# Low Cost System on Chip Design for Audio Processing

[1]Ayas Kanta Swain, [2]Kamala Kanta Mahapatra

***Abstract*— System-on-Chip (SoC) design is an integration of multi million transistors in a single chip for alleviating time to market and reduce the cost of the design. It uses the concept of design reuse to increase the productivity with reduction in time. In this paper we present a platform for a low cost SoC design using Open Core SoC design methodology. It offers flexible way of using reusable cores with low cost. In this proposed design a set of cores from Open Core is collected and integrated using Open Core WISHBONE interfacing for an audio processing application. The first primary result shows that the SoC can be implemented using field programmable gate array (FPGA).**

***Index Terms*— Open Core SoC Design, WISHBONE bus interface, Point-to-Point interconnection.**

## I. Introduction

The rapid development in the field of mobile communication, digital signal processing motivated the design engineer to integrate complex systems of multimillion transistors into a single chip. The integration of the transistor in a single chip greatly increases the performance of the system while reduction in system size. There is a considerable increase in the application front in recent time. Moore's law states that integration density gets doubled every two years so the complexity of the integrated systems also increases by keeping the used chip area constant. In order to keep pace with the levels of integration available, design engineers have developed new methodologies and techniques to manage the increased complexity in these large chips [1]. System-on-Chip (SoC) design is proposed as an extended methodology to this problem where IP cores of embedded processors, memory blocks, interface blocks, and analog blocks are combined on a single chip targeting a specific application.

The increased density of transistor increases the complexity of the system. The design requires a standard interface and reduction in power requirement of the circuit. The design again needs an expertise in both hardware and software levels for proper hardware and software co-design. Another important aspect of SoC integration is the development of a proper test methodology for post manufacturing test. All these integration issues makes the design time consuming and also expensive.

To deal with this inherent integration problems and reduction in design cycle time, platform based SoC design was proposed where new designs could be quickly created from the original platform over many design derivatives.

More specifically a platform is an abstraction level that covers a number of refinements to a lower level resulting in improvement of the design productivity [1].

In other side, a new concept that is gaining interest is the Open Core Soc design methodology which is based on publishing all necessary information about the hardware [2]. Open Core [3] group has provided many pre-synthesized and pre-verified hardware core for the designer under GPL/LGPL license. These cores are well documented with design specifications, RTL codes, and simulation test benches. Most of the designs are WISHBONE bus compatible for interfacing with other cores.

This paper is the result of the first step of the author's research activities towards SoC design area. The main objective of this paper is to explore the SoC design flow. A set of reusable IP cores supplied in "Aquarius" [4] project in the Open Cores has been used for exploring SoC design flows targeting a SoC architecture design for audio processing. The open cores bus protocol WISHBONE [5] has been used as the key communication interface between chosen IP cores. Finally, Xilinx FPGA is chosen for the implementation of the SoC architecture.

The rest of this paper is compiled as follows. A brief background of Open Core SoC design using WISHBONE bus interface is discussed in section II. Design methodology including hardware and software design flow is presented in section III. Soc Architecture for audio processing, synthesis and FPGA implementation results are presented in section IV and V respectively. Application example is demonstrated in section VI. Finally a conclusion is drawn in section VII.

## II. BACKGROUND

This section presents a brief background of Open Core SoC design and its WISHBONE bus interfacing. In order to bridge the technological, educational, and cultural gaps between developing countries new concept Open-Source hardware was proposed. It allows interaction of high-tech talents and qualifications in developing countries that are hidden due to market constraints. The application of Open -Source to hardware brings new benefits to the hardware development process. Open-Source hardware would reduce development time and design cost.

Open source IP cores could be reused, in black box or white box modes. The white box model allows designers to customize a particular core to their own requirements [5].

Manuscript received November 1, 2009. This work was supported by the Ministry of Communication and Information Technology, Government of India.

Ayas Kanta Swain is a Research Scholar at ECE Department, at National Institute of Technology, Rourkela, INDIA. (Phone: +91-9437341298; e-mail: swain.ayas@ gmail.com).

Kamala Kanta Mahapatra is Professor at ECE Department, at National Institute of Technology, Rourkela, INDIA. (Phone: 0661-2462454; e-mail: kmaha2@ gmail.com).

Open source hardware can be well implemented in CPLDS, FPGAs, and FPAAs. There are many organizations working to produce open hardware such as Open Core and Open Collector [6].

