XAPP351 (v1.0) November 7, 2000

# The CoolRunner CPLD IRL Demo: An Example of Using the Internet to Configure a CoolRunner CPLD


## Summary

This document details the process used to demonstrate configuring a CoolRunner® CPLD over the Internet. All of the VHDL source files and software files associated with this document can be downloaded. Please go to section **Software Disclaimer and Download Instructions**, page 8, for instructions.


## Introduction

Because of the widespread use of network connected digital devices, there is a tremendous opportunity in being able to quickly upgrade these devices electronically anywhere in the world. Internet Reconfigurable Logic (IRL) is a method of reconfiguring programmable logic via the Internet. Since a large number of handheld and portable devices are now providing Internet connections, the low-power CoolRunner CPLD is the logical choice for IRL hardware in these products. This technique of reconfiguring programmable logic combined with the low-power CoolRunner CPLD also provides a powerful advantage for intranet or network applications as well. This document will detail the CoolRunner CPLD IRL demonstration which shows that IRL is an effective and flexible method for upgrading CoolRunner CPLDs in a variety of systems.

## CoolRunner CPLD IRL Demo

To show the ease and effectiveness of reconfiguring a CPLD over the Internet, a CoolRunner CPLD is used to control messages on a BetaBrite™ scrolling LED sign. The message to be displayed on the BetaBrite sign is selected from a workstation or PC connected to the Internet. The CPLD is then reconfigured over the Internet to display the selected message on the BetaBrite sign. Though an application may be much more complex than controlling a scrolling LED sign, the idea of reconfiguring a CPLD over the Internet to provide additional features or functionality in your application is easily extended from this example.

Figure 1 shows the hardware required in this demo. The TINI™ board (Tiny InterNet Interface) from Dallas Semiconductor along with the STEP™ board (Systronix TINI Engineering Platform) from Systronix are used to provide the Internet connection. The CoolRunner CPLD is placed in the available prototyping area of the STEP board and communicates to the BetaBrite sign via

RS-232. The controller on the TINI board receives the CoolRunner CPLD design files from the Internet and controls the reconfiguration of the CoolRunner CPLD.

