User Manual
DN2076K10
DN2076K10
User Manual

DOCUMENT PATH
C:\dncvs\Boards\DN0211_DN2076K10\Documents\Manual\DN2076K10_manual_rev02.docx

LAST SAVED BY
Neal Harder

LAST SAVE DATE
7/16/2012 11:52:00 AM
## Contents

1 INTRODUCTION .................................................................................................................. 5
  1.1 AUDIENCE..................................................................................................................... 5
  1.2 CONVENTIONS.............................................................................................................. 5
  1.3 RESOURCES.................................................................................................................. 5
  1.4 ERRATA......................................................................................................................... 7

2 QUICK START GUIDE ......................................................................................................... 8
  2.1 STEPS TO FOLLOW....................................................................................................... 8

3 HARDWARE ......................................................................................................................... 16
  3.1 OVERVIEW.................................................................................................................... 16
  3.2 VIRTEX 6 FPGA............................................................................................................. 17
  3.3 CLOCK RESOURCES..................................................................................................... 19
  3.4 NMB BUS....................................................................................................................... 24
  3.5 FPGA INTERCONNECT.................................................................................................. 27
  3.6 SODIMM (DDR3) CONNECTORS.................................................................................. 28
  3.7 DAUGHTER CARD CONNECTORS................................................................................. 31
  3.8 FPGA CONFIGURATION.............................................................................................. 44
  3.9 MARVELL CPU............................................................................................................ 45
  3.10 CONFIG FPGA............................................................................................................ 49
  3.11 RS232 ......................................................................................................................... 51
  3.12 USER LEDS ................................................................................................................. 52
  3.13 SPI FLASH ................................................................................................................... 53
  3.14 USER TEST POINTS ................................................................................................... 53
  3.15 “CONFIG MCTOR” CONNECTOR ............................................................................. 54
  3.16 USER SATA ............................................................................................................... 54
  3.17 SFP AND GIGABIT ETHERNET ................................................................................ 55
  3.18 10 GIGABIT ETHERNET .......................................................................................... 57
  3.19 GTX EXPANSION (SEAM) HEADER ....................................................................... 57
  3.20 ENCRYPTION ............................................................................................................. 60
  3.21 JTAG ............................................................................................................................ 61
  3.22 MECHANICAL ............................................................................................................. 62
  3.23 POWER ....................................................................................................................... 63
  3.24 HEAT ........................................................................................................................... 65
  3.25 RESET ......................................................................................................................... 66
  3.26 SYSTEM MONITOR .................................................................................................. 68
  3.27 LED REFERENCE LIST ............................................................................................. 69
  3.28 TEST POINT REFERENCE LIST .............................................................................. 72

4 SOFTWARE .......................................................................................................................... 74
  4.1 EMU HOST SOFTWARE .............................................................................................. 74
  4.2 WRITING YOUR OWN SOFTWARE ............................................................................. 79
4.3 HOW EMULib WORKS ............................................................................................................. 79
4.4 MARVEL ENVIRONMENT ...................................................................................................... 80

5 REFERENCE DESIGN ............................................................................................................. 88
  5.1 NMB SPACE MAP ............................................................................................................. 88
  5.2 COMPILING ....................................................................................................................... 89

6 TROUBLESHOOTING ............................................................................................................. 90
  6.1 GENERAL PROCEDURE ..................................................................................................... 90

7 ORDERING INFORMATION ................................................................................................. 91
  7.1 PART NUMBER .................................................................................................................. 91
  7.2 HOW TO ORDER ............................................................................................................... 91
  7.3 BOARD OPTIONS ............................................................................................................. 91
  7.4 COMPLIANCE INFORMATION .......................................................................................... 91

8 GLOSSARY ............................................................................................................................ 92
1 INTRODUCTION

Welcome to the DN2076K10 FPGA Platform. This manual will assist in setting up, using, and understanding the hardware aspects of this product. Please consult the Emu User Manual for software documentation.

1.1 Audience

This product is marketed and sold to engineers who are familiar with circuit board design, physically probing AC waveforms, programming FPGAs, wiring HDL code, reading device data sheets, reading C source code, and writing software. The provided support material all assumes that the user already has these skills.

1.2 Conventions

Computer input or output
Bad advice you should ignore
http://links.to.web.pages

1.3 Resources

The following list includes the resources that you are expected to make use of.

1.3.1 Website

The product page for this product is on the internet, here:

http://www.dinigroup.com/DN2076k10.html

This page contains:
- Block Diagram of the board
- Marketing Product description
- List of supported features
- Latest Errata
- Latest software and firmware update package
- Latest version of this document.
1.3.2  **Product Package**

The board comes with a USB memory stick with files on it. On the root directory, there is a file called "Support Package Contents.pdf" that describes the contents and the directory structure.

This package contains the software installed on the board as well as the software that should be installed on your host computer.

1.3.3  **Reference Design**

The product package contains a set of FPGA designs written in Verilog HDL that produce working configuration files for the FPGAs on the DN2076K10. Project files and batch script files that use Xilinx ISE to build the designs are also provided. These example files can be used to quickly create working .bit files for the FPGAs.

The reference design implements every feature on the board, including DDR3 memory, Rocket I/O, and others. You are free to adopt any of the device controllers used in this reference design.

For most customers, the most interesting part of the reference design will probably be the UCF file, which contains a list of all the usable signals connected to the FPGA and the correct IOSTANDARD attribute to use with each.

1.3.4  **Schematics and Netlist**

This user manual does not list specifications for all of the devices connected to the FPGAs, and so to correctly use them, you will have to refer to the device datasheet and the schematic. The schematic is provided in PDF format. If you need a machine-readable format, you can use the provided ASCII netlist of the board. The ASCII netlist contains only nets on the DN2076K10 that are connected to usable I/O on the FPGA.

1.3.5  **Device Datasheet Library**

There is a PDF datasheet provided for every part used on the board. It is in the user support package.

1.3.6  **Xilinx**

Questions about the use of Virtex 6 FPGAs or ISE that aren't specific to the DN2076K10 should be directed to Xilinx.

1.3.7  **EMAIL AND Telephone Technical support**

Email support@dinigroup.com for the best and quickest connection to an actual Engineer that will be able to directly answer your questions.

Phone support is also available Pacific Standard Time from 9AM to 5PM from Monday through Friday, excluding USA federal holidays. Support is available in English. Support for boards purchased through distributors can additionally be provided by the distributor. Distributors are listed in the ordering information section.

Telephone (USA): 858-454-3419
1.4 Errata

The circuit board is currently in revision number: 02.
There are currently no known errata for revision 02.

Revision 01 Errata:
Issue: USB Connector is super weak and falls of the board when handled roughly or examined.
Affected Users: Any user with a revision 1 board who requires USB 3.0 connection to FPGA G
Work Around: Get a revision 2 circuit board.

End of Errata List
2 Quick Start Guide

This section will walk through an example session using the board.

2.1 Steps to Follow

2.1.1 Examine Contents of Box

The box containing the product should have come with the following units:

- [ ] DN2076K10 board (green)
- [ ] RS232 serial Cable
- [ ] DB9-to-IDC cable adapter
USB Stick containing user support package
USB Stick containing FPGA .bit files and a shell script.
Packing foam
Mysterious PSU "starter" device
ATX Power supply
PCI Express cable
PCI Express host adapter card

2.1.2 Before you power on

Take ESD precautions whenever handling the board.
Place the board on a clear desk with static control padding. You can use the silver-colored bag the board comes wrapped in as a static control surface. Make sure you neutralize the static in your fingers with the surface before every time you contact the board.

2.1.3 Install the PCI Express adapter board in a PCI Express slot (optional)

During the course of your project, if you intend to control the board using PCI Express, then you should complete this step.

The board fits into any 8x or 16x PCI Express slot. Make sure the computer is powered off when you install the board into it. There are two x4 cable connectors on the board. You should use only the bottom one with the DN2076K10 because the DN2076K10 PCI Express interfaces only support 1x or 4x PCI Express links. The top connector on the PCI Express cable adapter card is for use with 8x links.

Connect one end of the provided PCI Express cable to the bottom connector of the adapter card, and the other end of the cable to the DN2076K10. The correct connector is labeled “PCIE Marvell”.

2.1.4 Connect Ethernet Cable

This step is optional. During the course of your project, if you intend to use Ethernet to control the board, then you should complete this section.

Have a computer network. In order to be able to access the board over the network, the network must support DHCP. Otherwise, the board will fail to have a usable IP address. Connect the RJ45 connector on the DN2076K10 to your network. If DHCP is not available a static IP address can be assigned to the board. See the Emu User Manual for more information. Traffic must be allowed on certain ports in order for the Emu Software to communicate with the board. See the Emu User Manual for more information.

2.1.5 Connect USB Cable

This step is optional. During the course of your project, if you intend to control the board directly from a computer over USB, then you should complete this step.

Connect a USB cable from the host computer to the "USB type B" connector on the board. If the board is plugged into PCI Express, the computer that connects to the board with USB does not necessarily need to be the same computer.

2.1.6 Connect power cables

The DN2076K10 requires three PCI Express power cables to turn on. You cannot choose to use only some of the power connectors. The provided ATX power supply has appropriate power cables. The DN2076K10 uses a lot of power, so it is recommended that you only use the power supply supplied with the system because the wiring on the provided unit is sufficient to safely handle the heavy current load.

2.1.7 Power on the board

Turn on the power supply. The power supply will not turn on without a "PSU starter" device. One has been provided for you.

The board has a self-boot process that takes 30 to 45 seconds.

2.1.8 Using a USB pen drive to control the board

The board is provided with a USB pen drive that has on it a Linux shell script. If you plug the USB stick into the board, then the board will automatically run the shell script. The shell script on the provided pen drive will cause the FPGAs to load with the reference design .bit files. You can tell that the .bit files are loading because after each FPGA configures, a blue LED will appear on the board. A USB pen drive left connected to the board will execute each time the board is powered on, allowing the user to customize clock settings, FPGA load files, and other features, without the need for host control or software intervention.

2.1.9 Host Software

Whether the board is connected to a computer using USB, PCI Express or Ethernet, the board is controlled using a program that Dini Group has provided called "Emu". The Emu program is
provided in source and as binaries on the user support package. See the Emu User Manual for additional information on installing and using the Emu software.

The rest of this guide will assume you are using a Windows computer, however you can also use Linux. If you are using Linux the instructions may be slightly different.

If you are using PCI Express then you need to install a PCI Express driver. This can be done in Windows using the "device manager". The driver files are provided in the support package D:\Host_Software\emu\Drivers\windows_pci. Search the internet if you are unsure how to install a driver from device manager.

If you are using USB then you will need to install a USB driver. [Linux does not require a USB driver]. This can be done in Windows using the "device manager". The driver files are provided in the support package D:\Host_Software\emu\Drivers\windows_usb. Search the internet if you are unsure how to install a driver from the device manager.

Ethernet does not require a driver.

2.1.10 Selecting a board

Run the provided "emu" program, located in the user support package here:
D:/Host_Software/emu/App/out_release/emu_gui_win32.exe

This window will appear.
From the "Board" menu, choose "select board". If you are connected to the board over PCI Express, USB and Ethernet simultaneously, then there will be three options in the pull-down menu. Each interface is treated like a separate board. From the pull-down menu, you can see the serial number of each board. The serial number in this menu should match the serial number located on a sticker near DIMM D of your board.

Once you have selected a board, your window should look like this.
2.1.11 Configure an FPGA

You can configure an FPGA by clicking on the image of it in EMU and selecting "configure" from the pop-up window. There are some example .bit files that you can use in the support package located at

D:\FPGA_reference_designs\Programming Files\dn0211_dn2076k10\user_fpga\MAINTEST

Be sure to choose bit files that are compiled for the correct type of FPGA that you have installed on the board, as indicated by the folders they are stored in. The FPGA that the bitfile is intended for is indicated by its filename, like “fpga_a.bit” is for FPGA A.

After the FPGA successfully configures, a blue dot will appear next to any configured FPGA.
2.1.12 Setting board controls and options

The primary board settings that you will need to modify are the clock settings. There are six clocks on the board that have modifiable options. Let's change the clock frequency of G0 just for kicks.

In the EMU window, click on the right side where it says "CLOCKS: G0". A pop-up menu will allow you to change the frequency of clock G0. You can also change the frequency using the "Clocks/Temps" menu in the menu bar. The clock frequency of the seven main clocks are periodically measured and displayed on the screen.

2.1.13 Hardware Verification Test

To run the hardware test, from the "Test" menu, select "selected tests".

![Hardware Verification Test Menu](image)

The tests that you can run now are the temperature test, clock test, blockram test, intercon test, lvds test and flash test. Let's skip the DRAM test and the "factory tests" for now. After you hit the "OK" button, the program will ask you to locate the "bit file directory". This is where the test FPGA load files are stored. Select this folder in the user support package:

D:\FPGA_Reference_designs\Programming Files
After each test runs, the Emu window will print "PASS" or "FAIL", and at the end of testing a summary will be presented with the results of each test that was run.

Note that the factory tests require specialized hardware and therefore cannot be run in the field. The DRAM test requires that DDR3 SODIMM memory units are installed in the available SODIMM slots, with the slots set for the correct voltage level. If these conditions are not met then the test will fail.
3 Hardware

This section describes the board hardware.

3.1 Overview

Below is a block diagram of the board.

The board contains one Xilinx Virtex-6 FPGAs in the “FF1759” package, and six Xilinx Virtex-6 FPGAs in the "FF1760" package. There are 5 different Xilinx part numbers that come in this package. All 5 of these are compatible with this board. The board can come with any number from 1 to 7 of FPGAs installed, leaving the unused/unwanted chip positions vacant.

The Virtex 6 "ConfigFPGA" is not intended for your use, so you should think of it as more of a "controller/bus switch/monitor".
Interconnect between the FPGAs is fixed, and routed in a point-to-point fashion. The interconnect is represented in the block diagram as arrows between the FPGAs. All of the interconnect is user-defined. Notice that some of the interconnect on the board is colored gold instead of black. The yellow interconnect is only available for use if the FPGAs to which it connects are a "large package" FPGA, namely LX550T or SX475T.

There is one bus of 52 signals that connects all seven chips. This bus may be used for broadcast signals to all FPGAs.

To connect to other systems, or off-board I/O, there are three "daughtercard" connectors provided. In the block diagram these appear as yellow rectangles. The user is expected to buy a daughter card that contains the I/O interface that is required, or to design their own.

Three DDR3 sockets (red rectangles) are on the board to provide bulk memory for use in the FPGA. You can use standard, off-the-shelf laptop memory (SODIMM), or you can use one of the many memory technology DIMMs that the Dini Group provides.

Getting data on and off the board is accomplished through the Marvell CPU. It provides USB, Ethernet, PCI Express, interfaces. It connected to the user FPGAs through the "NMB" interface, which provides an easy-to-use 20-30MB link to all 7 FPGAs. A high-speed PCI Express only NMB solution is also available if greater bandwidth is required; up to 380MB/s. Contact Dini Group support for more information. The standard NMB interface inside the FPGA and on the host PC is very simple, because all of the software and hardware between has already been designed, proven, and optimized.

