Technical University of Košice

Faculty of Electrical Engineering and Informatics

Department of Electronics and Multimedia Communications

Secured opto-electronic system for product traceability

Architecture, design and feasibility study

Master's Thesis

Ľuboš Gašpar

Supervisor: Assoc. Prof. Miloš Drutarovský, PhD.

Consultant: Prof. Viktor Fischer

Košice 2009

Metadata Sheet

Author:	L'uboš Gašpar
Thesis title:	Secured opto-electronic system for product traceability
Subtitle:	Architecture, design and feasibility study
Language:	english
Type of Thesis:	Master's Thesis
Number of Pages:	76
Degree:	Master degree
University:	Technical University of Košice
Faculty:	Faculty of Electrical Engineering and Informatics (FEI)
Department:	Department of Electronics and Multimedia Communications (DEMC)
Study Specialization:	Electronics and Telecommunication Engineering
Town:	Košice
Supervisor:	Assoc. Prof. Miloš Drutarovský, PhD.
Consultant(s) :	Prof. Viktor Fischer
Date of Submission:	7. 5. 2009
Date of Defence:	27. 5. 2009
Keywords:	traceability, Embedded digital systems, data securization , applied cryptography, reconfigurable circuits, FPGA, cryptoprocessors, ARM
Category Conspectus:	Technika, technológia, inžinierstvo; Elektronika
Thesis Citation:	L'uboš Gašpar: Secured opto-electronic system for product traceability. Master's Thesis. Košice: Technical University of Košice, Faculty of Electrical Engineering and Informatics. 2009. 76 pages
Title SK:	Návrh opto-elektronického systému s kryptografickým zabezpečením pre kontrolu originality výrobkov
Subtitle SK:	Architektúra, návrh a štúdia uskutočniteľnosti
Keywords SK:	vystopovateľnosť, vložene digitálne systémy, hardwarové zabezpečenie údajov, aplikovaná kryptografia, rekonfigurovateľné obvody, FPGA, kryptoprocesory, ARM

Abstract in English

The goal of this thesis is an architecture design, feasibility study of the secured optoelectronic system for product traceability and prototype realization of selected blocks. The thesis demonstrates the implementation of the embedded system based on Actel Fusion FPGA (Field-programmable Gate Array) interconnected with the external small camera from ST Microelectronics. The system communicates with the PC via Cypress USB interface. The core of the system represents the SoC (System On Chip) in FPGA based on the ARM7TDMI processor. The system was implemented in the Actel System Management board based on Fusion FPGA. The thesis also contains a discussion about the implementation of the image processing algorithms with products' authentication, as well as analysis of the possible implementations of the cryptoprocessor for confidentiality and authenticity of transferred data. One part of the thesis deals with the problems related to the implementation of the AES cipher (Advanced Encryption Standard) in non-volatile FPGAs.

Abstract in Slovak

Cieľom práce je návrh architektúry, štúdia uskutočniteľnosti zabezpečeného optoelektronického systému pre vystopovanie produktov a realizácia prototypu z vybraných blokov. Práca demonštruje implementáciu vloženého embedded systému založeného na Actel Fusion FPGA (Field-programmable Gate Array) prepojeného s miniatúrnou externou kamerou od spoločnosti ST Microelectronics. Systém komunikuje s PC pomocou Cypress USB rozhrania. Jadro systému je tvorené SoC (System On Chip) založeným na ARM7TDMI procesore. Tento SoC bol implementovaný vo Fusion FPGA, ktorý je súčasťou Actel System Management board. Práca tiež zahŕňa diskusiu o implementácii algoritmov pre spracovanie obrazu spolu s autentifikáciou produktov, a taktiež analýzu možných implementácií kryptoprocesora pre utajenie a autenticitu prenášaných dát. Časť práce je venovaná problematike implementácie AES (Advanced Encryption Standard) šifrátora v nevolatilných (z ang. non-volatile) obvodoch FPGA.

TECHNICKÁ UNIVERZITA V KOŠICIACH

Fakulta elektrotechniky a informatiky

Katedra elektroniky a multimediálnych telekomunikácií

DIPLOMOVÁ PRÁCA

Študent: **Ľuboš GAŠPAR**

Študijný odbor: **Elektronika a telekomunikačná technika**

Akademický rok: 2008/2009

Názov práce v slovenskom a anglickom jazyku:

Návrh opto-elektronického systému s kryptografickým zabezpečením pre kontrolu originality výrobkov
Design of an opto-electronic system for cryptography-based product traceability

Pokyny na vypracovanie:

1. Navrhnúť architektúru opto-elektronického digitálneho systému pre kontrolu originality výrobkov na báze hradlových polí (FPGA) a overiť funkčnosť jeho jednotlivých častí. Budúci automatizovaný systém má nasnímať digitálny obraz vybranej oblasti na obale výrobku, vypočítať optický podpis výrobku podľa jednoduchého algoritmu, zašifrovať ho a poslať prostredníctvom USB zbernice do pripojeného PC.
2. Navrhnúť a overiť funkčnosť snímania obrazov z kamery do FPGA a ich prenos do PC cez rozhranie USB.
3. Analyzovať možnosť realizácie vybraných matematických operácií pre výpočet optického podpisu procesorom ARM vloženým v FPGA.
4. Navrhnúť jednoduchý kryptoprocesor s obmedzenou inštrukčnou sadou a overiť možnosť jeho realizácie vo vnútri hradlového poľa.
5. Navrhnúť možnosť prepojenia kryptoprocesora s procesorom ARM.
6. Vypracovať dokumentáciu podľa pokynov vedúceho práce (napr. hlavná časť 40 strán a viac, prílohy podľa charakteru práce, elektronická forma hlavnej časti a príloh).

Vedúci diplomovej práce: doc. Ing. Miloš Drutarovský, CSc.
Konzultant diplomovej práce: Prof. Viktor Fischer.

Dátum odovzdania diplomovej práce: 8. 5. 2009

.....
prof. Ing. Dušan Levický, CSc.
vedúci zadávajúceho vedecko-pedagogického pracoviska

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -13-

.....
prof. Ing. Liberios Vokorokos, PhD.
dekan

V Košiciach, dňa 20.2.2009

Thesis assignment

Study Specialization:	Electronics and Telecommunication Engineering
Thesis title:	Secured opto-electronic system for product traceability
Extent of Thesis:	Recommended number of pages: 40 and more
Date of Assignment:	20. 2. 2009
Date of Submission:	7. 5. 2009
Head of the Department:	Prof. Ing. Dušan Levický, CSc.
Dean of the Faculty:	Prof. Ing. Liberios Vokorokos, PhD.

Outline of Thesis:

1. Design the architecture of opto-electronic digital system for secured traceability of products based on the FPGA and verify the functionality of its particular parts. Future automated system has to capture a digital image of the selected region on the cover of the product, compute an optical signature of the product according to a simple algorithm, encrypt the signature and transfer it via the USB to the connected PC.

2. Design and verify functionality of the image acquisition from the camera to FPGA and the transfer of the images to a PC via the USB interface.

3. Analyze the possibility of the realization of selected mathematical operations for the optical signature calculation by the embedded ARM processor in the FPGA

4. Design a simple cryptoprocessor with limited instruction set and verify the possibility of its realization inside the FPGA.

5. Design the method for interconnecting the cryptoprocessor with the ARM processor.

6. Write the documentation according to the instructions given by the supervisor of the thesis (i.e. main part of 40 pages or more, appendices according to the type of the thesis, provide the main part and appendices in the electronic form).

Declaration

I hereby declare that this thesis is my own work and effort. Where other sources of information have been used, they have been acknowledged.

Košice 7. 5. 2009 . . . . . .

Signature

Acknowledgement

I would like to thank to my family for their unconditional love and support. I would like to express my sincere thanks to my supervisor Assoc. Prof. Miloš Drutarovský for his valuable recommendations and carefull review of the thesis. Special mention and thank should go to Prof. Viktor Fischer for his constant and constructive guidance throughout the thesis, support and patience during long phone calls and discussions. To all other who gave a hand, I say thank you very much.

Preface

Counterfeiting of the products represents a very serious threat nowadays. The trend of counterfeiting is as old as the humankind itself. However, the development of more advanced techniques of production goes hand in hand with enhancements in counterfeiting techniques. In order to be able to distinguish the original products from the fake ones, production techniques have to develop faster than enhancements in counterfeiting techniques. One of the promising solutions is the miniaturization of the optical markers for recognition of originals. However, a new issue shows up. This miniature markers have to be localized, scanned and the originality of the product has to be confirmed. Therefore it is essential to design secured opto-electronic system for product traceability.

Several years ago, the Hubert Curien Laboratory in Saint-Etienne, France together with the SignOptic private company have proposed an image processing algorithm enabling cheap and reliable product recognition and authentication. The system is already validated in industrial applications working in product area (factory). However, industrial needs have shown that it would be very useful to build a simple and portable device that should be able to authenticate products as well as to authenticate the factory in a hostile environment. The design of a secured embedded system suitable for such kind of applications has become the aim of a research and development project of the AMTeS (Architectures Matérielles pour le Traitement et la Sécurisation des Données) research team from the laboratory that is in charge of Professor Viktor Fischer. Professor Fischer and his team have invited me two times in order to work on the project. The objective of the first stay during the summer 2008 was to implement an image acquisition embedded system connected to a PC via USB interface. The aim of the second stay was the work on the overall system design. Although the system should be small and handy, its architecture will be certainly very complex and its detailed structure is not known, yet. For this reason, the objective of my work was to propose an architecture that would be:

1. evolutive (in order to permit to upgrade the system)

2. embeddable in digital devices (in order to be small in size)

3. featuring sufficient hardware and software means (in order to implement easily serial algorithms and parallel hardware structures)

4. secured (containing cryptographic means as random number generator and crypto-processor)

5. easily connectable to internet

For the above-mentioned reasons, AMTeS team has selected the reconfigurable logic devices (FPGAs) as the hardware platform.

The work on the project needed a large spectrum of knowledge and abilities: starting from programming and description languages like VHDL (Very High Speed Integrated Circuit Hardware Description Language), C and C++, communication protocols like I^2C (Inter-Integrated Circuit) and USB, microprocessor architectures, image processing, applied cryptography and cryptographic protocols. It was not easy to acquire all the necessary knowledge in a relatively short time. But the fact that I have the possibility to participate on the development of a product dedicated to very important industrial applications, has motivated me very much.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

2D **Two d**imensional

ABC **A**PB **B**us **C**ontroller

AES **A**dvanced **E**ncryption **S**tandard

AHB **A**dvanced **H**igh-performance **B**us

ALU **A**rithmetic **L**ogic **U**nit

AMBA **A**dvanced **M**icrocontroller **B**us **A**rchitecture

AMD **A**dvanced **M**icro **D**evices

AMTeS Architectures Matérielles pour le Traitement et la Sécurisation des Données

ANR **A**gence **N**ationale pour la **R**echerche - France national agency for research

APB **A**dvanced **P**eripheral **B**us

ARM **A**dvanced **R**ISC **M**achine. IP owned by ARM Ltd.

