ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/312130012

Integration System On Chip (SOC) in FPGA Board

Conference Paper · December 2013

CITATIONS
0

READS
413

1 author:

Abdelkarim Zemmouri
Université Ibn Tofail
18 PUBLICATIONS **47** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Image Processing Algorithms on FPGA View project

Study, design and software/hardware optimization of embedded systems: Improvement of the quality of multicolored LED display with large size FPGA-based View project

All content following this page was uploaded by Abdelkarim Zemmouri on 12 January 2019.

The user has requested enhancement of the downloaded file.

Integration System On Chip (SOC) in FPGA Board

Zemmouri Abdelkarim[1], Elgouri Rachid[2], Benbrahim Mohamed[2], Hlou Laamari[1]
[1]Laboratory of Electrical Engineering and Energy System. Faculty of Sciences, Ibn Tofail University, Kenitra, Morocco
[2]National School of Applied Sciences (ENSA), Ibn Tofail University, Kenitra, Morocco
abdelkarim.zemmouri@gmail.com, elgouri.rachid@yahoo.fr

***Abstract*— in this article, we present the implementation of an interface RS232 serial communication on a Xilinx FPGA, which allows bidirectional data transfer with an external application. The UART (Universal Asynchronous Receiver / Transmitter) is developed on the card XUPV5-LX110T prototyping Virtex-5.**

The proposed real-time application is developed around a FPGA hardware architecture that includes embedded processor Micro blaze on the FPGA VIRTEX-5. In particular, we are interested in the hardware implementation of a uniprocessor system that includes Switches, LEDs, XGI Expansion Headers and RS232 on the FPGA board. The aim of this work is to synthesize a processor on Xilinx FPGA chip, the device control and bus control UART communication as well as device control Expansion Headers inputs / outputs of the board through the implementation of a code compiled in C language on the Micro Blaze processor. This implementation has been simulated using ISim software tools and validated experimentally on FPGA Xilinx Vertex-5.

Keywords: Processor Micro blaze, FPGA, Virtex-5, EDK, SDK, RS232, UART, XGI Expansion Headers Protocol.

I. INTRODUCTION

In recent years, several architectures that combine processors and / or reconfigurable circuit (FPGA) have been proposed to accelerate the execution of applications more complex, such as digital signal generator navigation DDS [1]. It receives data and PC communication by the embedded processor Micro blaze and demodulated information to control FPGA is supported by different software to generate different navigation signals. And also the system design Micro Blaze ring which may be housed to act as a system of pulse-width modulation (PWM) [2].

Reconfigurable architectures represent an appropriate response through offering better performance compared to programmable architectures and more flexibility compared to wired solutions. It uses reconfigurable logic components that allow the user to modify the architecture in software after fabrication by software part, unlike ASICs whose algorithms are wired into the silicon. The term refers to the reconfiguration operation is to implement embedded components that present a new functionality without changing the hardware architecture of the system, such as the embedded processor that offers a Micro Blaze enables control in robotics technology [3], biological the doping level of a neural network (SNN) as the abstraction close to real neurons in the FPGA [4], signal processing and the studies are continuing to control the exchange with a satellite.

The Micro Blaze is a virtual microprocessor that is built by combining blocks of code called cores inside a Xilinx Field Programmable Gate Array (FPGA). The beauty to this approach is that you only end up with as much microprocessor as you need. FPGA contains programmable logic components called logic blocks and a hierarchy of reconfigurable interconnects that allow the blocks to be wired together. Logic blocks can be configured to perform complex combinational functions or merely simple logic gates like AND and XOR. In most of the FPGA`s the logic blocks also include memory elements which may be simple flip flops or more complete blocks of memory. You can also tailor the project to your specific needs like Flash, UART, General Purpose Input/output peripherals and etc.

The Virtex 5 board includes several input devices, output devices, and data ports, allowing many designs to be implemented without the need for any other components.