3.2 Virtex 6 FPGA

The Virtex 6 FPGA provides the latest features, largest sizes, and fastest speeds from Xilinx.

3.2.1 Stuffing Options

Each of the six main FPGA locations can be installed with any compatible density of FPGA, or not installed at all, in any combination. In this way, you only will have to pay the (significant) FPGA cost for the FPGAs that you will actually use.

Below there is a table that describes the major differences between the available FPGAs. Only FPGAs that Xilinx sells in the "FF1759" or "FF1760” package are compatible with this board.

<table>
<thead>
<tr>
<th>FPGA</th>
<th>Speed Grades</th>
<th>Flip-flops</th>
<th>Equivalent ASIC Gates</th>
<th>All Board Features?</th>
<th>25x18 multipliers</th>
<th>Memory (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LX760</td>
<td>1L, 1, 2</td>
<td>948,480</td>
<td>5.5M</td>
<td>Y</td>
<td>864</td>
<td>25.9M</td>
</tr>
<tr>
<td>LX550T</td>
<td>1L, 1, 2</td>
<td>687,360</td>
<td>4.0M</td>
<td>Y</td>
<td>864</td>
<td>22.8M</td>
</tr>
<tr>
<td>LX360T</td>
<td>1L, 1, 2, 3</td>
<td>455,040</td>
<td>2.6M</td>
<td>N</td>
<td>576</td>
<td>15.0M</td>
</tr>
<tr>
<td>LX240T</td>
<td>1L, 1, 2, 3</td>
<td>301,440</td>
<td>1.7M</td>
<td>N</td>
<td>768</td>
<td>15.0M</td>
</tr>
<tr>
<td>SX475T</td>
<td>1L, 1, 2</td>
<td>595,200</td>
<td>3.4M</td>
<td>Y</td>
<td>2016</td>
<td>38.3M</td>
</tr>
<tr>
<td>SX315T</td>
<td>1L, 1, 2, 3</td>
<td>394,000</td>
<td>2.3M</td>
<td>N</td>
<td>1344</td>
<td>25.3M</td>
</tr>
</tbody>
</table>
Each LX550T FPGA can emulate approximately 4 million ASIC gates reasonably; however this is only an estimate. It is strongly recommended that you synthesize your actual ASIC design, mapping to FPGA technology to get an accurate FPGA utilization estimate.

3.2.2 Speed Grades

Xilinx FPGAs usually come in three speed grades. There is no rule of thumb to estimate which speed grade you will need to run your design at your target frequency. You will only know this once you have run a synthesis, with FPGA place-and-route, targeting the actual FPGA device that you will be using.

3.2.3 Upgrades

If you would like to install only some FPGAs when you order the board, and later add FPGAs or upgrade FPGAs to larger parts, this is possible; however you should request this before ordering the board because there are risks involved in attempting to add FPGAs to an existing board.

3.2.4 Small FPGAs

When FPGA G is installed with a “small” FPGA (an LX240T, LX365T, SX315T), some of the features on the board are not available; namely the SMA high speed serial connectors and the SEAM daughter card. The “large” FPGAs are the SX475T and the LX550T. When using either of these two FPGAs in the G position, there is no limitation on functionality.

This is because these three FPGAs have fewer I/O than physically exist on the 1759-pin BGA package. The on-board devices that the unused physical package pins are connected to cannot be used if the populated FPGA is one of these "small" FPGAs.

The diagram below shows the peripheral devices attached to FPGA G when a “small” FPGA is used.
3.2.5 Safe Handling of FPGAs

There are three easy ways to break the FPGAs.

1) Static electricity
Make sure you keep the board on a static controlled surface, and that you neutralize your body with that surface before handling the board. Especially sensitive are the FPGA I/Os. These are exposed on the daughter card headers and other connectors.

2) High Voltage
The FPGA I/O can only withstand voltages below and up to the VCCO power supply. When interfacing the board to some external I/O, make sure your I/O signals are driven at levels that do not exceed VCCO. If you do not know what VCCO is, then you probably should not be interfacing the board to some external I/O. Note that the maximum allowable VCCO on Virtex 6 is 2.5V. If you are interfacing with a 3.3V device, then you will need voltage translators.

3) Board warp
If the board undergoes mechanical stress, the FPGA pins (balls) can separate from the PCB and result in non-connected signals. The most common mechanical stress occurs when installing and removing connectors. Make sure that when installing a connector, you are supporting the connector from the opposite side, so that board warp does not absorb the force of the insertion.

3.3 Clock Resources

The board provides many clocking resources for general use. The sources and speeds of these clocks are designed to be as flexible as possible to accommodate a wide range of applications.

3.3.1 Clock pins on the FPGA

The Virtex 6 has many fewer "global clock" (GC) pins than previous FPGA families from Xilinx. Instead there are "Clock Capable" (CC) pins that have restrictions on how they are used. Only some of the "global clock" networks on the DN2076K10 are connected to "GC" pins on the FPGA. The rest are connected on "clock capable" pins. A "clock capable" pin might be a MRCC ("multiple region clock capable") or SRCC ("single-region clock capable"). You will have to consult a datasheet to tell the difference.

The use of each type of pin, MRCC, SRCC, GC result in different effects on logic timing.

Also, banks 10-18 have different timing than banks 20-38. All "global clock" pins on the DN2076K10 are connected to banks 20-38.

It would be difficult and misleading if I tried to explain how clocks worked internally in a Virtex 6, so you will need to consult the data book. We tried to connect them to make them as useful and flexible as possible.
3.3.2 Global Networks

The "global clocks" are the clocks that are provided to all 6 FPGAs, with low skew between the arrivals of the clock pulses at each FPGA. There are 7 such networks. USER_R, USER_L, USER_T, G0, G1, G2, and CLK_25

![Diagram of Global Networks]

Each of them is suitable for synchronous communication between FPGAs.

3.3.3 Clocks G0, G1, G2

The clocks G0, G1 and G2 are from a synthesizer that can produce any frequency from 2KHz to 700MHz with a 50ppm tolerance or better.

![Diagram of Clocks G0, G1, G2]

The synthesizer used is a high-performance, low jitter, high-precision clock generator chip, the Si5326. To change the clock frequency you can use the EMU software.
The clocks G0, G1 and G2 can also be set to come from the config FPGA. The config FPGA in turn, can be set to source this clock from the FPGAs. In this way, FPGAs can drive the frequency onto the global clock networks. This can be useful for controlled clocks and step-clocks. It can also be useful when a local clock for an FPGA needs to be delivered to all 7 FPGAs with zero delay.

G0 can be sourced from FPGA A or FPGA D. The signal that the FPGA should drive when this mode is used is called CLK_TO_SPARTAN_Ap and CLK_TO_SPARTAN_Dp respectively.

G1 can be sourced from FPGA B or FPGA E. The signal that the FPGA should drive when this mode is used is called CLK_TO_SPARTAN_Bp and CLK_TO_SPARTAN_Ep respectively.

G2 can be sourced from FPGA C or FPGA F. The signal that the FPGA should drive when this mode is used is called CLK_TO_SPARTAN_Cp and CLK_TO_SPARTAN_Fp respectively.

3.3.4 CLK_25
This clock network is fixed at 25Mhz. You cannot change it, try as you might.
3.3.5 USER_L

This clock has no frequency synthesizer, but can come from a variety of sources.

The following list are the available sources for the clock USER_L:

<table>
<thead>
<tr>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA A SRC</td>
<td>The FPGA A should drive the signal CLK_USER_LEFT_OUTAp/n differentially</td>
</tr>
<tr>
<td>FPGA B SRC</td>
<td>The FPGA B should drive the signal CLK_USER_LEFT_OUTBp/n differentially</td>
</tr>
<tr>
<td>FPGA D SRC</td>
<td>The FPGA D should drive the signal CLK_USER_LEFT_OUTDp/n differentially</td>
</tr>
<tr>
<td>SMA</td>
<td>The user should supply a clock single-ended into the SMA P36, located on the bottom right of the board. Voltages up to +2.5V are acceptable.</td>
</tr>
<tr>
<td>MGT</td>
<td>The clock will be the same frequency as the &quot;MGT&quot; clock. This has a dubious and unknown use.</td>
</tr>
</tbody>
</table>

3.3.6 USER_R

This clock also has no synthesizer, but can come from a variety of sources. The primary use is for generating a frequency from an FPGA and then using that new frequency across the board. This is just like USER_L, except there are different choices for the inputs.

<table>
<thead>
<tr>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA C SRC</td>
<td>The FPGA C should drive the signal CLK_USER_LEFT_OUTCp/n differentially</td>
</tr>
<tr>
<td>FPGA E SRC</td>
<td>The FPGA E should drive the signal CLK_USER_LEFT_OUTEp/n differentially</td>
</tr>
<tr>
<td>FPGA F SRC</td>
<td>The FPGA F should drive the signal CLK_USER_LEFT_OUTFp/n differentially</td>
</tr>
<tr>
<td>SMA</td>
<td>The user should supply a clock single-ended or differential into the SMAs J50 and/or J41, located near the big block of SMA connectors. Voltage levels up to +2.5V are acceptable.</td>
</tr>
</tbody>
</table>
3.3.7 USER_T
This clock has no frequency synthesizer, but can come from a variety of sources.

The source of this clock can be set to FPGA A, FPGA D, FPGA B, or FPGA G

3.3.8 Frequency-only networks
Frequency only networks are networks that are provided to all six FPGAs, but do not guarantee low-skew between the inputs to the FPGAs. These networks should not be used for fully-synchronous communication between FPGAs, at least not without phase-adjustment.

3.3.9 Local Networks
Local networks are networks that are only delivered to a single FPGA.

3.3.10 CLK_TO_*
Some FPGAs have signals that connect from one FPGA to another FPGA's global clock input pin. These signals are single ended and are called "clk_to_*" where * is either A, B, C, D, E or F. These can be used for forward a clock from one FPGA to another, without having to use "local routing" within the FPGA.

3.3.11 Spartan/TP
There is a test point connected to at least one clock input of each FPGA. There is no known use.

3.3.12 Daughter card Feedback
Each FPGA that has a daughter card connector also has a signal that loops back from an FPGA output to a clock input of that same FPGA. The routing length of this signal is the same as the routing length of the I/O signals to the daughter card. The purpose of this is so that it is possible to have a clock in the FPGA that is phase-aligned to a clock arriving at the daughter card. The signal is called "DC*_FEEDBACK_P/N"

3.3.13 DIMM Feedback
Similarly, each FPGA that has a DIMM interface has a signal that is looped back from an output of the FPGA to a clock input of the FPGA. The routing length of this feedback is equal to the routing length of the signals to the DIMM connector. In this way, it is possible to have a clock inside the FPGA that is phase aligned with the arrival of the clock at the DIMM. The signal is called "DIMM*_FB_P/N"
3.3.14 From SEAM connector
The SEAM daughter card provides four clock inputs delivered to the MGT (GTP, GTX, HTX) tiles of the connected FPGA. This clock is intended to be used for RocketIO communication with the SEAM interface, however it can be used in the FPGA for other types of logic.

3.3.15 NMB
The NMB interface includes one clock signal running at 50 or 100MHz. This clock is free-running, and is not configurable. It is received by each FPGA at the same frequency, however the phase relationship between the arrival of the clock at each FPGA is indeterminate.

3.3.16 Generating clocks from FPGAs
Clocks generated in an FPGA using MMCM, logic, or other resources can be outputted onto many of the global clock networks, as stated earlier. That means you can do all of your frequency generation in an FPGA, if so desired.

3.3.17 Clocking features not implemented
Four of the networks can be driven from the configuration FPGA. If you need some special feature, then we could potentially add it. For example, single-step clocks, or clocks from FPGA-to-FPGA.

3.4 NMB Bus
The NMB bus is the primary means you will use to get high quantities of data on and off the board. If you want to use the provided software (EMU) to push data to the board over USB, Ethernet and PCI Express, then you are required to interface to NMB in your FPGA design.
3.4.1 Protocol

You are expected to know nothing about the protocol of NMB and only interface to it using the provided HDL interface wrapper in your FPGA on one end, and in the EMU C++ code on the other end. However, the marketing material constantly makes reference to the inner workings of the underlying hardware and software, so I feel obligated to sort of describe it a little. This short section describes the implementation of the interface. I recommend you skip this "protocol" section. It is useless to know anything here.

There is a block diagram of the NMB bus architecture above. It is physically point-to-point from each FPGA to the configuration FPGA. Each point-to-point connection consists of a 40-pin signal wire, which are used as 20 LVDS pairs. These pairs are further divided into 10 signals in each direction, with 1 clock signals, 1 control signal and 8 data signals. The clock frequency of the interface is 1GHz (500 MHz clock with DDR capture). The theoretical throughput is therefore 1GB/sec in both directions simultaneously, to all 6 FPGAs simultaneously.

The protocol supports four channels, demand mode, bursts, interrupts, link detection, some FIFO flags, and more. The data to/from the FPGA is stored in buffers in the DRAM of the Marvell processor.

3.4.2 User Interface

An HDL module is provided in the support package here:
/FPGA_Reference_designs/common/NMB/nmb_user_interface.v

implementing the NMB FPGA module. More or less the nmb_user_interface.v provides a simple Address/DataIn/DataOut type interface. You should think of the interface as a memory space.

On the C++ side of the NMB there are simple functions like
emu_nmb_read(address, buffer, size)
that can be used to view this memory space. The code for this is found in the support package here
/Host_Software/emu/App/out_release/[release]/Software/emu/App/source/EMULIB/diniboard.h


3.4.3 Memory Spaces
For additional information, read the Emu Manual sections on NMB:
D:\Documentation\Manual\Emu_Manual.pdf

The memory space is 64-bit. Each address represents a single byte, however the data is required to be read and written in blocks of 32-bits. Addresses supplied to the interface must be divisible by 4. Therefore, the bottom 2 bits of the address space are unused and always zero.

