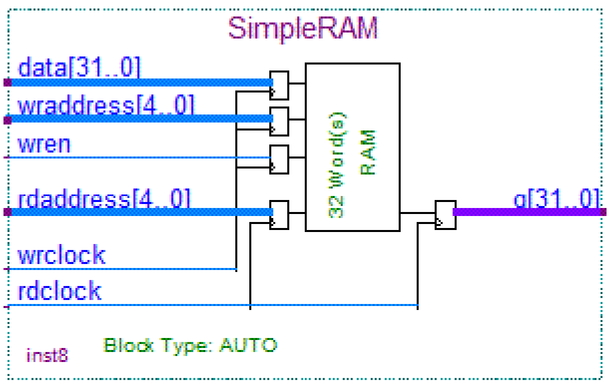
1x PLL



1x 5 bit Counter



1x 32 by 32 bit RAM



1x AND gate

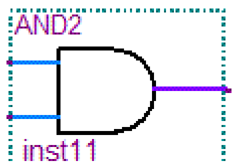


Figure 5-19: Phase 3 Dynamic Digital Sequencer Components

Design

This design functions as a simple dynamic digital pattern generator. A PLL drives a Counter which iterates through a 32 double word memory that outputs 32 bit wide digital patterns to the I/O Pins. The memory is loaded through the PCI bus, allowing users to program the device with vectors through the software front panel or the DLL API.

This phase will require the use of the **MegaFunction Wizard** to generate all three components, **PLL**, **RAM**, and **counter**. The wizard will allow you to customize the component for this particular application. The generated component will be stored in a file (.qip) that will automatically be included in the project.

First insert the PLL component by double clicking on an empty space in the design and clicking on **MegaFunction Plug-In Manager**. Choose to create a new MegaFunction variation and click **Next**. Then select the symbol called **ALTPLL** under the **I/O folder**. Name the new variation **SimplePLL** and click **Next**. The next dialog box will prompt you for the input clock frequency. We will be using a 10Mhz reference clock source so enter **10Mhz** into this field.

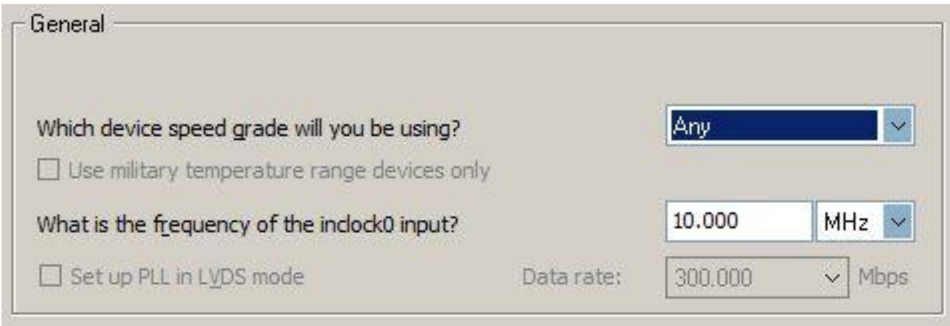


Figure 5-20: PLL Wizard Dialog Box 1

Proceed through the next few screens, with the default choices until you get to step 3 in the wizard entitled **Output Clocks**. Select **50** as the division factor as shown in the following figure:

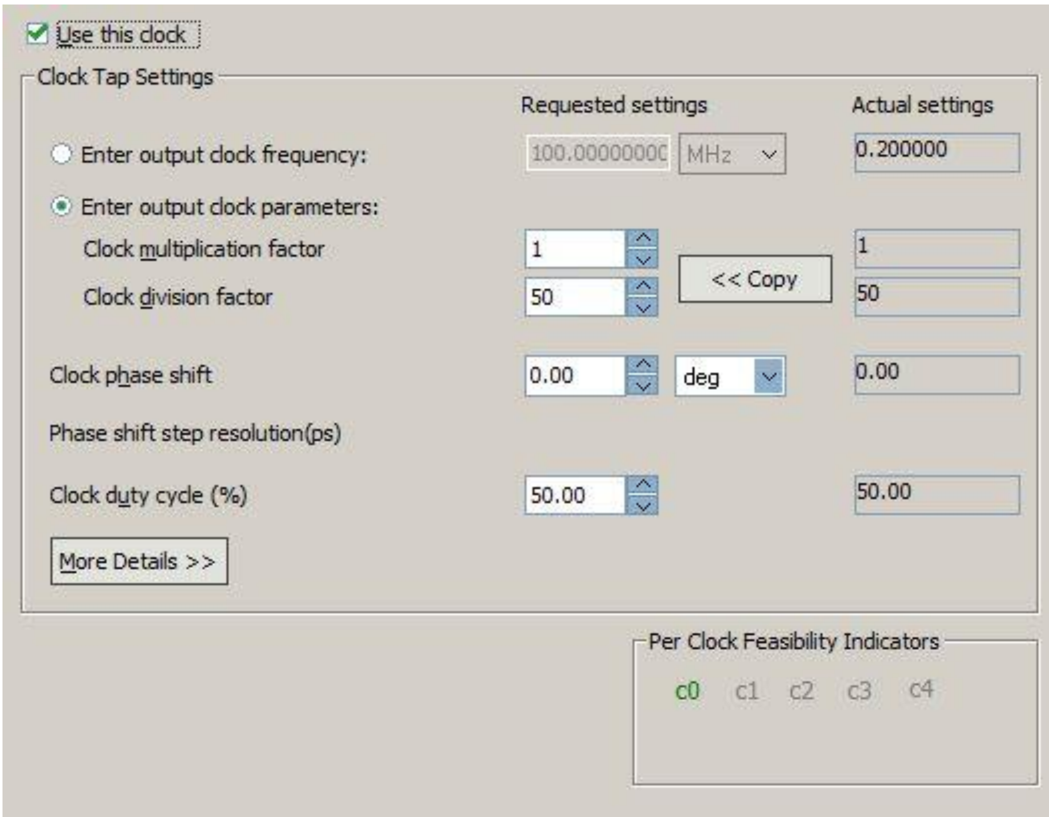


Figure 5-21: PLL Wizard Dialog Box 2

Click **Next** for the rest of the windows until you get to the last window showing you the files that will be created and then click **Finish**. The customized component will now be included in your project automatically so that you can start using it. Click **OK** to return to the design view, and then place the newly created symbol on your design.

Attach a wire to the inclk0 terminal of the PLL symbol, and name the wire **10Mhz**. This will connect the wire to the 10Mhz input pin that has already been created in the phase 2 design.

Repeat the previous steps to create a new custom component using the **MegaFunction Wizard** and select **LPM_COUNTE**R from the arithmetic folder. Name the custom component **SimpleCounter** and click next. Select 5 bits for the output bus width. We have chosen 5 bits for the width because we need to count from 0 to31 which requires 5 bits. You can now click next for the rest of the windows and finally click finish to place the symbol on your design.

Wire the c0 output terminal from the PLL to the clock input on the counter.

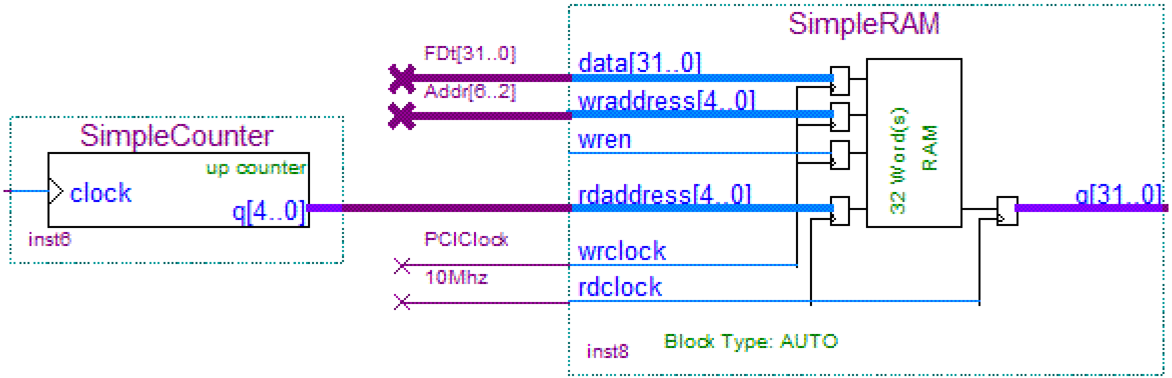


Figure 5-22: PLL and Counter Circuit

The last component needed is a 32 double word RAM. You will need to deploy the **MegaFunction Wizard** once again, and select the **2 port RAM** component from the **Memory Compiler** folder. Call the new component file **SimpleRAM** and click **Next**. Make sure to select 32 as the word length and 32 as the input width as the following figure shows:

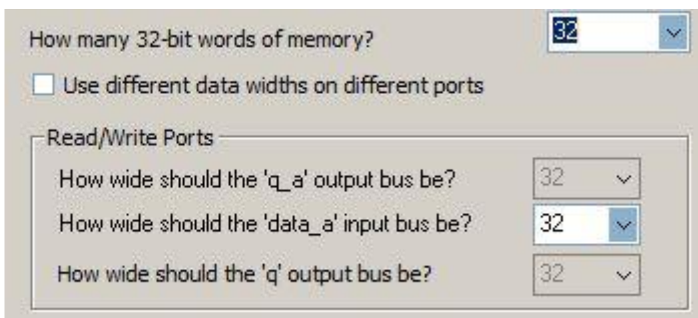


Figure 5-23: RAM Wizard Dialog Box 1

In the next window make sure to select a dual clock for reading and writing so that data can be written to the RAM from the PCI bus and read out to the IO pins concurrently.

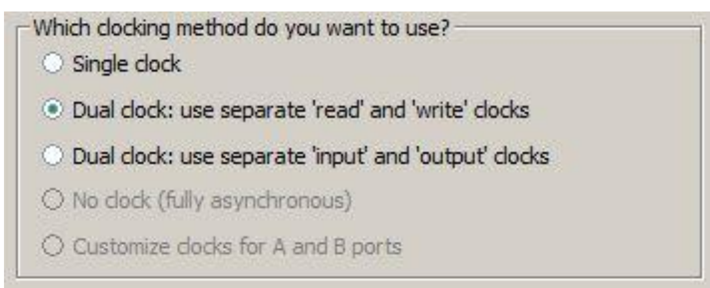


Figure 5-24: RAM Wizard Dialog Box 2

Click **Next** on the rest of the windows and click **Finish** placing the RAM component on your design. Wire the output bus, q[4..0], from the counter to the read address, rdaddress[31..0], of the RAM component.

Connect a bus to data[31..0] and wraddress[4..0]. Name these busses **FDt[31..0]** and **Addr[6..2]** respectively. Then connect wires to wrclock and rdclock and name the wires PCIClock, and 10Mhz respectively.

You will need to place an AND gate next to the RAM component and wire a new input pin called CS[2] and a wire named WrEn to it. The output of the AND gate should be connected to the wren input of the RAM. This AND logic ensures that only BAR2 PCI accesses are able to write to the RAM. This will allow us to use the FGPA Memory space to write out digital patterns to the sequencer instead of the FPGA Register space (which is being used for control). Note that when CS[2] is high, it signifies an access from BAR2.

Finally create a bus connected to the q[31..0] output from the RAM and name it **FlexIO[32..1]**. This connects the RAM output to the 32 physical IO pins.

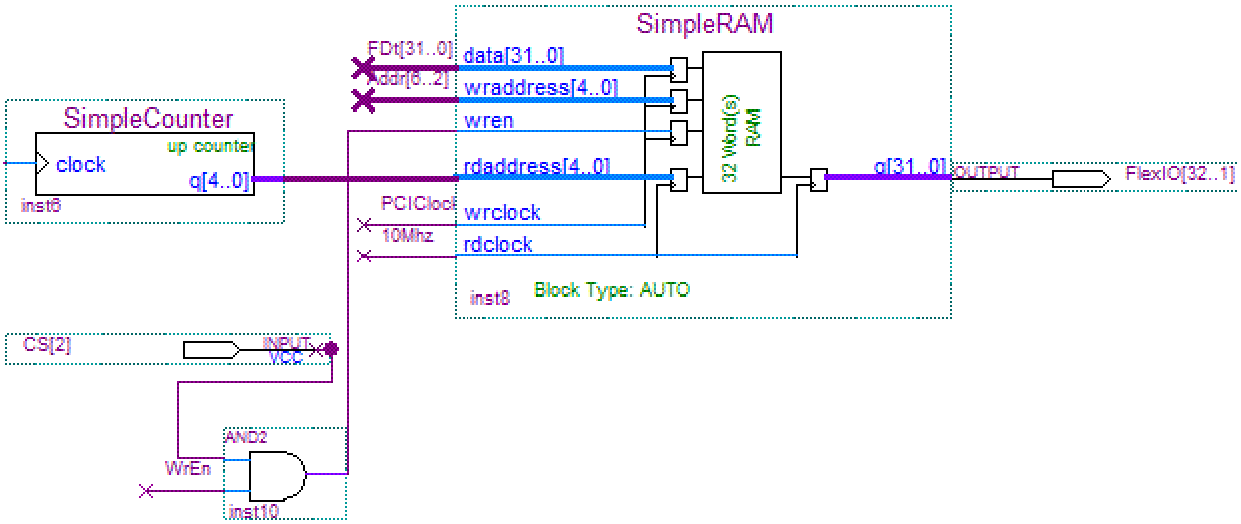


Figure 5-25: Dynamic Digital Sequencer Circuit

At this point the design is complete, continue with the next sections to generate SVF or RPD files and load your design to the GX3500.

Configure Project to Output SVF and RPD Files

To ensure that a SVF file is generated upon project compilation, go to the **Assignments**, **Device ...** and click on the **Device and Pin Options** button. Then click on the **Programming Files** tab and verify that the **Serial Vector Format File** checkbox has been selected.

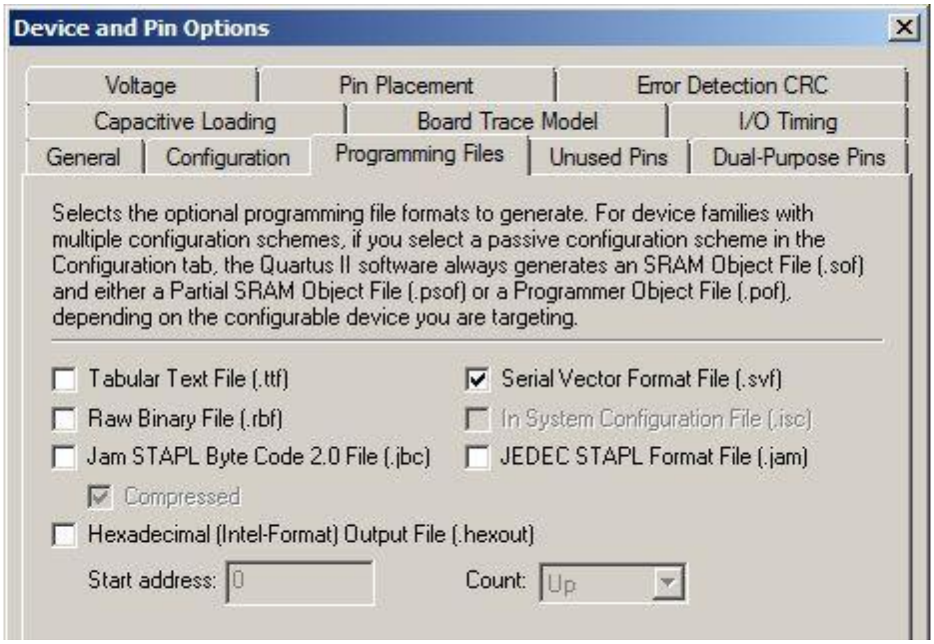


Figure 5-26: Select SVF as output file

Click on the **Configuration** tab and check the **Use Configuration Device** checkbox. Then select **EPCS16** as the configuration device from the drop down selection. Finally click on **OK** twice to exit the settings dialog boxes.

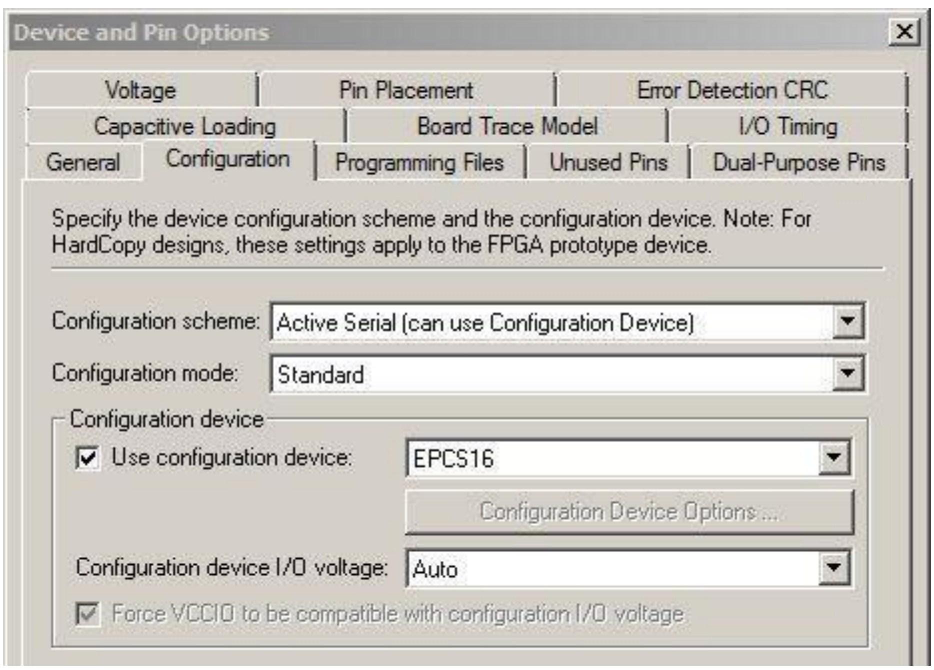


Figure 5-27: Select Configuration Device

Compile an Example Project and Build RPD and SVF Files

Click on the start compilation button to start the compilation process for the example project.

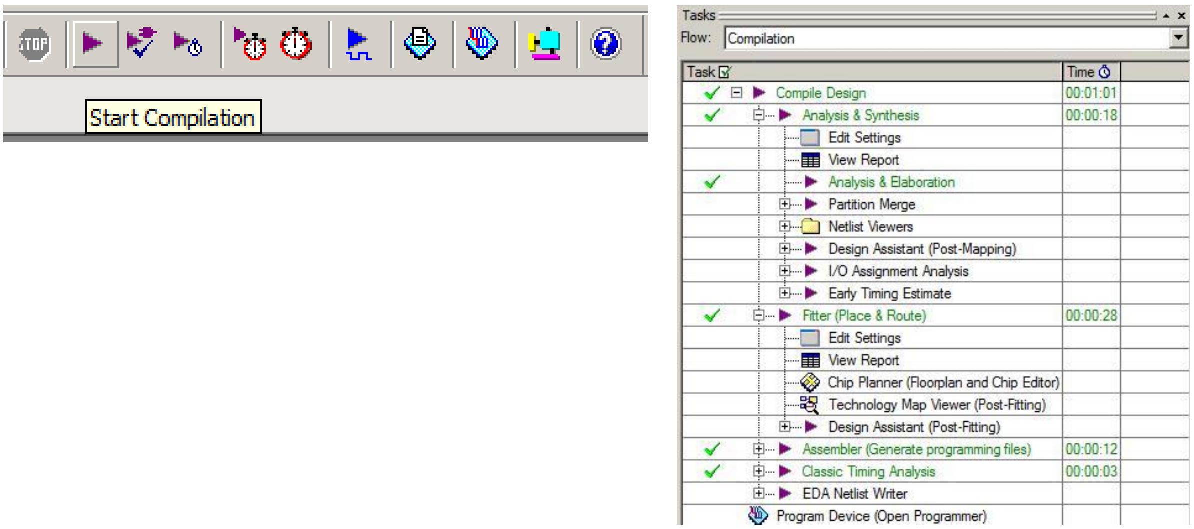


Figure 5-28: Compilation Tools and Status

The SVF file will be generated after the project compilation has finished. The **Compilation Task** window will show green check marks next to each major task to indicate completion.

To generate the RPD file, go to **File**, **Convert Programming Files …**

Select **Raw Programming Data File (.rpd)** as the **Programming file type** and **FPGATutorial.rpd** as the **File Name**. Click on the **Add File** button and select **FPGATutorial.pof**. The .pof file should now appear below the **POF Data** node as shown below. Finally, click the **Generate** button to create the RPD file.

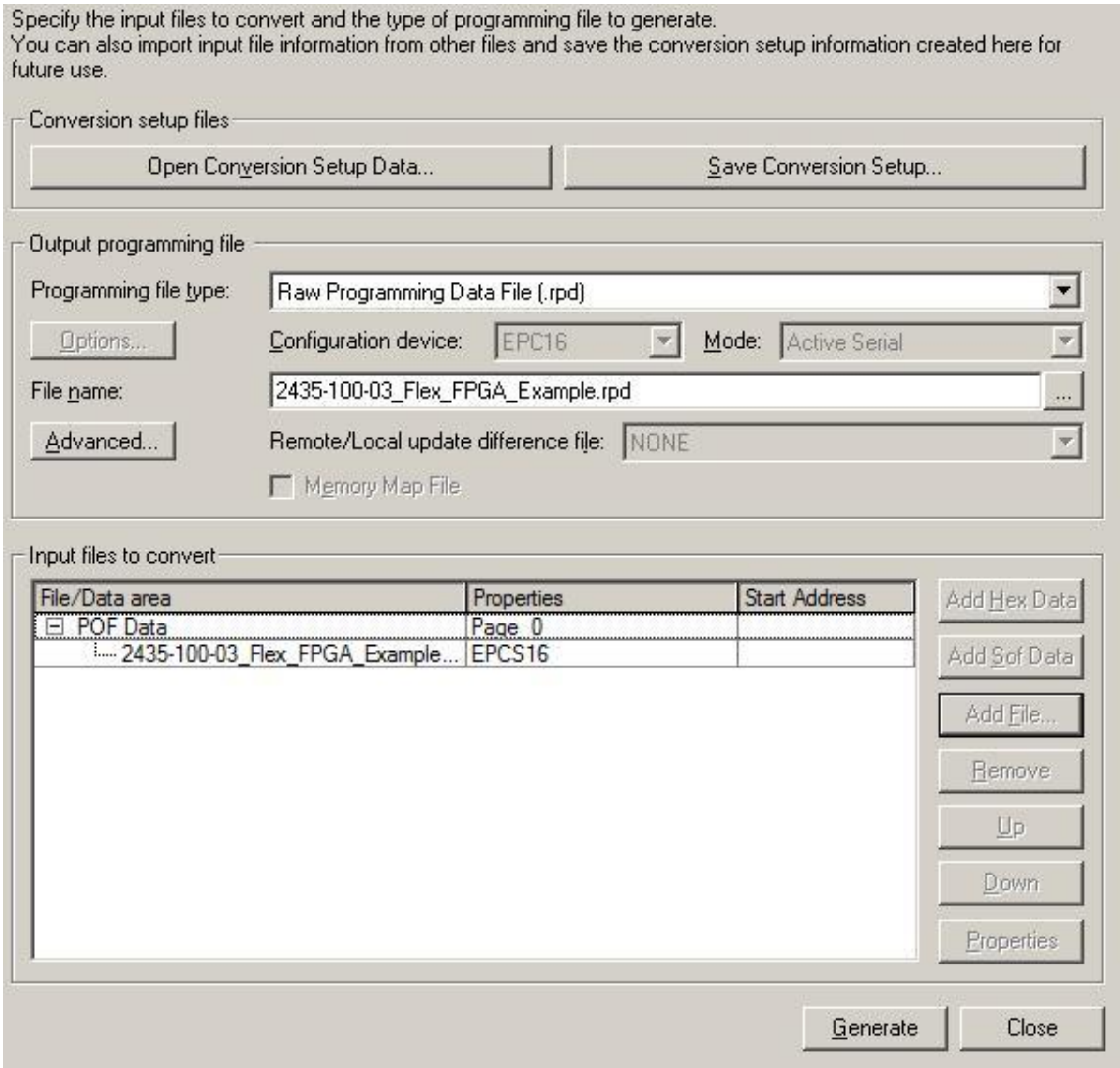


Figure 5-29: Convert Programming Files Dialog Box

Load Gx3500 with SVF File

Start the **GX3500 Panel** (from the Windows **Start** menu, **Marvin Test Solutions**, **GxFpga**) and initialize the instrument. Next, click on the **Volatile** radio box and then click on the Browse Button (…) to select the newly generated SVF file (**FPGATutorial.svf**). Finally click on the **Load** button to begin programming the card. You will see the progress bar indicate the status of the load. Once the load has completed, the status bar should be unfilled.