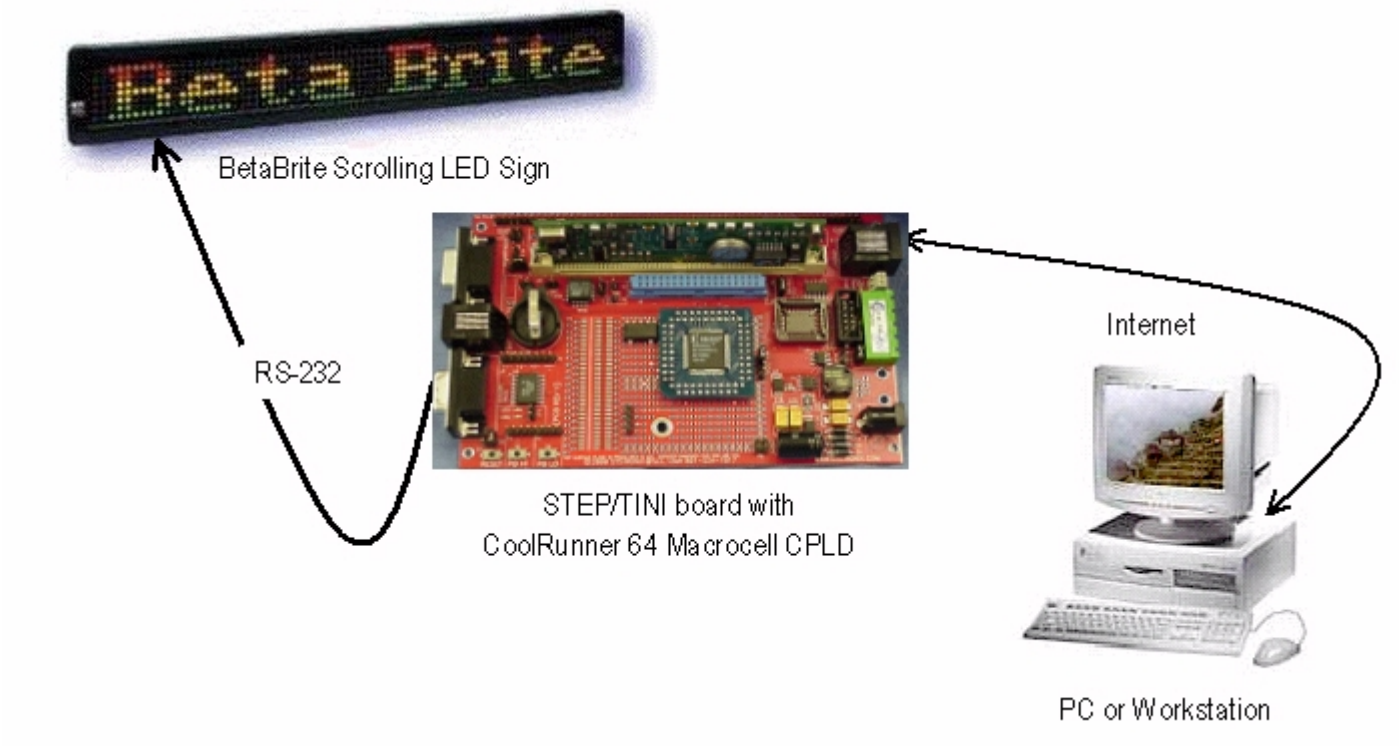


*Figure 1:* **CoolRunner CPLD IRL Demo**

## Running the Demo

The network computer uses Netscape or another web browser and connects to the IP address of the TINI board. The TINI board serves up its home page. The user then clicks on **XPLAServlet** to display the HTML file shown in Figure 2.

