ISTANBUL TECHNICAL UNIVERSITY

ELECTRICAL – ELECTRONICS ENGINEERING FACULTY

Hardware/Software Codesign and Implementation of a Smartcard System

BSc Thesis by

Caner Bulduk

Department: Electronics and Communication Engineering

Programme: Electronics and Communication Engineering

Thesis Advisor: Assoc. Prof. Dr. Sıddıka Berna Örs Yalçın

JUNE 2017

ISTANBUL TECHNICAL UNIVERSITY

ELECTRICAL – ELECTRONICS ENGINEERING FACULTY

Hardware/Software Codesign and Implementation of a Smartcard System

BSc Thesis by

Caner Bulduk
040120186

Department: Electronics and Communication Engineering
Programme: Electronics and Communication Engineering

Thesis Advisor: Assoc. Prof. Dr. Sıddıka Berna Örs Yalçın

JUNE 2017

FOREWORD

First of all, I would like to thank my supervisor, Assoc. Prof. Dr. Sıddıka Berna Örs Yalçın for guiding me through his advises and knowledge as well as sharing lots of her experiences with me.

Also, I would like to thank all my friends for their support. Their company has always kept me entertained and motivated in times of struggle.

Finally, I want to express my endless gratitude and appreciation to my family, who have supported my decisions and guided me with their experience.

June 2017 Caner Bulduk

TABLE OF CONTENTS

ABBREVIATIONS

FPGA : Field Programmable Logic Array
ICC : Integrated Circuit Card
ISO : International Standard Organization
RF : Radio Frequency
CPU : Central Processing Unit
RAM : Random Access Memory
ROM : Read Only Memory
EEPROM : Electronically Erasable Programmable Read-Only Memory
LUT : Look Up Table
APDU : Application Protocol Data Units
ATR : Answer to Reset
PPS : Protocol Parameter Selection
VHDL : Very-high-speed-integrated-circuit Hardware Description Language
ASIC : Application Specific Integrated Circuit
RS : Recommended Standard
DCE : Data Circuit-terminating Equipment
DTE : Data Terminal Equipment
ACS : Advanced Card Systems
MAP : Multi-application & Purse
ISE : Integrated Synthesis Environment
RFU : Reserved for Future Use
EDK : Embedded Development Kit
SDK : Software Development Kit
ISIM : ISE Simulator
RTL : Register-Transfer Level
RISC : Reduced Instruction Set Computing
XPS : Xilinx Platform Studio
SIPO : Serial In Parallel Out
UCF : User Consist File
PLB : Processor Local Bus
AXI : Advanced eXtensible Interface
IP : Intellectual Property
MPD : Microprocessor Peripheral Definition
UART : Universal Asynchronous Receiver-Transmitter
CAD : Computer Aided Design
ICC : Integrated Circuit Card
ICC : Integrated Circuit Card

SYMBOLS

V	**:** Voltage
I	**:** Current
F	**:** Clock rate conversion integer
D	**:** Baud rate adjustment integer
etu	**:** Elementary time unit
f	**:** Frequency
CLA	**:** Class byte
INS	**:** Instruction byte

LIST OF TABLES

LIST OF FIGURES

Hardware/Software Codesign and Implementation of a Smartcard System

SUMMARY

Nowadays smart cards have reached a very wide range of usage and become a part of our everyday life. The secure structure of smart cards is the biggest influence on their widespread use. Smart cards are devices that can communicate serially. In this project, an existing smart card reader was developed and implemented on an FPGA along with a processor. Implementation of the smart card reader on the FPGA will facilitate the integration of smart cards into the system in future projects.

The Spartan 3E FPGA development kit, produced by Xilinx, was used to implement the project. As a processor, Microblaze was implemented on the FPGA. As a smart card ACOS6 smart card, which is developed by Advanced Card Systems for multiple applications, was used. The smart card reader module is coded in VHDL and implemented on the FPGA. The smart card is connected to the processor via the implemented reader module. The applications for reading data from the smart card and sending data to the smart card have been implemented in the Xilinx Software Development Kit program using the C programming language.

The need to work is explained in the first part of the project. In the second part, general information about smart cards and FPGA design is given. In the third part, it is explained how smart card reader is implemented on FPGA using hardware and software development tools. Finally, the obtained results from this study has been discussed in the fourth section.

Bir Akıllı Kart Sisteminin
Donanım/Yazılım Tasarımı ve Gerçeklenmesi

ÖZET

Günümüzde akıllı kartlar oldukça geniş bir kullanım alanına ulaşmış ve günlük hayatımızın bir parçası olmuşlardır. Akıllı kartların güvenli yapısı, yaygın olarak kullanılmasındaki en büyük etkendir. Akıllı kartlar seri olarak haberleşilebilen aygıtalrdır. Bu projede, var olan bir akıllı kart okuyucusu geliştirilmiş ve bir işlemci ile beraber FPGA üzerinde gerçeklenmiştir. Akıllı kart okuyucusunun FPGA üzerinde gerçeklenmesi ileriye dönük projelerde akıllı kartların sisteme entegre edilmesini kolaylaştıracaktır.

Projenin gerçeklenmesinde Xilinx firmasının üretmiş olduğu Spartan 3E FPGA geliştirme kiti kullanılmıştır. İşlemci olarak, FPGA'in üzerinde Microblaze gerçeklenmiştir. Akıllı kart olarak ise, Advanced Card Systems firmasının çoklu uygulamalar için geliştirmiş olduğu ACOS6 akıllı kart kullanılmıştır. Akıllı kart okuyucu modülü yüksek hızlı tümleşik devreler için donanım tanımlama dili(VHDL) kullanılarak FPGA üzerinde gerçeklenmiştir. Akıllı kart gerçeklenen okuyucu modülü üzerinden işlemciye bağlanmıştır. Akıllı karttan veri okuma ve akıllı karta veri gönderme uygulamaları C programlama dili kullanılarak Xilinx Software Development Kit programında gerçekleştirilmiştir.

Projenin ilk bölümünde çalışmanın gerekliliği ve amacı anlatılmıştır. İkinci kısımında akıllı kartlar ve FPGA tasarımı hakkında genel bilgi verilmiştir. Üçüncü kısımda ise akıllı kart okuyucusu donanımsal ve yazılımsal araçlar kullanılarak FPGA üzerinde nasıl gerçeklendiği izah edilmiştir. Son olarak dördüncü bölümde çalışmadan elde edilen sonuçlara değinilmistir.

1. INTRODUCTION

In today's world, smart cards have reached an extremely wide range of usage. It is widely used from banks to healthcare institutions and even military areas. Smart cards are used in fields such as personal authentication, authentication, data storage, and application processing [7]. The secure structure of smart cards has made it so popular in our daily life.

Compared to conventional data transmission devices such as magnetic-stripe cards, smart cards offer enhanced security, convenience and economic benefits. In addition, smart card based systems are highly configurable to suit individual needs. [8] It can be used to store money and information electronically and can help to transfer it in a secure but portable medium. It acts like a mini-computer. Since smart cards are in use globally, the International Standard Organization has laid some standards so that they are universally compatible. ISO/IEC 7816 is a series of standards specifying integrated circuit cards and the use of such cards for interchange. [9]

Also smart card can be used at hardware circuit designs. That wide range of use of smart cards, give an opportunity to create secure embedded system designs. For these applications a stable and universal smart card controller is essential. A smart card controller can easily be added to the systems which will be designed.

1.1 Purpose of Thesis

A smart card controller should perform two task. These task are reading data which is received from smart card and write data to smart card. In this thesis, main goal is reading received data from smart card and writing these serial data as an 8-bit hexadecimal number. To archive this goal, a smart card controller will be designed on FPGA and designed controlled will be connected to microblaze which will be implemented on FPGA too.

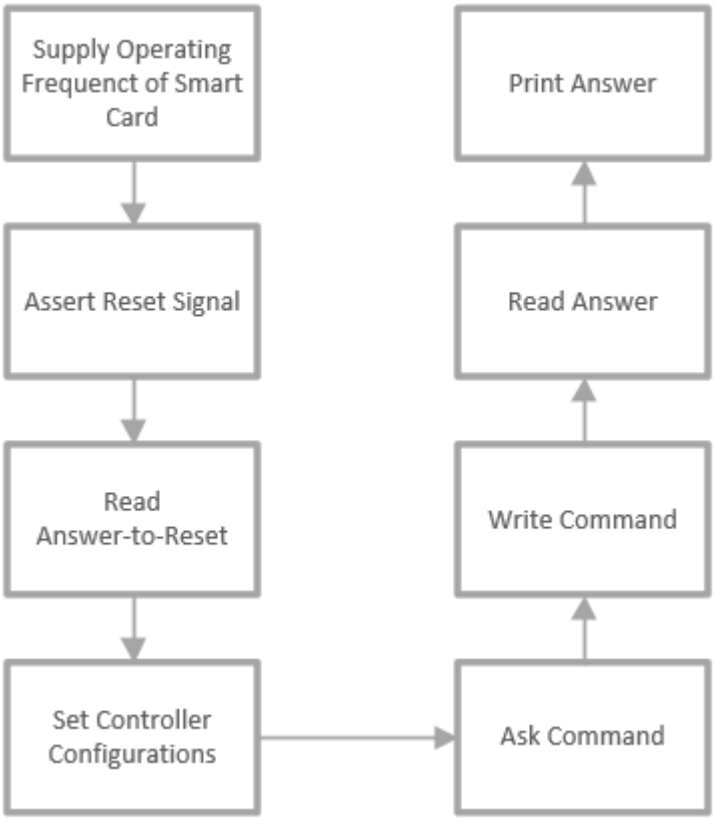

Figure 1.1 : Brief summary of the Smart Card Reader

User interface for smart card controller will be designed after microblaze implementation which is also consist a user interface of smart card reader. The complete flowchart of the project is shown at Figure 1.1.

2. SMART CARDS

2.1 Introduction to Smart Cards

Smart card is a credit card-sized plastic card with an embedded integrated circuits [2]. Smart cards are also known as integrated circuit card (ICC). Smart cards are widely used for various applications such as access control, authentication, security and medical. For different applications, different feature smart card is produced. Most common application areas with their memory and processing capacities are shown in Figure 2.1 [1].

