Embedded Network SoC Application Based on the OpenRISC Soft Processor

Faroudja Abid, Nouma Izeboudjen, Leila Sahli, Sabrina Titri, Dalila Lazib, Fatiha Louiz
Centre de Développement des Technologies Avancées
Microelectronics and Nanotechnology Laboratory
Lotissement 20 Août 1956, Baba Hassan, Alger
Email: fabid@cdta.dz , abidfaroudja@yahoo.fr

Abstract—**This paper presents an implementation of an embedded network application into FPGA, based on the Opencores and Opensources approaches. The embedded system is part of a System on Chip (SoC) platform for Voice over Internet Protocol (VoIP) application. The system includes a hardware part and a software part which are linked to each other through a µCLinux operating system. The hardware part is represented by a SoC architecture which is mainly based on the OpenRISC OR1200 soft processor and some basic peripherals such as the standard 10/100 Medium Access Control MAC/Ethernet circuit for network connection, the Universal Asynchronous Receiver Transmitter (UART) for serial transfer, a debug unit for debugging purpose and a memory controller with two external flash and SDRAM memories. The SoC architecture is mapped into the VirtexII3000 FPGA development board. Results show that the network SoC architecture occupies 39% of logic resources and 32% of IOBs.**

The software development of the embedded network application includes two parts: Configuration and compilation of µClinux and programming of network application. In this part we have chosen as test application an embedded network TFTP (trivial file transfer protocol) client using the 10/100 MAC/Ethernet as network controller.

***Index Terms*— Embedded system, MAC/Ethernet, System on Chip (SoC), OpenCores, OpenRISC, Opensource, TFTP.**

I. INTRODUCTION

AN embedded system on chip (SoC) refers to a mini computer independent system where all essentials parts of computing are integrated into a single FPGA or ASIC chip and where the application is executed by a program which is loaded into an on chip memory or an off the shelf component. The embedded SoC solution aims to realize portable systems, while reducing power dissipation, chip interconnects and device size.

Now with the advance of the microelectronic technology, it is possible to integrate a whole system into a single FPGA
circuit. Thus, a new field which integrate VoIP solutions' into hardware solution is the MAC/Ethernet circuit which is used to guarantee an Internet connection. In traditional solutions a PCI network card is inserted inside a computer which is based on a general purpose microprocessor. In this situation, the network application competes equally in processing time with other applications causing an overload in the processing. In order to overcome this problem, solutions implemented in dedicated hardware, ASICs or FPGA are available [1], [2], [3]. These solutions allow that part of the processing, instead of being realized by the microprocessor of general purpose, now can be executed separately by a dedicated hardware.

In this paper, we propose a solution which not only integrates the MAC/Ethernet hardware component into FPGA but also the software of the network application is embedded into the system. A case study of TFTP protocol is taken as an example. The proposed approach is based on the use of the OpenCores and Opensources design approaches. The benefit of using such methodology is flexibility; reuse and accessibility of the IP (intellectual property) cores at free cost. Thus the cost of the whole VoIP system is reduced significantly.

In section II, the Open System Interconnection Reference Model (OSI) is presented. In section III, a presentation of the FPGA embedded design methodology is given and finally a conclusion in IV.

II. NETWORK STACK

The Open Systems Interconnection (OSI) Model is an abstract description for layered communications and computer network protocol design. It was developed as part of the Open Systems Interconnection (OSI) initiative. Figure 1 shows the OSI and TCP/IP (Transmission Control Protocol/Internet Protocol) models. In its most basic form, OSI model divides network architecture into seven layers which, from top to bottom, are the Application, Presentation, Session, Transport, Network, Data-Link, and Physical Layers. Regarding to the TCP/IP model the network architecture is divided to five layers: Application, Transport, Internet and Network Access Layer [7].

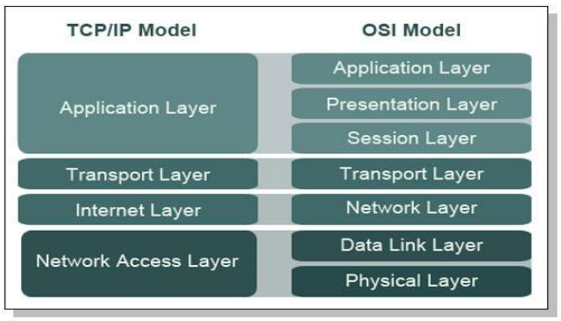

Fig. 1. OSI and TCP/IP Model.