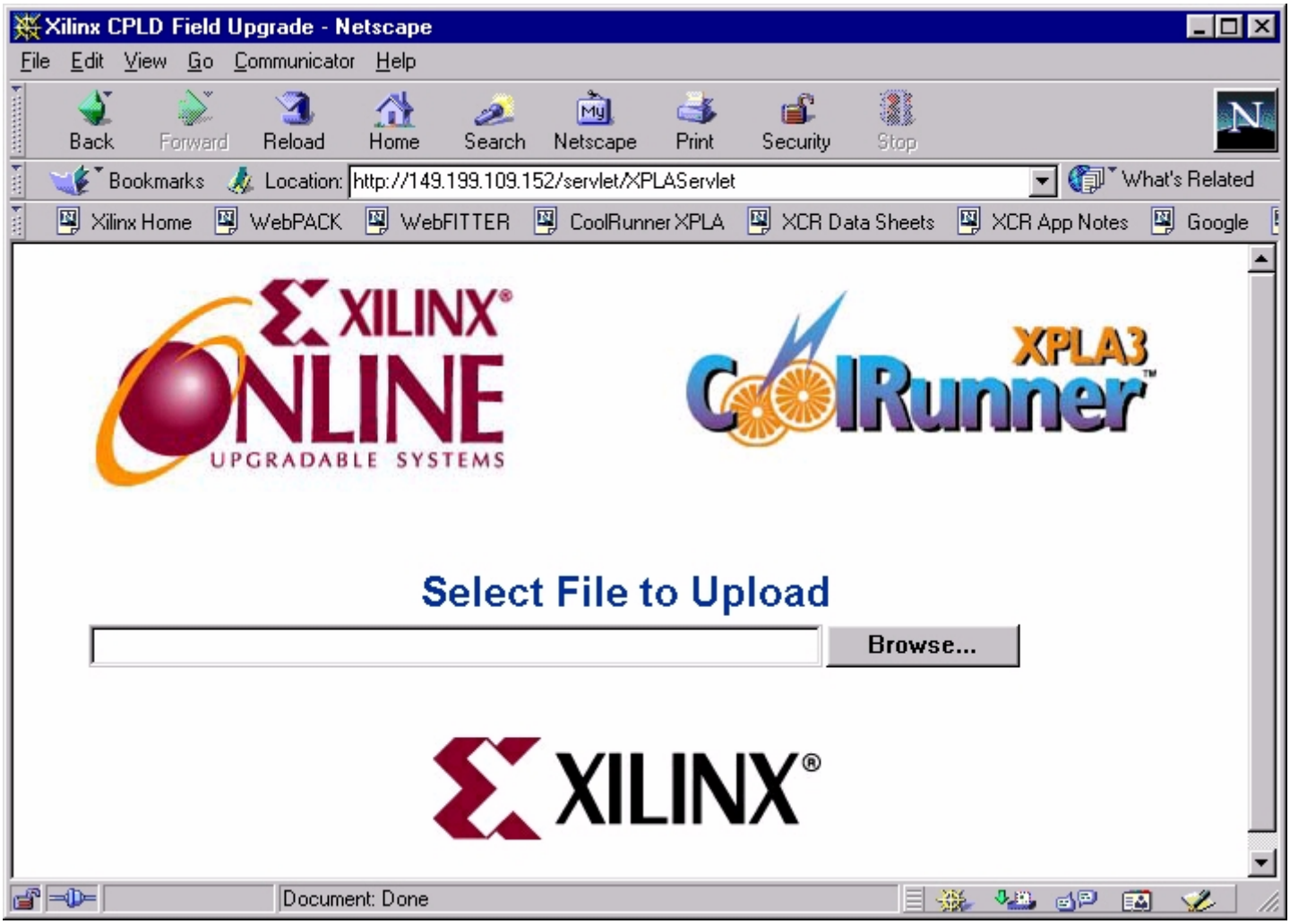


*Figure 2:* **IRL Demo HTML Page**

The user then selects the file to upload by clicking on the **Browse...** button. This file, corresponding to the message to be displayed, is then uploaded to the TINI board. The TINI board then programs the CoolRunner CPLD with this file which causes the message to change

on the BetaBrite sign. When the CoolRunner CPLD has been successfully programmed, the HTML file shown in Figure 3 is displayed.

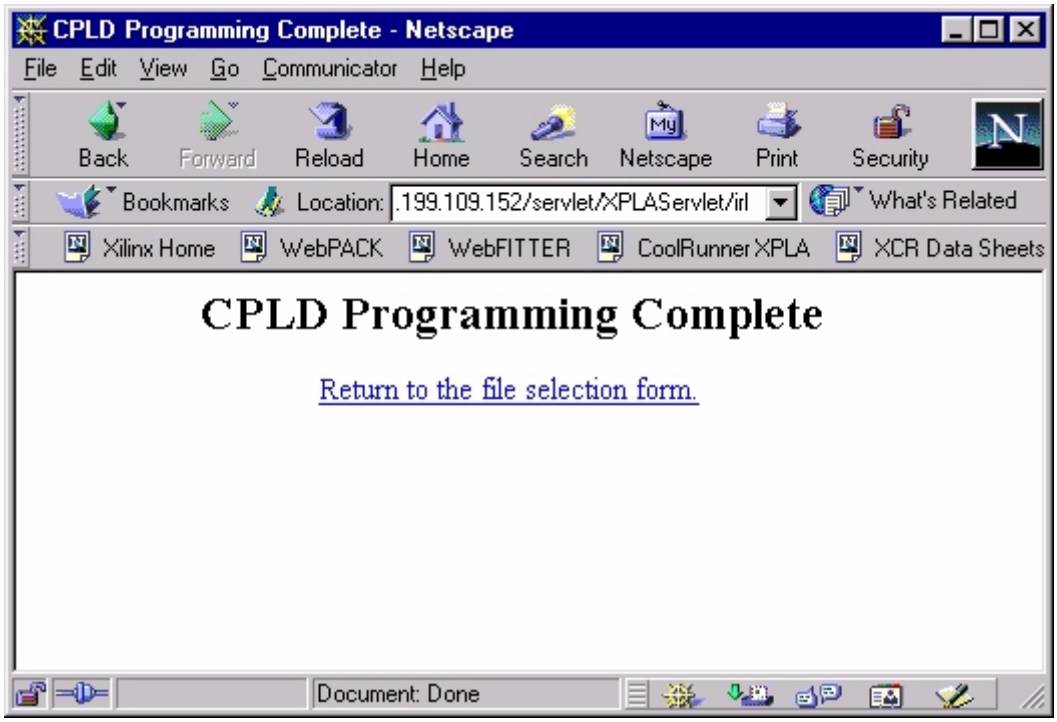


*Figure 3:* **HTML File when Programming is Successful**

## Details of the Demo

The TINI board is configured with an IP address for the network. When a computer on the network connects to the TINI board, the TINI board serves an HTML file allowing the user to run the XPLA Servlet. Once the XPLA Servlet has been chosen, the TINI board serves another HTML file which allows the user to select the CPLD configuration file. Different CPLD files corresponding to different messages on the BetaBrite sign are stored on the network computer. The user then selects the CPLD configuration file for the TINI board to upload. The TINI board then reconfigures the CoolRunner CPLD via the JTAG pins of the CPLD with the file just uploaded. The new CPLD design instructs the Betabrite sign to display a different message via the RS232 port and the new message is then displayed.