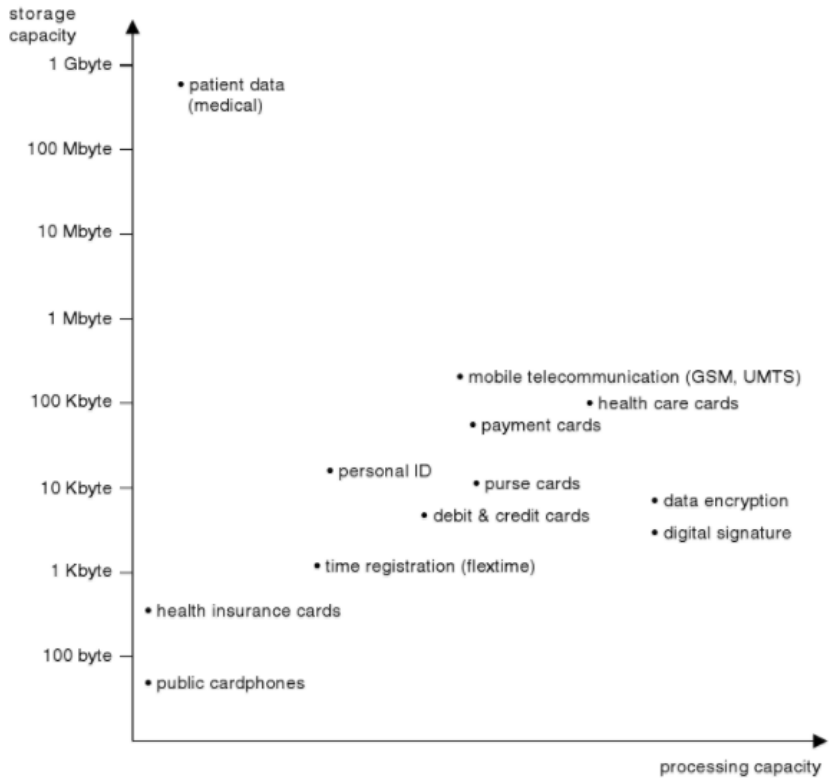

Figure 2.1 : Typical smart card application areas, and the required memory capacity and arithmetic processing capacity [1].

2.2 Physical Properties of Smart Cards

Physical properties of smart card are defined at ISO 7816-2. According to ISO, length and width of smart card should be 85,6 x 53,98 mm. Also the position of conductive pads and dimensions of the conductive pads are defined at that ISO. Minimum dimensions of the contacts should be 1.7 x 2 mm. The positions of conductive pads are shown in Figure 2.2

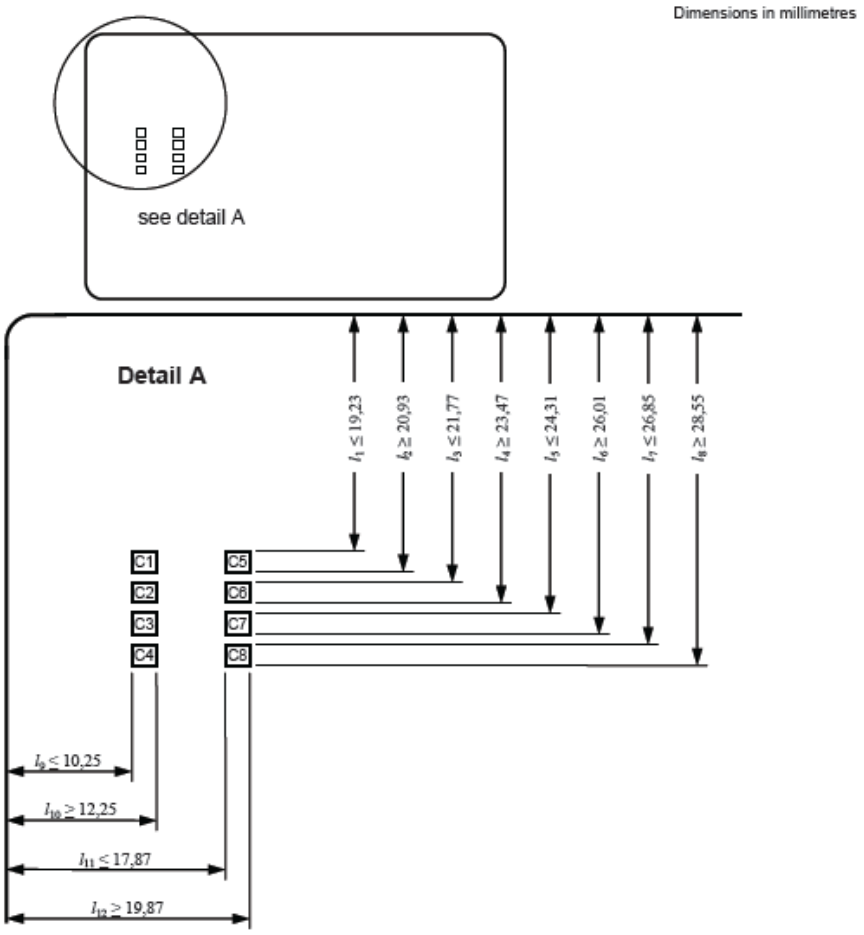

Figure 2.2 : Location of the contacts [2].

Smart cards are divided into contact and contactless according to the way they communicate with the outside world [1]. Contactless cards use an electromagnetic interface without physical contact; Contact cards provide direct physical contact with the card reader. Contacted cards are in 1 cm diameter, gold-plated, with a contact surface that accommodates eight contacts, while only 6 are actually used to communicate with the outside world. Explanation of these contact points are given at Table 2.1

Table 2.1 : Contact points of smart card.

Position	PIN Name	Description
C1	VCC	Supply Voltage(3.3V for Class B, 5V for Class A)
C2	RST	Reset signal
C3	CLK	Clock Signal
C4	RFU	Reserved for Future Applications
C5	GND	Ground Voltage
C6	VPP	Programming Voltage
C7	I/O	Serial Communication Port
C8	RFU	Reserved for Future Applications

2.3 Smart Card Types

Smart cards are exist in various form and specification. However, contact points of smart cards are generally similar. From aspect of dimension, smart cards formats: ID-1, ID-2, ID-3 and ID-000 [**?**]. These dimensions are stated at ISO 7810. Details of these cards are specified at Table 2.2.

Table 2.2 : Types of Smart Cards

Format	Dimension(mm)	Usage
ID-1	85.60 x 53.98	Most banking cars and ID cards
ID-2	105 x 74	French and other ID cars; Visas
ID-3	125 x 88	Passports
ID-000	25 x 15	SIM cards

Contactless and magnetic stripe smart cards are the other type of smart card. Smart cards with magnetic stripe is generally used for simple applications which only need a few memories. On the other hand, operational logic of contactless smart card is similar with contact smart card. At that type of smart card, operating voltage is supplied with a RF antenna and also the communication is provide with same antenna. The architecture of contactless smart cards is given at Figure 2.3

The ISO 7816-1 family of smart card standards is based the ID-1 card format, which is used for various applications. Because of wide area of usage smart cards have different capabilities and different features. Smart cards can be divided into 3 groups according to their processing ability:

- Memory chip

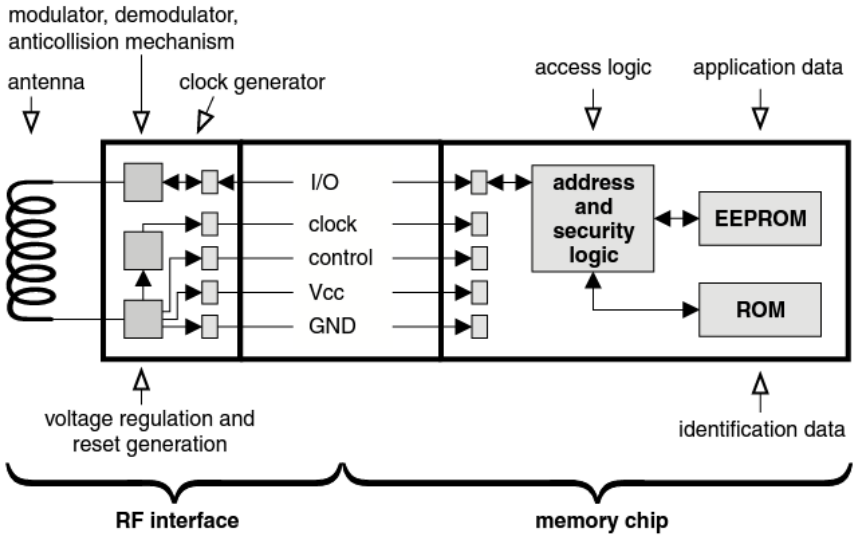

Figure 2.3 : Typical architecture of a memory card with security logic and a contactless interface [1].

- Microprocessor chip
- Cryptography chip

Smart cards with memory chips are the simplest type of smart cards. These type of smart cards are only keep the on chip. This type of card is not secured by any transaction specific to the card, but by the fact that the data to be stored on the card is encrypted.

In processor-based smart cards, access to the memory of the smart card is made entirely through the microprocessor. Processor type smart cards are divided into with and without security logic [1]. With security logic stored data can be protected against unauthorized access and manipulation by using personal identification number (PIN). General scheme of these type of smart cards is shown at Figure 2.4

Cryptography cards are the most advanced type of smart cards. These type of smart cards have an advanced microprocessor for cryptography applications where hardware precautions are needed to perform complex cryptographic functions at desired rates.

2.4 Smart Card Microcontrollers

An embedded microcontroller is the key component of the smart cards. That microcontroller is placed under the contacts of the smart card. Different types of microcontrollers are designed for various applications. Some components are

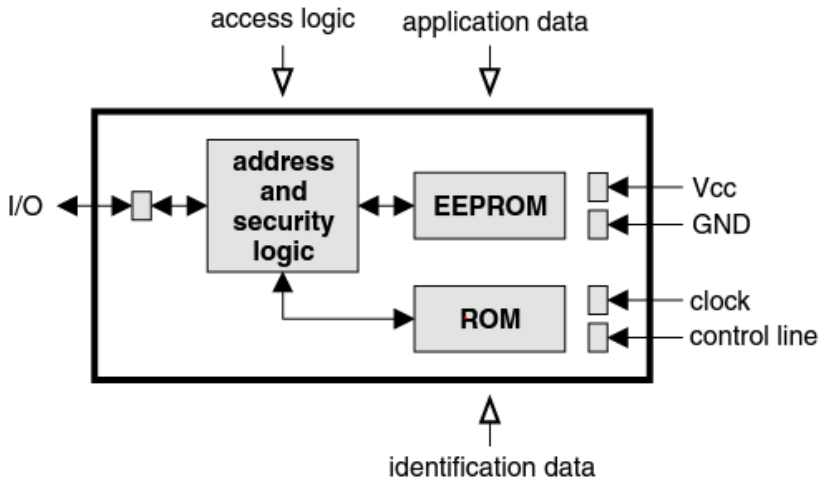

Figure 2.4 : Typical architecture of a contact memory card with security logic [1].

common for all smart cards while other components varied for applications. The major functional components of a typical smart card microcontroller are the Central Processing Unit (CPU), the address and data buses, and the various types of memory (Random Access Memory (RAM), Read Only Memory (ROM), and Electronically Erasable Programmable Read-Only Memory (EEPROM) or flash) [1].