Moreover, the configuration and programming of FPGA include follow two basic steps:

First, the configuration of the hardware that focuses on the choice of devices such as a programmed processor, memory, input and associated with the FPGA outputs.

In a second step, programming and implementation of C code in the FPGA board to manage the system.

These operations are to discover new tools from Xilinx EDK includes software (Xilinx Embedded Development Kit) which gives access to both environments: XPS (Xilinx Platform Studio) responsible in creating purely embedded components and the connections respectful enter them (Hardware side) and SDK (Software Development Kit) responsible in the programming of these components by the C/C++ (software side),

Writing software to control the MicroBlaze processor must be done in C/C++ language. Using C/C++ is the preferred method and is the format that the Xilinx Embedded Development Kit (EDK) software tools expect. The EDK tools have built in C/C++ compilers to generate the necessary machine code for the MicroBlaze processor and to update a Bite stream and download it an the board as figure 1 shows.

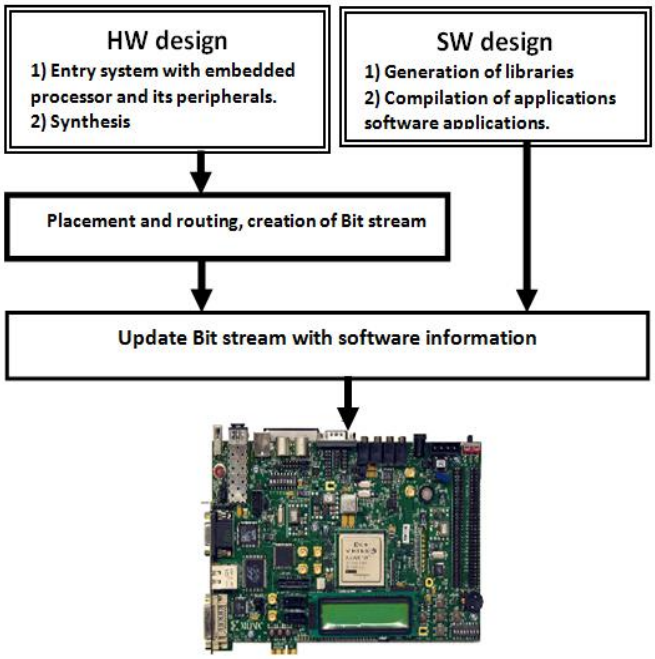


Figure 1: Standard FPGA design flow.

II. CONFIGURATION OF HARDWARE PART FOR LEDS AND DIPS SWITCH ON THE BOARD

In this context we will configure and implement the Micro blaze processor which is laid under license of Xilinx with its internal devices in a FPGA.

To configure these devices, we use a FPGA Virtex-5 XUPV5-LX110T and XPS environment of Xilinx ISE Design Suite 12.1 [5], [6]. Modules Micro blaze system used in this section are the following:

- Micro blaze.
- BRAM.
- LMB controller for the BRAM.
- LEDs 8Bit.
- DIPs Switch 8Bit.
- RS232 UART 1.

The Micro Blaze core is organized as a Harvard architecture with separate bus interface units for data accesses and instruction accesses. Micro Blaze does not separate between data accesses to I/O and memory (it uses memory mapped I/O). The processor has up to three interfaces for memory accesses: Local Memory Bus (LMB), IBM's On-chip Peripheral Bus (OPB), and Xilinx CacheLink (XCL). The LMB provides single-cycle access to on-chip dual-port block RAM (BRAM).

Using the XPS environment and after the hardware configuration, Figure 2 shows the final result as the different blocks (devices) and their connections with the Micro blaze processor.

