Linux Based Ethernet Communication for Xilinx FPGAs

Rodrigo Souza[1], Golberi Ferreira[1], André Fidalgo[2]
[1]*IFSC, Florianópolis, Brasil ;* [2]*ISEP, Porto, Portugal*
rodrigo.souza@ifsc.edu.br, golberi@ifsc.edu.br, anf@isep.ipp.pt

Abstract

This article presents the implementation of an Ethernet communication platform for use on Xilinx FPGAs. The proposed solution relies on a synthesized embedded system to provide network data transfer and control capabilities, for use with synthesizable electronic devices. Most TCP/IP stack services and protocols were implemented and the design is flexible to allow adaptation and/or expansion for different application scenarios. Currently this platform is being used on the development of a FPGA based JTAG controller, with remote access. The embedded system hardware requires a MicroBlaze softcore microprocessor running a Petalinux operating system.

1. Introduction

Computer networks and the Internet are increasingly present in order to meet the demand for quick real time access to information from remote locations. This is often true for common electronic products and particularly important for high-tech industries, were time-to-market and quality concerns are a major competitive factor. It can be very important and sometimes critical that an electronic device provides access to computer networks for data communication and remote control. In this context, the inclusion of network access on reconfigurable electronic devices is fundamental.

FPGAs are frequently used for prototyping and actual implementation of electronic devices, particularly if these are commercialized on small scale. One good example would be medium performance digital testers which can be modified to adapt to different target devices, better FPGAs or different customer needs. Although several FPGAs already include Ethernet port support, their use is often difficult to implement and/or requires expensive proprietary solutions, which are often difficult to customize or reuse.

Our objective was the implementation of a communication platform that would allow network access via the Ethernet port present on most Xilinx FPGAs, that could be synthesized together with user designed electronic devices (in VHDL format) and requiring only Xilinx development tools and open-source software. The embedded system concept is presented on Figure 1.

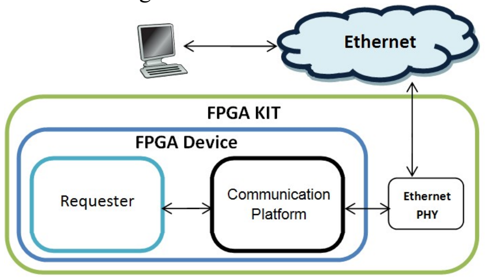

Fig. 1. System Concept

The Requester represents a generic electronic device, implemented on the FPGA, which requires and/or provides network services. The interface between the requester and the communication platform requires two FIFO devices for data buffering, both to address synchronization problems and avoid loss of data, while maintaining compatibility with most target devices. For communication between the Requester and an external computer it is necessary to implement the required TCP/IP layers protocols [1]. In our proposal this is performed by a synthesized embedded system, developed with Xilinx Platform Studio (XPS) [2], built around a MicroBlaze softcore microprocessor [3], running the Petalinux embedded Linux release [4]. This solution was chosen due to the academic focus of this implementation, the user-friendly interface of XPS, which facilitates the manipulation of MicroBlaze embedded system and the portability of Petalinux for Xilinx FPGAs. Furthermore, the wide range of networking features and the ease of integration of C applications provided by Petalinux was also relevant. XPS offers a transparent implementation process, which allows the direct use of most Xilinx FPGA models and a direct link with the Integrated Software Environment (ISE).

The communication platform concept and features are presented in section II while section III presents its implementation details. Section IV presents a brief result analysis, and in section V we draw some conclusions from the ongoing work.

2. Communication Platform

The communication platform consists of an embedded system synthesized into a FPGA, involving hardware and software components [5]. The TCP / IP protocols and most other necessary functions run on a microprocessor, therefore being mostly implemented in software.

A. The embedded system hardware

The main hardware component of an embedded system is the microprocessor. There are several types of microprocessors available that can be implemented in Xilinx FPGAs as the ZPU, MB-Lite, Wishbone High Performance Z80, among others. However, an embedded system requires other components like buses, RAM and ROM memories, and other specialized devices such as a MAC network controller. The integration of all these devices within an FPGA and the synthesis of the working embedded system is a complex task that requires considerable effort and/or experience.

The Xilinx XPS tool was selected as a development environment. It relies on the use of intellectual property (IP) cores from Xilinx, like the MicroBlaze softcore microprocessor , the PLB bus, the MAC network controller (known as Ethernet Lite), as well as external devices connected to the FPGA, such as RAM and FLASH memories, to build all the embedded system hardware.

A critical IP core is the General Purpose Input/Output (GPIO), which has a configurable number of data ports, where any device synthesized in the FPGA can be connected. This allows the use of the GPIO to interconnect any requester device (or devices) with the communication platform.

B. The embedded system software

The embedded system performs its functions in accordance with the software running in the microprocessor. The operating system and running applications must implement the required functions and protocols of the TCP / IP stack and manage communication with the requesting device. The Petalinux (from Petalogix) was chosen as the operating system, being a commercial Linux distribution for microprocessors implemented on Xilinx FPGAs, including the MicroBlaze softcore microprocessor and PowerPC 405 and 440 hardcore. The Petalinux has drivers for multiple IP cores from Xilinx, drivers for the GPIO and the Ethernet Lite, as well as native TCP/IP stack with various protocols such as ARP, TCP, UDP, ICMP, DHCP, and services such as HTTP, Telnet, SSH, FTP, among other valuable network features.