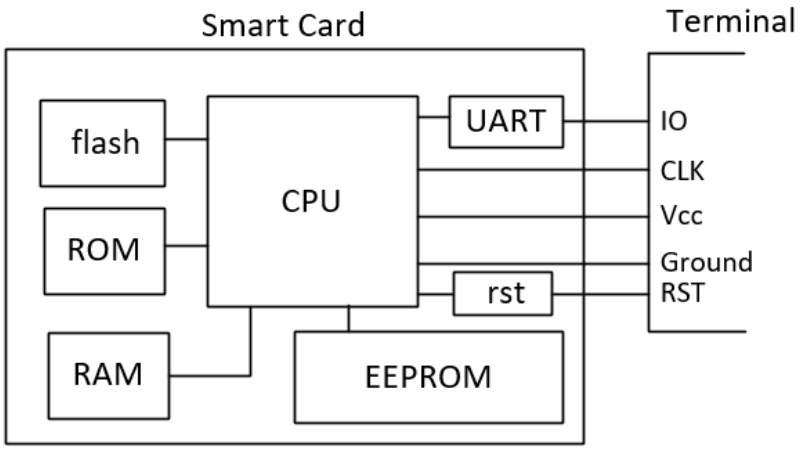

Figure 2.5 : The usual functional units of a smart card microcontroller.

Central Processing Unit(CPU) is the main part which is makes smart card "smart". Operating method of smart card CPU is same with typical CPUs. Size of the smart card CPU is the main difference. When first smart card issued in 1950, Smart card CPUs were 8-bit processors but now they are generally 16-bit or 32-bit [1]. Smart card CPU communicates with other peripherals and also CPU is the window to the world. In general, the serial interface is only an address that is connected to the I / O port and that can be accessed by the CPU.

Table 2.3 : Smart Card Memory Types

Component	Memory Type	Area on Chip
RAM	Volatile	20 %
ROM	non-Volatile	10 %
EEPROM		45 %
flash		15 %

After CPU, most important components are memories in different types. Smart cards are generally complete computers in small size. Memories are categorized as volatile memory and nonvolatile memory. ROM, EEPROM and flash are the nonvolatile types of memory. ROM is programmed during manufacturing process for operating system, Look Up Tables (LUT) and card Identifier (ID). EEPROM is take most of the space of smart cards. The physical size of the EEPROM can be as large as half of the total area of smart card chip [10]. RAM is the working storage of the CPU.

2.5 Smart Card Communications

Communication with the smart card is provided over a single 1-bit serial port [1]. Application Protocol Data Units (APDU) are used to communicate with smart cards. That protocol is defined at ISO/IEC 7816-4. For that communications 2 types of APDUs exist: command APDUs and response APDUs. A command APDU is sent by the reader to the card which contains a mandatory 4-byte header (CLA, INS, P1, P2) and from 0 to 65 535 bytes of data. A response APDU is sent by the card to the reader which contains from 0 to 65 536 bytes of data, and 2 mandatory status bytes (SW1, SW2).

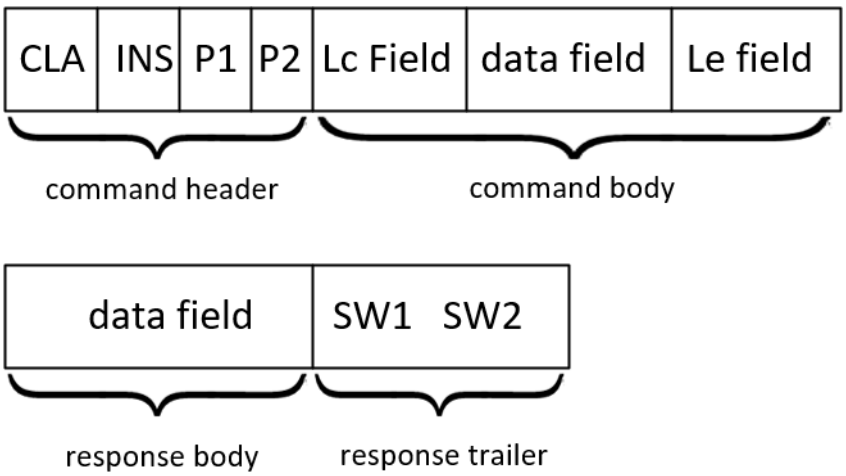

Figure 2.6 : Command and Response APDU structures.

According to ISO/IEC 7816-3 Electrical interface and transmission protocols, smart card reader should perform following functions:

- Turn on/off operating voltage of smart card
- Control the reset signal to reset smart card
- Supply the operating clock signal to smart card
- Read data from smart card
- Send data to smart card

The character transmission of a smart card communication starts according to ISO/IEC 7816-3 as given as Figure 2.7. The transmission of a single character requires an overhead of several bits as follows,

- Start bit
- 8-bit data
- Parity bit
- Guard time between two byte

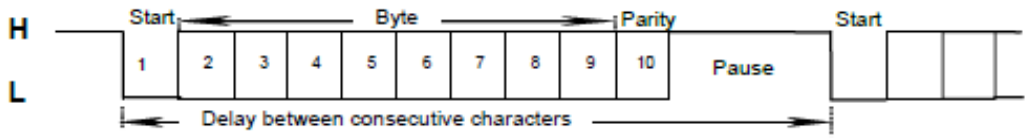

Figure 2.7 : Character frame [3]

As a first step, smart card reader should turn on the supply voltage (V_{cc}) and clock signal. Smart cards have three different operating class [11]. This class specifies the required supply voltage and maximum current drawn from V_{CC} for the smart card. Smart card reader should supply required supply voltage and current. These class are shown at Table 2.4 After that, handshake should start between smart card and reader. To start handshake reset should driven be low to high. Thus, state machine of smart card reset itself and go back to initial state. As an initial task, smart card is sent Answer to Reset (ATR) which is contains operating conditions.

Table 2.4 : Classes of Operation

Conditions	**Maximum V_{CC}**	**Minimum V_{CC}**	**Maximum I_{CC}**
Class A	4.50 V	3.50 V	60mA
Class B	2.70 V	3.30 V	50mA
Class C	1.62 V	1.98 V	30mA

ATR is the principal command which is sent by smart card. ATR contains different numbers of bytes. Maximum length of ATR can be 33 bytes which contains various parameters. It is always sent with a divisor value (clock rate conversion factor) of 372 in compliance with the ISO/IEC 7816-3 standard [1]. The details of ATR shown at Table 2.5.

Table 2.5 : Default Configuration of ATR

Name	**Defines**	**Encodes**
TS	The Initial Character(mandatory)	mandatory
T0	The Format Character	mandatory
TA_i, TB_i, TC_i, TD_i	The Interface Characters	optional
T1... TK	The Historical Characters (max. 15 character)	optional
TCK	Allow detection of accidental transmission error	conditional

TS defines the convention type of smart card. It can be either 0x3B which mean direct convention or 0x3F which mean inverse convention. Most of the smart cards work on direct convention mode. T0 specifies the communication method. Most significant 4 bit of T0 defines existence of interface characters, least significant 4 bit defines number of historical bits. The maximum number of historical bits can be 15.

To read data from smart card and send data to smart card, TA_1 byte of ATR is critical. The nominal duration of one moment on the electrical circuit I/O is named "elementary time unit" and denoted *etu* and the value is derived from TA_1. Where F_i is the clock rate conversion integer and D_i the baud rate adjustment integer. To calculate value of the *etu* equation 2.1 is used [3].

$$1etu = \frac{F}{D} \times \frac{1}{f} \tag{2.1}$$

For communicate with smart card ISO/IEC T = 0 or T = 1 protocols can be used. The information of communication type is provided at ATR. Transmission protocol is defined at TD_1. According to TD_1 also T= 2 .... 15 is available but reserved. T = 0 provides asynchronous half duplex character transmission while T = 1 for

asynchronous half duplex block transmission. Most of the smart cards operating under T = 0 protocol.

After reading ATR smart card starts to communicate with specifications which is stated at ATR. Also if the smart card reader wants to modify one or more of these parameters, it must perform a Protocol Parameter Selection (PPS) process in accordance with ISO/IEC 7816-3 before the transmission protocol is actually used. The PPS request must be sent immediately after the ATR.

3. IMPLEMENTATION

3.1 Used Equipment

3.1.1 Field Programmable Gate Array

Field Programmable Gate Array (FPGA) is an integrated logic device that can realize digital circuits via hardware description languages such as Verilog and Very-high-speed-integrated-circuit Hardware Description Language (VHDL). Besides, Computer Aided Design (CAD) software can be used for schematic FPGA design. Basic FPGA block consists of three basic capabilities: input/output (I/O) interfaces, basic building blocks, and interconnections. The logical structure of an FPGA device is shown in figure 3.1. FPGAs are high-speed devices because the system is located in a small area, which reduces the delay caused by the interconnections. Also, parallel processing capability speeds up the FPGAs.

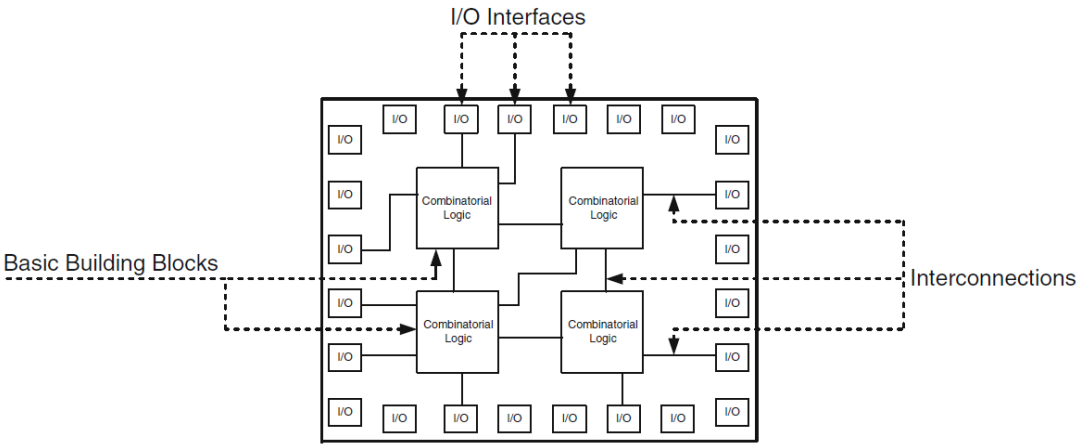

Figure 3.1 : Generic FPGA architecture [4].