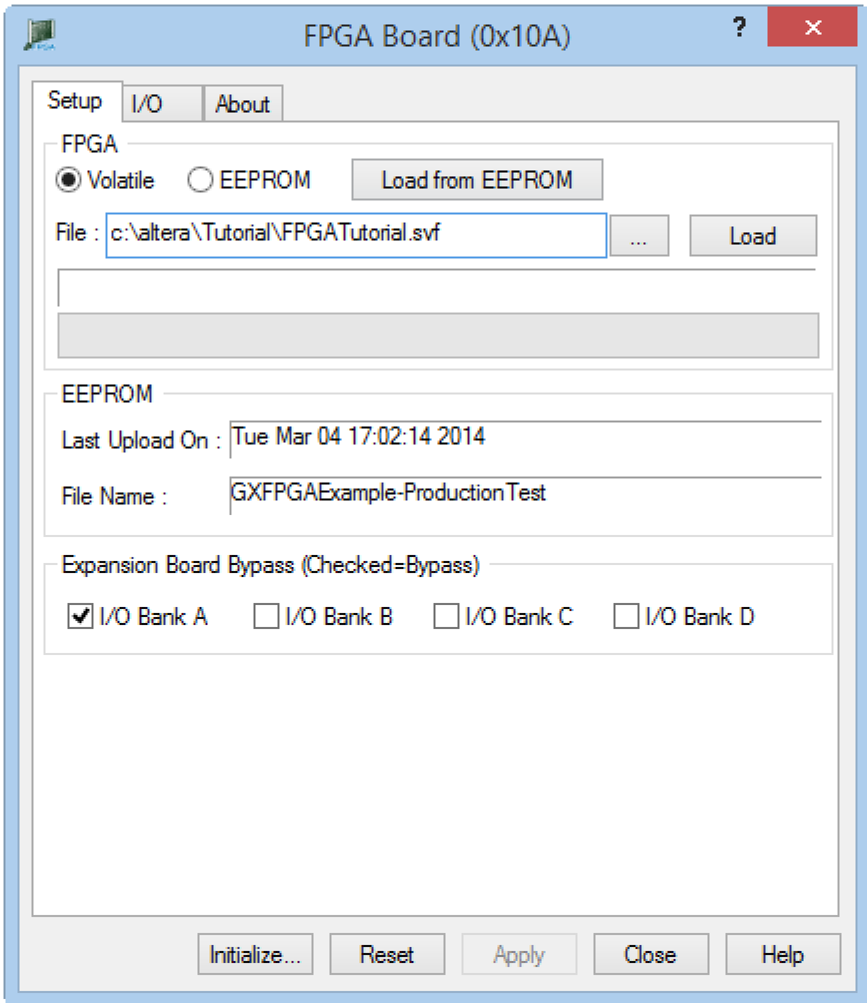


Figure 5-30: Software Front Panel

Before we proceed to testing the design, make sure that the **IO Bank A** checkbox is selected as shown above. This will allow the FPGA IO Pins to be routed directly to the front connector.

Testing the Design

Now that the design has been completed, compiled and loaded into the Gx3500, we can move on to the testing.

There are two ways to access the FPGA, either through the software front panel or through the driver API DLL. We will demonstrate the programming method using ATEasy to access the driver API DLL.

Adder Testing

The software front panel will be used to test Phase 1 of the design which adds two 32 bit numbers together. Click on the **I/O Tab** to get started. The Adder phase is controlled through the FPGA Register space.

Offset 0x0 points to the first 32-bit number that will be summed and offset 0x4 points to the second 32-bit number that will be summed. Write values to both these locations.

The sum can be obtained by reading the 32-bit value at offset 0x8. Verify that the correct sum is read back as shown in Figure 5-31.

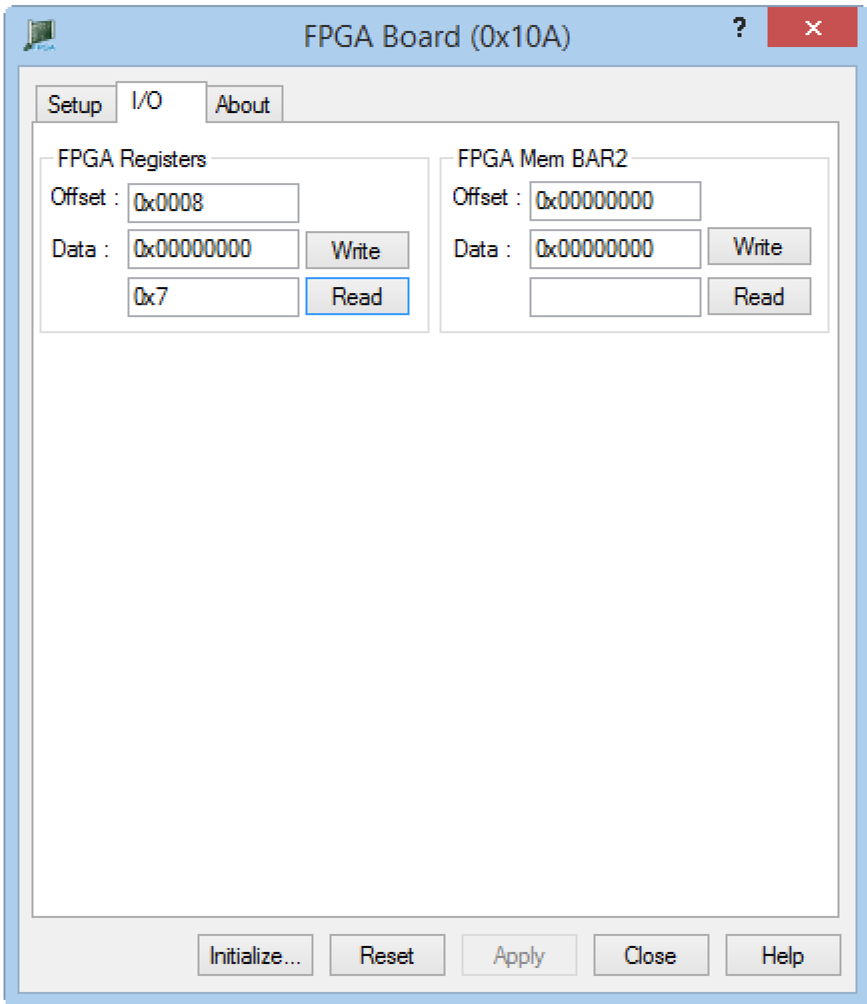


Figure 5-31: Using the Software Front Panel to read back the Sum

Clock Mux Testing

The software front panel will once again be used to test Phase 2 of the design. This part of the design uses a Mux to select between the PCI Clock and the 10 MHz reference clock. The selected clock is output to I/O Channel 33 which is located on pin 31 on the Flex I/O A connector of the GX3500. The Mux is controlled through the FPGA Register space.

Writing a 0x0 to offset 0xC will route the 33 Mhz PCI Clock signal to I/O Channel 33. Writing 0x1 to the same offset will route the 10 Mhz clock to I/O Channel 32. Try switching between both values while monitoring pin 31 with an oscilloscope. You should see the appropriate clock signals.

Digital Sequencer Testing

For this test, connect an oscilloscope to I/O Channel 1 (pin 35) to monitor the output signal of the sequencer. You can access the FPGA memory through the software front panel or through ATEasy. When using the software front panel, write values to the first 32 double words of the FPGA Memory space (offsets 0x0, 0x4, 0x8, 0xC etc.). As you write to these locations, the data patterns being output on I/O Channel 1 should be updating dynamically. If you fill the 32-double word memory with a clock pattern (alternating 1's and 0's), you should be able to measure a frequency of 100Khz.

When using ATEasy, include the GxFPGA.drv driver and set it up with the correct slot number. Add a variable called **i** of type **long**. You can then run the following code to write to the FPGA memory:

```
REDIM adwData[32]
adwData[0] = 1
For i=0 to 31
    FPGA Write Memory(i*4, 4, adwData[i])
Next
```

This code will set the first double word to 1 and the rest to 0's resulting in a frequency of 6.25 KHz.

Chapter 6 - GX3500 Expansion Boards

The GX3500 has provisions to accommodate a piggyback expansion board. Marvin Test Solutions provides a family of expansion boards to increase the functionality of the GX3500. In addition, custom expansion boards can be developed. The following information is provided to assist the user with developing expansion boards. In addition, the following standard expansion boards are described:

- **GX3501** - 80 Channel TLL Buffer expansion board for GX3500, this board when combined with the GX3500 is also known as **GX3601**.
- **GX3509** - 80 Channel Differential TLL expansion board for GX3500, this board when combined with the GX3500 is also known as **GX3609**.
- **GX3510** - 80 Channel M-LVDS Buffer expansion board for GX3500, this board when combined with the GX3500 is also known as **GX3610**.
- **GX3540** - ECL Expansion Board expansion board for GX3500, this board when combined with the GX3500 is also known as the **GX3640**.
- **GX3571** - Programmable Video Generator expansion board for the GX3500, this board when combined with the GX3500 is also known as the **GX3671**.

Expansion Board Design Guide

Board Description and Connectors

An expansion board mates with the GX3500 using four connectors P8-P11 and two mounting holes. Figure 6-1 depicts a bottom view of the expansion board and Figure 6-2 and Figure 6-3 detail the complete GX3500 with a typical expansion board assembly.

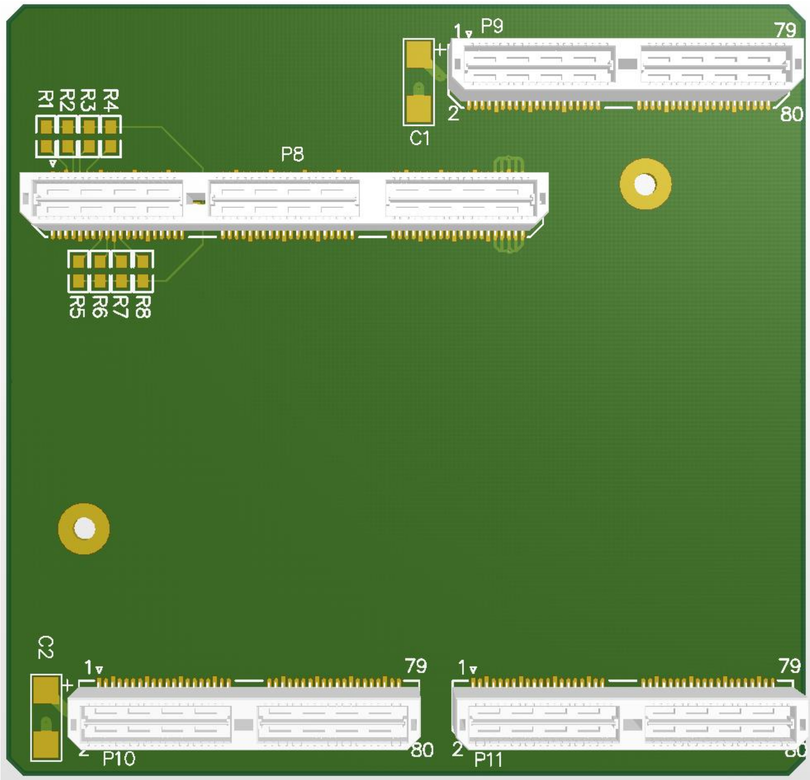


Figure 6-1: GX3500 Expansion Board – Bottom View

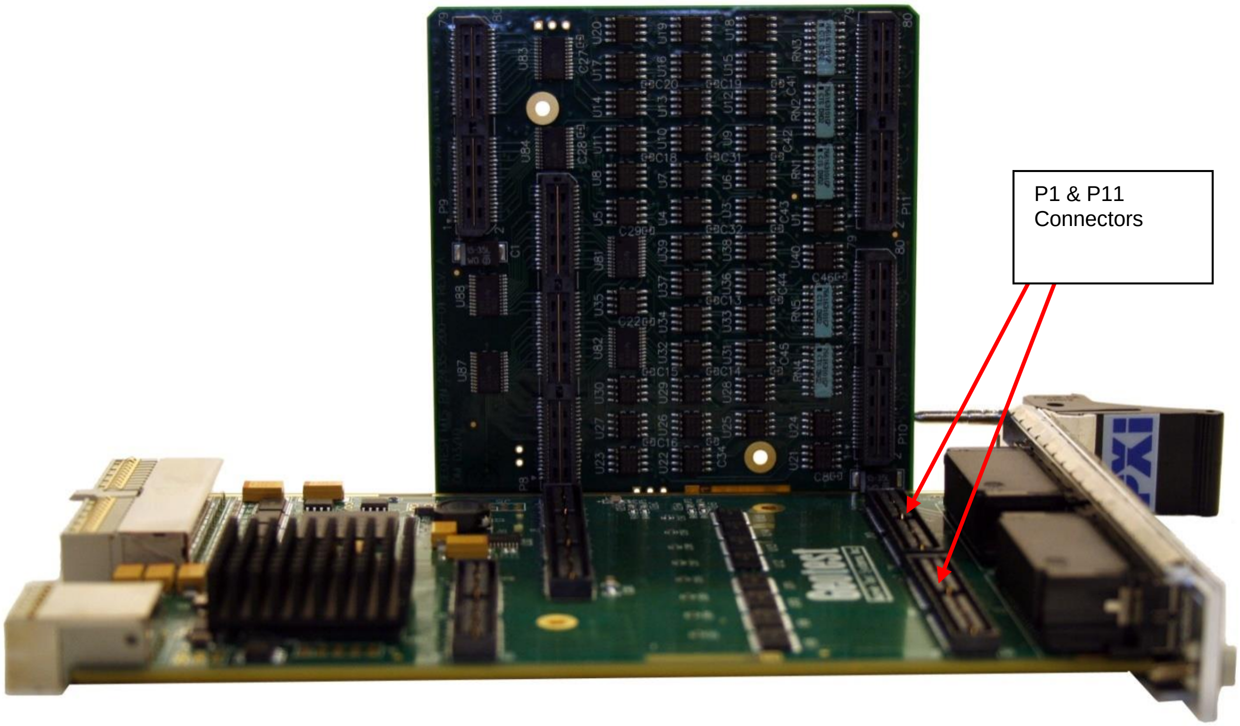


Figure 6-2: GX3500 Assembly with Expansion Board

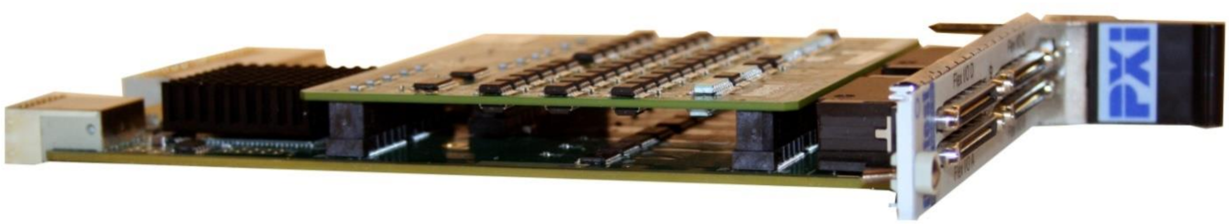


Figure 6-3: GX3500 with Assembled Expansion Board

Mechanical Guide

The locations of the mounting holes and connectors are critical to ensure a proper fit between the GX3500 and the expansion board. Figure 6-4 describes the mechanical details of a typical board and the locations of connectors and mounting holes. The figure presents a transparent view of the board from the top, with dimensions for critical component locations.

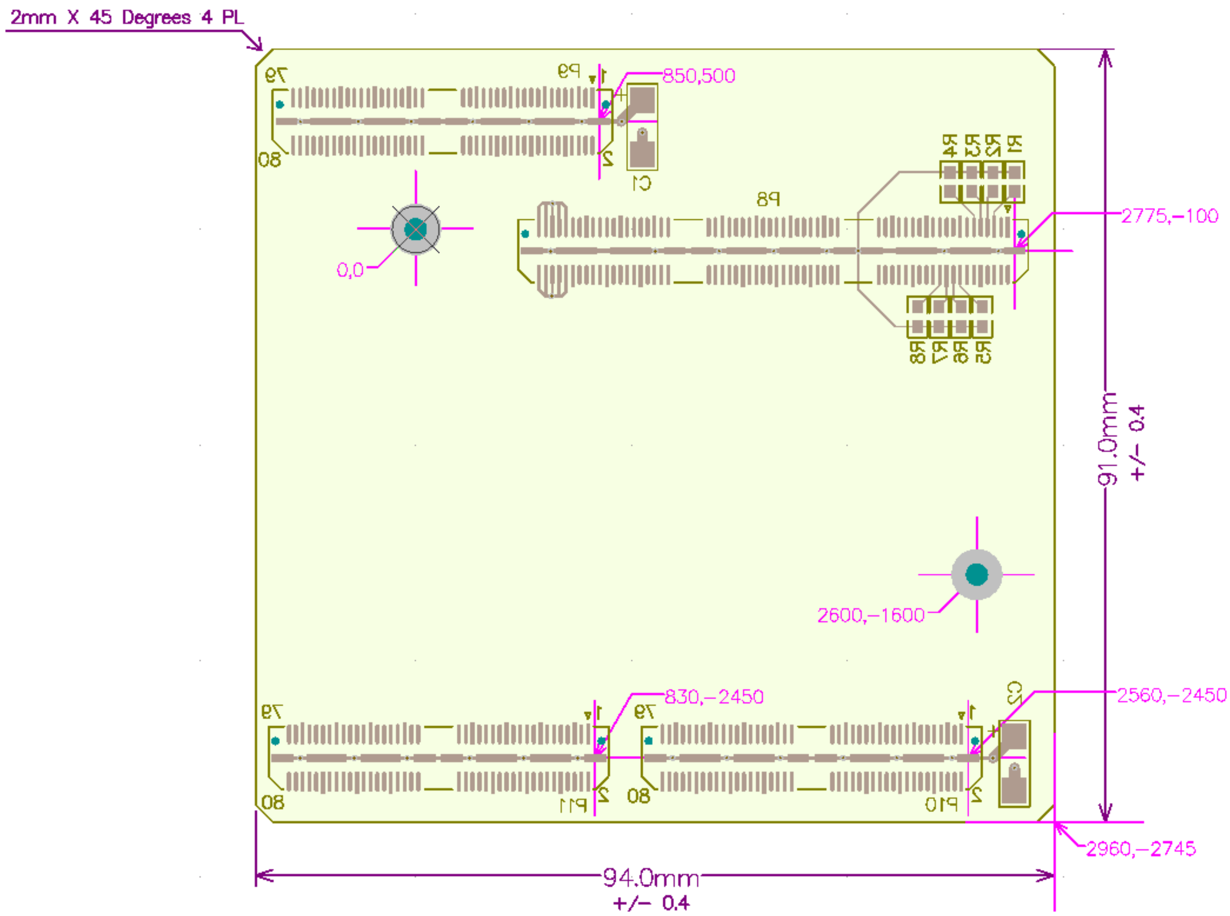


Figure 6-4: Mechanical Details – Top View of Typical Board

The coordinates for the connectors are pointing to the component reference point, which is the center of pad A of the footprint, as shown in Figure 6-5.

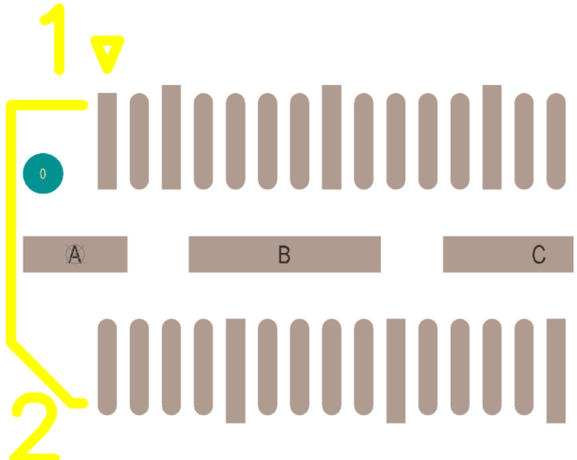


Figure 6-5: Component Reference Point

Figure 6-6 describes the recommended maximum dimensions for the expansion board and the recommended maximum component height. The maximum board area is about 100 Sq centimeters or about 16 sq inches.