ASB **A**dvanced **S**ystem **B**us

ASIC **A**pplication **S**pecific **I**ntegrated **C**ircuit

AT **A**ffine **T**ransfomation

AT_BT Combined **A**ffine **T**ransformation and **B**asis **T**ransformation

BCB **B**orland **C**++ **B**uilder

BT **B**asis **T**ransformation

C general-purpose computer programming language developed in 1972 by Dennis Ritchie at the Bell Telephone Laboratories

C++ general-purpose computer object programming language developed from **C** programming language

CBC **C**ipher **B**lock **c**haining – confidentiality block cipher mode

CCC **C**lock **C**onditioning **C**ircuit

CCM **C**ounter with **C**BC-**M**AC is a confidentiality and authenticity block cipher mode

CE **C**hip **E**nable signal

CFB **C**ipher **F**eed**B**ack – confidentiality block cipher mode

CMAC **C**ipher **MAC** – authenticity block cipher mode

CMOS **C**omplementary **M**etal-**O**xide-**S**emiconductor

cmp **C**o**mp**arison

CTR **C**oun**te**r mode – confidentiality block cipher mode

CTRo **C**ipher-**T**ext **R**egister **o**utput

DDR **D**ouble **D**ata **R**ate

DFF **D**-**f**lip-**f**lop circuit

DI **D**ata **I**n

DIP **D**ual **I**n-line **P**ackage of switches

DMA **D**irect **M**emory **A**ccess

DO **D**ata **O**ut

DPRAM **D**ual-**P**ort **RAM**

ECB **E**lectronic **C**ode**B**ook – confidentiality block cipher mode

ES **E**mbedded **S**ystem

FBGA **F**ine **B**all **G**rid **A**rray

FIFO **F**irst **I**n, **F**irst **O**ut

FIPS **F**ederal **I**nformation **P**rocessing **S**tandard

FLASH Flash memory is a non-volatile memory that can be electrically erased and re-programmed

FPGA **F**ield-**p**rogrammable **G**ate **A**rray

fps **f**rames **p**er **s**econd

GCM **G**alois/**C**ounter **m**ode – high-throughput block cipher mode for confidentiality and authenticity

GF **G**alois **F**ield

HSYNC **H**orizontal **Sync**hronization signal

I^2C **I**nter-**I**ntegrated **C**ircuit

I/O **I**nput/**O**utput

IDE **I**ntegrated **D**evelopment **E**nvironment

IP **I**ntellectual **P**roperty

JTAG **J**oint **T**est **A**ction **G**roup

KI **K**ey **I**n

KO **K**ey **O**ut

LCD **L**iquid **C**rystal **D**isplay

LED **L**ight-**E**mitting **D**iode

LFSR **L**inear **F**eedback **S**hift **R**egister

LHC **L**aboratoire **H**ubert **C**urien, Saint-Etienne, France

LSB **L**east **S**ignificant **b**it

LUT **L**ook-**U**p **T**able

MAC **M**essage **A**uthentication **C**ode

MK **M**ain **K**ey

N/A **N**ot **A**vailable

NIST **N**ational **I**nstitute for **S**tandards and **T**echnology

NLFSR **N**on-**L**inear **F**eedback **S**hift **R**egister

OFB **O**utput **F**eed**B**ack – confidentiality block cipher mode

PCI **P**eripheral **C**omponent **I**nterconnect - widely used computer peripheral bus

PLL **P**hase-**L**ocked **L**oop

PSC **P**rogrammable **S**ystem **o**n **C**hip

PTRi **P**lain-**T**ext **R**egister **i**nput

QVGA **Q**uarter **VGA** - display standard with resolution 320x240

RAM **R**andom **A**ccess **M**emory

RC **R**esistor-**C**apacitor based oscillator

REG **Reg**ister

RGB **R**ed-**G**reen-**B**lue color model

RISC **R**educed **I**nstruction **S**et **C**omputer

RNG **R**andom **N**umber **G**enerator

ROM **R**ead **O**nly **M**emory

RS232 **R**ecommended **S**tandard **232** is a standard for serial binary data transfer

S-box **S**ubstitution **box**

SCL **S**erial **Cl**ock signal of an I^2C bus

SDA **S**erial **Da**ta signal of an I^2C bus

SDF **S**tandard **D**elay **F**ormat

SecReSoC **Sec**ured **Re**configurable **S**ystem **o**n **C**hip

SK **S**ession **K**ey

SM **S**tate **M**achine

SoC **S**ystem **o**n **C**hip

SRAM **S**tatic **RAM**

TDPRAM **T**rue **D**ual-**P**ort **RAM**

USB **U**niversal **S**erial **B**us

VGA **V**ideo **G**raphic **A**rray

VHDL **V**HSIC **H**ardware **D**escription **L**anguage

VHSIC **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit

VSYNC **V**ertical **Sync**hronization signal

XOR E**x**clusive **OR** logic operation

YUV color model **YUV** containing Luminance component and two Chrominance components

Introduction

Humankind crossed the borders of the 21^{ST} century with new fascinating technologies and rapid technical advance. These technologies are, however the outcome of the previous progress and innovations change and replace each other to form a chain of our development. But as nothing is so simple and onefold, there are many shades of utilization of the technical progress. The only desired utilization of this progress is to help people, whether in everyday life, or beyond the visible innovations for common people, with a special emphasis on these improvements' legality and correspondence with particular international conventions and norms. Therefore the creation of new and original products is just as important for progress as the protection of seals of originality, authorship and copyrights. To keep it simple, it can be said that it is important to search for all the ways of preventing fake products to get a chance to enter the market and thus violate the rights of both authors and consumers.

Throughout the previous centuries there have been numerous attempts to create and introduce new procedures for distinguishing between fake and original products. Beginning with the old valuable paintings, proceeding throughout various works of arts and coming to the modern brands of fashion, no original product could have been saved from the threat of being copied. Smugglers have always tried to gain profit by using realizations of ideas of other people. However with the expansion of supplies and demands beyond the borders of individual countries and with the existence of the free market, the problem with fake products starts to be more important nowadays than ever before. And it is the task of new technologies to develop new systems and procedures to detect the originality of products. As a result, this thesis deals with one of the possible technologies that could be used for solving this problem.

The product authentication method proposed by the Hubert Curien Laboratory (later just LHC) together with the SignOptic company permits to recognize and to authenticate products using image processing. The image processing method algorithm proposed by this consortium is relatively simple so that it could be embedded in a small and portable

hardware unit. Since the authentication device can be used in potentially hostile environment, the authentication data and other confidential information has to be protected by secured cryptographic algorithms.

The aim of this thesis is to propose an architecture of secured embedded system (module) composed of several functional parts: simple camera with integrated optics and digital output, image processing unit, data security (encryption and authentication) block and communication block. The proposed architecture is determined by the requirement of the use of Actel FPGAs (security reasons) and embedded soft-core processor ARM7. Most of the hardware should be placed in one chip - FPGA. The only modules that could be implemented outside FPGA are the communication (USB) interface and program/data memory.

Since the required system is relatively complex, it cannot be designed and implemented in one thesis. Instead, the aim of this work is the design of a global architecture and a feasibility study of its selected blocks (e.g. image processing block). However, some functional blocks have been designed and simulated in VHDL (e.g. AES cipher optimized for non-volatile FPGAs), some were designed, simulated and tested in hardware (e.g. acquisition block and communication block using USB interface). The architecture of the crypto-processor has been proposed on a structural level (without description in VHDL and simulation).

Chapter 1 describes the overal secured embedded system for product authentication and the roles of its modules. Chapter 2 describes available hardware platform, including ARM processor and embedded camera module. Chapter 3 concerns all develpment tools that have been used during the work. Chapter 4 introduces implementation of image acquisition within the ARM7 subsystem. Chapter 5 analyses the used image processing, signature extraction and final product's authentication algorithms and its feasibility in the selected hardware. The analysis of the possible cryptoprocessor implementations, implementation of the AES cipher as well as implementation and mathematical description of S-box generation in the logic are described in Chapter 6. Firmware for ARM7 and software in PC development and its first version is introduced in Chapter 7.

1 Overview of the project

The architecture of the whole system is depicted in Fig. 1 – 1. It is composed of the secured authentication device and external authentication authority including product database (this authority usually is situated in the factory). The product authentication procedure is composed of the preparation and evaluation phase.

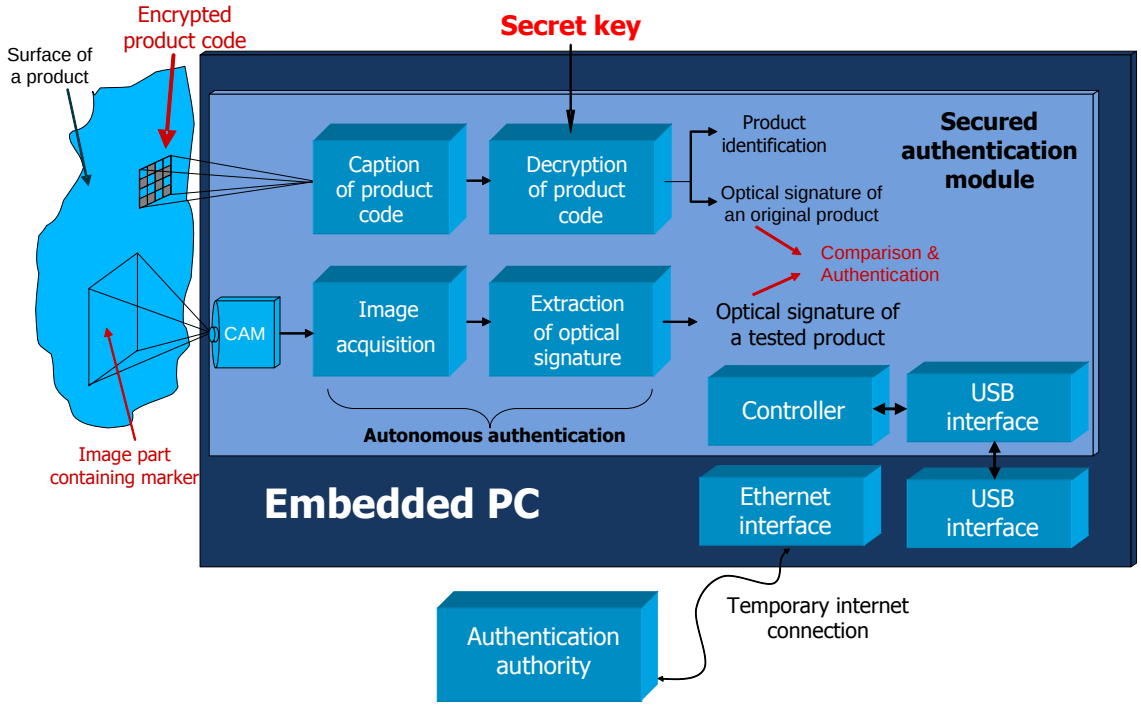

Fig. 1 – 1 Block diagram of the secured opto-electornic system for product traceability

In the preparation stage (in the factory) the system prints an uncopyable characteristic marker (e.g. a small point) on the product, scans this printed marker and calculates its optical signature (unclonable function). The method must guarantee that it is impossible to copy (to fake) the marker. The optical signature of the product is encrypted and printed on the product package, too. We will call this encrypted signature of the original product a reference signature.

In the authenticity evaluation phase a portable device containing camera (scanner) will read the area containing marker and calculate its optical signature. Then it will read the reference signature, decrypt it with the same key that was used in the factory. Finally, the

reference signature is compared with the acquired and calculated signature. If both are equal, the product is original.

In order to authenticate the authentication device itself and to exchange encryption keys, some protected channel (e.g. via Internet) must exist at least for some short time intervals.

For some practical reasons (modularity), the authentication device has to be composed of a secured authentication module connected to an embedded PC via USB interface. The PC itself is connected to the authentication authority via Internet.

Regarding Fig. 1 – 1, the aim of this work is the design of the secured authentication module including USB interface (without embedded PC and ethernet connection). For practical reasons, the architecture have been decomposed to several independent functional blocks that do not correspond exactly to those given in Fig. 1 – 1, but that were more easy to design together.

