Implementation, Verification and Validation of an OpenRISC-1200 Soft-core Processor on FPGA

Abdul Rafay Khatri
Department of Electronic Engineering,
QUEST, NawabShah, Pakistan

***Abstract*—An embedded system is a dedicated computer system in which hardware and software are combined to perform some specific tasks. Recent advancements in the Field Programmable Gate Array (FPGA) technology make it possible to implement the complete embedded system on a single FPGA chip. The fundamental component of an embedded system is a microprocessor. Soft-core processors are written in hardware description languages and functionally equivalent to an ordinary microprocessor. These soft-core processors are synthesized and implemented on the FPGA devices. In this paper, the OpenRISC 1200 processor is used, which is a 32-bit soft-core processor and written in the Verilog HDL. Xilinx ISE tools perform synthesis, design implementation and configure/program the FPGA. For verification and debugging purpose, a software toolchain from GNU is configured and installed. The software is written in `C` and `Assembly` languages. The communication between the host computer and FPGA board is carried out through the serial RS-232 port.**

Keywords*—FPGA Design; HDLs; Hw-Sw Co-design; OpenRISC 1200; Soft-core processors

I. Introduction

The field of microelectronics has revolutionary changes due to research and development in System on Chip (SoC) technology. This technology plays a vital role in the design of various embedded systems. Embedded systems are involved in medical applications, automotive, home appliances, industrial control system and many more. A general embedded system consists of a microprocessor for processing, memory for storage, output and input devices for displaying output and take inputs from the outside world respectively [1]. Fig. 1 shows the simple and general block diagram of an embedded system. A processor is the heart of an embedded system and processors are classified into two categories hard-core and soft-core processors. The complexity of integrated component inside the embedded system increased drastically, and it is not possible to design a microprocessor for every specific application. Therefore, it requires to develop the embedded application using a soft-core processor which reduces the time to market and cost for the design.

For that purpose, it is a good idea to use soft-core processor having reconfigurable, predefined and pretested Intellectual Property (IP) cores. It is an alternative solution. The use of IP cores or soft-cores designing using Hardware Description Languages (HDL) reduce the cost and time to market for the design of embedded systems. These cores can be realised to any FPGA devices from any vendor. The OpenRISC 1200 (OR1200) processor is also soft-core processor written in Verilog HDL. It is a 32-bit Reduced Instruction Set Computer

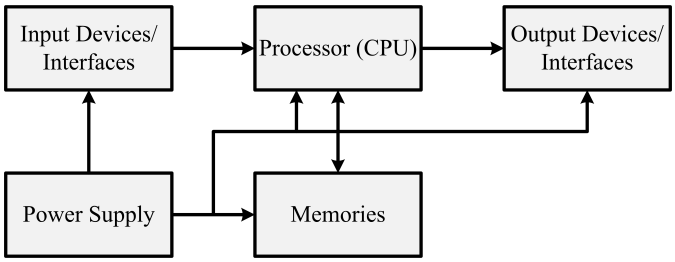

Fig. 1. General block diagram of embedded systems.

(RISC) processor. This processor consists of all necessary components which are available in any other microprocessor. These components are connected through a bus called Wishbone bus. In this work, the OR1200 processor is used to implement the system on a chip technology on a Virtex-5 FPGA board from Xilinx. The communication between host computer and the OR1200 processor on the FPGA device is carried out through a Universal Asynchronous Receiver Transmitter (UART) serial communication (RS-232).

The OR1200 processor core is available at open source community `opencores.org` [2]. This soft-core processor is used to develop a system on a chip. The soft-core processor is technology independent which means, it is implemented on any FPGA device or board. To develop and implement the embedded system with the OR1200 processor on Virtex-5, we used Xilinx ISE 12.4 to make a project. The synthesis, design implementation and bit file are generated through the same software. Also, in this work, the software platform is developed using the GNU toolchain so that the C and assembly programs can be compiled, linked and executed on this processor and UART communication is achieved for display output of the programs.

The organisation of the paper is as follows: Section II describes the detail about the available commercial and open source core processors. The architecture of OpenRISC 1200 processor is described, along with the detail description of various components in Section III. Section IV describes the development of a hardware platform and software platform. Section V explains the serial communication perform between OpenRISC processor and UART core. In the end, Section VI concludes the paper.

II. Soft core processors

Soft-core processors are microprocessors that can be adequately described by programming usually in HDL, mainly Verilog or VHDL. This code can be synthesized using different

tools depending on the manufacturer and can be implemented on FPGAs. Soft-core processors are provided by many companies and can be categorised into two ways:

- Commercial cores
- Open source cores

A. Commercial Cores

The three influential providers for commercial soft-core processors are Altera, Xilinx and Tensilica. They provide Nios II, Micro-Blaze, Pico-Blaze and Xtensa cores respectively [1], [3]. The sequel describes the feature of each soft-core processor available in the market.

