Paper No. T4-3.1, pp. 1-3

13th INTERNATIONAL SYMPOSIUM on POWER ELECTRONICS - Ee 2005
XIII Međunarodni simpozijum Energetska elektronika – Ee 2005
NOVI SAD, SERBIA & MONTENEGRO, November 2nd - 4th, 2005

FPGA IMPLEMENTATION OF AN RFID HUB

Marius-Cristian CERLINCA, Adrian GRAUR, Valentin POPA
"Stefan cel Mare" University of Suceava, Faculty of Electrical Engineering, Suceava, Romania

Abstract: *What exactly is RFID? RFID stands for Radio Frequency Identification. A typical system consists of a radio-enabled device that communicates or interrogates some tags or labels. If we will use RFID technology in security applications, we will need much more than one reader. There are two commercial solutions: each reader has one computer (or embedded device) or we can use a RFID multiplexer. First solution is very expensive and the second is time consuming, as we will see later. Our solution is to design a so-called "RFID hub" using FPGA technology and Xilinx embedded software tools.*
Key Words: ***RFID, FPGA, SoC, MicroBlaze, Xilinx EDK, embedded***

1. INTRODUCTION

RFID technology is often envisioned as a replacement for UPC (Universal Product Code) or EAN (European Article Numbering) bar-codes, having a number of important advantages over the older bar-code technology. The main advantages of RFID technology over old bar-code technology are: the ability to hold much more data, the ability to change the stored data as processing occurs, RFID does not require line-of-site to transfer data, and is very effective in harsh environments where bar code labels won't work.

What is a RFID hub and why should we develop one? The most RFID systems consist of one reader that is connected to a PC or some other device that read/write data from/to tags. If you want to develop a security system with 8 doors, you will have to use 8 PC's or 8 RFID dedicated embedded systems and 8 readers. This, of course will be a pretty expensive system.

Another solution is to use a RFID multiplexer witch in fact is an electronic device that allows a reader to have more than one antenna. Each antenna scans the field in a preset order. This reduces the number of readers needed to cover a given area, such as a dock door, and prevents the antennas from interfering with one another. From the costs point of view this is the best solution. However, there are some major disadvantages: the read time is 8 times slower and the write procedure can be a tricky issue because we don't know exactly haw faster the tags are trespassing the reader area.

To solve these problems we design a system that will have a much better average read/write speed with only one so called "RFID hub" witch is using at the same time up to 8 RFID readers.

2. RFID HUB STRUCTURE

In order to implement the RFID hub, we used Xilinx FPGA's. For embedded SoC design and FPGA synthesis we used Xilinx EDK (Embedded Development Kit) and XILINX ISE. Our system consists of (see Fig.1):

- one Spartan II FPGA
- an application for the Microblaze processor that will read/write tags (using the RTOS operating system)
- a PC server application that can handle multiple RFID hubs

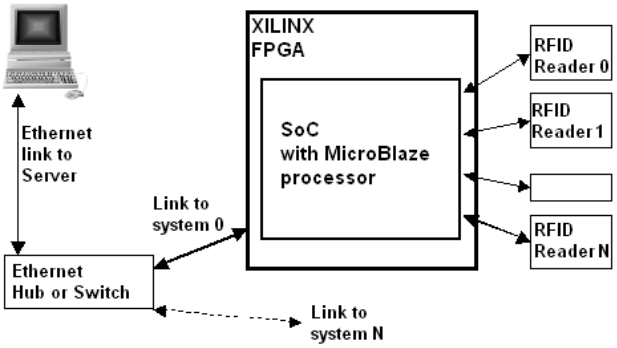

Fig. 1. *RFID Hub implementation*

This solution will be at the middle if we think at costs, but will solve the security holes that can appear with a multiplexer solution.

2.1. TE-XC2S Board

In order to test our SoC design we used the TE-XC2S board from Trenz Electronic Gmbh, which is equipped with a XILINX Spartan II FPGA. TE-XC2S development system provides a cost-effective and fast access to FPGA technology to engineers and students. It is designed for small to medium-sized designs and it could cover the most aspects of everyday usage. The system has four expansion connectors and is equipped with SRAM, FLASH, 7-segments displays, and could be equipped with other peripheral circuits also. [1]

2.2. RFID Reader

As RFID readers we used ISC.M02-B (Fig. 2) from FEIG Electronic GmbH which works with Smart Labels with an operating frequency of 13.56 MHz (ISO 15693, EPC).