Application Specific Integrated Circuit (ASIC) is an integrated circuit (IC) customized for a particular use which cause long design cycle and high design cost [12]. Low unit costs and full custom capability is the main benefits of ASIC design. Compared to ASIC, FPGA gives engineers the opportunity to redesign the digital circuit that is implemented on the device. This is rather significant advantage, because it decreases

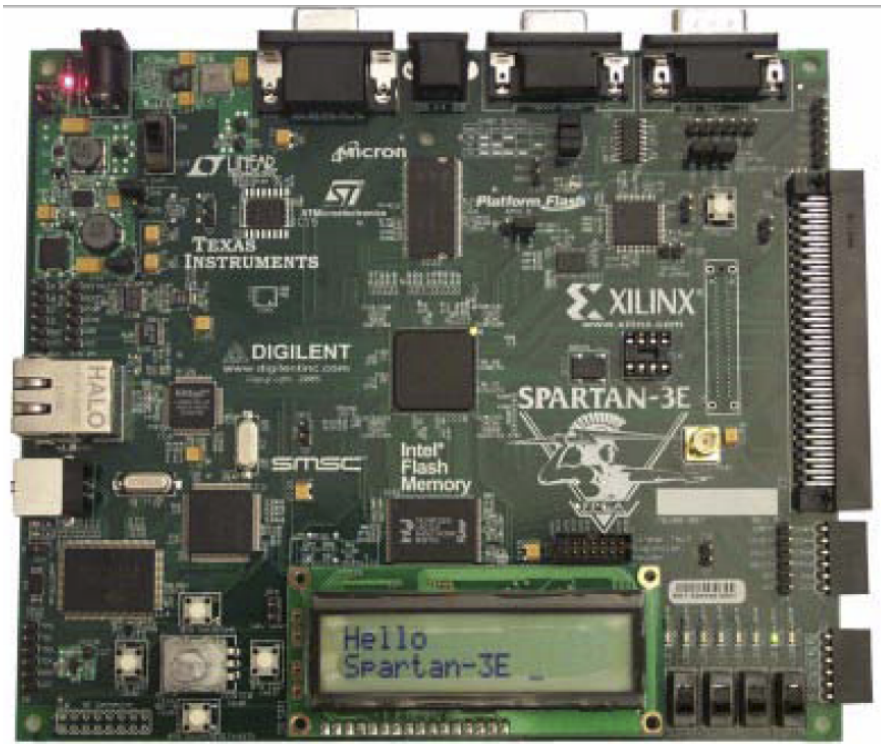

Figure 3.2 : Spartan 3E Starter Board.

the digital design costs remarkably. FPGA give an opportunity to test a circuit without manually creating a circuit. On the other hand, soft microprocessor systems can be implemented on FPGA device as well.

3.1.1.1 Spartan 3E Starter Board

The project aims to implement a smart card controller on an FPGA. Spartan 3E is a type of FPGA and Spartan 3E starter kit which is manufactured by Xilinx company has been used in this project [13]. A Spartan 3E starter board is shown at Figure 3.2.

Xilinx Spartan 3E board has three main packages;

- Spartan-3E FPGA (XC3S500E-4FG320C)
- CoolRunner^{TM}-II CPLD (XC2C64A-5VQ44C)
- Platform Flash (XCF04S-VO20C)

3.1.2 Serial-to-USB Communication Cable

In serial communication, as a recommended standard (RS) RS-232 is used for transmission of data. As shown in Figure 3.3, the Spartan-3E Starter Kit board has two

RS-232 serial ports: a female Data Circuit-terminating Equipment (DCE) connector and a male Data Terminal Equipment (DTE) connector. DCE-style port is not available at most of the computers in today's technology. To communicate with FPGA board connection is provided via RS232 Serial to USB cable. To make conversion USB to serial chip HL-340 is used.

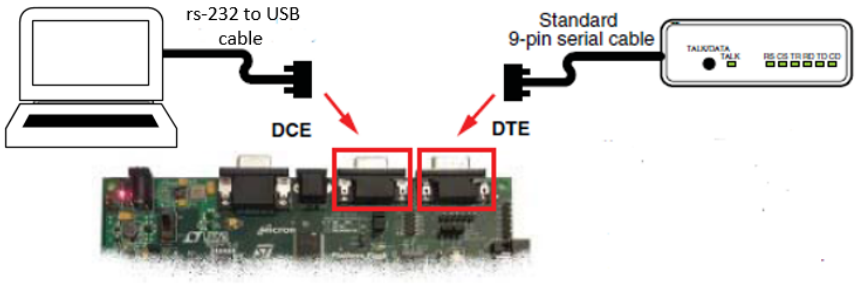

Figure 3.3 : RS-232 Serial Ports

3.1.3 ACOS6 Smart Card

The ACOS6 Multi-application & Purse Card (MAP Card) is a type of smart card which is manufactured by Advanced Card Systems (ACS). ACOS6 smart card is compatible with ISO 7816 parts 1, 2, 3, 4. ACOS6 smart card has a 64K EEPROM memory for smart card applications and storage. In ACOS6, two of the contact pads are not exist but It is not creating any issue for smart card reader implementation project. ACOS6 smart card works on T=0 mode at 9600 baud rate as a default.

3.2 Xilinx Software Environment

Xilinx is a company whom not only manufacture FPGAs also supports several software products which can be used to program FPGA [14]. Xilinx tools also can be used to configure and control its own unique microcontroller Microblaze that takes part inside FPGA. The interface software that are used in the thesis project are; Xilinx Integrated Synthesis Environment (ISE), Xilinx Embedded Development Kit (EDK) and Xilinx Software Development Kit (SDK)

3.2.1 Xilinx Integrated Synthesis Environment

Integrated Synthesis Environment (ISE) is a software program which is developed for synthesis and analysis of HDL designs [15]. Synthesize makes developer enable to perform timing analysis, view RTL schematics, simulate a design's and configure the

target device. Implementation steps of a generic digital circuit can be seen in the figure 3.4.

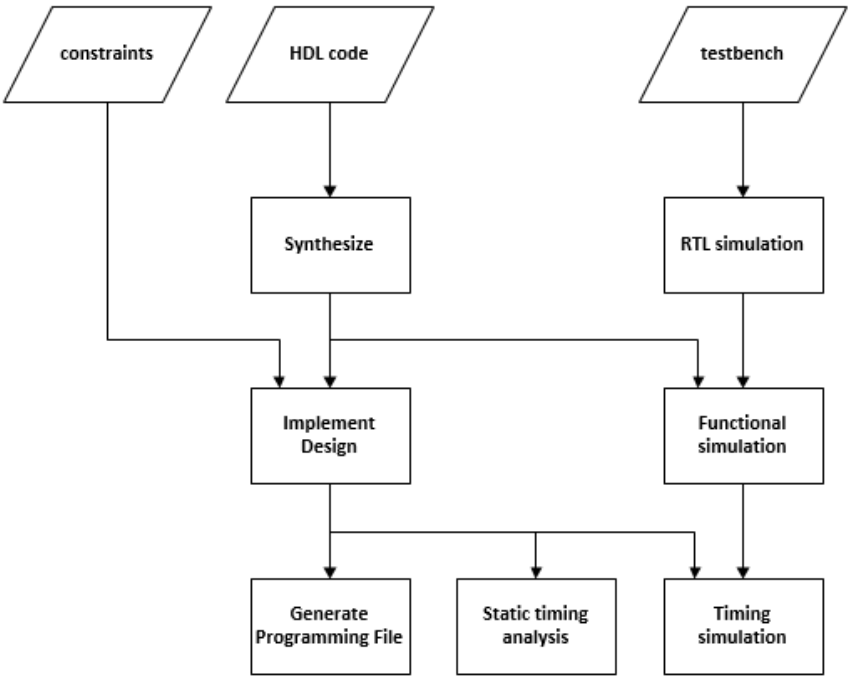

Figure 3.4 : Flow chart of design development [5]

Behavioral testing may be performed with ISIM which is a logic simulator. Test programs must also be written in Verilog or VHDL [5]. ISE Simulator (ISIM) uses behavioral verification, to verify logical and timing issues. Simulation which is performed at ISIM is just a test of HDL code.

3.2.2 Microblaze Processor

Microblaze is a reduced instruction set computing (RISC) based soft-core microprocessor which is designed by Xilinx for Xilinx FPGAs [6]. Soft-core processors can completely be implemented using logic synthesis. The functional block diagram of the Microblaze core is shown at Figure 3.5. Microblaze is 32-bit, configurable microprocessor from many aspects. Cache size, pipeline depth (3-stage or 5-stage), embedded peripherals, memory management unit, and bus-interfaces can be customized.

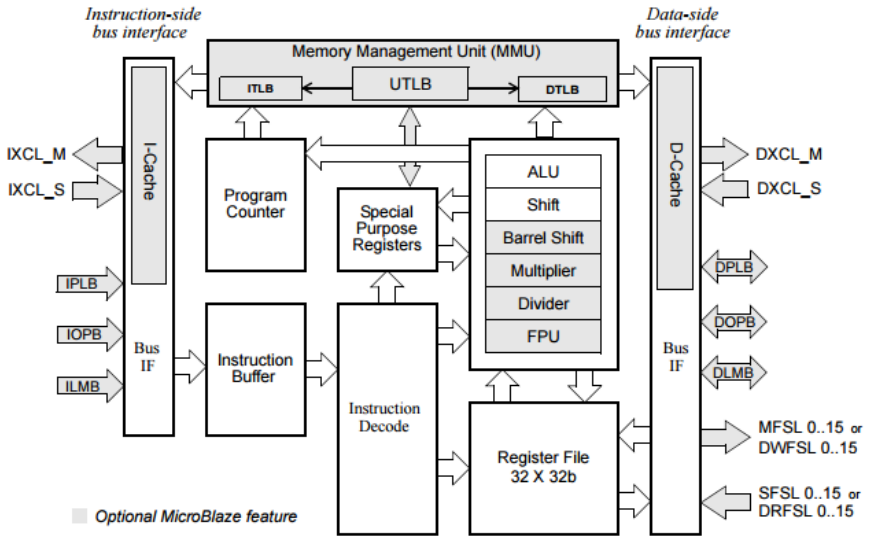

Figure 3.5 : MicroBlaze Core Block Diagram [6]

3.2.3 Xilinx Embedded Development Kit

The Embedded Development Kit (EDK) is an integrated development environment for designing embedded processing systems [16]. Designers use XPS (Xilinx Platform Studio) to configure and build the hardware specification of their embedded system. EDK lets the user select from various optional features, which is useful by means of creating microprocessor based designs without unnecessary parts.

3.2.4 Xilinx Software Development Kit

The Xilinx Software Development Kit (SDK) is a complete embedded software development environment for Xilinx Microblaze processors [17]. SDK is built on Eclipse which is an integrated development environment (IDE) used in computer programming.

3.3 Implementation of the Smart Card Controller

A smart card controller is retrieved from Xilinx CoolRunner-II Smart Card Reader application but the controller is designed for a single smart card and single operation [18]. Most of the units are changed except state control unit and data shift register. These two part is also improved. Entity view of smart card controller is given in Figure 3.6.

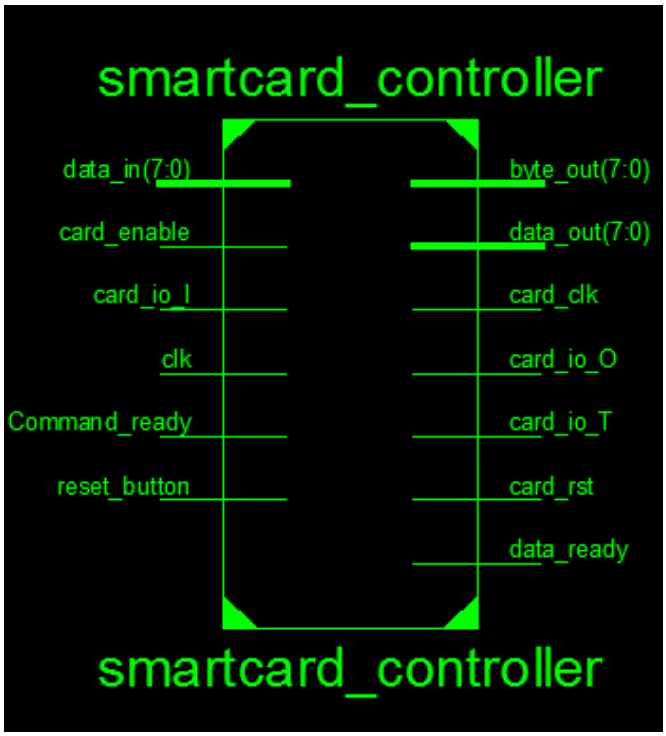

Figure 3.6 : RTL Schematic of Smart Card Controller

As it is seen from Figure 3.6, the module has a card enable and a reset signal as input. Card enable signal is connected to hardware switch which is detect existence of smart card. If a smart card is inserted to smart card slot, card enable switch is set on, otherwise set off. Also the reset signal is connected to microblaze and it is controlled by software.