Trivial File Transfer Protocol (TFTP) is a simple protocol to transfer files, with the functionality of a very basic form of File Transfer Protocol (FTP). FTP is one protocol of the application layer witch is a part of the Open System Interconnection Model. The TFTP has been implemented on top of the User Datagram Protocol (UDP). TFTP is designed to be small and easy to implement, it could be implemented using a very small amount of memory. It was therefore useful for booting computers such as routers which did not have any data storage devices. It is still used to transfer small amounts of data between hosts on a network, such as an operating system images. Therefore TFTP only reads and writes files (or mail) from/to a remote server. It can not list directories, and currently has no provisions for user authentication. The process of the file transfer is illustrated In Figure 2.

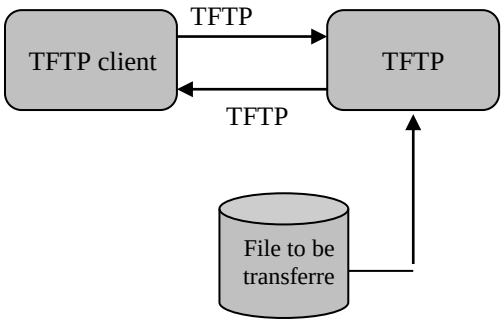

Fig. 2. File transfer process

III. PRESENTATION OF THE EMBEDDED DESIGN METHODOLOGY

Figure 3 shows the general architecture of an embedded system. This last one is composed of two parts: a hardware part which represents the system architecture which is mainly based on a processor and some peripherals components that communicate with each other through a suitable interconnect bus and a software part which is related to the application. A lot of approaches have emerged from industrial and academic research to design embedded systems into FPGA, such as the Xilinx approach which uses the Microblase processor, the Altera approach which is based on the use of the Nios processor, the IBM approach which uses the IBM processor and the Opencores approach which uses the OpenRisc processor. Each approach tries to promote its processor in the market. From of all theses approaches we have chosen the Opencores approach. This choice is justified by the following points:

- Availability of the cores and tools at free cost.
- Register Transfer Level (RTL) descriptions are given for all the cores or IPs (Intellectual property) components so that the whole system can be mapped into FPGA or ASIC; this allow flexibility and reusability of the cores.
- We can lean how to make an embedded system design reference from scratch.

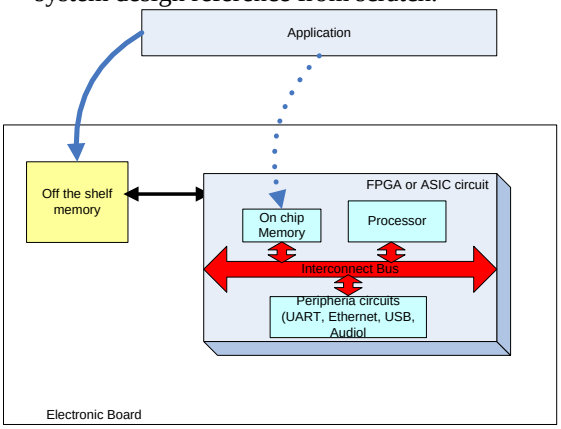

Fig.3 General view of an embedded system

Thus, based on the OpenCores and Opensource design approaches we have created our own platform for the design, implementation and test of embedded systems [4], [5]. Figure 4 gives an overview of the whole platform. The hardware part is related to synthesis, Place and root, Bitstream generation and downloading the generated file into FPGA circuit. The software part contains a set of tools such as the Or1k simulator, a GCC compiler, a GNU debugger that are used to debug and load the application into an on chip FPGA memory or an external memory depending on the application size.

