Towards an Open Embedded System on Chip for Network Applications

F. Abid, N. Izeboudjen, L.Sahli, D. Lazib, S.Titri, F. Louiz, M.Bakiri
Centre de Développement des Technologies Avancées
Lotissement 20 Août 1956, Baba Hassan, Alger
Email: fabid@cdta.dz, abidfaroudja@yahoo.fr

***Abstract*—** The embedded SoC (System on Chip) solution aims to realise portable systems, while reducing power dissipation, chip interconnects and device size. However the increasing complexity of embedded systems and the staggering costs associated with designing systems-on-chip imposes system designer and companies to seek collaboration on a variety of intellectual property issues. Consequently the cost of building SoC is increasing significantly when the system integrates several parts. This paper aims to exploit the advantage of a new approach and design method based on Opencores and Opensources design concepts in order to build in an embedded system on chip for network applications at free cost. This approach is based in the IP (Intellectual Property) reuse strategy facilitates the rapid creation and verification design process. In this paper we define the methodology adopted to construct the open SoC. The designed platform aims to provide a rapid prototype design for system on chip. The system includes a hardware part and a software part which are linked to each other through a µClinux operating system. For the hardware part, an HDL file describing all the cores of the library is created. The cores are configured as masters or slaves that communicate through the wishbone bus interface. Thus new SoC project can be created by adding, dripping or modifying a new core. The SoC architecture is mapped into the virtex5 *XC5VLX50-1FF676* FPGA development board. Results show that the SoC architecture occupies 27% of logic resources and 35% of IOBs (In/Out Blocs). The software part includes several packages, like the *GCC Compiler*, a *debugger* and an architectural simulator *Or1ksim* which is an instruction set simulator. The benefit of using the Opencores methodology is flexibility; reuse and the cores are available at free cost thus reducing the whole SoC cost.

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

I. Introduction

THE embedded systems are generally defined as special purpose computer-systems designed to perform one or more dedicated functions, usually with real-time constraints, and have computer hardware and software embedded as parts of a complete device or system, thus all essentials part of computing a system are integrated in a single chip and where the application is executed by a program, which is loaded into an on chip memory or an off shelf component. The SoC solution aims to reduce power dissipation, chip interconnects and device size.

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 integrates a network application into system on chip based FPGA circuit is emerging. This paper describes the embedded system on chip for network application which is part of the SoC platform based on Opencores and Opensources design concepts for Voice over Internet Protocol (VoIP) application [1], [2]. A Hardware/Software development of a system on chip (SoC) platform is described in [2].

One of the main components in a network application hardware solution is the MAC/Ethernet circuit (Media Access Controller) [3] 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 [4], [5] and [6]. These solutions allow that part of the processing, instead of being realised 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. Section 2 exposes the FPGA embedded design methodology. Section3 deals with the SoC hardware palatform and show synthesis, hardware simulation and implementation results of the embedded SoC. The Software design is presented in section4. Finally a conclusion in section 5.

II. FPGA Embedded Design Methodology

Figure 1 shows the general architecture of an embedded system. This 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. The hardware part and software parts are linked to each other through the OS (operating system).A lot of approaches have emerged from industrial and academic research to design embedded systems into FPGA. Among there approaches, the

Xilinx approach that 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 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 make an embedded system design reference from scratch.

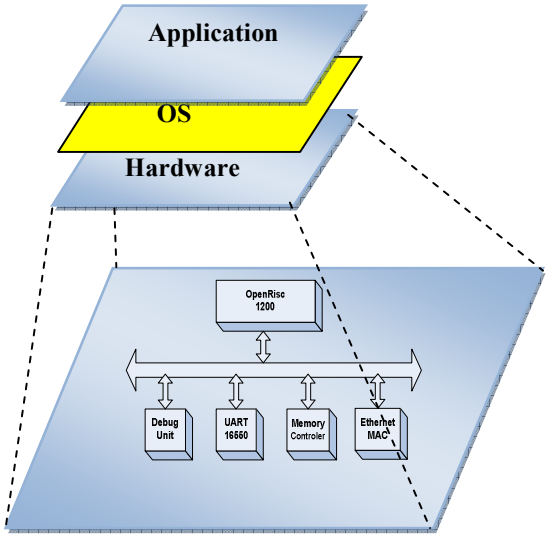

Fig.1 General view of an embedded system

Thus, Figure 2 gives an overview of our own platform created for the design, implementation and test of the embedded systems based on the OpenCores and Opensource design approaches. The hardware part is related to synthesis, place & roote, bitstream generation and downloading the generated file into FPGA circuit.