Open Core has published synthesizable IP cores for design reuse under GPL like licensing. The WISHBONE system on chip interconnection is proposed as the communication interface between these IP cores to foster design Reuse by alleviating system-on-chip integration problems. This improves the portability and reliability of the system, and results in faster time-to-market for the end user [7].

The WISHBONE architecture used to connect circuit functions together in a way that is simple, flexible and portable. The circuit functions are generally provided as IP Cores (Intellectual Property Cores), which are the functional building blocks in the system. Generally, the IP cores are developed independently from each other and are tied together and tested by a third party system integrator. WISHBONE aides the system integrator by standardizing the IP Core interfaces. This makes it much easier to connect the cores, and therefore much easier to create a custom System-on-Chip [7].

WISHBONE uses MASTER/SLAVE architecture for communicating between functional modules. Master can initiate data transaction to SLAVE interface through an interconnection. Figure.1 shows the types of interconnections supported are point-to-point, dataflow, shared bus and crossbar switch interconnection. The point-to-point interconnection is the simplest one that allows a single MASTER interface to connect to a SLAVE interface. The dataflow interconnection is needed for sequential data processing. In the shared bus interconnection two or more MASTERs can be connected with one or more SLAVEs and MASTER initiates a bus cycle to a target SLAVE. The target SLAVE then participates in one or more bus cycles with the MASTER. An arbiter is used to allow the MASTER to gain access to the shared bus. In the crossbar switch interconnection two or more WISHBONE MASTERs can access two or more SLAVEs at the same time.

More than one MASTER can use the interconnection as long as two MASTERs don't access the same SLAVE at the same time. WISHBONE supports all the popular data transfer bus protocols such as Single Read/Write, Block Read/Write and Read-Modify-Write (RMW). BIG ENDIAN and LITTLE ENDIAN data ordering are also supported by WISHBONE.

All designer using WISHBONE bus interface are allowed to upload their design in Open Core site and depending upon the design specification the designer can select the IP cores from the site and glue them to the WISHBONE bus architecture to design the final SOC. Other bus protocols available in market are AMBA [8], and Core Connect [9]. WISHBONE offers almost free royalty, hence reducing the overall cost of the system design.

## III. Design Methodology

This section presents a description of design methodology to implement SoC using WISHBONE bus interface. The complete SoC design flow shown in Figure.2 is divided in two main categories. One of them is hardware design flow and another one is software design flow. These flows are described individually in the next section.

The hardware design flow is started with a collection of a set of IP cores and integrating them for a chosen application, then arriving at the simulation step for verification of the entire SoC design.

Next synthesis and implementation is being done for ensuring the proper hardware mapping & routing of SoC in FPGA.

A software application required to run in the SoC hardware is being done by the software design flow. This can be run on parallel with the hardware platform.

Finally, the hardware model and software applications both implemented in the real hardware. A complete verification and monitor checking is being done for ensuring proper functionality of the system.