1) Nios II: The Nios II embedded processor belongs to the family of soft-core processors, which is designed and developed by Altera Corporation [1]. The Nios II processor system is equivalent to the micro-controller system or "computer on a chip" which includes I/O devices, memory (on-chip and off-chip) and processor with their interfaces can be implemented on the single Altera chip [4]. This processor is based on Load/Store RISC architecture and has flexibility for the users to choose between 16/32 bit data path for the customisation of design parameters [5]. It means many parameters like registers, cache, custom instruction and data bus size can be chosen at the time of design for speeding up the customise hardware. This processor has 5-integer pipeline with RISC architecture of 32-bit and has 512 general purpose registers [6], along with it has the capability for handling the instruction and data caches, hardware multiplication and division, interrupts handling and floating point precision operations. Software tools provided by Altera Corporation can do this all. Software toolset includes GNU C/C++ compiler along with Eclipse IDE and is called Nios II software Integrated Development Environment (IDE) [7].

2) MicroBlaze: MicroBlaze is also from the one of the reconfigurable processors family designed and provided by Xilinx. Just like Nios II, it can also be customised with I/O devices and memory configurations [8], [9]. This soft-core processor has a Harvard 32-bit RISC architecture with 32 (32-bit) wide general purpose registers, three stages pipeline with variable length flexibilities, 32-bit full address bus and two interrupt handlers optimised for Xilinx FPGA boards [1], [7], [8]. It has two addressing modes. There are also some advanced features such as barrel shifter, divider, multiplier, instruction and data caches, exception handling, debug logic, single precision Floating-Point Unit (FPU), interfaces and many others [8], [10]. It also has on-chip and off-chip memory for MicroBlaze to provide single cycle access to memory and they formed a bus known as on-chip peripheral bus and used to interface different peripheral and memory devices with MicroBlaze [1]. The size of memory and the number of I/O devices are attached to the system by the user, and it depends on the application under development. As this is soft-core processor so any feature which is not required, do not need to implement. To build a complete soft-core processor system with MicroBlaze processor, we require some interfaces like UART, Ethernet, Serial Peripheral Interfaces (SPI) and some other cores but they are implemented on the single chip of FPGA [5].

3) PicoBlaze: PicoBlaze is also a soft-core processor also provided by Xilinx. PicoBlaze is a compact, capable, cost-effective and efficient 8-bit micro-controller like Intel 8051 targeting simple data processing applications [11]. This micro-controller is optimised for Spartan and Virtex families [1]. It has the capability of interrupt handling but it does not perform division, multiplication and floating point operations [9]. PicoBlaze micro-controller is available in the form of synthesizing and configurable VHDL code and can be download from Xilinx website. The tools for programming the PicoBlaze processor are assembler and C compiler with integrated development environment and simulator for VHDL [9]. It also supports Xilinx system generator development environment. It has 16-bit wide general purpose data registers, 8-bit ALU with two flags carry and zero, 64 byte internal RAM and 256 inputs and 256 outputs ports for expansion and interfacing [11].

4) Xtensa: Xtensa is a soft-core, configurable microprocessor design and provided by Tensilica's Inc. [12]. It is designed by keeping in mind the ease of integration, customisation and extension. This processor is famous for its two main features [12]:

- Configurable: - It offers features to the designer a set of predefined parameters which are used to configure the processor for some applications.
- Extendable: - It also offers extendibility to the designer to invent some custom instruction and integrate logic very smoothly for the specific applications.

This Xtensa soft-core processor is written in Tensilica Instruction Extension (TIE) language which is similar to Verilog HDL language [1]. The TIE compiler compiles the code written in TIE. There is an advanced compiler available for this purpose such as XPRES which can also generate and compile code for TIE and HDL. There are two versions of this processor are available from Tensilica, the Xtensa LX, FLIX and the Xtensa-9 [1], [13].

B. Open Source Cores

Open source community provides the open source IP cores components for the development of an embedded system for both academic and research. Open source offers LEON and OpenRISC1000 soft-core processors. Sun micro-systems also produce soft-core processor OpenSPARC, which is widely used in Application Specific Integrated Circuits (ASICs) implementations. The sequel describes the summary of open source soft-core processors [1].

1) Leon SPARC: Leon SPARC (Scalable Processor Architecture) is the IP core processor based on the SPARC V8 architecture. The providers of this core are the European Space Agency and Gaisler Research [6]. Leon SPARC is most widely available in two versions namely LEON 2, LEON 3 and LEON 4 [1], [29]. LEON 2 and LEON 3 are 32-bit open source VHDL model having 5-stage integer and 7-stage pipeline respectively [1]. They also have divide, multiply, MAC units, 32-bit PCI bridge with optional DMA and FIFO, UARTs, timers & watchdog, GPIO port, interrupt controller, status registers, general purpose registers (2 to 32), CAN 2.0 interfaces, advanced on-chip debug support unit, JTAG/TAP

controllers and floating point unit. All these parts are interconnected through a bus called AMBA-AHB (Advanced Micro-controller Bus Architecture-Advanced High-speed Bus standard provided by the ARM and it is included in GRLIB [14]. This bus provides support for many master interfaces and achieving high bandwidth operations. GRLIB is an IP library based on the collection of VHDL libraries and is designed to enable the vendor to include their libraries for specific applications. It provides IP cores for functional and logistical interfaces for the development of SoC (System on Chip) [14].