The software part contains a set of tools such as the Or1ksim 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.

For the hardware part, an HDL file describing all the cores of the library is created. The cores communicate through the wishbone bus interface.

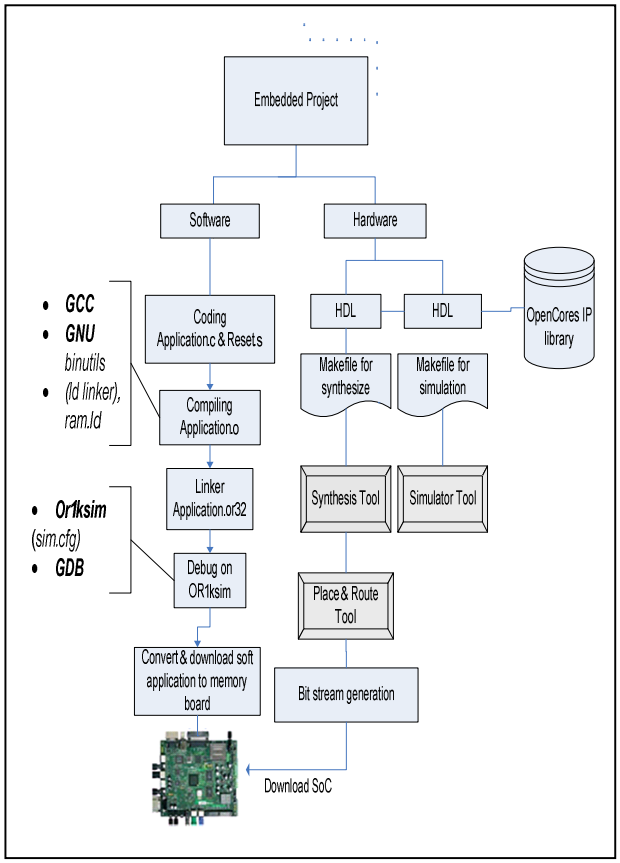

Fig. 2 Overview of the hardware/software platform

III. Hardware Platform

The designed embedded system is based on Xilinx ML501 development board with Virtex 5 FPGA [7].

Figure 3 shows the block diagram of the network SoC architecture. From the hardware point of view the embedded system on chip must be comprised of an embedded processor, embedded memory and required peripherals like Ethernet controller and input/output interface. In order to assure flexibility for different possible applications the embedded system was implemented on a FPGA platform. Soft-core processors like the Openrisc processor, Xilinx Microblaze, Altera Nios as well as any processor written in HDL languages can be implemented in FPGA devices. This avoids the need of a separate processor chip.

The soft-core processor configuration can be additionally tailored for specific application. Required peripheral devices are connected via wishbone bus. The embedded system must have sufficient memory to contain an operating system with a network application. In this work we have choosing µClinux the open source operating system.

We have developed an OpenRISC-based SoC platform that includes a 32bit Reduced Instruction Set Computer (RISC) processor [8]. The embedded network SoC includes the OR1200 core and a minimum set of elements needed to provide network functionality. These elements are the debug

unit for debugging purpose, a memory controller that controls an external memory that carries μClinux and a network application, an Universal Asynchronous Receiver Transmitter (UART), the MAC/Ethernet that transmit voice packets over the Internet. The Internet connection is established by this IP, all the cores communicate through the wishbone bus. The most difficult task in designing SoC hardware is how to write an HDL code that integrates all the SoC components codes and how these components communicate each with other. We have set the OR1200 processor as MASTER for all components. The MAC/Ethernet and Memory controller are configured as slaves compared to the OR1200 and a MASTERs to the UART and debug unit. For integration of all the cores we have created a SoC Verilog description.