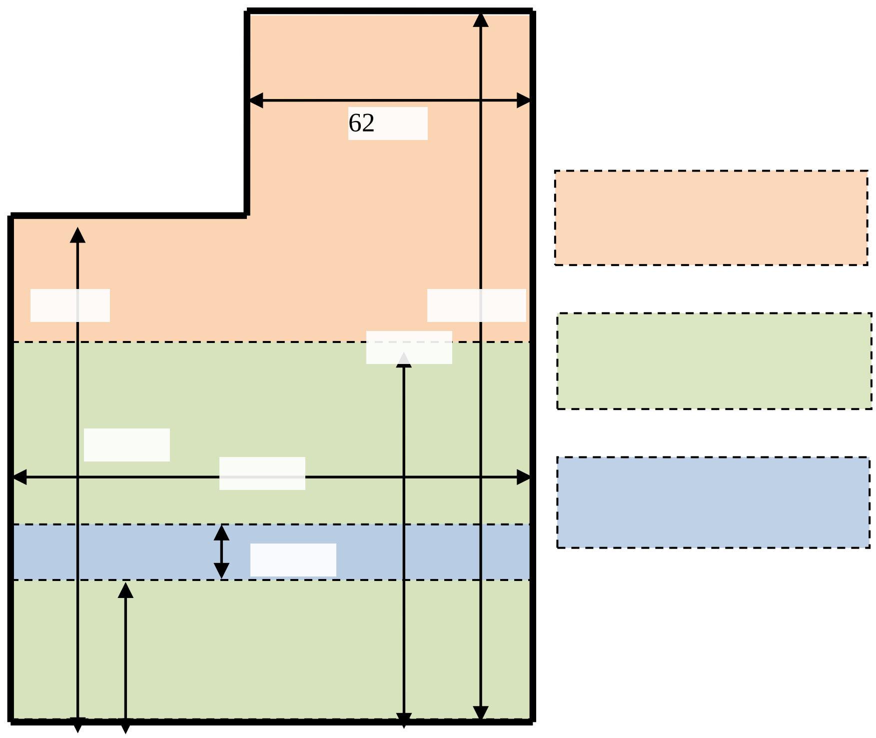


Figure 6-6: Mechanical Details – Top view, Maximum Board Dimensions

Connectors and Electrical Requirements

P8-P11are High Speed Terminal Strips manufactured by Samtec. They have a middle bar that is used for ground and power connections. The part number for P8 is QTE-060-02-L-D-A and the part number for P9-P11 is QTE-040-02-L-D-A.

P8 and P9 are used to connect the expansion board to the GX3500's FPGA Flex IO signals. Figure 6-7 shows a schematic diagram of P8 and P9. Table 6-1 lists the pin assignments for P8 and Table 6-2 lists the pin assignments for P9.

Pull downs may be required on Flex IO lines to set low logic levels during FPGA configuration.

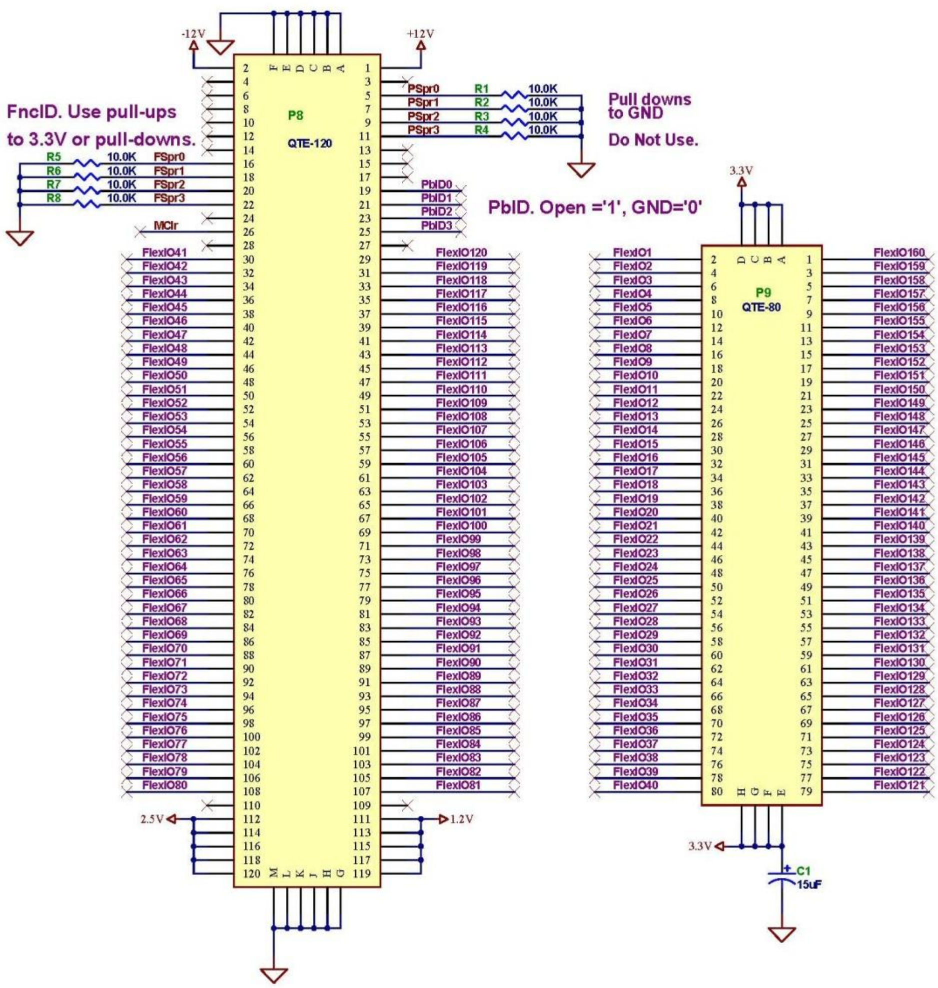


Figure 6-7: GX3500 Expansion Board – Host Connectors

P8 and P9 Host Connectors

The following table describes the GX3500 expansion board P8 and P9 pin assignments:

Pin #	Name	Function	FPGA Pin#	Notes	Pin #	Name	Function	FPGA Pin#	Notes
1	+12V	Power		1	2	-12V	Power		1
3	NC	Not Connected			4	NC	Not Connected		
5	PSpr0	Do Not Use		2	6	NC	Not Connected		
7	PSpr1	Do Not Use		2	8	NC	Not Connected		
9	PSpr2	Do Not Use		2	10	NC	Not Connected		
11	PSpr3	Do Not Use		2	12	NC	Not Connected		
13	NC	Not Connected			14	NC	Not Connected		
15	NC	Not Connected			16	FSpr0	Out, Spare	G9	4,6
17	NC	Not Connected			18	FSpr1	Out, Spare	F7	4,6
19	PbID0	Out, Piggy Back ID		3	20	FSpr2	Out, Spare	F9	4,6
21	PbID1	Out, Piggy Back ID		3	22	FSpr3	Out, Spare	G7	4,6
23	PbID2	Out, Piggy Back ID		3	24	NC	Not Connected		
25	PbID3	Out, Piggy Back ID		3	26	MClr	In, Master Clear		5
27	NC	Not Connected			28	NC	Not Connected		
29	FlexIO120	3.3V LVTTL IO	G8	6,7	30	FlexIO41	3.3V LVTTL IO	F14	6,7
31	FlexIO119	3.3V LVTTL IO	F8	6,7	32	FlexIO42	3.3V LVTTL IO	G11	6,7
33	FlexIO118	3.3V LVTTL IO	F10	6,7	34	FlexIO43	3.3V LVTTL IO	G14	6,7
35	FlexIO117	3.3V LVTTL IO	G10	6,7	36	FlexIO44	3.3V LVTTL IO	F13	6,7
37	FlexIO116	3.3V LVTTL IO	G13	6,7	38	FlexIO45	3.3V LVTTL IO	G16	6,7
39	FlexIO115	3.3V LVTTL IO	F11	6,7	40	FlexIO46	3.3V LVTTL IO	G15	6,7
41	FlexIO114	3.3V LVTTL IO	H16	6,7	42	FlexIO47	3.3V LVTTL IO	L6	6,7
43	FlexIO113	3.3V LVTTL IO	H17	6,7	44	FlexIO48	3.3V LVTTL IO	J17	6,7
45	FlexIO112	3.3V LVTTL IO	N17	6,7	46	FlexIO49	3.3V LVTTL IO	M6	6,7
47	FlexIO111	3.3V LVTTL IO	M16	6,7	48	FlexIO50	3.3V LVTTL IO	W20	6,7
49	FlexIO110	3.3V LVTTL IO	R18	6,7	50	FlexIO51	3.3V LVTTL IO	N7	6,7
51	FlexIO109	3.3V LVTTL IO	N16	6,7	52	FlexIO52	3.3V LVTTL IO	U19	6,7
53	FlexIO108	3.3V LVTTL IO	P20	6,7	54	FlexIO53	3.3V LVTTL IO	P7	6,7
55	FlexIO107	3.3V LVTTL IO	P17	6,7	56	FlexIO54	3.3V LVTTL IO	W19	6,7
57	FlexIO106	3.3V LVTTL IO	R17	6,7	58	FlexIO55	3.3V LVTTL IO	M5	6,7
59	FlexIO105	3.3V LVTTL IO	R19	6,7	60	FlexIO56	3.3V LVTTL IO	T18	6,7
61	FlexIO104	3.3V LVTTL IO	R20	6,7	62	FlexIO57	3.3V LVTTL IO	Y17	6,7
63	FlexIO103	3.3V LVTTL IO	T19	6,7	64	FlexIO58	3.3V LVTTL IO	W17	6,7
65	FlexIO102	3.3V LVTTL IO	T17	6,7	66	FlexIO59	3.3V LVTTL IO	U17	6,7
67	FlexIO101	3.3V LVTTL IO	T20	6,7	68	FlexIO60	3.3V LVTTL IO	U16	6,7
69	FlexIO100	3.3V LVTTL IO	N6	6,7	70	FlexIO61	3.3V LVTTL IO	U15	6,7

Pin #	Name	Function	FPGA Pin#	Notes	Pin #	Name	Function	FPGA Pin#	Notes
71	FlexIO99	3.3V LVTTL IO	U20	6,7	72	FlexIO62	3.3V LVTTL IO	Y15	6,7
73	FlexIO98	3.3V LVTTL IO	P6	6,7	74	FlexIO63	3.3V LVTTL IO	W15	6,7
75	FlexIO97	3.3V LVTTL IO	N18	6,7	76	FlexIO64	3.3V LVTTL IO	V16	6,7
77	FlexIO96	3.3V LVTTL IO	N19	6,7	78	FlexIO65	3.3V LVTTL IO	V15	6,7
79	FlexIO95	3.3V LVTTL IO	N20	6,7	80	FlexIO66	3.3V LVTTL IO	Y14	6,7
81	FlexIO94	3.3V LVTTL IO	J21	6,7	82	FlexIO67	3.3V LVTTL IO	W14	6,7
83	FlexIO93	3.3V LVTTL IO	K21	6,7	84	FlexIO68	3.3V LVTTL IO	V14	6,7
85	FlexIO92	3.3V LVTTL IO	L21	6,7	86	FlexIO69	3.3V LVTTL IO	U14	6,7
87	FlexIO91	3.3V LVTTL IO	M22	6,7	88	FlexIO70	3.3V LVTTL IO	Y13	6,7
89	FlexIO90	3.3V LVTTL IO	M21	6,7	90	FlexIO71	3.3V LVTTL IO	W13	6,7
91	FlexIO89	3.3V LVTTL IO	N22	6,7	92	FlexIO72	3.3V LVTTL IO	V13	6,7
93	FlexIO88	3.3V LVTTL IO	P22	6,7	94	FlexIO73	3.3V LVTTL IO	U13	6,7
95	FlexIO87	3.3V LVTTL IO	P21	6,7	96	FlexIO74	3.3V LVTTL IO	V12	6,7
97	FlexIO86	3.3V LVTTL IO	R22	6,7	98	FlexIO75	3.3V LVTTL IO	U12	6,7
99	FlexIO85	3.3V LVTTL IO	R21	6,7	100	FlexIO76	3.3V LVTTL IO	V11	6,7
101	FlexIO84	3.3V LVTTL IO	U22	6,7	102	FlexIO77	3.3V LVTTL IO	U11	6,7
103	FlexIO83	3.3V LVTTL IO	U21	6,7	104	FlexIO78	3.3V LVTTL IO	Y10	6,7
105	FlexIO82	3.3V LVTTL IO	V22	6,7	106	FlexIO79	3.3V LVTTL IO	W10	6,7
107	FlexIO81	3.3V LVTTL IO	V21	6,7	108	FlexIO80	3.3V LVTTL IO	V10	6,7
109	NC	Not Connected			110	NC	Not Connected		
111	1.2V	Power		8	112	2.5V	Power		9
113	1.2V	Power		8	114	2.5V	Power		9
115	1.2V	Power		8	116	2.5V	Power		9
117	1.2V	Power		8	118	2.5V	Power		9
119	1.2V	Power		8	120	2.5V	Power		9
A	GND	Power			B	GND	Power		
C	GND	Power			D	GND	Power		
E	GND	Power			F	GND	Power		
G	GND	Power			H	GND	Power		
J	GND	Power			K	GND	Power		
L	GND	Power			M	GND	Power		

Table 6-1: P8 Connector pin assignments

The following table describes the GX3500 expansion board P9 pin assignments:

Pin #	Name	Function	FPGA Pin#	Notes	Pin #	Name	Function	FPGA Pin#	Notes
1	FlexIO160	3.3V LVTTL IO	Y22	6,7	2	FlexIO1	3.3V LVTTL IO	W22	6,7
3	FlexIO159	3.3V LVTTL IO	Y21	6,7	4	FlexIO2	3.3V LVTTL IO	W21	6,7
5	FlexIO158	3.3V LVTTL IO	AA22	6,7	6	FlexIO3	3.3V LVTTL IO	V9	6,7
7	FlexIO157	3.3V LVTTL IO	AA21	6,7	8	FlexIO4	3.3V LVTTL IO	V8	6,7
9	FlexIO156	3.3V LVTTL IO	AB20	6,7	10	FlexIO5	3.3V LVTTL IO	W8	6,7
11	FlexIO155	3.3V LVTTL IO	AA20	6,7	12	FlexIO6	3.3V LVTTL IO	Y8	6,7
13	FlexIO154	3.3V LVTTL IO	AB19	6,7	14	FlexIO7	3.3V LVTTL IO	V7	6,7
15	FlexIO153	3.3V LVTTL IO	AA19	6,7	16	FlexIO8	3.3V LVTTL IO	W7	6,7
17	FlexIO152	3.3V LVTTL IO	AB18	6,7	18	FlexIO9	3.3V LVTTL IO	Y7	6,7
19	FlexIO151	3.3V LVTTL IO	AA18	6,7	20	FlexIO10	3.3V LVTTL IO	V6	6,7
21	FlexIO150	3.3V LVTTL IO	AB17	6,7	22	FlexIO11	3.3V LVTTL IO	W6	6,7
23	FlexIO149	3.3V LVTTL IO	AA17	6,7	24	FlexIO12	3.3V LVTTL IO	Y6	6,7
25	FlexIO148	3.3V LVTTL IO	AB16	6,7	26	FlexIO13	3.3V LVTTL IO	V5	6,7
27	FlexIO147	3.3V LVTTL IO	AA16	6,7	28	FlexIO14	3.3V LVTTL IO	Y4	6,7
29	FlexIO146	3.3V LVTTL IO	AB15	6,7	30	FlexIO15	3.3V LVTTL IO	Y3	6,7
31	FlexIO145	3.3V LVTTL IO	AA15	6,7	32	FlexIO16	3.3V LVTTL IO	V4	6,7
33	FlexIO144	3.3V LVTTL IO	AB14	6,7	34	FlexIO17	3.3V LVTTL IO	V3	6,7
35	FlexIO143	3.3V LVTTL IO	AA14	6,7	36	FlexIO18	3.3V LVTTL IO	U2	6,7
37	FlexIO142	3.3V LVTTL IO	AB13	6,7	38	FlexIO19	3.3V LVTTL IO	U1	6,7
39	FlexIO141	3.3V LVTTL IO	AA13	6,7	40	FlexIO20	3.3V LVTTL IO	T3	6,7
41	FlexIO140	3.3V LVTTL IO	AA10	6,7	42	FlexIO21	3.3V LVTTL IO	T4	6,7
43	FlexIO139	3.3V LVTTL IO	AB10	6,7	44	FlexIO22	3.3V LVTTL IO	T5	6,7
45	FlexIO138	3.3V LVTTL IO	AA9	6,7	46	FlexIO23	3.3V LVTTL IO	R4	6,7
47	FlexIO137	3.3V LVTTL IO	AB9	6,7	48	FlexIO24	3.3V LVTTL IO	R3	6,7
49	FlexIO136	3.3V LVTTL IO	AA8	6,7	50	FlexIO25	3.3V LVTTL IO	R2	6,7
51	FlexIO135	3.3V LVTTL IO	AB8	6,7	52	FlexIO26	3.3V LVTTL IO	R1	6,7
53	FlexIO134	3.3V LVTTL IO	AA7	6,7	54	FlexIO27	3.3V LVTTL IO	P4	6,7
55	FlexIO133	3.3V LVTTL IO	AB7	6,7	56	FlexIO28	3.3V LVTTL IO	P3	6,7
57	FlexIO132	3.3V LVTTL IO	AA6	6,7	58	FlexIO29	3.3V LVTTL IO	P2	6,7
59	FlexIO131	3.3V LVTTL IO	AB6	6,7	60	FlexIO30	3.3V LVTTL IO	P1	6,7
61	FlexIO130	3.3V LVTTL IO	AA5	6,7	62	FlexIO31	3.3V LVTTL IO	N2	6,7
63	FlexIO129	3.3V LVTTL IO	AB5	6,7	64	FlexIO32	3.3V LVTTL IO	N1	6,7

Pin #	Name	Function	FPGA Pin#	Notes	Pin #	Name	Function	FPGA Pin#	Notes
65	FlexIO128	3.3V LVTTL IO	AA4	6,7	66	FlexIO33	3.3V LVTTL IO	M4	6,7
67	FlexIO127	3.3V LVTTL IO	AB4	6,7	68	FlexIO34	3.3V LVTTL IO	M3	6,7
69	FlexIO126	3.3V LVTTL IO	AA3	6,7	70	FlexIO35	3.3V LVTTL IO	M2	6,7
71	FlexIO125	3.3V LVTTL IO	AB3	6,7	72	FlexIO36	3.3V LVTTL IO	M1	6,7
73	FlexIO124	3.3V LVTTL IO	AA2	6,7	74	FlexIO37	3.3V LVTTL IO	W2	6,7
75	FlexIO123	3.3V LVTTL IO	AA1	6,7	76	FlexIO38	3.3V LVTTL IO	W1	6,7
77	FlexIO122	3.3V LVTTL IO	Y2	6,7	78	FlexIO39	3.3V LVTTL IO	V2	6,7
79	FlexIO121	3.3V LVTTL IO	Y1	6,7	80	FlexIO40	3.3V LVTTL IO	V1	6,7
A	3.3V	Power		10	B	3.3V	Power		10
C	3.3V	Power		10	D	3.3V	Power		10
E	3.3V	Power		10	F	3.3V	Power		10
G	3.3V	Power		10	H	3.3V	Power		10

Table 6-2: P9 Connector pin assignments

Notes for Host connectors:

1. Maximum 0.5A. May be limited by PXI chassis. Connect a 10uF-22uF capacitor if using these pins.
2. PSpr[3..0] are reserved. Should be connected to ground using 1K-50K resistors.
3. PbID[3..0] are used to identify the expansion board. Leave pins unconnected for logic ‘1’ or connect to ground for logic ‘0’. The GX3500 software driver can read these pins to identify the specific expansion board installed on the GX3500.
4. FSpr[3..0] are spare pins connected to the user FPGA. Should be connected to ground or 3.3V using 1K-50K resistors if not used in the design. Can also be used as an additional identification field.
5. MClr is a Master Clear input to the Expansion board. It is active low and is asserted by the controller at power-up or by a software command at any time.
6. These signals must never drive more than 3.3V. If 5V logic is used in the Expansion board design, these pin must be protected.
7. During the user FPGA configuration phase, these pins have a weak pull-up that may cause an un-intentional condition in the Expansion board design. Pull-down resistors should be used where necessary.
8. Connect these pins together. Maximum 1A for the 1.2V rail. Connect a 10uF-22uF capacitor if using these pins.
9. Connect these pins together. Maximum 0.5A for the 2.5V rail. Connect a 10uF-22uF capacitor if using these pins.
10. Connect these pins together. Maximum 4A for the 3.3V rail. May be limited by PXI chassis. Connect a 10uF-22uF capacitor if using these pins.

P10 and P11 Connectors

Connectors P10 and P11 are used to connect the GX3500's front panel VHDCI connectors to the expansion board. Figure 5 shows the schematic diagram of P10 and P11. Table 6-3 lists the pin assignments for P10 and Table 6-4 lists the pin assignments for P11. P10 connects user I/O signal groups B and C to J3 and J4 on the GX3500 front panel and P11 connects groups A and D to J1 and J2.