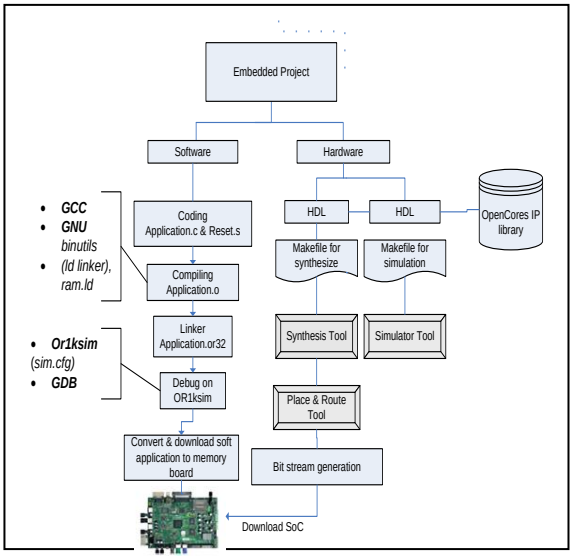

Fig. 4 Overview of the hardware/software platform

III.1 Presentation of Hardware Network SoC Architecture

We have developed an OpenRISC-based SoC platform that includes a 32bit reduced instruction set computer (RISC) processor optimized for implementation in FPGAs. The embedded network SoC includes OR1200 core and a minimum set of elements needed to provide network functionality.

These elements are debug unit for debugging purpose, a memory controller that controls an external memory that carry µClinux the open sources operating system and a network application., an Universal Asynchronous Receiver Transmitter (UART), the MAC/Ethernet that transmit voice packets over The Internet. All the cores are connected through the WISHBONE bus interface. The Internet connection is established by the opencores MAC/Ethernet.

For integration of all cores we created a SoC Verilog description. Figure 5 shows the block diagram of the network SoC architecture.

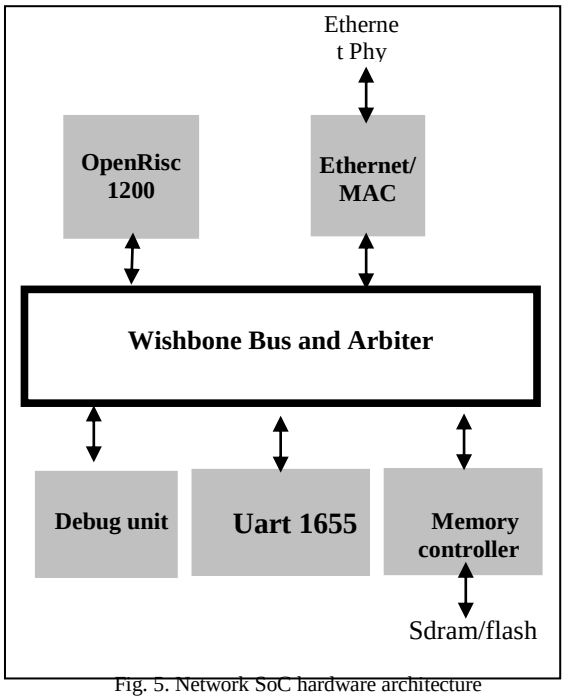

Fig. 5. Network SoC hardware architecture

III.1.1 Presentation of OpenRISC processor and the MAC/Ethernet

A. OpenRISC processor

The OR1200 core is a 32-bit scalar RISC with Harvard micro-architecture, 5 stage integer pipeline, virtual memory support (MMU). It includes debug unit for real-time debugging easing software development, a high resolution tick timer, a programmable interrupt controller and a power management support [8].

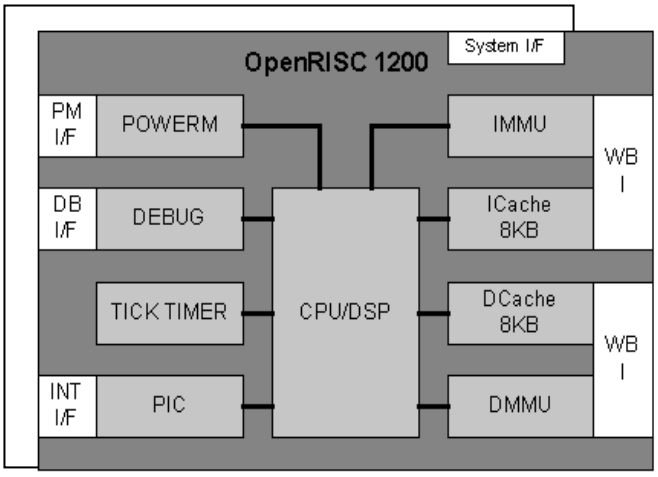

Fig. 6. OR1200 Architecture

B. Ethernet/MAC

The Ethernet core [6], [9] is a 10/100 Media Access Controller. It consists of synthesizable Verilog RTL core that provides all features necessary to implement the layer 2 protocol of the standard Ethernet. It is designed to run according to the IEEE 802.3 specification that defines the 10 Mbps and 100Mbps for Ethernet and Fast Ethernet applications respectively. In this work the Ethernet/MAC allows Internet connection.