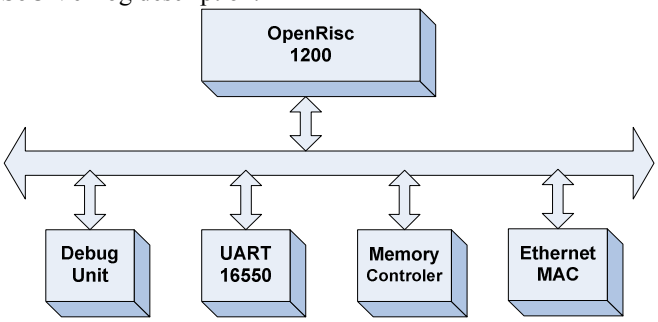

Fig. 3 Network SoC hardware architecture

III.1 PRESENTATION OF OPENRISC PROCESSOR AND THE MAC/ETHERNET

The Openrisc processor and the MAC/Ethernet cores are the basic IPs in our architecture, thus we give some details of their architecture.

A. OpenRISC processor

The OpenRISC 1200 is a synthesizable processor developed and managed by a team of developers at OpenCores [9]. OpenRISC 1200 is a 32-bit RISC processor implementing the 32-bit OpenRISC 1200 architecture. An overview of the OpenRISC 1200 processor architecture can be seen in Figure 4. The processor uses big endian byte ordering. The processor is intended for embedded, portable and network applications. OpenRISC 1200 is an open source IP-core freely available from the OpenCores website as a Verilog model, licensed under the GNU LGPL license.

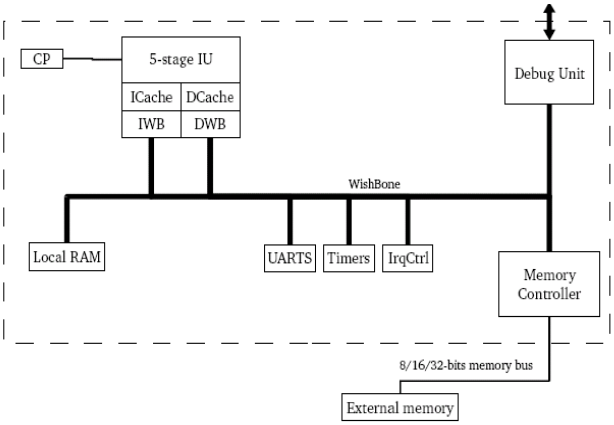

Fig. 4 OR1200 Architecture

B. Ethernet/MAC

The Ethernet core [10], [3] 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.

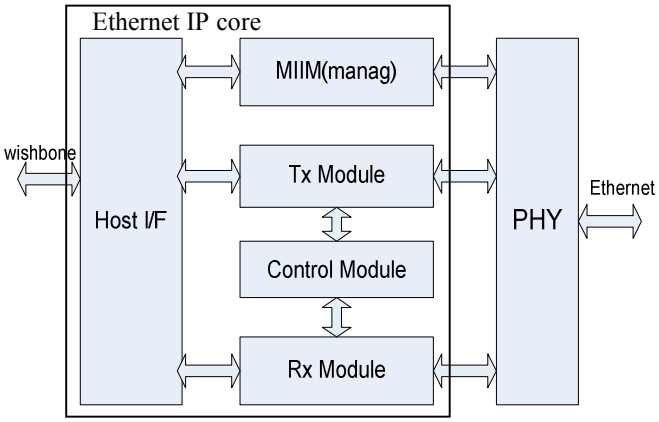

Fig. 5 MAC/Ethernet Architecture

Figure 5 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 transmission and reception 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 connected device can force PHY to run at 10 Mbps with frequency of 2.5 MHz versus 100 Mbps (25 MHz) or to configure it to run at full or half duplex mode. The wishbone interface connects the Ethernet core to the RISC peocessor and to external memory. To adapt this IP to our application we have configured it then tested the transmit and

Number of Slices	5705 out of 14336 27%
Number of bonded IOBs	155 out of 484 35%
Number of BRAMs	11 out of 96 12%
Number of DSP48Es	5 out of 16 6%

receive process which are the two basic process in our application [3].