Additionally, since there are no chip-selects on the NMB bus, it is necessary to pre-allocate address ranges for devices on the bus. On this board, there are seven devices, and the NMB address ranges that they are able to respond to on the bus are given here:

<table>
<thead>
<tr>
<th>Target</th>
<th>Starting Address</th>
<th>End Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA A</td>
<td>0x00000000_00000000</td>
<td>0x00FFFFFF_FFFFFFFF</td>
</tr>
<tr>
<td>FPGA B</td>
<td>0x01000000_00000000</td>
<td>0x01FFFFFF_FFFFFFFF</td>
</tr>
<tr>
<td>FPGA C</td>
<td>0x02000000_00000000</td>
<td>0x02FFFFFF_FFFFFFFF</td>
</tr>
<tr>
<td>FPGA D</td>
<td>0x03000000_00000000</td>
<td>0x03FFFFFF_FFFFFFFF</td>
</tr>
<tr>
<td>FPGA E</td>
<td>0x04000000_00000000</td>
<td>0x04FFFFFF_FFFFFFFF</td>
</tr>
<tr>
<td>FPGA F</td>
<td>0x05000000_00000000</td>
<td>0x05FFFFFF_FFFFFFFF</td>
</tr>
<tr>
<td>FPGA G</td>
<td>0x06000000_00000000</td>
<td>0x06FFFFFF_FFFFFFFF</td>
</tr>
</tbody>
</table>

3.4.4 Error Conditions
Occasionally error conditions may occur and the interface may return error codes. There is no defined way to distinguish these error codes from real data. The error codes are as follows:

<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xDEADBEE0_DEADBEE1</td>
<td>Read request to the config FPGA timed out.</td>
</tr>
<tr>
<td>0xDEADBEE2_DEADBEE3</td>
<td>Read request to the user design (BARs 1,2,4) timed out</td>
</tr>
<tr>
<td>0xDEADBEE4_DEADBEE4</td>
<td>Read request to config FPGA while the config interface is in reset. The config interface may be in reset because the config clock (Spartan FPGA -&gt; PCIe FPGA) is not being properly received or the BAR0 register for resetting the config interface is set high or the DMA engines are being cleared.</td>
</tr>
<tr>
<td>0xDEADBEE5_DEADBEE5</td>
<td>Read request to user space while the user interface is in reset. The config interface may be in reset because the BAR0 register for resetting the user interface is set high or the DMA engines are being cleared.</td>
</tr>
</tbody>
</table>
3.5 FPGA Interconnect

Most of the I/O on the FPGA are used to connect each FPGA to other FPGAs. Most interconnect is routed point-to-point between FPGAs, in a nearest neighbor topology. The exact topology is shown in the diagram below.

3.5.1 I/O protocol

The protocol for the use of the I/O is user defined. The VCCO pin of the FPGA on each bank that is used for interconnect is +2.5V. This means that LVCMOS25 and LVDS are both reasonable choices for the IOSTANDARD attribute.

The board features necessary to use terminated standards, such as LVDCI or SSTL are not provided on the board, and so DCI cannot be used for FPGA-to-FPGA communication. When using LVDS, you should use the DIFF_TERM attribute to terminate signals.

Since the "global clocks" on the board are delivered to each FPGA with low skew, any of the "global clocks" are suitable for use for clocking FPGA interconnect.

3.5.2 High-Speed interconnect

The interconnect between FPGAs are divided into "banks". A single bank on one FPGA always connected to a single bank on one other FPGA. This pairing up of banks allows the use of some of the high-speed features of the Virtex 6, such as BUFR clocking, and ISERDES and OSERDES. Every "bank" of interconnect contains at least one LVDS pair in each direction that goes to a "clock capable" pin on the FPGA. This pair can be used as a clock to input all of the bits on that bank. The net name of this pair ends in "_CC" in the schematic.
To achieve the highest switching rates on the interconnect banks, you must use the LVDS IOSTANDARD.

3.5.3  Signal net length report
The shortest FPGA-to-FPGA interconnect signal is 40mm long. The longest is 290mm long. This corresponds to a skew of about 2ns.

Within any single bus, the greatest skew is 75mm. This corresponds to 0.5 ns of skew.

3.6  SODIMM (DDR3) Connectors
For memory expansion, the DN2076K10 has three socket connectors connected to three of the user FPGAs. The sockets accept standard off-the-shelf DDR3 laptop memory. The interface and reference design is compatible with any density or organization of memory.

These sockets can also be used for types of memory other than DDR3 DRAM. Dini Group has a variety of modules that are compatible with the DN2076K10, including synchronous SRAM and others.
The SODIMM interface is also potentially usable as an expansion interface for custom daughter cards.

3.6.1 Memory Interface Generator
The provided reference design uses a memory controller that is based on the memory controller that is produced by the "Memory Interface Generator" (MIG), part of the Xilinx ISE software. However it has been modified. The modifications allow the use of dual-rank DIMMs, and also set some parameters automatically, such as RAS and CAS latencies, and total DIMM density.

Some of the signals that are connected between the SODIMM connector and the FPGA are not used by MIG. These include the SODIMM "NC" (no connect) pins, the upper two address pins, and the EVENTn pin.

Additionally, the "feedback" clock is not used by MIG, or the DDR3 controller provided by Dini Group.

If you want to use MIG you should use the HDL files produced by MIG, and use the UCF file provided by Dini Group, removing the unused signals.

3.6.2 IO Standards
The DIMM interface is voltage-selectable. When using DDR3 memory, it is suggested to use the 1.5V IO voltage. When using this voltage, signals to the SODIMM should be of the IO Standard SSTL15_T_DCI, DIFF_SSTL15, SSTL15, or LVCMOS15. The necessary board features to make SSTL work properly are provided. See the sample UCF files provided with the MainRef example design for details.

The EVENTn pin is also voltage selectable. It will be the same voltage as the rest of the DIMM. There are some LED signals on the FPGA that are connected to the DIMM bank. The IOSTANDARD attribute of these signals must be changes to match the voltage of the DIMM.

The IIC signals (SDA and SCL) are fixed at +2.5V and should use a 2.5V standard.
3.6.3 Voltage Selection

Off-the-shelf DDR3 DRAM always uses 1.5V core power and IO signaling levels. If you are using DRAM, then you would never need to change the voltage output of the DRAM interfaces. However, when using alternate memory modules from Dini Group, or when designing your own daughter cards, you may need to supply a different voltage to the SODIMM and to the attached pins of the FPGA.

Each SODIMM interface has an isolated regulator that allows you to independently select the voltage. There is a header next to each SODIMM where you can install jumpers to affect the output voltage of this regulator. To change the voltage, remove any jumpers currently installed on the header, and install a jumper next to the silkscreen text showing the voltage that you want. Be sure to probe the DIMM_VDD test point once you have completed the change.

3.6.4 Daughter cards

If you want to make your own custom daughter card for use in the SODIMM sockets, you are encouraged to download the complete schematic and layout files for one of our existing custom modules. These are available on our website with no restrictions on use.

The signals on the DN2076K10 circuit board are routed as 50-ohm impedance. Every signal connecting to the DIMM is length-matched, including the feedback clock.

Check this webpage to see existing DIMMs and to access the schematic and layout files for existing DIMMs:
http://www.dinigroup.com/new/DNSODMM204.php
3.6.5 Net Length Report
The length of all signals from the FPGA to the SODIMM are length and delay-matched.

3.7 Daughter Card Connectors
The primary means of interfacing to the FPGA with external IO are through the 400-pin MEG-Array expansion connectors. There are three of these high-speed, high-density connectors on the board, two connected to FPGA A and one connected to FPGA D. The FPGA connections to each of the three connectors are the same. The physical layout requirements of each of the three connectors are the same.

They are located on the top side of the board.
3.7.1 Electrical Spec

The part number of the connector part installed on the DN2076K10 is "84520102LF" from FCI. It is intended that the part number that will be used to connect to this board is the FCI 74390-101 part.

The part 84520102LF is called the "plug" and the 74390-101 is called the "receptacle"

The DN2076K10 is the "host" side of the interface. The mating card is called the "daughtercard"

All signals from the FPGA to the connector are length matched to each other with a minimum tolerance of 50ps on all Dini Group host boards. All Dini host boards route the FPGA I/O signals as 50-ohm transmission lines.

It is recommended that daughter cards provide a bypass capacitor between the pins B0_VCC0, B1_VCC0, B2_VCC0, B3_VCC0, B4_VCC0 and ground close to the connector pin on the daughter card.

Note that Virtex 6 is incompatible with +3.3V I/O signaling completely. +2.5V is the maximum supported I/O voltage. If you require +3.3V I/O, you must use voltage translation devices.

3.7.2 IO Electrical

The part number of the connector part installed on the DN2076K10 is "84520102LF" from FCI. It is intended that the part number that will be used to connect to this board is the FCI 74390-101 part.

The part 84520102LF is called the "plug" and the 74390-101 is called the "receptacle"

The DN2076K10 is the "host" side of the interface. The mating card is called the "daughtercard"

All signals from the FPGA to the connector are length matched to each other with a minimum tolerance of 50ps on all Dini Group host boards. All Dini host boards route the FPGA I/O signals as 50-ohm transmission lines.

It is recommended that daughter cards provide a bypass capacitor between the pins B0_VCC0, B1_VCC0, B2_VCC0, B3_VCC0, B4_VCC0 and ground close to the connector pin on the daughter card.

Note that Virtex 6 is incompatible with +3.3V I/O signaling completely. +2.5V is the maximum supported I/O voltage. If you require +3.3V I/O, you must use voltage translation devices.

3.7.2.1 _V pins

Any pin name that ends in the string "_V" is a "VREF" pin.
For some settings of the FPGA I/O attribute IOSTANDARD, you are required to supply a 1/2 VCCO voltage onto the "VREF" pins of the FPGA. If a daughtercard requires one of these IOSTANDARD settings, it must generate the VREF voltage, and drive it back on the _V pins of the daughtercard.

There are no-stuff capacitor locations on the host board attached to all "VREF" pins that can be populated with capacitors, if this is required by your electrical analysis. Note that installing these capacitors will negatively impact the switching speed of these signals when used as regular I/O.

3.7.2.2  _CC pins
Any pin whose name ends in "_CC" is a “clock capable” “CC” pin.

CC pins and _MRCC pins are connected to "CC" pins or "MRCC" pins on the host board's FPGA. You should see the Virtex 6 SelectIO user guide for the implications of this. But in general, these pins are suitable for I/O clocks driven from the daughtercard to the host FPGA.

The "CC" and “MRCC” pins are able to clock I/O in other FPGA banks in some cases. For this capability, there are requirements that the banks have certain physical relationships to each other on the silicon die of the FPGA device. During the assignment of FPGA banks to the daughtercard, no provisions were made to restrict the bank selection to make cross-bank clocking consistent from one daughtercard header to the next. It is recommended that if CC pins are required, that a separate copy of the clock is driven to each FPGA bank that requires it, and do not rely on "multi-clock" regions.

3.7.2.3  VRP and VRN Pins
On all FPGA banks that are connected to the daughtercard, VRP and VRN pins are correctly connected to allow DCI to be used with the daughtercard. Note that there is still the Virtex 6 I/O rule that only a single type of DCI may be used per bank of the FPGA. This requirement may limit the use of DCI on the daughter card. The requirement is too complicated to describe here, so you may need to run a test place-and-route of your design to determine whether your desired pin out is acceptable. See the appropriate Xilinx FPGA user guide for more information.

3.7.3  Reset Signal
The signal RSTn_3.3V_TOLERANT (connected to net DAUGHTERCARD_RESETn in the DN2076K10 schematic) is valid when +2.5V_LDO is above 0.7V. At all times, when the signal RSTn_3.3V_TOLERANT is valid, and has a voltage measuring below 0.7V, then all boards using this interface (host and daughtercard) must tri-state all I/O signals connected to the interface. Daughter cards that fail to tri-state signals when RSTn_3.3V_TOLERANT is asserted (below 0.7V) may risk damaging the DN2076K10 board. The DN2076K10 will weakly pull up this signal to +2.5V. A daughter card is free to also pull up this signal weakly to any voltage between 0.7V and 3.3V. If the daughtercard supplies the VCCO rails, then the reset signal should also disable the regulators, see the VCCO Power section below for details.
3.7.4 Power

The DN2076K10 supplies power to the daughtercard at two voltages, 12V and 3.3V. Each pin of the Meg connector can supply no more than 1A of current, so the effective power limit of the daughtercard is 2A x 12V + 3A x 3.3V = 33.9W.

It is strongly recommended that daughter cards provide a means of isolating (series resistor) their power net with the host board, and provide a means of bypassing the power input with an external power connector.

On other Dini Group boards, the pins C1 and H1 may be power pins. On the DN2076K10, these pins are no-connects.

The daughtercard should never be capable of supplying current back onto the host board on the +12V or +3.3V nets. This could potentially damage the host board.

3.7.5 VCCO Power

The FPGA I/O power pins are connected directly to the meg-array daughtercard interface. The intention of this design is for the daughtercard to drive the necessary I/O voltage back onto the host board. There are linear voltage regulators on the DN2076K10 that bias these power rails to 1.2V, however it is not recommended that you use these to power the daughtercard I/O. These regulators can supply up to 1A of current.

If you build a daughter card that drives current back to the host board, it must be able to supply the current not only for the daughter card I/O, but also for the I/O current requirements of the host board.
To respect the power sequencing requirements of Virtex 6 devices, the daughter card must not power the VCCO rails while RSTn_3.3V_TOLERANT is asserted (active low). This is typically achieved by connecting RSTn_3.3V_TOLERANT to the SHDNn pins of the voltage regulators.

If you forget to include voltage regulators for VCCO on the daughtercard, you can rework the DN2076K10 bias regulators to output your required voltage, so long as the combined current requirements of the FPGA banks and the daughtercard do not exceed 1A per bank. Note that the 5 voltage regulators on the host board are not set up to evenly share current loads, and that each must be individually load limited to 1A.

There is a row of test points next to each daughtercard header allowing the easy probing of the 5 voltage rails on the daughtercard interface.

This is the bias regulator. There is one for each of the 5 banks on each daughtercard interface.

### 3.7.6 Clock Pins

The clock pins E1, F1, E3, F3 are always connected to 2.5V I/O on the FPGA. They are connected to clock input pins near the center of the FPGA, making them suitable for sending a clock from the daughter card to the host FPGA. They are not externally terminated on the board, but you can use DIFF_TERM in the FPGA if you are using differential signaling.

In addition to these two clocks, there are variable-voltage clock inputs to the host FPGA. The signals, B0_P/N0_GCC_DN and B0_P/N8_GCC_BUS are located on bank 0. The input levels will be determined by the bank 0 voltage, and the IOSTANDARD settings of the I/O in the FPGA. There are external end-termination resistors on the host board, terminating to a voltage of VCCO/2. This termination scheme will result in a high current on the signal.
3.7.7  Net Lengths

All daughter card signals are matched between 60mm and 80mm. This corresponds to a maximum signal skew of 180ps, in addition to the skew caused by the FPGA device.

3.7.8  Physical Spec

Daughter cards are mounted on the "front" or "component" side of the DN2076K10 PCB. The board-to-board clearance between the daughter card and the host board is 14mm. Attention must be paid to be sure that newly designed daughter cards have appropriate clearance over components on the DN2076K10 “component” side. It is intended that daughter cards extend out away from the FPGA’s and off the edge of the DN2076K10 PCB, which relieves mechanical restrictions.

The "front" or "component" side of the daughter card is the face that points away from the host board. Therefore, the "back" or "solder" side of the daughter card faces toward the “front” or “component” side of the host board.

400-pin MEG daughter cards typically have four mounting holes with a fixed offset from the MEG connector. On the DN2076K10 PCB two of these mounting holes are present for each of the three MEG connectors. The other two hole positions are off the side of the DN2076K10 PCB. The diagram below shows the standard mounting hole positions on a MEG daughter card.
In the above diagram, a daughtercard is designed that has holes matching the position of the holes on the base board, so that a 14mm, M3 size standoff can be used to stabilize it onto the base board. The dimensions of the daughtercard above are guaranteed to physically fit onto any Dini Group host board. Boards larger than the above given dimensions may not physically fit onto your Dini Group host board. If you cannot fit your daughter card design in this form factor, then you must examine the physical dimensions of the target host board to determine the maximum dimensions available.

