ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 4, April 2012

Implementation of Multi-User FPGA Environment Over TCP/IP

Pansare S.D, S.M.Turkane

Abstract - Optimizing the single target device as a simulator in a multi user environment is our aim to achieve. It is important step to use FPGA to design and improve flow of multi user data in a high speed computer network environment. Hence, we create a system where multiple users (computers) will use one target device (FPGA) through Ethernet without the need of actually carrying target device to each and every user. For this we are using a controller module which consist of control logic and user interface module. It provides a more efficient way of utilizing a single board for hardware simulation. It could place itself in educational establishments where the prototyping board does not need to be changed from workstation to workstation and also provides an efficient way of managing expensive resources.

Index Terms— **Ethernet, FPGA, Picoblaze, SPI.**

I. INTRODUCTION

In this project we look at a simple type of hardware simulation strategy where multiple users can acquire the same board for their hardware simulation but at different time connecting through TCP/IP. We explain the FPGA development platform we use for controller, which includes the TCP/IP interface through Ethernet for communication with user. Fig.1 shows the system architecture that we will use. We will also see the future possibilities to reduce the queue and accessibility of the FPGA board problem. A user can connect to target device either by Ethernet or WI-Fi / Wi-Max [1], internet [2]. Out of these transmission Medias we will concentrate on use of Ethernet. We will use FPGA in both target device and controller.

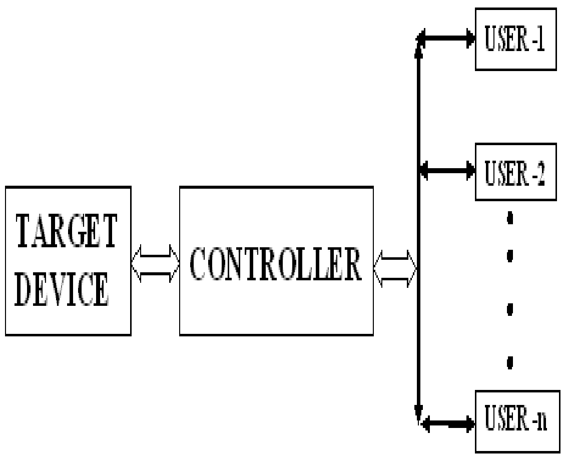

Fig1. System Architecture

The TCP/IP is an additional interface that is added to provide interface capabilities. The TCP/IP protocol uses a two-layer protocol. The higher layer, TCP, provides communication between the target and source and breaks application layer information into packets. TCP/IP provides two methods of data delivery but our board interface is only implemented for connection-orientated delivery using TCP. IP is the protocol responsible for addressing and routing packets between networks. It ensures they reach the correct destination network. IP deals with the physical Network interface layer and for this we use Ethernet. The Ethernet module we designed to fit our requirements is based on a standard module provided by WIZNET W5100. The module contains an Ethernet port.

II. TARGET DEVICE

Our aim is to share target device in a network. For this we have taken our target device as FPGA (SPARTAN 3E). Spartan-3E FPGAs are programmed by loading configuration data into robust, reprogrammable, static CMOS configuration latches (CCLs) that collectively control all functional elements and routing resources. The FPGA's configuration data is stored externally in a PROM or some other non-volatile medium, either on or off the board. After applying power, the configuration data is written to the FPGA using any of seven different modes: master Serial from a Xilinx Platform Flash PROM, Serial Peripheral Interface (SPI) from an industry-standard SPI serial Flash, Byte Peripheral Interface (BPI) Up or Down from an industry-standard x8 or x8/x16 parallel NOR Flash, Slave Serial typically downloaded from a processor, Slave Parallel, typically downloaded from a processor, JTAG, typically downloaded from a processor or system tester. Furthermore, Spartan-3E FPGAs support Multi boot configuration, allowing two or more FPGA configuration bit streams to neither be stored in a neither single parallel NOR Flash. The FPGA application controls which configuration to load next and when to load it. We will be using slave serial mode for our target device.