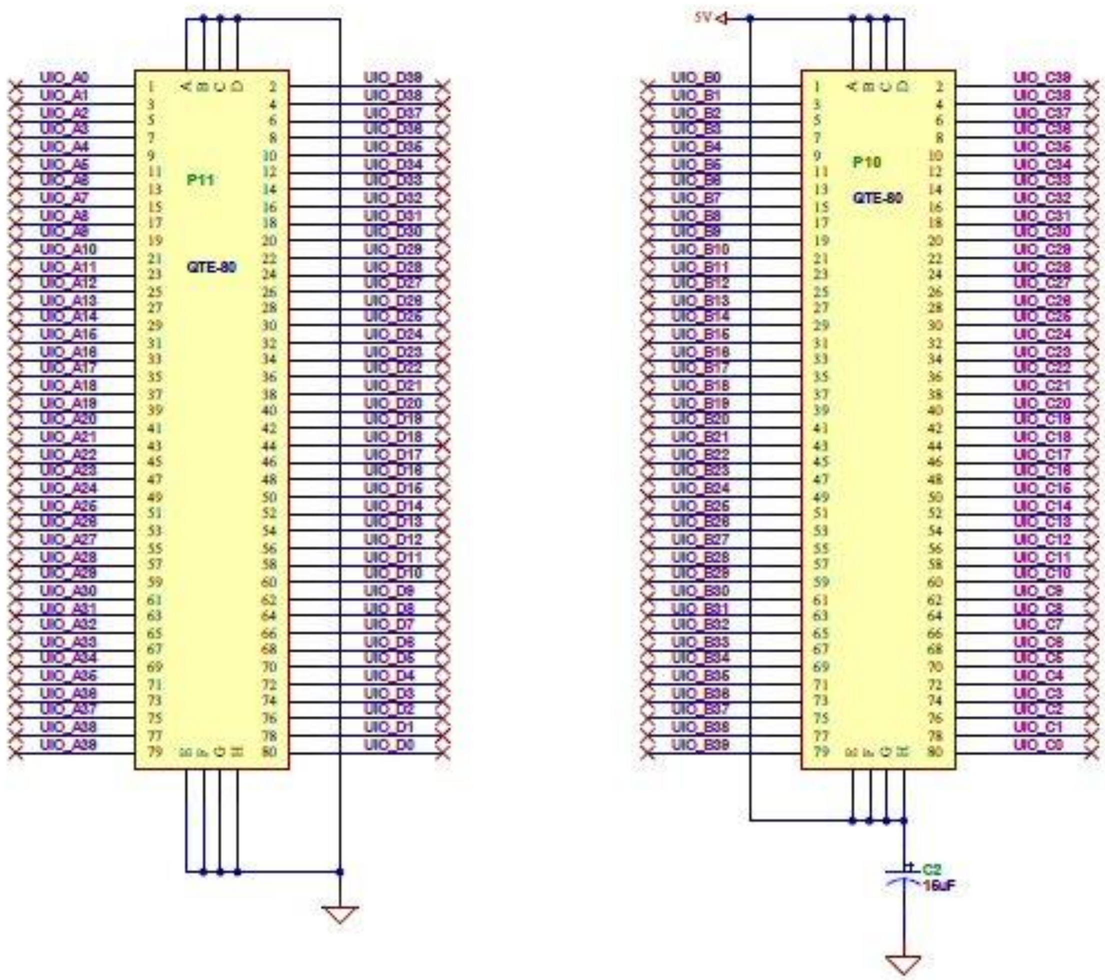


Figure 6-8: GX3500 Expansion Board – User Connectors

P10 Connector

The following table describes the GX3500 expansion board P10 pin assignments:

Pin #	Name	Front Panel Connection	Pin #	Name	Front Panel Connection
1	UIO_B0	J3-35	2	UIO_C39	J4-68
3	UIO_B1	J3-36	4	UIO_C38	J4-34
5	UIO_B2	J3-37	6	UIO_C37	J4-67
7	UIO_B3	J3-38	8	UIO_C36	J4-33
9	UIO_B4	J3-39	10	UIO_C35	J4-66
11	UIO_B5	J3-40	12	UIO_C34	J4-32
13	UIO_B6	J3-41	14	UIO_C33	J4-65
15	UIO_B7	J3-42	16	UIO_C32	J4-31
17	UIO_B8	J3-43	18	UIO_C31	J4-64
19	UIO_B9	J3-44	20	UIO_C30	J4-30
21	UIO_B10	J3-45	22	UIO_C29	J4-63
23	UIO_B11	J3-46	24	UIO_C28	J4-29
25	UIO_B12	J3-47	26	UIO_C27	J4-62
27	UIO_B13	J3-48	28	UIO_C26	J4-61
29	UIO_B14	J3-49	30	UIO_C25	J4-60
31	UIO_B15	J3-50	32	UIO_C24	J4-59
33	UIO_B16	J3-51	34	UIO_C23	J4-58
35	UIO_B17	J3-52	36	UIO_C22	J4-57
37	UIO_B18	J3-53	38	UIO_C21	J4-56
39	UIO_B19	J3-54	40	UIO_C20	J4-55
41	UIO_B20	J3-55	42	UIO_C19	J4-54
43	UIO_B21	J3-56	44	UIO_C18	J4-53
45	UIO_B22	J3-57	46	UIO_C17	J4-52
47	UIO_B23	J3-58	48	UIO_C16	J4-51
49	UIO_B24	J3-59	50	UIO_C15	J4-50
51	UIO_B25	J3-60	52	UIO_C14	J4-49
53	UIO_B26	J3-61	54	UIO_C13	J4-48
55	UIO_B27	J3-62	56	UIO_C12	J4-47
57	UIO_B28	J3-29	58	UIO_C11	J4-46
59	UIO_B29	J3-63	60	UIO_C10	J4-45
61	UIO_B30	J3-30	62	UIO_C9	J4-44
63	UIO_B31	J3-64	64	UIO_C8	J4-43
65	UIO_B32	J3-31	66	UIO_C7	J4-42
67	UIO_B33	J3-65	68	UIO_C6	J4-41
69	UIO_B34	J3-32	70	UIO_C5	J4-40

Pin #	Name	Front Panel Connection	Pin #	Name	Front Panel Connection
71	UIO_B35	J3-66	72	UIO_C4	J4-39
73	UIO_B36	J3-33	74	UIO_C3	J4-38
75	UIO_B37	J3-67	76	UIO_C2	J4-37
77	UIO_B38	J3-34	78	UIO_C1	J4-36
79	UIO_B39	J3-68	80	UIO_C0	J4-35
A	5V	Power*	B	5V	Power*
C	5V	Power*	D	5V	Power*
E	5V	Power*	F	5V	Power*
G	5V	Power*	H	5V	Power*

Table 6-3: P10 Connector pin assignments

*Connect these pins together. Maximum 4A for the 5V rail. May be limited by PXI chassis. Connect a 10uF-22uF capacitor if using these pins.

P11 Connector

The following table describes the GX3500 expansion board P11 pin assignments:

Pin #	Name	Front Panel Connection	Pin #	Name	Front Panel Connection
1	UIO_A0	J1-35	2	UIO_D39	J2-68
3	UIO_A1	J1-36	4	UIO_D38	J2-34
5	UIO_A2	J1-37	6	UIO_D37	J2-67
7	UIO_A3	J1-38	8	UIO_D36	J2-33
9	UIO_A4	J1-39	10	UIO_D35	J2-66
11	UIO_A5	J1-40	12	UIO_D34	J2-32
13	UIO_A6	J1-41	14	UIO_D33	J2-65
15	UIO_A7	J1-42	16	UIO_D32	J2-31
17	UIO_A8	J1-43	18	UIO_D31	J2-64
19	UIO_A9	J1-44	20	UIO_D30	J2-30
21	UIO_A10	J1-45	22	UIO_D29	J2-63
23	UIO_A11	J1-46	24	UIO_D28	J2-29
25	UIO_A12	J1-47	26	UIO_D27	J2-62
27	UIO_A13	J1-48	28	UIO_D26	J2-61
29	UIO_A14	J1-49	30	UIO_D25	J2-60
31	UIO_A15	J1-50	32	UIO_D24	J2-59
33	UIO_A16	J1-51	34	UIO_D23	J2-58
35	UIO_A17	J1-52	36	UIO_D22	J2-57
37	UIO_A18	J1-53	38	UIO_D21	J2-56
39	UIO_A19	J1-54	40	UIO_D20	J2-55
41	UIO_A20	J1-55	42	UIO_D19	J2-54
43	UIO_A21	J1-56	44	UIO_D18	J2-53
45	UIO_A22	J1-57	46	UIO_D17	J2-52
47	UIO_A23	J1-58	48	UIO_D16	J2-51
49	UIO_A24	J1-59	50	UIO_D15	J2-50
51	UIO_A25	J1-60	52	UIO_D14	J2-49
53	UIO_A26	J1-61	54	UIO_D13	J2-48
55	UIO_A27	J1-62	56	UIO_D12	J2-47
57	UIO_A28	J1-29	58	UIO_D11	J2-46
59	UIO_A29	J1-63	60	UIO_D10	J2-45
61	UIO_A30	J1-30	62	UIO_D9	J2-44
63	UIO_A31	J1-64	64	UIO_D8	J2-43
65	UIO_A32	J1-31	66	UIO_D7	J2-42
67	UIO_A33	J1-65	68	UIO_D6	J2-41
69	UIO_A34	J1-32	70	UIO_D5	J2-40
71	UIO_A35	J1-66	72	UIO_D4	J2-39

Pin #	Name	Front Panel Connection	Pin #	Name	Front Panel Connection
73	UIO_A36	J1-33	74	UIO_D3	J2-38
75	UIO_A37	J1-67	76	UIO_D2	J2-37
77	UIO_A38	J1-34	78	UIO_D1	J2-36
79	UIO_A39	J1-68	80	UIO_D0	J2-35
A	GND	Power	B	GND	Power
C	GND	Power	D	GND	Power
E	GND	Power	F	GND	Power
G	GND	Power	H	GND	Power

Table 6-4: P11 Connector Pin Assignments

Programming Support for Expansion Boards

The following diagram shows the connectors associated with the expansion board (P8 to P11) and the bi-directional switches used to connect the Flex I/O signals directly to the front panel connectors depending whether jumpers J3 to J6 are installed or not.

An option the user has when designing his application is the ability to select in groups of 40 I/O channels which signals will go to the daughter board and which signals will go directly to the front panel connectors.

By selectively connecting JP3, JP4, JP5, and JP6 the user can route I/O channels to the front panel connectors or the daughter card in groups of 40 I/O channels depending on the needs of the application.

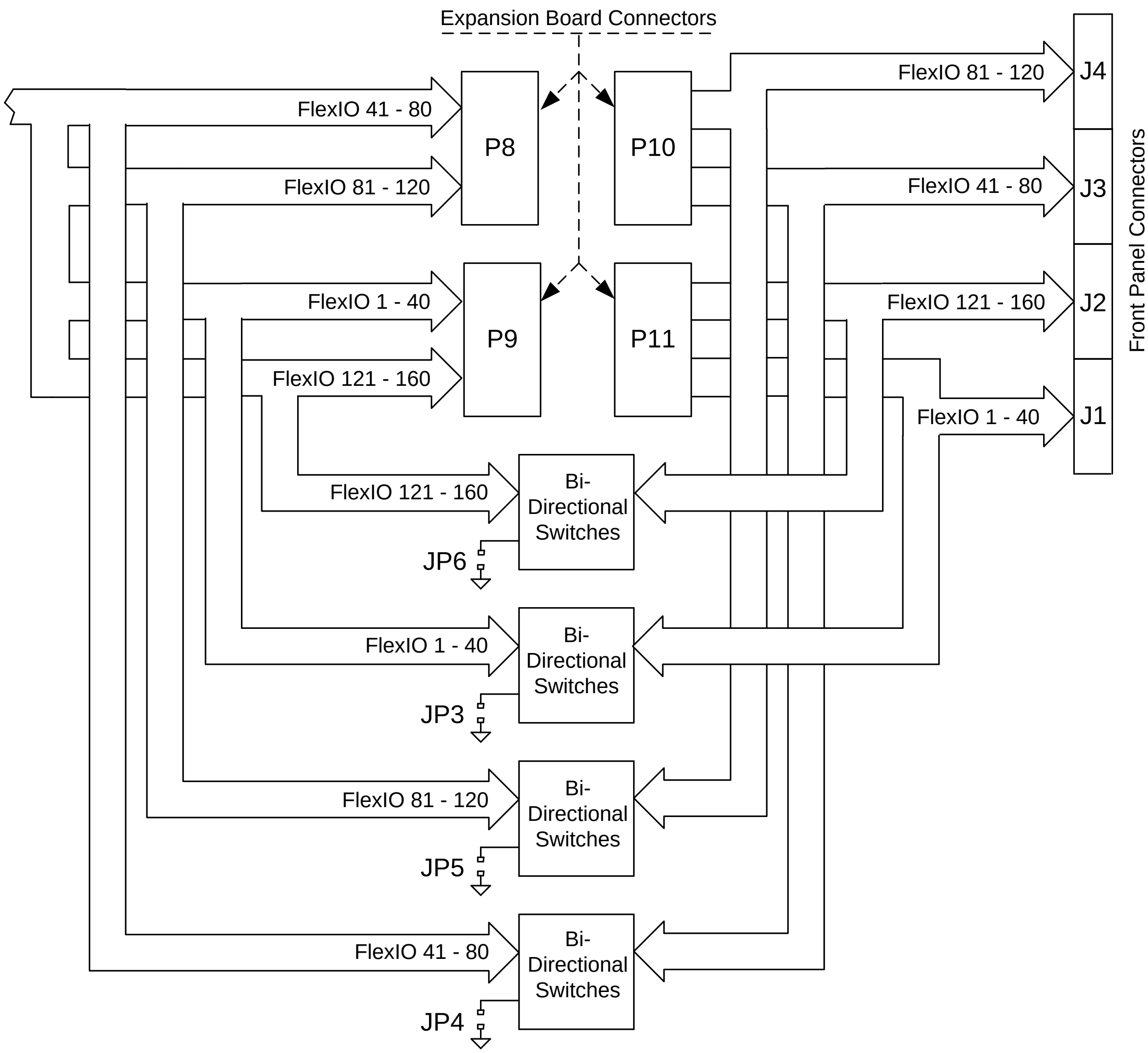


Figure 6-9: GX3500 Expansion Board Connectors and Bi-Directional Switches

Block Diagram

The bi-directional switches, when enabled by connecting their respective jumpers, will be turned on and the corresponding signals between the Flex IO and the front panel connectors will be permanently connected, these are pass-through switches and there is no direction control signal. When developing a design, the user needs to be aware that signals that are either inputs or outputs only are defined as inputs or outputs on the FPGA, these signals can be left driving or enabled all the time; however, signals that are bi-directional like bus signals need to be defined as bi-directional on the FPGA but make sure to drive the output to high impedance or tristate level when the signal is not driving or is inactive. This will prevent signal contention if two signals connected to the same switch from opposite directions are turned on or enabled at the same time.

Programming expansion board is done using the GXFPGA registers and memory access functions. In addition, a **GxFpgaSetExpansionBoardBypass** can be used to direct the I/O banks to the expansion board. The **GxFpgaGetExpansionBoardID** can also be used to detect the installed expansion board ID. The expansion board ID is read from P8 pins 19, 21, 23 and 25 to form a 4-bit integer (0-15).

GX3501, GX3509, GX3510 and GX3540 (PIO) Expansion Boards

The GX3500 is available with a family of expansion modules used to support different voltage levels:

- **GX3501** - 80 Channel TLL Buffer Expansion Card for GX3500. Each group of 40 channels can be configured with an on-board jumper to support TTL of LVTTL logic levels. Each channel can be configured to an input or output under software control. Together with the GX3500 the combined card is called **GX3601**.
- **GX3509** - 80 Channel Differential TLL Expansion Board for GX3500. Each channel can be configured as an output or input under software control via the GX3500's function calls. Together with the GX3500 the combined card is called **GX3609**.
- **GX3510** - 80 Channel M-LVDS Buffer Expansion Board for GX3500. Together with the GX3500 the combined card is called **GX3610**.
- **GX3540** - ECL Expansion Board Expansion Board for GX3500, this board when combined with the GX3500 is also known as the **GX3640**. The GX3540 consists of 20 differential ECL drivers and 20 differential ECL receivers. Each channel can be accessed via software commands for use in static I/O applications. The board includes on-board terminators for each differential channel.

Internal Connections

The following table describes the channel assignment and connections for the GX3501, GX3509, GX3510 and GX3540 expansion boards. Differential signals are preceded by Hi for positive signals by Lo for negative signals, in turn they are followed by the User Signal Name or the connector pin in the User Connector or the Front Panel Connection. The GX3501 uses only the Hi signals from the differential pair when connecting to the user side.

The GX3540 uses user signals UIO_Axx (See Table 6-5: P11 Connector Pin Assignments) for the input channels and user signal UIO_Dxx for the output channels. User signals UIO Bxx (See Table 6-3: P10 Connector pin assignments) and UIO_Cxx are not used on the GX3540. The channels with the I/O Direction function are also not used.

The following image show differential channel 1 and 2. DfE1 (Flex I/O Channel 1) is I/O Data for channel 1 and Diff1 (Flex I/O Channel 2) is the I/O Direction control for channel 1, see Table 6-6: P11 Connector Pin Assignments.

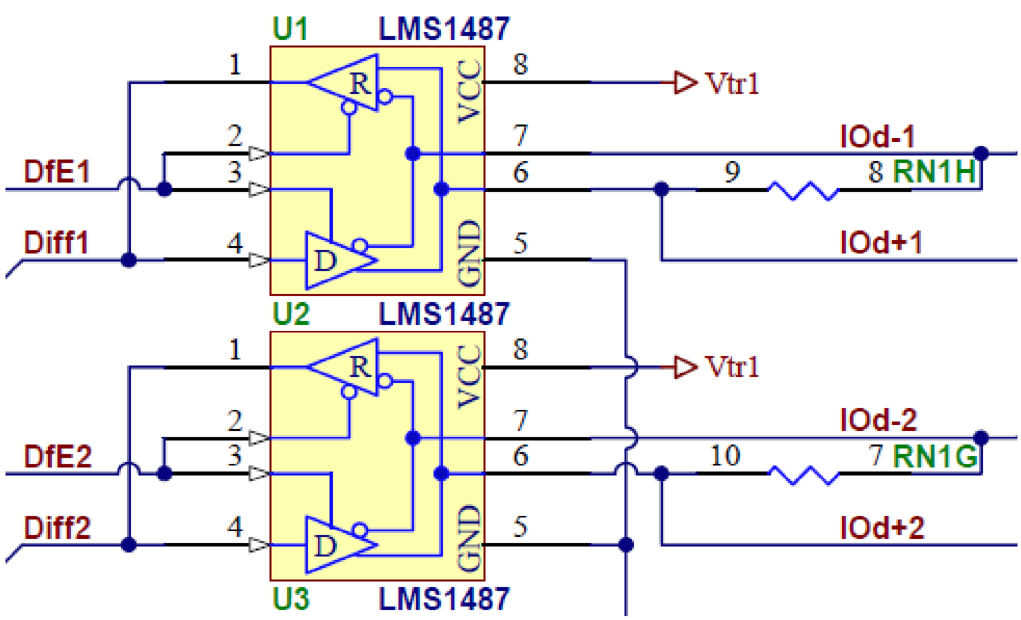


Figure 6-10: Differential Channel Schematic

Flex I/O Channel	**Host Connector Pin**	**Related User Channel**	**Function**	**Related User Signal Name**	**User Connector Pin**	**Front Panel Connection**
1	P9-2	1	I/O Data	Hi A0, Lo A1	Hi P11-1, Lo P11-3	Hi J1-35, Lo J1-36
2	P9-4	1	I/O Direction			
3	P9-6	2	I/O Data	Hi A2, Lo A3	Hi P11-5, Lo P11-7	Hi J1-37, Lo J1-38
4	P9-8	2	I/O Direction			
5	P9-10	3	I/O Data	Hi A4, Lo A5	Hi P11-9, Lo P11-11	Hi J1-39, Lo J1-40
6	P9-12	3	I/O Direction			
7	P9-14	4	I/O Data	Hi A6, Lo A7	Hi P11-13, Lo P11-15	Hi J1-41, Lo J1-42
8	P9-16	4	I/O Direction			
9	P9-18	5	I/O Data	Hi A8, Lo A9	Hi P11-17, Lo P11-19	Hi J1-43, Lo J1-44
10	P9-20	5	I/O Direction			
11	P9-22	6	I/O Data	Hi A10, Lo A11	Hi P11-21, Lo P11-23	Hi J1-45, Lo J1-46
12	P9-24	6	I/O Direction			
13	P9-26	7	I/O Data	Hi A12, Lo A13	Hi P11-25, Lo P11-27	Hi J1-47, Lo J1-48
14	P9-28	7	I/O Direction			
15	P9-30	8	I/O Data	Hi A14, Lo A15	Hi P11-29, Lo P11-31	Hi J1-49, Lo J1-50
16	P9-32	8	I/O Direction			
17	P9-34	9	I/O Data	Hi A16, Lo A17	Hi P11-33, Lo P11-35	Hi J1-51, Lo J1-52
18	P9-36	9	I/O Direction			
19	P9-38	10	I/O Data	Hi A18, Lo A19	Hi P11-37, Lo P11-39	Hi J1-53, Lo J1-54
20	P9-40	10	I/O Data			
21	P9-42	11	I/O Data	Hi A20, Lo A21	Hi P11-41, Lo P11-43	Hi J1-55, Lo J1-56
22	P9-44	11	I/O Direction			
23	P9-46	12	I/O Data	Hi A22, Lo A23	Hi P11-45, Lo P11-47	Hi J1-57, Lo J1-58
24	P9-48	12	I/O Direction			
25	P9-50	13	I/O Data	Hi A24, Lo A25	Hi P11-49, Lo P11-51	Hi J1-59, Lo J1-60
26	P9-52	13	I/O Direction			

Flex I/O Channel	Host Connector Pin	Related User Channel	Function	Related User Signal Name	User Connector Pin	Front Panel Connection
27	P9-54	14	I/O Data	Hi A26, Lo A27	Hi P11-53, Lo P11-55	Hi J1-61, Lo J1-62
28	P9-56	14	I/O Direction			
29	P9-58	15	I/O Data	Hi A28, Lo A29	Hi P11-57, Lo P11-59	Hi J1-29, Lo J1-63
30	P9-60	15	I/O Direction			
31	P9-62	16	I/O Data	Hi A30, Lo A31	Hi P11-61, Lo P11-63	Hi J1-30, Lo J1-64
32	P9-64	16	I/O Direction			
33	P9-66	17	I/O Data	Hi A32, Lo A33	Hi P11-65, Lo P11-67	Hi J1-31, Lo J1-65
34	P9-68	17	I/O Direction			
35	P9-70	18	I/O Data	Hi A34, Lo A35	Hi P11-69, Lo P11-71	Hi J1-32, Lo J1-66
36	P9-72	18	I/O Direction			
37	P9-74	19	I/O Data	Hi A36, Lo A37	Hi P11-73, Lo P11-75	Hi J1-33 Lo J1-67
38	P9-76	19	I/O Direction			
39	P9-78	20	I/O Data	Hi A38, Lo A39	Hi P11-77, Lo P11-79	Hi J1-34, Lo J1-68
40	P9-80	20	I/O Data			
41	P8-30	21	I/O Data	Hi B0, Lo B1	Hi P10-1, Lo P10-3	Hi J3-35, Lo J3-36
42	P8-32	21	I/O Direction			
43	P8-34	22	I/O Data	Hi B2, Lo B3	Hi P10-5, Lo P10-7	Hi J3-37, Lo J3-38
44	P8-36	22	I/O Direction			
45	P8-38	23	I/O Data	Hi B4, Lo B5	Hi P10-9, Lo P10-11	Hi J3-39, Lo J3-40
46	P8-40	23	I/O Direction			
47	P8-42	24	I/O Data	Hi B6, Lo B7	Hi P10-13, Lo P10-15	Hi J3-41, Lo J3-42
48	P8-44	24	I/O Direction			
49	P8-46	25	I/O Data	Hi B8, Lo B9	Hi P10-17, Lo P10-19	Hi J3-43, Lo J3-44
50	P8-48	25	I/O Direction			
51	P8-50	26	I/O Data	Hi B10, Lo B11	Hi P10-21, Lo P10-23	Hi J3-45, Lo J3-46
52	P8-52	26	I/O Direction			