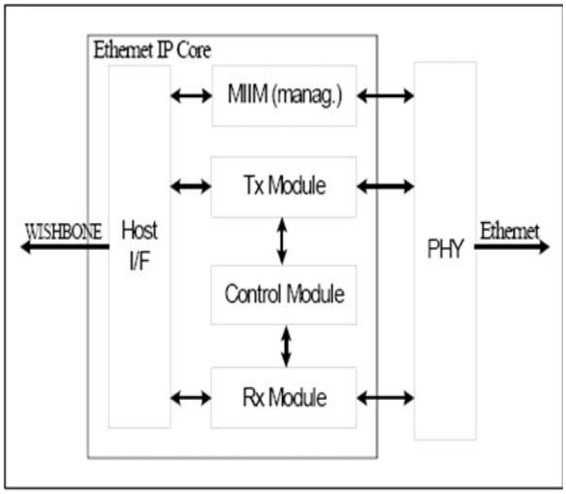

Fig. 7. MAC/Ethernet Architecture

Figure 7 shows the general architecture of the IP. It consists of several building blocks: a Tx module an RX module, a control module, a management block and a WISHBONE interface.

The TX and RX modules provide full transmit and receive functionality respectively. Cyclic Redundancy Check (CRC) generators are incorporated in both modules for error detection purposes. The control module provides full duplex flow control. The management module provides the

standard IEEE 802.3 Media Independent Interface (MII) that defines the connection between the PHY and the link layer.

Using this interface, the device connected can force PHY to run at 10 Mbps with frequency of 2.5 MHz versus 100 Mbps (25 MHz) or configure it to run at full versus half duplex mode. The WISHBONE interface connects the Ethernet core to the RISC and to external memory. To adapt this IP to our application we have determined the specification required to the VoIP application [5].

III.1.2 SYNTHESIS RESULTS

We have chosen as target FPGA the Virtex-II XC2V3000 to implement this system using the ISE 9.2i Xilinx [10] tool. As shown in table1 the network SoC architecture occupies 39 % of logic resources and 32 % of IOBs.

TABLE 1

SYNTHESIS RESULTS

NUMBER OF SLICES	5705 OUT OF 14336	39%
NUMBER OF BONDED IOBS	155 OUT OF 484	32%
NUMBER OF BRAMS	11 OUT OF 96	11%
NUMBER OF GCLKS	5 OUT OF 16	31%

III. 2 PRESENTATION OF THE SOFTWARE DESIGN

The software platform includes several tools [4] including a GCC Compiler, an assembler and a debugger that is used for system debugging and software development. First, the software part is used to test the embedded network application using the 10/100 MAC/Ethernet as network controller. In this part we have chosen as test application an embedded network TFTP loader.

The network application runs on µClinux (microcontroller Linux version) operating system [9]. The µClinux (Linux Kernel v2.0.38) is a port of Linux to systems without a Memory Management Unit (MMU) designed for small systems like microcontrollers and small microprocessors which are suitable for implementation in FPGA.

The µClinux operating system and the network application were stored in the RAM memory. The software development of the embedded system includes two parts:

1. Configuration and compilation of µClinux operating system
2. Programming the network application.

A. µClinux Configuration

First we have set configutration acording to our work, figures below show hardware configuration and TCP/IP networking configuration .