In addition, the Petalinux provides facilities that allow adding a user space program to its command list. In this manner it is possible to write a user space program in C that manages communication with the requester device.

3. Implementation

The communication platform implementation is somewhat dependent on the requester device architecture, because the GPIO ports must be configured in a way that allows its interconnection. Given this fact, we defined a common scenario, where a remote computer uses Telnet and FTP services (hosted on the embedded system), to send and receive hexadecimal data (directly or in file format) to the requester device (synthesized on the FPGA), via an Ethernet network. This represents the typical environment and is sufficient for most applications.

The requester device has its own internal logic, but must implement, two additional FIFOs, for data buffering, called GPIO_to_FIFO (input buffer) and the FIFO_to_GPIO (output buffer). The internal logic of the requester device receives data from GPIO_to_FIFO and sends data to FIFO_to_GPIO.

The communication platform implements two communication ports which can be connected to any module present on the FPGA, including user made modules. These are called GPIO_OUT and a GPIO_IN and are connected respectively, to GPIO_to_FIFO and FIFO_to_GPIO on the requester device. The connection thus created allows the communication of data between the remote computer and the requester device, via the GPIO ports and buffer FIFOs. The prototype scenario was based on a simple requester device, coded in VHDL and designed to control some LEDs and read the state of several switches, on the FPGA. This device and the communication platform were synthesized together on a Spartan 3E500 Starter Kit, using Petalinux v0.40 and the ISE and XPS 11.5 version.

A. Hardware Implementation

As explained, the communication platform hardware is implemented using XPS. However, the system is not fully synthesized within the FPGA, and other devices in the Spartan 3E500 Starter Kit (connected to the FPGA) are also part of the embedded system (e.g. the Ethernet Port). Even so, most of the embedded system devices are synthesized within the FPGA and connected to the PLB bus. All the devices connected to the PLB are slaves controlled by the MicroBlaze which is the master. Figure 2 illustrates the embedded system architecture on the Spartan 3E Starter kit, showing how hardware components are connected and which are synthesized within the FPGA and which are autonomous ICs.

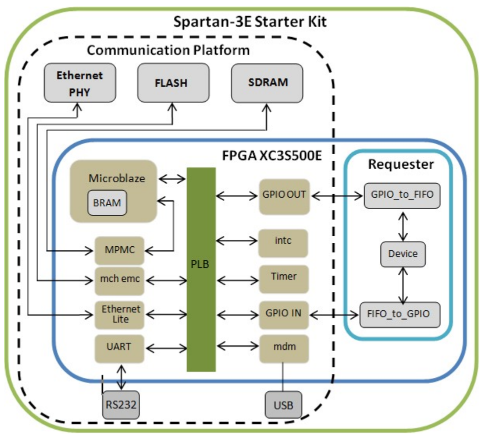

Fig. 2. Embedded System Architecture

To connect the requester device to the GPIO port is necessary to export the embedded system design created in XPS to ISE. This is required to connect the requester device ports to the embedded system GPIO port.

B. Software Implementation

In the communication platform, the Petalinux is responsible for managing and implementing the communication with the requester device, as well as executing part of the TCP/IP stack functions. Thus, its components must be configured and implemented according to the type of network communications required and in compliance with the requester device architecture. The Petalinux System Development Kit (SDK) is a required tool to configure and compile Petalinux for the target embedded system. The tool has, among other features, a setup menu, automation scripts of Petalinux configuration routines and implementation and integration with the Xilinx XPS.

1) Configuration of the TCP/IP stack: The Petalinux has several TCP/IP stack services and protocols that can be added or removed through the setup menu. Some of them, such as TCP, UDP, IP, ICMP, DNS, web server, Telnet server and FTP file transfer server, are added by default during cross compiling. However, there are other protocols, which may be enabled on setup menu (e.g. DHCP). Despite this, Petalinux implements only a portion of the TCP/IP stack layers. The remaining are implemented by a MAC network controller (Ethernet Lite), and by the IC responsible for the interface with the analogue environment (Ethernet PHY). The Table 1 shows which layers are implemented by the Petalinux, the MAC network controller and the Ethernet PHY, as well as some of the protocols stipulated, according to the proposed implementation scenario.

Responsible	TCP/IP stack layers		Protocols
Petalinux	Application		Telnet, FTP, DHCP
	Transport		TCP, UDP
	Internet		IP, ICMP
	Host-to-network	LLC	Ethernet
Ethernet Lite		MAC	
Ethernet PHY		Physical	

Table 1. TCP/IP stack implementation

2) Implementation of the Translator: The translator is the software that manages and executes the communication with the requester device through the GPIOs. Thus, the two main functions of the translator are: (1) send and receive data from the requester device and (2) format it so that it can be understood by the application layer service used in communication. To ease the data formatting task for the application layer service, the translator was developed in the C programming language, as a user space application and added to the Petalinux command list. In this way, the application layer services can easily access the translator, invoking the command line that calls it (in Petalinux), in order to send or receive data, passing the necessary arguments and handling command returns. Implemented this way, the translator can be accessed via Telnet commands. Considering the scenario proposed for the communication platform, six separate translator programs were developed, their functions being shown in Table 2.