2) OpenSPARC: OpenSPARC is also called UltraSPARC launched by Sun Micro-systems in December 2005. The Sun Micro-systems surprised the industry by distributing it as an open source processor in 2006. After one year in 2007, they launched another UltraSPARC, which is more advanced than the first processor and named as OpenSPARC T1 and OpenSPARC T2 [15]. The OpenSPARC is designed for academics as well as commercial use. In academics, OpenSPARC can be taught to the students in different course regarding computer architecture, VLSI design, compilation and generation of code. The commercial use of this processor is to provide a springboard for the design of new custom processors with the complete and fully verified suite, which reduces the time drastically to market factor. OpenSPARC T1 and OpenSPARC T2 architectures are based on UltraSPARC architecture in 2005 and 2007, respectively [15]. The general features of this soft-core processors include a linear 64-bit address space, few addressing modes, 32-bit full instructions, floating point unit, fast trap handlers, multiprocessor synchronisation instructions, hardware trap stack. These features are also compatible with SPARC V9. Some features which are only available in UltraSPARC are dominant mode; Chip Level Multi-threading (CMT), extended instruction set, multiple levels of global registers and many more... Tools for OpenSPARC T1 and OpenSPARC T2 are mostly the same. EDA simulation tools include VCS and NCVerilog from Synopsys and Cadence respectively. EDA synthesis tools required to perform Verilog Register Transfer Level (RTL) are designed compiler from Synopsis, Synplicity Pro from Synplicity and Xilinx Synthesis Technology (XST) from Xilinx. FPGA tools are required to download bit-stream and emulate it. Those tools are Embedded Development Kit, Integrated Synthesis Environment (ISE) from Xilinx and Modelsim from Mentor graphics [15], [16].

3) OR1200 OpenRISC: The most widely used soft-core processor from open source community `opencores.org` is OR1200 processor. This processor optimises to zero cost, smaller power consumption, higher performances, and versatility in various modern applications such as networking, home appliances, and embedded automotive consumer products. This processor belongs to the OR1000 family of microprocessors, and it has 32/64-bit scalar RISC Harvard architecture [17]. The features include 5-integer pipeline, separate memory for data and instruction, virtual memory caches and DSP capabilities. This processor can be synthesized and downloaded on both Xilinx and Altera FPGA boards. The architecture, features and performance are described. The OR1200 soft-core processor is compatible with a real-time OS such as Linux, Windows (Cygwin). The software can be written and compiled in C/C++. This processor is wishbone bus compatible [1], [18], [19], [20].

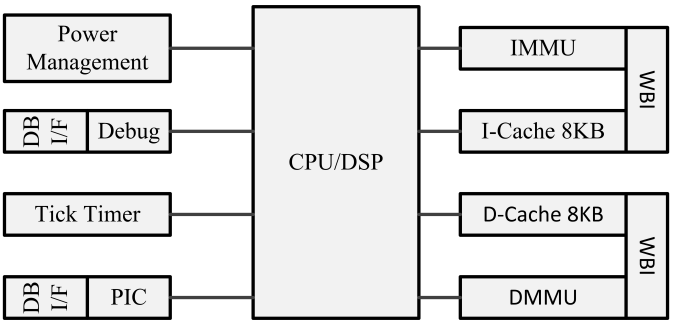

Fig. 2. Block diagram of OpenRISC 1200 processor architecture.

C. Advantages and Disadvantages

There are certain advantages and disadvantages of both commercial and open source cores. The sequel describes few merits and demerits of soft-core processors.

1) Advantages: There are many advantages of using soft-core processors in the embedded design on FPGAs. Some of them described below [1], [4], [21].

1) Flexible and easily customizable for a specific application.
2) Technology independent hence can be synthesized and implemented on an ASIC and FPGA technology.
3) Soft-core processor's architecture and behaviour are described by HDL at higher level of abstractions hence are easy to understand the overall design.
4) Peripherals in the processor can be changed, add and remove as per requirement with ease.
5) Reduced obsolescence risk.

2) Disadvantages: There are also some disadvantages of using soft-core processors. Significant trade-offs are described below [1], [7], [18].

1) Size and area is large
2) Power consumption is large
3) Performance is lower than ASICs

III. OpenRISC (OR1200) Architecture

OpenRISC 1200 processor is an implementation of the OR1000 family of open and free soft-core processors [19], [22]. The OpenRISC 1000 processor is a development of the open cores modern architecture and is a base for the family of 32/64 bit RISC and DSP processors [20]. The OR1200 processor is the 32-bit scalar RISC with Harvard micro-architecture. OpenRISC 1200 processor consists of 5 stages integer pipeline, virtual memory support, two default caches for data and instruction physically tagged together, MMUs are implemented, high-resolution tick timer, power management unit, a programmable interrupt controller (PIC) and debug unit for interfacing and real-time debugging facilities as shown in Fig. 2. OR1200 can run on any operating system and is used into the development of System on a Chip, embedded application and networking application. Each block describes their features below in detail [17], [22], [23].