III.2 Synthesis and Simultion results

A. Hardware simulation results

The simulation is done with Mentor Graphics simulator ModelSim. Figure 7 shows the MAC/Ethernet simulation.

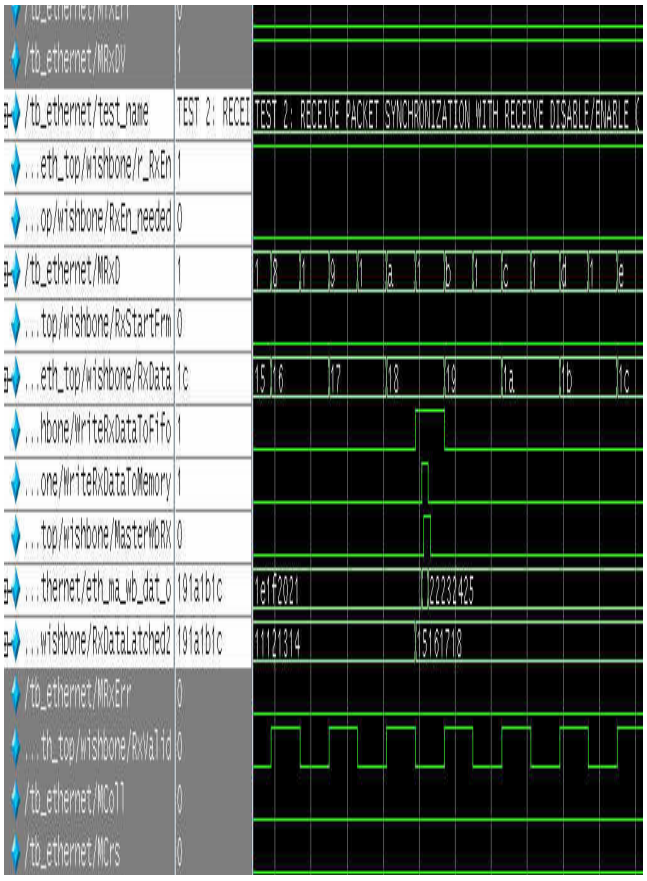

Fig. 7 MAC/Ethernet simulation results

B. Synthesis results

We have chosen the virtex5 *XC5VLX50-1FF676* FPGA FPGA as target to implement this system using the ISE 9.2i Xilinx [11] tool. As shown in table1 the network SoC architecture occupies 27 % of logic resources and 35 % of IOBs. We synthesized the architecture using the XST (Xilinx Synthesis Tool) [11].

TABLE 1
SYNTHESIS RESULTS

The figure 6 shows the mapping of the architecture into a virtex5 FPGA using FPGA Editor tool [11].

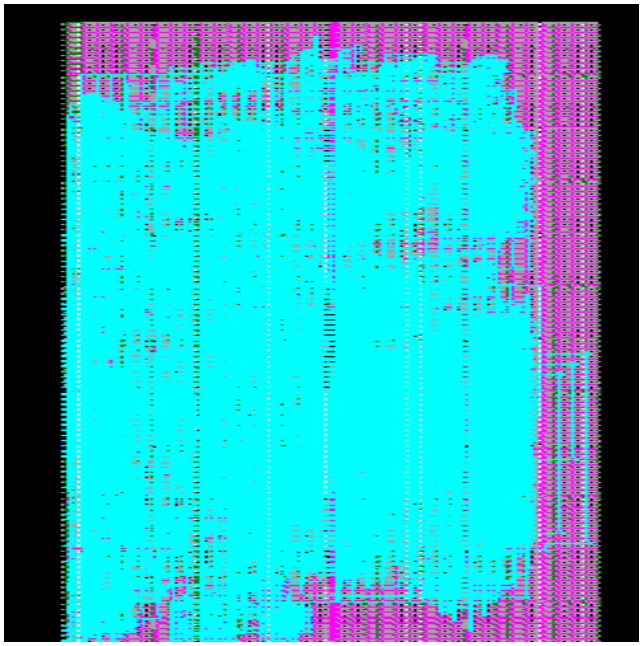

Fig.6 Mapping of the network SoC in Virtex5 FPGA

IV. Software Design

In the software part, the GNU toolchains [2] are used to compile link programs and generate the binary file of the application.

