Your Reconfiguration Is in the E-Mail

With Xilinx Internet Reconfigurable Logic technology and Virtex Platform FPGAs, you can perform fast and easy remote field upgrades via e-mail using microcontrollers.

by Marc Defossez
Senior Staff Applications Engineer
Xilinx, Inc
marc.defossez@xilinx.com

Since the beginning of the FPGA technology, Xilinx has pushed the boundaries of reconfiguration. In earlier FPGA families, it was only possible to reconfigure the whole FPGA. With the introduction of the Virtex™ FPGA families, it became possible to partially configure or partially reconfigure an FPGA. It is also now possible to reconfigure a remote FPGA via the Internet using Xilinx Internet Reconfigurable Logic (IRL™) technology. However, only a few companies and a few of all FPGA designs make use of IRL technology, because of the perception it is expensive, complicated, and mostly a proprietary solution.

What if we could securely reconfigure FPGAs in the field simply by sending an e-mail message? In this article, we will show you just how easy and cost-effective that can be.

Protocol Stack

Xilinx IRL reconfiguration technology uses the same transmission protocols as everyday Internet e-mail:

- **TCP/IP** – Transmission Control Protocol over Internet Protocol transports the e-mail over the Internet to its destination.
- **SMTP** – Simple Mail Transfer Protocol is used to deliver the messages.
- **POP3** – Post Office Protocol 3 retrieves the messages.

Figure 1 shows a complete Internet protocol stack. Each layer of the protocol stack is an abstraction level hiding details from other layers on top or below. For example, the network access layer does not need to know what kind of data it is carrying. Whether the data is video, voice, or other, it is unimportant to the network access layer. The only thing this layer needs to do is deliver the data in good quality to the upper layers.

- **Application Layer** – This is the user interface layer. The POP3 and SMTP protocols necessary for IRL technology to work reside in this layer.
- **Transport Layer** – This layer implements reliable, full-duplex communication over the Internet. TCP works in this layer.
- **Internet Layer** – Addressing and routing of data happens in the Internet Layer, where IP is implemented.
- **Network Access Layer** – Also called the "link layer," this is where the hardware interface is managed.
- **Physical Layer** – This is the actual medium for communication across great distances. Such transmission media include co-axial cable, phone lines, fiber optics, and wireless broadcasting.

Implementation

If you want to implement the Internet stack into an FPGA as hardware, it will:

- Take a lot of time (including VHDL or Verilog design and simulation).
- Require a robust FPGA.
- Consume a lot of money.

On the other hand, microcontrollers are good for protocol handling and can mitigate the time and cost of building an IRL solution for the remote reconfiguration of FPGAs.

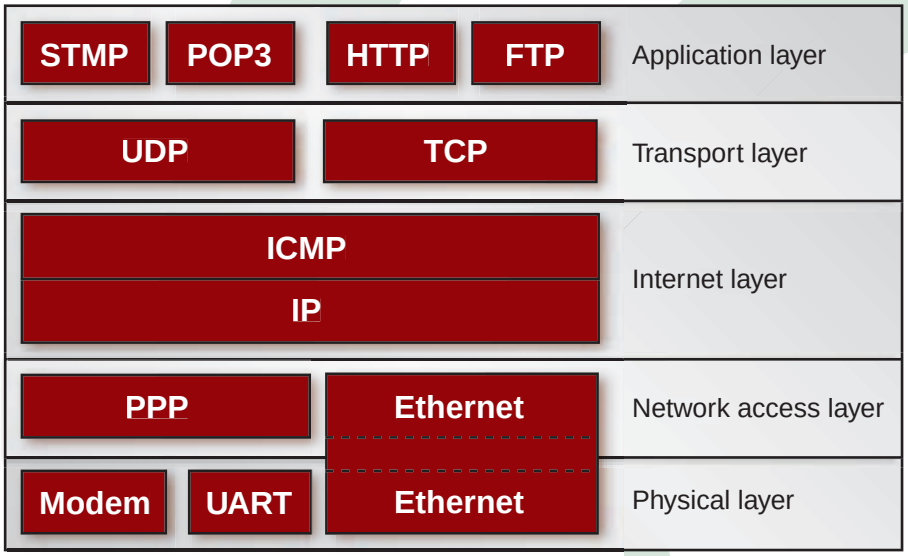


Figure 1 - Internet protocol stack

Two microcontroller solutions are possible:

- Use external microcontrollers.
- Put a microcontroller inside a Virtex Platform FPGA.

There are two ways to embed a microcontroller in a Virtex device.

- You can use the software MicroBlaze™ microcontroller in Virtex, Virtex-E, Virtex-II, or the new Virtex-II PRO™ Platform FPGAs.
- You can buy a Virtex-II PRO Platform FPGA with a hard-wired IBM® PowerPC™ 405 microcontroller already onboard.