A. CPU/FPU/DSP

The primary and central processing part of OR1200 processor is CPU/FPU/DSP. CPU/DSP uses the architecture of OR1000 processor family and implements 32-bit operations while 64-bit is not realised for OR1200. Also, vector and floating point operations are not developed. Fig. 3 shows the block diagram of OR1200 CPU/DSP.

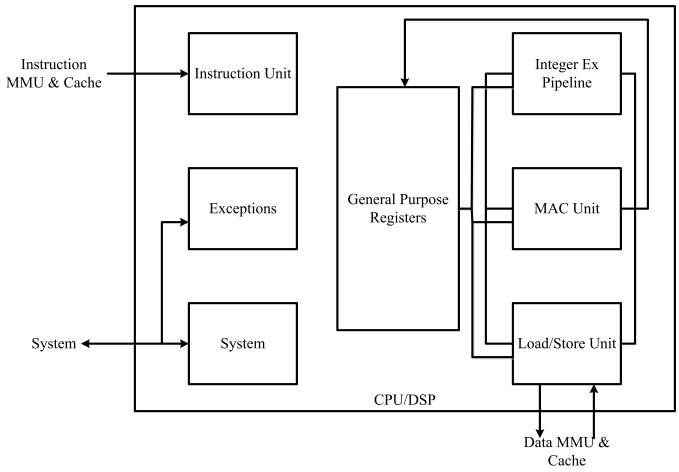

Fig. 3. Block diagram of OpenRISC 1200 CPU architecture.

1) Instruction Unit: Instruction unit inside the CPU implements the basic pipeline instructions, fetching instructions from memory and executing them in the proper order. It also performs some conditional jump and branch instructions. The instruction unit of OR1200 processor handles only ORBIS32 class while this architecture does not support other classes ORFPX32/64 and ORVDX64.

2) General Purpose Registers: There are 32 general purpose registers (GPRs). Each GPR is 32-bit wide and is implemented in OR1200 architecture. Two synchronous dual port memories are implemented in OR1200 from GPR with the capacity of 32 words by 32 bits per word [19]. In ORBIS instructions these registers can be accessed as source and destination registers. They are used to hold scalar data, pointers and vectors [20], [22].

3) Load/Store Unit: The Load/Store Unit is abbreviated as LSU and is used to load data from memory or to store data to the memory. It is an independent execution unit. It may also be used in vector processing. All load/store instructions are implemented in hardware. Those instructions define the addressing modes of operands. The operand may be located in address register operands, source data register operand for store instructions and destination data register operands for the load instruction.

4) Integer Execution Pipeline: The following instructions are a 32-bit integer and implemented in this core. Most of the instructions take one cycle of time during execution.

- Arithmetic instructions
- Logical instructions
- Compare instructions
- Shift and rotate instructions

5) MAC Unit: This unit is responsible for DSP MAC operations which are 32x32 with the 48-bit accumulator. It can accept new MAC operation in each new clock cycle and is fully pipelined.

6) System Unit: This unit provides the interfaces to those signals to the CPU/DSP which cannot be connected through instruction and data interfaces. This unit implements the system's special purpose registers, e.g. Supervisor Registers.

7) Exceptions: The core exception can be generated when exception handling occurs. In the OR1200 processor, there are some causes for exceptions to happen and given below.

1) Illegal op-codes
2) External interrupt request
3) System call
4) Breakpoints exceptions (internal exceptions)
5) Memory access conditions

Exceptions take place in the supervisor mode. When it occurs, control transfers to exception handler at an offset depends on the type of encountered exceptions.

B. Data Cache & MMU

The OR1200 is based on Harvard architecture; it means data and instruction caches are separate entities. The default cache configuration for data is 8 K byte which is 1-way direct-mapped data cache for rapid access of data for the core. This configuration can be changed in many ways, e.g. 1 K byte, 2 K byte, 4 K byte and 8 K byte per set.

Data MMU is separated from Instruction MMU. The OR1200 implements a virtual memory management system. The primary function of MMU provides the memory access and translation from useful addresses to physical addresses.

C. Instruction Cache & MMU

The instruction cache is a separate entity. The default cache configuration for instruction is also 8 K byte which is 1-way direct mapped instruction cache for rapid access of instruction for the core. This configuration can be changed in some ways, e.g. 1 K byte, 2 K byte, 4 K byte and 8 K byte per set. The Least Recently Used (LRU) replacement algorithm is implemented in each set of this cache.

Instruction MMU is also a separate entity. The OR1200 also implements a virtual memory management system. It provides the memory access and translation from effective addresses to physical addresses. The page size is also 8 K bytes and has a comprehensive page protection scheme. The following configuration of 1-way direct mapped hash based Translation Look-aside Buffer (ITLB) can be implemented as 16, 32, 64 (default) and 128 entries per way or ITLB entries. Hash-based design provides the higher performance.