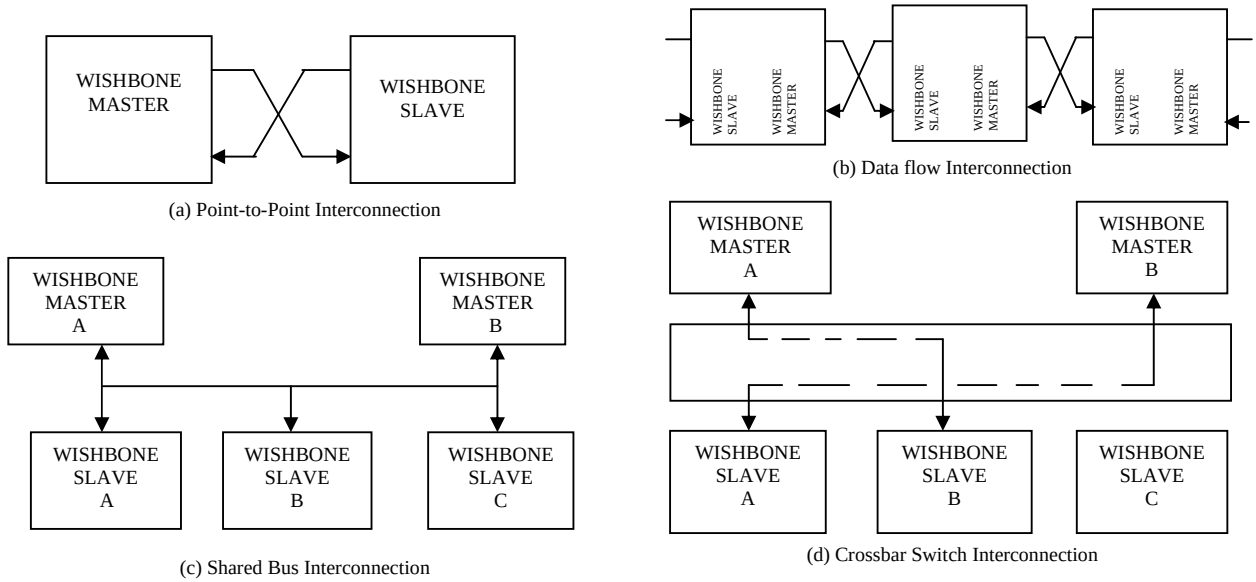


Figure.1 Types of WISHBONE Interconnection

ISBN: 978-988-18210-4-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

### A. *Hardware design flow*

The SoC research activities at author's group began with a collection of Open Core Project Aquarius aiming initially for an implementation of simple SoC design consisting of a 32 bit RISC processor (SuperH-2 [10] compatible) and a set of peripherals for data transmission with PC. The first step is to configure and adoption of the processor model and the selection of any additional peripherals needed for the design. A set of IP cores of 32 bit RISC CPU, RAM, parallel input output (PIO), memory and UART, were collected from the project and are kept back for the future applications.

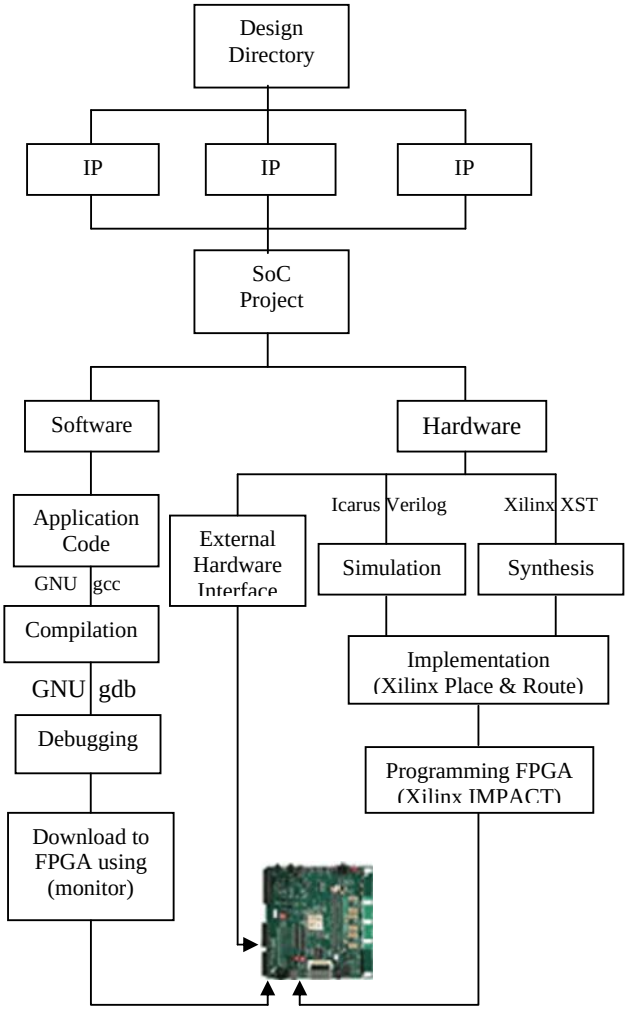


Figure.2 Design Methodology

An AC97 controller core is also selected for audio processing application. All these cores provided with the synthesis and verification results; hence a proper grasp on this code has done initially to ensure a complete control on the design.

After the selection of IP cores different steps of the hardware design can be achieved in parallel, such as external hardware interfacing, simulation and synthesis of the design. A set of test bench have been developed in order to simulate the hardware model. Aquarius developer has given a test bench with the project. Icarus Verilog [11] tool is used to run this test bench to check the results obtained are accurate to our application.

A gate level net list is generated from a set of given RTL code for modeling the SoC architecture into hardware. This can be achieved by the synthesis step. Xilinx-Synthesis-Tool [12] is used to compile the RTL behavior of SoC to generate a gate level net list for the FPGA.

