XAPP441 (v1.1) September 9, 2006

Remote FPGA Reconfiguration Using MicroBlaze or PowerPC Processors

Author: KY Park and Hyuk Kim

Summary

Field upgradeability is one of the key features of recent FPGA based systems. This application note describes techniques for remote FPGA reconfiguration through an Ethernet port. Remote reconfiguration as demonstrated in this document will require the use of either MicroBlaze™ or PowerPC™ embedded processors, external Flash, SDRAM memory, and a Xilinx CPLD. "Watch-dog" monitoring is included in the solution.

This application note also presents a system level solution using a Xilinx CPLD and Flash memory to configure and monitor Xilinx FPGA configuration status.

Introduction

For this application note, Xilinx designed a special daughter card, which can be plugged into a P-160 module connector on an Insight Virtex™-II Pro development board. But, you can design and build your own system using this application note. The main target of our reference design is Wireless BTS (Base Tranceiver Station), but you can target any other system which requires a remote FPGA reconfiguration solution.

Table 1: **Supported Devices**

MicroBlaze Processor	Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-4, Spartan™-II, Spartan-IIE, Spartan-3 and Spartan-3E
PowerPC Processor	Virtex-II Pro and Virtex-4 FX

RFCS (Remote FPGA Configuration System) System Overview

There are five main devices in this system:

- **Target FPGA**: The target FPGA is used to implement the main user design. It also incorporates the MicroBlaze or PowerPC processor with GPIO, EMC, UART, EMAC and the SDRAM controller
- **CPLD**: The CPLD is used to implement the FPGA reconfiguration, and perform hardware and software configuration watch-dog and status monitoring. This design can use the XC95288XL or the XC2C256 CPLD device.
- **Flash memory**: Used to store FPGA hardware and firmware image. It is usually divided into four areas
  - Factory default hardware sector: Used to store known good data and reconfigure the FPGA if the hardware update fails
  - Factory default software sector
  - Hardware update sector. After power up, the CPLD gets the FPGA bit stream data from this sector and tries to configure the FPGA three times. If this fails or results in a software loading timeout, then the CPLD sets the Flash bank address (2-bit MSB) to the factory default area (known good data) and reverts to the previous FPGA configuration
  - Software update sector
- **Ethernet PHY device**

© 2005-2006 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc.
All other trademarks are the property of their respective owners.

- **SDRAM device:** In this application the memory type used on the board is SDRAM. However, DDR SDRAM, DDR-II SDRAM, SRAM, and other devices can be used as well.

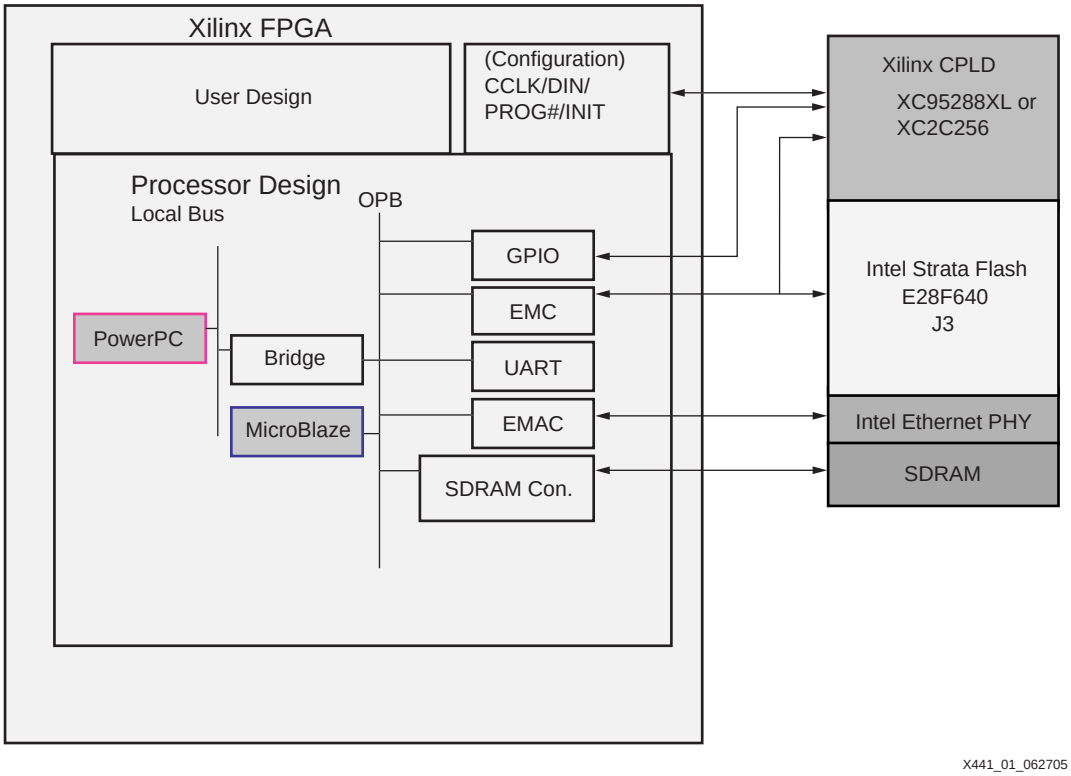


Figure 1: **System Diagram**

During normal FPGA operation, you can reload the hardware and software update sectors through the Ethernet. In this design, Xilinx used UDP protocol for the update. The FPGA processor stores the incoming bit-stream to SDRAM first, using the SDRAM controller. When all the data has been received, the processor then uses the EMC (External Memory Controller) to write the data to the update bank on Flash memory. Once the processor finishes writing all the data to Flash memory, it can initiate FPGA reconfiguration by driving “gpio_reconfig” to high.

Also you can boot up from the factory default area by driving special GPIO, even if there is no error on update.

With the combination of software and CPLD functionality, you can change the FPGA configuration at any time, even during normal operation. The allowed mode changes are:

1. From Update Configuration to Factory Default
2. From Update Configuration to Update Configuration
3. From Factory Default to Update Configuration

In our reference design, Master Serial configuration mode is used. For more information about Master Serial configuration mode, refer to each device data sheet, or ISE software online help.

There are three watch-dog functions in CPLD design:

1. CRC Error checking during configuration
2. No sync word found (no bit-stream on Flash)
3. Software loading time out

There are six GPIO pins to control and monitor FPGA configuration:

1. “gpio_rdb” : GPIO Read
2. “gpio_writeb” : GPIO Write
3. “gpio_reg(2:0)” : GPIO Register (Bidirectional)
4. “gpio_reconfig” : GPIO for FPGA Reconfiguration

In Figure 2, you can find the details of device connection. Also check the linked RFCS evaluation board schematic found in the **zip file** for this application note.

Note: there is no SDRAM interface part on our board schematic because Xilinx used the SDRAM on the Insight Virtex-II Pro development board. If you want to build your own application system for RFCS, you need to consider an SDRAM interface part on the board.