3.3.1 Block Diagram of the Designed System

Smart card controller project aimed to communicate with a smart card through a Microblaze processor. In Figure 3.7 the general block diagram of the smart card, smart card controller and microblaze is given. Smart card controller supply clock signal, reset signal and communicate with smart card through inout port. Smart card controller receive data from serial communication port and store input data up to 8-bit. When the data valid, smart card controller convey the data to microblaze. Also smart card controller receives 8-bit data from microblaze and transmit the data to smart card bit by bit.

The retrieved smart card controller is developed for universal usage. The retrieved controller was operating asynchronous. Therefore, smart card controller was edited entirely. First communication between smart card and smart card controller start after

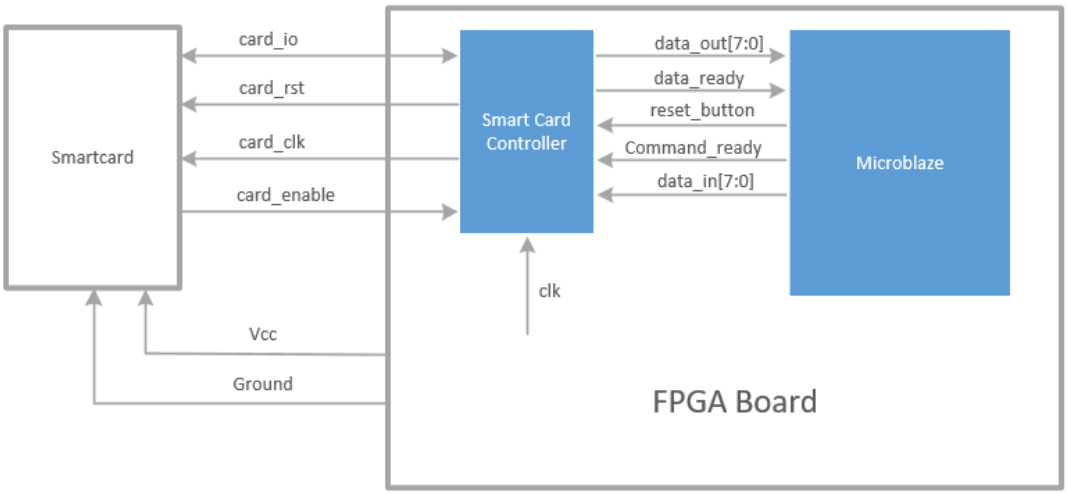

Figure 3.7 : General block diagram

reset signal which should be driven low to high. Baud rate counter is started to count with first change on inout port. Block diagram and device utilization of controller given Figure 3.8 and 3.9. Also, implemented modules of smart card controlled are explained below:

Baud Rate Counter: As a standard during the ATR, baud rate counter count to 372 for each data bit. After ATR transmission baud rate starts to count the value stated in ATR. The ISO 7816-3 standard aligns with the use of two widely used external clock frequencies, 3.579545 MHz and 4.91522 MHz in order to produce a 9600 bit per second (not exact but within tolerance) serial communication speed. When 372 clock cycles have been counted, its value is reset. [9].

Clock Divider: Clock available from FPGA board is 50MHz. Clock divider divides this frequency by 14 to obtain frequency of 3.57 MHz

Bit Counter: In each cycle of baud rate counter, bit counter increases. The purpose of bit counter is to count all the 12 bits in one-character frame. When 12 bits have been received, bit counter is reset and wait for next byte.

Bytecounter: Since a character frame consists of 12 etu, byte counter counts number of characters (or bytes) received.

Shift Register: It is a Serial In Parallel Out (SIPO) shift register. It receives serial data from card via card I/O pin (C7) and saves it parallel in a data buffer. Data is sampled when baud rate counter were at the middle of the counting. When 12 etu are

completed, i.e., when bit counter counts 11, bit2 to bit9 of the received character frame is the byte which gets saved in *data_out* register.

Communication Mode Selection: The first implementation was conversion selector. After first byte of ATR read conversion of communication selected refer to this byte. Also the baud rate of communication and T=0 or T=1 mode selection made, refer to the TA_1 byte of ATR. Also, if an unexpected ATR occurs, an error signal asserted to output.

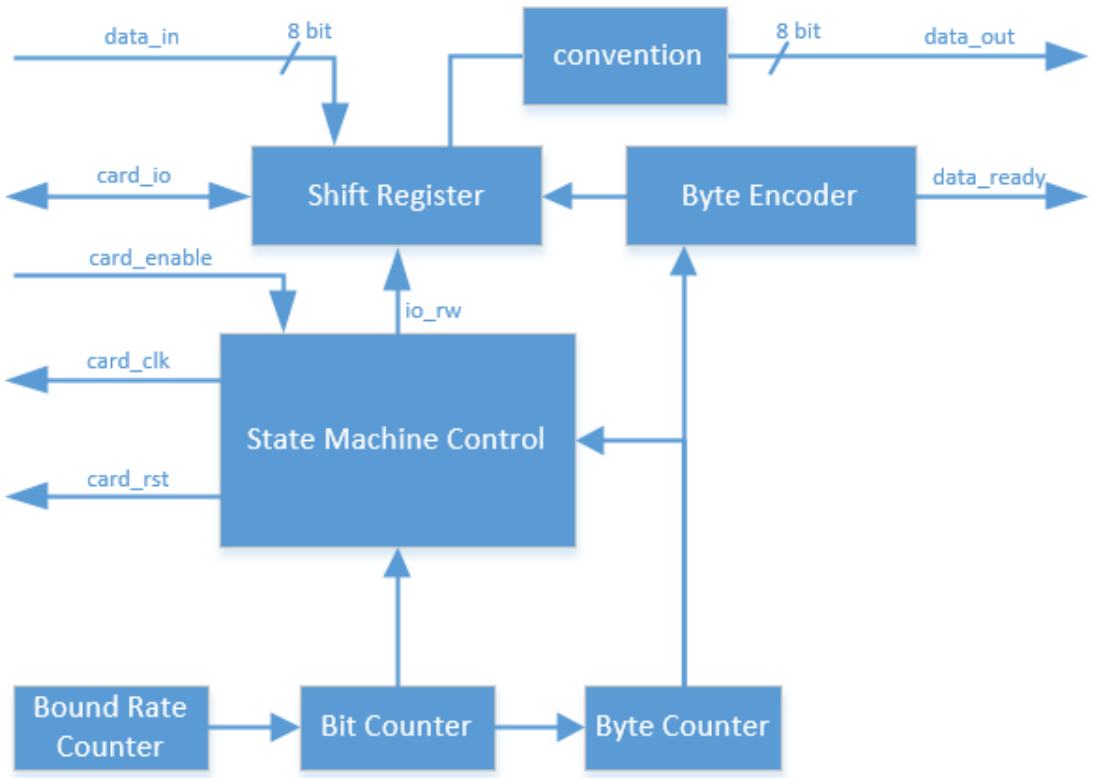

Figure 3.8 : Smart Card Controller Block Diagram

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	81	9,312	1%	
Number of 4 input LUTs	91	9,312	1%	
Number of occupied Slices	77	4,656	1%	
Number of Slices containing only related logic	77	77	100%	
Number of Slices containing unrelated logic	0	77	0%	
Total Number of 4 input LUTs	139	9,312	1%	
Number used as logic	91			
Number used as a route-thru	48			
Number of bonded IOBs	34	232	14%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	2.97			

Figure 3.9 : Device Utilization Summary of Smart Card Controller

3.3.2 State Machine

State machine of smart card controller consist five states. These states are shown in Figure 3.10. *IDLE* state is the initial state of the state machine and this state is controlled by reset signal. Second state is *WaitForData* state. In that state smart card controller wait for a data from smart card or write command from microblaze. If smart card starts to control serial port, smart card controller jump to *ReadData* state. Or, if the write command is ready, it goes to the *WriteCommand* state. After read or write state, smart controller process read or written data at *ProcessData* state and jump back to *WaitForData* state when *ProcessData* state complete.

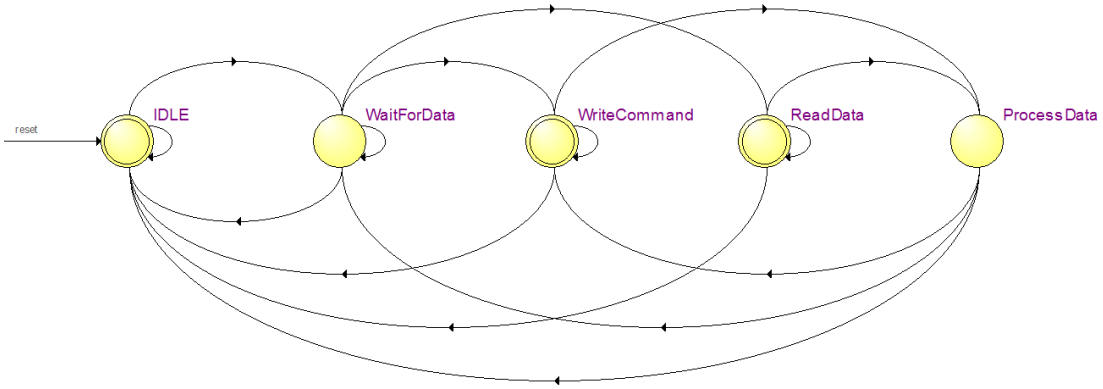

Figure 3.10 : Smart card controller state machine

3.3.3 Simulation of Smart Card Controller

Behavioral testing of the smart card controller was performed with ISim which is a simulation software of Xilinx. At the test-bench, a smart card model is implemented and external clock signal generated. In the Figure 3.11, a small part of ATR command is obtained.

As its seen from Figure 3.11, when each byte read from smart card a one-byte *data_out* byte signal obtained as an output signal. Also when *data_out* signal valid, *data_ready* signal occurs and both of them stay for 1 *etu*. In this simulation, duration of the *data_ready* signal is same as *data_out* signal. This is critical because reading from smart card module is going to happen when *data_ready* signal is valid.