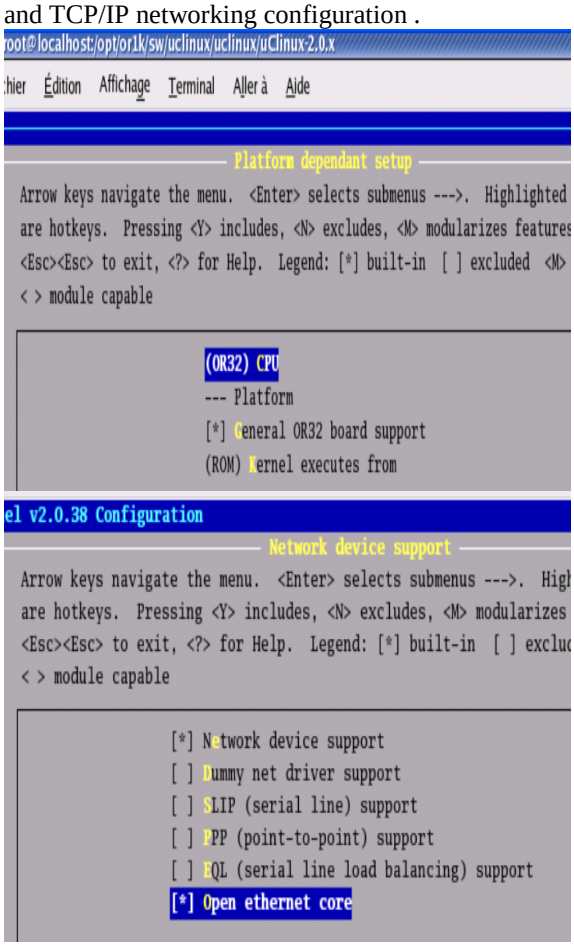

Fig. 8. µClinux CPU and network device configuration

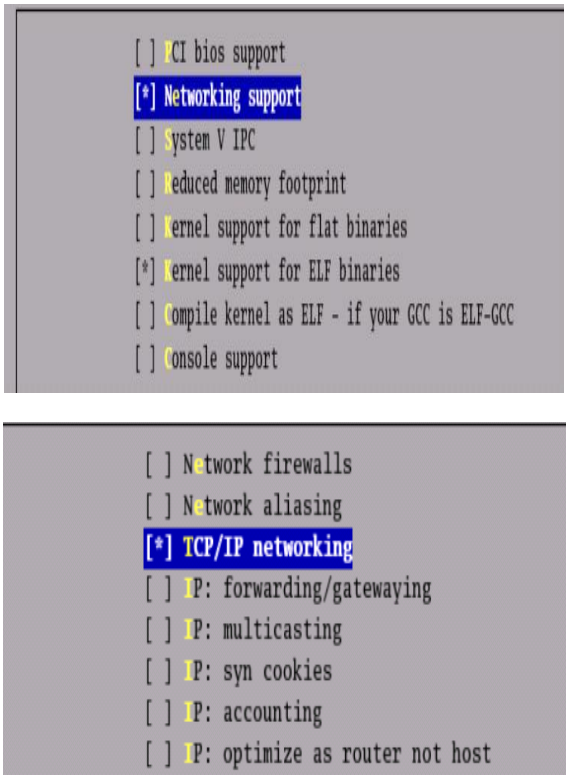

Fig. 9. µClinux network *Configuration*

B. Software simulation using Or1ksim Simulator

In order to run µClinux on Or1ksim simulateur [12] we configured Or1ksim to be used with µClinux via the configuration file *"sim.cfg"*.

The "*sim.cfg"* file contains the default configurations of peripherals and a set of simulation environments which are similar to the actual hardware situation. Mainly we specified the detailed behavior of the CPU (virtual memory, caches etc...), the various memories to be attached and the behavior of the various peripherals, in particular the MAC/Ethernet core which is used for the network application. Figure 10 shows the results of running µClinux on Or1ksim, the execution of the *"ifconfig"* command and the Internet protocols needed for a network application (ICMP (Internet Control Message Protocol) for ping, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) under µClinux.

Fig. 10. Running µClinux on Or1ksim

C. TFTP (trivial file transfer protocol) loader test application

In this test we have chosen the open source TFTP loader program with a lot of useful utilities and extra functions -- e.g. memory dump, flash loading, it useful and fast to load in most applications. It is accessible using serial port or VGA terminal. First we have adjusted the "*board.h"* file according to our system platform. The "*board.h"* file contains the hardware platform configurations. After that we have used the TFTP loader as shown in figure 11, to set the network configuration according to CDTA/LAN configuration (inet 10.1.70.74, netmask 255.255.0.0).