III. CONTROLLER

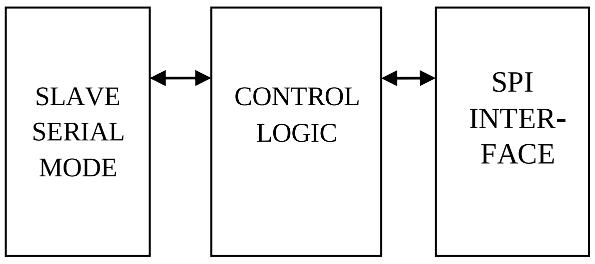

Fig2. Controller Architecture

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 4, April 2012

Control logic will be implemented using FPGA (SPARTAN 3E). Fig.2 shows the block diagram for controller architecture. Controller architecture will consist of three main building blocks. We will see these blocks in detail in following sections.

A. Slave Serial Mode

In Slave Serial mode, an external host such as microcontroller writes serial configuration data into the FPGA (target device), using the synchronous serial interface.

B. Control Logic

We are going to use soft processor core "PicoBlaze" an open source from Xilinx as our external host [3].This PicoBlaze soft processor core will be implemented in FPGA(control logic). The PicoBlaze microcontroller is a compact, capable, and cost-effective fully embedded 8-bit RISC microcontroller core optimized for the Xilinx FPGA families. The PicoBlaze microcontroller core is totally embedded within the FPGA and requires no external resources. The PicoBlaze microcontroller is extremely flexible. The basic functionality is easily extended and enhanced by connecting additional FPGA logic to the microcontroller's input and output ports. The PicoBlaze peripheral set can be customized to meet the specific features, function, and cost requirements of the target application. Because the PicoBlaze microcontroller is delivered as synthesizable VHDL source code, the core is future-proof and can be migrated to future FPGA architectures, effectively eliminating product obsolescence fears. Being integrated within the FPGA, the PicoBlaze microcontroller reduces board space, design cost, and inventory.

The PicoBlaze microcontroller is specifically designed and optimized for the Spartan-3 family and with support for Spartan-6, and Virtex-6 FPGA architectures. Because it is delivered as VHDL source, the PicoBlaze microcontroller is immune to product obsolescence as the microcontroller can be retargeted to future generations of Xilinx FPGAs, exploiting future cost reductions and feature enhancements. Before the advent of the PicoBlaze and MicroBlaze embedded processors, the microcontroller resided externally to the FPGA, limiting the connectivity to other FPGA functions and restricting overall interface performance. By contrast, the PicoBlaze microcontroller is fully embedded in the FPGA with flexible, extensive on-chip connectivity to other FPGA resources. Signals remain within the FPGA, improving overall performance. The PicoBlaze microcontroller reduces system cost because it is a single-chip solution, integrated within the FPGA and sometimes only occupying leftover FPGA resources. The PicoBlaze microcontroller is resource efficient. Consequently, complex applications are sometimes best portioned across multiple PicoBlaze microcontrollers with each controller implementing a particular function, for example, keyboard and display control, or system management. Microcontrollers and FPGAs both successfully implement practically any digital logic function. The program memory requirements grow with increasing complexity. Programming control sequences or state machines in assembly code is often easier than creating similar structures in FPGA logic. Microcontrollers are typically limited by performance. Each instruction executes sequentially. As an application increases in complexity, the number of instructions required to implement the application grows and system performance decreases accordingly. A microcontroller embedded within the FPGA provides the best of both worlds.

C. SPI Interface

When simulating over TCP/IP there is a certain amount of latency due to the distance between host PC and FPGA, the amount of traffic on the network, and the amount of data transfer in the simulation model to the FPGA. The clocking of the hardware simulation block can either be driven synchronously with the model, or the board independently clocks itself so that the inputs become asynchronous. There are two ways of connecting the FPGA over TCP/IP to the host computer. The first is that the host computer communicates with another computer that has a FPGA board attached to it. This client computer then communicates to the FPGA board via ISA. The client computer requires some sort of interface program to allow communication to the board. For hardware this is a TCP/IP service program that communicates with the FPGA board. This does add an extra layer of communication and the need for an extra computer. The second way is to connect directly to the FPGA board. The communication goes through an SPI interface [4].