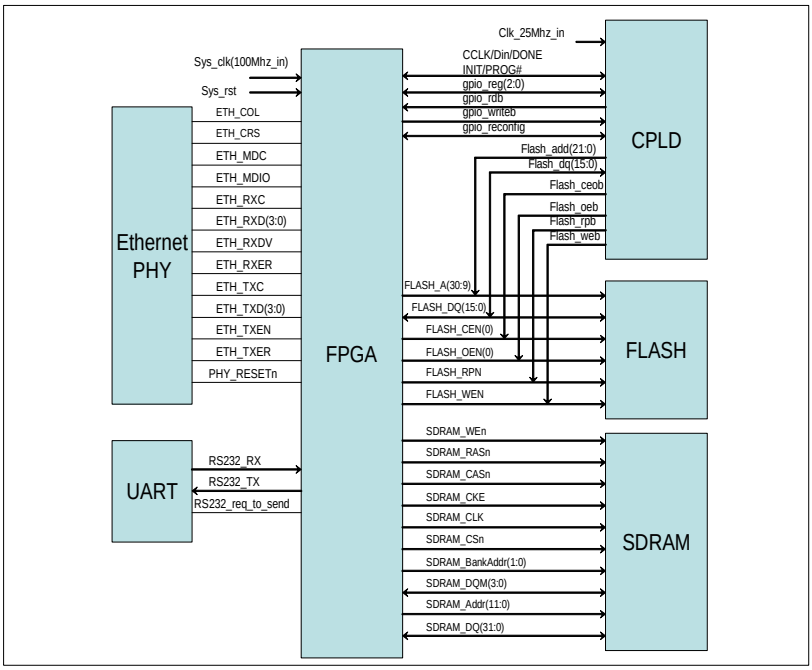


Figure 2: **RFCS System Device Connection Diagram**

RFCS System Development and Test Procedure

Hardware Implementation

1. FPGA design includes Processor design
   a. Unzip the attached EDK design, and then add or remove peripherals.
   b. Adjust the address map on MHS.
   c. Build a user design.
   d. Or, build your own processor plus a user design using EDK and ISE tools.
2. CPLD design

   You can use the attached reference VHDL design with UCF file. Adjust the Flash address bit width to support enough Flash memory space so it can store the full FPGA hardware and software Images for each factory and update bank. Also, if you only need configuration functions, then you can remove all other functionality from the CPLD design file (such as watch-dog monitoring, status and control, GPIO interface).
3. System Level design

   Refer to Figure 2 to build your own system, and adjust the address width in each.

Software Design

Xilinx used a server and client application to communicate through Ethernet between a PC and the target FPGA board. The server program works on the target FPGA board, and the client program works on a PC. The client program is able to open a data file and send it through Ethernet to predefined IP addresses on the target FPGA. The server will receive and save data until an End-of-File (EOF) arrives. The data is saved to an external SDRAM. Between the server and client there is an additional protocol to mark the EOF.

Test procedure

1. Prepare good and bad hardware and software images in binary format. A bad hardware image can be created from a good hardware image by changing some of bits, and a bad software image can be created from good source code by removing the loading time-out clear routine. If you use BRAM to store a small user application program, then you do not need to use the software loading time-out function because the software image is merged into the hardware image, and, in that case, you also need to remove the software loading time-out watch-dog function from the CPLD reference design.
2. Connect a PC to the Ethernet port on the board and set the FPGA configuration mode to Master Serial.
3. Open a terminal window and set 115200,n,8,1
4. Set the IP address on the PC to 165.213.112.168, and the Subnet Mask to 255.255.255.0 (you need to match the board IP address range)
5. In EDK, open the processor design and then open the XMD tool to download the `factory.bit` (known good design) file to the FPGA through the JTAG debug port.
6. Ping test to 165.213.112.167 (board) to check the Ethernet connection status.
7. Using a menu on the terminal window, download good software and hardware images to the Flash memory update and factory default areas. Then, power the board off and on, and check the configuration status (Succeed on User update flash bank).
8. Download a bad hardware image to the update flash bank, and a good hardware image to the factory flash bank. Then turn the power off and on, and check the CPLD watch-dog function. Try reconfiguration three times by driving the “PROGRAM” pin to low, then observe the CRC error status. Next, reconfigure again after changing the flash hardware address to the Factory default bank (also check to see if the Factory area is displayed on the terminal window).
9. Download a good hardware update area image with a bad user update software image, then check the configuration status after powering off and on. You can find the software application time-out report, and if configuration has succeeded on Factory default hardware and software sectors.
10. Download a good user update software image to Flash again. After configuration success of the user update area, drive in manual mode to invoke reconfiguration of the Factory default hardware and software sectors. After success configuring the user update area, invoke reconfiguration again by driving the GPIO_RECONFIG to check live reconfiguration with the new user update configuration from the user update configuration area (assume Flash update image update case).

Real Hardware and Software Implementation

Required Hardware Settings and Design Files

In this example application, Xilinx used a Memec XC2VP7 Rev. 4 board, and an RFCS board (P160 replacement). However, you can use your own board as long as it is based on this application note.

- Memec DS-KIT-2VP7 Rev. 4 board, and an RFCS board (or your own board based on Figure 1)

- Six flying wires to connect the configuration daisy-chain in/out port on the RFCS with the Memec board daisy chain ports on System ACE™ header (for your own board, all of device connection will use PCB routing). On the Memec Virtex-II Pro board, change the jumper position to disable PROM on JP26.
- `User_appbad.bin` : Bad software image to check software application loading error
- `User_bitbad.bin` : Bad hardware bit-stream to check CRC error
- `User.bin` : Good normal user `.bit` file. User application is merged in `.bin` file
- `Fact.bin` : Good normal factory `.bit` file and Fact default application software image are already merged in this `.bin` file.
- Socket program for making a client interface, which runs on a PC.
- XPS project file so that you can download to an FPGA with JTAG first. With this step, you can download a MicroBlaze/PowerPC based design to an FPGA, then start UDP Ethernet communication between a PC and the board to write the software and hardware images, depending on the test procedure.

Hardware Design

Processor

- `GPIO_WR` is a signal to write data to a CPLD register.
- `GPIO_RD` is a signal to read data from a CPLD register.
- `GPIO_REG` is a data bus to read and write data from/to a CPLD
- `GPIO_RECONFG` signal is used to invoke FPGA reconfiguration from a CPLD
- `OPB_UART` is a terminal for command interface
- `OPB_ETHERNET` is used for downloading data through UDP.
- `OPB_SDRAM` is used to save data from the Ethernet temporarily
- `OPB_EMC` is used to program Intel Strata Flash memory.

Note: In an `.mhs` file there is a statement to make the `fpga_init` pin high after configuration

```
PORT fpga_init = net_vcc, DIR = OUT
```

Because the CPLD is using the `fpga_init` pin (INIT# of FPGA) as a main reset input, if you leave it as unused I/O, the `fpga_init` pin will be set to ground after configuration (as the default bitgen option). This causes the CPLD to keep its reset status forever.