D. Power Management Unit

The primary function of this unit is to optimise the power consumption by deactivating or activating specific internal modules which are not in use. OR1200 implements this feature. There are three modes namely slow/idle mode, sleep mode and doze mode. The low power dividers are available in external

clock generation circuitry. The slow/idle mode takes advantage of those dividers. It enables the functionality but on less frequency and hence the power is reduced. Both the sleep and doze mode are left to normal mode by the occurrence of a pending interrupt. In the sleep mode, all the internal units of OR1200 are disabled and clock gated while in doze mode software operation is suspended. The clock signal to all RISC modules/units is disabled except tick timer. The other modules on the chip can continue their functions as in the normal mode.

E. Tick Timer

The primary function of the tick timer is to measure time and schedule system tasks. It is used by operating system and driven by RISC clock. The tick timer facility is implemented in OR1200. Tick timer has a maximum timer count of 2^{32} clock cycles and a maximum period of 2^{28} clock cycles during interrupts. The interrupt for tick timer can be masked. It is a single run, restartable or continuous timer.

F. Debug Unit

The purpose of a debug unit in the OR1200 is very significant because it provides a means to interact with the host computer for debugging our programs and check the status of various registers during the process. It can also help to load program to the internal memories with the help of JTAG cable. Basically, in OR1000 architecture more features are available such as watch-points, breakpoints and flow controlling but the debug unit implemented in OR1200 supports for basic debugging.

G. Programmable Interrupt Controller

The task of an interrupt controller is to receive interrupts from external sources and peripherals and send them to the CPU so that it can activate the corresponding interrupt handler according to their LOW and HIGH priorities. The PIC implemented in OR1200 has three special purpose registers (SPRs), and 32 interrupts lines. The interrupt line '0' and '1' are always enabled by connecting with a HIGH and LOW priority interrupt inputs respectively. Remaining interrupts can be programmed and masked as well.

H. Wishbone Interfaces

Wishbone bus provides an interface to connect OR1200 to different modules, memory subsystem, and external peripherals. The width of this bus is 32-bit wide, and it does not support other sizes. The OR1200 is compatible with wishbone SoC interconnection Rev. B specifications.

I. UART Core

This soft-core is a free available from open source community `opencores.org` [2]. UART stands for Universal Asynchronous Receiver/Transmitter. This core provides communication capabilities with the modem or external devices like PC using RS-232 protocol or serial cable. This core is maximum compatible with the national semiconductors' 16550A industry standard devices [24]. The core consists of transmitter unit TX, receiver unit RX, interrupt block, modem logic block, wishbone interface bus block and various registers [25]. This core is attached with OR1200 processor with the wishbone SoC interface bus. It is compatible with an 8-bit data bus [24]. When this core is connected to the OR1200 based system, the transmitter unit converts parallel data into a serial form to the host while receiver unit process that serial data [25]. Following are some general features of this core,

- Wishbone bus width is selectable 8-bit or 32-bit modes for this module.
- Perform only FIFO (First In, First Out) operations.
- 32-bit debug interface.

IV. Development of Hardware and Software Platforms

Embedded systems require high reliability, high performance, low power consumption and low cost. Embedded systems are developed on FPGAs or ASICs platforms by soft IP cores. When different IP cores are integrated into a single FPGA chip, it is called System on a Chip (SoC) design. Soft-core processors and IP core components described earlier are freely available under the license of Lesser GNU Public License (LGPL) for open source community.

There are two major HDL languages, Verilog HDL and VHDL. The soft IP cores are written in different languages. The core used in this paper is written in Verilog HDL. The `opencores.org` provides a project named MinSoC (Minimal OpenRISC System on Chip) contains soft IP cores for OpenRISC 1200 processor, UART, Ethernet MAC (Media Access Control), debug unit, start-up module, JTAG tap module and SPI. This generic core is provided with a synthesizable core which can be downloaded to every FPGA and also compatible with every FPGA without the changing of its code. Only minor changes have to do.

A. Hardware Platform

The board used is equipped with Virtex-5 (XC5VLX110T) FPGA device. The configuration bit-stream is downloaded to this board. The hardware consists of an FPGA board, Xilinx platform cable with JTAG cable and RS-232 serial null modem cable. To download the configuration, several steps have to perform to make the configuration file, i.e. `*.bit` file. The Virtex-5 board from Xilinx is shown in Fig. 4 used to implement the work. The device utilisation summary for the implementation of Open RISC 1200 processor is shown in Fig. 5.

1) Programming FPGA with JTAG: The Xilinx software ISE 12.4 is used to synthesize, design implementation and generation of bit file to configure FPGA. The iMPACT tool is also available with ISE package. The iMPACT tool is used to download the bit file into FPGA chip. This programming required a PC with iMPACT tool, Xilinx platform cable and JTAG cable is needed to make a physical connection with the board. The following steps take place for the configuration of FPGA.

- Double click on iMPACT.
- Double click on boundary scan chain.