First, the data acquisition block including camera initialization, ARM controller and USB interface have been designed, simulated and tested. In order to validate the functionality, acquired image data have been saved in external RAM (outside FPGA, but inside the module) and then transferred to the PC via USB bus in order to be visualized on the screen. This part of the work has permitted to connect external devices to the module (camera, external memory, PC via USB bus).

In the second phase of the project it was necessary to design the core of the cryptographic module - the AES cipher including key expansion procedure. The cipher/decipher had to be adapted to the required non-volatile FPGA family. Namely, S-boxes implemented in a volatile embedded memory had to be generated inside the device. The cipher/decipher block has been designed and simulated in VHDL and mapped to the selected FPGA device.

The third phase of the project consisted in the feasibility study of the image processing algorithm proposed by the laboratory. Together with A. Idrissa, which has implemented the algorithm in software, I had to analyze each function (if possible) from the point of view of its implementation in hardware.

In the fourth and the last phase of the project I had to work on the architecture of the crypto-processor. This work has started with a study of existing crypto-processors and the possibilities of the use of existing open-source microprocessor designs as a basis to the future crypto-processor architecture. Following this study I have proposed two architectures of the crypto-processor, having 128-bit and 32-bit datapath, respectively.

Course of development is illustrated in Fig. 1 – 2 to Fig. 1 – 6 in detail, respectively. All stages carried out by the AMTeS team are visualized by green color. My own work or in collaboration with the AMTeS team are illustrated by blue color. Acquired first version of AES cipher (without decipher, key expansion and S-box generation) is shown by red color. Expected development in the future is visualized by brown color.

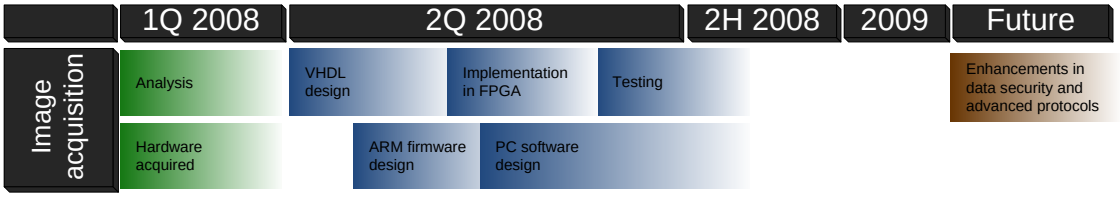

Fig. 1 – 2 Image acquisition development stage

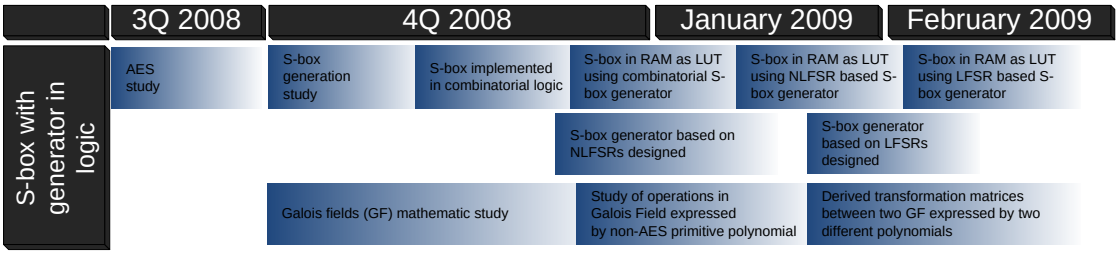

Fig. 1 – 3 S-box development stage

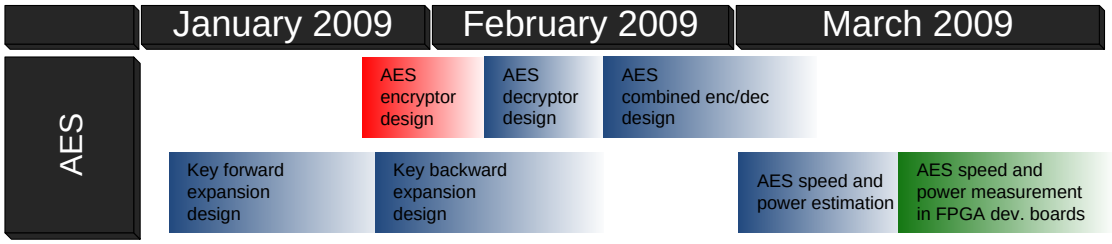

Fig. 1 – 4 AES development stage

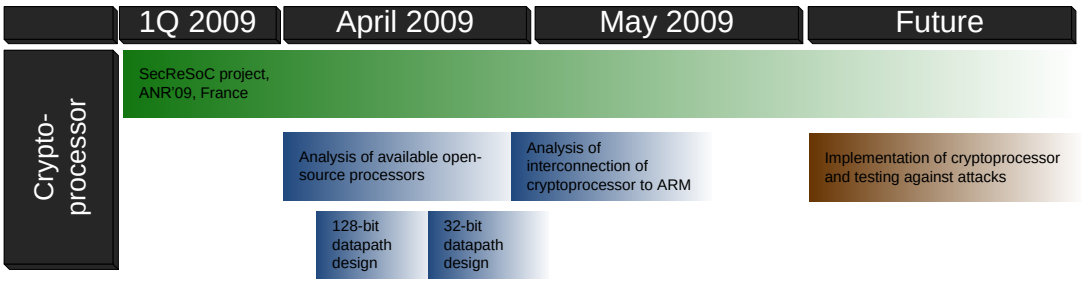

Fig. 1 – 5 Cryptoprocessor development stage

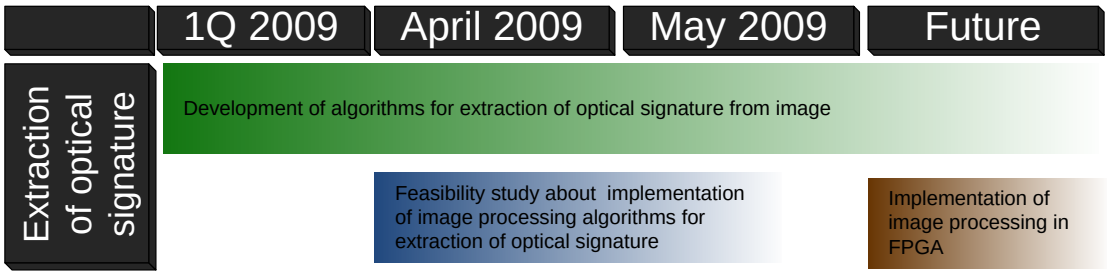

Fig. 1 – 6 Development and analysis of potential implementation of image processing and product authentication stage

2 Description of hardware platform

Designing of such complex product as the product authentication device for secured traceability of original products (later just scanner) requires certain subsequent steps in the product's design. As long as the product has to represent stand-alone system, the embedded system (later just ES) has to be developed in the first step. Subsequently CMOS camera chip has to be connected to the embedded system [5]. ES has to contain these elements:

- SoC design inside the FPGA based on the ARM7 processor
- Static RAM memory
- USB interface

The hardware platform is the most important part of the design because it directly influences device's costs, performance, power consumption and reliability. The following hardware platform was obtained for the prototype:

- ST Microelectronics X24 development board [6] with 1.3 Mpix embedded camera chip VS6524 [5]
- Actel System Management Board [7] with embedded:
  - Actel Fusion M7AFS600 FBGA484 FPGA [2] (with ARM7 support)
  - Santa Cruz interface
  - Two SRAM modules (GSI GS88018BT-200) - 2 MB in total [8]
  - Two FLASH modules (ST M29W800DT) - 2 MB in total [9]
- USB board [10] with Cypress USB module (CY7C68013A-100AXC) [11] - Santa Cruz compliant
- PC (AMD Athlon XP 2600+, 1 GB DDR ram, USB 2.0 support)

The selection of VS6524 camera chip was a very good choice for its rich image processing settings. In order to omit initial configuration of the FPGA, Actel FPGA was selected as the only producer of non-volatile FPGAs on the market [2]. Cypress USB module complies the Santa Cruz interface specifications [10] so it is suitable for the Actel System Management Board. Flash memory modules were not used in this project but are available for being used in the future.

2.1 Actel system management board

Actel System Management Board is a development kit [7]. It includes *Actel Fusion M7AFS600 FPGA* [2]. ARM7 processor is also available for instantiation inside the FPGA [2], [12]. The board is suitable for creating systems with enhanced power management, thermal management, serial and ethernet communication, ARM7 microprocessor subsystem with external memories, PCI ports and *Santa Cruz* extension connectors. Board also includes simple user interface - small keyboard, DIP switches, red and green LEDs and LCD display. Configuration as well as debugging is carried out through JTAG circuitry. Additionally second *A3P250 FPGA* is present. *M7AFS600 FG484* couldn't provide enough free pins, therefore *A3P250* was included [7]. Fig. 2 – 1 illustrates *Actel System Management Board* [7]. The following resources from all available are used:

- *M7AFS600 FPGA* [2]
- two SRAM memory chips [8]
- *Santa Cruz* interface
- *Legacy* interface
- Reset circuitry
- *JTAG* circuitry (for FPGA programming only)
- *RS232* port (used during development instead of USB)