Finally, implementation step is done where the gate level net list generated in the previous step is used by Xilinx place and route tools to do mapping, placing & routing for target FPGA. A bit-stream file is generated to program the FPGA with the hardware model obtained in the complete flow. The additional hardware needed for the application is developed and interface with the FPGA.

### B. *Software design flow*

Software design flow usually runs in parallel to the hardware design flow. This section explains the design flow for software applications and the tools provided for designer to choose for the work.

The simulator of Verilog-HDL codes and the compiler/assembler of the application code development run on the UNIX environment. Cygwin [13] is selected as the preferred environment for this purpose. To simulate verification program and to develop the application program, the SuperH-2 assembler and compiler are necessary. GNU tool chain is preferred which can be used by the designer for software development. The other steps involve compilation, debugging and implementation of the software on the hardware.

The project in Open Core provided with simple and useful resources for logic verification and application code development. Application codes are developed in C and compiled with GCC compiler. Debugging the application is an important step in the software development flow, to validate the results obtained with the program in the SoC. This is done by GNU GDB debugger. Finally, the code is downloaded to SoC model using monitor program dumped in the FPGA.

## IV. SoC ARCHITECTURE FOR AUDIO PROCESSING APPLICATION

Figure.3 shows the proposed architecture for speech processing application. This architecture utilizes processor and peripherals from the Aquarius [4] project designed by Thorn Aitch published in Open Core [3] under GPL license. This section describes about the IP cores and their specifications used for building this SoC architecture. The architecture consists of a 32 bit Super-H compatible CPU MASTER, a memory unit, a universal asynchronous receiver transmitter, a system controller, an audio codec for voice coding and a parallel input output. The cores are connected to each other through WISHBONE interface with a point to point interconnection scheme.

The ***CPU* is** a RISC processor based on superH-2 Instruction set Architecture..This synthesizable core written in Verilog and can be implemented in FPGA. This is published under GPL license in opencore.org. The SuperH-2 is a 5-stage pipelined architecture with 16 32bit general registers. It can handle interrupt requests like (NMI) non-maskable interrupt and (IRQ) interrupt request. In a lower module it comprises memory access controller, a data path unit, a multiply unit and a decoder unit.

ISBN: 978-988-18210-4-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

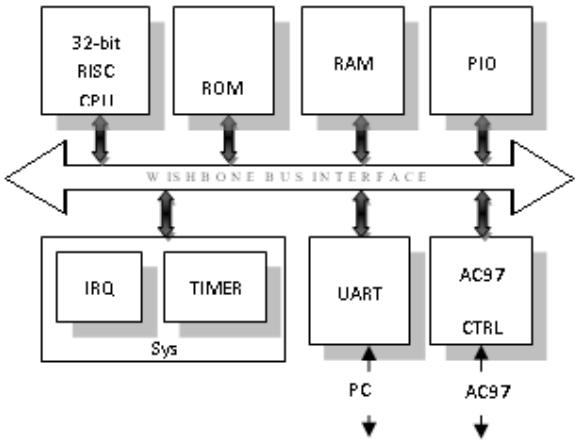


Figure.3 Soc Architecture for Audio Processing Application

The memory access controller sends fetched instruction bit field to the decoder unit, in turn decoder unit decode the instruction bit fields and throws many control signals for execution and data read/write access towards data path unit, multiplication unit and memory access controller. The data path unit has sixteen general registers, Status Registers, Global Base Registers, Vector Base Register, Procedure Register, and Program Counter. The CPU also has 32 bit architecture for enhanced data processing ability as multiply and accumulation like DSP functionality.

The **System Controller** (SYS) is used to generate and emulate exceptions of hardware event like NMI, IRQ and CPU address error and manual reset. It has 12 bit Interval Timer to generate IRQ. It also controls the priority level among the requests of hardware exception. The SYS has 2 32bit length registers which reset to 0x00000000 when power on reset.

The **Parallel I/O (PIO)** has two 32-bit registers to control Port Pins. There are 4 byte-size registers for PORT Output and 4 byte-size registers for PORT Input. In order to access PORT input the registers have to read and to access PORT Outputs the registers have to be written. Each registers can be accessed by byte, word or long operand size.

The **on chip memory** is a simple memory module used as ROM and RAM. The total size is 16 kb. Xilinx BRAM is used as a memory module during FPGA implementation.