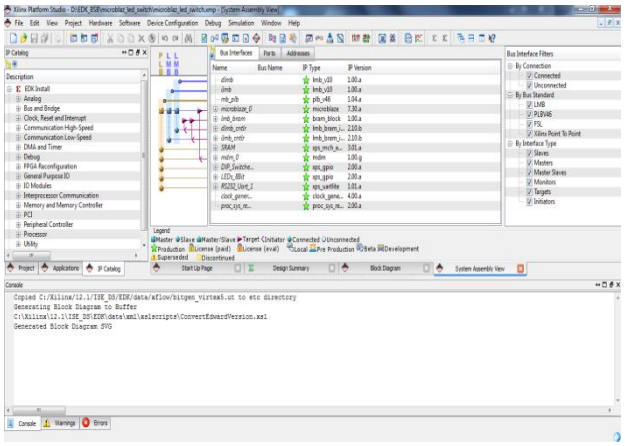


Figure 2: Result of configuration in XPS.

The verification step is necessary in this phase before moving into the configuration of the software part, in which case you can generate the file to be downloaded into the FPGA. Just run the commands in the XPS menu. This part is generated libraries and drivers for each device when the processor associated them by the generation of the bit stream.

III. CONFIGURATION OF THE SOFTWARE PART

In this section, we will first export our database project XPS SDK. We will show the concept of communicating with a device from the C code for the functionality of the Micro blaze [7].

The C code will query the switches continuously so that any change will be reflected in the terminal window. More specifically, it will read the DIP switch settings and display them on Putty terminal or the terminal SDK environment.

Once the file "system.bit" is generated in the hardware and compiled in the software part, C program can be loaded into the FPGA. Figure 3 shows the simulation results on the screen of Putty terminal.

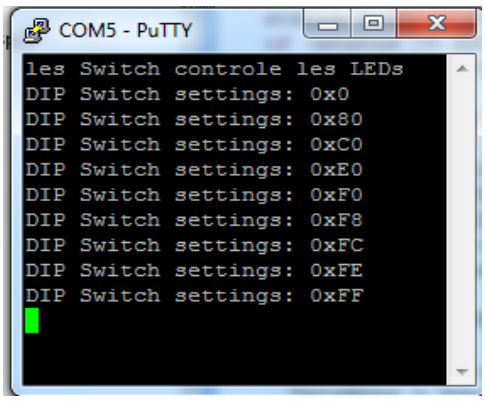


Figure 3: Simulation results on Putty terminal.

We should see "DIP switch settings: 0×0" written in the terminal window. If the DIP switch is changed to ON and OFF, it will change the display terminal to reflect the new settings. The parameters are displayed in hexadecimal number, where 0×0 means zero (all switches are in the OFF position) and 0xFF means that all switches are ON. The least significant bit corresponds to 8 switches while the most significant bit

corresponds to one switch. In the screenshot below, it has gradually turned all switches ON.

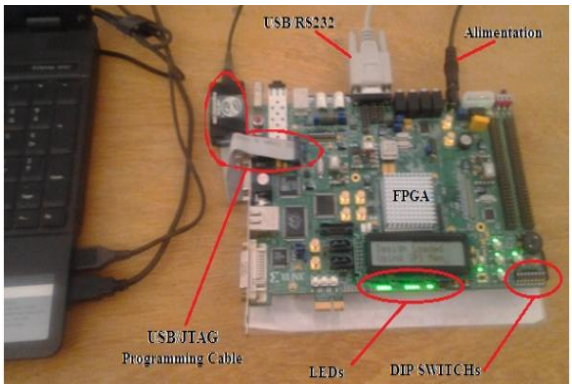


Figure 4: Experimental Validation on the board Virtex 5 XUP in connection with the PC.

IV. Addition, Configuration and Programming of two externals peripherals using the dip switch and XGI expansion headers.

The Virtex 5 board contains expansion headers for easy expansion or adaptation of the board for other applications (Figure 5). The expansion connectors use standard 0.1 inch headers. The expansion connectors contain connections to single-ended and differential FPGA I/Os, ground, 2.5V/3.3V/5V power, JTAG chain, and the IIC bus. All signals on connectors J4 and J6 have matched length traces that are matched to each other. All differential signals are routed with 100Ω differential trace impedance.