For a larger vertical clearance between the host and daughter cards, a vertical extender board is available. The diagram below shows the vertical clearance when used with an extender.
3.7.9 Insertion and removal

Due to the small dimensions of the very high speed Meg Array connector system, the pins on the plug and receptacle of the Meg Array connectors are very delicate. When plugging in a daughter card, make sure to align the daughter card first before pressing on the connector. Be absolutely certain that both the small and the large keys at the narrow ends of the Meg Array line up BEFORE applying pressure to mate the connectors!

Place it down flat, then press down gently.

Mating can be started from either end. Locate and match the connector’s A1 position marking [triangle] for both the Plug and Receptacle. (Markings are located on the long side of the housing.) Rough alignment is required prior to connector mating as misalignment of >0.8mm could damage connector contacts. Rough alignment of the connector is achieved through matching the Small alignment slot of the plug housing with the Small alignment key of the receptacle housing and the large alignment slot with the large alignment key. Both connector housings have generous lead-in around the perimeter and will allow the user to blind mate assemble the connectors. Align the two connectors by feel and when the receptacle keys start into the plug slots, push down on one end and then move force forward until the receptacle cover flange bottoms on the front face of the plug.

Like mating, a connector pair can be unmated by pulling them straight apart. However, it requires less effort to un-mate if the force is originated from one of the slot/key ends of the assembly.

(Reverse procedure from mating) Mating or un-mating of the connector by rolling in a direction perpendicular to alignment slots/keys may cause damage to the terminal contacts and is not recommended.

3.7.10 Clocking Methods

A wide variety of clocking topologies were considered when designing the MEG interface. There should be at least one clocking method possible that will meet your needs. A list of reasonable clocking methods are listed and diagramed below.
3.7.10.1 Manual phase alignment
You can use a PLL inside the FPGA to manually align the phase of a clock that you send from the FPGA to a MEG card.

3.7.10.2 Synchronous with daughter card
You may need to have a clock with edges that arrive simultaneously to the FPGA and to the MEG daughter card. Each FPGA with a MEG interface has a feedback path that can be used to align a clock going to the MEG with one going to the FPGA.

This allows a device to operate fully synchronously with the FPGA.

3.7.10.3 Forwarding a global synchronous clock
If you need a device on the MEG card to operate fully synchronously to all FPGAs on the DN2076K10, then you can use a PLL in the FPGA as shown in the diagram.
This circuit will delay a copy of a "global clock" to the MEG card such that the phase will arrive at the MEG card at the same time that it arrives at all of the FPGAs.

### 3.7.10.4 Source-Synchronous inputs

Since the signals to the daughter card are length-matched, you can rely on a tight timing relationship between the clock and data that you send from the daughter card to the FPGA. Since the FPGA has a zero-hold-time input, sending a clock from the MEG card whose rising edges are aligned with the data transitions on your data lines will result in reliable communication with the FPGA.
3.7.10.5 Source-synchronous outputs
You can repeat this setup in the opposite direction, as long as the hold time on the device on the MEG card is zero or negative.

3.7.10.6 Skewed Clocks
The I/O on the FPGA can be used on either rising or falling edges of a clock, so it is easy to invert the clock on the FPGA device, so that it operates 180 degrees out of phase with the daughter card. Or you can invert the clock as you output it to the daughter card. The maximum frequency of the interface when using this method is effectively reduced by half.
3.7.11 Incorrect Clocking Methods

The following are methods that don't work.

3.7.11.1 Hold time violation

The following diagram shows a method that potentially violates hold time on the device on the MEG card.
3.7.11.2 PLL cascade
When using PLLs in the FPGA, make sure that there is not another PLL anywhere in the feedback loop of the PLL or it will result in an unreliable phase output.

3.7.12 Compatibility with older Dini Daughter Boards
The "MEG" interface has been included on Dini Group board since 2005 with the Virtex 4 series of boards, however the features provided by the interface have changed since then. Any daughter card that meets the following criteria will be compatible with the DN2076K10:
- Designed for use with a Dini Group Xilinx FPGA board (not Altera)
- Uses only a single voltage for the whole daughter card
- Does not use I/O signal levels above +2.5V
- Uses the "reset" signal as an input only
- Does not require the use of the +5.0V power pin
- Uses no differential signals

All Virtex 6 boards provide the same features on all MEG interfaces, so any daughter card designed for use on a Virtex 6 board will work on any MEG connector on any Virtex 6 board that Dini Group sells.

3.7.13 How to make a daughter card
It is recommended that you start with one of the Dini Group daughter cards as a design template. The ORCAD schematic, and layout files are all provided on the Dini Group website for several daughter cards.
3.7.13.1 Ensuring backward compatibility

In order to build a daughter card that is compatible with all Dini Group host boards, you should use the following guidelines:
1) Use the standard 2.75" x 4.25" form factor.
2) Use the same I/O voltage for all I/O pins on the daughter card
3) Supply I/O voltage from the daughter card on all 6 VCCO pins
4) Do not use CC pins
5) Only use differential signaling standards for clocks that drive to the base board
6) Do not use the +5.0V power pin, if one is present.
7) If VREF is needed, it must be driven onto all VREF pins of the entire MEG interface

3.8 FPGA CONFIGURATION

3.8.1 Select map

High speed select map configuration is controlled by the ConfigFPGA and Marvell CPU. FPGAs can be configured over Ethernet, USB, PCIe, from USB Flash Drives, external SATA hard drives, from onboard NAND Flash on the Marvell Processor, and more. See the EMU Manual for more information on these Configuration methods.

3.8.2 JTAG

A standard JTAG header is provided to which a Xilinx Platform USB JTAG cable or other JTAG cable can be connected. All seven FPGAs are on a single JTAG chain and can be configured using the Xilinx Impact software. This JTAG port can also be used for JTAG-based FPGA debug.
3.9 Marvell CPU

The clock networks, off-board I/O, NMB and configuration is controlled by a Marvell ARM CPU. This section describes the hardware attached to the Marvell processor. For information about programming the Marvell and the software running on the Marvell, see the software section of this manual, and the Emu User Manual.

![Diagram of Marvell CPU connections]

The primary purpose of the Marvell processor is to pump data from a host interface (PCI Express, Gigabit Ethernet, SATA, Flash, or USB) to and from the user's design in the FPGA.

Additionally, these host interfaces can also be used to control the settings of the board's resources, like clock synthesizers, FPGA configuration, and logic resets.

The software that comes pre-loaded on the Marvel processor is running the Linux operating system. For this section, it is assumed that you have a working understanding of Linux.

3.9.1 RS232

The console output of Linux is directed to an RS232 port. In order to connect a terminal to this port, you will need to set the terminal settings to 19200 baud, 8bit, no parity, no flow control, 1 stop bit.
The Linux console is connected to a shell, so you can interact with the system, however the main purpose of the console is to receive kernel debugging messages, so it is recommended that you instead use a telnet terminal, as this shell will be much less chatty.

In order to interact with the boot loader, or see output from the boot process, you must use the RS232 console, you cannot use telnet for these purposes.

3.9.2 PCI Express

The Marvell device natively contains a PCI Express device. The PCI express pins of the Marvell processor are connected directly to an iPass cable connector on the DN2076K10 board. The provided PCIe cable and adapter card can be used to connect to a host PC.

The device appears as three 1MB memory regions to the host machine. You are expected to use the Emu software and drivers (source code provided) to interface with PCI Express. There is no description available of the function of DN2076K10 as a PCI Express device.
3.9.3 SATA

The Marvell device contains natively two SATA host ports. In the pre-installed software these ports are managed by Linux, and you are not intended to directly interact with SATA hardware. Instead, when a device is installed, it is automatically mounted as a storage device with a filesystem.

From that point, you should use Linux scripts and programs to interact with the mounted filesystem.

3.9.4 DEV Bus: NAND

The root file system of the Linux installation is contained on the NAND Flash device connected to the Marvell's "device bus". The NAND is 256MB in size, with the first 100MB already assigned to the Linux installation.

3.9.5 DEV Bus: FPGA

The Marvell "device bus" is also attached to the "config FPGA". This connection is unused.

3.9.6 USB

The Marvell device natively provides a USB host and peripheral device. The two "host" devices are managed by Linux. You are not expected to interact directly with the host ports. Instead, when a device is detected, Linux automatically mounts the devices as storage devices with filesystems. You should write linux scripts and programs to interact with the mounted filesystems.
The single "device" interface is intended to be used as a connection to a host PC that will be used with the provided "EMU" software. You are expected to use the EMU controller library to interact with this device.

3.9.7 IIC Bus: Temperature Sensors

The temperature sensors for the user FPGAs, for the Marvell processor and for the config FPGA are connected to the Marvell’s two-wire serial interface. The installed software will poll these IIC interfaces and measure the temperature of the FPGAs. If the FPGAs are not within a specified temperature range, the software will automatically clear the overheated FPGA both to alert the user to the problem, and to prevent damage to the FPGAs.

3.9.8 ICE

There is an interface for running a hardware debugger on the Marvell processor. It is not expected that anyone will use this interface, and no header is factory installed, so details are omitted here.

3.9.9 SDRAM

The Marvell environment has 1GB of DRAM managed by Linux.
3.9.10 SPI Bus: Flash
The linux kernel and uboot bootloader are contained on a SPI flash device. The marvell boots by running instructions from address 0x0 of the SPI flash device. There is a 4-pin SPI programming header attached to this SPI flash. In order to program the flash device, the Marvell must be continually held in reset to prevent interference with the SPI programming process.

3.9.11 Ethernet
The Marvell Processor natively contains three gigabit Ethernet ports. Only one of these three ports are enabled. The ethernet port is managed by linux. You are expected to use standard Linux programming APIs to access these ports from the Marvell side of the link, and to write your own software for the host side of the link. The provided Emu software can be used to communicate with the board over ethernet, and you can use the communication framework provided to interact with the board from your own application.

3.9.12 Multi-CPU
The Marvell Processor has two CPUs. The first, CPU0 is used by the Linux operating system. Since Linux does not support symmetric multi-processing, the second CPU, CPU1 must be operated in un-hosted mode, in its own area of DRAM, without accessing devices. I/O must be accomplished through CPU0 under Linux.

The use of PCI Express and the DMA engine in the configuration FPGA is possible. Interrupts may also be used by CPU1. The second CPU is disabled by default.

3.10 Config FPGA
The "eighth" FPGA on the board, the config FPGA (FPGA Q) is not really intended for the user. It is a "cleanup" FPGA that controls all the clock circuits on the board, configures the other FPGAs, and multiplexes the NMB bus from the Marvell CPU to the user FPGAs. You don't need to know anything about it or how it works. You are encouraged to skip this section.
3.10.1 PCI Express
The config FPGA is connected to the Marvell processor through a PCI Express interface. The config FPGA is a PCI Express endpoint, and the Marvell acts as a root port.

3.10.2 Configuring the Config FPGA
The configuration signals of the config FPGA are connected to the Marvell CPU. The Marvell CPU gets the .bit file for configuring the "config FPGA" off of the NAND flash. The "config FPGA" also has a SPI flash attached to the configuration signals that can be used to store configuration data, however it is unused in the default setup.

3.10.3 RS232
The "config" FPGA passes through the user's RS232 signals to the RS232 buffer.

3.10.4 Count Clocks
All of the boards "global clocks" are connected to the config FPGA. The config FPGA measures the frequency so that software can report it.

3.10.5 JTAG
The config FPGA has a dedicated JTAG chain and connector that can be used with the IMPACT program from Xilinx. It has no purpose.

3.10.6 Device Bus
The "config FPGA" has a "device bus" interface connected between it and the Marvell CPU. In this way the CPU can access the config FPGA as a memory-mapped device. This interface serves no purpose on this board.

3.10.7 Blink LEDs
The config FPGA is connected to 6 green LEDs. It blinks these LEDs incessantly. The LEDs have no purpose or meaning.

3.10.8 Controlling Clocks
The control signals for the "global clock" synthesizer chips, and the multiplexer chips are connected to the config FPGA. Software is able to set these control signals.

3.10.9 Clock MUX
The config FPGA serves as one stage of multiplexers for the clocking network. On the schematic you may see that clocks G0, G1 and G2 come from the config FPGA. The config FPGA drives out clock signals coming from the user FPGAs on the "TO_SPARTAN" wires, depending on software settings.

3.10.10 Configuring the FPGAs
The configuration FPGA is connected to the "selectmap" configuration bus of the six user FPGAs. It uses this bus to configure and read back the configuration data of the user FPGAs.
3.10.11 Marvel to NMB Bridge

The NMB interface connects each user FPGA to the config FPGA. The data that goes to and from the FPGA winds up in the Marvell CPU’s DRAM. Since the config FPGA has a PCI Express link to the Marvell Processor, it is able to directly manipulate memory in the Marvell's DRAM. The config FPGA has a DMA controller inside of it that pushes data directly from the FPGAs to the Marvell DRAM.

There is a bunch of detail about how to use this DMA controller that you don't need to know documented in the "PCIe DMA (ConfigFPGA design) User Manual" PDF document. You should not read this document.

3.11 RS232

There is a pair of signals (RX and TX) for RS232 "Serial" communication to the FPGAs.
RS232 is shared by all 7 FPGAs. Only one FPGA should drive the TX line at a time. The dedicated RS232 lines go only to FPGAs A, B and E and should not be used. Lines from the YMB shared bus are used instead, because they go to all 7 FPGAs. See the MainRef design and UCF file to figure out what pins to use.

### 3.12 USER LEDS

The FPGAs are all connected to LEDs which have the ON and OFF capability. Each FPGA has a unique number of LEDs connected to it.

There is not a lot to say about LEDs.
If you pulse a signal to them, then they will be varying levels of brightness. They are sort of yellow in color. Other yellow items include sunflowers and fire.

### 3.13 SPI FLASH

Two of the user FPGAs, namely FPGA B and F, have serial flash devices attached to them for general usage.

The interface is SPI. The SPI signals are at +2.5V levels. The datasheet for the SPI Flash devices can be found in the “Datasheets” folder in the board support package.

### 3.14 USER TEST POINTS

Each FPGA, except FPGA G, has a CLK_TP test point connected to an SRCC clock capable pin. On FPGAs A, B, and D these are differential test points. On FPGAs C, E, and F these are single ended SMA connectors. External termination resistors are provided. The CLK_TP lines can be used as clock inputs or general IO- just be aware of if/how the termination resistors affect your usage and adjust them if necessary.
3.15 “Config Mictor” Connector

There is a Mictor connector on the top side of the board. Its primary purpose is to provide a method by which daughtercard FPGAs can be programmed from the host software. A mictor cable connects to the daughtercard and contains all of the required SELECTMAP signals including chip selects that are supported from the EMU software’s “FPGA->Configure” menu.

This mictor can also be used for logic analyzers. The only signals that you can actually drive from an FPGA are SELECTMAP_HIGH[8] – SELECTMAP_HIGH[15]. The rest of them cannot be used because they are driven from the config FPGA at all times.