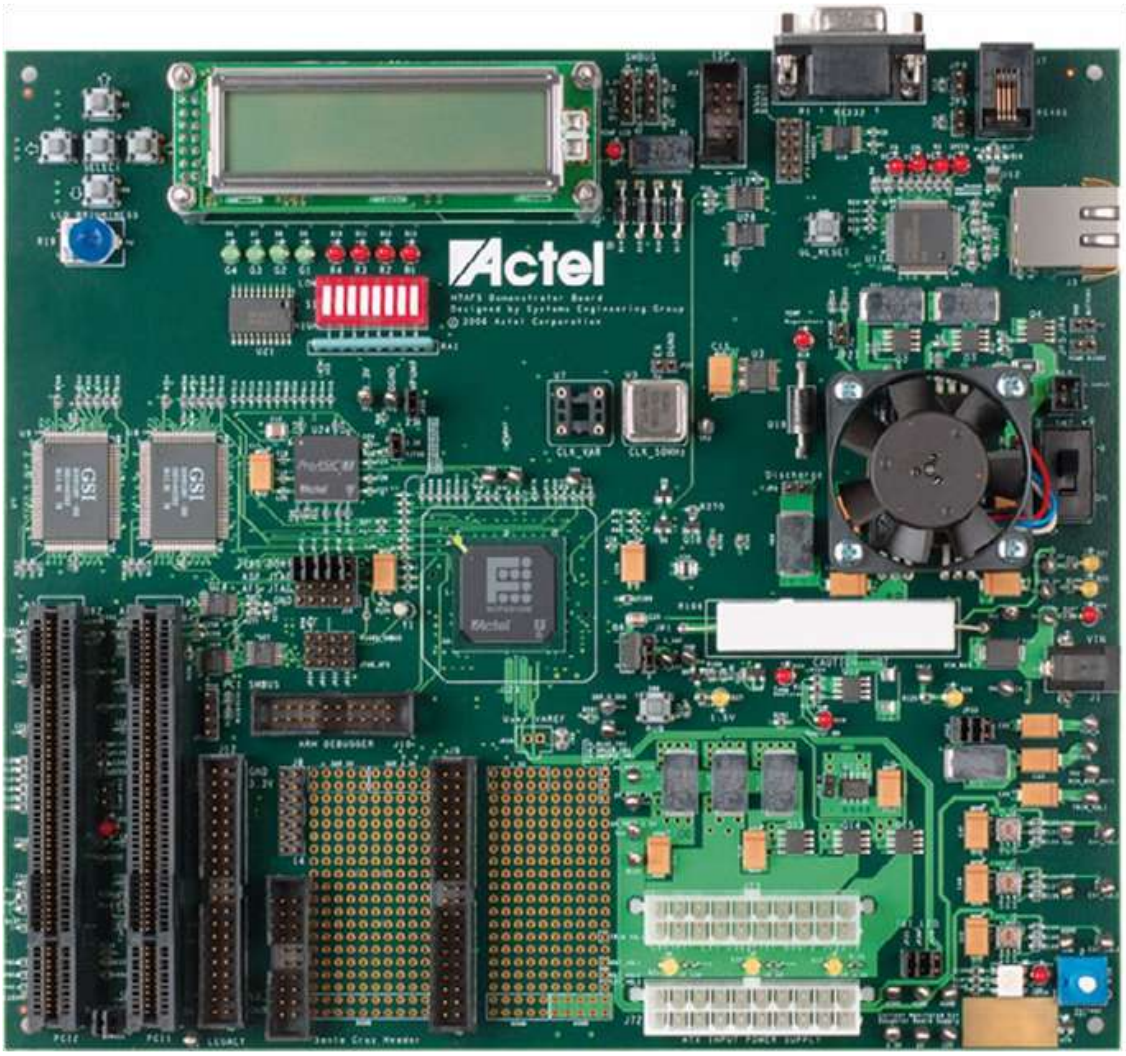

Fig. 2 – 1 Actel System Management Board

2.2 Actel Fusion FPGA architecture

2.2.1 Actel Fusion

Actel Fusion was designed as mixed-signal Programmable System Chip (PSC). It combines advantages of flash FPGA core, flash memory blocks and configurable analog blocks [2]. Flash-based non-volatile memory retains configuration even if power is turned off, so in spite of other producer's products, Actel Fusion can be used as a single-chip solution. Fusion devices have a 128-bit flash-based lock and industry-leading AES decryp-

tion, used to secure programmed intellectual property (IP) and configuration data. Actel Fusion is produced using 130 nm process technology. Highest supported clock frequency is 350 MHz, but I/O pads allow at most 250 MHz. Chip's source voltage is 3.3 V, however the core is powered by 1.5 V (voltage regulated by on-chip 1.5 V regulator) [2].

The Fusion family consists of AFS250, AFS600 and AFS1500 devices [2]. Most powerful AFS1500 has up to 1.5 M system gates. AFS600 and AFS1500 support CoreMP7 processor (derived from ARM7TDMI) and are denoted as M7AFS600 and M7AFS1500. The particular design was targeted for M7AFS600 device, so it will be described in greater details [2].

2.2.2 M7AFS600 device

M7AFS600 contains 600 000 system gates, which form 13,824 VersaTiles (also called tiles) [2]. Chip includes two PLLs , six CCCs as well as internal RC oscillator (100 MHz) and Crystal oscillator (50 MHz external crystal oscillator was used instead) [2]. Accuracy of RC oscillator is 1%. The device includes 4 Mb of flash memory, which is divided into two separate 2 Mb blocks. Flash ROM of capacity 1 kb is also included. Total amount of SRAM is 108 kb divided into 24 SRAM blocks. Each block can store 4608 bits and allows these configurations: 4608x1, 2304x2, 1152x4, 512x9 (Dual-Port) and 256x18 (Single-Port). M7AFS600 is packed in FG484 FBGA from which 172 pins form digital I/Os and 40 pins form analog I/Os. Fig. 2 – 2 shows structure of M7AFS600 [2].

2.2.3 Fusion core

The Fusion core consists of logic cells called VersaTiles, which support 3-input LUTs, Latches with clear or set and D-flip-flops with clear or set and enable [2]. LUTs are very important when used to create combinatorial logic. Latches and D-flip-flops are essential structures for sequential logic. Structure of VersaTile [2] is shown in Fig. 2 – 3.

Every VersaTile is connected to its 8 neighbors by ultra-fast local lines [2]. For connecting longer distances, long-line and very-long-line resources are used. Fig. 2 – 4 shows

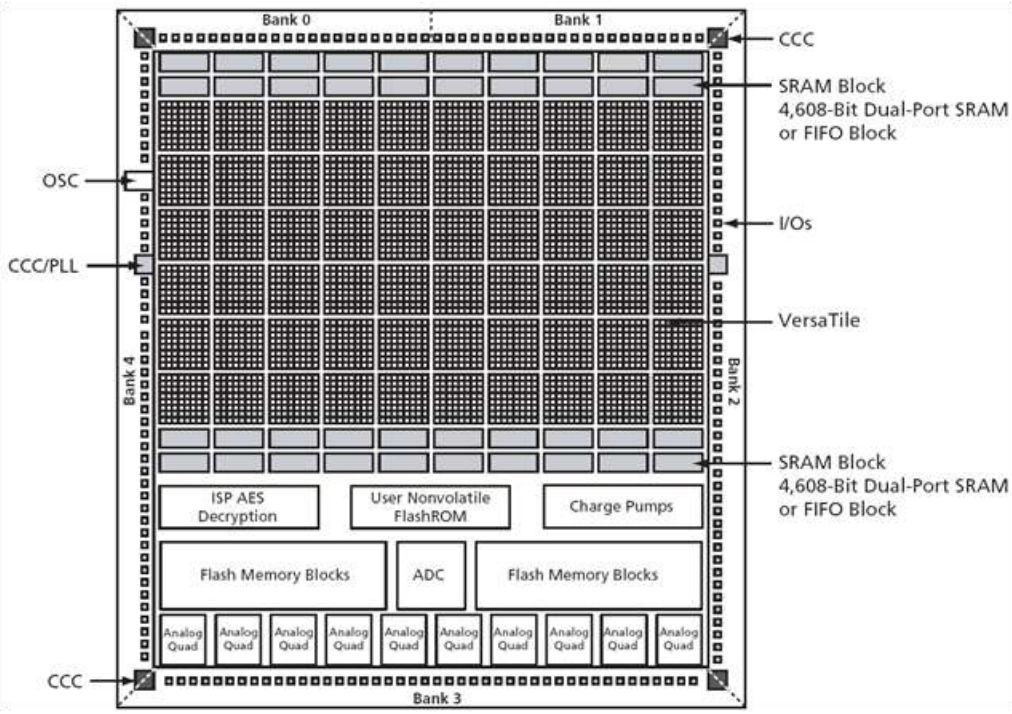

Fig. 2 – 2 Structure of Actel Fusion

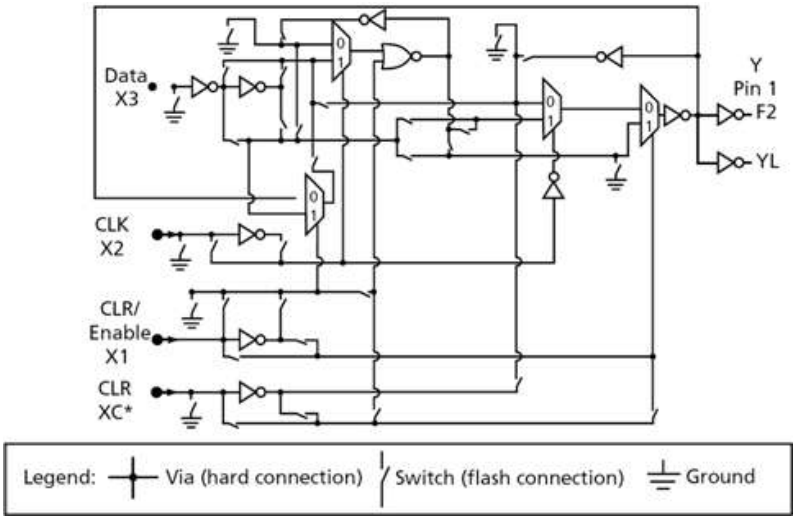

Fig. 2 – 3 Structure of VersaTile

ultra-fast local lines system. Clock and reset signals can be connected to VersaNet global networks, which are connected to six CCCs (clock conditioning circuits). The core is organized into quadrants. Three CCCs are connected to west global resources (in west

quadrants) and three CCCs are connected to east global resources (in east quadrant). For more detailed description of Actel Fusion's inner structure, please refer to Actel Fusion's documentation [2].

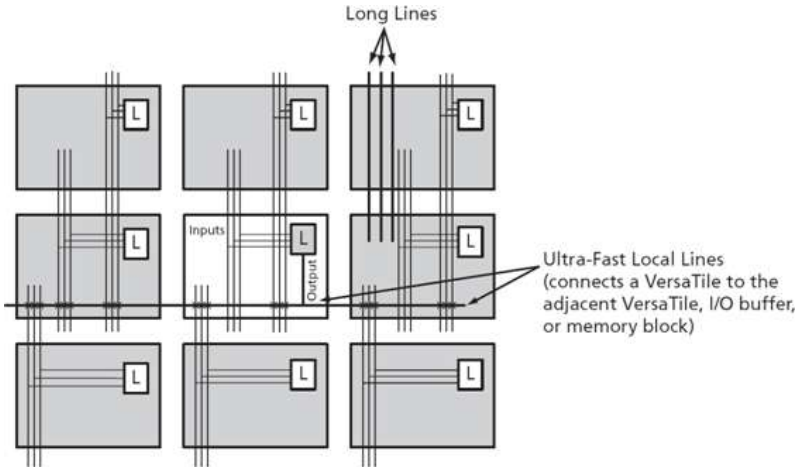

Fig. 2 – 4 Ultra-Fast Local Lines connected to the eight nearest neighbors

Analog blocks were not used in this project, so these parts will not be described in this documentation. For further information about analog blocks, please refer to Actel Fusion's documentation [2].

2.3 IP processor based on ARM7TDMI

2.3.1 The ARM7TDMI-S processor architecture

ARM7TDMI-S (further just ARM7) [13], [12] is world-wide-spread soft IP processor [12]. This processor was designed by ARM Ltd. company to be a part of ASIC's design to form powerful, low cost and flexible SoC [12]. ARM7 has found its application in Game Boy Advance, Nintendo DS or iPod [14]. Many producers include this soft IP in their microcontrollers, because of its unique features (e.g. Atmel AT91SAM7) [12]. ARM7 is 32-bit RISC processor, which also support 16-bit instructions via Thumb instruction set. Registers are orthogonal. It has Von Neumann architecture with a 32-bit data bus carrying both instructions and data. ARM7 uses 3-staged pipeline. The address bus provides 4 GB of linear addressing space. Fig. 2 – 5 shows block diagram of ARM7 [13], [12]. Processor

is connected via the bridge to the AMBA bus [15], [13], [12]. This bus is a standard and it includes AHB, ASB and APB buses. AHB bus is a high-performance system bus, widely used to interconnect ARM7 core with memory interface, PCI interface, and other interfaces which require short access time and high throughput. It is a multi-master bus, so more ARM7 cores can share it. ASB was formerly a high-performance system bus, but it was replaced by AHB. APB is a peripheral bus. It is optimized for minimal power consumption and reduced interface complexity. It is low-performance and is not pipelined. Main advantage is its simplicity [15]. Only one bus master is permitted, the peripherals are connected to the APB as its slave devices [15], [13], [12].