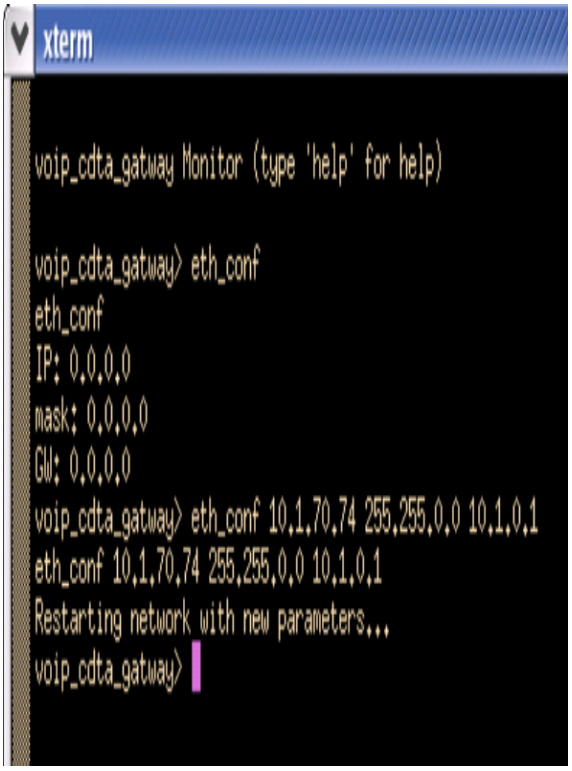

Fig. 11. Device network parameters set

IV. CONCLUSION

In this work we have developed an Openrisc soft processor based SoC solution for various network devices. The system constitutes a SoC that incorporates software and hardware part. To do this we created a SoC Verilog description. The synthesis results show that the whole network SoC architecture can be mapped into an FPGA Virtex-II XC2V3000 circuit. The network application based on the MAC/Ethernet and µClinux operating system is the first application tested. This last constitutes the main step in a VoIP application. Our next objective is to finalise the VoIP gateway integration in order to connect to the public phone system through a gateway, record and archive calls on a computer system and then test the VoIP gateway performances in Internet Network area.

REFERENCES

[1] A. Löfgren, L. Lodesten, S. Sjoholm, and H. Hansson. *Ananalysis of FPGA-based UDP/IP stack parallelism forembedded Ethernet connectivity*. In Proceedings of NORCHIPConference, Oulu, Finland, pages 94–97, 2005.

[2] F. L. Herrmann, G. Perin and al. "An UDP/IP Network Stack in FPGA", http://gmicro1.ct.ufsm.br/batista/images/stories/Artigos/Sforum_2009.pdf.

[3] K.Morita, , K. Abe, "Implementation of UDP/IP Protocol on FPGA and its performance evaluation", IPSJ General Conference. Special5, Pages 157-158.

[4] S.Titri, N.Izeboudjen, L.Sahli, D.Lazib "OpenCores Based System on chip Platform for Telecommunication Applications: VOIP".DTIS'07**,** International Conference on design & Technology of Integrated Systems in Nanoscale Era, Sep. 2-5, 2007., Rabat (Morocco), pp. 253-256.

[5] Abid Faroudja, Nouma. Izeboudjen, Sabrina. Titri, Leila.Sahli, Fatiha.Louiz, Dalila.Lazib "Hardware /Software Development of System on Chip Platform for VoIP Application ", ICM, International Conference on Microelectronics, December 19-22, 2009, Marakech, Morocco, pp 62-65.

[6] F.Abid, N.Izeboudjen, L.Sahli, S.Titri, D.Lazib, F.Louiz "Integration of the Opencores' MAC/Ethernet in a VOIP based system on chip application", ESC, Embedded System Conference, Mai 5-6,2009. Alger (Algeria).

[7] http://en.wikipedia.org/wiki/OSI_model.

[8] OPENCORES – Project: OpenRISC 1200. Available: http://www.opencores.org/projects/or1k/

[9] Igor Mohor "Ethernet IP Core Specification ", Rev. 0.4 October 29, 2002.

[10] XILINX ISE 9.2 user manual. www.xilinx.com

[11] http://www.opencores.org/projects/or1k/orpsoc/sw

[12] Jeremy Bennett "Or1ksim User Guide", Embecosm, 2008.