3.16 USER SATA

FPGA G has two SATA connectors attached to its MGTs. These can be used to make a SATA interface. One of the connectors can only be used when the FPGA is acting as a device. One of the connectors can only be used when the FPGA is acting as a controller. It is impossible to emulate a raid controller with this setup.

There is a dedicated oscillator connected to an appropriate REFCLK input. The frequency select pins of the oscillator are connected to the FPGA so that the user can select an appropriate REFCLK frequency for SATA I or SATA II.
3.17 SFP and Gigabit Ethernet

FPGA G is also attached directly to 1 SFP connector. There are 4 other SFP connectors next to it that are discussed below. The directly connected SFP connector is useful for Fibrechannel, Ethernet, and other interfaces. Since the FPGA is only capable of serial speeds up to 6Gbs, SFP+ modules probably can’t be used here.
The SFP connector also has some low-speed sideband signals which also attach to the FPGA. These signals are fixed at +2.5V. A dedicated oscillator is provided that has a random variety pack of frequencies. The oscillator is shared with the SATA ports described earlier. You can select between these frequencies using the frequency select pins of the oscillator, which are attached to the FPGA. If the frequencies provided in the variety pack aren't appropriate you have one of the following options:

1. Replace the oscillator with a better one. We will help you do this if you want.
2. Use the CLK_MGT network. This network can be driven from a frequency synthesizer that can hit a wide variety of frequencies. The disadvantage is that then the entire CLK_MGT network must run at this frequency, which you may want to be using for other functions.

The following photo shows the SFP connectors but the top one has a funny cage and the bottom one is missing the cage entirely. You’d think we would have a better photo but we don’t. The bottom one, which is missing the cage in this photo, is the one that connects directly to FPGA G.
3.17.1 IIC
There are sideband signals on the SFP. They are attached to the FPGA. You are expected to figure out what to do with them.

3.18 10 Gigabit Ethernet
The DN2076K10 can be used to directly interface to 10Gb/sec Ethernet. There are 4 SFP+ module sockets that can each be used for 10Gb/sec optical modules, or copper direct-attach modules.

The Xilinx GTX serial transceivers on the FPGA G are not fast enough to operate at 10Gb/sec, so there is a QT2225 4-to-1 serializer for each SFP+ module. Each 10Gb channel will require four serial streams to the FPGA operating at 3.125 Gb/s each. The operation of the QT2225 is super-complex and you will probably need to read the datasheet to use it properly. The EPROMs connected to the QT2225 have already been programmed with what might be a reasonable configuration for operation at 10 Gb/second. The devices can also be configured with new settings.

3.19 GTX Expansion (SEAM) HEADER
Brand new in Virtex 6 we have added a new type of header, the "GTX Expansion Header Interface". This header uses the SEARAY connector (or SEAM).

There is one of these on the DN2076K10, attached to FPGA G.
We have several off-the-shelf SEAM daughter cards, visit [www.dinigroup.com](http://www.dinigroup.com) to see what is available, or send a request to sales@dinigroup.com to inquire about custom designs. Note there is no mechanical standard for these daughtercards, so before purchasing an off-the-shelf SEAM daughtercard from Dini Group, confirm with sales@dinigroup.com that the card will be able to plug into the DN2076K10 board.

Each SEAM provides 8 channels of high-speed serial data to the FPGA. There are four REFCLK signals from the SEAM connector to the FPGA, making four independent interfaces on one SEAM connection feasible. Additionally, for the purpose of control, there are 16 "low speed" I/O that go to the FPGA "regular" I/O pins.

Here is a photo of the SEAM connector, which is on the top side of the board.
See the schematic for additional details on the connections between the FPGA and the SEAM card. The connection is electrically straight-through. The low speed I/O is fixed at +2.5V signaling levels. Three voltages are provided to the SEAM card, +3.3V, +12V and "VCCO". The VCCO supply is fixed at +2.5V, however the daughter card designer should keep in mind that future Dini Boards may choose to supply a different (probably lower) fixed voltage here, such as +1.8V. The below diagram shows the SEAM pin out.

The pin out is designed for extreme high speed.

### 3.19.1 Mechanical

The connector on the DN2076K10 is a Samtec SEAM-20-03.5-S-08-2-A. The mating connector for daughter card use is Samtec SEAF-20-06.5-S-08-2-A. This makes a total stacking height of 10mm. The connection scheme, like all sane connection schemes, is 1-1, 2-2, 3-3, 4-4, 5-5, 6-6, etc.
3.20 ENCRYPTION

The Virtex 6 FPGA allows the use of encrypted bit files. You would want to encrypt a bit file if you want some person to pay you for the use of your IP, or to not steal it and reverse engineer it.

In order to support encryption, we have provided the necessary battery on the board. This battery supplies voltage to the VBATT pin of the FPGA, even when the board is off. It also provides power to the VBAT pin of the real time clock on the board. Note that the real time clock does not work on revision 01 of the circuit board.

The above photo shows where one might install a battery on the DN2076K10. The DN2076K10 comes with a battery installed that will last for quite some time.

Use a CR1220 (or BR1220) type lithium battery in the socket. The difference between CR series and BR series in the lithium range is CR batteries provide higher voltage during the first half of discharge and better overall current capability. In order to change the battery without the loss of the encryption key you can attempt one of the following feats of daring.

1. Switch the battery out while the board is powered on.
2. Attach some sort of external power supply to test point TP41.
3. Switch the battery out in less than 10 seconds.
4. Reprogram the encryption key after changing the battery.
3.21 JTAG

The FPGA JTAG chain of the Virtex 6 FPGAs is available for your use. It isn't used by any other circuit on the board. It just connects right up to a header that the Xilinx programmer cable connects to. The Xilinx Impact software, or other JTAG software can be used. JTAG based debug programs can also be used.

The order of the FPGAs on the chain is A, B, C, D, E, F, G. The configuration FPGA is not attached to the same chain as the user FPGAs. Note that if you have ordered the board with fewer than seven FPGAs, the JTAG chain will still operate normally but will not contain the missing FPGAs. So if you have only FPGA C, E and F on the board, then the chain will appear to only have three FPGAs on it in the order C, E, F.
3.22 MECHANICAL

If you wish to build a plate of aluminum with M3 holes pre-drilled to act as a base plate for the board, then the diagram below will be very helpful to you.

The holes, except for the "SEAM mounting" holes are 3mm. They are all attached to "ground" on the board. The back side clearance is 14mm. The front side clearance is 1.75". The board thickness is 93 mils (2.3622mm).
### 3.23 POWER

Power is measured in watts, or joules per second.

#### 3.23.1 Power Headers

There are three ingress points for power on the board. They are all compatible with the so-called "PCI Express graphics power" connector. This connector has 6-pins, and is increasingly common on standard ATX power supplies. There are also some 8-pin versions of "PCI Express graphics power" connectors on some power supplies. These can be plugged into the DN2076K10 as well, with the extra two pins dangling uselessly off to the side. All three power connectors must be installed on the board for it to operate properly. The current pulled from each connector can exceed 15A (at 12V) in extreme designs. We recommend using at least a 750W ATX power supply which can supply at least 200W per 12V supply. An appropriate ATX supply is shipped with each board.

The three 12V power connectors power three distinct 12V rails on the board. There are jumper points on the board that can connect the three 12V rails of the board together, in case you really want to run the board with one cable, which is not recommended.

Here is a summary of the three rails. The next section shows this in chart format.

<table>
<thead>
<tr>
<th>RAIL</th>
<th>PRIMARY FEATURES</th>
</tr>
</thead>
<tbody>
<tr>
<td>+12V_T (J63)</td>
<td>VCCINT to FPGA A,B,D, and E, and +12V to MegArray D</td>
</tr>
<tr>
<td>+12V_R (J77)</td>
<td>VCCINT to FPGA C,F, and G, DIMMD_VDD, fans, +1.2V rail, and +12V to Searay G</td>
</tr>
<tr>
<td>+12V (J3):</td>
<td>VCCINT to Config FPGA, DIMMB_VDD, DIMMG_VDD, +12V to MegArray AA and AB, +2.5V and +3.3V rails, +5.0V USB, Marvell CPU power, fans</td>
</tr>
</tbody>
</table>

From 300W PCI Express Specification

PCIe Power Cable (4A/pin) accepts 6-pin or 8-pin connector

place LED near PCI Express power connector

So you could use only one cable
place kind of far apart from each other (1 inch or more)
The above is a schematic clippings show the three power connectors, the power rails they drive, and the jumper points where they could be connected together.

This is what the the power connectors look like.
3.23.2 Distribution Chart

Power for the board is all derived from 12V. Lower voltages are generated either directly by converting 12V DC input, or by down converting another voltage that was down converted from 12V.

Above is a chart that shows what each +12V rail powers.

3.24 HEAT

Without proper cooling this board gets very hot. Other hot items include the sun and boiling water.

3.24.1 Total thermal performance

The heat sinks that come installed on the board are each capable of dissipating one Watt for every degree (in Celsius) that the FPGA under it raises above the ambient temperature. For example, if you are using the board in a room at 25 degrees, and you configure one of the FPGAs with a design that uses 31 Watts, the core temperature of the FPGA will rise by 31 degrees, for a total temperature of 56 degrees.

You can use the Xpower tool in Xilinx to determine how much power your FPGA design uses. The amount of power that your FPGA uses may limit the maximum ambient temperature that your board
can operate in. The FPGA is not guaranteed to function properly at core temperatures above 80 degrees C.

For example, if you put the board in a computer case with the lid closed, and the temperature inside the computer case is 65 degrees C, then the maximum power that your FPGA design can use and still operate reliably is 15 Watts. Note that ambient (air near the board) temperature measurements must be taken at full power.

### 3.24.2 FANS

The fans that are sitting on top of the heat sink are plugged into the board for power. They have a tachometer. The frequency of operation, in revolutions per minute, can be read from the EMU host software.

![Cooling Fan Circuit](image)

If your fans start to make noise they need to be replaced. We will send you new ones.

### 3.24.3 Temperature Sensors

Each FPGA has a temperature sensor attached to it to measure the temperature of the FPGA ("die temperature"). Since correct operation of the FPGA is not guaranteed by Xilinx when the core temperature increases above 80C degrees, we helpfully reset the FPGA for you when the temperature hits 80C. This also prevents the temperature from increasing further and reaching the point at which permanent damage can result. This behavior can be changed if you want, but you'll have to email support@dinigroup.com and ask us how.

### 3.25 RESET

There is a board-wide reset circuit called "SYS_RESET". Its purpose is as follows:

1) Cause power supplies to come up in a particular order

2) Cause each device on the board to get a reset pulse after power is applied as required by the device datasheet with the minimum pulse width specified in the datasheet
3) Prevent the user from using the board if any power supply is off by even a little bit

4) Allow the user a way to reset the board to a repeatable state without having to power the board down and back up.

### 3.25.1 Particular order

The power supplies are allowed to supply voltage in a particular order.

![Diagram of power supply order]

### 3.25.2 Power supply failure detection

There is a circuit on the board to detect when any of the voltages on the board falls below some minimum voltage.

If this happens, the result is like holding down the "SYS RESET" button. A red LED comes on, and the board won't work at all.

### 3.25.3 Reset Button

The "SYS_RESET" button asserts the same signal that the power fail circuit does. It resets the entire board, similar to what a power-off then power-on cycle does.

The SYS_RESET button does not cause the power supplies to power down or the power-up sequence to be repeated.
3.25.4 User Reset

There is another reset button on the board called the "USER RESET" button. It is located as shown below.

This button has nothing to do with the power monitors or power on. All it does is assert the USER_RESETn signal to the FPGAs. It can be used as a general-purpose pushbutton by the user in the FPGA.

The USER_RESETn signal to the FPGA is also automatically asserted by the configuration FPGA while it is configuring FPGAs. After an FPGA is configured, the config FPGA will de-assert the USER_RESETn signal. For this reason it is useful for the purpose of resetting your logic.

Additionally, software can assert this signal. The Emu software does so when the “FPGA->Reset” menu option is chosen.

3.26 SYSTEM MONITOR

Each Virtex 6 FPGA has an internal block called the "system monitor". The system monitor is enabled on the DN2076K10, however none of the recommended power supply filtering for the A-to-D converter are provided, so there may be unknown amounts of error caused in the A-to-D because
of this. The VP and VN (analog) inputs of the FPGA are connected to an 0402 package 33 Ohm resistor. You will have to connect with wires to the board in order to use them.

You can also read back the voltages on these pins, the FPGA temperature, and the VCCINT and VCCAUX voltages using IMACT and the JTAG chain.

### 3.27 LED REFERENCE LIST

There are a lot of LEDs on the board. This list describes them. In general, you can search for the text in the “Meaning” column, or for the reference designator, in the schematic to gain more knowledge about what makes them turn on and off.