2.3.2 The Actel CoreMP7 processor architecture

Actel's CoreMP7 is the only soft IP ARM7 core that is used in FPGAs [12]. It is fully compatible with ARM7TDMI-S core. Actel has minimized the size of ARM7 core and maximized speed, so it could run in Actel's ProASIC3 and Fusion flash-based FPGAs [12], [2]. It is provided with or without the on-chip debug circuitry. The debug circuitry is about half the size of ARM7 core, so being able to remove it after the debug may allow design to be implemented in a smaller device. Instructions for CoreMP7 are stored in the internal flash memory, so there is no need to connect external memory. Internal SRAM blocks are also used. Using Actel's CoreConsole software, CoreMP7 can be implemented with other IP cores to form more complex design [12].

2.4 Micronic development module with Cypress USB

To establish communication and control between PC and FPGA design, *Cypress* USB module is needed. Company *Micronic* designed *USB_CYPRESS module* [10], which contains *Cypress CY7C68013A - 100AXC* chip [11]. This module complies with *Santa Cruz* specification, therefore it fits to *Actel System Management Board*. After power-up, while there is no non-volatile memory on the module, firmware for *Cypress chip* has to be always programmed inside. Socket for ROM memory is also on the board, but during the

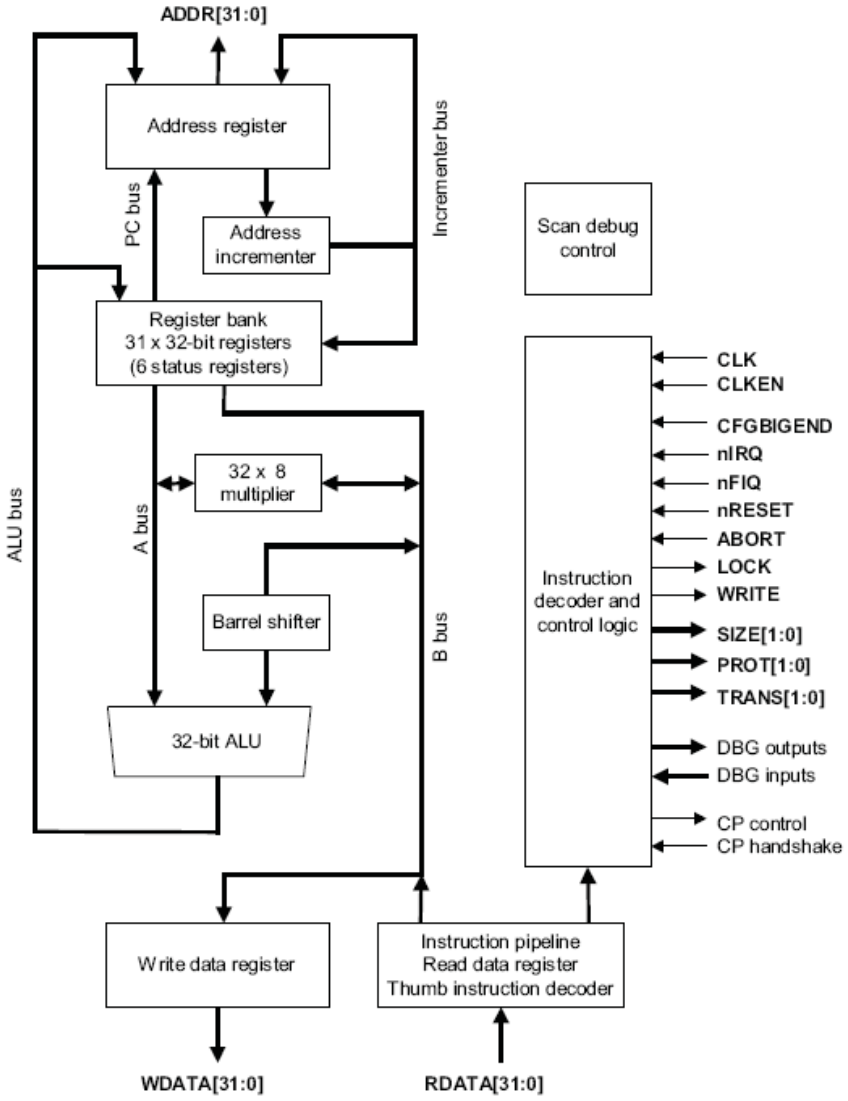

Fig. 2 – 5 ARM7TDMI-S architecture

design period this memory wasn't available. *Cypress chip* supports *USB 2.0* and communicates with *Santa Cruz* interface by 16-bit wide tri-state bus. For more details about the module, programmer's model and firmware please refer to Cypress CY7C68013A - 100AXC manual [11] or *Micronic* documentation [10]. The module is illustrated in Fig. 2 – 6 and Fig. 2 – 7.

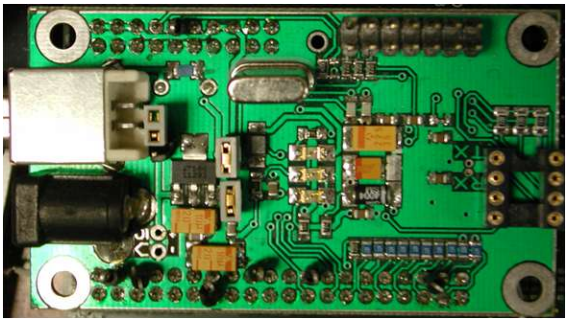

Fig. 2 – 6 Micronic USB_CYPRESS module - front side

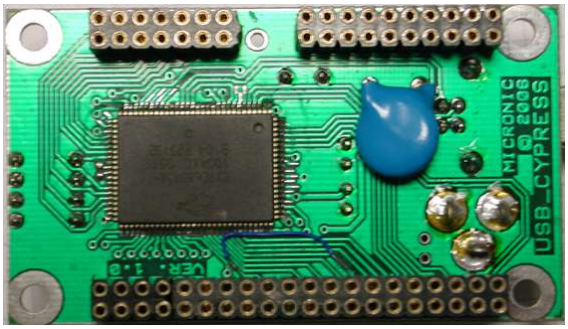

Fig. 2 – 7 Micronic USB_CYPRESS module - back side

2.5 X24 camera integration kit

X24 kit [6] is a product of *ST Microelectronics*. It was created to demonstrate camera chip features, to simplify design for developers and to improve time-to-market. *X24* board is shown in the Fig. 2 – 8. Camera chip (VS6524) [5] is soldered in a small connector board, which is placed in the *X24* board's socket [6]. Logically, *X24* evaluation board is divided into 3 parts (see Fig. 2 – 8). The *left part* contains *Cypress USB controller*, *Altera FPGA* and *USB connector*. This part was used in the first stages of project design. It helps the developer to find the appropriate *configuration* for camera chip easily. This part was later disconnected by *jumpers* in *middle part*. The *middle part* contains interface select *jumpers*, *12 MHz crystal oscillator* and *camera socket*. *Camera mini-board* is placed in this socket. Camera chip is soldered to this mini-board in its center. The *Right part* contains *voltage translation chip* and connectors for *external interface*. The camera operates with *2.8 V* [5], but *Actel FPGA board* supports *3.3 V* [2]. This is the reason, why

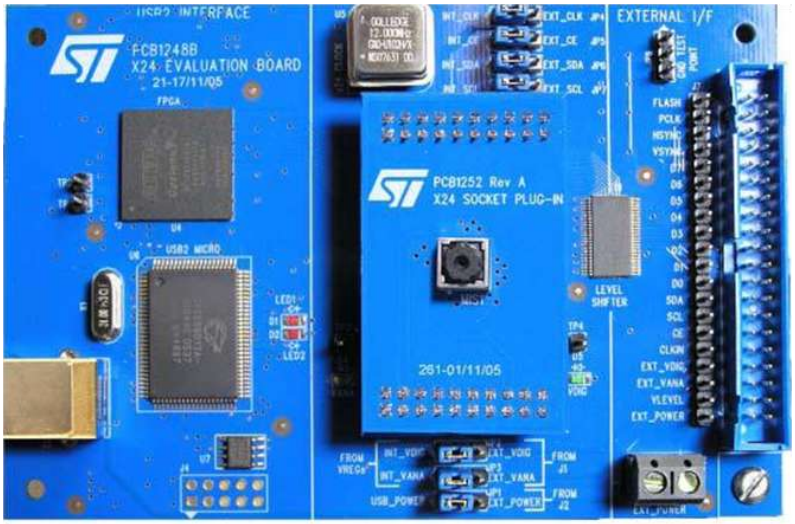

Fig. 2 – 8 X24 camera integration kit

voltage translation chip is used in *X24 board*. *X24 board* has to be connected to *Actel FPGA System Management Board* [7].

2.6 STM VS6524 camera

Camera chip *VS6524* from *ST Microelectronics* is a configurable system on chip [5]. Camera chip has *1.3 MPixel* CMOS array [5]. Fig. 2 – 9 shows this camera chip in scale 7:1 [5].

System consists of a microprocessor, clock generation circuit, video timing circuit, VGA pixel array, video pipe and I^2C interface [5].

Inside the camera, clock generator is driven with the external clock signal [5]. In this project 12 MHz external clock source was used.

All the external control of the camera is carried out by modifying internal camera registers through I^2C interface (signals *SDA* and *SCL*) [5]. By modifying register's content, user is able to adjust many camera features like contrast, saturation, white balance, output format, fps, blanking data, *HSYNC* and *VSYNC* signals, sleep state, play-pause-stop control, etc. [5].

Microprocessor monitors and controls all camera operations [5]. It is the main control

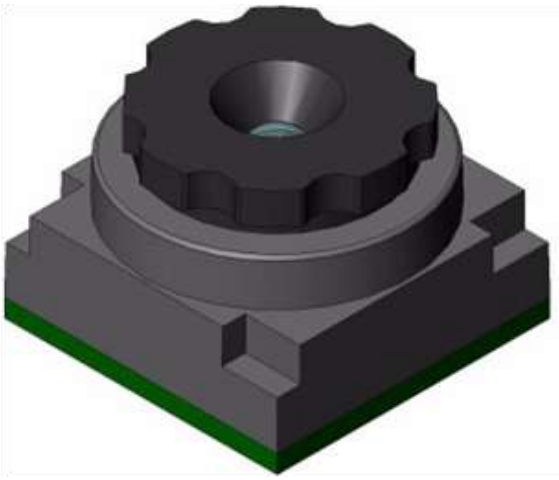

Fig. 2 – 9 Camera chip VS6524 in scale 7:1

element inside the camera chip. It performs the following tasks: handles I^2C communication, video pipe configuration, automatic exposure control, flicker cancellation, automatic white balance, dark calibration and active noise management.

According to the selected output video format, video timing and video pipe are adjusted [5]. *VS6524* has very rich image processing features. All image processing features are processed in video pipe (sharpening, gamma correction, YUV conversion, output format adjustment, cropping, contrast adjustment, saturation adjustment, etc.). There are two equal video pipes in *VS6524* (*pipe 0* and *pipe 1*). Each pipe can have its own image processing settings. Only one pipe can output at a time. Pipes can be used to alter between two sets of settings very quickly. There is also possibility to alter pipes every second frame. In this project, only *pipe 0* was used.

Fig. 2 – 10 illustrates simplified structure of camera chip [5].

After power-up, *initialization* has to take place. During the initialization, internal camera registers are being accessed and the camera is being configured. After the *initialization* has taken place, it is possible to start the *capturing process* and data are sent to the output interface. These are the most important signals: *PCLK*, *HSYNC*, *VSYNC* and *D7...D0*. A detailed description of *initialization* will be discussed later. For further camera's operation control description, refer to the camera reference manual [5].