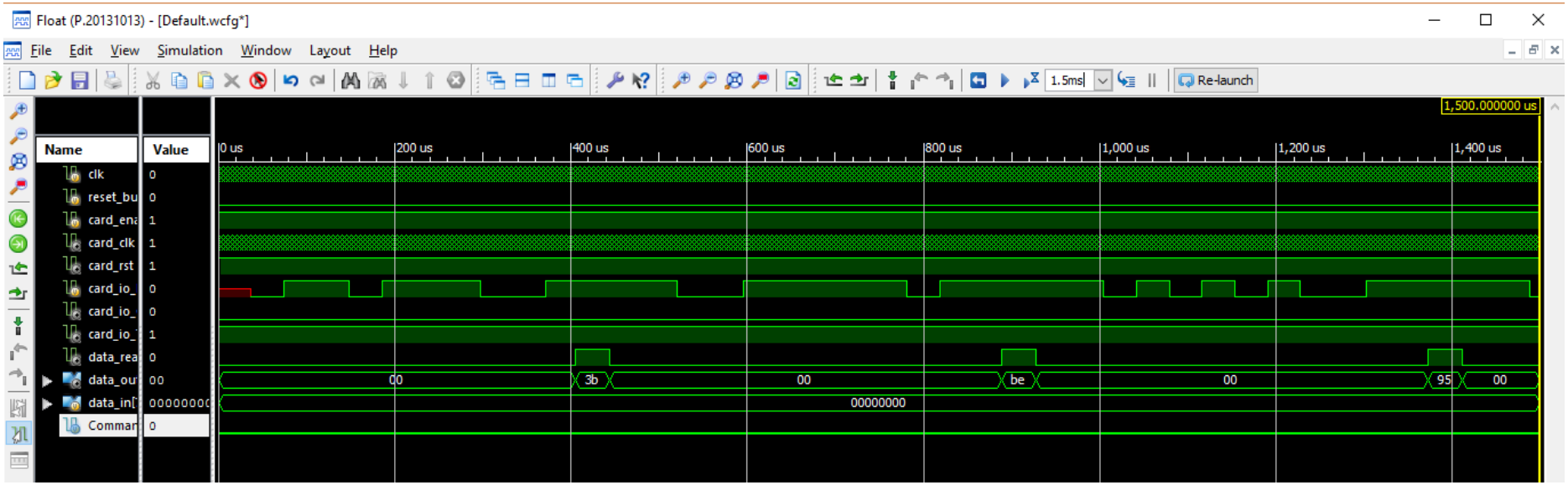

Figure 3.11 : Smart card controller simulation

3.3.4 Test of Smart card

Simulation of the smart card module was a software test. Testing smart card controller on FPGA was the next testing step. For this test smart card is inserted a smart card slot and that slot connected to FPGA board. Smart card slot and its FPGA connection shown at Figure 3.12. The aim of the test setup is observing the ATR which is sent by smart card.

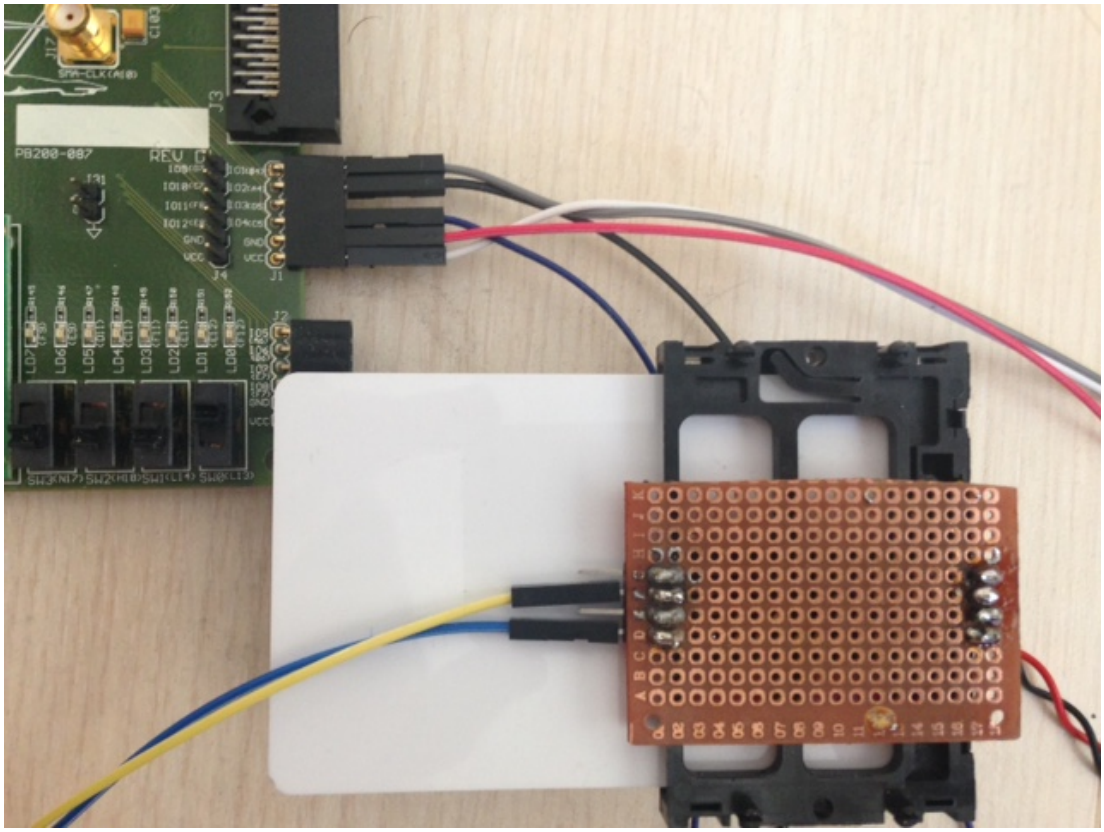

Figure 3.12 : Smart card FPGA connection

Before create programming file, User Consist File (UCF) is created on ISE which is given in Figure 3.13. Reset and card enable signals are connected to switches on FPGA board. Then, the signal to smart card is connected to I/O ports of FPGA. Lastly, *data_out* signal is connected to LEDs to observe ATR in binary format. To make changes on LEDs visible, clock frequency of smart card is decreased.

After UCF created, design is implemented virtually on FPGA and refer to this implementation bit-stream file FPGA programming is created on Xilinx ISE. To program FPGA ISE Impact software is used which is shown in Figure 3.14. Card

```
NET card_clk LOC = "C5" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
Net card_io LOC = "A4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
Net card_rst LOC = "B4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
NET clk LOC = C9 | IOSTANDARD = LVCMOS33 ;
NET "card_enable" LOC = L13;
NET "reset_button" LOC = D18;
##Spartan 3E Starter Kit Leds
NET "byte_out[7]" LOC = F9 |  IOSTANDARD = LVCMOS33;
NET "byte_out[6]" LOC = E9 |  IOSTANDARD = LVCMOS33;
NET "byte_out[5]" LOC = D11 |  IOSTANDARD = LVCMOS33;
NET "byte_out[4]" LOC = C11 |  IOSTANDARD = LVCMOS33;
NET "byte_out[3]" LOC = F11 |  IOSTANDARD = LVCMOS33;
NET "byte_out[2]" LOC = E11 |  IOSTANDARD = LVCMOS33;
NET "byte_out[1]" LOC = E12 |  IOSTANDARD = LVCMOS33;
NET "byte_out[0]" LOC = F12 |  IOSTANDARD = LVCMOS33;
```

Figure 3.13 : UCF file of smart card controller

Table 3.1 : Default configuration and historical bytes of the ATR

Parameter	TS	T0	TA1	TB1	TD1	T1	T2	T3	T4	T5
ATR	3B	BE	95	00	00	41	03	00	00	00

Parameter	T6	T7	T8	T9	T10	T11	T12	T13	T14
ATR	00	00	00	00	00	00	02	90	00

enable switch pulled high and reset switch pulled low and FPGA was programmed with success.

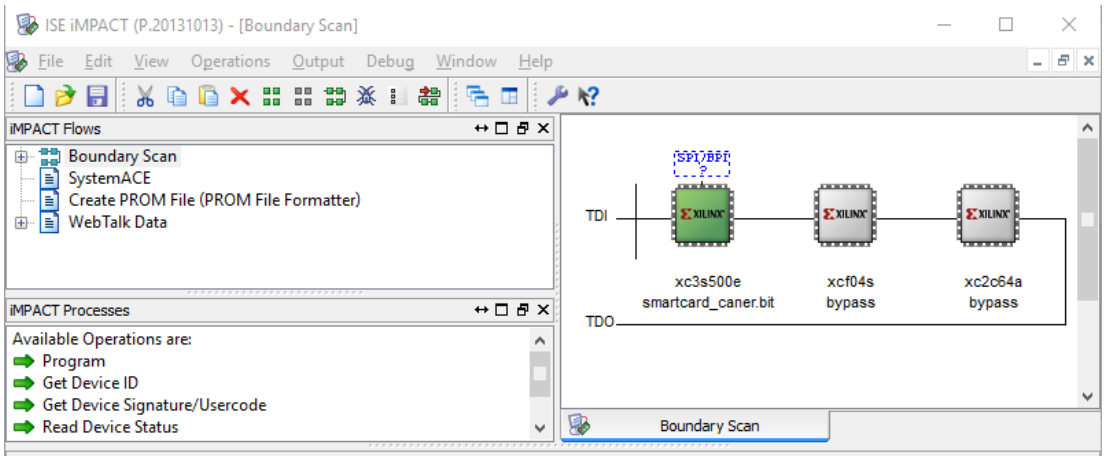

Figure 3.14 : Programming FPGA via ISE Impact

As a last step, reset switch pulled low to high and LEDs observed. ATR sequence of ACOS6 smart card tried to be observed according as manual of ACOS6 which is given in Table 3.1 [19]. As a result, because of the 12 of zero bytes only 7 non-zero byte of ATR sequence was observed on LEDs. Also one more experiment was executed to crosscheck. In the second implementation, bytecounter signal is assigned to LEDs and the right length of ATR, nineteen, was observed. First observed byte of ATR, which is 3B, and bytecounter signal is given in Figure 3.15.

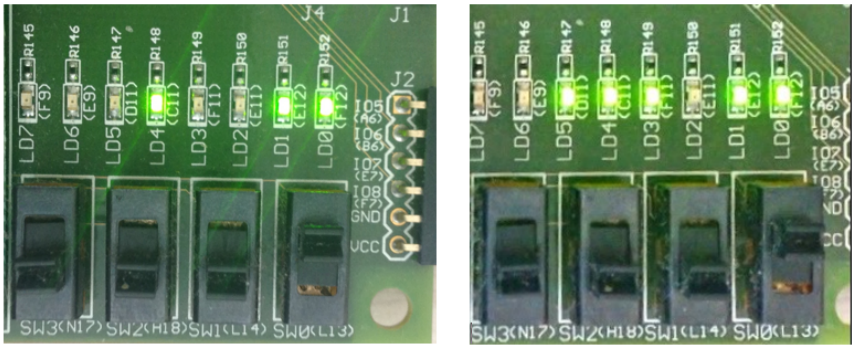

Figure 3.15 : First byte(3B) and Bytecounter(19) after ATR reading.