Header J6 contains 32 single-ended signal connections to the FPGA I/Os. This permits the signals on this connector to carry high-speed, single-ended data. All single-ended signals on connector J6 are matched length traces. The VCCIO of these signals can be set to 2.5V or 3.3V by setting jumper J20.

Figure 5: Extensions connectors (XGI) and 8 Dips Switches.

By adding to the hardware configuration uniprocessor two externals peripherals using the XPS interface "Create or import a new peripheral Wizard" command [7], the first is the Dip switch 1 will enter a signal of 1byte (On or Off) and the second is the pin N°18 of J6 connector that will provide output signals to an external LED (Figure 6). In our experience the J20 adjusted at 3.3v.

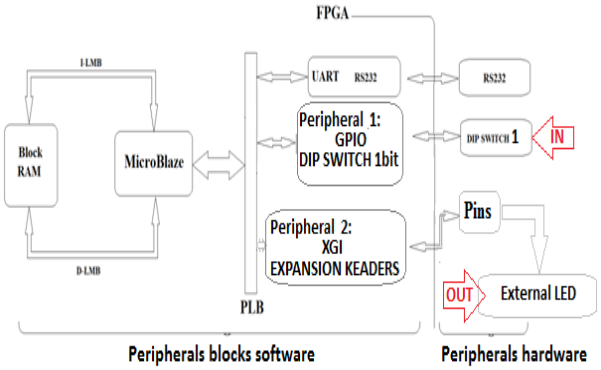


Figure 6: blocks associated with the Micro blaze

Creating embedded custom peripherals using Xilinx Platform Studio (XPS) is straightforward because XPS automates most of the design creation. The Base System Builder (BSB) wizard reduces the design effort to a series of selections. We can then further customize our design in Project Navigator and XPS. Design customization can be as simple as tweaking a few parameters on existing intellectual property (IP) cores (for example, changing the baud rate for the AXI UARTLite), or as complex as designing custom IP and integrating it into the existing design.

It creates for each custom peripheral the framework of the design, including bus interface logic, LUTs, registers, and provides an HDL template so that we can integrate our custom logic in an understandable manner.

The tables I and II, shows the number of the used logic slice Registers, Slice LUTs and others for led_1_0 peripheral and switch_1_0 peripheral.

TABLE I. POST SYNTHESIS DEVICE UTILIZATION OF led_1_0

Resource Type	Used	Available	Percent
Slice Registers	143	69120	0
Slice LUTs	60	69120	0
LUT Flip Flop pairs used	157	NA	NA
fully used LUT-FF pairs	46	157	29
unique control sets	14	NA	NA
IOs	202	NA	NA
bonded IOBs	0	640	0

TABLE II. POST SYNTHESIS DEVICE UTILIZATION OF switch_1_0

Resource Type	Used	Available	Percent
Slice Registers	40	69120	0
Slice LUTs	23	69120	0
LUT Flip Flop pairs used	53	NA	NA
fully used LUT-FF pairs	10	53	18
unique control sets	9	NA	NA
IOs	202	NA	NA
bonded IOBs	0	640	0

The communication that takes place between the peripherals and the Micro blaze is OPB. The significance of the top level module of your custom peripherals must contain a set of bus ports that conforms to OPB or PLB (Processor Local Bus) protocol, so it can be connected to the system or OPB, PLB bus. See figure 7.

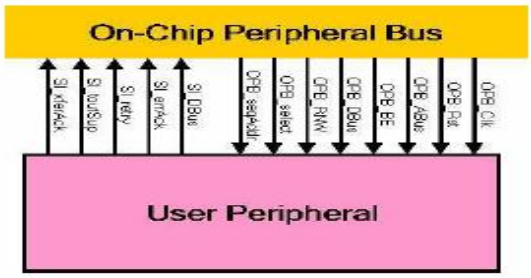


Figure 7: Example the OPB bus protocol used in a Micro Blaze system [7].