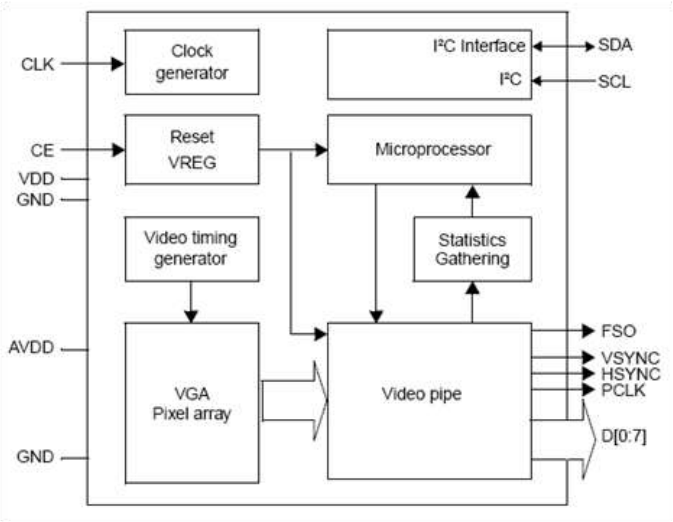

Fig. 2 – 10 Simplified block diagram of ST VS6524 camera chip

2.6.1 Initialization of the camera

As mentioned before, the camera *initialization* is carried out by configuring appropriate internal registers [5]. First of all, the camera chip has to be connected to 2,8 V voltage source, external clock source (6,5 to 26 MHz), and CE pin set HIGH [5]. Subsequently internal camera registers can be accessed through I^2C interface (pins *SDA* and *SCL*) [5]. I^2C 8-bit *device address* of the camera is permanently set to 0x20 while LSB bit is shared with the read/write flag (will be described later).

Before any commands can be sent to the camera, internal *microprocessor* must be enabled [5]. By writing value 0x06 to the register 0xC003 internal microprocessor is enabled. Subsequently, digital *I/O* pins has to be enabled by writing 0x01 to the register 0xC034. After these two registers have been configured, rest of registers can be configured.

The next step could be the setting of contrast (reg. 0x039A) to 0xA0 and switching the camera mode to *RUN* by writing 0x2 to the register 0x0180.

In this application, the image size was configured to QVGA by writing 0x2 to the register 0x0380, synchronization codes were turned off (0x00 to the reg. 0x2104), *HSYNC* synchronizes *active lines* only (0x0F to the reg. 0x2106) and clock is *free-running* (0x85

to the reg. 0x210A). Table and detailed description of all registers in the *VS6524* camera can be found in its user manual [5].

2.6.2 Output data format and synchronization

When *initialization* of the camera chip is finished, captured video data are sent to the output pins of the camera chip [5]. There are 8 data output pins available. Synchronization is carried out by monitoring *HSYNC*, *VSYNC* signals and synchronizing PLL to *PCLK* signal producing camera synchronized clock domain inside the FPGA. All output data are synchronized to *PCLK*. Data are sent to output on *falling edge* of *PCLK*. Also control signals *HSYNC* and *VSYNC* are changed on the *falling edge* of *PCLK*.

Video stream consists of frames sent one after another. There are *interframe blanking data* between each two frames [5]. Also between every two lines of frame there are *interline blanking data*. By default, blanking data contains 0x1080 values. These values can be changed in registers 0x2110 and 0x2112. Fig. 2 – 11 illustrates output frame.

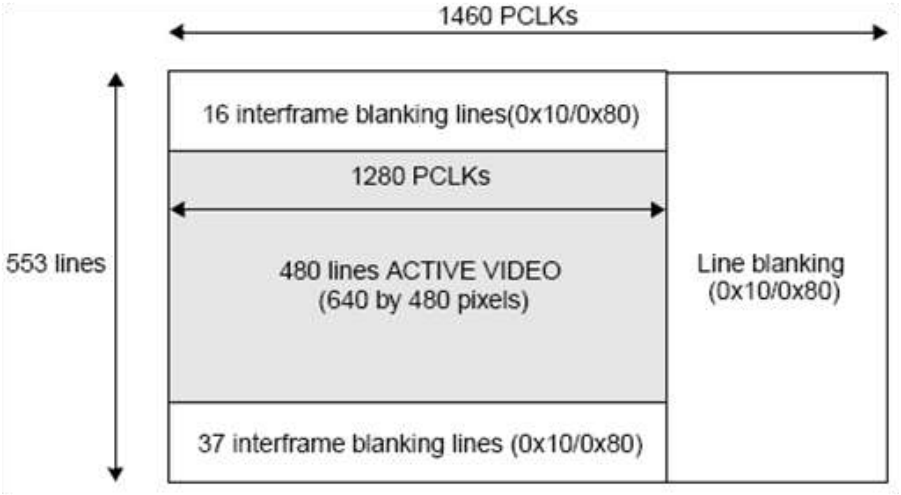

Fig. 2 – 11 VGA 30fps output frame

Each pixel is represented by 2 bytes [5]. It depends on the chosen format how these 2 bytes will be used (format can be chosen by adjusting register 0x0416). Possible choices are: *YUV 4:2:2*, *RGB 565*, *RGB 444* and *Bayer 10-bit* [5]. I have chosen *YUV 4:2:2* format, because only grayscale image was required. To obtain grayscale image, only

luminance component (Y component) has to be read from YUV stream. I have also decided to use the resolution *320x240* (half resolution). According to the requirements for further image processing, this resolution was suitable. The design can be easily adapted to a different resolution in the future. This resolution is adjusted in the camera (register 0x0400) and in the output stream every second pixel is skipped and every second line is skipped too [5]. This ensures the resolution *320x240*. Therefore one *Y* component is sent per every 4 clock periods as illustrated in the Fig. 2 – 12 to Fig. 2 – 15. Also camera was set to transmit the *Y* component first and the *C* component (chrominance) as the second. Fig. 2 – 12 to Fig. 2 – 15 illustrates the output format.

Signals *HSYNC* and *VSYNC* provide *line* and *frame* synchronization and *PCLK* provides pixel synchronization [5]. You may have noticed that every line has only 319 pixels not 320. This issue was corrected in hardware part by copying the last pixel.

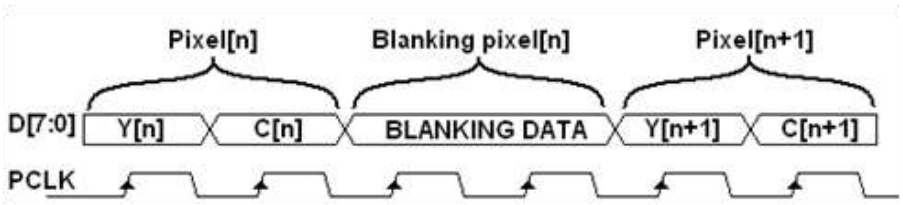

Fig. 2 – 12 n^{TH} pixel in video stream

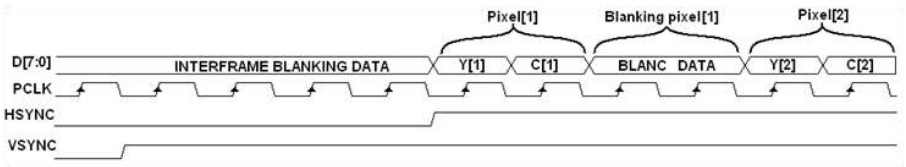

Fig. 2 – 13 Beginning of the frame

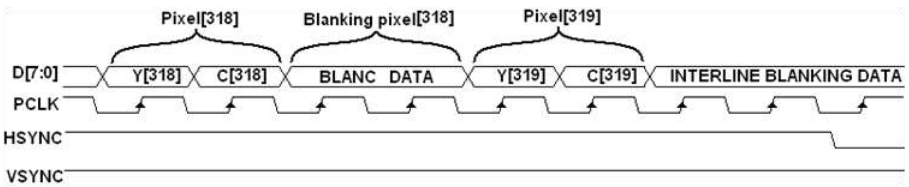

Fig. 2 – 14 End of the first line

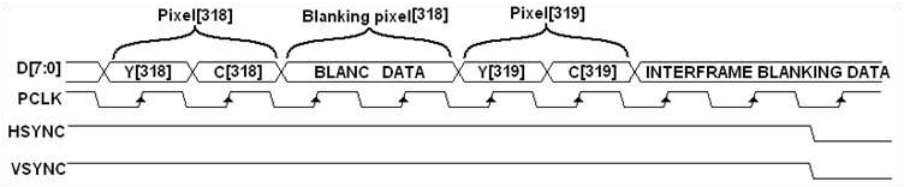

Fig. 2 – 15 End of the frame

2.6.3 I^{2}C communication description

The camera chip is controlled through I^{2}C interface [16] where it acts as I^{2}C slave peripheral with its device address 0x20 [5]. In our case, ARM7 processor acts as master device and has full control of the I^{2}C bus. According to unique states of *SDA* and *SCL*, data exchange can be established. Basic parts of data exchange are: *start*, *stop*, *acknowledge*, *not-acknowledge* commands, *device address* (containing *read/write* flag), *register address* (high and low byte) and *data part* [16], [5].

Each communication (register access) starts with start command (both *SDA* and *SCL* signals are pulled LOW) following by *8-bit device address* [16], [5]. LSB bit of the *device address* distinguishes if the next operation will be writing to the slave (0) or reading from slave (1) (in this case *device address* is 0x21). Therefore *write flag* is sent to the bus (0). Subsequently slave device confirms successful transfer by sending *acknowledge* bit. After *acknowledge* a short pause follows, after which high 8 bits of register address are sent to the slave. This *pause* is required by camera, so changes inside the chip have enough time to take effect. Slave confirms transfer by sending *acknowledge* to the master device. After *acknowledge*, a short pause follows after which low 8 bits of register address are sent to the slave. Slave confirms transfer by sending *acknowledge* to the master device, afterwards a short pause follows. The next action will be different when writing to the slave than when reading from the slave.

WRITING TO THE SLAVE: After pause 8 bits of data are sent to the slave device. Slave confirms successful transfer by sending *acknowledge* [16], [5]. Short pause follows. After pause master sends *stop* command by pulling both *SDA* and *SCL* signals to HIGH.

READING FROM THE SLAVE: After a pause, *stop* command is passed to the bus by

pulling both signals to HIGH [16], [5]. A longer pause period follows (so the slave device has enough time to prepare data to be sent to the master). It can be *100xSCLperiod* long. After this period, *start* command is sent to the bus. The device address follows with the read flag (1). Slave confirms transfer by sending *acknowledge* and a short pause follows. Subsequently the slave starts to send data to the master. It sends 8 bits of data. After transfer is completed, master sends negative *not-acknowledge* to the slave. Immediately after *acknowledge*, master sends *stop* command to force slave not to send more data [16], [5].

If access to the other register is required, the whole process has to be repeated again after a longer pause [5]. After configuring all the required registers, the camera should operate according to the expectations. To prevent incorrect configuration of the camera chip, always check the configuration by reading the content of the register after it has been modified. Fig. 2 – 16 and Fig. 2 – 17 illustrate write and read operation with the register of address 0xC003.