3.4 Microblaze Implementation

Microblaze is used to control smart card controller. Programming Microblaze is made on Xilinx SDK. As a first step, a new XPS project is created. While creating XPS project for Spartan 3E FPGAs Processor Local Bus (PLB) is selected. For newer FPGAs Advanced eXtensible Interface (AXI) should be selected. In FPGA selection step, Spartan 3E Starter board is selected. Then, except RS-232 peripheral other peripheral is removed. After selection of cache memory microblaze project is created.

```
// --USER logic implementation added here
smartcard_caner sc (
  .reset_button(slv_reg9[0]),      //                              1 bit
  .card_enable(card_enable),       //                              1 bit
  .clk(Bus2IP_Clk),                //                              clk
  .card_clk(card_clk),             //port olarak eklendi     -      1 bit
  .card_rst(card_rst),             //port olarak eklendi     -      1 bit
  .card_io_I (card_io_I),          //port olarak eklendi     -      1 bit
  .card_io_T (card_io_T),          //port olarak eklendi     -      1 bit
  .card_io_O (card_io_O),          //port olarak eklendi     -      1 bit
  .data_out(data_out),             //slv_reg[0:8]'e eklendi  -      8 bit
  .data_in(slv_reg9[2:9]),         //                               8 bit
  .Command_ready(slv_reg9[1]),     //                              1 bit
  .data_ready(data_ready),         //wire olarak birakildi         1 bit
  .byte_out    (bytecounter)       //case select                   8 bit
  );
```

Figure 3.16 : Modified user_logic File

3.4.1 Hardware Implementation

To create custom Intellectual Property (IP) on EDK environment some step should perform. This steps are:

1. At the hardware tab, new peripheral is created. While creating new peripheral, number of register is chosen. Also, user logic file is created in Verilog and template

driver files is created to implement software interface. As a peripheral name *sc_controller* is chosen.

2. Smart card module is instantiated at created *user_logic* file. Also, all ports of smart card controller are assigned. Microblaze ports are connected to data buses and smart card ports assigned as external port.

3. Some changes at *sc_controller.vhd* file also should perform because of the external ports. In *.vhd* file all external ports are assigned and mapped.

4. After changes at user logic files, smart card controlled is imported to design. The changed files and smart card controller *vhd* file is added to created peripheral.

5. Smart card controller IP core is added to the system from the IP Catalog. Then, In the "Bus Interfaces" tab peripheral is connected to PLB. In Figure 3.17 *sc_controller* is IP core and connected to PLB as shown. Also, in the ports tab all ports of smart card controller were made external.

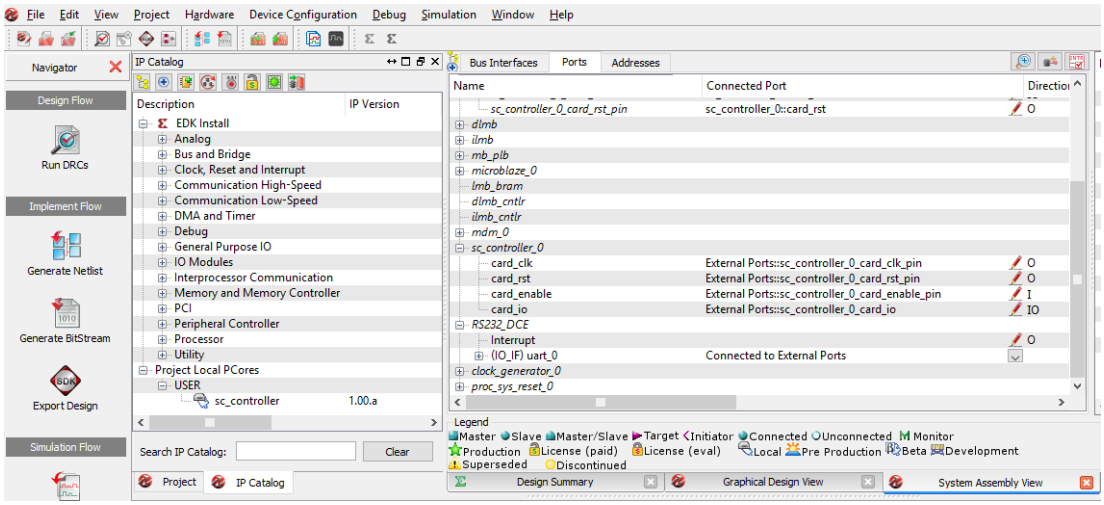

Figure 3.17 : EDK Environment with Custom *sc_controller_ip*

6. Microprocessor Peripheral Definition (MPD) file contains all of the available ports and hardware parameters for a peripheral [20]. Due to external inout port, MPD file should be changed. In MPD file, *card_io_I*, *card_io_O* and *card_io_T* ports are commented. Instead of this, bidirectional signal *card_io* is inserted. In Figure 3.18 changed MPD file is shown. Also top view of implemented *sc_controller_ip* show at Figure 3.19

```
BEGIN sc_controller

PORT card_clk = "", DIR = O
PORT card_rst = "", DIR = O
##PORT card_io_T = "", DIR = O
##PORT card_io_O = "", DIR = O
##PORT card_io_I = "", DIR = I
PORT card_io = "", TRI_O = card_io_O, TRI_T = card_io_T, DIR = IO, TRI_I = card_io_I, THREE_STATE = TRUE
PORT card_enable = "", DIR = I
```

Figure 3.18 : MPD File of Smart Card Controller IP

Figure 3.19 : Schematic view of *sc_controller_ip*

7. UCF file of project was edited for connecting IP pin to the FPGA physical pin. Port assignments of *sc_controller* were made for external ports and Universal Asynchronous Receiver-Transmitter (UART) port RS-232. Also input clock signal is connected to FPGAs clock pin. UCF file is show at Figure 3.20.

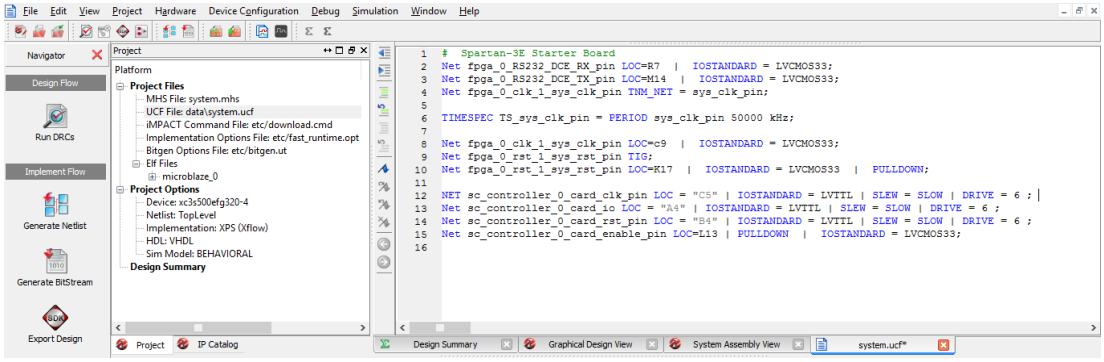

Figure 3.20 : UCF File of Smart Card Controller IP

8. As a last step of implementation, Design Rule Checking (DRC) is run, netlist and bit-stream is generated and project is exported to SDK environment. To start export process "Export & Launch SDK" is selected as shown at Figure 3.21.

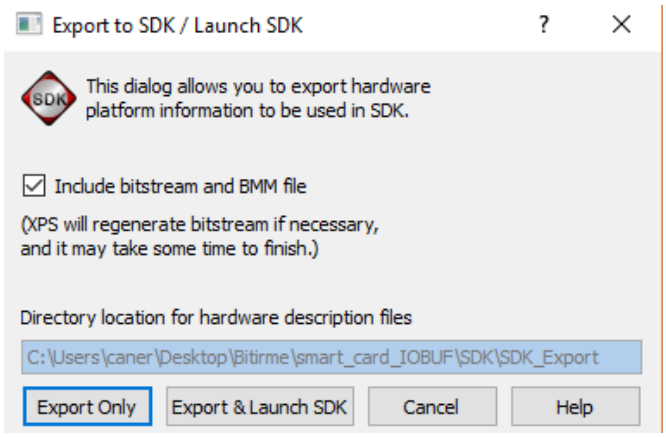

Figure 3.21 : Export Project & Launch SDK

The hardware design is imported to the SDK environment in order to implement the software part.

3.4.2 Software Implementation

SDK is used for software applications. These application can be written in C or C++. However, these programming languages use a lot of space at programming memory. To create a hardware C project;

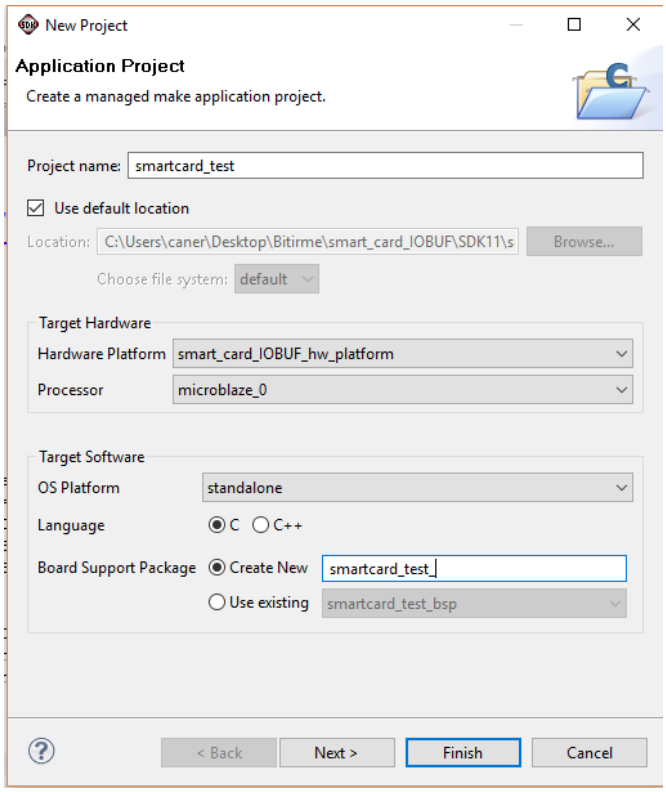

Figure 3.22 : New Application Project

1. As a first step a Template C program is created as show at Figure 3.22.

2. To read ATR from smart card a simple C code is written. Written C code is given at Figure 3.23 on SDK. According to this C code, microprocessor consistently reads *data_ready*, *data_out* signals and if *data_ready* signal assert high, *data_out* signal is printed to terminal as 8-bit hexadecimal number. Also, smart card controller's *bytecounter* signal is read as *byte_out* and printed with *data_out* signal for verification of system.

Figure 3.23 : New Application Project

3. RS-232 UART cable is connected to computer. Under "Run Configurations" made an adjustment to watch terminal via RS-232 cable. Also terminal settings are changed.

