Some considerations regarding software/hardware implementation of DES algorithm for an RFID enabled device

Marius Cristian Cerlinca, Adrian Graur

Faculty of Electrical Engineering and Computer Science,
"Stefan cel Mare" University of Suceava, Romania
mariusc@eed.usv.ro, adriang@eed.usv.ro

Abstract

Let's suppose that you want to build some device that use FPGA/ASIC technology and your system should provide a certain level of security when dealing with data from external world. This means that some data should be somehow encrypted. Two important questions could appear in this stage: first, what encryption algorithm should we use and second, do we need a hardware implementation or software one? In this paper we will provide some answers, taking in consideration a real world application.

1. Introduction

First, let's describe our real world application that needed an encryption algorithm. We developed a so-called "RFID hub" (1) that use FPGA technology and is functioning almost the same as a RFID multiplexer.

There are in fact two different implementations of our hub, as you can see in figures 1 and 2:

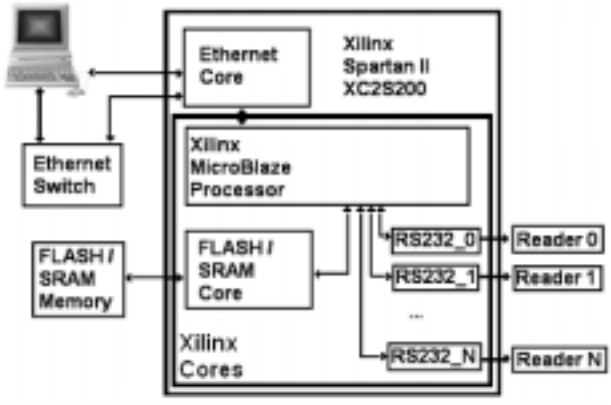

Figure 1: Xilinx dependant RFID Hub (2)

First implementation was developed using Xilinx EDK software and consequently, Xilinx dependant cores. As you can see in figure above, we used our own Ethernet Core (3) and the rest of them (Xilinx Microblaze, SRAM/FLASH, RS232) were generated using Xilinx EDK software IDE.
The second implementation (see figure 2) of our RFID Hub was vendor independent so this could be synthesized using ASIC devices. There are a lot of discussions what version is best when we take in consideration different aspects like costs, communication speed, and so on. We will not made this discussion here, we will try only to present our solutions regarding the encryption algorithms for every implementation of our RFID enabled device.

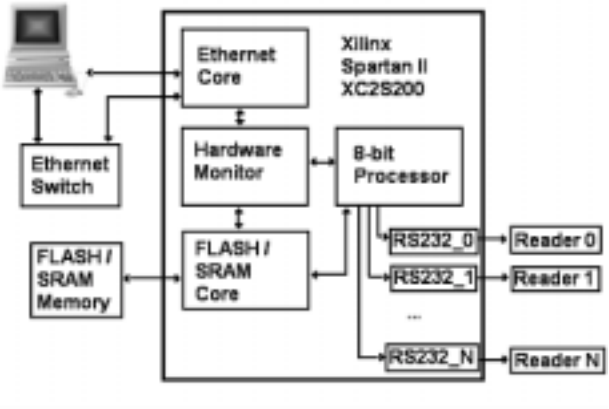

Figure 2: ASIC ready RFID Hub (2)

For a better understanding of the system, our device should use up to 8 RFID readers in the same time, but much more faster than commercial RFID multiplexers which scan the field in a preset order and this in fact means that a single RFID reader is working at a time.

1.1. Security needs

If we take a closer look at the pictures above, the device is communicating data two ways:

- first communication channel is the Ethernet through some data is send/received to/from a central PC server;
- the second communication channel is not so obvious when first look at the figures, but is in fact real data that is read or write from/to RFID labels.

We can consider that first communication channel is somehow safe because this is in fact a local network with no connection to Internet or some other networks. Just imagine this could be a security system for a big building, with no connections to external world.
The second communication channel is the data that is write or read to/from RFID labels. We cannot change the ISO 15693 standard to made somehow a safe communication with these devices, but we can encrypt actual data from them. If we do not provide an encryption mechanism, anybody can read this RFID labels and can duplicate them. The worst-case scenario is the implementation of such a system in a supermarket without any encryption algorithm provided.