V. SIMULATION AND EXPERIMENTAL RESULTS

After the implementation of custom peripherals and Micro blaze processor on the FPGA through the SDK, we charged a compiler C language in the memory of the FPGA in order to execute it by the Micro blaze processor.

Using the Xilinx® Microprocessor Debugger (XMD) to debug programs and verify systems using the Micro Blaze processor as show in figure 8 and 9.

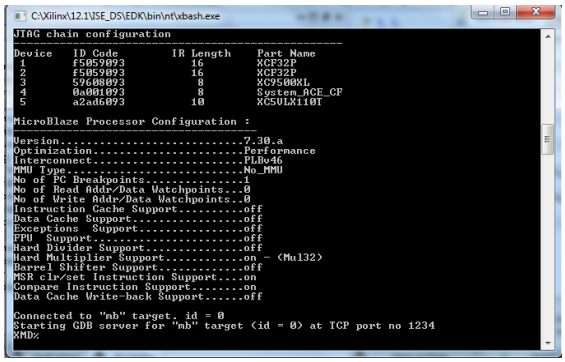


Figure 8: TCP/IP connected with the local host and the targeted device.

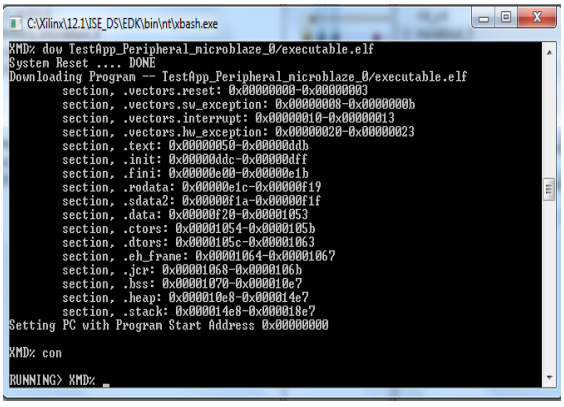


Figure 9: Downloads the given ELF (with the data option) onto the memory of the current target and Continues from current PC <Execute Start Address>.

In ISim simulator software (Figure 10), we force the input clock (switchè_1_0_switch_pin) to the value 1 and we observe the change in output (led_1_0_led_pin). With the period 10 ns of simulation of the system clock, we note that the output follows the input, for example when the input is not enabled automatically output = 0 (area A) against during the activating of the input (switch on, 1), the output become 1 (X = 1, area B) that is to say, the LED lights (figure 12) and so on. The rs232_uart is activated throughout the simulation to provide feedback Micro blaze in the screen (Putty).

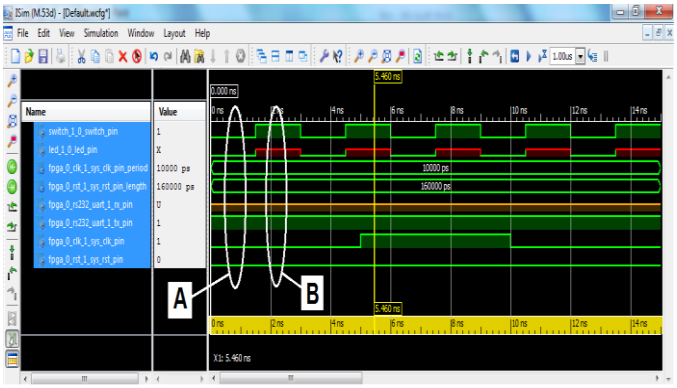


Figure 10: Simulation of the input and output with ISim.

Experimentally the outputs are obtained by giving orders to the respective devices and checking the change happens on Virtex5 board. For data orders, the output will be displayed and based on the UART communication. The change of LED status shows the same as the output of the simulation and the control device by the UART communication. By using the Xilnix12.1 EDK tools the following results are obtained (figure 11 and 12).