<table>
<thead>
<tr>
<th>REF. DES.</th>
<th>COLOR</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS1</td>
<td>RED</td>
<td>FAIL_3.3V</td>
</tr>
<tr>
<td>DS2</td>
<td>RED</td>
<td>FAIL_2.5V</td>
</tr>
<tr>
<td>DS3</td>
<td>RED</td>
<td>FAIL_1.0V_Q</td>
</tr>
<tr>
<td>DS4</td>
<td>RED</td>
<td>12V_WARN_L</td>
</tr>
<tr>
<td>DS5</td>
<td>RED</td>
<td>FAIL_1.1V_CPU</td>
</tr>
<tr>
<td>DS6</td>
<td>RED</td>
<td>FAIL_1.0V_CPU</td>
</tr>
<tr>
<td>DS7</td>
<td>RED</td>
<td>RST_PORN_LED</td>
</tr>
<tr>
<td>DS8</td>
<td>RED</td>
<td>FAIL_1.8V</td>
</tr>
<tr>
<td>DS9</td>
<td>RED</td>
<td>FLASH_BUSY</td>
</tr>
<tr>
<td>DS10</td>
<td>RED</td>
<td>RST_CPU_MPP13N</td>
</tr>
<tr>
<td>DS11</td>
<td>GREEN</td>
<td>SATA1_ACT</td>
</tr>
<tr>
<td>DS12</td>
<td>GREEN</td>
<td>SATA0_ACT</td>
</tr>
<tr>
<td>DS13</td>
<td>RED</td>
<td>USB0_OCN</td>
</tr>
<tr>
<td>DS14</td>
<td>RED</td>
<td>FAIL_5.0V</td>
</tr>
<tr>
<td>DS15</td>
<td>BLUE</td>
<td>CFPGA_DONE</td>
</tr>
<tr>
<td>DS16</td>
<td>GREEN</td>
<td>LEDCF_0</td>
</tr>
<tr>
<td>DS17</td>
<td>GREEN</td>
<td>LEDCF_1</td>
</tr>
<tr>
<td>DS18</td>
<td>GREEN</td>
<td>LEDCF_2</td>
</tr>
<tr>
<td>DS19</td>
<td>GREEN</td>
<td>LEDCF_3</td>
</tr>
<tr>
<td>DS20</td>
<td>GREEN</td>
<td>LEDCF_4</td>
</tr>
<tr>
<td>DS21</td>
<td>GREEN</td>
<td>LEDCF_5</td>
</tr>
<tr>
<td>DS22</td>
<td>RED</td>
<td>FRONT_PANEL_D1</td>
</tr>
<tr>
<td>DS23</td>
<td>YELLOW</td>
<td>LED_B0</td>
</tr>
<tr>
<td>DS24</td>
<td>YELLOW</td>
<td>LED_B1</td>
</tr>
<tr>
<td>DS25</td>
<td>YELLOW</td>
<td>LED_B2</td>
</tr>
<tr>
<td>DS26</td>
<td>YELLOW</td>
<td>LED_B3</td>
</tr>
<tr>
<td>DS27</td>
<td>YELLOW</td>
<td>LED_B4</td>
</tr>
<tr>
<td>DS28</td>
<td>YELLOW</td>
<td>LED_C0</td>
</tr>
<tr>
<td>DS29</td>
<td>YELLOW</td>
<td>LED_C1</td>
</tr>
<tr>
<td>DS30</td>
<td>YELLOW</td>
<td>LED_C2</td>
</tr>
<tr>
<td>DS31</td>
<td>YELLOW</td>
<td>LED_C3</td>
</tr>
<tr>
<td>DS32</td>
<td>YELLOW</td>
<td>LED_C4</td>
</tr>
<tr>
<td>DS33</td>
<td>YELLOW</td>
<td>LED_C5</td>
</tr>
<tr>
<td>DS34</td>
<td>YELLOW</td>
<td>LED_C6</td>
</tr>
<tr>
<td>DS35</td>
<td>YELLOW</td>
<td>LED_C7</td>
</tr>
<tr>
<td>DS36</td>
<td>GREEN</td>
<td>FRONT_PANEL_D2</td>
</tr>
<tr>
<td>DS37</td>
<td>YELLOW</td>
<td>LED_B9</td>
</tr>
<tr>
<td>DS38</td>
<td>YELLOW</td>
<td>LED_B8</td>
</tr>
<tr>
<td>DS39</td>
<td>YELLOW</td>
<td>LED_B7</td>
</tr>
<tr>
<td>DS40</td>
<td>YELLOW</td>
<td>LED_B6</td>
</tr>
<tr>
<td>DS41</td>
<td>BLUE</td>
<td>CONFIG_DONE_A</td>
</tr>
<tr>
<td>DS42</td>
<td>GREEN</td>
<td>PCIE_CPU_PWRR</td>
</tr>
<tr>
<td>DS43</td>
<td>RED</td>
<td>SYNTH_LOL_G1</td>
</tr>
<tr>
<td>DS44</td>
<td>RED</td>
<td>SYNTH_LOL_G2</td>
</tr>
<tr>
<td>DS45</td>
<td>BLUE</td>
<td>CONFIG_DONE_B</td>
</tr>
<tr>
<td>DS46</td>
<td>BLUE</td>
<td>CONFIG_DONE_C</td>
</tr>
<tr>
<td>DS47</td>
<td>YELLOW</td>
<td>LED_A0</td>
</tr>
<tr>
<td>DS48</td>
<td>YELLOW</td>
<td>LED_A1</td>
</tr>
<tr>
<td>DS49</td>
<td>RED</td>
<td>FAIL_1.0V_B</td>
</tr>
<tr>
<td>DS50</td>
<td>RED</td>
<td>FAIL_1.0V_C</td>
</tr>
<tr>
<td>DS51</td>
<td>RED</td>
<td>FAIL_1.0V_A</td>
</tr>
<tr>
<td>DS52</td>
<td>RED</td>
<td>FAIL_1.0V_G</td>
</tr>
<tr>
<td>DS53</td>
<td>RED</td>
<td>FAIL_1.0V_D</td>
</tr>
<tr>
<td>DS54</td>
<td>RED</td>
<td>FAIL_1.0V_E</td>
</tr>
<tr>
<td>DS55</td>
<td>RED</td>
<td>FAIL_1.0V_F</td>
</tr>
<tr>
<td>DS56</td>
<td>BLUE</td>
<td>CONFIG_DONE_E</td>
</tr>
<tr>
<td>DS57</td>
<td>YELLOW</td>
<td>LED_E6</td>
</tr>
<tr>
<td>DS58</td>
<td>YELLOW</td>
<td>LED_E5</td>
</tr>
<tr>
<td>DS59</td>
<td>YELLOW</td>
<td>LED_E4</td>
</tr>
<tr>
<td>DS60</td>
<td>YELLOW</td>
<td>LED_E3</td>
</tr>
<tr>
<td>DS61</td>
<td>YELLOW</td>
<td>LED_E2</td>
</tr>
<tr>
<td>DS62</td>
<td>YELLOW</td>
<td>LED_E1</td>
</tr>
<tr>
<td>DS63</td>
<td>YELLOW</td>
<td>LED_E0</td>
</tr>
<tr>
<td>DS64</td>
<td>BLUE</td>
<td>CONFIG_DONE_G</td>
</tr>
<tr>
<td>DS65</td>
<td>DS72</td>
<td>DS74</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>BLUE</td>
<td>BLUE</td>
<td>YELLOW</td>
</tr>
<tr>
<td>CONFIG_DONE_D</td>
<td>LED_F0</td>
<td>LED_F1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DS66</th>
<th>DS67</th>
<th>DS68</th>
<th>DS69</th>
<th>DS70</th>
<th>DS71</th>
<th>DS72</th>
<th>DS73</th>
<th>DS74</th>
<th>DS75</th>
<th>DS76</th>
<th>DS77</th>
<th>DS78</th>
<th>DS79</th>
</tr>
</thead>
<tbody>
<tr>
<td>YELLOW</td>
<td>YELLOW</td>
<td>YELLOW</td>
<td>YELLOW</td>
<td>RED</td>
<td>YELLOW</td>
<td>YELLOW</td>
<td>RED</td>
<td>YELLOW</td>
<td>YELLOW</td>
<td>YELLOW</td>
<td>YELLOW</td>
<td>YELLOW</td>
<td>BLUE</td>
</tr>
<tr>
<td>LED_F0</td>
<td>LED_F1</td>
<td>LED_F2</td>
<td>LED_F3</td>
<td>SYNTH_LOL_G0</td>
<td>LED_F8</td>
<td>LED_F10</td>
<td>LED_F11</td>
<td>LED_D0</td>
<td>LED_D1</td>
<td>LED_D2</td>
<td>LED_D3</td>
<td>LED_D4</td>
<td>CONFIG_DONE_F</td>
</tr>
</tbody>
</table>

| DS80 | DS81 | DS82 | DS83 | DS84 | DS85 | DS86 | DS87 | DS88 | DS89 | DS90 | DS91 | DS92 | DS93 | DS94 | DS95 | DS96 | DS97 | DS98 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| GREEN | GREEN | RED  | GREEN | GREEN | RED  | GREEN | RED  | YELLOW | YELLOW | YELLOW | YELLOW | YELLOW | RED  | RED  | GREEN | GREEN | GREEN | RED  | RED  |
| ETH_F_LINK100  | PHY_P4_LED2  | SFP2_LOS  | PHY_P4_LED1  | PHY_P3_LED1  | SFP1_LOS  | PHY_P3_LED2  | SFP4_LOS  | LED_G0  | LED_G1  | LED_G2  | LED_G3  | LED_G4  | PHY_P1_LED2  | PHY_P1_LED1  | PHY_P2_LED1  | PHY_P2_LED2  | SFP3_LOS  | 12V_WARN_R |
### 3.28 TEST POINT REFERENCE LIST

There are a lot of testpoints on the board. This list describes them. The reference designator in the lefthand column appears on the circuit board, and can be searched for in the schematic. The net names in the PIN columns are schematic net names, that can also be searched for in the schematic.

<table>
<thead>
<tr>
<th>TESTPOINT</th>
<th>PIN 1</th>
<th>PIN 2</th>
<th>PIN 3</th>
<th>PIN 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP1</td>
<td>CFPGA_PROGNTP</td>
<td>CFPGA_PROGN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP2</td>
<td>+12V</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP3</td>
<td>+1.1V_CPU</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP4</td>
<td>+1.0V_CPU</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP5</td>
<td>RST_CPU</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP6</td>
<td>+1.8V_CPU</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP7</td>
<td>RST_PORN</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP8</td>
<td>+3.3V</td>
<td></td>
<td>PCIE_Q_PWR</td>
<td>PCIE_Q_PWR_RTN</td>
</tr>
<tr>
<td>TP9</td>
<td>+3.3V</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP10</td>
<td>+DIMMB_VDD</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP11</td>
<td>+2.5V</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP12</td>
<td>+1.0V_Q</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP13</td>
<td>DEV_BURSTN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP14</td>
<td>DEV_READY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP15</td>
<td>+VBATT</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP16</td>
<td>+5.0V</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP17</td>
<td>GND</td>
<td></td>
<td>+0.9V_VTT_M</td>
<td></td>
</tr>
<tr>
<td>TP18</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP19</td>
<td>VBATT_EXT</td>
<td></td>
<td></td>
<td>GND</td>
</tr>
<tr>
<td>TP20</td>
<td>GEO_TSTP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP21</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP22</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP23</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP24</td>
<td>CLK_TP_BP</td>
<td></td>
<td>CLK_TP_BN</td>
<td></td>
</tr>
<tr>
<td>TP25</td>
<td>PEX_TP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP26</td>
<td>SATA_USB_TP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP27</td>
<td>DAUGHTERCARD_RESET_POWER</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP28</td>
<td>+DIMMG_VDD</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP29</td>
<td>RST_PEXON_CPU</td>
<td></td>
<td>PCIE_CPU_CPERSTN_FPGA</td>
<td></td>
</tr>
<tr>
<td>TP30</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP31</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP32</td>
<td>+12V</td>
<td></td>
<td>+12V_R</td>
<td></td>
</tr>
<tr>
<td>TP33</td>
<td>+12V</td>
<td></td>
<td>+12V_R</td>
<td></td>
</tr>
<tr>
<td>TP34</td>
<td>+12V</td>
<td></td>
<td>+12V_T</td>
<td></td>
</tr>
<tr>
<td>TP35</td>
<td>+1.0V_A</td>
<td></td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>TP</td>
<td>Voltage</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----------</td>
<td>----------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP36</td>
<td>+1.0V_B</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP37</td>
<td>+12V</td>
<td>+12V_R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP38</td>
<td>+1.0V_C</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP39</td>
<td>+1.0V_E</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP40</td>
<td>+1.0V_F</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP41</td>
<td>+12V</td>
<td>+12V_T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP42</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP43</td>
<td>+1.0V_D</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP44</td>
<td>CLK_TP_DP</td>
<td>CLK_TP_DN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP45</td>
<td>+12V_R</td>
<td>+12V_T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP46</td>
<td>+3.3V</td>
<td>PCIE_G_PWR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCIE_G_PWR_RTN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP47</td>
<td>+1.0V_G</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP48</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP49</td>
<td>+12V_R</td>
<td>+12V_T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP50</td>
<td>+1.2V_10G</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP51</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP52</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP53</td>
<td>+DIMMD_VDD</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP54</td>
<td>+12V_T</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP55</td>
<td>+12V_R</td>
<td>+12V_T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP56</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP57</td>
<td>+1.8V_10G</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP58</td>
<td>+12V_R</td>
<td>GND</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4 Software

Using the board requires configuring FPGAs, setting board controls such as clock frequency settings, and transferring data on and off board. For these purposes software has been provided. The best place to find details about the Emu software and programming API is in the “Emu_Manual.pdf” which can be found in the user package here: D:/Host_software/Emu/Documents/Emu_manual.pdf

4.1 Emu host software

In the user support package, here D:\Host_Software\emu\App
There is a program called ”EMU”. It can be used on Windows or Linux PCs. This program allows you to control the board.

The window shown above is the main window of EMU.
There is also a command-line version of EMU.

### 4.1.1 Selecting a board

To connect to a board using the EMU program, make sure the board is connected to the computer either over Ethernet, USB, or PCI Express. If the board is connected to Ethernet, make sure the network supports DHCP, or else you may not be able to connect to the board. If DHCP is not available, see the Emu_Manual.pdf document for more information on setting up a static IP address.

When using USB on Windows, a USB driver must be installed before the EMU program can detect the board. The Windows USB driver is located here:

`Host_Software\emu\Drivers\win32_usb`

You are expected to know how to install a windows driver using device manager.

Linux does not require a USB driver.

When using PCI Express on Windows, a PCI Express driver must be installed before the EMU program can detect the board. The Windows PCI Express driver is located here:

`Host_Software\emu\Drivers\win32_pci`

You are expected to know how to install a windows driver using device manager.

When using PCI Express on Linux, a driver is required. The Linux PCI Express driver is located here:

`Host_Software\emu\Drivers\linux86_pci`

There is also provided a shell script that will load the kernel module, and create suitable device nodes on the file system. You must have root privileges to run this shell script.
In Emu, from the Board menu, select Board->Select Board.

A drop-down menu will appear allowing you to select which board you wish to control, and over which interface. If you have multiple boards connected to the system, you can only control one at a time using each instance of the Emu program. After you have selected the board, the main window will update to show a picture of the board you are using.

Note that it may take about a minute from the time a board is powered on until when it becomes selectable from the EMU window. This is the time it takes the Marvell CPU to boot into Linux.

4.1.2 Configuring FPGAs
To configure an FPGA you can select the option FPGA->ConfigureFPGA from the menu bar. Or you can click on the photo of the board near one of the FPGA labels and select "configure FPGA" from the pop-up window. The program will ask for the path to the .bit file that you wish to use.

When the FPGA is done being configured, a blue dot will appear next to any FPGA that has been configured. The dot will stay blue until you clear the FPGA. Also note that a blue LED will light on the board itself.

You can clear an FPGA by clicking on it, and selecting "clear" from the pop-up window.

4.1.3 Clocks
There are 6 global clock networks on the board that have user-controllable settings. Each of those clocks has its frequency continually monitored and displayed on the main EMU window on the right side of the board photo. (The “Auto-Update” check box controls if EMU monitors the board for changes, or whether it only updates the screen when the user does an action).

To change the settings of the clocks, you can click on the text displaying that clock's frequency. A pop-up window will display options for the clock. Each clock may have different options, and all options may not be available for all clocks. For example, clocks G0, G1 and G2 can be set to a user-specified frequency, but USER_L and USER_R can not.

4.1.4 Sending data to and from the FPGA
The EMU program can also be used to transfer data to and from the FPGAs. The name of the interface on the FPGA that can be accessed from EMU is called "NMB". The "NMB" interface can be thought of as an address space. The EMU program can read and write to addresses on that space. Select “NMB Advanced” from the "Data" menu. There are also several other NMB options to play with including a memory space browser and file transfer functions.
In order to transfer data to your FPGA, your design must implement an NMB endpoint. The code necessary to implement an NMB endpoint is provided in the user package at D:\FPGA_Reference_designs\common\nmb

For help in understanding this endpoint, read the document here:
D:\Documentation\Manual\Dini Buses User FPGA Design Manual.pdf

The "main ref" reference design provided is here:
D:\FPGA_Reference_designs\Fpga_programming_files\user_fpga\main_ref\nCorrectly implements an NMB endpoint. You can load these test files into one or more FPGAs and use EMU's NMB functions to send data to the reference design.