External Microcontrollers

Several microcontroller manufacturers have Internet capable components. For instance, two Ubicom microcontrollers – SX52BD and IP2022 – can be used for IRL applications. Such external microcontrollers require "virtual peripherals" from Ubicom and some small modifications and additions to control downloading to an FPGA.

These small controllers and peripherals make the implementation of the TCP/IP, SMTP, and POP3 components of IRL technology easy and fast. Due to the small amount of internal memory of the microcontrollers, however, the Internet protocol stack is tuned to only perform the necessary functions.

Basic Setup

The simplest setup consists of an FPGA and a small controller, as shown in Figure 2. Here's what happens when the system is powered up:

- The FPGA is empty.
- The microcontroller waits for a certain time until all components of the IRL design have reached a stable state.
- Then the controller connects to the network by sending AT commands to an external modem through an RS-232 device, or by sending AT commands to an onboard chip modem, or by other physical implementation.
- Once the link has been set up with the e-mail server, the microcontroller asks if there is e-mail available. When there is, the microcontroller checks the e-mail header.
- If the header is not of a particular type, the controller will delete the mail message on the server.
- When the e-mail has the correct header type, the microcontroller downloads it. The /PROGRAM pin is toggled, the contents are serialized onto an output pin, and a bit clock is generated.
- When the DONE pin goes High, the microcontroller deletes the e-mail on the server.
- The microcontroller breaks the connection.

If the DONE pin does not go High after a period of time, the download operation is repeated until the DONE pin goes High.

Although this is the simplest approach, it has its shortcomings:

- An Internet connection is obligatory.
- The server must always have mail ready for the application, or else the application cannot start.
- When configuration fails, no fail-safe recovery mechanism exists.
- The design can go into an endless loop trying to download its configuration.

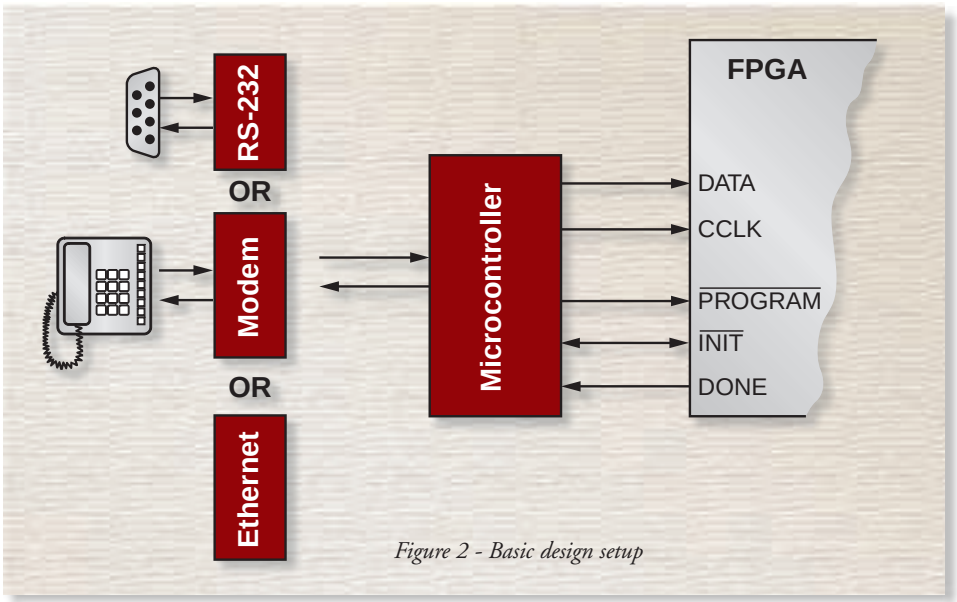


Figure 2 - Basic design setup

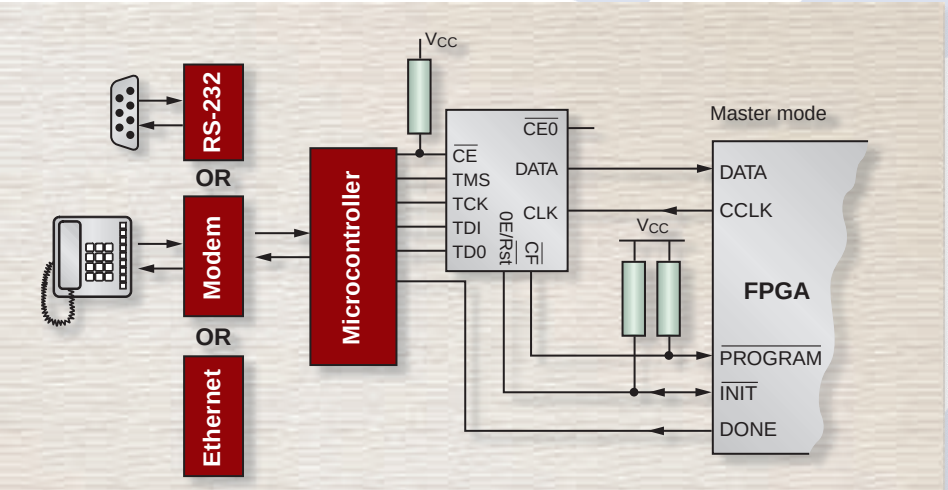


Figure 3 - Safe download

Fail-Safe Setup

You can make the download more reliable by storing the downloaded bitstream into semi-permanent memory (flash RAM). The FPGA can then be reconfigured from the flash memory. See Figure 3.