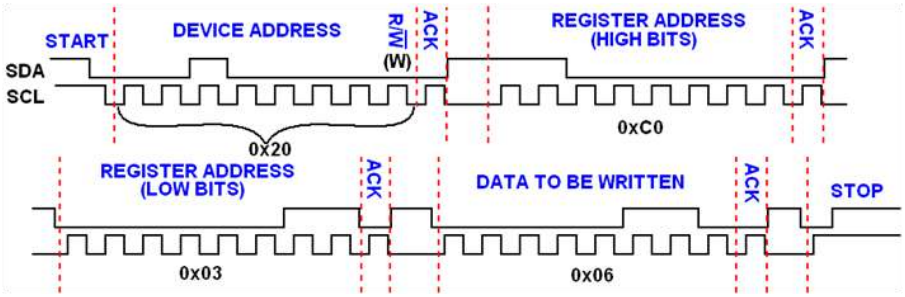

Fig. 2 – 16 Write operation with the internal register 0xC003

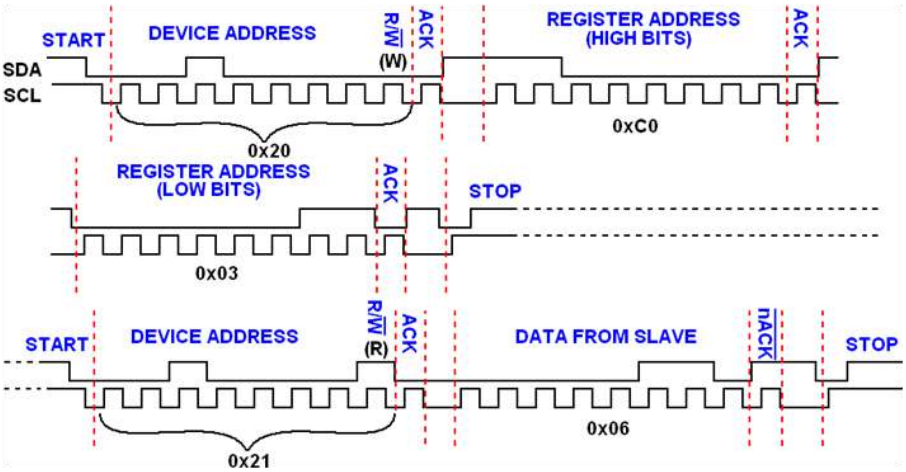

Fig. 2 – 17 Read operation with the internal register 0xC003

3 Description of the software development tools

The quality of special design environments with development tools is essential during the design stage. The absence of debugging capabilities, syntax highlighting or possibility to change unpractical eye-tiring background color can slow down the creative process and lead to user's exhaustion. Therefore a good development environment becomes essential.

During the design of the particular project,the following development environments were used:

- Libero IDE from Actel Corporation, version 8.4 [17] with SP1
- ModelSim from Mentor Graphics Corporation, version 6.3g [18]
- μVision3 from Keil Corporation, version 3.62c [19]
- Borland C++ Builder, version 6.0 [20]

3.1 Libero IDE

Libero IDE represents a special development environment created by Actel Corporation [17]. The user deals with this environment during all stages of VHDL design. Libero IDE is illustrated in the Fig. 3 – 1.

Design flow with the Libero IDE consists of the following steps:

- Creation the VHDL code or generation by wizards
- Synthesis of the VHDL code by Synplify tool from Synplicity, Inc.
- Placement and Routing by Actel Designer tool [22]
- Generation of the configuration file and back annotation for timing analysis
- Configuration of the FPGA with the Actel FlashPro tool [23]

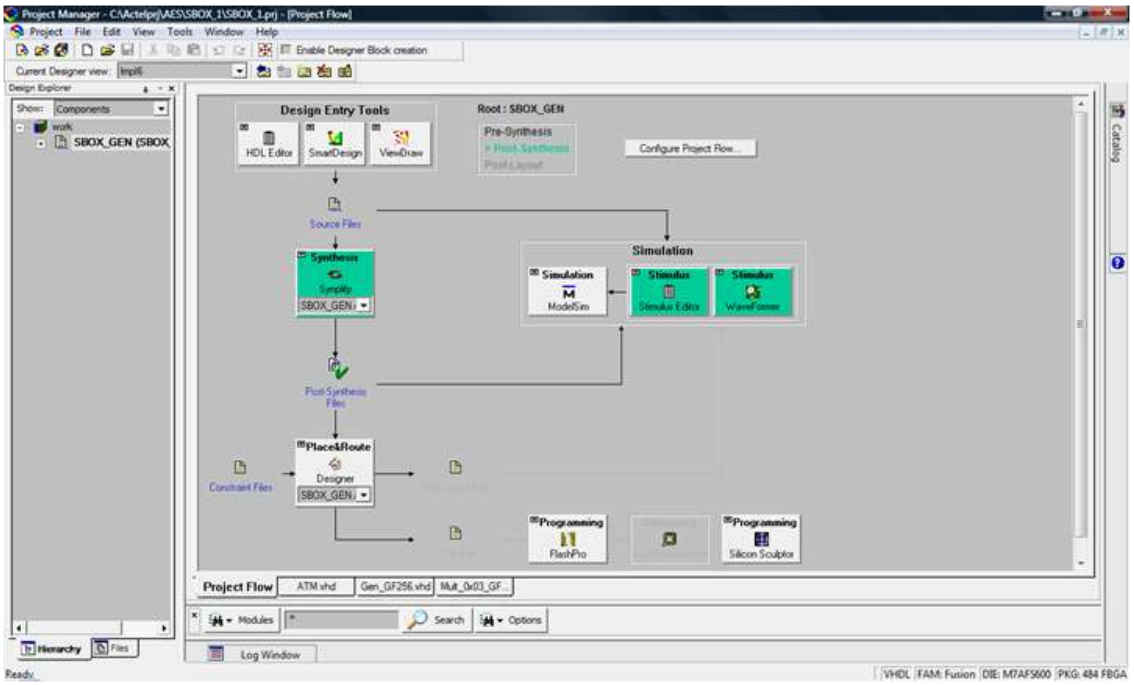

Fig. 3 – 1 Libero IDE

At the beginning VHDL structure has to be created. User can choose whether to write his own entities, or to take an advantage of generating the entities. When writing the code, user can insert some specific code structures from templates. For including ARM7 processor [12] with its peripherals to the design, CoreConsole tool [27] has to be used. CoreConsole represents powerful tool for ARM7 inclusion in the design together with its buses, peripherals and other AMBA compliant [15] IP peripheral blocks. If ARM7 is not suitable for the application, other processor can be used (ARM Cortex, 8051, ABC, etc.).

After the VHDL design is created, it has to be synthesized with respect to FPGA resources. Synthesis is carried out by Synplify tool [21]. During synthesis, combinatorial and sequential logic minimization is carried out. At the end, user is provided with the summary of the synthesis. Another synthesis with compatibility check is performed in the Designer tool (button Compile).

Before configuring the FPGA, all FPGA resources used in the design have to be assigned to a special location of the FPGA. This stage is called Placement. When all re-

sources are placed, network of interconnections between resources has to be built up. This stage is called Routing. There are several options available to further optimize the Placement and Routing. The last stage is to create the configuration file. Back annotation can also be generated for use in Timing analysis. Placement, Routing, configuration file generation and back annotation generation operations are carried out by Actel Designer tool (Layout button). The Designer tool [22] is illustrated in the Fig. 3 – 2.

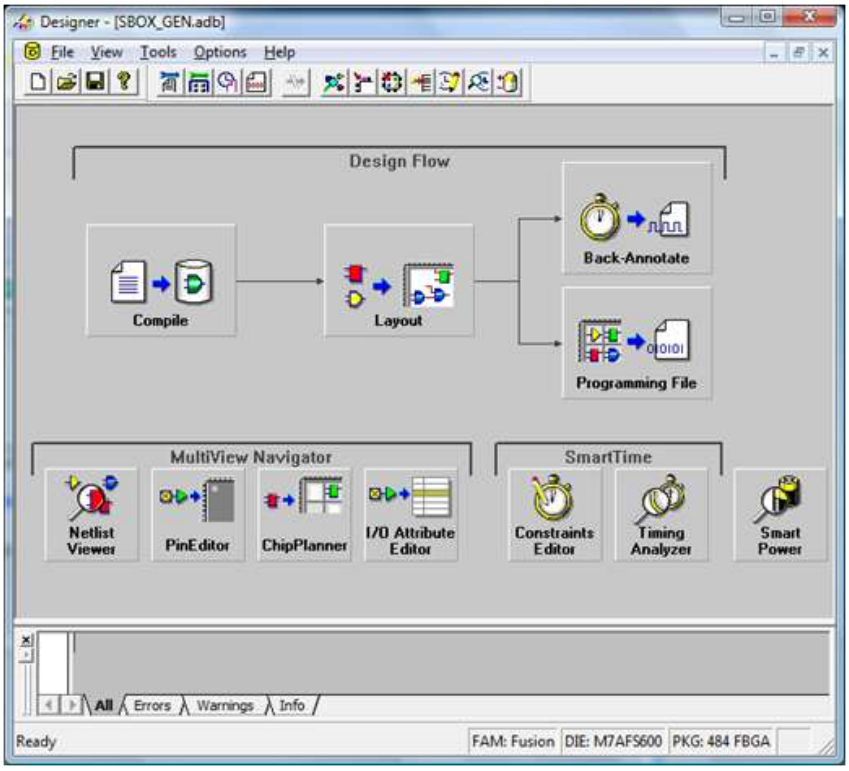

Fig. 3 – 2 Actel Designer tool

The last stage of the design is the Configuration of the FPGA with FlashPro tool [23]. This tool communicates with the FlashPro3 device which transfers the configuration from the PC to the FPGA via the JTAG interface. FlashPro tool allows the user to configure separately the flash array and the internal flash memory. After the configuration is finished, FlashPro performs configuration check.

Before the design is configured to the FPGA, several simulations on the PC are available. All simulations are performed by ModelSim [18]. When starting simulation, Libero

IDE generates batch file *.do* and executes it in the ModelSim. The following simulations are available:

- Pre-synthesis
- Post-synthesis
- Post-layout

Pre-synthesis simulation can be carried out before the VHDL code is synthesized. This simulation is only functional. Post-synthesis simulation is carried out on the synthesized VHDL code. Although the simulation is still functional, it can be useful when comparing the impact of the synthesis with the original design. Post-layout simulation represents the timing simulation of the design and can be performed after the back annotation was created [17], [22], [18]. This type of simulation is very critical when estimating the limits of the design (maximal frequency, delays, glitches and other unusual behavior).

Maximum frequency and delays between gates can be also estimated by SmartTime tool (part of the Designer tool) [24], [22]. Power consumption can be estimated by the SmartPower tool (part of the Designer tool) [25], [22].

3.2 ModelSim

One of the most important issues during the design period is when the design doesn't behave properly after being configured into the FPGA. The method for correction of the issue is called debugging. However, localization of the issue can be very difficult and time consuming. The best way to localize difficulties is to run a simulation. There are two common types of the simulation:

- functional simulation
- timing simulation

Functional simulation is used to observe waveforms of the signals without respect to placement, routing and behavior of the FPGA. Timing simulation respects placement and

routing and all characteristics of the targeted FPGA. To perform timing simulation, *.sdf* file [22], has to be imported into the ModelSim [18].

One of the most commonly used tools for VHDL simulation is ModelSim [18]. Before the simulation can be performed, VHDL testbench has to be created. Afterwards design with testbench is imported to the ModelSim. All used libraries have to be imported too. Once imported, design has to be compiled. After the compilation is performed, simulation can be executed. The designer has to specify the duration of the simulation. If the design is very complex, computation of the simulation take very long.

Results of the simulation are displayed in the Waveform window (see Fig. 3 – 3) [18]. According to the complexity of the project, it is advantageous to read input values from the text file and store the results of the simulation. Special warning messages or error messages can be conditionally displayed in the transcript window.