FLASH Memory (Intel Strata Flash)

Flash memory is divided into two sectors in this application. The first is the Factory sector and the second is the User sector. Usually, the Factory and User sectors are divided into the hardware image and the software application image, but, in this implementation, all applications are located in BRAM on the FPGA. In this design, the good Factory and good User sectors have the same images except for messages displayed on the UART terminal the first time after FPGA configuration. You are always able to overwrite data on the User sector of Flash memory.

CPLD Design

You can find a full behavioral level VHDL design in the attached files (`rfcs_top.vhd`), and in that design you can also find a detailed description in comments for each block of the CPLD design. The major functions of the CPLD design support the following:

- Read hardware image from Flash and configure/reconfigure FPGA.
- Hardware and Software Watch-dog block
- Status register
- Zip of source code (HDL, UCF)

Pin Description

- clk25m : input, 25 Mhz clock for Intel Ethernet PHY and CPLD

----- Intel Flash Memory Interface

- `flash_dq(15:0)` : input, Flash data port
- `flash_add(21:0)` : output, Flash address
- `flash_oeb` : output, Flash output enable, Active Low
- `flash_web` : output, Flash write enable, Active Low
- `flash_ceob` : output, Chip enable, Active Low
- `flash_rpb` : output, Reset and Power down, Active Low, High during normal operation

----- FPGA Configuration and Monitoring

- `cclk` : input, FPGA configuration clock from FPGA
- `fpga_din` : output, FPGA configuration data to FPGA
- `fpga_init` : input, FPGA INIT# from FPGA, used as main reset
- `fpga_done` : input, FPGA DONE pin
- `fpga_progb` : inout, FPGA PROGRAM#, used to re-program FPGA when output, and status monitoring when input

----- FPGA GPIO Interface for Status Monitoring and Control

- `gpio_reconfig` : input, FPGA reconfiguration invoked by driving this pin to high
- `gpio_writeb` : input, GPIO Write, usually drives to “101” for write operation
- `gpio_rdb` : input, GPIO Read, usually drives to “101” for read operation, but when initializing the watch-dog counter on the CPLD, this pin will be driven to low with “gpio_reconfig” pin.
- `gpio_reg(2:0)` : in/out, GPIO register.
- `LED6,LED8` : output, connect to LED, Active high.
- `TP1` : output, Test point

----- GPIO Register Description

- `write_register(2)` : Software writes this bit to '1' to request Reconfiguration (software should drive this register)
- `write_register(1)` : Changes the Flash bank to the Factory area and reconfigures the FPGA. '1' when request manual mode (drives `request_factory`, software should drive this register)
- `write_register(0)` : Software sets this register to '1' when software loads successfully (software should drive this register)
- `read_register(2)` : Same as `Write_register(2)` , '1' if reconfiguration request issued, or '0' if not.
- `read_register(1)` : Same as `Write_register(1)` , Manual mode '1', or Normal mode '0'
- `read_register(0)` : Status for software fail; software fail '1', software loaded successfully '0'

CPLD Design Features

Reset Generation

Generate an active low reset signal internally using the power on reset status of a counter with 25 MHz clock input.

Flash Interface

Enable three-state during configuration, and set the I/O for Flash read operation.

Parallel to Serial Counter

Counts up to “FF” to check 16-bit data shift out to FPGA.

Flash Address Counter

When the parallel to serial counter counts up to “FF”, it increments the Flash Address Counter. After a 16-bit shift, it loads new data from Flash, and creates the Flash data load signal and address.

Flash Data Byte Swap

When a `.bit` file is converted to a `.bin` or `.Hex` file, byte swap is required before configuring the FPGA.

Flash Data Load and Shift

Loads byte swap Flash data and shifts out to FPGA.

GPIO Read and Write

GPIO Read/Write interface. For more detailed information on each GPIO register bit, refer to the GPIO Register Description, page 6. To prohibit abnormal operation due to a glitch on a signal, SHIFT and AND logic are used for the GPIO Write signal.

Generate GPIO Check Enable

To prohibit unwanted operation during transition periods, such as end-of-configuration and beginning of FPGA normal operation, the CPLD counts 12 bits, and once the counter values all go high, then “gpio_chk_en” changes to high and enables normal GPIO interface operation.

“gpio_reconfig” Interface to Reconfigure FPGA

This part of the design is used to interface User section to User section reconfiguration. You can request FPGA reconfiguration by driving this pin to high.

No Sync. Word Detection

If the FPGA `.bit` section is blank on Flash memory, then the FPGA could not receive the sync word configuration data from the CPLD during its configuration process. If this happens, CCLK will come out continuously and the flash address generation counter will continuously cycle 0 to its maximum value. Once the CPLD watch-dog function detects this situation by counting eight cycles on the flash address generation counter, it will change the configuration bank address to the Factory default and request an FPGA reconfiguration.

CRC Error Check

Senses falling edge of INIT while PROGRAM# = '1'

Count Retry Configuration Three Times on Update(User) Area in CRC Error Case

If a CRC error is found during configuration, the program retries FPGA configuration on the User section three times, and then, if it still cannot configure the FPGA successfully, it changes the Flash bank to Factory and tries reconfiguration.

Flash MSB Address Change in Each Reconfiguration Case

Changes Flash MSB address to “0” or “1” according to each reconfiguration situation.

"Reconfig_clear" Generation to Reset Previous Reconfiguration Request Status

Clears each previous reconfiguration request status by driving "gpio_rdb" and "gpio_reconfig" simultaneously to low from the processor.

Software Loading Timeout Counter

If software loads successfully, then software writes '1' to write_register(0), but if write_register(0) is not set to '1' within a specific period (depending on the requirements of software loading time out), then the CPLD will retry FPGA configuration with the Factory default configuration, and report the software fail status to the GPIO register.

FPGA Reconfiguration by Driving PROGRAM# to Low

Combines each reconfiguration request and then sends a low pulse on to the "PROGRAM#" pin to issue an FPGA reconfiguration. The minimum pulse width to drive low the PROGRAM# pin is 300 ns in this design. Refer to the FPGA data sheet for more detailed information.

Software Design

This design uses a server and client application. The server works on a Memec board, and the client works on a PC. The client program is able to open a data file and send it through Ethernet to predefined IP addresses. The server will receive data until end-of-file (EOF), and save it to external SDRAM during receiving. Between the server and client there is an additional protocol to identify the EOF.

Server Application Program Running on Memec Board

- **Server function using UDP**. This function is used to set up Ethernet MAC and IP addresses, and open one UDP socket. After that it is ready to get data from the UDP port. If data arrives from the UDP port, the application saves it to SDRAM memory until all of it is saved.
- **Flash read and write function**. You can use the application to write and erase the Factory or User sections of Flash memory. If you choose the "Data write" command from the terminal window, the processor will read data from SDRAM and write it to the correct Flash address.
- **Software application certification function**. The CPLD has a one bit register that should be set to '1' by the software application so that the CPLD will know the software application is working correctly. If the software application does not set this register, the CPLD will assume that there is an application error, and issue a reconfiguration. But, because the read and write port is different for this register, we have to keep the data on this register to check the previous configuration status until the processor reads the data and clears it--even if the CPLD reset signal is driven by the FPGA reconfiguration process.