An even more secure solution is to work with two memories. A basic configuration can be loaded into the FPGA when it is shipped from the manufacturer. During operation in the field, the microcontroller can connect to the Internet and download a new configuration into the second memory. The new configuration bitstream would be downloaded into the FPGA at next boot.

When the download works, the new configuration will be used. If the new programming bitstream fails, the microcontroller will boot again from original memory.

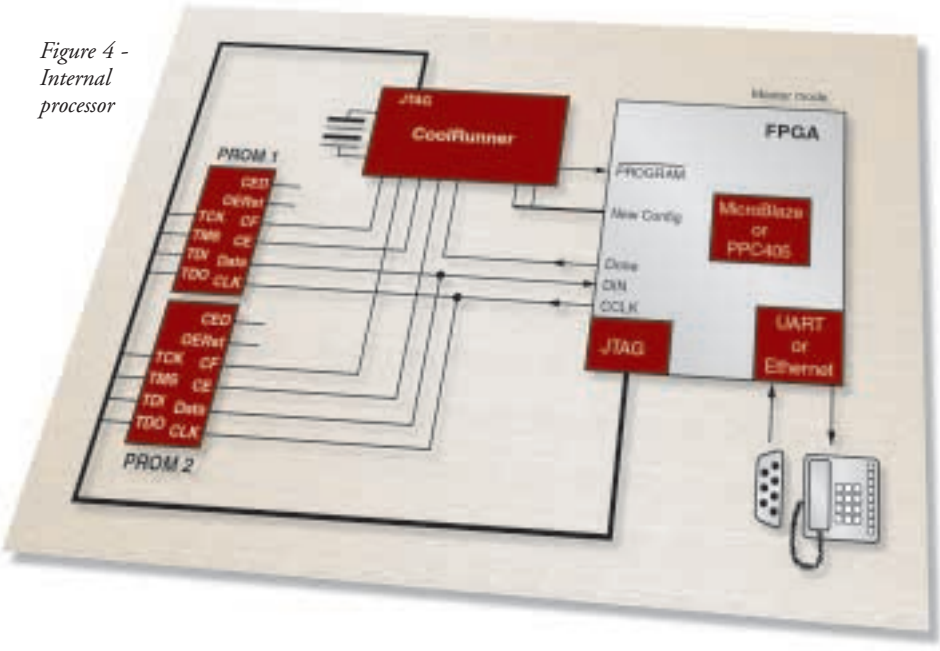
Internal Microcontrollers

Internal microcontrollers for FPGAs come in the form of MicroBlaze software and the hardware-embedded PowerPC 405 processor in Virtex-II PRO devices. (As a point of interest, the Virtex-II PRO Platform FPGA can be configured with both the MicroBlaze software and the PowerPC 405 processor, but that is beyond the scope of this article.)

While there are significant advantages of having microcontrollers onboard Virtex Platform FPGAs, there are also factors, in this design case, that require special consideration:

- In both cases of soft and hard microcontrollers, the Internet protocol stack must be programmed (ported) onto the FPGA.
- The FPGA must get a basic (possible partial) bitstream to download the MicroBlaze controller, its memory, and peripherals.
- The PowerPC Virtex-II PRO Platform FPGA must have memory and peripherals downloaded before it's able to boot.
- The small control algorithm that was done in the previous description within the external microcontrollers must now be implemented in a small CoolRunner™ CPLD, as shown in Figure 4.

Figure 4 - Internal processor



In either case, the first bitstream must contain the basic application of the FPGA. This way the system can operate as a standalone unit without problems. However, during operation, the FPGA microcontroller can contact the Internet and download new bitstreams as they become available.

The "basic setup" for external microcontrollers described above cannot be applied to internal microcontrollers, because the downloaded bitstream must be stored by the FPGA in some semipermanent (flash) memory. The final design of embedded microcontrollers can have different levels of fail-safe operation, depending on the system requirements.

Conclusion

It is possible to perform IRL operations in a safe and inexpensive way using external or internal microcontrollers – and some extra hardware. Xilinx IRL technology can now be used in almost any design that requires field upgradability. Reconfiguration by e-mail eliminates the need for complex and expensive design solutions. Furthermore, with a little ingenuity, the IRL solution can be extended to designs that were not previously considered for field upgradability. Application notes with detailed solutions will be posted in the coming months on the Xilinx website: *www.xilinx.com/apps/appsweb.htm.* Look for "Virtex FPGA" or "FPGA Configuration."