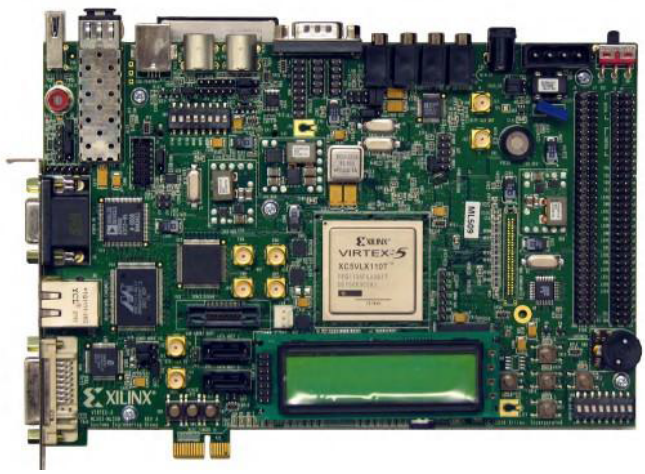

Fig. 4. Virtex 5 Board from Xilinx used in this work.

Device Utilization Summary			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	3,613	69120	5%
Number of Slice LUTs	9,094	69120	13%
Number of fully used LUT-FF pairs	2,262	11039	17%
Number of bonded IOBs	4	640	0%
Number of Block RAM/FIFO	11	148	5%
Number of BUFG/BUFGCTRLs	4	32	12%
Number of DCM_ADVs	1	12	8%
Number of DSP48Es	4	64	6%

Fig. 5. Device utilisation summary of OR1200 with UART core processor.

- There are three configuration mode select DIP switches mode [2:0], the mode select must be at value 1 0 1.
- Right click and initialize chain.
- Bypass each device to reach at FPGA chip and configure it with the bit file.
- When it will ask to add SPI flash device, and ignore it.
- Right click on the FPGA chip and programmed it. It takes some time and shows program successful.

In this way, FPGA can be programmed by downloading bit file into it.

2) Programming FPGA with SPI Flash: Serial Peripheral Interface (SPI) is a four wire, synchronous serial data bus and is used by SPI flash memories. This serial communication is now used to configure the Xilinx FPGAs. This system consists of a master and a slave device.

In this work, the SPI flash memory has been programmed using the in-direct system programming method. This method involves the use of iMPACT tool with the graphical user interface. We need to generate SPI flash PROM image file whose extension is `*.mcs` file. When `*.mcs` file is created then nearly same procedure is used to configure SPI flash. SPI flash is programmed and when we turn the power OFF and ON the SPI loads the configuration file into FPGA device and the design is implemented.

B. Software Platform

Once the hardware flow completes, then we need to develop the software platform. Hardware can not work alone. Software toolchain needs to be configured and installed to cross-compile the firmware for the specific architecture, i.e. OR32-elf. This toolchain includes the GNU Binutils, GCC, GDB, and or1ksim. GNU toolchain is required to convert the `C` or `Assembly` source code into the executable OpenRISC instructions. To install these toolchains, Cygwin runs on Windows operating systems which provides the LINUX-like environment [26]. It is a little bit easier to install these GNU tools on Cygwin environment.

1) Cygwin Environment: In 1995 Cygnus solution developed Cygwin. It is now part of Red Hat Inc. Cygwin is a Linux-like environment for Windows. It consists of a dynamic link library named `Cygwin1.dll` which acts as an emulation layer providing a collection of tools. It provides a Linux look & feel and POSIX system call functionality. Cygwin works with all x86 and AMD Windows NT and XP. Cygwin contains many UNIX utilities, and they are used from bash shell or windows command prompt. Adding more to it, it allows programmers to write Win-32 console or GUI applications and those applications can use standard Microsoft Win and Cygwin API. So it is possible to port many significant programs. The program includes the configuring and building of GNU tools. Cygwin supports both path styles POSIX and Windows NT. Installation of Cygwin is straightforward only a few steps are required. The setup.exe for the current version is available at `www.cygwin.com`. Download the setup.exe file and double click to install. To configure and build the GNU toolchain on Cygwin, we need the following packages to install with Cygwin. Those packages are util-linux, wget, subversion, patch, gcc, make, libncurses-devel, ioperm, libusb-win32, flex, bison, and zlib-devel etc.

2) GNU Toolchain: The GNU toolchains are used to create a cross-compiling environment for the target OR1200 architecture. OpenRISC toolchain is available with 32-bit GNU toolchain supported by C and C++ which used to convert C or Assembly source file into an executable file for the specific target. All these tools are freely available from open source community under LGPL license. In this work, these tools are configured, built and installed for target OR1200 architecture [26], [27], [28]. The OpenRISC toolchain is available in two forms:

- Based on newlib library for metal bare use.
- Based on µClibc library for Linux applications.

3) OR1KSIM: The or1ksim is the low-level simulator which simulates the behaviour of OpenRISC processor based on the executable file. `C` source code level debugging can be performed by or1ksim because through it we can debug the target [26]. The configuration, building and installation of or1ksim are done with the same procedure by running the configure file with a specific target and root directory. Fig. 6 shows a general procedure to install the toolchains.