  The code is as follows:

```
i = cpld_value = read_data();
i = i | SOFTWARE_OK;
write_data(i)
;
```

- **Clear counter in CPLD function**. This design uses only three GPIO ports as control signals to read, write and reconfigure. To clear the value on a counter in a CPLD, this design uses combinations of read and reconfiguration signals. Driving both to low will clear the counter in a CPLD. The code is as follows:

```
void clear_software_fail()
{
XGpio_mSetDataReg(XPAR_OPB_GPIO_RD_BASEADDR, 0xff);
XGpio_mSetDataReg(XPAR_OPB_GPIO_RECONFIG_BASEADDR, 0xff);
XGpio_mSetDataReg(XPAR_OPB_GPIO_RD_BASEADDR, 0x00);
XGpio_mSetDataReg(XPAR_OPB_GPIO_RECONFIG_BASEADDR, 0x00);
```

```
XGpio_mSetDataReg(XPAR_OPB_GPIO_RD_BASEADDR, 0xff);
XGpio_mSetDataReg(XPAR_OPB_GPIO_RECONFIG_BASEADDR, 0xff);
}
```

- **Software functions to be done before trying reconfiguration**. Because the CPLD has so many functions to monitor in the configuration flow, it uses many latches to keep the status. So, you should reset it before configuration. The following code will reset the CPLD latches:

```
// Clear manual reconfiguration bit
clear_software_fail() {
i = cpld_value = read_data();
i = i & ~0x02;
i = i | SOFTWARE_OK;
write_data(i);
}
```

You can also try to reconfigure in manual mode, using a manual bit held in one of the CPLD registers. You should clear this bit before reconfiguration using the following code.

```
i = i & ~0x02;
i = i | SOFTWARE_OK;
write_data(i);
```

- **Software functions to make the CPLD trigger reconfiguration**. After clearing all information in the CPLD registers, you can issue the FPGA reconfiguration by driving the "GPIO_RECONFIG" pin to high > low > high as you can see in the application function below.

```
void try_reconfiguration()
{
Xint16 i;
clear_software_fail();
i = cpld_value = read_data();
i = i & ~0x02;
i = i | SOFTWARE_OK;
write_data(i);
i = read_data();
xil_printf("\n\r Manual mode disable CPLD reg value 0x%02x", i);
XGpio_mSetDataReg(XPAR_OPB_GPIO_RECONFIG_BASEADDR, 0xff);
XGpio_mSetDataReg(XPAR_OPB_GPIO_RECONFIG_BASEADDR, 0x00);
XGpio_mSetDataReg(XPAR_OPB_GPIO_RECONFIG_BASEADDR, 0xff);
}
```

Client Application Program Running on a Laptop Computer

Client function using UDP. This function is able to open a data file and send it to a predefined IP address through UDP. It was created using a Borland C++ compiler. To run it, you have to have the executable file as well as a **`.dll`** file.

Protocol Between Server and Client

The client application opens a data file and reads 0 x 80 bytes of data in each transfer cycle and then adds one byte of data to let the application software know how much data will be sent out

in each cycle. Once all of the data are transferred, the client will set "0x00" on the additional data byte to let the application know "EOF."

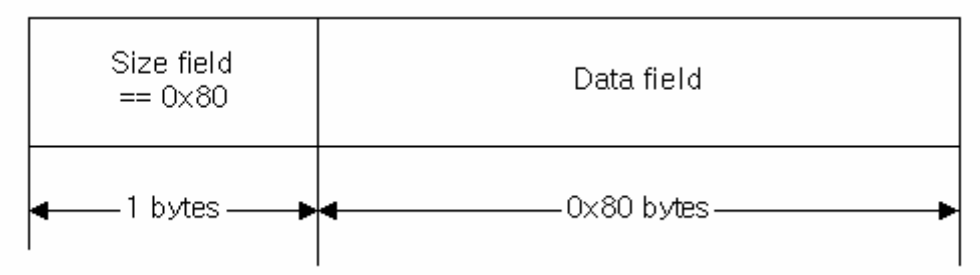


Figure 3: **Data File**

The client sends 0 x 81 bytes, and waits until the server returns 0 x 81 bytes. The server will return until the whole data transmit is complete.

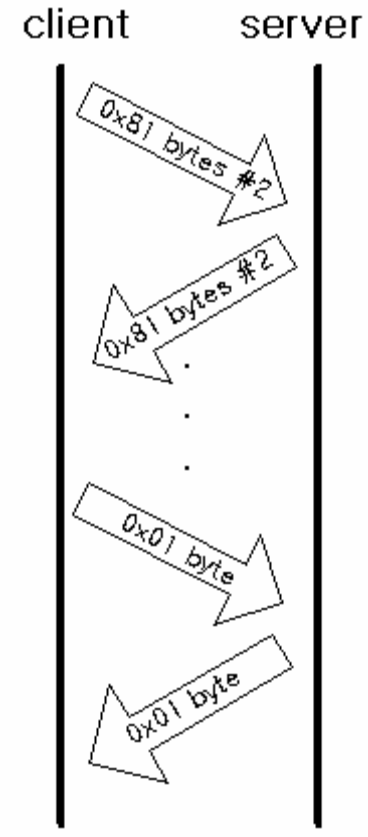


Figure 4: **Data Transfer**

The client application reads 0 x 80 bytes, and adds one byte to define how much data is saved. Most of the data stream has 0 x 80 + 0 x 01 data size. But the last packet's data size in the data field would be less than 0 x 80.

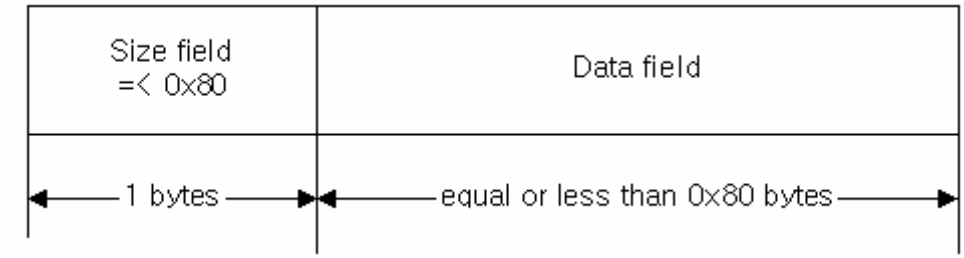


Figure 5: **Last Packet**

To terminate the data flow, the client sends one more packet, but in the size field there is a zero to tell the server to terminate this flow.