### BetaBrite Messages

The BetaBrite sign allows many messages with special effects and colors to be stored in memory. Messages can be programmed into the memory of the BetaBrite sign either by a handheld remote control or by a software package that runs on the PC. Messages are then downloaded to the BetaBrite sign via RS232.

Each message in memory is indexed by a file label. The sign can then be instructed to display a message or series of messages from its memory by sending it a *Set Run Sequence* command string which contains the file label(s) to be displayed.

For the CoolRunner IRL demo, the BetaBrite software is used to program different messages with special effects into the BetaBrite sign. Each message is given a unique file label by the BetaBrite software. The CoolRunner CPLD design then simply transmits the *Set Run Sequence* command string with a particular file label or file labels to the BetaBrite sign over RS232. The BetaBrite sign then displays the message sequence associated with the file label(s).

The format of transmission frames to the BetaBrite sign and the format of the *Set Run Sequence* command can be found in the Alpha™ Sign Communications Protocol Document, Revision C.

### CoolRunner CPLD Design

The CoolRunner CPLD design consists of a controlling state machine which steps through the bytes of the *Set Run Sequence* command and a low-level state machine which serializes the bytes into bits to be transmitted over the RS232. Note that the STEP board provides the RS232 level translator.

This design only transmits over RS232 - it does not receive data. It consists of two major design components, the Command State Machine and the Transmit UART State Machine. The Command State Machine sequences through the bytes that comprise the *Set Run Sequence* command to the BetaBrite sign. The message label to be displayed is a constant in the command state machine code and can be easily modified.

The Transmit UART State Machine controls the reception of a byte of data from the Command State Machine and the transmission of this data byte over the serial port. The current design transmits an IDLE bit, a START bit, eight DATA BITS, and a STOP BIT. It does not transmit a parity bit. This serial protocol can be changed in the transmit code.

The block diagram for this design is shown in Figure 4. The Transmit Buffer Register (TBR) is loaded with data from the Command State Machine when the WRN signal is asserted. This will cause the Transmit Buffer Register Empty (TBRE) flag to negate which starts the Transmit UART State Machine. This state machine loads the data into the Transmit Shift Register (TSR) and shifts the data out on SDO.

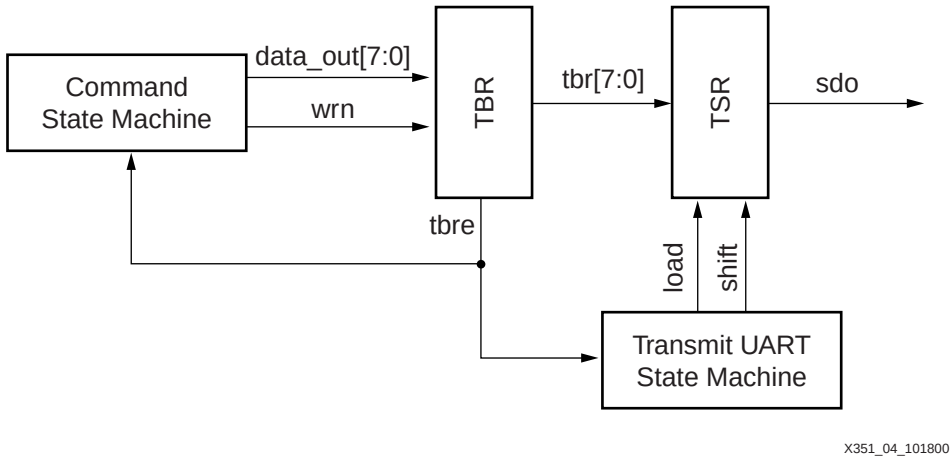


*Figure 4:* **CoolRunner CPLD Design Block Diagram**

### Command State Machine

The code for the Command State Machine consists of a block of constants that define the transmission frame format and the *Set Run Sequence* command format for the particular command and file label to be sent to the BetaBrite sign. This operation occurs only once when the reset signal from the TINI board is negated. The number of bytes in the frame is also set in the constant, WORD_COUNT. Using constants in this manner keeps the state machine small, concise, and consistent even though the file label or labels of the messages differ from design to design.