Flex I/O Channel	Host Connector Pin	Related User Channel	Function	Related User Signal Name	User Connector Pin	Front Panel Connection
53	P8-54	27	I/O Data	Hi B12, Lo B13	Hi P10-25, Lo P10-27	Hi J3-47, Lo J3-48
54	P8-56	27	I/O Direction			
55	P8-58	28	I/O Data	Hi B14, Lo B15	Hi P10-29, Lo P10-31	Hi J3-49, Lo J3-50
56	P8-60	28	I/O Direction			
57	P8-62	29	I/O Data	Hi B16, Lo B17	Hi P10-33, Lo P10-35	Hi J3-51, Lo J3-52
58	P8-64	29	I/O Direction			
59	P8-66	30	I/O Data	Hi B18, Lo B19	Hi P10-37, Lo P10-39	Hi J3-53, Lo J3-54
60	P8-68	30	I/O Direction			
61	P8-70	31	I/O Data	Hi B20, Lo B21	Hi P10-41, Lo P10-43	Hi J3-55, Lo J3-56
62	P8-72	31	I/O Direction			
63	P8-74	32	I/O Data	Hi B22, Lo B23	Hi P10-45, Lo P10-47	Hi J3-57, Lo J3-58
64	P8-76	32	I/O Direction			
65	P8-78	33	I/O Data	Hi B24, Lo B25	Hi P10-49, Lo P10-51	Hi J3-59, Lo J3-60
66	P8-80	33	I/O Direction			
67	P8-82	34	I/O Data	Hi B26, Lo B27	Hi P10-53, Lo P10-55	Hi J3-61, Lo J3-62
68	P8-84	34	I/O Direction			
69	P8-86	35	I/O Data	Hi B28, Lo B29	Hi P10-57, Lo P10-59	Hi J3-29, Lo J3-63
70	P8-88	35	I/O Direction			
71	P8-90	36	I/O Data	Hi B30, Lo B31	Hi P10-61, Lo P10-63	Hi J3-30, Lo J3-64
72	P8-92	36	I/O Direction			
73	P8-94	37	I/O Data	Hi B32, Lo B33	Hi P10-65, Lo P10-67	Hi J3-31, Lo J3-65
74	P8-96	37	I/O Direction			
75	P8-98	38	I/O Data	Hi B34, Lo B35	Hi P10-69, Lo P10-71	Hi J3-32, Lo J3-66
76	P8-100	38	I/O Direction			
77	P8-102	39	I/O Data	Hi B36, Lo B37	Hi P10-73, Lo P10-75	Hi J3-33 Lo J3-67
78	P8-104	39	I/O Direction			

Flex I/O Channel	Host Connector Pin	Related User Channel	Function	Related User Signal Name	User Connector Pin	Front Panel Connection
79	P8-106	40	I/O Data	Hi B38, Lo B39	Hi P10-77, Lo P10-79	Hi J3-34, Lo J3-68
80	P8-108	40	I/O Data			
81	P8-107	41	I/O Data	Hi C0, Lo C1	Hi P10-80, Lo P10-78	Hi J4-35, Lo J4-36
82	P8-105	41	I/O Direction			
83	P8-103	42	I/O Data	Hi C2, Lo C3	Hi P10-76, Lo P10-74	Hi J4-37, Lo J4-38
84	P8-101	42	I/O Direction			
85	P8-99	43	I/O Data	Hi C4, Lo C5	Hi P10-72, Lo P10-70	Hi J4-39, Lo J4-40
86	P8-97	43	I/O Direction			
87	P8-95	44	I/O Data	Hi C6, Lo C7	Hi P10-68, Lo P10-66	Hi J4-41, Lo J4-42
88	P8-93	44	I/O Direction			
89	P8-91	45	I/O Data	Hi C8, Lo C9	Hi P10-64, Lo P10-62	Hi J4-43, Lo J4-44
90	P8-89	45	I/O Direction			
91	P8-87	46	I/O Data	Hi C10, Lo C11	Hi P10-60, Lo P10-58	Hi J4-45, Lo J4-46
92	P8-85	46	I/O Direction			
93	P8-83	47	I/O Data	Hi C12, Lo C13	Hi P10-56, Lo P10-54	Hi J4-47, Lo J4-48
94	P8-81	47	I/O Direction			
95	P8-79	48	I/O Data	Hi C14, Lo C15	Hi P10-52, Lo P10-50	Hi J4-49, Lo J4-50
96	P8-77	48	I/O Direction			
97	P8-75	49	I/O Data	Hi C16, Lo C17	Hi P10-48, Lo P10-46	Hi J4-51, Lo J4-52
98	P8-73	49	I/O Direction			
99	P8-71	50	I/O Data	Hi C18, Lo C19	Hi P10-44, Lo P10-42	Hi J4-53, Lo J4-54
100	P8-69	50	I/O Data			
101	P8-67	51	I/O Data	Hi C20, Lo C21	Hi P10-40, Lo P10-38	Hi J4-55, Lo J4-56
102	P8-65	51	I/O Direction			
103	P8-63	52	I/O Data	Hi C22, Lo C23	Hi P10-36, Lo P10-34	Hi J4-57, Lo J4-58
104	P8-61	52	I/O Direction			

Flex I/O Channel	Host Connector Pin	Related User Channel	Function	Related User Signal Name	User Connector Pin	Front Panel Connection
105	P8-59	53	I/O Data	Hi C24, Lo C25	Hi P10-32, Lo P10-30	Hi J4-59, Lo J4-60
106	P8-57	53	I/O Direction			
107	P8-55	54	I/O Data	Hi C26, Lo C27	Hi P10-28, Lo P10-26	Hi J4-61, Lo J4-62
108	P8-53	54	I/O Direction			
109	P8-51	55	I/O Data	Hi C28, Lo C29	Hi P10-24, Lo P10-22	Hi J4-29, Lo J4-63
110	P8-49	55	I/O Direction			
111	P8-47	56	I/O Data	Hi C30, Lo C31	Hi P10-20, Lo P10-18	Hi J4-30, Lo J4-64
112	P8-45	56	I/O Direction			
113	P8-43	57	I/O Data	Hi C32, Lo C33	Hi P10-16, Lo P10-14	Hi J4-31, Lo J4-65
114	P8-41	57	I/O Direction			
115	P8-39	58	I/O Data	Hi C34, Lo C35	Hi P10-12, Lo P10-10	Hi J4-32, Lo J4-66
116	P8-37	58	I/O Direction			
117	P8-35	59	I/O Data	Hi C36, Lo C37	Hi P10-8, Lo P10-6	Hi J4-33 Lo J4-67
118	P8-33	59	I/O Direction			
119	P8-31	60	I/O Data	Hi C38, Lo C39	Hi P10-4, Lo P10-2	Hi J4-34, Lo J4-68
120	P8-29	60	I/O Direction			
121	P9-79	61	I/O Data	Hi D0, Lo D1	Hi P11-80, Lo P11-78	Hi J2-35, Lo J2-36
122	P9-77	61	I/O Direction			
123	P9-75	62	I/O Data	Hi D2, Lo D3	Hi P11-76, Lo P11-74	Hi J2-37, Lo J2-38
124	P9-73	62	I/O Direction			
125	P9-71	63	I/O Data	Hi D4, Lo D5	Hi P11-72, Lo P11-70	Hi J2-39, Lo J2-40
126	P9-69	63	I/O Direction			
127	P9-67	64	I/O Data	Hi D6, Lo D7	Hi P11-68, Lo P11-66	Hi J2-41, Lo J2-42
128	P9-65	64	I/O Direction			
129	P9-63	65	I/O Data	Hi D8, Lo D9	Hi P11-64, Lo P11-62	Hi J2-43, Lo J2-44
130	P9-61	65	I/O Direction			

Flex I/O Channel	Host Connector Pin	Related User Channel	Function	Related User Signal Name	User Connector Pin	Front Panel Connection
131	P9-59	66	I/O Data	Hi D10, Lo D11	Hi P11-60, Lo P11-58	Hi J2-45, Lo J2-46
132	P9-57	66	I/O Direction			
133	P9-55	67	I/O Data	Hi D12, Lo D13	Hi P11-56, Lo P11-54	Hi J2-47, Lo J2-48
134	P9-53	67	I/O Direction			
135	P9-51	68	I/O Data	Hi D14, Lo D15	Hi P11-52, Lo P11-50	Hi J2-49, Lo J2-50
136	P9-49	68	I/O Direction			
137	P9-47	69	I/O Data	Hi D16, Lo D17	Hi P11-48, Lo P11-46	Hi J2-51, Lo J2-52
138	P9-45	69	I/O Direction			
139	P9-43	70	I/O Data	Hi D18, Lo D19	Hi P11-44, Lo P11-42	Hi J2-53, Lo J2-54
140	P9-41	70	I/O Data			
141	P8-39	71	I/O Data	Hi D20, Lo D21	Hi P11-40, Lo P11-38	Hi J2-55, Lo J2-56
142	P9-37	71	I/O Direction			
143	P9-35	72	I/O Data	Hi D22, Lo D23	Hi P11-36, Lo P11-34	Hi J2-57, Lo J2-58
144	P9-33	72	I/O Direction			
145	P9-31	73	I/O Data	Hi D24, Lo D25	Hi P11-32, Lo P11-30	Hi J2-59, Lo J2-60
146	P9-29	73	I/O Direction			
147	P9-27	74	I/O Data	Hi D26, Lo D27	Hi P11-28, Lo P11-26	Hi J2-61, Lo J2-62
148	P9-25	74	I/O Direction			
149	P9-23	75	I/O Data	Hi D28, Lo D29	Hi P11-24, Lo P11-22	Hi J2-29, Lo J2-63
150	P9-21	75	I/O Direction			
151	P9-19	76	I/O Data	Hi D30, Lo D31	Hi P11-20, Lo P11-18	Hi J2-30, Lo J2-64
152	P9-17	76	I/O Direction			
153	P9-15	77	I/O Data	Hi D32, Lo D33	Hi P11-16, Lo P11-14	Hi J2-31, Lo J2-65
154	P9-13	77	I/O Direction			
155	P9-11	78	I/O Data	Hi D34, Lo D35	Hi P11-12, Lo P11-10	Hi J2-32, Lo J2-66
156	P9-9	78	I/O Direction			

Flex I/O Channel	Host Connector Pin	Related User Channel	Function	Related User Signal Name	User Connector Pin	Front Panel Connection
157	P9-7	79	I/O Data	Hi D36, Lo D37	Hi P11-8, Lo P11-6	Hi J2-33 Lo J2-67
158	P9-5	79	I/O Direction			
159	P9-3	80	I/O Data	Hi D38, Lo D39	Hi P11-4, Lo P11-2	Hi J2-34, Lo J2-68
160	P9-1	80	I/O Direction			

Table 6-6: P11 Connector Pin Assignments

GX3601 J1-J4 Banks A-D Connectors

Connections to the GX3501 made with a 68-pin VHDCI male plug connector. Shielded cables with matching connectors are available from Marvin Test Solutions.

GX3601 J1 Flex I/O Bank A Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	I/O 0	52	NC
2	GND	19	GND	36	NC	53	I/O 9
3	GND	20	GND	37	I/O 1	54	NC
4	GND	21	GND	38	NC	55	I/O 10
5	GND	22	GND	39	I/O 2	56	NC
6	GND	23	GND	40	NC	57	I/O 11
7	GND	24	GND	41	I/O 3	58	I NC
8	GND	25	GND	42	NC	59	I/O 12
9	GND	26	GND	43	I/O 4	60	NC
10	GND	27	5 V UUT	44	NC	61	I/O 13
11	GND	28	5 V UUT	45	I/O 5	62	NC
12	GND	29	I/O 14	46	NC	63	NC
13	GND	30	I/O 15	47	I/O 6	64	NC
14	GND	31	I/O 16	48	NC	65	NC
15	GND	32	I/O 17	49	I/O 7	66	NC
16	GND	33	I/O 18	50	NC	67	NC
17	GND	34	I/O 19	51	I/O 8	68	NC

Table 6-7: GX3601 J1 Flex IO Bank A Connector

GX3601 J3 Flex I/O Bank B Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	I/O 20	52	NC
2	GND	19	GND	36	NC	53	I/O 29
3	GND	20	GND	37	I/O 21	54	NC
4	GND	21	GND	38	NC	55	I/O 30
5	GND	22	GND	39	I/O 22	56	NC
6	GND	23	GND	40	NC	57	I/O 31
7	GND	24	GND	41	I/O 23	58	NC
8	GND	25	GND	42	NC	59	I/O 32
9	GND	26	GND	43	I/O 24	60	NC
10	GND	27	5 V UUT	44	NC	61	I/O 33
11	GND	28	5 V UUT	45	I/O 25	62	NC
12	GND	29	I/O 34	46	NC	63	NC
13	GND	30	I/O 35	47	I/O 26	64	NC
14	GND	31	I/O 36	48	NC	65	NC
15	GND	32	I/O 37	49	I/O 27	66	NC
16	GND	33	I/O 38	50	NC	67	NC
17	GND	34	I/O 39	51	I/O 28	68	NC

Table 6-8: GX3601 J3 Flex IO Bank B Connector

GX3601 J4 Flex I/O Bank C Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	I/O 40	52	NC
2	GND	19	GND	36	NC	53	I/O 49
3	GND	20	GND	37	I/O 41	54	NC
4	GND	21	GND	38	NC	55	I/O 50
5	GND	22	GND	39	I/O 42	56	NC
6	GND	23	GND	40	NC	57	I/O 51
7	GND	24	GND	41	I/O 43	58	NC
8	GND	25	GND	42	NC	59	I/O 52
9	GND	26	GND	43	I/O 44	60	NC
10	GND	27	5 V UUT	44	NC	61	I/O 53
11	GND	28	5 V UUT	45	I/O 45	62	NC
12	GND	29	I/O 54	46	NC	63	NC
13	GND	30	I/O 55	47	I/O 46	64	NC
14	GND	31	I/O 56	48	NC	65	NC
15	GND	32	I/O 57	49	I/O 47	66	NC
16	GND	33	I/O 58	50	NC	67	NC
17	GND	34	I/O 59	51	I/O 48	68	NC

Table 6-9: GX3601 J4 Flex IO Bank C Connector

GX3601 J2 Flex I/O Bank D Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	I/O 60	52	NC
2	GND	19	GND	36	NC	53	I/O 69
3	GND	20	GND	37	I/O 61	54	NC
4	GND	21	GND	38	NC	55	I/O 70
5	GND	22	GND	39	I/O 62	56	NC
6	GND	23	GND	40	NC	57	I/O 71
7	GND	24	GND	41	I/O 63	58	NC
8	GND	25	GND	42	NC	59	I/O 72
9	GND	26	GND	43	I/O 64	60	NC
10	GND	27	5 V UUT	44	NC	61	I/O 73
11	GND	28	5 V UUT	45	I/O 65	62	NC
12	GND	29	I/O 74	46	NC	63	NC
13	GND	30	I/O 75	47	I/O 66	64	NC
14	GND	31	I/O 76	48	NC	65	NC
15	GND	32	I/O 77	49	I/O 67	66	NC
16	GND	33	I/O 78	50	NC	67	NC
17	GND	34	I/O 79	51	I/O 68	68	NC

Table 6-10: GX3601 J2 Flex IO Bank D Connector

GX3609 and GX3610 J1-J4 Banks A-D Connectors

Connections to the GX3509 and GX510 are made with a 68-pin VHDCI male plug connector. Shielded cables with matching connectors are available from Marvin Test Solutions.

GX3609, GX3610 J1 Flex I/O Bank A Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	I/O 0+	52	I/O 8-
2	GND	19	GND	36	I/O 0-	53	I/O 9+
3	GND	20	GND	37	I/O 1+	54	I/O 9-
4	GND	21	GND	38	I/O 1-	55	I/O 10+
5	GND	22	GND	39	I/O 2+	56	I/O 10-
6	GND	23	GND	40	I/O 2-	57	I/O 11+
7	GND	24	GND	41	I/O 3+	58	I/O 11-
8	GND	25	GND	42	I/O 3-	59	I/O 12+
9	GND	26	GND	43	I/O 4+	60	I/O 12-
10	GND	27	5 V UUT	44	I/O 4 -	61	I/O 13+
11	GND	28	5 V UUT	45	I/O 5+	62	I/O 13-
12	GND	29	I/O 14+	46	I/O 5-	63	I/O 14 -
13	GND	30	I/O 15+	47	I/O 6+	64	I/O 15-
14	GND	31	I/O 16+	48	I/O 6-	65	I/O 16-
15	GND	32	I/O 17+	49	I/O 7+	66	I/O 17-
16	GND	33	I/O 18+	50	I/O 7-	67	I/O 18-
17	GND	34	I/O 19+	51	I/O 8+	68	I/O 19-

Table 6-11: GX3609, GX3610 J1 Flex IO Bank A Connector

GX3609, GX3610 J3 Flex I/O Bank B Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	I/O 20+	52	I/O 28-
2	GND	19	GND	36	I/O 20-	53	I/O 29+
3	GND	20	GND	37	I/O 21+	54	I/O 29-
4	GND	21	GND	38	I/O 21-	55	I/O 30+
5	GND	22	GND	39	I/O 22+	56	I/O 30-
6	GND	23	GND	40	I/O 22-	57	I/O 31+
7	GND	24	GND	41	I/O 23+	58	I/O 31-
8	GND	25	GND	42	I/O 23-	59	I/O 32+
9	GND	26	GND	43	I/O 24+	60	I/O 32-
10	GND	27	5 V UUT	44	I/O 24 -	61	I/O 33+
11	GND	28	5 V UUT	45	I/O 25+	62	I/O 33-
12	GND	29	I/O 34+	46	I/O 25-	63	I/O 34 -
13	GND	30	I/O 35+	47	I/O 26+	64	I/O 35-
14	GND	31	I/O 36+	48	I/O 26-	65	I/O 36-
15	GND	32	I/O 37+	49	I/O 27+	66	I/O 37-
16	GND	33	I/O 38+	50	I/O 27-	67	I/O 38-
17	GND	34	I/O 39+	51	I/O 28+	68	I/O 39-

Table 6-12: GX3609, GX3610 J3 Flex IO Bank B Connector

GX3609, GX3610 J4 Flex I/O Bank C Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	I/O 40+	52	I/O 48-
2	GND	19	GND	36	I/O 40-	53	I/O 49+
3	GND	20	GND	37	I/O 41+	54	I/O 49-
4	GND	21	GND	38	I/O 41-	55	I/O 50+
5	GND	22	GND	39	I/O 42+	56	I/O 50-
6	GND	23	GND	40	I/O 42-	57	I/O 51+
7	GND	24	GND	41	I/O 43+	58	I/O 51-
8	GND	25	GND	42	I/O 43-	59	I/O 52+
9	GND	26	GND	43	I/O 44+	60	I/O 52-
10	GND	27	5 V UUT	44	I/O 44 -	61	I/O 53+
11	GND	28	5 V UUT	45	I/O 45+	62	I/O 53-
12	GND	29	I/O 54+	46	I/O 45-	63	I/O 54 -
13	GND	30	I/O 55+	47	I/O 46+	64	I/O 55-
14	GND	31	I/O 56+	48	I/O 46-	65	I/O 56-
15	GND	32	I/O 57+	49	I/O 47+	66	I/O 57-
16	GND	33	I/O 58+	50	I/O 47-	67	I/O 58-
17	GND	34	I/O 59+	51	I/O 48+	68	I/O 59-

Table 6-13: GX3609, GX3510 J4 Flex IO Bank C Connector

GX3609, GX3610 J2 Flex I/O Bank D Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	I/O 60+	52	I/O 68-
2	GND	19	GND	36	I/O 60-	53	I/O 69+
3	GND	20	GND	37	I/O 61+	54	I/O 69-
4	GND	21	GND	38	I/O 61-	55	I/O 70+
5	GND	22	GND	39	I/O 62+	56	I/O 70-
6	GND	23	GND	40	I/O 62-	57	I/O 71+
7	GND	24	GND	41	I/O 63+	58	I/O 71-
8	GND	25	GND	42	I/O 63-	59	I/O 72+
9	GND	26	GND	43	I/O 64+	60	I/O 72-
10	GND	27	5 V UUT	44	I/O 64 -	61	I/O 73+
11	GND	28	5 V UUT	45	I/O 65+	62	I/O 73-
12	GND	29	I/O 74+	46	I/O 65-	63	I/O 74 -
13	GND	30	I/O 75+	47	I/O 66+	64	I/O 75-
14	GND	31	I/O 76+	48	I/O 66-	65	I/O 76-
15	GND	32	I/O 77+	49	I/O 67+	66	I/O 77-
16	GND	33	I/O 78+	50	I/O 67-	67	I/O 78-
17	GND	34	I/O 79+	51	I/O 68+	68	I/O 79-

Table 6-14: GX3609, GX3610 J2 Flex IO Bank D Connector

GX3501, GX3509 and GX3510 Programming

Use the GXFPGA driver **GxFpgaPioxxx** functions to program the PIO (GX3501, GX3509 and GX3510) boards. The functions are described in detail in Chapter 7. Some of the functions are also available from the software front panel.

GX3501 TTL Buffer Expansion Board Specification

Number of Channels	80 I/O signals. Direction is configurable by software on a per pin basis
Logic Family	TTL or LVTTL, 5 volt tolerant inputs; Each group of 40 channels can be jumper configured to support TTL or LVTTL levels
Output Current	+/- 50 mA, sink or source
Input Leakage Current	+/- 5 uA
Power On State	All channels are configured as inputs
Input Protection	Overvoltage: -0.5 V to 6.5 V (input)

GX3509 Differential TTL Expansion Board Specification

Number of Channels	80 I/O signals. Direction is configurable by software on a per pin basis
Logic Family	Differential TTL, RS-485 compatible
Data Rate	35 Mbps (Max.)
Driver Output Current	+/- 60 mA (sink or source) (Max.)
Differential Output Voltage	1.5V min, 6V max (no load)
Receiver Input Current	+ 20 uA / -100 uA
Differential Input Voltage	+/- 12V (Max.)
Power On State	All channels are configured as inputs

GX3510 MLVDS Expansion Board Specification

Number of Channels	80 I/O signals Direction is configurable by software on a per pin basis
Logic Family	M- LVDS. Type 2 receiver
Data Rate	200 Mbps (Max.)
Differential Output Voltage	480 mV min, 650 mV max
Differential Input Voltage Threshold	150 mV max (positive going input) 50 mV max (negative going input)
Absolute Maximum Input Voltage Range	-1.8V to 4V
Power On State	All channels are configured as inputs

GX3540 – 40 Channel ECL I/O Expansion Board

The GX3540 consists of 20 differential ECL drivers and 20 differential ECL receivers. Each channel can be accessed via software commands for use in static I/O applications. The board includes on-board terminators for each differential channel. The **GX3640** model is used when ordering the GX3500 with the GX3540.

GX3540 Termination Resistors and Voltage Sources

The board is supplied with 510-ohm terminators that are terminated to –5.2 volts. Optionally, the board can be configured to use terminations with an onboard –2 volt termination voltage source. The terminating resistors are socketed, allowing the user to change the termination resistors and voltage sources for specific applications.

To change the termination voltage sources and termination resistors, remove the five resistor nets attached to the GX3540 expansion board (positions J1-J5). J1-J5 resistor sockets have eleven plugs and the provided resistor nets have ten pins. Pin 1 of the resistor net should go into the appropriate hole on the socket depending on the target termination voltage source. The GX3540 PCB is marked with –2.0V on one side and –5.2 V on the other. 51-ohm resistor nets and 510-ohm resistor nets are provided with the GX3540. When using the –2-volt source, a minimum of 49-ohm resistance is required and when using the –5.2-volt source, a minimum of 390-ohm resistance is required.