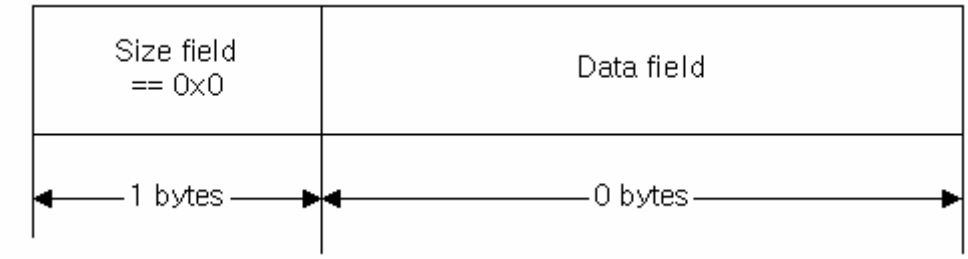


Figure 6: **Packet Flow Termination**

Make Bin File from Bit File

`.bin` file format is the most popular data format for CPLDs to read and configure FPGAs. You will need to convert **`.bit`** files to **`.bin`** files. The iMPACT tool provides a way to make **`.bin`** files, but a command line batch file may be a more convenient way to convert files. Usually, XPS makes a downloadable **`.bit`** file after "update bitstream." This design uses the following batch file to make it:

```
promgen -u 0x0 download -p bin -w
cd..
copy ..\implementation\download.bin .
```

Demo Board Setup

Xilinx strongly recommends you use the Memec XC2VP7 Rev. 4 board instead of Rev. 1 and Rev. 2, because Rev. 1 and Rev. 2 boards do not support the PROM disable function. So, you would have to desolder the D0 pin on each PROM of Rev. 1 and Rev. 2 boards to disable the PROM. On the Memec XC2VP7 Rev. 4 boards, the PROM disable function is supported and you can easily disable the PROM by connecting a jumper header on J22.

Jumper Setting (on Memec XC2VP7 Rev. 4 boards)

- J22 – Set to PROM disable
- J23, J24 – Set V_{CCIO} on bank 3 and 4 to 2.5V
- P12 Configuration Mode pin – Set to all zeros to select master serial configuration mode

Flying Wire (Rev. 4)

Requires you to connect all Daisy-chain pins on the RFCS board to the System Ace jumper header. Connect the daisy chain between RFCS and the FPGA on the System ACE connector of the Insight Virtex-II Pro board

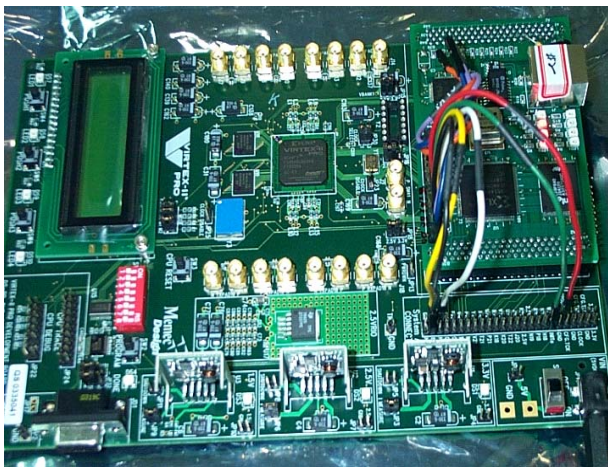


Figure 7: **Flying Wire Connection Between RFCS Board and Memec Virtex-II Pro Board**

HyperTerminal Setting

- 115200, n , 8, 1

PC IP Setup for Client Function

Set your PC's IP address as shown in Figure 8.

```
165.213.112.168
```

Set the Subnet Mask as shown in Figure 8:

```
255.255.255.0
```

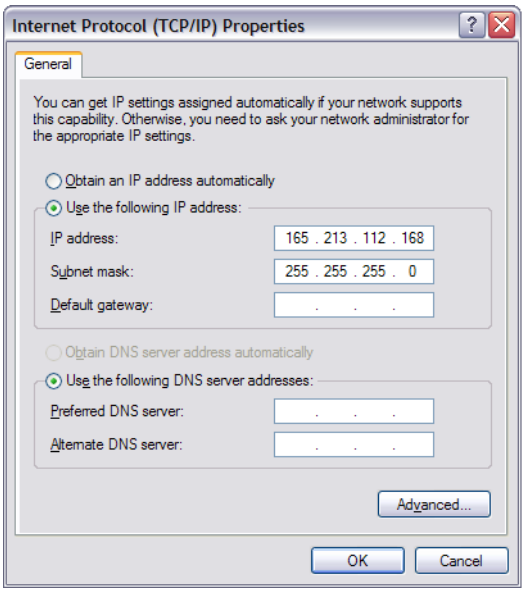


Figure 8: **IP Address and Subnet Mask**

Running the Application

1. Download the **`fact.bit`** file to the FPGA using XMD
2. Test the Ethernet connection using a ping test
3. Program the **`fact.bin`** file and **`user.bin`** file to Flash memory
4. Test **`user_badbit.bin`**
5. Test **`user_appbad.bin`**
6. Test manual mode

1. Download the fact.bit File Using XMD

This application shows the CPLD configuration status register's value at the top of the message every configuration cycle.

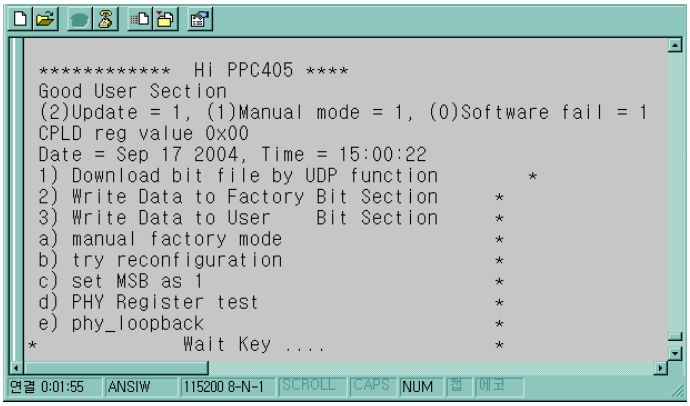


Figure 9: **CPLD Configuration Status Register**

By choosing menu "1", you can set up the board's MAC and IP addresses, and then the processor will wait on the data until the client sends a data packet.

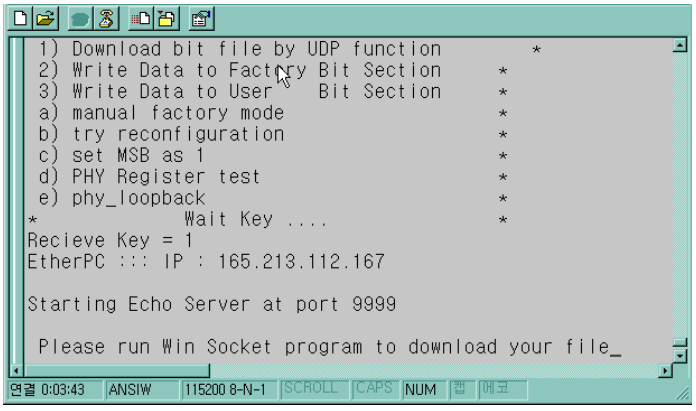


Figure 10: **MAC and IP Addresses**

2. Ping Test

The client application has a pre-defined IP address to download data. The address is 165.213.112.167, so you have to set the correct IP address. Before downloading a data file, running a ping test is recommended to verify the connection.