4.1.5 Hardware Tests
To detect hardware failures, the EMU program is capable of testing the board. If you want to run a complete hardware test of the board, select the board in emu. From the Test Menu, select "Selected Tests". This window will appear.

All of the check boxes are tests on the board that can be run independently. The items in the "Factory Tests" area all require test fixture hardware to pass, so they will be of limited use to users for testing the board. The tests in the "field tests" area can all pass without special test fixtures. Note that the
DRAM Test requires that there is a DDR3 SODIMM (any density) installed in all 4 SODIMM slots on the board. If they are not installed, the test will fail.

Before running, the test may ask for the path to the ".bit" files used to program the FPGAs. A directory with an appropriate structure is provided on the user package in this location: D:\FPGA_Reference_Designs\Programming_Files

After the test(s) complete, the program will print out a message like this:

```
ONESHOT: rocketio_field_test PASSED
####################################################
ONESHOT: CUSTOM TEST COMPLETE.
rocketio_field_test : PASS
####################################################
CUSTOM TEST (REP 1/1) PASSED
DN0200_DN2076K10 1003019
18:18:02 5/26/2010
```

If the test stops or fails, you may need to hit the "q" key to regain control of the EMU program.

### 4.1.6 Command Line version

The EMU program compiles into two versions. The GUI version and the command-line version. Both versions can run in Windows or in Linux.

The menu options in the command-line version and in the GUI version are identical. The command-line version lends itself well to scripting.
4.1.7 Scripting with EMU

The command-line version of EMU uses stdin and stdout for input and output, and so it is possible to write scripts that interact with it. In this way you potentially can use the board without ever having to write any software of your own.

If you run the program with the `-c` switch, then EMU processes commands from stdin instead of presenting the text menu. Run EMU with the `-h` script for a list of available commands. The parameters for each command are identical to what EMU would expect if the same command was run from the text menu. Experiment with each command from the text menu to learn how to put it into a script.

4.1.8 Emu on the Marvell Linux environment

The command-line version of EMU is also installed on the Linux system that is installed on the Marvell CPU. This allows EMU commands to be issued to the board using the RS232 terminal or over a telnet session to the board. The command is “emu_mv”.

4.2 Writing your own software

To write your own software it is recommended that you start by attempting to compile the existing gui or command-line version of Emu. There is an Emu software manual that describes the process in the support package.

D:/Host_Software/emu/Documents

4.3 How EmuLib works

The Emu program and EMULIB communicate to the board through a tall stack of software linking the host PC to the Marvell processor, the Marvell processor to the FPGA I/Os, and the FPGA I/Os to your HDL.

However you are not expected to understand any of the workings of this stack of software.
On the host PC side, you are expected to understand and use the interface provided by the file diniboard.h (See the software Emu User Manual for details), found in the user package. On the FPGA side, you are expected to understand and use the interface provided by the file nmb_user_interface.v also found in the user package. See documentation here:

D:\Documentation\Manual\Dini Buses User FPGA Design Manual.pdf

The layers of software and hardware in between should operate transparently. For no particular reason details are given here:
1) Your C++ code calls nmb_write() function in the diniboard.h interface.
2) The emulib library determines which of the three interfaces the board is connected on. Let's assume ethernet
3) The emulib library creates a packet of data with a header and the data you supplied.
4) Emulib sends the packet to a special port at the ip address of the board.
5) On the board, a program called DiniCmos is listening to that very same port. It takes the data in the packet and drops it in the DRAM of the marvel.
6) Over PCI Express the DiniCmos program sets some registers in the DMA controller in the configuration FPGA.
7) The data is transferred to the configuration FPGA and then over the appropriate NMB link to the target FPGA.

4.4 Marvel Environment

The Marvell CPU is running a complete Linux operating system. Most standard Linux applications and utilities are already installed. You are able to program the Marvell processor with your own code so that the board can operate as a stand-alone device, without the need for a host computer in production environments.

4.4.1 Linux Provided

The Linux kernel on the board is version 2.6.22.18 as of this writing.

There is no particular name for the "distribution" on the board, however many of the common Linux utilities are provided by busybox.

4.4.2 Operating from the shell terminal

You can get a linux shell terminal by using telnet to access the board. The board will register its host name with the dhcp server, if the dhcp server supports dns. The host name of the board is: DN2076K10-xxxxxxx

where xxxxxxx is the 7-digit serial number of the board. The serial number can be found on the serial number sticker under the DIMM D memory socket. The host name of the board can also be found by connecting to the Marvell RS232 terminal and typing “hostname” at the prompt. To do this, follow the instructions in the next paragraph.
You can also access a terminal using the RS232 connector located near the lower right corner of the board, labeled "Marvell Serial". This connector is a standard computer "serial" port. The 2 x 5 connector can be connected to with the provided IDC-to-DB9 adapter cable.

The terminal settings are:
- Baud: 19200
- Data Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: OFF
- Emulation: VT200

The RS232 output is also the system console, so you will also see system messages on your terminal in addition to the shell output. If this bothers you, you can use telnet instead. To do this, follow the instructions in the previous paragraph.

### 4.4.3 Running EMU in the terminal

You can run EMU from the Linux terminal. From here you can configure FPGAs, set clocks, and send data to and from FPGAs. The command is “emu_mv”. Using the program is identical to using the command-line version on the EMU program on the host. You should connect to the board by ip address at address 127.0.0.1, which happens automatically when “emu_mv” is run.

If you really want, you can also control other boards from this board over Ethernet.

### 4.4.4 Compiling code on the Marvell

The compiler GCC and standard C headers and libraries are installed on the board. You can compile standard C and C++ programs that run in user space. For more information see the “Marvell Processor” section in the Emu_Manual.pdf.

### 4.4.5 Kernel Space

If you want to run code on the Marvell Processor in kernel space, the complete kernel code is required. The kernel code is not installed on the board, so compiling kernel mode code for the board cannot be done on the board. For this, you will need a build environment. Likewise, to modify the kernel itself also requires a separate build environment. We can provide a VMWare virtual machine with a cross-compiler installed that is capable of building the kernel and kernel modules. Alternatively, a cross-compile platform can be set up. See the Emu_Manual.pdf for additional information on building the kernel and the uboot boot loader.

When modifying the kernel, you should maintain your code as diff files, because if and when Dini Group modifies the kernel, we will not provide you with a change list, and you will have to re-port your changes to the new kernel source. Alternatively, you can develop the kernel changes you require and provide the change list to Dini Group, and we will mainline your changes.

Note that you may not need to make kernel modifications at all. We have provided a device located at...
That gives user-mode programs read/write access to the CPU0 memory space. If the only thing you need to do is access protected kernel memory, we recommend you just use this method.

### 4.4.6 Boot Sequence

When the board powers on, the CPU executes, in place, address 0xF60000 of the SPI device. In this case, the SPI device is an ATMEL data flash. The data flash contains code that initializes DRAM and loads the last 512KB of memory from the data flash into DRAM. The last 512KB of flash contains a u-boot bootloader image.

When u-boot runs, it reads a set of environment variables off the flash. It runs the u-boot command defined by the "bootcmd" variable. That command will copy the entire contents of the data flash image from address 0x0 of the data flash to address 0x2000000 of DRAM. At address 0x0 in the data flash is a compressed linux binary image. u-boot uncompresses the image and jumps to it. U-boot can pass a single string of information to the linux image, called the "boot arguments". The value of these boot arguments can be modified using u-boot environment variables.

Linux boots, setting up all the devices, including NAND flash. The NAND flash shows up as four MTD block devices, representing four different regions in the NAND flash. As the last step of booting, it mounts a JFFS2 file system from one device, specified in the boot arguments. Before starting a shell for the user, linux will run a shell script on the filesystem called startup.sh. This script loads the software that Dini Group wrote that controls the FPGAs and other functions of the board. This program is called "DiniCmos" and runs in the background. You don't need to worry about it.

See the Emu_Manual.pdf document for additional information about u-boot, kernel, and the root filesystem. There is a chapter in there called “The Marvell Processor” that is really good.

### 4.4.7 Compiling U-boot

Like the kernel, compiling u-boot requires a virtual machine or cross-compile platform. It is not recommended that you do this. Instead, submit your change request to Dini Group so that we can mainline your changes. See the Emu_Manual.pdf for additional information.

### 4.4.8 Creating the Root File system

There is no root file system build process. Instead we maintain a golden image in the form of a .tar file containing the contents of the root file system on the Marvell. Changes have to be made to the .tar file manually.

### 4.4.9 Update the software and firmware

When updates are made to the root file system or to the host software or drivers, an email is sent out to the emu-update mailing list. To subscribe to this mailing list, send an email to:

emu-update@dinigroup.com

with “Subscribe” in the subject line. The latest software package is always available at:

http://www.dinigroup.com/files/web_packs/emu.zip
The following sections give instructions on advanced update techniques for updating the various pieces of the firmware running on the board. Typically the Root File System (RFS) is the only firmware piece that users will ever update. There are no plans to ever modify the kernel or Uboot boot loader. For the best instructions on doing a firmware update, see the Emu_Manual.pdf file. Advanced users only may read the following section, but typically should not attempt any of these procedures without contacting support@dinigroup.com first.

4.4.9.1 Installing a kernel update

To get the current version of the Linux kernel, you can check the boot messages for this line:

```
## Booting image at 02000000 ...
    Image Name: Linux-2.6.22.18
    Created: 2010-04-02 1:27:51 UTC
    Image Type: ARM Linux Kernel Image (uncompressed)
    Data Size: 2840632 Bytes = 2.7 MB
    Load Address: 00008000
    Entry Point: 00008000
    Verifying Checksum ... OK
```

Check the "Created" date.

1) Connect the serial terminal to the board.

2) Power on the board. You should see u-boot boot messages in the terminal. At some point u-boot should print

Hit any key to stop autoboot: 3

At this step, press any key. You will then receive a u-boot prompt like this

```
>>
```

3) Type this u-boot command

```
protect off 1:0-63
```

U-boot will print this:

```
Un-Protect Flash Sectors 0-63 in Bank # 1
.................................................................
done
```

4) boot into linux by typing this u-boot command

```
boot
```

5) Once linux is done booting you will receive a command prompt like this
6) Type the following command

```bash
mount -t tmpfs tmpfs /mnt/ram -o size=32M
```

7) Type the following command

```bash
cd /mnt/ram
```

8) Get the update files from the Dini Group website. Put these files on the board. If you have the board connected to an internet-enabled network, you can use the `wget` command.

```bash
wget http://dinigroup.com/~marvellfiles/uImage
```

If you do not have access to the internet, then you will need to use some other method to transfer files to the board. You can use a USB key, a network mount, or any other Linux trick you know.

9) Type this command

```bash
cat uImage > /dev/partition_spi
```

This command updates the Linux kernel. If this command fails, the recovery procedure is still possible, but is more complicated.

10) Type this command

```bash
reboot
```

It is important that you use the reboot command, and do not simply power-cycle the board. If you power cycle the board, then the SPI flash may not get written completely due to write buffering.

### 4.4.9.2 Installing an RFS (root file system) update

This procedure will result in the loss of user data on the Linux file system. Back up your data. To check the version number of your root file system, you can type this Linux command:

```bash
cat /root/image.date
```

1) Connect the serial terminal to the board.

2) Power on the board. You should see u-boot boot messages in the terminal. At some point u-boot should print

```
Hit any key to stop autoboot: 3
```

At this step, press any key. You will then receive a u-boot prompt like this
3) At the prompt, type this command
run 'spi_boot_recoveryfs'
5) Once Linux is done booting you will receive a command prompt like this
   -sh#

6) At the command prompt type this Linux command

   sh /root/recover.sh

7) The recovery process takes about 10 minutes, plus longer for the 180MB download from

8) When the recovery script is complete. Type this at the shell prompt

   Reboot

   Do not power off the card without typing “reboot” or “halt” and
   waiting until the kernel totally shuts down. The “sync” command
   can also help. Otherwise the NAND Flash may become corrupt due to
   write buffering in linux.

4.4.9.3 Installing a U-boot update
A failure during this process will cause the board to be un-recoverable. Please consult Dini Group
before starting this process. There is normally no reason for the user to use this procedure. To check
your current version of u-boot, you can check the boot messages for this line

   The compile date of mv_main.c is May 14 2010

1) Connect the serial terminal to the board.

2) Power on the board. You should see u-boot boot messages in the terminal. At some point u-boot
should print

   Hit any key to stop autoboot: 3

   At this step, press any key. You will then receive a u-boot prompt like this

   >>

3) Type this u-boot command

   protect off 1:0-63

   U-boot will print this:
Un-Protect Flash Sectors 0-63 in Bank # 1
................................................................
done

4) boot into Linux by typing this u-boot command

```
boot
```

5) Once Linux is done booting you will receive a command prompt like this

```
-she
```

6) Type the following command

```
mount -t tmpfs tmpfs /mnt/ram -o size=32M
```

7) Type the following command

```
cd /mnt/ram
```

8) Get the update files from the Dini group website. Put these files on the board. If you have the board connected to an internet-enabled network, you can use the wget command.

```
wget http://dinigroup.com/~marvellfiles/u-boot-db78200_MP.bin
wget http://dinigroup.com/~marvellfiles/update_uboot
wget http://dinigroup.com/~marvellfiles/uImage
```

If you do not have access to the internet, then you will need to use some other method to transfer files to the board. You can use a USB key, a network mount, or any other Linux trick you know.

9) Type this command

```
chmod 500 update_uboot
```

10) Type this command

```
./update_uboot
```

11) Type this command

```
diff new_contents.bin u-boot-db78200_MP.bin
```

The file new_contents.bin contains the current contents of the SPI flash. This command is to make sure that the update was written successfully to the SPI flash. If this diff fails (shows the files are different) then something went wrong with the update, and you should not turn off your board. If you turn off the board at this time, the board will never boot again, and you will have to send it back to the factory for re-programming.
12) Was the diff clean? If not then stop here!!

13) Type this command

cat uImage > /dev/partition_spi

This command updates the Linux kernel. If this command fails, the recovery procedure is still possible, but is more complicated.

12) Type this command

reboot

It is important that you use the reboot command, and do not simply power-cycle the board. If you power cycle the board, then the SPI flash may not get written completely due to write buffering.
5 REFERENCE DESIGN

FPGA reference code is provided in the Customer Support Package at:
D:\FPGA_Reference_Designs\boards\dn0211_dn2076k10\MainRef

A very large portion of the design is contained in generic HDL modules which can be found in:
D:\FPGA_Reference_Designs\common
Much of this generic Verilog HDL code can be very useful in writing custom designs.

5.1 NMB Space Map

The upper 8 bits of the NMB address select the FPGA. 0x00=FPGA A, 0x01=FPGA B, 0x02=FPGA C, 0x03=FPGA D, etc. This is true for all NMB designs, it is not specific to the MainRef example design.