The Command State Machine is shown in Figure 5 and operates off the system clock. When there is a rising edge on TBRE (TBRE_RE =1) indicating that the Transmit Buffer Register (TBR) is empty, the state machine transitions from the START state to the SET_DATA state. In this state, the first data byte of the frame is output to the TBR and the word counter is incremented. In the next state, the WRN signal is asserted to load the data into the TBR. The state machine returns to the START state to wait for TBRE_RE to assert. This continues until

all words in the frame have been output. At that point, the state machine transitions from the START state to the STOP state and the transmission of the frame is complete.

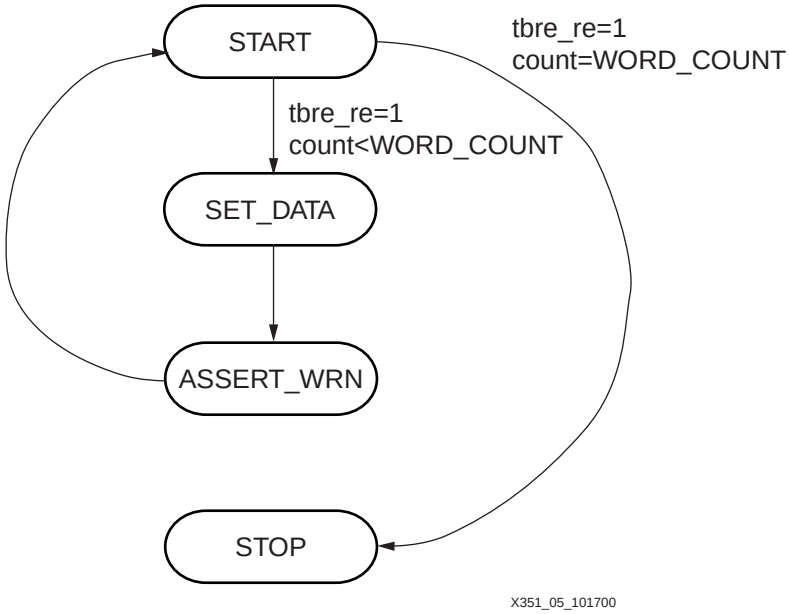


*Figure 5:* **Command State Machine**

### Transmit UART State Machine

The Transmit UART State Machine (Figure 6) operates from a divided version of the system clock as determined by the baud rate of the transmission. The state machine moves from the IDLE state to the START_BIT state when the TBRE signal negates, indicating that the Transmit Buffer Register is no longer empty. The START_BIT state outputs the start bit as defined in the RS232 specification. The state machine then shifts the actual data bits out of the Transmit Shift Register (TSR). When all eight bits have been output, the state machine transitions to the STOP_BIT state to output the RS232 stop bit. The start bit, data bits, and one stop bit protocol are defined as the communication parameters required by the BetaBrite sign.