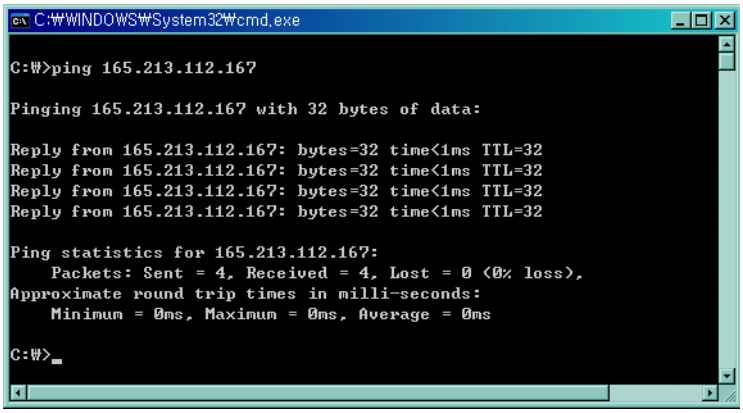


Figure 11: **Ping Test**

3. UDP and Client Program (PC)

If you run **`client_echo.exe`**, it will display the following screen. You can select a new bitstream file to pre-define an address (165.213.112.167)

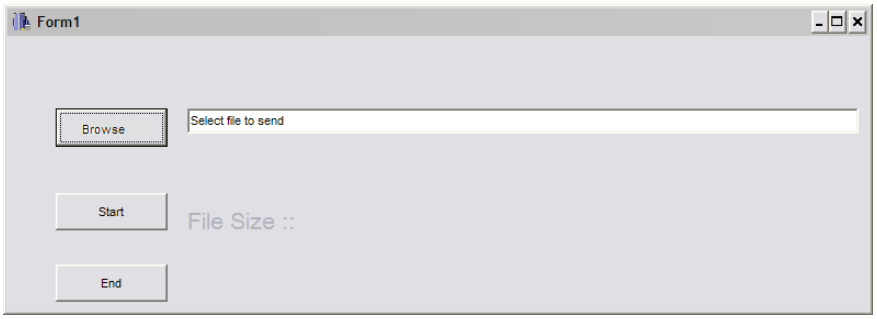


Figure 12: **Select Bitstream File**

4. Download Data File

Download time takes less than 6 seconds, and the server program displays ‘z’ in series to show the progress of the download

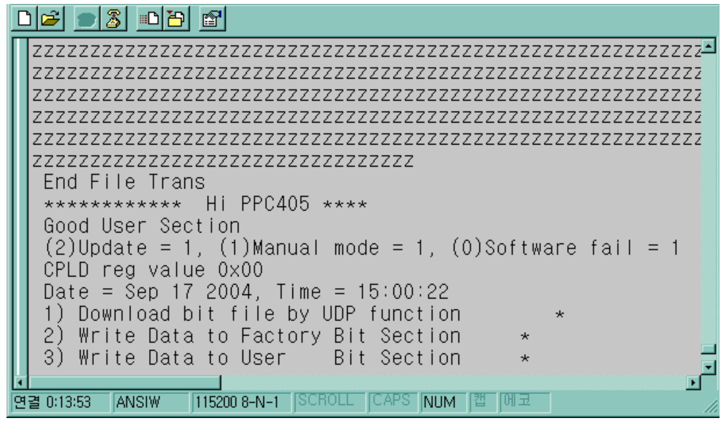


Figure 13: **Data Download**

The client program displays a message when transfer is complete

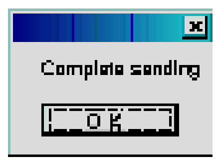


Figure 14: **Data Transfer Complete**

5. Write Data to Flash’s Correct Section from SDRAM

Now, new data is stored in SDRAM. The server application also has functions to program Flash memory. The `.bit` file size is always the same so that there is a #define state to define FILE_SIZE, and in RFCS there is 16-bit Intel Strata Flash memory. If you choose ‘3’, the data will be written to the User section of Flash memory.

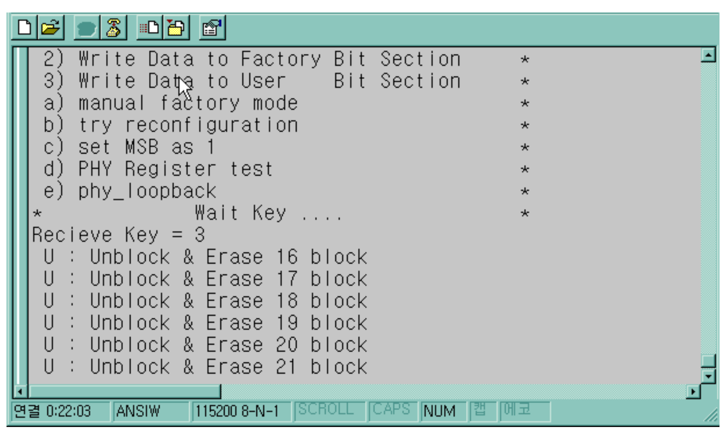


Figure 15: **Write Data**

6. And It Writes Data to Flash Memory

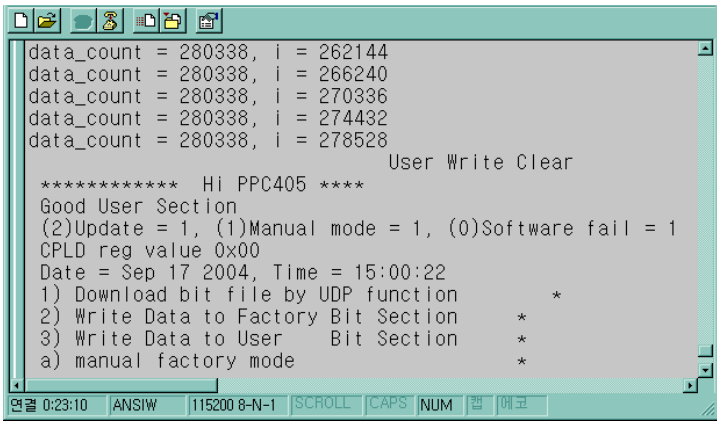


Figure 16: **Flash Memory Write**

7. Set CPLD to Reconfigure the FPGA

Now you should reconfigure the FPGA. You can choose ‘b’ to reconfigure, and, if you do, you can see the done LED turn off and on in one second. The message on the hyper-terminal is changed from “User Section” to “Fact Section”

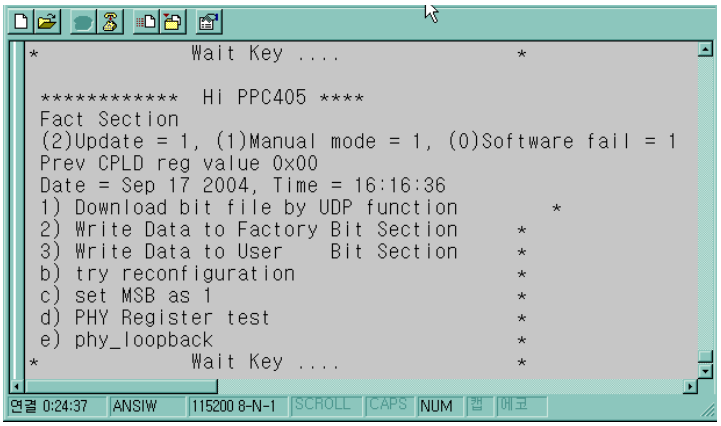


Figure 17: **Fact Section**

8. Update Flash User Section with the `user.bin` File

Choose '1' in the hyper-terminal again, and download the **`user.bin`** file with the **`client_echo.exe`** program. Update the User section, and reconfigure the FPGA again.