The first step consists on developing a C program "**application.c**". This file implements the main function needed for VoIP application. In the next step, the GCC tool is used to compile the program, in this phase the object file "**application.o**" is generated, in order to be used by the linker (*ld linker*). This file is linked with the linker file (*ram.ld*) which is used to map all the instructions, variables, data and stacks to the corresponding address in the memory.

The resulting binary file "**application.or32**" is used with the configuration file "*sim.cfg"* for the debug on Or1ksim step. In simulation step the "*board.h"* file which contains the hardware platform configurations is also used.

The "*sim.cfg"* file contains the default configurations of peripherals and a set of simulations environments which are similar to the actual hardware situation. In this phase the

Or1ksim simulator and the GDB tool are invoked. Finally the binary file "**application.or32**" is converted to the memory initialisation file and downloaded into the SDRAM memory.

We envisage to run the network application on µClinux (microcontroller Linux version) operating system. 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.

In this part the µClinux kernel has adjusted to the target system and compiled.

2. Programming the network application.

A. µClinux Configuration

Mainly we have configurated µClinux to use the opencores MAC/Etehrnet core as network controller and include de TCP/IP stack that provides the IP conection.

B. The network application

First we have chosen as application test a frame transfer between FPGA board and PC under Linux. Figure 8 show the test environment. The serial port is used to visualise the farme transfer.

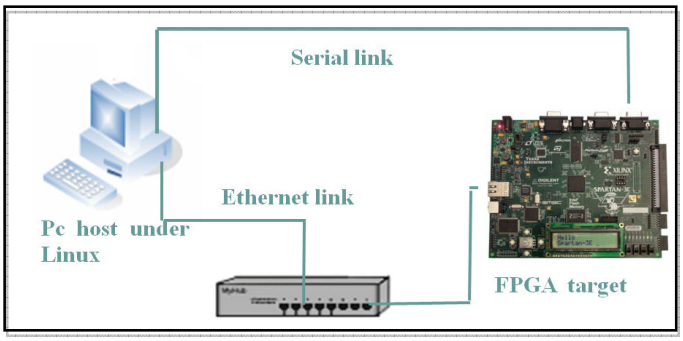

Fig. 9 Test environment

First we have adjusted the "*board.h*" file according to our system platform. The "*board.h*" file contains the hardware platform configurations and the network configuration according to CDTA/LAN configuration. The result of this test is shown in next section. We aim to test the file transfer protocol in our SoC using the 10/100 MAC/Ethernet network controller and µClinux operating system.

C. Software simulation results

At first we have run µClinux on Or1ksim simulateur [12] we have configured Or1ksim to be used with µClinux via the configuration file "sim.cfg". This file contains the default configurations of peripherals and a set of simulation environments which are similar to the actual hardware situation. Figure 9 shows the results of running µClinux on Or1ksim 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. Figure 10 shows the FPGA board-PC frame transfert test result.

```
Memory available: 7896k/8180k RAM, 0k/0k ROM (292k kernel data, 597k code)
Swansea University Computer Society NET3.035 for Linux 2.0
NET3: Unix domain sockets 0.13 for Linux NET3.035.
Swansea University Computer Society TCP/IP for NET3.034
IP Protocols: ICMP, UDP, TCP
uClinux version 2.0.38.1pre3 (root@dmn189b368) (gcc version 3.2.3) #3 Tue Jun 15
 16:22:18 CEST 2010
Serial driver version 4.13p1 with no serial options enabled
ttyS00 at 0x90000000 (irq = 2) is a 16550A
Ramdisk driver initialized : 16 ramdisks of 2048K size
Blkmem copyright 1998,1999 D. Jeff Dionne
Blkmem copyright 1998 Kenneth Albanowski
Blkmem 0 disk images:
eth0: Open Ethernet Core Version 1.0
RAMDISK: Romfs filesystem found at block 0
RAMDISK: Loading 215 blocks into ram disk... done.
VFS: Mounted root (romfs filesystem).
Executing shell ...
Shell invoked to run file: /etc/rc
Command: ifconfig eth0 inet 10.1.1.133 netmask 255.0.0.0 hw ether 00:01:02:03:04
:05
```