4) Software Development Flow: The software development flow consists of the tools, which converts the `C` and `Assembly` source files to executable files for the target or32-elf. Firstly install all the toolchains for this OpenRISC

```
./configure --target=or32-elf --prefix=/opt/
    or32 --enable-languages=c
make all
make install
```

Fig. 6. General commands for installing toolchains.

processor. This paper describes the step by step process to create executable files. This project also provides us C source codes for drivers, support and UART interfacing with a makefile. Those makefile files work according to the flow. First, the or32-elf-gcc compiler converts C source codes into the object files. Using linking operation, the codes are converted into executable files (*.or32). Before it, all object codes are combined by an or32-elf-ar utility. The index for object codes after ar utility is produced by running an or32-elf-ranlib utility. Some files are also linked with the linker to generate an executable file. This file is enough to run on the machine to debug the processor. Two more utilities or32-elf-objcopy and bin2hex are run to create binary file and hex file respectively.

V. Result and Discussion

In this paper, two ways are presented to program FPGA using configuration bit-stream file. Firstly, the OR1200 processor interfaced with the UART module. The iMPACT tool downloads the bit file into the FPGA. In the second method, the configuration is done by SPI flash.

A. Serial Communication using UART

After successful completion of hardware and software toolchain flow, now we can debug the processor by running the small C code on the OR1200 processor. The communication between the OR1200 and the host PC is carried out using RS-232 null modem cable on the FPGA board. Once this is completed, the following steps are used to debug or run the "Hello World" example on processor and output is displayed on Windows hyper terminal. The steps are shown in Fig. 7 whereas, the output appears on the terminal window as shown in Fig. 8.

```
1) Open cygwin terminal
2) Adv_jtag_bridge -b /directory xpc_usb and
    press Entre.
3) Open another terminal of cygwin
4) Go to the directory of project
5) make all
6) or32-elf-gdb uart.or32
7) set remotetimeout 10
8) target remote: 9999
9) load
10) set $pc=0x100
11) c
12) Open HyperTerminal with the following
    setting 57600-8-N-1.
```

Fig. 7. Debugging commands of OpenRISC 1200 for software platform.

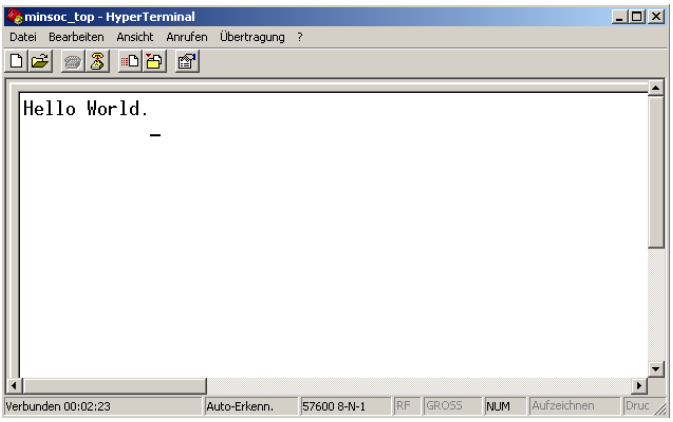

Fig. 8. Serial communication outputs on hyper terminal.

VI. Conclusion

In this paper, soft-core processors are studied, and OpenRISC 1200 processor from opencores.org is discussed in detail. The OR1200 processors with its peripheral components are implemented on a Virtex-5 FPGA device. With the help of Xilinx ISE tools, the project is created. All source code files for OR1200 and UART core are attached to the project. Xilinx ISE also does the synthesis, design implementation and bit file generation processes. FPGA is configured through two methods JTAG port and SPI serial flash, and successfully implemented. The software platform is created, configured and installed. The toolchains GNU GCC, GDB, or1ksim are installed on Cygwin which provides easy installation of these tools. The software is written in C and assembly languages. After making a successful connection between FPGA board and PC, "Hello World" program is run along some other programs like the addition of two numbers in hex are also tested successfully.

References

[1] J. G. Tong, I. D. L. Anderson, and M. A. S. Khalid, "Soft-Core Processors for Embedded Systems," in *2006 International Conference on Microelectronics*. IEEE, 2006, pp. 170–173. [Online]. Available: http://ieeexplore.ieee.org/document/4243676/

[2] Open source community, "Home :: OpenCores." [Online]. Available: https://opencores.org/login

[3] A. R. Khatri, N. Nizamani, E. Ali, and A. S. Saand, "Selecting Right FPGA for the Right Application : A Technical Survey for Xilinx FPGAs," *Quaid-e-Awam Univeristy of Engineering, Science and Technology*, vol. 15, no. 1, pp. 46–49, 2016.