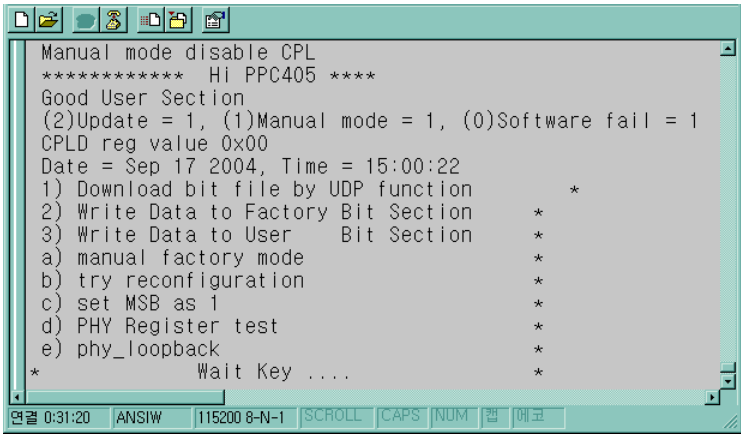


Figure 18: **Reconfigure FPGA**

9. Manual Reconfiguration

Sometimes you will need to reconfigure an FPGA with the Factory default configuration bit file manually after you configure the FPGA. with the User update configuration bit file. In that case, you can choose menu item "a" to manually configure the FPGA with the Factory default bit file.

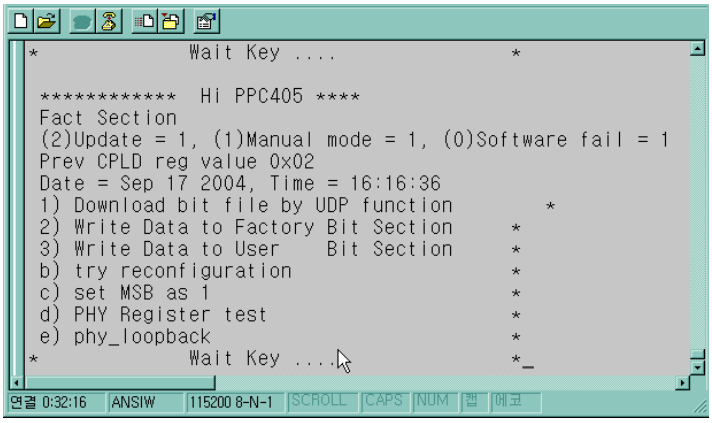


Figure 19: **Manual Reconfiguration**

In this case you can reconfigure with the User bit section again. Choosing ‘b’ will make CPLD reconfigure the FPGA with the User bit section.

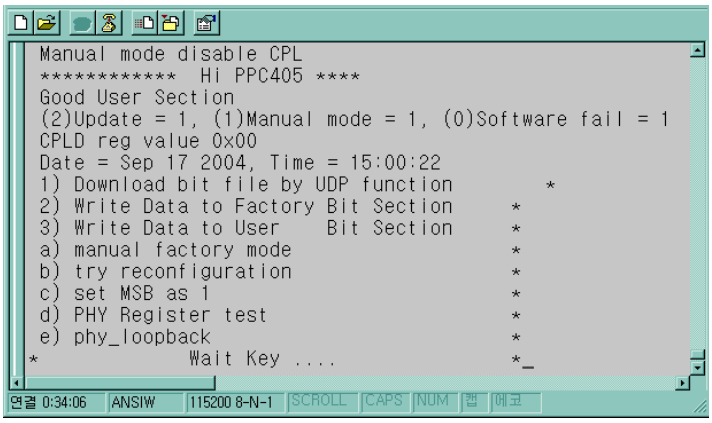


Figure 20: **Good User Section**

10. Bad User `.bit` File Download

If you download a **`.bit`** file in which some sections were broken already, the CPLD can detect a configuration fail, and then reconfigure the FPGA with the Factory default bit section. Choose ‘1’ to download the **`user_bitbad.bin`** file to the hyper-terminal, and then download it with **`client_echo.exe`**. After updating the User section, choose ‘b’ to reconfigure the FPGA. Because the file was broken, the done signal will not go high. The CPLD will watch the done signal, and, if it does not go high, the CPLD will use the Factory default bit section and reconfigure the FPGA. The CPLD will try three times to reconfigure the FPGA with the User section, so the time to wake up the FPGA will be longer than normal.

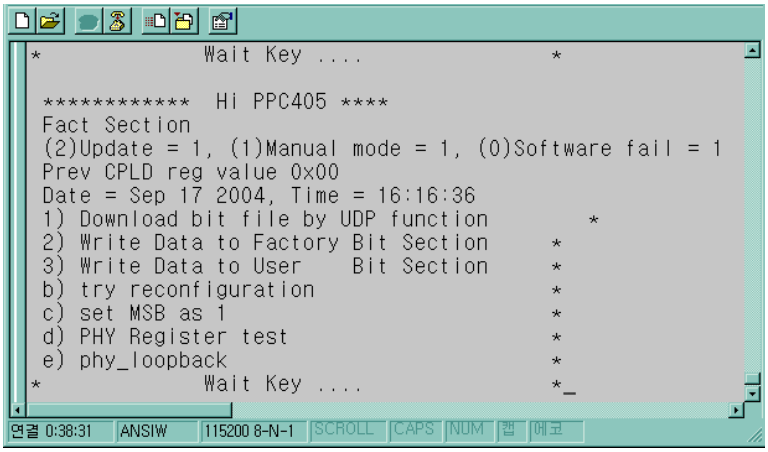


Figure 21: **Bad `.bit` File**

11. Bad User Application File Download.

In the CPLD there is one register which should be set to ‘1’ by software to let the CPLD know the software has loaded successfully. If it does not set to ’1’, the CPLD watchdog function will think there is a software application problem. The CPLD will then reconfigure the FPGA with the Factory default configuration `.bit` file. The total time for the application software to write ’1’ to the CPLD is about 10 seconds in this example, and you can adjust it by modifying the counter bit width on the CPLD design.