GX3540 Connectors

Connections to the GX3540 are made with a 68-pin VHDCI male plug connector. Shielded cables with matching connectors are available from Marvin Test Solutions.

GX3540 J1 – Flex I/O Bank A Inputs Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	ECL In 0+	52	ECL In 8-
2	GND	19	GND	36	ECL In 0-	53	ECL In 9+
3	GND	20	GND	37	ECL In 1+	54	ECL In 9-
4	GND	21	GND	38	ECL In 1-	55	ECL In 10+
5	GND	22	GND	39	ECL In 2+	56	ECL In 10-
6	GND	23	GND	40	ECL In 2-	57	ECL In 11+
7	GND	24	GND	41	ECL In 3+	58	ECL In 11-
8	GND	25	GND	42	ECL In 3-	59	ECL In 12+
9	GND	26	GND	43	ECL In 4+	60	ECL In 12-
10	GND	27	Reserved	44	ECL In 4-	61	ECL In 13+
11	GND	28	Reserved	45	ECL In 5+	62	ECL In 13-
12	GND	29	ECL In 14+	46	ECL In 5-	63	ECL In 14-
13	GND	30	ECL In 15+	47	ECL In 6+	64	ECL In 15-
14	GND	31	ECL In 16+	48	ECL In 6-	65	ECL In 16-
15	GND	32	ECL In 17+	49	ECL In 7+	66	ECL In 17-
16	GND	33	ECL In 18+	50	ECL In 7-	67	ECL In 18-
17	GND	34	ECL In 19+	51	ECL In 8+	68	ECL In 19-

Table 6-15: GX3540 J1 Flex IO Bank A Inputs Connector

GX3540 J2 – Flex I/O Bank D Outputs Connector

Pin#	Function	Pin#	Function	Pin#	Function	Pin#	Function
1	GND	18	GND	35	ECL Out 0+	52	ECL Out 8-
2	GND	19	GND	36	ECL Out 0-	53	ECL Out 9+
3	GND	20	GND	37	ECL Out 1+	54	ECL Out 9-
4	GND	21	GND	38	ECL Out 1-	55	ECL Out 10+
5	GND	22	GND	39	ECL Out 2+	56	ECL Out 10-
6	GND	23	GND	40	ECL Out 2-	57	ECL Out 11+
7	GND	24	GND	41	ECL Out 3+	58	ECL Out 11-
8	GND	25	GND	42	ECL Out 3-	59	ECL Out 12+
9	GND	26	GND	43	ECL Out 4+	60	ECL Out 12-
10	GND	27	Reserved	44	ECL Out 4-	61	ECL Out 13+
11	GND	28	Reserved	45	ECL Out 5+	62	ECL Out 13-
12	GND	29	ECL Out 14+	46	ECL Out 5-	63	ECL Out 14-
13	GND	30	ECL Out 15+	47	ECL Out 6+	64	ECL Out 15-
14	GND	31	ECL Out 16+	48	ECL Out 6-	65	ECL Out 16-
15	GND	32	ECL Out 17+	49	ECL Out 7+	66	ECL Out 17-
16	GND	33	ECL Out 18+	50	ECL Out 7-	67	ECL Out 18-
17	GND	34	ECL Out 19+	51	ECL Out 8+	68	ECL Out 19-

Table 6-16: GX3640 J2 Flex IO Bank D Output Connector

GX3540 Programming

Use the GXFPGA driver **GxFpgaPioxxx** functions to program the GX3540. Before using a **GxFpgaPioxxx** function, check the function reference to ensure the function applies to the GX3540. The functions are described in detail in Chapter 7. Some of the functions are also available from the software front panel.

GX3540 Specification

Number of Channels	20 differential inputs 20 differential outputs
Logic Family	MECL 10K compatible
Driver Output VOH	-0.98V (Min.) -0.81V (Max.)
Driver Output VOL	-1.95V (Min.) -1.63V (Max.)
Receiver Input VIH	-1.13V (Min.) -0.81V (Max.)
Receiver Input VIL	-1.95V (Min.) -1.48V (Max.)
Receiver Differential Input Level	150mV (typ.) 1.0V (Max.)
Receiver Common Mode Range	-2.85V (Min.) +0.3V (Max.)
Maximum Absolute Input Voltage Range	0 to -5.2V

ECL I/O Terminations	Each input and output is terminated with a 390-ohm resistor which is connected to -5.2V. Optionally, the board can be configured to use terminations with an onboard -2V termination voltage source
I/O Connectors	(2) SCSI III, VDHCI type, 68 pin female One for ECL outputs, one for ECL inputs

GX3571 – Video Generator Board

The GX3571 is a programmable video generator card which allows the user to generate video test images for displays or video processors. The module when combined with the GX3500 is called GX5671. The board provides a cost effective and flexible test solution for video product test applications. VGA, Composite, and S-video outputs are supported as well NTSC and PAL formats.

Features

The GX3771 supports a 640 x 480 format with standard frame rates of 60 Hz or 50 Hz. An on-board RAM (512Kb x 32) allows customers to create and load their own specific video test patterns. Up to 4 frames of video can be supported. Three 10-bit D to A converters support up to 10 bits of RGB color generation. The board supports VGA, Composite and S-video outputs via an external interface module which interfaces to the PXI card using a SCSI3 connector. Two on-board clock generators (14.318 MHz and 17.734 MHz) support the NTSC and PAL video standards respectively.

Programming

Use the GXFPGA driver **GxFpga3751xxx** functions to program the GX3571. The functions are described in details in Chapter 7. Some of the functions are also available from the software front panel.

Specifications

Video Output Characteristics	
Video Output formats	VGA (680 x 480) Composite video S-video
Video Frame memory	512 Kb x 32
Video Formats	NTSC & PAL
VGA Output	R,G,B, H sync, V sync R,G,B signal level: 0 – 0.7 volts, 75 ohm source / load termination H sync & V sync: “1” is > 2 volts, “0” < 0.8 volts
Composite Video	Output level: 2.5 V p-p (NTSC & PAL)
S - Video	Chrominance level: 1.8 V p-p (NTSC & PAL) Luminance level: 1.8 V p-p (NTSC & PAL)
I/O Connectors	(2) SCSI III, VHDCI type, 68 pin female Video out (interfaces to breakout module) Reserved for future expansion

Chapter 7 - Function Reference

Introduction

The GXFPGA driver functions reference chapter is organized in alphabetical order. Each function is presented starting with the syntax of the function, a short description of the function parameters description and type followed by a Comments, an Example (written in C), and a See Also sections.

All function parameters follow the same rules:

- Strings are ASCIIZ (null or zero character terminated).
- Most function's first parameter is *nHandle* (16-bit integer). This parameter is required for operating the board and is returned by the **GxFpgaInitialize** or the **GxFpgaInitializeVisa** functions. The *nHandle* is used to identify the board when calling a function for programming and controlling the operation of that board.
- All functions return a status with the last parameter named *pnStatus*. The *pnStatus* is zero if the function was successful, or less than a zero on error. The description of the error is available using the **GxFpgaGetErrorString** function or by using a predefined constant, defined in the driver interface files: GXFPGA.H, GXFPGA.BAS, GXFPGAVB, GXFPGA.PAS or GXFPGA.DRV.
- Parameter name are prefixed as follows:

Prefix	Type	Example
a	Array, prefix this before the simple type.	*anArray* (Array of Short)
n	Short (signed 16-bit)	nMode
d	Double - 8 bytes floating point	dReading
dw	Double word (unsigned 32-bit)	dwTimeout
l	Long (signed 32-bit)	lBits
p	Pointer. Usually used to return a value. Prefix this before the simple type.	pnStatus
sz	Null (zero value character) terminated string	szMsg
w	Unsigned short (unsigned 16-bit)	wParam
hwnd	Window handle (32-bit integer).	hwndPanel

Table 7-1: Parameter Prefixes

GXFPGA Functions

The following list is a summary of functions available for the GX3500:

Driver Functions	Description
General Functions	
GxFpgaInitialize	Initializes the driver for the board at the specified slot number using HW. The function returns a handle that can be used with other GXFPGA functions to program the board
GxFpgaInitializeVisa	Initializes the driver for the specified slot using VISA. The function returns a handle that can be used with other GXFPGA functions to program the board.
GxFpgaReset	Resets the GX3500 board to its default settings.
GxFpgaGetBoardSummary	Returns the board summary.
GxFpgaGetBoardType	Returns the board type.
GxFpgaGetDriverSummary	Returns the driver name and version.
GxFpgaGetErrorString	Returns the error string associated with the specified error number.
GxFpgaPanel	Opens the instrument panel dialog to used to interactively control the board.
FPGA Settings Functions	
GxFpgaGetEepromSummary	Returns the timestamp and filename of the last FPGA configuration written to EEPROM.
GxFpgaGetExpansionBoardBypass	Returns the current state of the Expansion Board Bypass.
GxFpgaGetExpansionBoardID	Returns the current Expansion Board ID.
GxFpgaLoad	Loads the volatile FPGA or the non volatile EEPROM with FPGA configuration data in the form of SVF or RPD files respectively.
GxFpgaLoadFromEeprom	Loads the FPGA with the contents of the EEPROM.
GxFpgaLoadStatus	Returns the progress of the last asynchronous load in percentage.
GxFpgaLoadStatusMessage	Returns a string describes the current load progress of the last asynchronous load.
GxFpgaReadMemory	Reads a 32 bit memory location that is a part of the FPGA's memory space.
GxFpgaReadRegister	Reads a 32 bit FPGA register.
GxFpgaSetExpansionBoardBypass	Sets the current state of the Expansion Board Bypass.
GxFpgaWriteMemory	Writes a buffer of 32 bit double words to the FPGA's memory space.
GxFpgaWriteRegister	Writes a buffer of 32 bit double words to the FPGA's register space.
Event (Interrupt) Functions	
GxFpgaSetEvent	Enables or disables an event handler
GxFpgaDiscardEvents	Clears the events queue
GxFpgaWaitOnEvent	Waits until event received or timeout occurred
PIO (GX3501, GX3509, GX3510, GX3540) Expansion Board Functions	
GxFpgaPioReset	Resets the specified PIO expansion board to its default settings.

Driver Functions	Description
GxFpgaPioGetChannel	Returns the specified PIO expansion board channel value.
GxFpgaPioGetChannelDirection	Returns the specified PIO expansion board channel direction.
GxFpgaPioGetGroup	Returns the specified PIO expansion board group’s channel values.
GxFpgaPioGetGroupDirection	Returns the specified PIO expansion board group’s channel direction.
GxFpgaPioSetChannel	Sets the specified PIO expansion board channel value.
GxFpgaPioSetChannelDirection	Sets the specified PIO expansion board channel direction.
GxFpgaPioSetGroup	Sets the specified PIO expansion board group’s channel values.
GxFpgaPioSetGroupDirection	Sets the specified PIO expansion board group’s channel direction.
GX3571 Expansion Board Functions	
GxFpga3571Reset	Resets the GX3571 to its default settings.
GxFpga3571GetAdvancedMode	Returns whether the board is in advanced mode.
GxFpga3571GetVideoMode	Returns the board’s current video output mode.
GxFpga3571LoadColor	Loads the video memory with the specified color.
GxFpga3571LoadArray	Loads the video memory from the specified array.
GxFpga3571LoadFile	Loads the video memory from the specified file.
GxFpga3571ResetModeline	Restores the modeline parameters to their defaults.
GxFpga3571SetAdvancedMode	Sets the board’s advanced mode.
GxFpga3571SetModeline	Sets the specified modeline parameter.
GxFpga3571SetVideoMode	Sets the board’s video output mode.
Upgrade firmware functions	
GxFpgaUpgradeFirmware	Upgrades the board’s firmware.
GxFpgaUpgradeFirmwareStatus	Monitor the firmware upgrade process.

GxFpga3571GetAdvancedMode

Purpose

Returns whether the board is in advanced mode.

Syntax

GxFpga3571GetAdvancedMode (*nHandle*, *pucMode*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pucMode	PBYTE	The mode can be as follows: 0. Off 1. On
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When the GX3571's advanced mode is turned on, the user-specified modeline parameters will be used. Modeline parameters can set using the **GxFpga3571SetModeline** function. When advanced mode is off, default modeline parameters are used to generate the video.

Example

The following example reads the state of the advanced mode into *ucMode*:

```
BYTE ucMode;
GxFpga3571GetAdvancedMode (nHandle, &ucMode, &nStatus);
```

See Also

GxFpga3571SetAdvancedMode, GxFpga3571SetModeline, GxFpga3571ResetModeline, GxFpgaGetErrorString

GxFpga3571GetVideoMode

Purpose

Returns the board's current video output mode.

Syntax

GxFpga3571GetVideoMode (*nHandle, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pnMode	PSHORT	The video output mode can be as follows: 0. GXFPGA_3571_VIDEO_MODE_STANDBY – Stand-by / Programming Mode 1. GXFPGA_3571_VIDEO_MODE_VGA – Output VGA Mode 2. GXFPGA_3571_VIDEO_MODE_NTSC – Output NTSC Mode
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

In Stand-by mode, video output is disabled. The board will automatically enter Stand-by mode when programming the video memory with the functions **GxFpga3571LoadColor**, **GxFpga3571LoadArray**, and **GxFpga3571LoadFile**.

Example

The following example reads the current video output mode into *nMode*:

```
SHORT nMode;
GxFpga3571GetVideoMode (nHandle, &nMode, &nStatus);
```

See Also

GxFpga3571SetVideoMode, GxFpgaGetErrorString

GxFpga3571LoadArray

Purpose

Loads the video memory with data from the specified byte array.

Syntax

GxFpga3571LoadArray (*nHandle*, *pucData*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pucData	BYTE[ ]	Specifies the array to be loaded into the video memory.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The array should contain at least 921,600 elements. Every three elements in the array specifies the red, green and blue values of a single pixel, respectively. The order of the pixels in the array should be row (top to bottom) and column (left to right). For example:

Byte Array Element	Pixel / Color Referenced
pucData[0]	Row 0, Column 0, Red Value
pucData[1]	Row 0, Column 0, Green Value
pucData[2]	Row 0, Column 0, Blue Value
pucData[3]	Row 0, Column 1, Red Value
pucData[1920]	Row 1, Column 1, Red Value
pucData[1923]	Row 1, Column 1, Red Value
pucData[921599]	Row 479, Column 639, Blue Value

Example

The following example loads the values from *pucData* into the Gx3571's video memory. The resulting video output will be all black pixels:

```
BYTE pucData[921600];
for (int i = 0; i++, i<921600)
{
    pucData[i]=0xFF;
}
GxFpga3571LoadArray (nHandle, pucData, &nStatus);
```

See Also

GxFpga5731LoadColor, GxFpga5731LoadFile, GxFpgaGetErrorString

GxFpga3571LoadColor

Purpose

Loads the video memory with the specified color.

Syntax

GxFpga3571LoadColor (*nHandle*, *dwRgbColor*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
dwRgbColor	SHORT	Specified the 24-bit RGB value to load into the video memory. See comments.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The GX3571 must be set to standby/programming mode using **GxFpga3571SetVideoOutputMode** prior to calling this function.

This function will program the video memory to output the user-specified RGB pixel. *dwRgbColor* should be formatted as follows:

32-bit DWORD = [2 bits XX] [8 bits red] [2 bits XX] [8 bits green] [2 bits XX] [8 bits blue] [2 bits XX]

The following constants are provided:

GXFPGA_3571_COLOR_BLACK = 0x00000000

GXFPGA_3571_COLOR_WHITE = 0x3FCFF3FC

GXFPGA_3571_COLOR_RED = 0x3FC00000

GXFPGA_3571_COLOR_GREEN = 0x000FF000

GXFPGA_3571_COLOR_BLUE = 0x000003FC

GXFPGA_3571_COLOR_YELLOW = 0x3FCFF000

GXFPGA_3571_COLOR_CYAN = 0x000FF3FC

GXFPGA_3571_COLOR_MAGENTA = 0x3FC003FC

Example

The following example sets the GX3571 to output a green screen:

```
GxFpga3571LoadColor (nHandle, GXFPGA_3571_COLOR_GREEN, &nStatus);
```

See Also

GxFpga3571SetVideoMode, GxFpga3571LoadArray, GxFpga3571LoadFile, GxFpgaGetErrorString

GxFpga3571LoadFile

Purpose

Loads the video memory with data from the specified bitmap file.

Syntax

GxFpga3571LoadFromBMPFile (*nHandle, szFileName, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
szFilename	LPCSTR	Pointer to a null-terminated string that specifies the name of the BMP file to be loaded in video memory.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function expects the bitmap file to be 640 pixels by 480 pixels.

Example

The following example loads the image data from "image.bmp" to the Gx3571's video memory:

```
GxFpga3571LoadFile (nHandle, "image.bmp", &nStatus);
```

See Also

GxFpga5731LoadArray, GxFpga5731LoadFile, GxFpgaGetErrorString

GxFpga3571Reset

Purpose

Resets the GX3571 to its default settings.

Syntax

GxFpga3571Reset (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function will set the GX3571 to output a white screen.

See Also

GxFpga3571ResetModeline, GxFpgaGetErrorString

GxFpga3571ResetModeline

Purpose

Restores the modeline parameters to their defaults.

Syntax

GxFpga3571ResetModeline (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Calling this function will set the modeline parameters as follows:

Modeline Parameter	Default Value
Horizontal Sync	640 (0x280)
Horizontal Front Porch	656 (0x290)
Horizontal Sync Pulse	752 (0x2F0)
Horizontal Back Porch	800 (0x320)
Vertical Sync	480 (0x1E0)
Vertical Front Porch	491 (0x1EB)
Vertical Sync Pulse	493 (0x1ED)
Vertical Back Porch	524 (0x20C)

Example

The following example resets the modeline parameters:

```
GxFpga3571ResetModeline (nHandle, &nStatus);
```

See Also

GxFpga3571SetModeline, GxFpgaGetErrorString

GxFpga3571SetAdvancedMode

Purpose

Sets the board's advanced mode.

Syntax

GxFpga3571SetAdvancedMode (*nHandle*, *ucMode*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
ucMode	BYTE	The mode can be as follows: 0. Off 1. On
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When the GX3571's advanced mode is turned on, the user-specified modeline parameters will be used. Modeline parameters can set using the **GxFpga3571SetModeline** function. When advanced mode is off, default modeline parameters are used to generate the video.

Example

The following example turns on advanced mode for the GX3571:

```
GxFpga3571GetAdvancedMode (nHandle, 0x1, &nStatus);
```

See Also

GxFpga3571GetAdvancedMode, GxFpga3571SetModeline, GxFpga3571ResetModeline, GxFpgaGetErrorString

GxFpga3571SetModeline

Purpose

Sets the specified modeline parameter.

Syntax

GxFpgaSetModeline (*nHandle*, *nScroll*, *nSignal*, *dwValue*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nScroll	SHORT	The scroll can be as follows: 0. GXFPGA_3571_MODELINE_SCROLL_HORIZONTAL 1. GXFPGA_3571_MODELINE_ SCROLL_VERTICAL
nSignal	SHORT	The modeline signal parameter can be as follows: 0. GXFPGA_3571_MODELINE_ SIGNAL_ SYNC 1. GXFPGA_3571_MODELINE_ SIGNAL_FRONT_PORCH 2. GXFPGA_3571_MODELINE_ SIGNAL_SYNC_PULSE 3. GXFPGA_3571_MODELINE_ SIGNAL_BACK_PORCH
dwValue	PDWORD	See comments below.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The value for *dwValue* is dependent on the signal parameter requested:

GXFPGA_3571_MODELINE_ SIGNAL_SYNC – The returned value is the number of pixels per line.

GXFPGA_3571_MODELINE_ SIGNAL_FRONT_PORCH – The number of black pixels drawn to the right / bottom of the screen.

GXFPGA_3571_MODELINE_ SIGNAL_SYNC_PULSE – The amount of time it takes: to start another line / move back up to the first line.

GXFPGA_3571_MODELINE_ SIGNAL_BACK_PORCH – The number of black pixels drawn to the left / top of the screen.

Example

The following example sets the number of pixels per line on the horizontal scroll to 720:

```
GxFpga3571SetModeline (nHandle, GXFPGA_3571_MODELINE_SCROLL_HORIZONTAL, GXFPGA_3571_MODELINE_
SIGNAL_SYNC, 720, &nStatus);
```

See Also

GxFpga3571ResetModeline, GxFpga3571SetAdvancedMode, GxFpga3571GetAdvancedMode, GxFpgaGetErrorString

GxFpga3571SetVideoMode

Purpose

Sets the board's video output mode.

Syntax

GxFpga3571SetVideoMode (*nHandle, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nMode	SHORT	The video output mode can be as follows: 0. GXFPGA_3571_VIDEO_MODE_STANDBY – Stand-by / Programming Mode 1. GXFPGA_3571_VIDEO_MODE_VGA – Output VGA Mode 2. GXFPGA_3571_VIDEO_MODE_NTSC – Output NTSC Mode
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

In Stand-by mode, video output is disabled. The board will automatically enter Stand-by mode when programming the video memory with the functions **GxFpga3571LoadColor**, **GxFpga3571LoadArray**, and **GxFpga3571LoadFile**.

Example

The following example sets the Gx3571 to output NTSC:

```
GxFpga3571SetVideoMode (nHandle, GXFPGA_3571_VIDEO_MODE_NTSC, &nStatus);
```

See Also

GxFpga3571SetVideoMode, GxFpgaGetErrorString

GxFpgaDiscardEvents

Purpose

Clears the event queue.

Syntax

GxFpgaDiscardEvents (*nHandle*, *nEventType*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nEventType	SHORT	Event type. Use the constant GT_EVENT_INTERRUPT (1). No other value is supported.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function clears the event queue and remove all pending events. Setting an event handler using the **GxFpgaSetEvent** automatically clears the event queue.

Example

The following example uses discard events to reset the queue after lengthy operation:

```
GxFpgaInitialize (1, &nHandle, &nStatus);
GxFpgaSetEvent(nHandle, GT_EVENT_INTERRUPT, TRUE, NULL, (PVOID)1, &nStatus);
while (TRUE)
{
    ! wait up to 1000 ms for the event
    GxFpgaWaitOnEvent(nHandle, GT_EVENT_INTERRUPT, 1000, &nStatus);
    if (nStatus!=0)    ! success event occurred
    {    printf("no event occurred - exiting");
         break;
    }
    else
    {   ! do something lengthy …
        ! now ready to receive more events
        GxFpgaDiscardEvents(nHandle, GT_EVENT_INTERRUPT, &nStatus);
}
```

See Also

GxFpgaInitialize, **GxFpgaGetErrorString, GxFpgaWaitOnEvent, GxFpgaSetEvent**

GxFpgaGetBoardSummary

Purpose

Returns the board information.