[4] Altera Corporation, "Nios II Processor Reference Handbook," Tech. Rep., 2014. [Online]. Available: https://www.intel.co.jp/content/dam/altera-www/global/ja_JP/pdfs/literature/hb/nios2/n2cpu_nii5v1.pdf

[5] F. Plavec, "Soft-Core Processor Design," Ph.D. dissertation, University of Toronto, 2004. [Online]. Available: https://pdfs.semanticscholar.org/f44e/9931f3182c44f4290b70d8cc8ea53a64333a.pdf

[6] Matthew Jonathan Andrew D'Souza, "Embedded Bluetooth Stack Implementation with Nios Softcore Processor - Final Year Thesis," Ph.D. dissertation, University of Queensland, 2001. [Online]. Available: https://www.finalyearthesis.com/embedded-bluetooth-stack-implementation-with-nios-/softcore-processor/

[7] P. B. Minev and V. S. Kukenska, "Implementation of Soft-core Processors in FPGAs," in *INTERNATIONAL SCIENTIFIC CONFERENCE 23 – 24 November 2007, GABROVO*, nov 2001, pp. 1–4.

[8] R. Jesman, F. M. Vallina, and J. Saniie, "Creating a Simple Embedded System and Adding Custom Peripherals Using Xilinx EDK Software Tools," Embedded Computing and Signal Processing Laboratory, Illinois Institute of Technology, Tech. Rep. [Online]. Available: http://ecasp.ece.iit.edu

[9] Sijmen Woutersen, "The X32 Softcore A top-down approach on processor design," Ph.D. dissertation, Delft University of Technology. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.9809{&}rep=rep1{&}type=pdf

[10] Xilinx Inc., "MicroBlaze Processor Reference Guide," Tech. Rep.

[11] ——, "PicoBlaze 8-bit Embedded Microcontroller User Guide," Tech. Rep., 2004.

[12] Tenilica Inc., "Xtensa Customizable Processors," Tech. Rep., 2012.

[13] R. Gonzalez, "Xtensa: a configurable and extensible processor," *IEEE Micro*, vol. 20, no. 2, pp. 60–70, 2000. [Online]. Available:http://ieeexplore.ieee.org/document/848473/

[14] "Dual-Core LEON3FT SPARC V8 Processor User's Manual," Tech. Rep., 2018. [Online]. Available: www.cobham.com/gaisler

[15] D. L. Weaver, *OpenSPARC™ Internals OpenSPARC T1/T2 CMT Throughput Computing*, 1st ed. The address: Sun Microsystems, Inc, 2008.

[16] I. Sun Microsystems, "Opensparc™ t1 processor design and verification user's guide," The institution that published, Santa Clara, California 95054, U.S.A., Tech. Rep., 3 2006.

[17] Tsung-Han Heish and Rung-Bin Lin, "Via-configurable structured ASIC implementation of OpenRISC 1200 based SoC platform," in *2013 International Symposium on Next-Generation Electronics*, vol. 1200. Kaohsiung, Taiwan: IEEE, feb 2013, pp. 21–24. [Online]. Available: http://ieeexplore.ieee.org/document/6512280/

[18] Damjan Lampret, "OpenRISC 1200 IP Core Specification (Preliminary Draft)," Tech. Rep., 2001. [Online]. Available: www.opencores.org

[19] ——, "OpenRISC 1200 IP Core Specification," Tech. Rep., 2002. [Online]. Available: www.opencores.org

[20] ——, "OpenRISC 1000 Architecture Manual1," Tech. Rep., 2004. [Online]. Available: www.opencores.org

[21] "FPGA Soft Processor Design Considerations — EE Times." [Online]. Available: https://www.eetimes.com/document.asp?doc{_}id=1274472

[22] J. Baxter, "Open Source Hardware Development and the OpenRISC Project," Ph.D. dissertation, KTH, 2011. [Online]. Available: http://kth.diva-portal.org/smash/get/diva2:458625/FULLTEXT01.pdf

[23] C. H. . Wen, L.-C. Wang, and K.-T. Cheng, "Simulation-based functional test generation for embedded processors," *IEEE Transactions on Computers*, vol. 55, no. 11, pp. 1335–1343, Nov 2006.

[24] M. Litochevski, "Uart to bus core specifications," opencores, Tech. Rep., 2010.

[25] S. Titri, N. Izeboudjen, L. Sahli, D. Lazib, and F. Louiz, "Open cores based system on chip platform for telecommunication applications: Voip," in *2007 International Conference on Design Technology of Integrated Systems in Nanoscale Era*, Sept 2007, pp. 245–248.

[26] X. LI, "Open core platform based on openrisc processor and de2-70 board," Master's thesis, Royal Institute of Technology, School of Information and Communication Technology, Stockholm, Sweden, 2011.

[27] J. Bennett, "Howto: Porting the gnu debugger," Embecosm, Tech. Rep., 11 2008.

[28] M. Bakiri, S. Titri, N. Izeboudjen, F. Abid, F. Louiz, and D. Lazib, "Embedded system with linux kernel based on openrisc 1200-v3," in *2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, March 2012, pp. 177–182.

[29] Magnus, Själander and Habinc, Sandi and Gaisler, Jiri, "LEON4 : Fourth Generation of the LEON Processor," Aeroflex Gaisler, Kungsgatan 12, SE-411 19 Göteborg, Sweden, pp. 1–5.