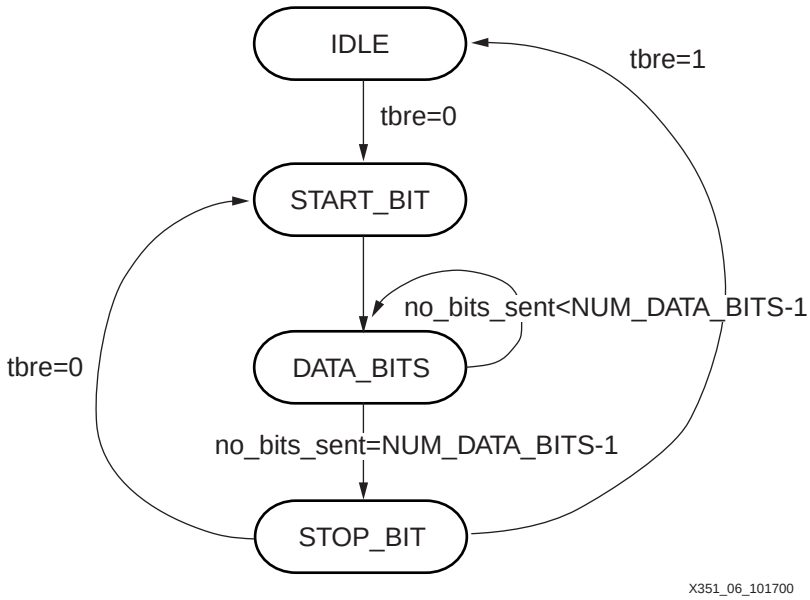


*Figure 6:* **Transmit UART State Machine**

### *CoolRunner CPLD Design Implementation*

The CoolRunner CPLD design is implemented in an XPLA3 64-macrocell device in a VQ100 package (XCR3064XL-10VQ100). This design uses ~51 macrocells.

### TINI Board Software

The TINI board contains a JAVA Virtual machine. Code has been written for the TINI board to program the CPLD over the JTAG pins using a *.BIF file. This file is written as a Simplified Binary File (SBF) for configuring CoolRunner CPLDs in embedded systems. The SBF for the design is created by running the CoolRunner ISP software. For more information about SBF, download **XAPP326: Simplified "In-System Programming" for Embedded Systems Using CoolRunner CPLDs** available at **http://www.xilinx.com/xapp/xapp326.pdf**

Each byte of data in the SBF contains 4-bit pairs with the values of TMS and TDI. These bit pairs are applied to the TMS and TDI pins of the device and a rising edge on TCK is provided. To run more efficiently on the JAVA Virtual machine, the SBF file is first "unwound" so that the resulting file contained simple bit pairs of TMS/TDI values. These "unwound" files are the files that are transferred from the network computer to the TINI board and used by the TINI board to reconfigure the CoolRunner CPLD. The JAVA application that "unwinds" the *.BIF files is BifUnwinder.jar.

The TINI board not only contains the software to re-configure the CPLD, but the software to serve the HTML pages as well. All of the TINI software is found in the file *irldemo.zip* which is available for download. Please see **Software Disclaimer and Download Instructions**, page 8 for instructions.

### *TINI/STEP and CoolRunner CPLD Board Connections*

Figure 7 shows the block diagram of the implementation of the CoolRunner CPLD on the prototyping area of the STEP board. Schematics for the TINI board and STEP board are available from the Systronix website and the Dallas Semiconductor website. See Table 2, page 9 for the URL to these websites. The shaded blocks are components that are on the prototyping area of the STEP board, the other blocks are components present on the STEP board.