Name	Function
writereset	Reset and clean FIFO IN memory
writefifo	Write hexadecimal data in FIFO IN
writefifofile	Write hexadecimal data file in FIFO IN
readreset	Reset and clear FIFO OUT memory
readfifo	Reads hexadecimal data from FIFO OUT
readfifofile	Stores in a file the hexadecimal data read from FIFO OUT

Table 2. Program Translators

These programs were designed to be flexible and can be directly used (or adapted to) by different requester hardware. The operational logic of the six programs is based on the FIFO operation paradigm, whereas the GPIOs are connected to the FIFOs, and also adjusted to the function of each program, which can be the return of a reading or the storage of such data on a file. To access the GPIO, the program translators use the IOCTL commands from xgpio ioctl.h library, which is the C library on Linux for Xilinx GPIO.

The GPIO speed of operation depends on how long it takes for MicroBlaze to process the IOCTL instructions of the translator program. That time is not deterministic, since the microprocessor may have to handle multiple tasks that, in a non real-time operating system, may require a variable number of machine cycles. Therefore, the speed of reading and writing data from/to the GPIO is not precise or even constant. In the proposed scenario, the readfifofile translator program was able to read data from FIFO OUT at a speed of up to 3.3 kHz, sometimes decreasing to 2 kHz.

4. Results

The main outcome from the described work was the communication platform itself, which provides a technical path to implement connectivity via a computer network, for FPGA synthesized devices. The use of Xilinx tools and the modular design allows the use of the proposed solution on most Xilinx FPGAs, requiring only an Ethernet connector and the necessary logic area. It should be noted that on a Spartan-3E FPGA, which is a rather small FPGA for modern standards, the embedded system required less than 4300 Slice Flip Flops of the target FPGA. For this purpose, the Microblaze must be configured on a reduced configuration, without MMU, thus reducing the logic occupation of the communication platform. Once the systems is present, it is possible add additional GPIOs to connect other synthesized devices to the communication platform.

The main software contribution is the functional Petalinux port for the embedded system (created by XPS), with all required drivers. Petalinux proved to be robust and reliable, providing various computers network services and the required flexibility for applications development. Another important software contribution is the six translators programs that access and control the GPIO ports. These are invoked as Petalinux commands, being accessed by the application layer services. In practice this means that they can be invoked via the computer network.

The proposed plataform is curretly being used to implement a VHDL based JTAG controller, with network support [6]. Considering the robustness and flexibility of the platform, it is clear that it comprehensively addresses the communication needs of such devices, particularly if synthesized into FPGAs, which may prove to be an important contribution for the remote maintenance of electronic devices.

In short, the proposed communication platform is a very effective and versatile solution to provide FPGA synthesized devices with Ethernet network access, for efficient data communication and remote online control.

5. Conclusions

The validation of the communication platform evaluation scenario showed that the implementation of a TCP/IP stack, using an embedded system was accomplished. The communication platform implemented met effectively the required communication functionalities, using a common computer network. The use of XPS makes most concepts of hardware implementation of an embedded system practically transparent, including the FPGA model. Combining this with the flexibility and robustness of Petalinux, allows the creation of a simple environment, but one that is rich in features and implementation options. Basing the communication platform in these two tools allows the transfer of part of the transparency and ease of implementation to the communication platform itself.

If we consider a device that will use the platform to communicate with a network of computers, our proposal provides three important solutions: (1) the configuration of the TCP/IP layer services; (2) the implementation of the translators and (3) the configuration of the GPIO. The platform development has other requirements, which are related to operation, not functionality. The inclusion of Petalinux SDK and Xilinx XPS tools makes the development of the hardware and software included on the platform simpler and more transparent, which also translates into making the platform much more user-friendly (for users and developers).

As future developments on this area it is planned to automatize several development steps, like the translator implementation, the platform creation (on XPS) and the Petalinux compilation, among others. The GPIO operating frequency, controlled by Petalinux, is still a problematic factor for the platform, namely when the requester device requires high data rate communications. Thus, in order to improve platform performance, we are currently analyzing the possibility of connecting the requester device directly to the embedded system bus (and its clock signals).

References

[1] Tanenbaum, A.S., Computer Networks, 4 ed., Prentice Hall PTR, 2003

[2] Xilinx, Xilinx Platform Studio (XPS), 2010 [Online]

[3] Xilinx, MicroBlaze Soft Processor, 2010 [Online]

[4] Petalogix, PetaLinux SDK Docummentation, 2010 [Online]

[5] Rodrigo Neri de Souza, “Plataforma de Comunicacão Ethernet para dispositivos embarcados em FPGAs da Xilinx”, Undergraduate Thesis, IFSC, Brazil, 2010

[6] Ricardo Costa, “Controlador Boundary Scan”, Master’s Thesis, ISEP, Portugal,2010