Syntax

GxFpgaGetBoardSummary (*nHandle*, *pszSummary*, *nMaxLen*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pszSummary	PSTR	Buffer to contain the returned board info (null terminated) string.
nMaxLen	SHORT	*pszSummary* buffer size.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the board information including the board firmware version and the board serial number.

Example

The following example returns the board information:

```
CHAR szSummary[1024];

GxFpgaGetBoardSummary (nHandle, szSummary, 1024, &nStatus);
```

See Also

GxFpgaInitialize, GxFpgaGetEepromSummary, GxFpgaGetErrorString

GxFpgaGetBoardType

Purpose

Returns the board type.

Syntax

GxFpgaGetBoardType (*nHandle*, *pnType*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pnType	PSHORT	Returned board type: 0. GXFPGA_UNKNOWN_BOARD_TYPE: unknown board type 1. GXFPGA_BOARD_TYPE_GX3500: board type is GX3500 2. GXFPGA_BOARD_TYPE_GX3500E: board type is GX3500E 3. GXFPGA_BOARD_TYPE_GX3700: board type is GX3700 4. GXFPGA_BOARD_TYPE_GX3700E: board type is GX3700E
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example returns the board type:

```
SHORT nType;
GxFpgaGetBoardType(nHandle, &nType, &nStatus);
```

See Also

GxFpgaInitialize, GxFpgaGetEepromSummary, GxFpgaGetErrorString

GxFpgaGetEepromSummary

Purpose

Returns the timestamp and filename of the last FPGA configuration written to EEPROM.

Syntax

GxFpgaGetEepromSummary (*nHandle, pszSummary, nMaxLen, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pszSummary	PSTR	Buffer to contain a summary indicating last FPGA EEPROM write timestamp and file name.
nMaxLen	SHORT	*pszSummary* buffer size.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the time stamp and file name indicating the last recorded EEPROM loading.

Example

The following example returns the EEPROM summary:

```
CHAR szSummary[1024];

GxFpgaGetEepromSummary (nHandle, szSummary, 1024, &nStatus);
```

See Also

GxFpgaLoad, GxFpgaGetBoardSummary, GxFpgaGetErrorString

GxFpgaGetExpansionBoardBypass

Purpose

Returns the current state of the Expansion Board Bypass.

Syntax

GxFpgaGetExpansionBoardBypass (*nHandle*, *puBankBypassControl*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
puBankBypassControl	PBYTE	4 Bit value for the FPGA I/O Bypass Control
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each bit in *puBankBypassControl* represents the switching state of a particular IO Bank. A high bit indicates that the I/O bank has been routed directly to the front I/O Connector. A low bit indicates that the I/O bank has been routed to the expansion board.

By default, all IO Banks are routed to the expansion board. The IO Banks can be routed to the external connectors or the expansion board by calling **GxFpgaSetExpansionBoardBypass**.

Example

The following example returns the 4 bit I/O Bypass Control value:

```
GxFpgaGetExpansionBoardBypass (nHandle, &uBankBypassControl, &nStatus);
```

See Also

GxFpgaSetExpansionBoardBypass, GxFpgaGetErrorString

GxFpgaGetExpansionBoardID

Purpose

Returns the current Expansion Board ID.

Syntax

GxFpgaGetExpansionBoardID (*nHandle*, *pucExpansionBoardID*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pucExpansionBoardID	PBYTE	Returned value that identifies the currently installed expansion board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The returned expansion board ID identifies the type of expansion board being used:

ucExpansionBoardID	Type of board	Examples
0x1	PIO expansion board	GX3501, GX3509, GX3510
0x2	ECL I/O board	GX3540
0x8	Video Generator	GX3571
0xF	No expansion board installed	N/A

Comments

The expansion board ID is read from P8 pins 19, 21, 23 and 25 to form a 4 bit integer (0-15).

Example

The following example returns the expansion board ID to the *ucExpansionBoardID*:

```
BYTE ucExpansionBoardID;
GxFpgaGetExpansionBoardID (nHandle, &ucExpansionBoardID, &nStatus);
```

See Also

GxFpgaGetErrorString

GxFpgaGetDriverSummary

Purpose

Returns the driver name and version.

Syntax

GxFpgaGetDriverSummary (*pszSummary ,nSummaryMaxLen, pdwVersion, pnStatus*)

Parameters

Name	Type	Comments
pszSummary	PSTR	Buffer to the returned driver summary string.
nSummaryMaxLen	SHORT	The size of the summary string buffer.
pdwVersion	PDWORD	Returned version number. The high order word specifies the major version number where the low order word specifies the minor version number.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The returned string is: "GXFPGA Driver for GX3500. Version 1.00, Copyright © 2009 Marvin Test Solutions – MTS inc.".

Example

The following example prints the driver version:

```
CHAR sz[128];
DWORD dwVersion;
SHORT nStatus;

GxFpgaGetDriverSummary (sz, sizeof sz, &dwVersion, &nStatus);
printf("Driver Version %d.%d", (INT)(dwVersion>>16), (INT)
    dwVersion &0xFFFF);
```

See Also

GxFpgaGetBoardSummary, GxFpgaGetErrorString

GxFpgaGetErrorString

Purpose

Returns the error string associated with the specified error number.

Syntax

GxFpgaGetErrorString (*nError*, *pszMsg*, *nErrorMaxLen*, *pnStatus*)

Parameters

Name	**Type**	**Comments**
nError	SHORT	Error number.
pszMsg	PSTR	Buffer to the returned error string.
nErrorMaxLen	SHORT	The size of the error string buffer.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the error string associated with the *nError* as returned from other driver functions.

The following table displays the possible error values; not all errors apply to this board type:

Resource Errors

0	No error has occurred
-1	Unable to open the HW driver. Check if HW is properly installed
-2	Board does not exist in this slot/base address
-3	Different board exist in the specified PCI slot/base address
-4	PCI slot not configured properly. You may configure using the PciExplorer from the Windows Control Panel
-5	Unable to register the PCI device
-6	Unable to allocate system resource for the device
-7	Unable to allocate memory
-8	Unable to create panel
-9	Unable to create Windows timer
-10	Bad or Wrong board EEPROM
-11	Not in calibration mode
-12	Board is not calibrated
-13	Function is not supported by the specified board

General Parameter Errors

-20	Invalid or unknown error number
-21	Invalid parameter
-22	Illegal slot number
-23	Illegal board handle
-24	Illegal string length
-25	Illegal operation mode

-26 Parameter is out of the allowed range

VISA Errors

-30 Unable to Load VISA32/64.DLL, make sure VISA library is installed

-31 Unable to open default VISA resource manager, make sure VISA is properly installed

-32 Unable to open the specified VISA resource

-33 VISA viGetAttribute error

-34 VISA viInXX error

-35 VISA ViMapAddress error

Miscellaneous Errors

-41 Unable to enable interrupt or event

-42 Unable to disable interrupt or event

-43 Event or interrupt timeout

-44 Event or interrupt wait error

Board Specific Errors

-50 Offset is out of range

-51 File Name is not valid

-52 Programming file could not be opened

-53 User FPGA Volatile Programming error

-54 User FPGA EEPROM Programming error

-55 Cannot program through software, External Programmer Detected

-56 FPGA or EEPROM is currently being loaded and is busy

-57 FPGA could not be reloaded with the EEPROM data

-58 Size and Offset must be multiple of 4

-59 Expansion board required for function not found

Example

The following example initializes the board. If the initialization failed, the following error string is printed:

```
CHAR    sz[256];
SHORT   nStatus, nHandle;
GxFpgaInitialize (3, &Handle, &Status);
if (nStatus<0)
{
    GxFpgaGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz);// prints the error string returns
}
```

GxFpgaInitialize

Purpose

Initializes the driver for the board at the specified slot number. The function returns a handle that can be used with other GXFPGA functions to program the board.

Syntax

GxFpgaInitialize (*nSlot*, *pnHandle*, *pnStatus*)

Parameters

Name	Type	Comments
nSlot	SHORT	GX3500 board slot number on the PXI bus.
pnHandle	PSHORT	Returned handle for the board. The handle is set to zero on error and <> 0 on success.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **GxFpgaInitialize** function verifies whether or not the GX3500 board exists in the specified PXI slot. The function does not change any of the board settings. The function uses the HW driver to access and program the board.

The Marvin Test Solutions HW device driver is installed with the driver and is the default device driver. The function returns a handle that for use with other Counter functions to program the board. The function does not change any of the board settings.

The specified PXI slot number is displayed by the **PXI/PCI Explorer** applet that can be opened from the Windows **Control Panel**. You may also use the label on the chassis below the PXI slot where the board is installed. The function accepts two types of slot numbers:

- A combination of chassis number (chassis # x 256) with the chassis slot number. For example, 0x105 (chassis 1 slot 5).
- Legacy nSlot as used by earlier versions of HW/VISA. The slot number contains no chassis number and can be changed using the **PXI/PCI Explorer** applet (1-255).

The returned handle *pnHandle* is used to identify the specified board with other GX3500 functions.

Example

The following example initializes two GX3500 boards at slot 1 and 2.

```
SHORT nHandle1, nHandle2, nStatus;
GxFpgaInitilize (1, &nHandle1, &nStatus);
GxFpgaInitilize (2, &nHandle2, &nStatus);
if (nHandle1==0 || nHandle2==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

See Also

GxFpgaInitializeVisa, GxFpgaReset, **GxFpgaGetErrorString**

GxFpgaInitializeVisa

Purpose

Initializes the driver for the specified PXI slot using the default VISA provider.

Syntax

GxFpgaInitializeVisa (*szVisaResource, pnHandle, pnStatus*)

Parameters

Name	Type	Comments
szVisaResource	LPCTSTR	String identifying the location of the specified board in order to establish a session.
pnHandle	PSHORT	Returned Handle (session identifier) that can be used to call any other operations of that resource
pnStatus	PSHORT	Returned status: 0 on success, 1 on failure.

Comments

The **GxFpgaInitializeVisa** opens a VISA session to the specified resource. The function uses the default VISA provider configured in your system to access the board. You must ensure that the default VISA provider support PXI/PCI devices and that the board is visible in the VISA resource manager before calling this function.

The first argument *szVisaResource* is a string that is displayed by the VISA resource manager such as NI Measurement and Automation (NI_MAX). It is also displayed by Marvin Test Solutions PXI/PCI Explorer as shown in the prior figure. The VISA resource string can be specified in several ways as follows:

- Using chassis, slot, for example: "PXI0::CHASSIS1::SLOT5"
- Using the PCI Bus/Device combination, for example: "PXI9::13::INSTR" (bus 9, device 9).
- Using alias, for example: "FPGA1". Use the PXI/PCI Explorer to set the device alias.

The function returns a board handle (session identifier) that can be used to call any other operations of that resource. The session is opened with VI_TMO_IMMEDIATE and VI_NO_LOCK VISA attributes. On terminating the application the driver automatically invokes **viClose**() terminating the session.

Example

The following example initializes a GX3500 boards at PXI bus 5 and device 11.

```
SHORT nHandle, nStatus;
GxFpgaInitializeVisa ("PXI5::11::INSTR", &nHandle, &nStatus);
if (nHandle==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

See Also

GxFpgaInitialize, GxFpgaReset, **GxFpgaGetErrorString**

GxFpgaLoad

Purpose

Loads the volatile FPGA or the non-volatile EEPROM with FPGA configuration data in the form of SVF or RPD files respectively.

Syntax

GxFpgaLoad (*nHandle, nTarget, szFileName nMode,, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nTarget	SHORT	Target can be as follows: 0. GXFPGA_LOAD_TARGET_VOLATILE 1. GXFPGA_LOAD_TARGET_EEPROM
szFileName	LPCSTR	Path and file name of the file containing the FPGA configuration data. If the programming mode is Volatile, then the file will have a .SVF extension. If the programming mode is EEPROM, then the file will have an .RPD extension.
nMode	SHORT	The loading mode can be as follows: 0. GXFPGA_LOAD_MODE_SYNC 1. GXFPGA_LOAD_MODE_ASYNC
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function can operate in synchronous mode or asynchronous mode. Synchronous mode means that the function is blocking and does not return until after the load operation has completed. The Asynchronous mode means that the function is non-blocking and returns immediately and allows the calling program to check the load status by calling **GxFpgaLoadStatus.**

Use the **GxFpgaLoadFromEeprom** function to load the volatile memory from the EEPROM. By default, when the card is powered up the volatile memory will be automatically load the configuration from the EEPROM.

Example

The following example loads the volatile FPGA with a Serial Vector File (SVF) in synchronous mode

```
GxFpgaLoad(nHandle, GXFPGA_LOAD_TARGET_VOLATILE, “C:\\MyDesign.SVF”, GXFPGA_LOAD_MODE_SYNC
&nStatus);
```

See Also

GxFpgaLoadStatus, GxFpgaLoadStatusMessage, GxFpgaGetEepromSummary, GxFpgaLoadFromEeprom, GxFpgaGetErrorString

GxFpgaLoadFromEeprom

Purpose

Loads the FPGA with the contents of the EEPROM.

Syntax

GxFpgaLoadFromEeprom (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

By default, when JP2 jumper is present, when the card is powered up the volatile memory will be automatically loaded with the configuration from the EEPROM.

Example

The following example loads the FPGA with the contents of the EEPROM:

```
GxFpgaLoadFromEeprom (nHandle, &nStatus);
```

See Also

GxFpgaLoad, GxFpgaGetErrorString

GxFpgaLoadStatus

Purpose

Returns the progress of the last asynchronous load in percentage.

Syntax

GxFpgaLoadStatus (*nHandle*, *pnPercentCompleted*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pnPercentCompleted	PSHORT	The percent complete of the current load, 0-100.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

100 percent indicates that the load has completed. This function is used to check the load status after calling **GxFpgaLoad** in Asynchronous mode.

Example

The following load an FPGA file in asynchronous mode and prints the progress:

```
SHORT nPercentage=0, nPriorPrecentage, nStatus, n;
CHAR  szMsg[1024];

GxFpgaLoad(nHandle, GXFPGA_LOAD_TARGET_VOLATILE, “C:\\MyDesign.SVF”, GXFPGA_LOAD_MODE_ASYNC
&nStatus);
while (nStatus==0 && nPrecentage<100)
{   GxFpgaLoadStauts (nHandle, &nPercentage, &nStatus);
    GxFpgaLoadStautsMessage (nHandle, szMsg, sizeof szMsg, &n);
    if (nPrecentage!=nPriorPrecentage)
       printf(“Load Complete=%i, Status=%s”, nPrecentage, szMsg);
    nPriorPrecentage=nPrecentage;
    sleep(300);
}
printf(“Load Complete=%i, Status=%s”, nPrecentage, szMsg);
```

See Also

GxFpgaLoad, **GxFpgaLoadStatusMessage, GxFpgaGetErrorString**

GxFpgaLoadStatusMessage

Purpose

Returns a string describes the current load progress of the last asynchronous load.

Syntax

GxFpgaLoadStatusMessage (*nHandle*, *pszMsg*, *nMsgMaxLen*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pszMsg	PSTR	A buffer to the returned message describing the current load status.
nMsgMaxLen	SHORT	Size of the pszMsg.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns the current load status into the user-supplied buffer. You can use the function to display the status progress and result after calling **GxFpgaLoad** in Asynchronous mode.

Example

The following load an FPGA file in asynchronous mode and prints the progress:

```
SHORT nPercentage=0, nPriorPrecentage, nStatus, n;
CHAR  szMsg[1024];

GxFpgaLoad(nHandle, GXFPGA_LOAD_TARGET_VOLATILE, “C:\\MyDesign.SVF”, GXFPGA_LOAD_MODE_ASYNC
&nStatus);
while (nStatus==0 && nPrecentage<100)
{   GxFpgaLoadStauts (nHandle, &nPercentage, &nStatus);
    GxFpgaLoadStautsMessage (nHandle, szMsg, sizeof szMsg, &n);
    if (nPrecentage!=nPriorPrecentage)
       printf(“Load Complete=%i, Status=%s”, nPrecentage, szMsg);
    nPriorPrecentage=nPrecentage;
    sleep(300);
}
printf(“Load Complete=%i, Status=%s”, nPrecentage, szMsg);
```

See Also

GxFpgaLoad, **GxFpgaLoadStatus, GxFpgaGetErrorString**

GxFpgaPanel

Purpose

Opens a virtual panel used to interactively control the GX3500.

Syntax

GxFpgaPanel (pnHandle, hwndParent, nMode, phwndPanel, pnStatus)

Parameters

Name	Type	Comments
pnHandle	PSHORT	Handle to a GX3500 board.
hwndParent	HWND	Panel parent window handle. A value of 0 sets the desktop as the parent window.
nMode	SHORT	The mode in which the panel main window is created. 0 for modeless window and 1 for modal window.
phwndPanel	HWND	Returned window handle for the panel.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function is used to create the panel window. The panel window may be open as a modal or a modeless window depending on the *nMode* parameters.

If the mode is set to modal dialog (*nMode*=1), the panel will disable the parent window (*hwndParent*) and the function will return only after the window was closed by the user. In that case, the *pnHandle* may return the handle created by the user using the panel Initialize dialog. This handle may be used when calling other GXFPGA functions.

If a modeless dialog was created (*nMode*=0), the function returns immediately after creating the panel window returning the window handle to the panel - *phwndPanel.* It is the responsibility of calling program to dispatch windows messages to this window so that the window can respond to messages.

Example

The following example opens the panel in modal mode:

```
DWORD dwPanel;
SHORT nHandle=0, nStatus;

GxFpgaPanel(&nHandle, 0, 1, &dwPanel, &nStatus);
```

See Also

GxFpgaInitialize, GxFpgaGetErrorString

GxFpgaPioGetChannel

Applies to

GX3501, GX3509, GX3510, GX3540

Purpose

Returns the specified PIO expansion board channel value.

Syntax

GxFpgaPioGetChannel (*nHandle*, *nGroup*, *nChannel*, *pnData*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nGroup	SHORT	Group value is as follows: 0. GXFPGA_PIO_GROUP_A 1. GXFPGA_PIO_GROUP_B 2. GXFPGA_PIO_GROUP_C 3. GXFPGA_PIO_GROUP_D
nChannel	SHORT	Channel range is 0 to 19.
pnData	PSHORT	Channel value: 0. Logic low 1. Logic high
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When using the GX3540, this function is limited to reading back channel values from GXFPGA_PIO_GROUP_A and GXFPGA_PIO_GROUP_D. Attempting to read back values from GXFPGA_PIO_GROUP_B or GXFPGA_PIO_GROUP_C will result in an error.

Example

The following example reads the logic level from channel 4 of Flex I/O group A into the *nData* variable:

```
SHORT nData;
GxFpgaPioGetChannel (nHandle, GXFPGA_PIO_GROUP_A, 0x4, &nData, &nStatus);
```

See Also

GxFpgaPioSetChannel, GxFpgaPioSetChannelDirection, GxFpgaGetErrorString

GxFpgaPioGetChannelDirection

Applies to

GX3501, GX3509, GX3510

Purpose

Returns the specified PIO expansion board channel direction.

Syntax

GxFpgaPIOGetChannelDirection (*nHandle, nGroup, nChannel, pdwDirection, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nGroup	SHORT	Group value is as follows: 0. GXFPGA_PIO_GROUP_A 1. GXFPGA_PIO_GROUP_B 2. GXFPGA_PIO_GROUP_C 3. GXFPGA_PIO_GROUP_D
nChannel	SHORT	Channel range is 0 to 19.
pnDirection	PSHORT	The channel direction can be as follows: 0. GXFPGA_PIO_DIRECTION_INPUT 1. GXFPGA_PIO_DIRECTION_OUTPUT
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example reads the direction from channel 4 of Flex I/O group A into the *nDirection* variable:

```
SHORT nDirection;
GxFpgaPIOGetChannelDirection (nHandle, GXFPGA_PIO_GROUP_A, 0x4, &nDirection, &nStatus);
```

See Also

GxFpgaPioSetChannelDirection, GxFpgaGetErrorString

GxFpgaPioGetGroup

Applies to

GX3501, GX3509, GX3510, GX3540

Purpose

Returns the specified PIO expansion board group's channel values.

Syntax

GxFpgaPioGetGroup (*nHandle, nGroup, pdwData, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nGroup	SHORT	Group value is as follows: 0. GXFPGA_PIO_GROUP_A 1. GXFPGA_PIO_GROUP_B 2. GXFPGA_PIO_GROUP_C 3. GXFPGA_PIO_GROUP_D
pdwData	PDWORD	Group's logic level values. Each of the low 20 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 19 is the last channel in the group.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function will return the logic level of all the channels in the specified group. When using the GX3540, this function is limited to reading back values from GXFPGA_PIO_GROUP_A and GXFPGA_PIO_GROUP_D. Attempting to read back values from GXFPGA_PIO_GROUP_B or GXFPGA_PIO_GROUP_C will result in an error.

Example

The following example gets the values of Flex I/O group A's channel and stores them into *dwData*:

```
DWORD dwData;
GxFpgaPioGetGroup (nHandle, GXFPGA_PIO_GROUP_A, &dwData, &nStatus);
```

See Also

GxFpgaPioSetGroup, GxFpgaPioSetGroupDirection, GxFpgaGetErrorString

GxFpgaPioGetGroupDirection

Applies to

GX3501, GX3509, GX3510

Purpose

Returns the specified PIO expansion board group's channel direction.

Syntax

GxFpgaPioGetGroupDirection (*nHandle, nGroup, pdwDirection, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nGroup	SHORT	Group value is as follows: 0. GXFPGA_PIO_GROUP_A 1. GXFPGA_PIO_GROUP_B 2. GXFPGA_PIO_GROUP_C 3. GXFPGA_PIO_GROUP_D
pdwDirection	PDWORD	Group's direction values. Each of the low 20 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 19 is the last channel in the group.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function will return the direction of all the channels in the specified group.

Example

The following example reads the direction of Flex I/O group A into the *dwDirection* variable:

```
DWORD dwDirection;
GxFpgaPioGetGroupDirection (nHandle, GXFPGA_PIO_GROUP_A, &dwDirection, &nStatus);
```

See Also

GxFpgaPioSetGroupDirection, GxFpgaGetErrorString

GxFpgaPioReset

Applies to

GX3501, GX3509, GX3510, GX3540

Purpose

Resets the specified PIO expansion board to its default settings.

Syntax

GxFpgaPioReset (*nHandle, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX3501, GX3509, GX3510: This command will set all channels for all four I/O groups to input.

GX3540: This command will set all channel in GXFPGA_PIO_GROUP_D to output a logical low.

Example

The following example resets the specified PIO expansion board:

```
GxFpgaPioReset (nHandle, &nStatus);
```

See Also

GxFpgaGetErrorString

GxFpgaPioSetChannel

Applies to

GX3501, GX3509, GX3510, GX3540

Purpose

Sets the specified PIO expansion board channel value.