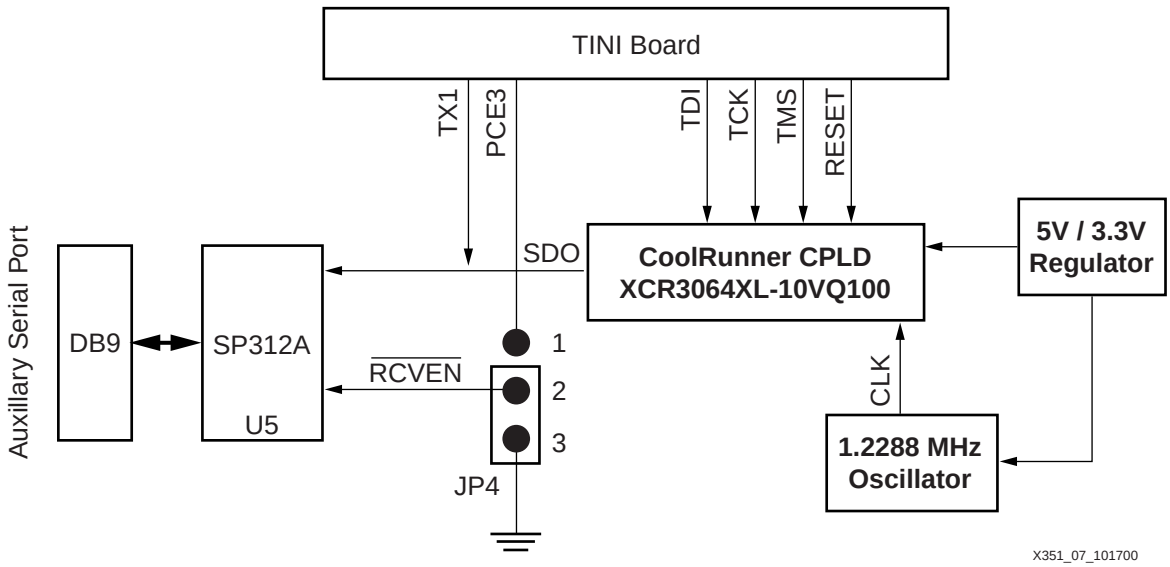


*Figure 7:* **CoolRunner CPLD Integration on STEP/TINI Board**

TMS and TDI on the CoolRunner CPLD are connected to the TINI data bus pins D1 and D0. TCK is connected to Peripheral Chip Enable 0 (PCE0). The CPLD is memory mapped so that accesses to the CPLD appear as memory cycles, thus PCE0 provides the correct edges for TCK. Note that the SDO output from the CPLD is connected to the TX1 pin of the TINI board.

The TX1 pin is 3-stated for this application, therefore there is no signal contention on this line. Since the CoolRunner CPLD is utilizing the RS232 drivers on the STEP board, jumper JP4 on the STEP board must be in the 2-3 position to enable these drivers.

Table 1 shows the CoolRunner CPLD connections to the TINI board and the oscillator added to the prototyping area.

*Table 1:* **CoolRunner CPLD Connections**

| CoolRunner CPLD XCR3064XL-10VQ100 | | TINI Board | |
|---|---|---|---|
| **Signal Name** | **Pin Number** | **Signal Name** | **Pin Number** |
| TMS | 15 | D1 | 50 |
| TDI | 4 | D0 | 51 |
| TCK | 62 | PCE0 | 28 |
| SDO | 85 | TX1 | 12 |
| Reset | 61 | P5.0 CTX | 8 |
| TDO | 73 | D2 | 49 |
| Clock | 90 | Oscillator | |

## Software Disclaimer and Download Instructions

All VHDL source code, VHDL testbenches, and software files associated with this design are available. THE DESIGN IS PROVIDED TO YOU "AS IS". XILINX MAKES AND YOU RECEIVE NO WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. This design should be used only as an example design, not as a fully functional core. XILINX does not warrant that the performance, functionality, or operation of this Design will meet your requirements, or that the operation of the Design will be uninterrupted or error free, or that defects in the Design will be corrected. Furthermore, XILINX does not warrant or make any representations regarding use or the results of the use of the Design in terms of correctness, accuracy, reliability or otherwise.

XILINX EXPRESSLY DISCLAIMS ANY WARRANTY OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE, THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OR REPRESENTATION THAT THE IMPLEMENTATION IS FREE FROM CLAIMS OF ANY THIRD PARTY. FURTHERMORE, XILINX IS PROVIDING THIS REFERENCE DESIGNS "AS IS" AS A COURTESY TO YOU.

**XAPP351** - **http://www.xilinx.com/products/xaw/coolvhdlq.htm**

## Conclusion

This demo shows that the mechanics of upgrading a CPLD over the Internet are feasible and accomplished without difficulty. The implementation of IRL can easily be extended to more complex programmable logic designs. By implementing IRL, designs can be upgraded whether it is in the next office or on the other side of the world, providing enormous field service cost savings.

The URLs provided in Table 2 provide detailed information about topics and hardware discussed in this application note.

*Table 2:* **URLs for More Information about IRL Demo**

| | |
|---|---|
| Xilinx Online and IRL | http://www.xilinx.com/xilinxonline/index.htm |
| Xilinx CoolRunner CPLDs | http://www.xilinx.com/products/xpla3.htm |
| Dallas Semiconductor TINI Board | http://www.ibutton.com/TINI/index.html |
| Systronix STEP Board | http://www.systronix.com/home.htm |

## Revision History

The following table shows the revision history for this document.

| Date | Version # | Revision |
|---|---|---|
| 11/07/00 | 1.0 | Initial Xilnix release. |
| | | |