2. The solutions

2.1. Why software/hardware implementations?

We had to choose an algorithm that could be implemented both in software and hardware. You may ask why do we need a hardware implementation if we are using some processors inside our SoC? There are a number of answers to such a question:

- our second solution which is ASIC ready should use as little memory space as possible, so a software implementation could be costly from memory point of view;
- again, this second solution uses a processor that does not provide a C/C++ compiler (4), but only an assembler, and a encryption algorithm implementation using just assembly language can be tricky;
- if we are taking in consideration the first version which uses a Microblaze processor, a software encryption algorithm developed in C/C++ should meet all performance requirements. Also we can use BRAM memory specific to Xilinx FPGA's for a software implementation.

Taking this in consideration we needed both software and a hardware implementation of an encryption algorithm.

2.2. Choosing an algorithm

Our final choice of an encryption algorithm should meet some conditions:

- easy to implement in software and hardware;
- to be a block cipher;
- the software implementation should not use too much memory;
- the hardware implementation should use just a small number of logical gates;
- both software and hardware implementation should be fast enough for a normal RS232 communication;
- to algorithm should be safe enough and use keys with minimum length of 8 bytes (length of RFID unique label).

We identified two candidates: DES algorithm and AES (Rijndael) algorithm. We choose the DES algorithm because after some preliminary tests we found that both the compiled source and the synthesized cores were much more suitable to our needs.

2.3. RFID Label structure

In order to follow our own defined security needs (5), data written on transponders will have the structure described below (see Table 1):

- 4 bytes used for authentication (use of transponder unique ID and data that follows) (0xAB, 0x49, 0xF1, 0x52) computed with a SHA1 type algorithm;
- 2 bytes used to store the length of real data (0x00, 0x10);
- real data (DES encrypted) (0x35, 0x9A, … , 0x01, 0x0D)

Real data from transponder is usually DES encrypted. There are some exceptions where we don't use DES, but another very simple encryption algorithm. For example on powerful embedded devices like Microblaze Soft Processor we used DES encryption but for 8051 compatible devices (which are not using FPGA's) we used other very simple encryption algorithm.(5)

0xAB	0x49	0xF1	0x52
0x00	0x10	0x35	0x9A
0x22	0x01	0x95	0x42
0x00	0x66	0x17	0x03
0xB6	0x1A	0x9F	0x90
0x01	0x1D		

Table 1: Real data on transponder

3. DES Implementations

Once we choose our encryption algorithm, we had to implement both the software and hardware versions for our different versions of RFID Hubs.

3.1. DES Software implementation

We implemented the standard DES ECB (Electronic Code Book) using ANSI C standard. The compiler used is GNU mb-gcc (microblaze gcc, used inside Xilinx EDK IDE, see Figure 3). To test the speed of the software algorithm we implemented two versions of DES:

- a simple and short one, with no improvements in speed at all, we just wanted this version to be ass small as possible;
- a version improved for speed, but without any requirement regarding the memory used.

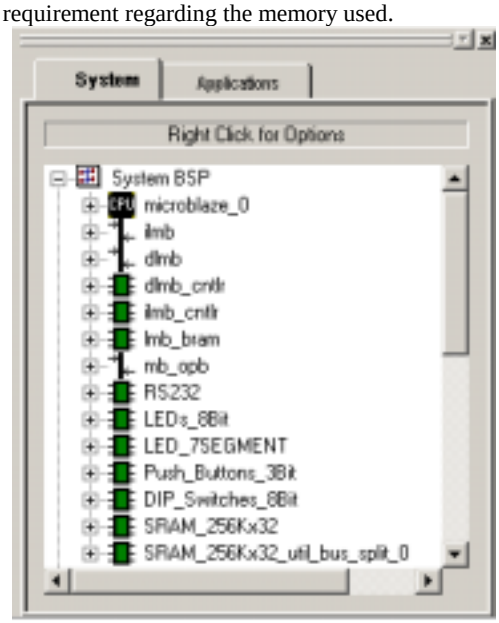

Figure 3: Xilinx EDK Screenshot

Finally, after testing both versions we decide that the first one is the best for our application, because both of them worked just fine with RFID readers attached to the system, so the speed was not a real issue, but when compiled, first one is using just 6K of memory against 54K for the second one. And 6K of memory means in fact that we can use a cheap Xilinx Spartan II/III 200K Device (11K of BRAM memory) instead of a much more expensive Virtex 2/3.