The ***A97*** Controller supports AC97 audio codec that converts analog voice from microphone to digital data for processing. The features includes variable and fixed sample rate support up to 48 KHz , 16, 18 and 20 bit Sample Size. This is a fully WISHBONE compatible IP core available in Open Core.

## V. SYNTHESIS AND FPGA IMPLEMENTATION OF SOC

After the completion of both the hardware flow and the software flow, we have obtained a hardware model. The next step is to implement the SoC model on the target FPGA and to run the application software on the SoC to verify the result.

One of the most important tasks is to integrate all the cores provided. Hence there is a need of a top module HDL code which comprises of a structural modeling of the IP cores used for the implementation. A top module HDL code is provided where WISHBONE is the main communicating interface between the IP cores and all the IP cores are interconnected with the point to point interconnection. The 32 bit RISC processor is the master for all the components. All other component cores are configured as a slave to this master processor. At this time only CPU, UART, on chip memory, System controller and PIO are integrated. Every slave component is defined by a specific address map and an address decoder has used for decoding which slave will be accessed by the master. We synthesize the architecture using Xilinx 9.1i XST tool. The synthesis result is shown in Table. I.

TABLE. I
SYNTHESIS RESULT

**Design Information**
**Target Device: xc2vp30-7fg896**

**Device Utilization Summary**

| Logic Utilization | used | available | utilization |
|---|---|---|---|
| Number of Slices | 2,847 | 13,696 | 23% |
| Number of Slice Flip Flops | 1,378 | 27,392 | 5% |
| Number of 4 input LUTs | 5,131 | 27392 | 18% |
| Number of bonded IOBs | 37 | 556 | 6% |
| Number of GCLKs | 2 | 16 | 12% |
| Number of Block RAMs | 32 | 136 | 24% |
| Number of MULT18X18s | 2 | 136 | 1% |
| Number of DCMs | 1 | 8 | 12% |

Final implementation of SoC architecture has done in a Virtex-II Pro (xc2vp30) FPGA evaluation board. ISE place and route tool has been used for mapping and implementing the SoC in the FPGA. At the end of the hardware design flow we obtained a bit stream file which is used by XILINX [12] Impact programming tool to program the FPGA. The running of the application code of the SoC architecture is described in the next section. Table-1 shows the synthesis result which shows the SoC occupies 23% of the number of slices, available in the FPGA. The number of input / output used is 6%.

## VI. SIMULATION RESULTS

A verification step plays an important role in SoC design flow. The peripheral cores used are pre-verified cores that function accurately. Hence, a test bench was written in Verilog that verifies the CPU's operation. The instructions of the CPU are simulated considering bus transactions, signal levels and register contents, etc. This is done by using an open source simulator Icarus Verilog [11]. Icarus Verilog supports only text mode of output viewer. Hence to view the output wave form GTKWave [14] is used. Upon simulation the Icarus Verilog generates a Value Change Dump (VCD) file. GTKWave open this VCD file to show the wave form of the simulation. GTKWAVE is supported by all the platforms including Linux and Windows.