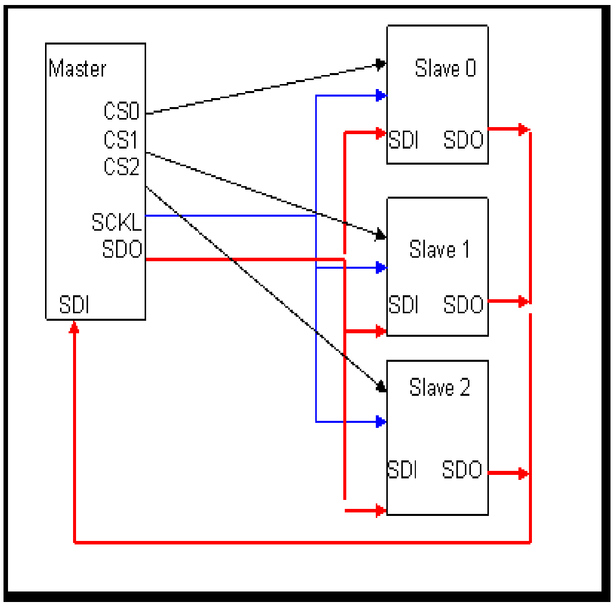

Fig.3 SPI interface block diagram

If independent slaves are to be connected to a master an other bus structure has to be chosen, as shown in Fig.3. Here, the clock and the SDI data lines are brought to each slave. Also the SDO data lines are tied together and led back to the master. Only the chip selects are separately brought to each SPI device.

IV. ETHERNET CONTROLLER

We are using W5100 as a Ethernet controller. It is a full-featured, single-chip Internet-enabled 10/100 Ethernet controller designed for embedded applications where ease of integration, stability, performance, area and system cost control are required. It has been designed to facilitate easy implementation of Internet connectivity without OS [5]. It is IEEE 802.3 10BASE-T and 802.3u 100BASE-TX compliant. Fig.4 shows the block diagram for our Ethernet controller. The W5100 includes fully hardwired, market-proven TCP/IP stack and integrated Ethernet MAC & PHY. Hardwired TCP/IP stack supports TCP, UDP, IPv4, ICMP, ARP, IGMP and PPPoE which has been proven in various applications for several years. 16Kbytes internal buffer is included for data transmission. No need of consideration for handling Ethernet Controller, but simple socket programming is required. For easy integration, three different interfaces like memory access way, called direct, indirect bus and SPI, are supported on the MCU side. TCP is the connection based communication method that will establish connection in advance and deliver the data through the connection by using IP Address and Port number of the systems.

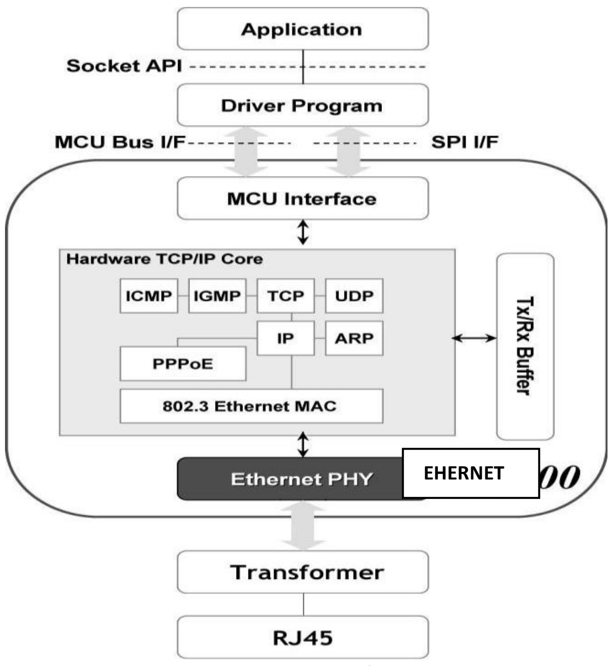

Fig.4 Block diagram of WIZNET

There are two methods to establish the connection. One is SERVER mode (passive open) that is waiting for connection request. The other is CLIENT mode (active open) that sends connection request to a server.