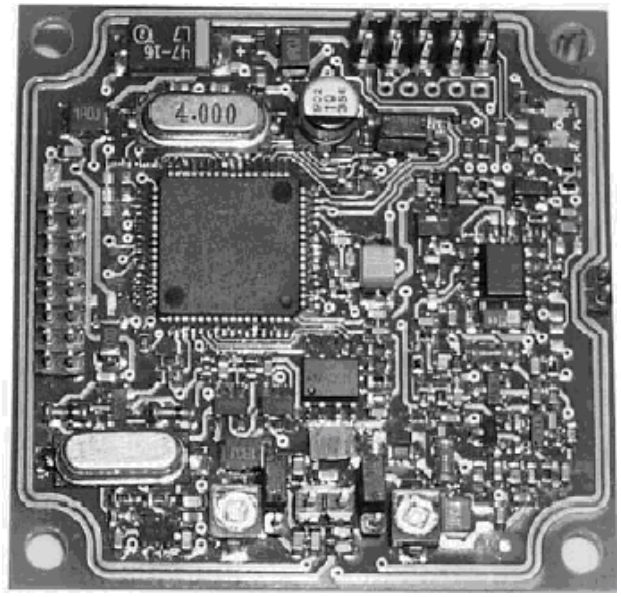

Fig. 2. *ID.ISC.M02 RFID Reader*

The Read/Write PCB has a maximum reading distance of up to 10 cm* by using the integrated antenna. By using external antennas, distances of 7 cm (antenna size 30 x 40 mm) resp. 14 cm (antenna size 100 x 100 mm) will be reached. To communicate with a PC or some other embedded device it uses UART (RS232-TTL) interface.

3. SOC STRUCTURE

The SoC is implemented using Xilinx specific software tools: Xilinx EDK for MicroBlaze development and Xilinx ISE for FPGA place and route and synthesis.

For the first tests we used just 4 UART interfaces, because just one compilation (generate libraries, build user applications, generate linker script, generate netlist, generate bitstream, download) of such a SoC is taking about 10 minutes (see Table 1 – just the FPGA routing part of the process).

Table 1. *Total Real Time for FPGA route*

Phase	Unrouted	Total Real Time (secs)
1	11457	32
2	10163	39
3	3457	45
4	3457	46
5	3917	114
6	4044	130
7	0	261
8	0	265

The SoC that will have synthesized contains (see Fig. 3):

- 1 Microblaze processor
- 4 UART cores for every processor we use that are synthesized in the same FPGA
- 1 Ethernet core for sending data to the central PC
- 1 debug core
- 1 Generic External Memory core

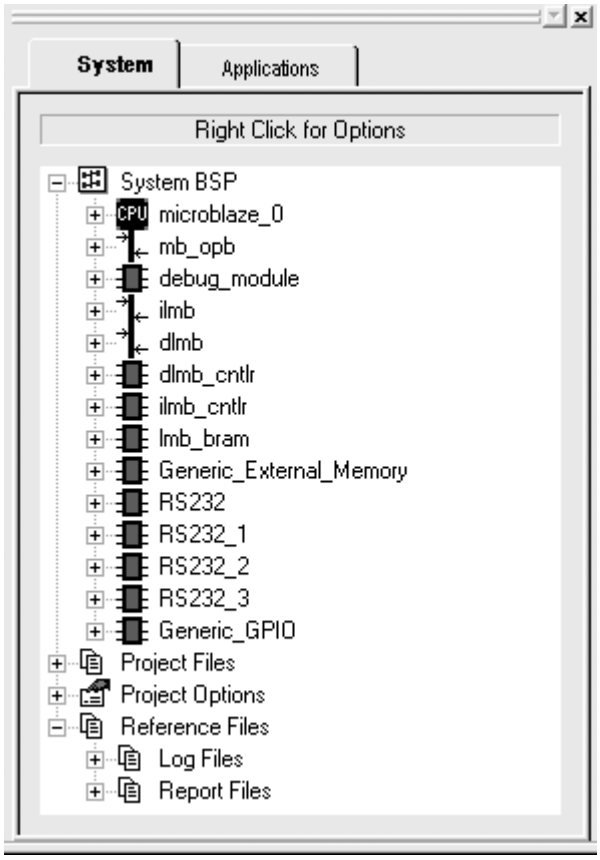

Fig. 3. *SoC Structure (Xilinx EDK)*

The "Generic_External_Memory" core is a Xilinx core and is using the SRAM Memory from TE-XC2S board (512 K). This is needed because the BRAM memory (with Xilinx EDK) that Spartan 2 XC2S200 is using for local data and instruction memory is just 4K bytes and is not enough for our applications.

Generic External Memory core , debug core and UART cores (from RS232 to RS232_3) are connected to the OPB (On-chip Peripheral Bus) of the MicroBlaze processor.

The debug core is used to control the processor and his peripherals. All the OPB cores are in fact memory mapped addresses that can be readed / writed using XMD debugger from Xilinx EDK (see Fig. 5).

The entire SoC designed and developed for the RFID hub is using 1453 out of 2352 (61%) slices from a XC2S200 FPGA.

3.1. Ethernet core [2]

The Ethernet core consists in fact in two separate Verilog modules: one for reception that is using just one input from the FPGA (rxd) and another module for sending data to the PC, which is using two output ports (txdp, txdm). This part of the SoC is using a 48Mhz clock so, if another target system is used, some DLL's (Delay-Locked Loop) should be instantiated in order to

assure a clock that is closer to this value. From our tests, a clock between 46MHz and 50Mhz should work just fine.