The MainRef sample design implements the following memory spaces. The underscore is to make the 64-bit addresses easier to read and should be omitted when entering addresses into software.

0x00010000_00000000: BLOCKRAM
0x00020000_00000000: DRAM
0x00040000_00000000: INTERCON TEST CONTROLS
0x00080000_00000000: REGISTERS

The REGISTERS space looks roughly like this:

0x08: NMB_REG_IDCODE   (Always reads 0x05000211)
0x10: NMB_REG_DESIGNTYPE
  | NMB_DESIGNTYPE_NO_NMB (0x00000000)
  | NMB_DESIGNTYPE_MISC (0x34560000)
  | NMB_DESIGNTYPE_MAINREF (0x34561111)
  | NMB_DESIGNTYPE_LVDSABC (0x34562222)
  | NMB_DESIGNTYPE_LVDSBCA (0x34563333)
  | NMB_DESIGNTYPE_FASTINTERCON (0x34564444)
0x18: NMB_REG_SODIMMSIZE
0x20: NMB_REG_SODIMMEEPROM
0x60: NMB_REG_LVDSINTERCONBANKS
0x68: NMB_REG_SINGLEINTERCONBANKS
0x70: NMB_REG_SPIFLASH
0x78: NMB_REG_VRN_PULLUPS
0x80: NMB_REG_VRP_PULLDOWNS
0x88: NMB_REG_MISC_PULLUPS
0x98: NMB_REG_MISC_PULLDOWNS
0xA0: NMB_REG_SPD_SCL
0xA8: NMB_REG_SPD_SDA
0xB0: NMB_REG_SINGLEFAST_TYPE
0x100: NMB_REG_ROCKETIO_STATUS
0x108: NMB_REG_ROCKETIO_TXRESET
0x110: NMB_REG_ROCKETIO_RXRESET
0x118: NMB_REG_USB2_PHY_ID
0x800: NMB_REG_CLKSTART

5.2 Compiling

The MainRef design builds in the Xilinx ISE software. Scripts for building it are found at:
D:\FPGA_Reference_Designs\boards\dn0211_dn2076k10\MainRef\buildxst

There are many, many Verilog macro defines that control how the design compiles. These are found at the top of the main Verilog file:

D:\FPGA_Reference_Designs\boards\dn0211_dn2076k10\MainRef\source\fpga.v
6 TROUBLESHOOTING

General tips for troubleshooting are presented here. If in doubt, send an inquiry to support@dinigroup.com before attempting anything that may damage the board.

6.1 General Procedure

If a board is not functioning properly follow this general procedure for troubleshooting:

1. Disconnect all peripherals, cables, daughtercards, and anything else that can be disconnected
2. Visually inspect the board for damage- burn marks, physical damage, etc.
3. Using a Multimeter, probe all power testpoints for GND shorts. Use the TEST POINT REFERENCE LIST in this manual to locate power test points. DO NOT POWER ON THE BOARD IF ANY POWER RAIL IS SHORTED TO GND.  If any shorts are present, stop now and send email to support@dinigroup.com with the findings.
4. Plug power back in to all 3 power connectors and power on the board. Look for RED LEDs that indicate things are broken. Note that the clock LOL (loss of lock) LEDs will be on until linux finishes booting (about 30 seconds), and this is normal. If any RED LEDs are on, stop now and send email to support@dinigroup.com with the findings.
5. [OPTIONAL]: Use a multimeter to probe each testpoint checked in step 3, only this time check that the voltage is correct. BE VERY CAREFUL NOT TO SHORT ANYTHING WITH YOUR MULTIMETER PROBES. If any voltage rails are wrong, stop now and send email to support@dinigroup.com with the findings.
6. Connect to the Marvell serial port (J18) with the supplied adapter. Terminal settings are 19200bps, no parity, no flow control. Power cycle the board and watch for text on this terminal. If no text is seen stop now and send email to support@dinigroup.com with the findings.
7. If text is seen, wait a minute for linux to boot. Eventually it will print out something like:
   
   DiniCmos Version: 2011.07.26
   DN0211_DN2076K10 - 0912065
   DiniCmos is now ready for host connections
   Pressing enter will then give a linux prompt. If this stage is not reached, stop now and send email to support@dinigroup.com with the findings. Include all of the text that came out of the serial port that did not result in a successful linux boot.
7 ORDERING INFORMATION

7.1 Part Number
All orders should reference the product as “DN2076K10”.

7.2 How to Order
Send email to sales@dinigroup.com for all sales inquiries. North American sales are handled directly by the main Dini Group office in La Jolla, CA. Dini Group also has sales representation all over the world. Include your geographic location in your sales inquiry so we may direct your request to the appropriate sales office.

7.3 Board Options
The number, type, speed grade, and position of FPGAs must be specified. DRAM options must be requested specifically. Chassis requests must be discussed. Many daughtercards are available, see www.dinigroup.com for more information. Contact sales@dinigroup.com to discuss available board options or for recommendations for your application.

7.4 Compliance Information

7.4.1 EMISSIONS CERTIFICATIONS
We are willing to obtain an FCC, CE, or other certification for volume production.

7.4.2 ROHS
The board is ROHS compliant. We can obtain a certification if necessary for volume production.

7.4.3 PCI-SIG
The board passes our internal PCI Express compliance test. If you need the board added to the PCI-SIG integrator's list for some reason, this may be done upon special request.
# 8 Glossary

You can expect this manual, marketing materials, diagrams, emails and source code from Dini Group to contain a lot of shorthand and imprecise phraseology. To ensure consistency and precision, here is a list of terms and words that we made up below:

<table>
<thead>
<tr>
<th>Word that exists</th>
<th>Thing that word means</th>
</tr>
</thead>
<tbody>
<tr>
<td>.bit file</td>
<td>All of these words interchangeably refer to the file containing the data for the SRAM configuration bits for the FPGA. This file is generated by the Xilinx &quot;bitgen&quot; program.</td>
</tr>
<tr>
<td>Load file</td>
<td></td>
</tr>
<tr>
<td>Configuration file</td>
<td></td>
</tr>
<tr>
<td>Bitstream</td>
<td></td>
</tr>
<tr>
<td>Hex file (rarely)</td>
<td></td>
</tr>
<tr>
<td>BUFG</td>
<td>Each of these are primitive HDL modules available in the Virtex 6 FPGA. It is assumed that the reader is familiar with the behavior and use of any primitives mentioned in each section. Please see the Virtex 6 user guide for the behavior of these modules.</td>
</tr>
<tr>
<td>BUFH</td>
<td></td>
</tr>
<tr>
<td>BUFIO</td>
<td></td>
</tr>
<tr>
<td>BUFR</td>
<td></td>
</tr>
<tr>
<td>DCM</td>
<td></td>
</tr>
<tr>
<td>IDDR</td>
<td></td>
</tr>
<tr>
<td>IDELAY</td>
<td></td>
</tr>
<tr>
<td>ISERDES</td>
<td></td>
</tr>
<tr>
<td>MMCM</td>
<td></td>
</tr>
<tr>
<td>ODDR</td>
<td></td>
</tr>
<tr>
<td>ODELAY</td>
<td></td>
</tr>
<tr>
<td>OSERDES</td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td>These words refer interchangeably to HDL code.</td>
</tr>
<tr>
<td>IP</td>
<td></td>
</tr>
<tr>
<td>Verilog</td>
<td></td>
</tr>
<tr>
<td>VHDL</td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td></td>
</tr>
<tr>
<td>DCI</td>
<td>Digitally controlled impedance. It refers to the ability of the FPGA to output optimally-terminated signals.</td>
</tr>
<tr>
<td>DIFF_TERM</td>
<td>Attribute that can be set on differential inputs to differentially end-terminate them.</td>
</tr>
<tr>
<td>DCM</td>
<td>Digital clock manager</td>
</tr>
<tr>
<td>DLL</td>
<td>Delay-locked loop</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase-locked loop</td>
</tr>
<tr>
<td>MMCM</td>
<td>something something clock manager</td>
</tr>
<tr>
<td>DDR</td>
<td>These words are (incorrectly) used interchangeably in this guide.</td>
</tr>
<tr>
<td>DRIVE attribute</td>
<td>Usually means &quot;double data rate&quot;, meaning that a signal's value is significant on both rising and falling edges of a clock. Sometimes this guide also uses &quot;DDR&quot; to refer to &quot;DRAM&quot;, just to add confusion.</td>
</tr>
<tr>
<td>Drive strength</td>
<td>This refers to the ability of the FPGA to produce a variety of output impedaces on its I/O.</td>
</tr>
<tr>
<td>FPGA Q Config FPGA NMB Bridge U0 Spartan V5 FPGA</td>
<td>These words are used interchangeably to refer to the controller FPGA on the DN2076K10, which is not generally available for user designs.</td>
</tr>
<tr>
<td>GND Ground</td>
<td>This refers to both a net on the DN2076K10, and to the absolute potential of that net at any given time. When an absolute voltage is given in this manual, it should be interpreted as a voltage relative to this net.</td>
</tr>
<tr>
<td>IOSTANDARD attribute</td>
<td>This is an attribute of the I/O buffer primitive.</td>
</tr>
<tr>
<td>ISE Bitgen iMpact XST CoreGen MIG</td>
<td>These are each software programs provided by Xilinx</td>
</tr>
<tr>
<td>LOC constraint</td>
<td>This is a constraint type that controls which physical pin on the FPGA device an I/O will be mapped to.</td>
</tr>
<tr>
<td>MB KB</td>
<td>Megabyte. Usually 1,000,000 bytes, but sometimes 1,048,576 bytes. Kilobyte. Usually 1,000 bytes, but sometimes 1,024 bytes.</td>
</tr>
<tr>
<td>Mbs Mb/s MTs MT/s</td>
<td>Mega(b)its divided by seconds. Mega(T)ransfers per second. These are used interchangeably with MHz, to avoid any possible confusion when dealing with DDR interfaces</td>
</tr>
<tr>
<td>MEG-Array Daughtercard Connector</td>
<td>There are several types of user-available connectors on this board. Usually, &quot;the daughtercard connector&quot; refers to one of the three &quot;Meg-Array&quot; connectors attached to FPGA or D.</td>
</tr>
<tr>
<td>MHz</td>
<td>1,000,000 divided by seconds.</td>
</tr>
<tr>
<td>Mux</td>
<td>Multiplexer</td>
</tr>
<tr>
<td>MV78200 Marvell Processor CPU MCU uP</td>
<td>The on-board processor manufactured by Marvell.</td>
</tr>
<tr>
<td>Net Signal Rail Plane</td>
<td>These interchangeably refer to a physical wire on the printed circuit board. A net on the circuit board is all points that are electrically shorted together.</td>
</tr>
<tr>
<td>NMB Main Bus</td>
<td>These interchangeably refer to the 40-signal bus between each FPGA and the configuration FPGA.</td>
</tr>
<tr>
<td>Bus</td>
<td>This word can mean any one of the following: An interface that is more than one signal wide. An interface that has more than two endpoints. An interface.</td>
</tr>
<tr>
<td>PCIE RGMII SGMII</td>
<td>PCI Express</td>
</tr>
<tr>
<td></td>
<td>This is an interface specification used for Ethernet physical interface devices.</td>
</tr>
<tr>
<td><strong>RocketIO</strong>&lt;br/&gt;<strong>GTP</strong>&lt;br/&gt;<strong>GTX</strong>&lt;br/&gt;<strong>MGT</strong></td>
<td>In this guide, these all refer interchangeably to the 6Gbs, high-speed serial transceivers available on Virtex 6 devices.</td>
</tr>
<tr>
<td><strong>RS232</strong>&lt;br/&gt;<strong>Serial Port</strong></td>
<td>This could refer to one of the following:&lt;br/&gt;The serial port on a computer&lt;br/&gt;The connector on the DN2076K10 that is intended to connecting to a computer's serial port.</td>
</tr>
<tr>
<td><strong>SATA</strong></td>
<td>Serial ATA</td>
</tr>
<tr>
<td><strong>SEAM</strong>&lt;br/&gt;<strong>Serial Daughtercard</strong></td>
<td>These refer to three connectors attached to FPGAs A,C and D. These connectors carry high-speed serial signals from the Virtex-6 FPGAs. SEAM refers to the name of the Samtec brand connector series that were selected for this board.</td>
</tr>
<tr>
<td><strong>SFP</strong></td>
<td>Small form-factor pluggable module. This refers to the socket attached to FPGA F that is intended for Gigabit Ethernet.</td>
</tr>
<tr>
<td><strong>SMA</strong></td>
<td>These are little gold-colored screw-in coax connectors that are on the board in various places</td>
</tr>
<tr>
<td><strong>SODIMM</strong>&lt;br/&gt;<strong>DDR3</strong>&lt;br/&gt;<strong>SDRAM</strong>&lt;br/&gt;<strong>DIMM</strong></td>
<td>These all refer interchangeably to the four SODIMM connectors attached to FPGA A, C, D and F. These connectors are intended for memory modules to attach.</td>
</tr>
<tr>
<td><strong>Source Synchronous</strong></td>
<td>This refers to a clocking strategy that involves introducing a controlled amount of clock skew to devices with respect to each other.</td>
</tr>
<tr>
<td><strong>SPI</strong></td>
<td>I don’t know what it stands for. Serial something interface maybe?? It is the protocol used for the serial Flash on this board.</td>
</tr>
<tr>
<td><strong>SSTL</strong>&lt;br/&gt;<strong>LVDS</strong>&lt;br/&gt;<strong>LVCMOS</strong>&lt;br/&gt;<strong>HCSL</strong>&lt;br/&gt;<strong>HSTL</strong>&lt;br/&gt;<strong>CML</strong>&lt;br/&gt;<strong>LVTTL</strong>&lt;br/&gt;<strong>LVDS_EXT</strong></td>
<td>These are all signaling standards. They each require certain voltage levels and termination schemes.</td>
</tr>
<tr>
<td><strong>System Synchronous</strong></td>
<td>A clocking strategy where each device receives a low-skew clock. In this guide, I often use this word as shorthand for &quot;using one of the provided global clock networks&quot;</td>
</tr>
<tr>
<td><strong>UCF</strong></td>
<td>I used this as shorthand for &quot;the set of place-and-route constraints that you input into ISE&quot;. It refers to the .ucf file extension of Xilinx ISE constraint files. It stands for “Universal Constraint File”.</td>
</tr>
<tr>
<td><strong>CC</strong>&lt;br/&gt;<strong>GC</strong>&lt;br/&gt;<strong>MRCC</strong>&lt;br/&gt;<strong>SRCC</strong>&lt;br/&gt;<strong>VREF</strong>&lt;br/&gt;<strong>VRN</strong>&lt;br/&gt;<strong>VRP</strong></td>
<td>These are all special-purpose I/O pins on the Virtex 6 FPGA. For each section that mentions one of these pins, you are expected to know the purpose and usage of each.</td>
</tr>
<tr>
<td><strong>XAPP</strong>&lt;br/&gt;<strong>UG</strong></td>
<td>These refer to reference documents produced by Xilinx.</td>
</tr>
</tbody>
</table>