3.2. DES Hardware implementation

Our DES Hardware implementation is derived from SystemC/Verilog DES Core, by Javier Castillo Villar (see www.opencores.org). Like the software version, it is also an ECB implementation which and it is area optimized. The synthesis was done using the Xilinx ISE IDE (see Figure 4) and Xilinx Place&Route software.

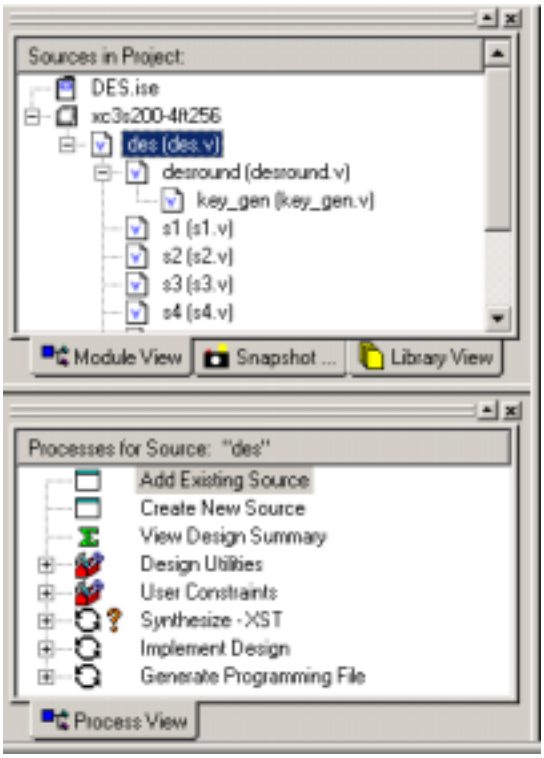

Figure 4: Xilinx ISE Screenshot

If we are using a Spartan III device, our hardware DES implementation has next characteristics:

- maximum frequency: 77.207 Mhz (maximum frequency for Spartan III is 200MHz);
- number of occupied Slices: 357 out of 1,920 18%;
- cycles per block: 16.

If this version of DES core is used, the RFID hub with 8-bit processor needs at least a device with 150K gates. The best FPGA that can be used for testing before going ASIC are Xilinx Spartan II and III. We used both types of devices with almost the same performances. Spartan III device was just a fraction faster, but we used also different clock sources (48MHz vs. 50MHz).

4. Conclusions

We wanted that our "RFID Hub" to have some encryption mechanism for data from transponders. We used two mechanisms:

- one for authorizing labels (first 4 bytes);
- one for encrypting actual data from labels (using DES).

Because we developed two versions of this device, we needed two different implementation of DES:

- a software one, implemented in ANSI C, for Xilinx dependant version;
- a hardware one, for "ASIC ready" version.

Both versions worked correctly and there is no difference from outside world between the devices. The main differences are in the way they are internally organized and how each of them is using DES algorithm.

From the point view of costs, the first version is preferable because does not require any external memory and could be used no mater the FPGA/ASIC vendor. If we take in consideration future developments and improvements, the version that is using Xilinx dependant cores is one to be preferred.

5. References

[1] Marius Cerlinca, Adrian Graur, Valentin Popa, "FPGA Implementation of an RFID Hub", 13th INTERNATIONAL SYMPOSIUM on POWER ELECTRONICS, Novi Sad, 2005, printed in Electronics, Banjaluka, 2005, ISSN 1450-5843, pp.57-59

[2] Marius Cerlinca, Adrian Graur, Valentin Popa, " FPGA Implementation of an RFID Dedicated SoC", ECUMICT, Gent, 2006, ISBN 9-08082-552-2, pp.201-204

[3] Marius Cerlinca, Adrian Graur, "FPGA Implementation of a 10Base-T Ethernet Interface", Distributed Systems, Suceava, 2004, ISBN 973-666-143-1, pp.80-85

[4] Marius Cerlinca, Adrian Graur, "BUILDING A SOC ARCHITECTURE IN AN FPGA", AECE, Suceava, 2004, ISSN 1582-7445 - No 2 / 2004, pp.77-81

[5] Marius Cerlinca, Adrian Graur, Tudor Cerlinca, "A Script Language for RFID Systems", ECUMICT, Gent, 2006, ISBN 9-08082-552-2, pp.345-352