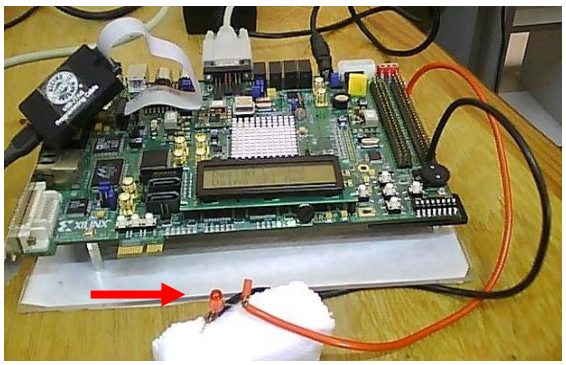


Figure 11: Experimental set up Virtex5 board, Dip Switch 1 "OFF".

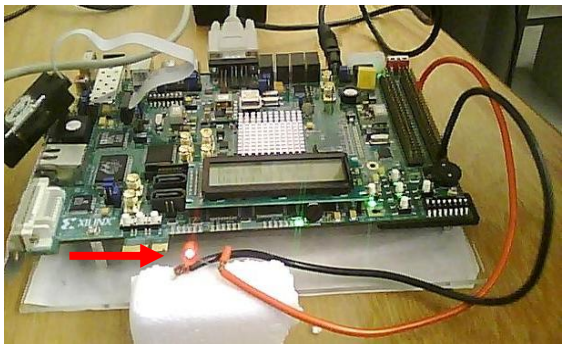


Figure 12: Experimental set up Virtex5 board, Dip Switch 1 "ON".

VI. CONCLUSION

The objective was achieved by designing an embedded based project on EDK using Micro Blaze as a processor where the operations are performed by the processor and developed on XILINX 12.1 and implementing the code in C language in a FPGA Virtex-5 LX110T-XUPV5 Xilinx. By this we can conclude that by using a serial communication we can control the Input/output peripherals which will be having applications like controlling the traffic lights and the power circuits (like the electrical engine) etc. The out puts are verified by giving the commands in the putty terminal and we can see the output status of each peripheral on the same terminal also.

References

[1] S. MohammedInthiyaz, B.Sumithra, "FPGA Implementation of Radio Navigation Based on MicroBlaze," International Journal of Engineering Research and Applications (IJERA) Vol. 3, Issue 3, May-Jun 2013, pp: 1032-1039, ISSN: 2248-9622.

[2] R. MAZIN KHALIL, SAJA B. MAHMOOD, "DESIGNING OF A PULSE WIDTH MODULATION SYSTEM USING EMBEDDED SYSTEM DESIGN TECHNIQUES," Journal of Theoretical and Applied Information Technology, Vol. 49, No.1, March 2013, pp :101-106, ISSN: 1992-8645, E-ISSN: 1817-3195.

[3] SG Tzafestas, KM Deliparaschos, GP Moustris, "Fuzzy logic path tracking control for autonomous non-holonomic mobile robots: Design of System on a Chip," School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece , pp: 1017-1027, 2010.

[4] LP Maguire, TM McGinnity, B. Glackin, A. Ghani, A. BELATRECHE, J. Harkin, "Challenges for large-scale implementations of spiking neural networks on FPGAs," School of Computing and Intelligent Systems, Magee Campus, University of Ulster, Derry, Northern Ireland, pp: 13-29, 2007.

[5] Xilinx Virtex-5 FPGA Configuration User Guide.