Fig. 9 Trace of booting µClinux on Or1ksim

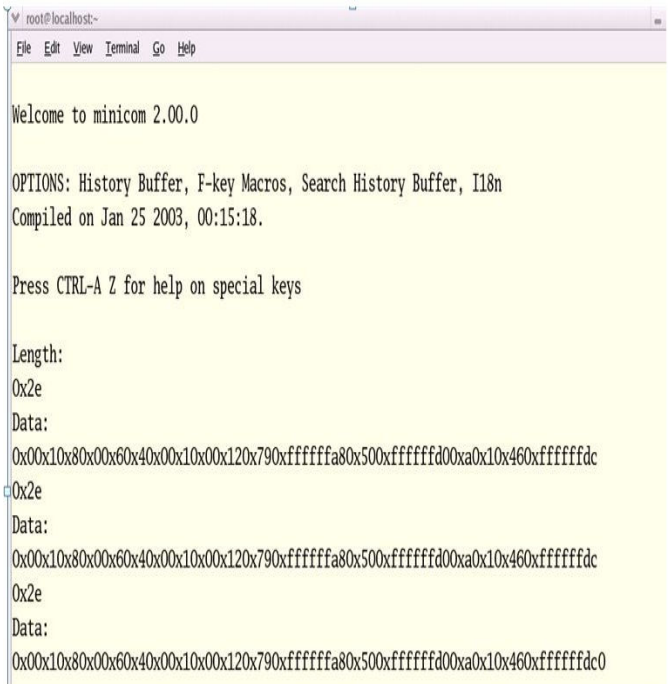

Fig. 10 FPGA board-PC Frame tansfer test result

V. Conclusion

In this work we have developed an embedded system on chip based on the Openrisc soft processor. The system based opencores/opensources approch constitues a solution for various network devices. It incorporates software and hardware parts. To bluilt the Hrdware part, we have created a SoC Verilog description. The network SoC architecture is mapped into an FPGA *XC5VLX50-1FF676* FPGA development board. Synthesis results show that the SoC architecture occupies 27% of logic resources and 35% of IOBs (In/Out Blocs). Concerning the software part the network application based on the MAC/Ethernet and µClinux operating system is the first test application. The latter 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 and record and archive calls on a computer system. Then test the VoIP gateway performances in Internet Network Area.

References

[1] S.Titri, N.Izeboudjen, L.Salhi, 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.

[2] Abid Faroudja, Nouma. Izeboudjen, Sabrina. Titri, Leila.Salhi, 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.

[3] 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).

[4] F.Abid, N.Izeboudjen, L.Sahli, S.Titri, D.Lazib, F.Louiz "Embedded Network SoC Application Based on the OpenRISC Soft Processor ", ICMOSS, Embedded System Conference, Mai 29-31, 2010. Tiaret (Algeria), pp 161-166

[5] 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.

[6] K.Morita, K. Abe, "Implementation of UDP/IP Protocol on FPGA and its performance evaluation", IPSJ General Conference. Special5, Pages157-158.

[7] Xilinx, ML501 Evaluation Platform User Guide, version 1.4, August 24, 2009.

[8] Mjan Lampret "OpenRISC 1200 IP Core Specification", Rev. 0.7 Sep 6, 2001, http://www.opencores.org/projects/or1k/

[9] www.opencores.org

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

[11] XILINX ISE 9.2 user manual. www.xilinx.com B. Smith, "An approach to graphs of linear forms (Unpublished work style)," unpublished.

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