Syntax

GxFpgaPioSetChannel (*nHandle ,nGroup*, *nChannel*, *nData*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nGroup	SHORT	Group value is as follows: 0. GXFPGA_PIO_GROUP_A 1. GXFPGA_PIO_GROUP_B 2. GXFPGA_PIO_GROUP_C 3. GXFPGA_PIO_GROUP_D
nChannel	SHORT	Channel range is 0 to 19.
nData	SHORT	Channel value: 0. Logic low 1. Logic high
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function will return the logic level of all the channels in the specified group. When programming the GX3540, this function is limited to setting values to GXFPGA_PIO_GROUP_D. Attempting to set values to any other group will result in an error.

Example

The following example sets the logic level of channel 4 of Flex I/O group A to logic high:

```
GxFpgaPioGetChannel (nHandle, GXFPGA_PIO_GROUP_A, 0x4, 0x1, &nStatus);
```

See Also

GxFpgaPioGetChannel, GxFpgaPioSetChannelDirection, GxFpgaGetErrorString

GxFpgaPioSetChannelDirection

Applies to

GX3501, GX3509, GX3510

Purpose

Sets the specified PIO expansion board channel direction.

Syntax

GxFpgaPioSetChannelDirection (*nHandle* , *nGroup*, *nChannel*, *nDirection*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nGroup	SHORT	Group value is as follows: 0. GXFPGA_PIO_GROUP_A 1. GXFPGA_PIO_GROUP_B 2. GXFPGA_PIO_GROUP_C 3. GXFPGA_PIO_GROUP_D
nChannel	SHORT	Channel range is 0 to 19.
nDirection	SHORT	The channel direction can be as follows: 0. GXFPGA_PIO_DIRECTION_INPUT 1. GXFPGA_PIO_DIRECTION_OUTPUT
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets the direction of channel 4 of Flex I/O group A to output:

```
GxFpgaPioSetChannelDirection (nHandle, GXFPGA_PIO_GROUP_A, 0x4, GXFPGA_PIO_DIRECTION_OUTPUT,
&nStatus);
```

See Also

GxFpgaPioGetChannelDirection, GxFpgaGetErrorString

GxFpgaPioSetGroup

Applies to

GX3501, GX3509, GX3510, GX3540

Purpose

Sets the specified PIO expansion board group's channel values.

Syntax

GxFpgaPioSetGroup (*nHandle*, *nGroup*, *dwData*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nGroup	SHORT	Group value is as follows: 0. GXFPGA_PIO_GROUP_A 1. GXFPGA_PIO_GROUP_B 2. GXFPGA_PIO_GROUP_C 3. GXFPGA_PIO_GROUP_D
dwData	DWORD	Group's logic level values. Each of the low 20 bits represents a channel in the group. Bit 0 is the first channel in the group and bit 19 is the last channel in the group.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function will return the logic level of all the channels in the specified group. When programming the GX3540, this function is limited to setting values to GXFPGA_PIO_GROUP_D. Attempting to set values to any other group will result in an error. This function will set the logic level of all the channels in the specified group.

Example

The following example sets the values of all the Flex I/O Bank A's channels to logic high:

```
GxFpgaPioSetGroup (nHandle, GXFPGA_PIO_GROUP_A, 0xFFFFF, &nStatus);
```

See Also

GxFpgaPioGetGroup, GxFpgaPioSetGroupDirection, GxFpgaGetErrorString

GxFpgaPioSetGroupDirection

Applies to

GX3501, GX3509, GX3510

Purpose

Sets the specified PIO expansion board group's channel direction.

Syntax

GxFpgaPioSetGroupDirection (*nHandle*, *nGroup*, *nDirection*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nGroup	SHORT	Group value is as follows: 0. GXFPGA_PIO_GROUP_A 1. GXFPGA_PIO_GROUP_B 2. GXFPGA_PIO_GROUP_C 3. GXFPGA_PIO_GROUP_D
nDirection	SHORT	The channel direction can be as follows: 0. GXFPGA_PIO_DIRECTION_INPUT 1. GXFPGA_PIO_DIRECTION_OUTPUT
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function will set the direction of all channels in the specified group.

Example

The following example sets the direction of all the channels of Flex I/O group A to input:

```
GxFpgaPioGetGroupDirection (nHandle, GXFPGA_PIO_GROUP_A, GXFPGA_PIO_DIRECTION_INPUT, &nStatus);
```

See Also

GxFpgaPioGetGroupDirection, GxFpgaGetErrorString

GxFpgaReadMemory

Purpose

Reads a 32-bit memory location that is a part of the FPGA's memory space.

Syntax

GxFpgaReadMemory (*nHandle, dwOffset, pvData, dwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
dwOffset	DWORD	The offset in the FPGA's shared memory space in terms of bytes, must be aligned to 4-bytes address.
pvData	PVOID	A buffer that will contain the data read. Buffer size must be as indicated by the *dwSize*.
dwSize	DWORD	The number of bytes to be read from the memory location must be multiple of 4.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function will read one or more double words from the FPGA's memory. The offset to be read from must be 4-byte aligned.

The Maximum value of dwOffset is 0x40000.

Example

The following reads an buffer of double words from the FPGA's memory:

```
DWORD adwData[100];
GxFpgaReadMemory (nHandle, 0x8, &adwData, 400, &nStatus);
```

See Also

GxFpgaWriteMemory, GxFpgaReadRegister, GxFpgaWriteRegister, GxFpgaGetErrorString

GxFpgaReadRegister

Purpose

Reads a 32-bit FPGA register.

Syntax

GxFpgaReadRegister (*nHandle, dwOffset, pvData, dwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
dwOffset	DWORD	The offset in the FPGA's register space in terms of bytes, must be aligned to 4-bytes address.
pvData	PVOID	A buffer that will contain the data read. Buffer size must be as indicated by the *dwSize*.
dwSize	DWORD	The number of bytes to be read from the memory location must be multiple of 4.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function will read one or more double words from the FPGA's registers. The offset to be read from must be 4-byte aligned.

The Maximum value of dwOffset is 0x400.

Example

```
DWORD adwData[100];
GxFpgaReadRegister (nHandle, 0x8, &adwData, 400, &nStatus);
```

See Also

GxFpgaWriteMemory, GxFpgaReadMemory, GxFpgaWriteRegister, GxFpgaGetErrorString

GxFpgaReset

Purpose

Resets the GX3500 board to its default settings.

Syntax

GxFpgaReset (*nHandle*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

After calling this function I/O Banks are connected to the expansion board

Example

The following example initializes and resets the GX3500 board:

```
GxFpgaInitialize (1, &nHandle, &nStatus);
GxFpgaReset (nHandle, &nStatus);
```

See Also

GxFpgaInitialize, **GxFpgaGetErrorString**

GxFpgaSetEvent

Purpose

Enables or disables an event handler.

Syntax

GxFpgaSetEvent (*nHandle, nEventType, bEnable, procCallback,pvUserData, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nEventType	SHORT	Event type. Use the constant GT_EVENT_INTERRUPT (1). No other value is supported.
bEnable	BOOL	Enable (<>0) or disable (0) the event.
procCallback	PROCEDURE	Optional. User supplied procedure, called by the driver when an event occurred.
pvUserData	PVOID	User data (pointer or value) that is passed to the callback procedure when an event occurred.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If NULL is passed in to the *procCallback* parameter, the only way to get notified that an event has occurred is to call the **GxFpgaWaitOnEvent** function.

The *procCallback* should be defined as follows:

GxFpgaCallback (*nHandle, nEventType,,pvUserData, pnStatus*) : Long

Example

The following example output whether an event received during 1 second:

```
GxFpgaInitialize (1, &nHandle, &nStatus);
GxFpgaSetEvent(nHandle, GT_EVENT_INTERRUPT, TRUE, NULL, (PVOID)1, &nStatus);
! wait up to 1000 ms for the event
GxFpgaWaitOnEvent(nHandle, GT_EVENT_INTERRUPT, 1000, &nStatus);
if (nStatus==0)        ! success event occurred
    printf("event occurred");
else
    printf("No event occurred");
GxFpgaSetEvent(nHandle, GT_EVENT_INTERRUPT, FALSE, NULL, (PVOID)1, &nStatus);
```

See Also

GxFpgaInitialize, **GxFpgaGetErrorString, GxFpgaWaitOnEvent, GxFpgaDiscardEvents**

GxFpgaSetExpansionBoardBypass

Purpose

Sets the current state of the Expansion Board Bypass.

Syntax

GxFpgaSetExpansionBoardBypass (*nHandle*, *uBankBypassControl*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
uBankBypassControl	BYTE	4 Bit value for the FPGA I/O Bypass Control.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

By default, all IO Banks are routed to the expansion board. The expansion board bypass settings can be read by calling **GxFpgaGetExpansionBoardBypass**.

Each bit represents the switching state of a particular IO Bank. A high bit indicates that the I/O bank has been routed directly to the front I/O Connector. A low bit indicates that the I/O bank has been routed to the expansion board.

Example

The following example connects all banks to the expansion board:

```
GxFpgaSetExpansionBoardBypass (nHandle, 0xF, &nStatus);
```

See Also

GxFpgaGetExpansionBoardBypass, GxFpgaReset, GxFpgaGetErrorString

GxFpgaUpgradeFirmware

Purpose

Upgrades the board's firmware.

Syntax

GxFpgaUpgradeFirmware (*nHandle, szFile, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
szFile	PCSTR	Path and file name of the firmware file. The firmware file extension is RPD.
nMode	SHORT	The upgrading firmware mode can be as follows: 0. GT_FIRMWARE_UPGRADE_MODE_SYNC: the function returns when upgrading firmware is done or in case of an error. 1. GT_ FIRMWARE_UPGRADE_MODE_ASYNC: the function returns immediately. The user can monitor the progress of upgrading firmware using the **GxFpgaUpgradeFirmwareStatus** API.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function used in order to upgrade the board's firmware. The firmware file can only be obtained by request from Marvin Test Solutions.

Note: Loading an incorrect firmware file to the board can permanently damage the board.

Example

The following example loads Upgrades the board's firmware using synchronous mode:

```
GxFpgaUpgradeFirmware (nHandle, "C:\\Gx3500Fw.rpd", GT_LOAD_MODE_SYNC, &nStatus);
```

See Also

GxFpgaUpgradeFirmwareStatus, GxFpgaGetErrorString

GxFpgaUpgradeFirmwareStatus

Purpose

Monitor the firmware upgrade process.

Syntax

GxFpgaUpgradeFirmwareStatus (*nHandle, pszMsg, nMsgMaxLen, pnProgress, pbIsDone, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
pszMsg	PSTR	Buffer to contain the message from the firmware upgrade process.
nMsgMaxLen	SHORT	*pszMsg* buffer size.
pnProgress	PSHORT	Returns the firmware upgrades progress.
pbIsDone	PBOOL	Returned TRUE if the firmware upgrades is done.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function is used in order to monitor the firmware upgrade process whenever the user called **GxFpgaUpgradeFirmware** API with GT_ FIRMWARE_UPGRADE_MODE_ASYNC mode.

Note: In order to prevent CPU over load if the function is called form within a loop, a delay of about 500mSec will be activated if the time differences between consecutive calls are less than 500mSec.

Example

The following example loads Upgrades the board's firmware using asynchronous mode, and ten monitors the firmware upgrade process:

```
CHAR    sz[256];
CHAR    szMsg[256];
BOOL    bIsDone=FALSE;
GxFpgaUpgradeFirmware (nHandle, "C:\\Gx3500Fw.rpd", GT_UPGRADE_FIRMWARE_MODE_ASYNC, &nStatus);
if (nStatus<0)
{   GxFpgaGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz);// prints the error string returns
}
While (bIsDone==FALSE || nStatus<0)
{   GxFpgaUpgradeFirmwareStatus (nHandle, szMsg, sizeof szMsg, &nProgress, &bIsDone, &nStatus);
    printf("Upgrade Progress %i", nProgress);
    sleep(1000);
}
if (nStatus<0)
{   GxFpgaGetErrorString(nStatus, sz, sizeof sz, &nStatus);
    printf(sz);// prints the error string returns
}
```

See Also

GxFpgaUpgradeFirmware, GxFpgaGetErrorString

GxFpgaWaitOnEvent

Purpose

Waits until event received or timeout occurred.

Syntax

GxFpgaWaitOnEvent (*nHandle*, *nEventType*, *lTimeout*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
nEventType	SHORT	Event type. Use the constant GT_EVENT_INTERRUPT (1). No other value is supported.
lTimeout	LONG	Timeout to wait in mill seconds.
pnStatus	PSHORT	Returned status: 0 on success (event occurred), negative number on failure.

Comments

The function suspends the current thread until an event occurred or until the specified timeout expired.

Example

The following example output whether an event received during 1 second:

```
GxFpgaInitialize (1, &nHandle, &nStatus);
GxFpgaSetEvent(nHandle, GT_EVENT_INTERRUPT, TRUE, NULL, (PVOID)1, &nStatus);
! wait up to 1000 ms for the event
GxFpgaWaitOnEvent(nHandle, GT_EVENT_INTERRUPT, 1000, &nStatus);
if (nStatus==0)        ! success event occurred
    printf("event occurred");
else if (nStatus==GT_EVENT_WAIT_TIMEOUT)
    printf("No event occurred (timeout)");
else
    printf("Event error");
GxFpgaSetEvent(nHandle, GT_EVENT_INTERRUPT, FALSE, NULL, (PVOID)1, &nStatus);
```

See Also

GxFpgaInitialize, **GxFpgaGetErrorString, GxFpgaSetEvent, GxFpgaDiscardEvents**

GxFpgaWriteMemory

Purpose

Writes a buffer of 32 bit double words to the FPGA's memory space.

Syntax

GxFpgaWriteMemory (*nHandle*, *dwOffset*, *pvData*, *dwSize*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
dwOffset	DWORD	The offset in the FPGA's shared memory space in terms of bytes, must be aligned to 4-bytes address.
pvData	PVOID	A buffer that will be written to the FPGA's shared memory. Buffer size must be as indicated by the *dwSize*.
dwSize	DWORD	The number of bytes to be read from the memory location, must be multiple of 4.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function will write one or more double words to the FPGA's memory. The offset to be written to must be 4-byte aligned.

The Maximum value of dwOffset is 0x40000.

Example

The following example writes 400 bytes to the card memory space at offset 8:

```
DWORD adwData[100];
GxFpgaWriteMemory (nHandle, 0x8, &adwData, 400, &nStatus);
```

See Also

GxFpgaReadMemory, GxFpgaReadRegister, GxFpgaWriteRegister, GxFpgaGetErrorString

GxFpgaWriteRegister

Purpose

Writes a buffer of 32 bit double words to the FPGA's register space.

Syntax

GxFpgaWriteRegister (*nHandle, dwOffset, pvData, dwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle for a GX3500 board.
dwOffset	DWORD	The offset in the FPGA's register space in terms of bytes, must be aligned to 4-bytes address.
pvData	PDWORD	A buffer that will be written to the FPGA's registers. Buffer size must be as indicated by the *dwSize*.
dwSize	DWORD	The number of bytes to be written to the registers must be multiple of 4.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function will write one or more double words to the FPGA's registers. The offset to be written to must be 4-byte aligned

The Maximum value of dwOffset is 0x400.

Example

The following example writes 400 bytes to the card register space at offset 8:

```
DWORD adwData[100];
GxFpgaWriteRegister (nHandle, 0x8, &adwData, 400, &nStatus);
```

See Also

GxFpgaReadRegister, GxFpgaReadMemory, GxFpgaWriteMemory, GxFpgaGetErrorString

Index

.

.NET ... ii

A

Adder Circuit ... 45

Adder Circuit with PCI Bus Connection ... 46

Adder Components ... 39

Adder Wizard ... 44, 45

Altera ... 3, 33

Applications ... 3

Architecture ... 1

Architecture

ATEasy ... 17, 29, 30

B

Board Description ... 1, 4, 59

Board Handle ... 32

Board Installation ... 19

Borland ... 29

Borland-Delphi ... 29

Bus Wiring Tool ... 42

C

C/C++ ... 29

Components Used ... 39

Connectors ... 21, 22, 23, 24, 25, 59, 63, 83, 87, 92

Corrupt files ... 16

Creating the FPGA Design ... 48, 49

Cyclone III ... 34

D

D Flip Flops ... 43

Decoder Properties ... 41

Delphi ... ii, 29

Design ... 40

Design File ... 37

Device Selection ... 34

Directories ... 17

Distributing the Driver ... 32

Driver

  Directory ... 17

  Files ... 17

Dynamic Digital Sequencer Circuit ... 52

E

Error-Handling ... 32

ESD ... 19

Examples ... 32

Expansion Board Design Guide ... 59

Expansion Boards ... 59

F

Features ... 3

Function Reference ... 95

G

Getting Started ... 15

GX3500Pins.tcl ... 36

GX3501 ... 59, 75

GX3501 Specification ... 90

GX3509 ... 59, 75

GX3509 Specification ... 91

GX3510 ... 59, 75

GX3510 Specification ... 91

GX3540 ... 59, 75, 92

GX3540 Specification ... 90, 93

GX3571 ... 59, 94

GX3571 Functions ... 97

GX3601 ... 59, 75, 83

GX3601 J1 ... 83

GX3601 J2 ... 86

GX3601 J3 ... 84

GX3601 J4 ... 85

GX3609 ... 59, 75, 87

GX3609, GX3610 J1 ... 87

GX3609, GX3610 J2 ... 90

GX3609, GX3610 J3 ... 88

GX3609, GX3610 J4 89
GX3610 ....59, 75, 87
GX3640 ....59, 75
GX3640 J1 Connector 92
GX3640 J2 Connector 93
GX3671 59
GXFPGA 1, 16
Driver-Description 29
Header-file 29
GXFPGA Driver 29
GXFPGA driver functions 31
GXFPGA Functions 96
GXFPGA Software 17
GXFPGA.bas 29
GXFPGA.dll 30
GXFPGA.EXE 16
GXFPGA.h 29
GXFPGA.lib 29
GXFPGA.llb 30
GXFPGA.pas 29
GXFPGA.vb 29
GxFpga3571GetAdvancedMode 97, 98
GxFpga3571GetVideoMode....97, 99
GxFpga3571LoadArray 97, 100
GxFpga3571LoadColor 97, 101
GxFpga3571LoadFile 97, 102
GxFpga3571Reset 97, 103
GxFpga3571ResetModeline 97, 104
GxFpga3571SetAdvancedMode 97, 105
GxFpga3571SetModeline 97, 106
GxFpga3571SetVideoMode 97, 107
GXFPGA64.DLL 29
GXFPGA64.lib 29
GXFPGABC.lib....29
GxFpgaDiscardEvents 96
GxFpgaGetBoardSummary 96, 109
GxFpgaGetBoardType....96, 110
GxFpgaGetDriverSummary....32, 96, 114
GxFpgaGetEepromSummary 96, 111
GxFpgaGetErrorString 32, 95, 96, 115, 116
GxFpgaGetExpansionBoardBypass....96, 112
GxFpgaGetExpansionBoardID 74, 96, 113
GxFpgaInitialize 9, 18, 31, 32, 95, 96, 117
GxFpgaInitializeVisa 9, 18, 31, 32, 95, 96, 118
GxFpgaLoad 96, 119
GxFpgaLoadFromEeprom 96, 120
GxFpgaLoadStatus 96, 121
GxFpgaLoadStatusMessage 96, 122
GxFpgaPanel 96, 123
GxFpgaPioGetChannel 97, 124
GxFpgaPioGetChannelDirection 97, 125
GxFpgaPioGetGroup 97, 126
GxFpgaPioGetGroupDirection 97, 127
GxFpgaPioReset 96, 128
GxFpgaPioSetChannel....97, 129
GxFpgaPioSetChannelDirection....97, 130
GxFpgaPioSetGroup....97, 131
GxFpgaPioSetGroupDirection....97, 132
GxFpgaReadMemory 96, 133
GxFpgaReadRegister 96, 134
GxFpgaReset 96, 108, 135, 136, 140
GxFpgaSetEvent 96
GxFpgaSetExpansionBoardBypass 74, 96, 137
GxFpgaUpgradeFirmware 97, 138
GxFpgaUpgradeFirmwareStatus 97, 139
GxFpgaWaitOnEvent 96
GxFpgaWriteMemory 96, 141
GxFpgaWriteRegister 96, 142
H
Handle 19, 20, 31, 32
HW 16, 17, 21, 29, 32
I
If You Need Help i
Installation 15, 16

Precautions-Static-Electricity 19
Procedures-All-Boards 19, 21
Installation Directories 17
Interfaces 15
Internal Connections..75
Introduction 3

J

J1 ....22, 25, 83, 87, 92
J1-J4..83, 87
J2 ....22, 28, 86, 90, 93
J3 ....22, 26, 84, 88
J4 ....22, 27, 85, 89
JP2 22
JP3 22
JP4 22
JP5 22
JP6 22
Jumpers 4, 22

L

LabView 30
LabView/Real Time 30
Linux 30

M

Mechanical Guide..61
MegaWizard Plug-In 40

N

nHandle 30

O

OnError...30
Overview 3

P

P10 and P11 Connectors..68
P10 Connector 69
P11 Connector 71, 73
P8 and P9 Host Connectors 64
Packing List..15
Panel 8, 10, 11, 12, 13, 16, 31, 123
Panel About Page..13
Part / Model Number 15
Pascal...29
PCI...17
PCI Address Decoder Circuit 42
Pin Assignment...34
PLL Wizard Dialog Box...50
Plug & Play...21
programming 29
Programming
Borland-Delphi 29
Error-Handling..32
Visual..29
Programming Examples 32
Programming the GX3500......................................29
PXI...................................4, 5, 16, 18, 19, 20, 21, 31
PXI System...18
PXI/PCI Explorer 9, 18, 31, 32, 117, 118
PXIeSYS.INI 9
PXISYS.INI..9

Q

Quartus ....ii, 1, 3, 33, 34

R

RAM Wizard Dialog Box.......................................51
README.TXT 17
Removing a Board..21
Reset 32
RPD 53, 54

S

Safety and Handling i
Setup...16, 17
Setup Maintenance 16
Setup-and-Installation...15
Slot..9, 16, 19, 21, 31
Software..16
Specifications 1, 7
Specifications

SVF..53, 54, 56
Symbol Insert Dialog Box 40
Symbol Properties...47
System 16
Directory...17
System Requirements 16

T

Task Flow 37
TCL script...34
Testing the Design 57

U

User FPGA Timing Access

V

Virtual Panel 8, 9, 10, 11, 12, 13, 16, 123
Initialize Dialog 8, 9
VISA............................9, 17, 18, 31, 32, 96, 117, 118
Visual..29
Visual Basic.. ii, 29
Visual Basic .NET 29
Visual C++ .... ii, 29

W

Warranty i