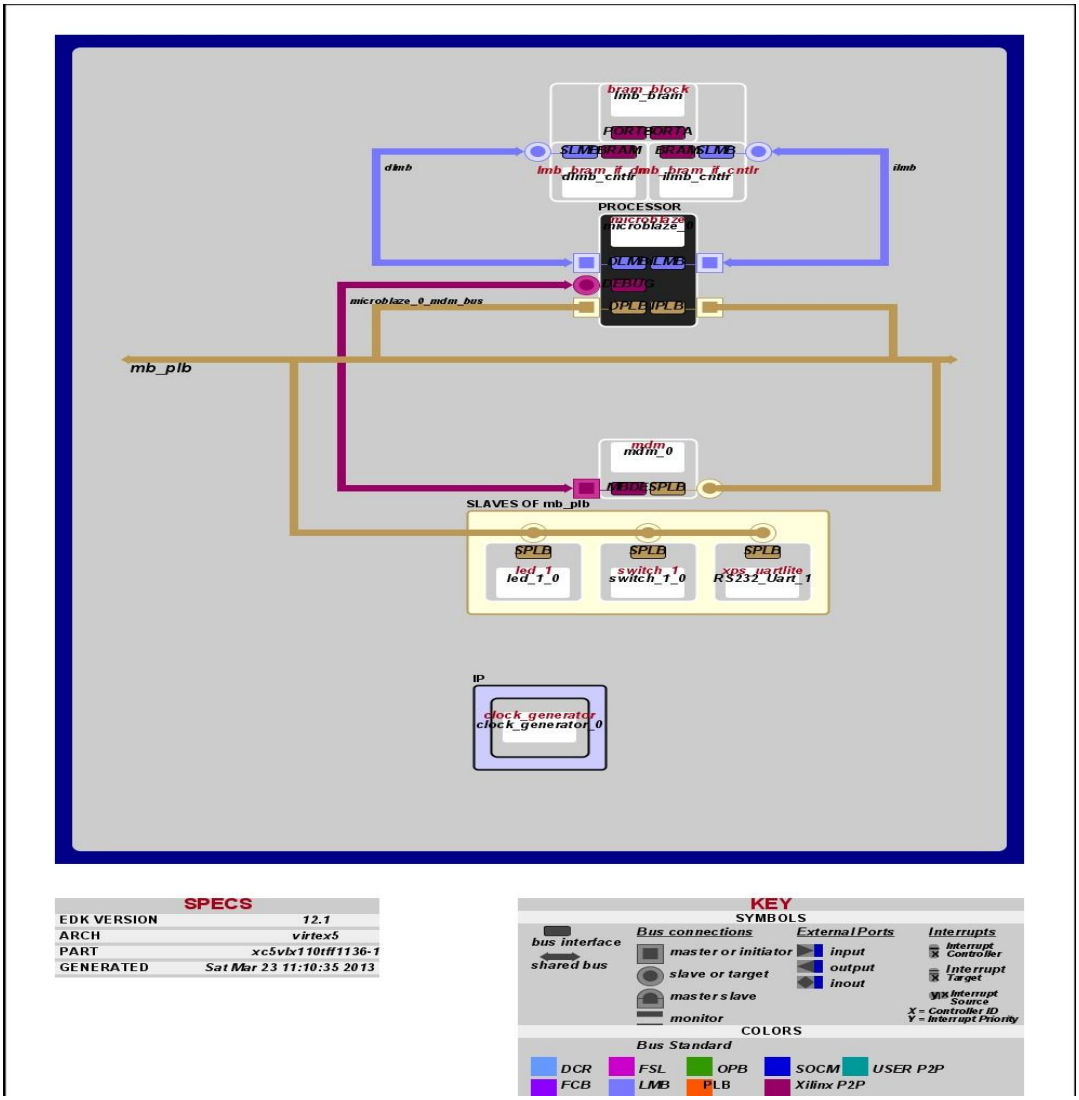
[6] Xilinx Micro Blaze User guide.

[7] Micro Blaze Tutorial Creating a Simple Embedded System and Adding Custom Peripherals Using Xilinx EDK Software Tools by Rod Jesman Fernando Martinez Vallina Jafar Saniie for Illinois Institute of Technology.

[8] Xilinx Virtex 5 Pro datasheet.

[9] www.Xilinx.com

[10] www.Wikipidea.org



The resultant block diagram of hardware issued by the Xilinx platform studio (XPS).

View publication stats