4. As a last step, FPGA programmed with generated linker script which shown at Figure 3.24.

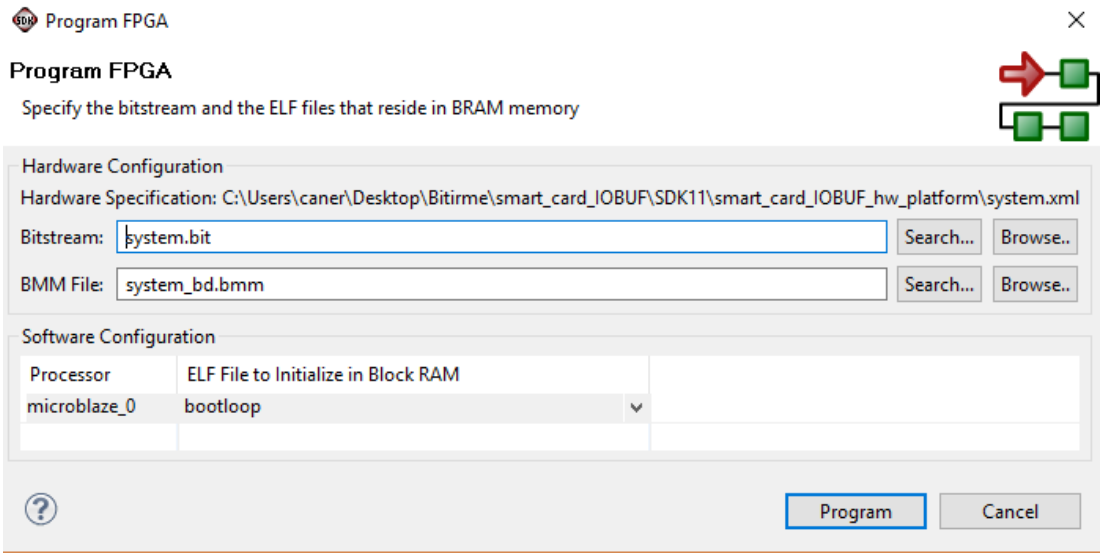

Figure 3.24 : New Application Project

After FPGA programmed, *smartcard_test* application is run on microblaze. Then, smart card is inserted to smart card slot and on the FPGA board *card_enable* switch is asserted high. As a result, 19 byte ATR command observed at terminal as shown at Figure 3.26. ATR bytes received correctly and also at same the order.

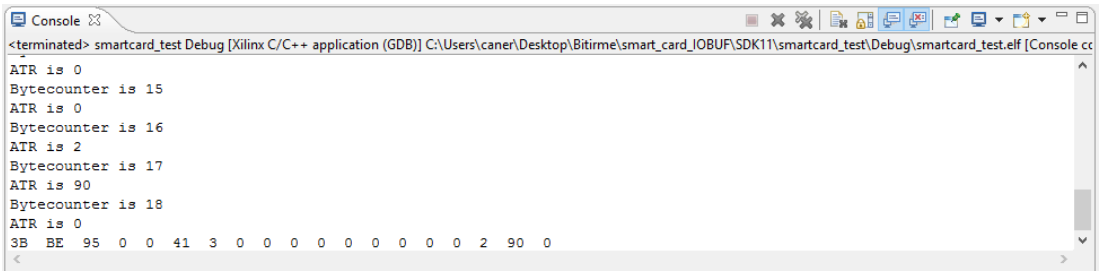

Figure 3.25 : Read ATR Sequence of Smart Card

3.5 Sending a Sequence to Smart Card

ATR is the first protocol of smart card communication. After reading ATR, smart card waits for output commands. To write command to smart card controller, command should be asserted to *data_in* port and *command_ready* should be driven high for 1 cycle. When *command_ready* signal is high, smart card controller reads data from *data_in* port and writes data to serial communication *card_io* bit by bit. General flowchart of process is given at Figure 3.26

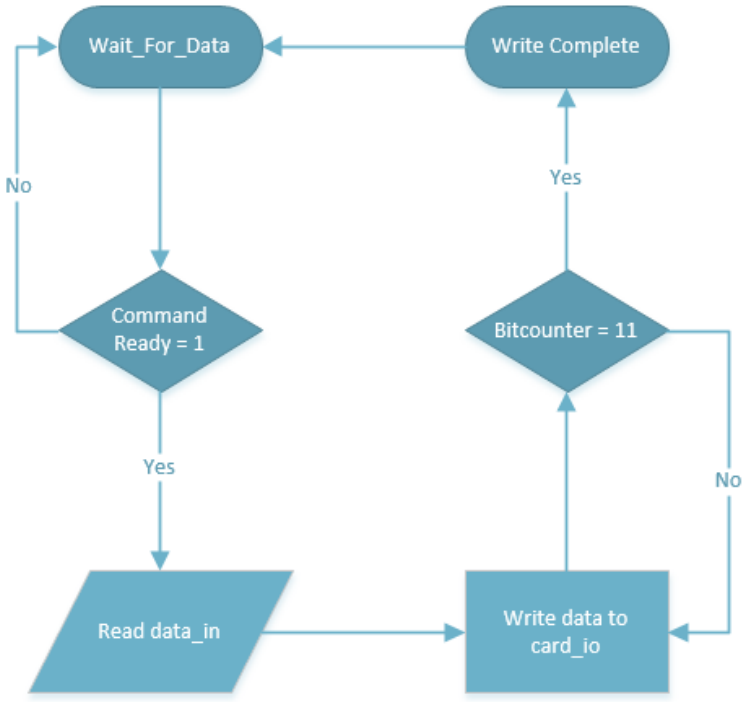

Figure 3.26 : Flowchart of Writing Process

As a first step of writing a command to smart card, ISIM simulation is made. A test bench is written to send *data_in* and *command_ready* signals. The result of simulation was as it should be. After ISIM simulation a write operation is implemented via microblaze. Writing operation is made on same SDK environment at 3.4.2. To send data to smart card controller *data_in* and *command_ready* inputs are connected to *slaveregisters*. To control these registers a simple C code implemented on SDK. First *data_in* signal is sent to smart card controller right after that signal *command_ready* asserted high. 6-byte command data are sent as refer to the smart card's data sheet [19]. ATR should to be change after this command. However, as result of write process any changes were observed. A working system for sending commands to smart card via microblaze could not implemented.

Also a hardware design implemented to send a command to smart card. For this implementation, a simple 6-byte ROM implemented and same 6-byte data was written to ROM. And this ROM, added to design as a trigger of 6-byte write process. When this switch asserted high, smart card controller reads data byte by byte from ROM and instead of *data_in* signal, uses read ROM data. After ATR read, that switch asserted high to write command to smart card nevertheless, the result was same with previous one. A successful write operation could not be performed.

4. CONCLUSIONS

Smart cards have a wide range of applications that require security. The demand for smart cards has led to the production of a variety of smart cards; nevertheless, their properties are strongly based on international standards. Because of this, smart card reader must be implemented according to international standards.

In this project, a smart card module is implemented on the FPGA and this module added to the processor as a custom peripheral. As a microprocessor, Microblaze soft-core microprocessor is implemented on FPGA board. The smart card is connected to FPGA via I/O ports and the power supplied to smart card from FPGA. For microblaze processor a C application is coded and run on microprocessor.

As a result, reading data from the smart card was successful achieved. Also, ATR reading was performed using different smart cards. However, sending commands and writing data to the smart card could not be performed successfully.

4.1 Future Work

As a future work, command sending process should be improved. For instance, smart cards have three different class, a universal smart card reader can be designed in terms of class selection. Also, a simple user interface can be implemented with C language.

REFERENCES

[1] **Rankl, W. and Effing, W.** (2010). *Smart Card Handbook*, Wiley, https://books.google.com.tr/books?id=C55-4kVUQ14C.

[2] **ISO**, (2007), ISO/IEC 7816-3:2006 Identification cards — Integrated circuit cards — Part 2: Cards with contacts — Dimensions and location of the contacts, www.iso.org/standard/45989.html.

[3] **ISO**, (2006), ISO/IEC 7816-3:2006 Identification cards — Integrated circuit cards — Part 3: Cards with contacts — Electrical interface and transmission protocols, www.iso.org/standard/38770.html.

[4] **Smith, G.** (2010). *FPGAs 101: Everything You Need to Know to Get Started*, Newnes, Newton, MA, USA.

[5] **Chu, P.P.** (2007). *FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version*, Wiley.

[6] **Xilinx**, (2008), MicroBlaze Processor Reference Guide, www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf.

[7] **Hendry, M.** (2007). *Multi-application Smart Cards: Technology and Applications*, Cambridge University Press, New York, NY, USA, 1st edition.

[8] **Bidwai, S.S. and Bidwai, S.S.** (2012). Manifestation of a Smart Card Reader using VLSI Technology, *IJCST*, *3*(1).

[9] **Koul, Y. and Pathak, A.** (2016). Design and FPGA-based implementation of Smartcard Reader, *International Journal of Science, Engineering and Technology Research*, *5*(2).

[10] **Selimis, G.**, **Fournaris, A.**, **Kostopoulos, G. and Koufopavlou, O.** (2009). Software and hardware issues in smart card technology, *IEEE Communications Surveys & Tutorials*, *11*(3).

[11] **EMV**, (2011), Integrated Circuit Card Specifications for Payment Systems - Application Independent ICC to Terminal Interface Requirements, https://www.emvco.com/specifications.aspx?id=223, citation date: 24 May 2017.

[12] **Kuon, I. and Rose, J.** (2007). Measuring the gap between FPGAs and ASICs, *IEEE transactions on computer-aided design of integrated circuits and systems*, *26*(2), 203–215.

[13] **Xilinx**, (2011), Spartan-3E FPGA Starter Kit Board User Guide, www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf.

[14] **Pedroni, V.A.** (2010). *Circuit Design and Simulation with VHDL, Second Edition*, The MIT Press, 2nd edition.

[15] **Xilinx**, (2012), ISE In-Depth Tutorial, www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ise_tutorial_ug695.pdf.

[16] **Xilinx**, (2013), EDK Concepts, Tools, and Techniques, www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/edk_ctt.pdf.

[17] **Xilinx**, (2015), Xilinx Software Development Kit (SDK) User Guide, www.xilinx.com/support/documentation/sw_manuals/xilinx2015_1/SDK_Doc/index.html, citation date: 20 May 2017.

[18] **Xilinx**, (2003), CoolRunner-II Smart Card Reader, www.xilinx.com/support/documentation/application_notes/xapp372.pdf.

[19] **Santos, Q.**, ACOS6 Reference Manual, retrieved at 21 .04.2017 via e-mail.

[20] **Xilinx**, (2012), Platform Specification Format Reference Manual for Embedded Development Kit (EDK) 14.1, www.xilinx.com/support/documentation/sw_manuals/xilinx14_4/psf_rm.pdf.

CURRICULUM VITAE

Name - Surname: Caner Bulduk

Place and Date of Birth: Karadeniz Eregli, 26/11/1194

Undergraduate: Istanbul Technical University
Electronics and Communications Engineering (2012 - 2017)