The Ethernet interface was implemented using the 10BaseT standard so we have a maximum speed of 10Mb/s.

For the receiver part we used:

- a differential receiver circuit (see Fig. 4);
- a clock extraction circuit ;
- a preamble synchronizer, de-serializer and Ethernet checksum checker.

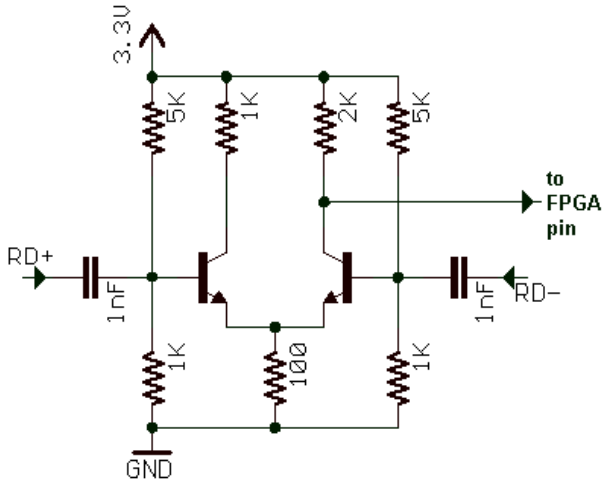

Fig. 4. *Differential circuit for reception*

The differential circuit for receiving data is used because Spartan 2 FPGA's does not offer differential inputs / outputs. However, this is not the case if you are using Spartan 3 or Virtex FPGA's.

After synthesis the Ethernet part of the SoC is using just 3% of our target (XC2S200).

We didn't use the Xilinx Ethernet core because is too complex and use too much space on the FPGA. Our core was developed in fact for another SoC [1] but we imported [3] to this specific MicroBlaze project using "Create – Import Peripheral" tool which comes with Xilinx EDK software.

3.2. UART core and IEE 15693 commands

RFID readers we used (ID.ISC.M02) use UART RS232-TTL port in order to be connected to a PC or another embedded device. The settings we used for every UART port inside the SoC are: 38400, 8 data bits, 1 stop bit and odd parity.

In order to communicate with RFID readers we implemented the next C functions that send ISO 15693 commands:

- OBID_Baud_Rate_Detection() : to detect baud rate of the reader;
- OBID_Inventory() : returns the number of transponders on the field and their unique ID's;
- OBID_Read() : read data from one transponder;
- OBID_Write() : write data to transponder;
- OBID_RF_Reset() : resets the RFID reader;
- CalcCRC16CheckSum() : to calculate the CRC sum of packets sent or received from RFID device.

4. THE SERVER PROGRAM

In order to send/receive different commands to/from a PC linked to this type of SoC we developed a PC server program. Because we use our own Ethernet core, we just modified the "Ethernet IDE application" [1] and used the SRAM part like useful data. All mapped addresses (see Fig. 5) inside the MicroBlaze SoC can now be accessed through this application. Also, there are 2 variables (one unsigned char and the other unsigned char [100]) at fixed addresses on the MicroBlaze part that are used for receive/send commands/data from/to the PC server application.

eripherals | Bus Connections | Addresses | Ports | Parameters

Assign Address ranges for all peripherals and bus arbiters

Instance	Prefix	Base Address	High Address
mb_opb			
debug_module		0x41400000	0x4140ffff
dlmb_cntlr		0x00000000	0x00000fff
ilmb_cntlr		0x00000000	0x00000fff
Generic_External...	MEM0	0x20080000	0x200fffff
RS232		0x40660000	0x4066ffff
RS232_1		0x40640000	0x4064ffff
RS232_2		0x40620000	0x4062ffff
RS232_3		0x40600000	0x4060ffff
Generic_GPIO		0x40000000	0x4000ffff

Fig. 5. *Address ranges for SoC peripherals*

The program was developed using Microsoft Visual C++ and the WinPCap Ethernet sniff library developed and maintained at Politecnico di Torino, Italy.

For a faster Ethernet performance we used two windows threads: one for writing the data over the Ethernet interface and one for reading' it.[1]

4. CONCLUSIONS AND FUTURE WORK

Because the commercial RFID multiplexers can be slow sometimes, we tried to de design a low-cost RFID hub. Using FPGA and SoC technologies we succeeded in our attempt to offer another solution for security applications which use RFID technology. In the future we will try to adapt our own SoC [1] for an RFID hub, without using the MicroBlaze processor and Xilinx EDK. This way, the so-called RFID Hub can be implemented no matter the FPGA vendor.

5. REFERENCES

[1] M.C. Cerlinca, A. Graur, "Building a SoC Architecture in an FPGA" , AECE, Suceava, 2005

[2] M.C. Cerlinca, A. Graur, "FPGA Implementation of a 10BaseT interface", Tehnologii Informaționale, Suceava, 2004

[3] ***, "Designing Custom OPB Slave Peripherals for MicroBlaze.pdf" – Xilinx Literature

[4] IEE 15693 Standard

[5] IEE 802.3 Standard