12. Update User Section with `user_appbad.bin` File and Reconfigure the FPGA.

The software displays the “BAD User Section” message on hyper terminal

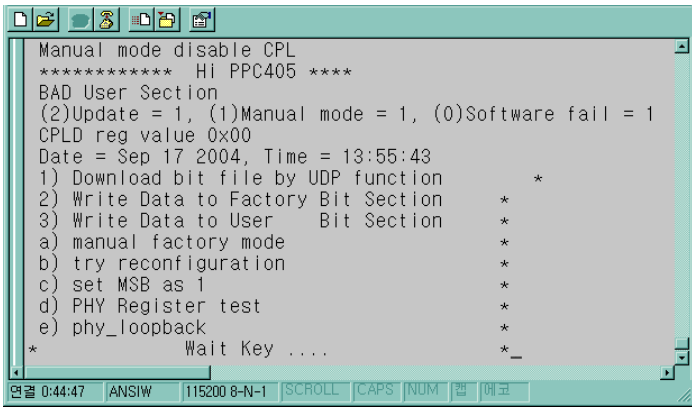


Figure 22: **BAD User Section**

For about 10 seconds, the message shows “Factory Section.” This means the CPLD is reconfiguring the FPGA with the Factory default section. Because the software failed to set the software loading timeout register in the CPLD to ’1,’ the watchdog function recognizes the software loading timeout situation and tries to reconfigure the FPGA with the Factory default configuration `.bit` file..

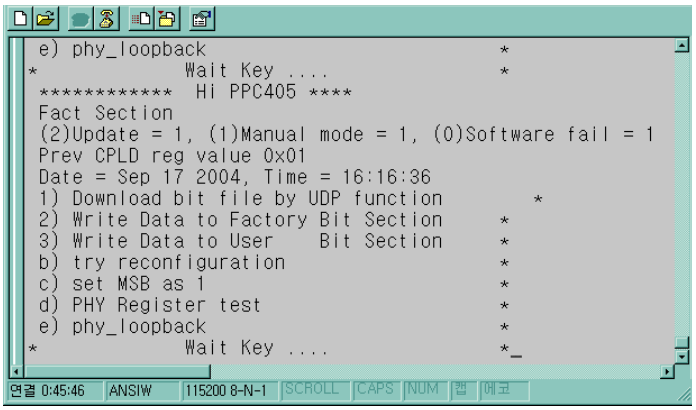


Figure 23: **Fact Section**

Special Notes

- Do not connect the CPLD JTAG pins with the FPGA JTAG pins when using the PPC405 JTAG pins, because this will cause a TAP controller error during the Flash write operation.
- Xilinx recommends the Insight Virtex-II Pro board rev. 4 for the demonstration, and the custom RFCS demo board.
- Revisions 1 and 2 of the Insight Virtex-II Pro development board need modifications to use for this demonstration. If you want to use Rev. 1 or 2, then the “dout” pin of the PROM should be disconnected from the board, and you should not connect any flying wires to the System Ace daisy chain because daisy chain signals from Memec boards are connected to the P-160 connector in Rev. 1 and 2 boards, while Rev. 4 boards do not have a connection between them.

Future Plan for Improvement RFCS

Xilinx is considering adding write protect and enable functions to the Factory section .

RFCS Demo Board

Target Insight Virtex-II Pro development board.

Xilinx designed a new board to replace P-160 Module for remote configuration demo.

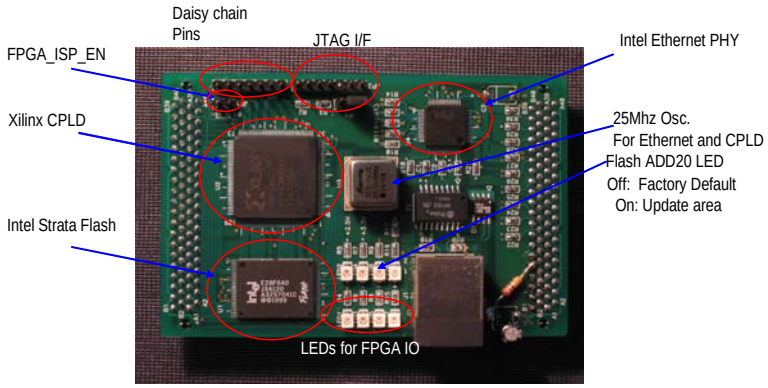
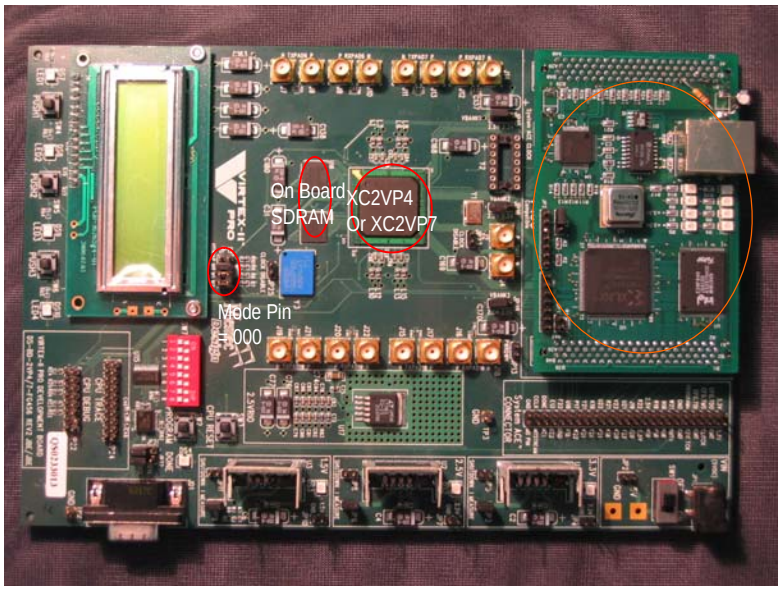


Figure 24: **RFCS Demo Board**

Xilinx designed a custom P-160 replacement module, which is called the RFCS board to test the PowerPC/MicroBlaze based remote configuration design. There is no SDRAM on the replacement module because Xilinx used the SDRAM on main Insight Virtex-II Pro board.



RFCS Demo Board

Figure 25: **RFCS and Memec Virtex-II Pro Development Boards**

Additional Information

Application Notes

[XAPP079: Configuring Xilinx FPGA Using an XC9500 CPLD and Parallel PROM]

[XAPP137: Configuring Virtex FPGAs from Parallel EPROMs with a CPLD]

[XAPP138: Virtex FPGA Series Configuration and Readback]

Reference Design Files on xapp441_designfiles.zip file

- FPGA XPS Processor Design
- FPGA Application Design
- CPLD Design : VHDL + UCF
- UDP Application
- User Application
- RFCS Demo Board Schematic

To obtain the design files, click here:

Design Files

Summary

This application note provides a system level block diagram, as well as descriptions of a complete processor reference design that allows sending new FPGA software and hardware bit-streams through Ethernet, saving them to Flash memory for FPGA reconfiguration. Also, a complete design file for a Xilinx CPLD design has been included to provide Xilinx FPGA device configuration via Master Serial mode, with a watch-dog function. This application note targets the Insight Virtex-II Pro board with a new P-160 daughter card for demonstration, but, it can be used in any other design, which requires the same features.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/18/05	1.0	Initial Xilinx release.
09/09/06	1.1	Add link to design files