ISBN: 978-988-18210-4-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

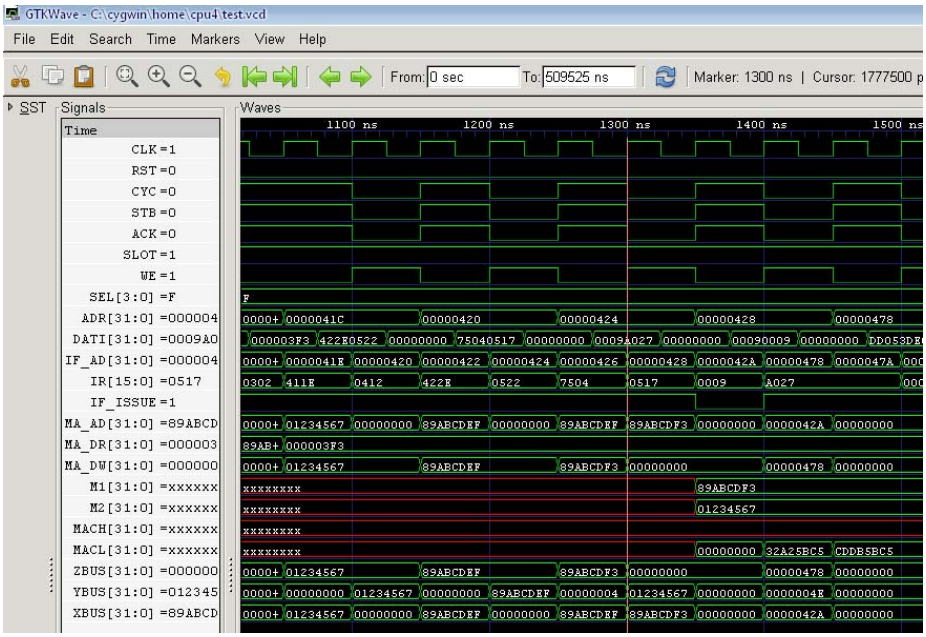


Figure.4 Simulation Results in GTK wave form viewer

Figure.4 shows the simulation result, which shows the WISHBONE signals and internal signals of CPU for assembly language instructions. The bus transaction and internal signal behavior for a multiplication instruction is presented in the simulation waveform.

It was observed from the waveforms that, the system works on positive edge of clock pulse (CLK) when reset (RST) is active low. The cycle (CYC) indicates the single and block Read/Write cycles. CYC high indicates a block Read/Write cycle is initiated. During CYC is high, CPU makes strobe (STB) high to inform Slave that a valid bus transaction is initiated. Write enable (WE) high indicates a write operation else a read operation is initiated. As a response Slave responds the CPU Master by asserting the ACK signal as shown in the Figure.4. The select signal SEL is continuously high indicating "F" in the SEL bus of the waveform. The address signal (ADR) show the 32-bit address bus of the system, and DATI show the 32-bit data bus of the system.

IF_ISSUE ="1" indicates instruction fetch started and memory access controller sends the instruction code "0517" to instruction register IR [15:0] of the decoder unit. For the MUL instruction R1 and R5 are multiplied. The R1 value from YBUS and R5 value from XBUS are transferred to multiplier latch M1 and M2. Hence, M1= "89ABCDF3" and M2="01234567". In the next clock pulse when ACK is high M1 and M2 multiplied and the output register MACL contains the multiplication value of "CDDB5BC5".

## VII. Application

In this final section we use all the steps in the proposed methodology to implement SoC architecture in the FPGA. A set of application code is provided with the project which was developed by GNU C compiler.

During FPGA implementation, ROM is configured by BRAM to reduce the consumption of logic cells. A "genram" utility is given in the Aquarius project that converts the object file to Block RAM's INIT statements. FPGA operating clock frequency is set to 20 MHz. A set of LCD and key board is interfaced with the FPGA. Finally UCF file is written with the BRAM INIT statements.

After the implementation of SoC in the FPGA, a monitor program is run on the SoC platform which has very basic functionality as memory editor, program loader from PC, jumping to program, setting break point and reading registers. To test the peripherals application code such as LCD test, and interrupt clock are compiled in C and verified in the hardware architecture.

## VIII. Conclusion

We have successfully implemented a SoC Platform by adopting the Open Cores design methodology. The first primary results show that the part of the architecture can be mapped into an FPGA . The corresponding simulation result demonstrates the accurate functionality of WISHBONE bus signal and CPU internal signals for a multiplication instruction. The future work is to integrate the AC97 controller with the processor, adopt a filter algorithm and to check the result with audio processing application.

## References

[1] Resve Saleh,Steve Wilton ,System-on-chip: Reuse and integration, Proceedings of the IEEE| vol.94 ,No.6, June 2006.
[2] S.Titri, N.Izebdjen, L.Sahli, D.Lazib, F.Louiz, Open Cores based System on Chip Platform for Telecommunication Applications: VOIP , IEEE conference 2007
[3] OpenCores project site http://www.opencores.org
[4] Aquarius project site http://www.opencores.org/project,aquarius
[5] Mohamed A. Salem,Jamil Khatib, "An introduction to open-source hardware development", EEDesign.com

ISBN: 978-988-18210-4-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[6] OpenCollector site http://collector.hscs.wmin.ac.uk
[7] WISHBONE Specification site
www.opencores.org/downloads/wbspec_b3.pdf
[8] www.arm.com
[9] http://www-03.ibm.com/technology/index.html
[10] http://sg.renesas.com/
[11] Icarus Verilog Manual, http://www.icarus.com/eda/verilog/
[12] *Xilinx user manual* www.xilinx.com
[13] www.cygwin.com
[14] GTK Wave User Manual, http://gtkwave.sourceforge.net./

ISBN: 978-988-18210-4-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)