Process of using General SPI Master Device:

1. Configure Input/output direction on SPI Master Device pins.
   /SS (Slave Select): Output pin
   SCLK (Serial Clock): Output pin
   MOSI (Master Out Slave In): Output pin
   MISO (Master In Slave Out): Input pin
2. Configure /SS as 'High'
3. Configure the registers on SPI Master Device.
   SPI Enable bit on SPCR register (SPI Control Register)
   Master/Slave select bit on SPCR register
   SPI Mode bit on SPCR register
   SPI data rate bit on SPCR register and SPSR register (SPI State Register)
4. Write desired value for transmission on SPDR register (SPI Data Register).
5. Configure /SS as 'Low' (data transfer start)
6. Wait for reception complete
7. If all data transmission ends, configure /SS as 'High'

V. DEVICE OPERATIONS

The W5100 is controlled by a set of instruction that is sent from a host controller, commonly referred to as the SPI Master. The SPI Master communicates with W5100 via the SPI bus which is composed of four signal lines: Slave Select (/SS), Serial Clock (SCLK), MOSI (Master out Slave In), MISO (Master in Slave Out). The SPI protocol defines four modes for its operation (Mode 0, 1, 2, 3). Each mode differs according to the SCLK polarity and phase - how the polarity and phase control the flow of data on the SPI bus. The W5100 operates as SPI Slave device and supports the most common modes - SPI Mode 0 and 3.The only difference between SPI Mode 0 and 3 is the polarity of the SCLK signal at the in active state. With SPI Mode 0 and 3, data is always latched in on the rising edge of SCLK and always output on the falling edge of SCLK. There are only two data lines used between SPI devices. So, it is necessary to define OP-Code. W5100 uses two types of OP-Code - Read OP-Code and Write OP-Code. Except for those two OP-Codes, W5100 will be ignored and no operation will be started. In SPI Mode, W5100 operates in "unit of 32-bit stream". The unit of 32-bit stream is composed of 1 byte OP-Code Field, 2 bytes Address Field and 1 byte data Field. OP-Code, Address and data bytes are transferred with the most significant bit (MSB) first and least significant bit(LSB) last. In other words, the first bit of SPI data is MSB of OP-Code Field and the last bit of SPI data is LSB of Data-Field.

VI. CONCLUSION

It provides a more efficient use of utilizing a single FPGA board for hardware simulation. It could also place itself in educational establishments where the prototyping board does not need to be changed from workstation to workstation and also provides an efficient way of managing expensive FPGA resources. Now a day, this type of multi-user FPGA board use has much further development until it becomes a viable multi-user hardware simulation environment.

VII. FUTURE WORK

Future work, include the management of multiple boards and modules on a rack with only one TCP/IP address or Wi-Fi/ Wi-Max. This would allow remote control of a rack in a laboratory. If two or more users are waiting for a board and there are further constant

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 4, April 2012

requests for hardware co-simulation, the assignment of the next user is very much random and there is the possibility for a user never to acquire the board and to always be queuing means Deadlock can be there. To avoiding these problems we can use different available protocols supported by this process. This is currently unacceptable so therefore some sort of queuing stack needs to be implemented. Further milestones include queuing of multiple hardware co-simulations with an option for priorities and also batch processing of models. Early work also continues on the use of FPGA device management, whereby devices on a board populated with many FPGAs can be targeted individually.

REFERENCES

[1] Piyush Kumar Shukla,Dr. S. Silakari,Dr. Sarita S.Bhadoria,Prof. Anuj Garg," Multi-User FPGA - An Efficient Way of Managing Expensive FPGA Resources Using TCP/IP, Wi-Max/ Wi-Fi in a Secure Network Environment" in Proceedings of the 15th IEEE International Workshop on Rapid System Prototyping.

[2] Fallside, Smith," Internet Connected FPGAs", In Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, Napa Valley, USA, April 2000.

[3] PicoBlaze 8-bit Embedded Microcontroller User Guide, http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf.

[4] Daniel Denning, James Irvine, Derek Stark, Malachy Devlin "Multi-User FPGA Co-Simulation Over TCP/IP" in Proceedings of the IEEE International Conference on Field-Programmable Custom Computing Machines.

[5] WIZnet datasheet,http://www.wiznet.co.kr/UpLoad_Files/ReferenceFiles/W5100_Datasheet_v1.2.2.pdf.