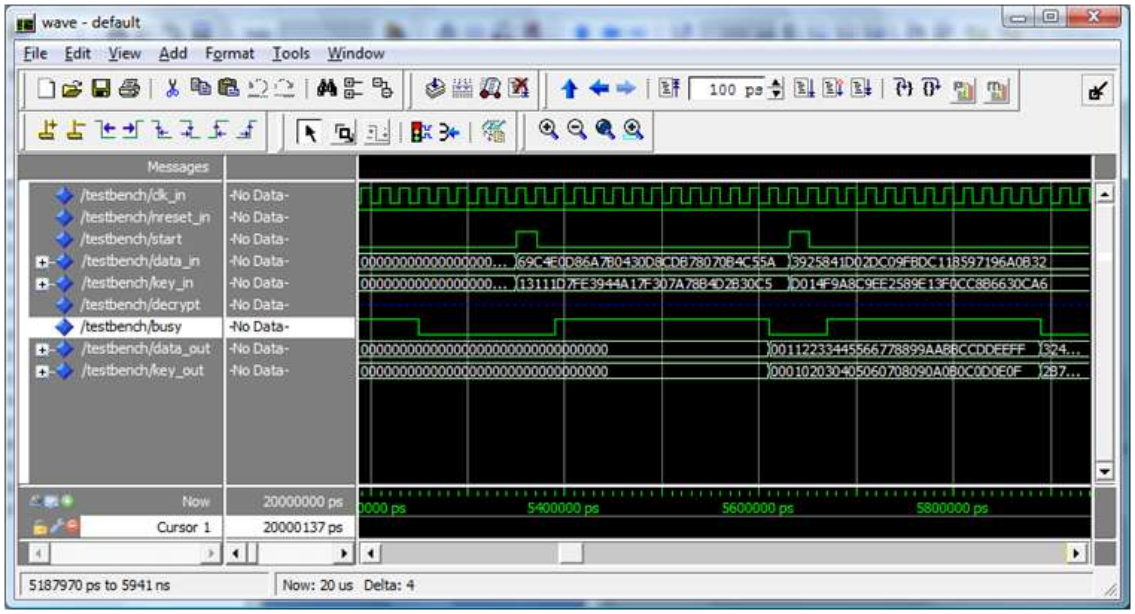

Fig. 3 – 3 Waveform view in ModelSim

Furthermore, the whole simulation process can be automated by creating .do batch file [18]. File can be executed inside the Modelsim command line by the command *do filename.do*.

3.3 Keil μVision 3

After the processor is put into the design, the source code has to be created. One of the best firmware development tools is Keil μVision 3 [19]. This professional development tool allows users to write program in c code or assembler code, simulate the code, generate programming Intel-Hex file and debug the code directly in hardware if the hardware is compatible with μVision 3. Keil μVision 3 main window is illustrated in the Fig. 3 – 4. Since the original Keil μVision 3 compiler is not compatible with Actel CoreMP7 (de-

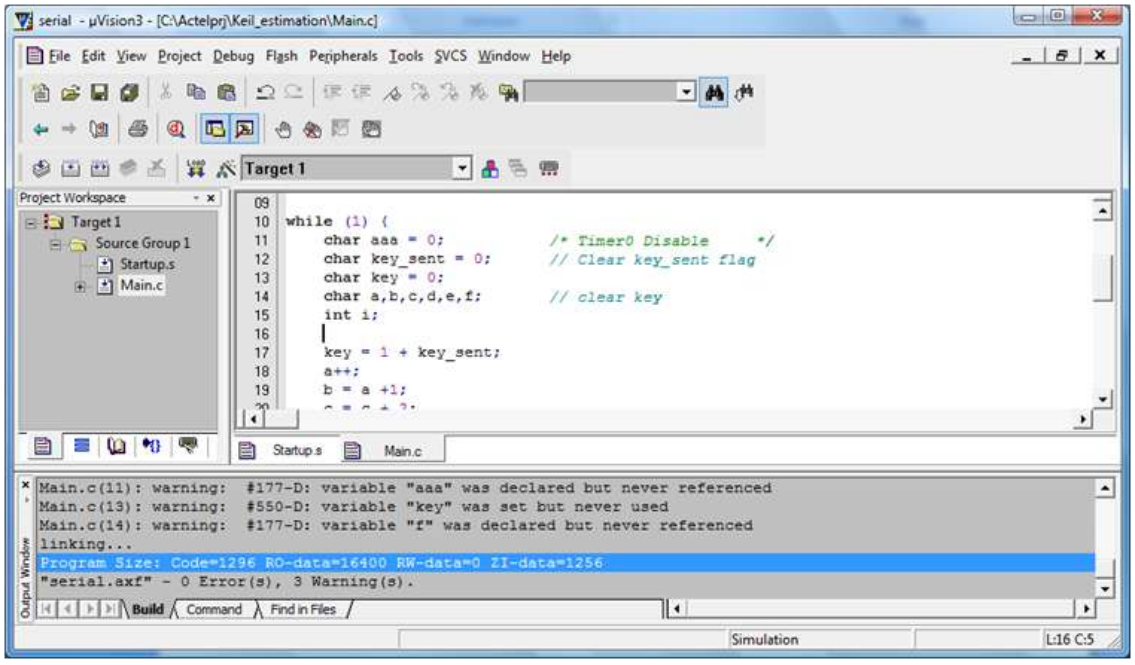

Fig. 3 – 4 Keil μVision 3 development tool

rived from ARM7) external GNU Sourcery G++ compiler had to be used. GNU debugger hasn't been adapted for μVision 3 yet, however Actel SoftConsole application [26] can be used for debugging instead.

After the firmware is written, Intel-Hex file is generated. This file is subsequently imported into the Flash memory system builder in Actel Libero IDE [17], [23]. Afterwards, the firmware is transferred into the internal flash memory of the FPGA during the configuration stage.

3.4 C++ design environment (Borland C++ Builder)

After the system is developed, a communication link has to be established with the PC via the USB. To control the transfer over the USB line, computer application has to be programmed. This application should be user-friendly and needs to be able to communicate with the USB driver.

One of the easiest ways to program application containing GUI is by using Borland C++ Builder [20]. This development tool is illustrated in the Fig. 3 – 5

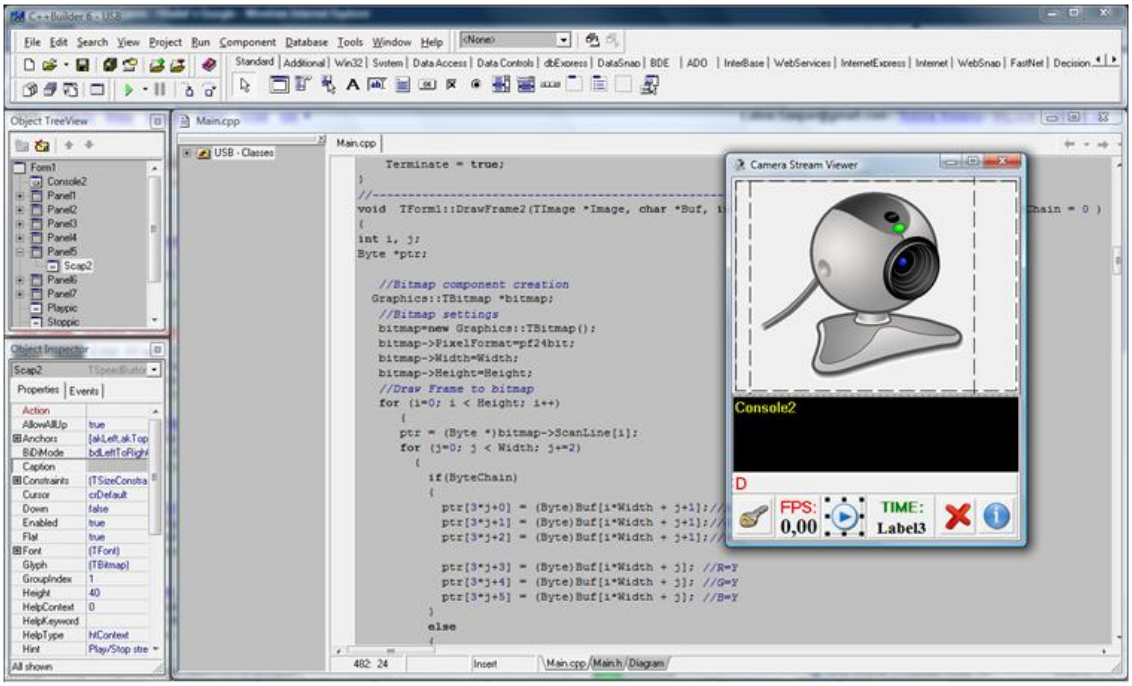

Fig. 3 – 5 PC software development environment Borland C++ Builder

When a new project is created, an empty window is displayed. The user has to insert all the objects like buttons, images, labels, combo boxes, etc. [20]. To create more attractive application, object properties have to be tunned. After the visual part of the application is created, functionality has to be programmed using C++ code. Every object in the application is represented by a class in the hierarchy of classes. The class at the bottom of a hierarchy inherits all attributes and properties of the above classes in the hierarchy.

Before the application is compiled, the compiler has to be set to retail mode. In this mode *.exe* file will be created without linking debug libraries [20]. And what is more, special settings have to be specified, so resulting application will not depend on Borland C++ Builder libraries, but only on standard windows libraries.

4 Structure of the image acquisition subsystem

4.1 Interconnection of the development boards

This chapter describes the structure of the image acquisition subsystem. During the development phase, the system was composed of several development boards. The structure of the system with its interconnected subparts is illustrated in the Fig. 4 – 1.

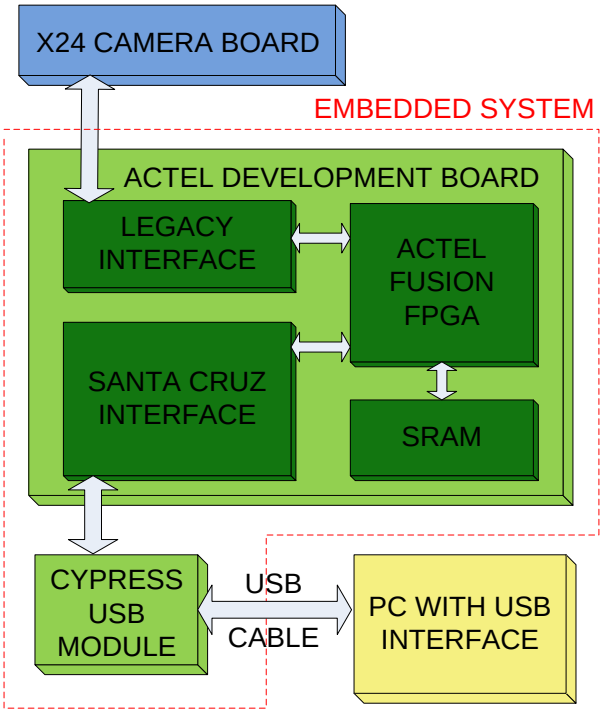

Fig. 4 – 1 Interconnection of development boards forming the system

The system is composed of the embedded system (ES) and the camera [5] (placed in the X24 camera evaluation board). During the acquisition, images are transferred from the camera to the ES where image processing has to take place. The control of the operation by a PC, as well as data exchange, is provided via USB interface.

USB communication is established via the Cypress USB module [10] placed in the Santa Cruz interface located on the Actel System Management board [7]. Essential part of the ES is the System on Chip (SoC) based on the ARM7TDMI processor [12], [13].

4.2 System on chip based on the ARM7 processor

The SoC is instantiated inside the Actel Fusion FPGA (as shown in the Fig. 4 – 2).

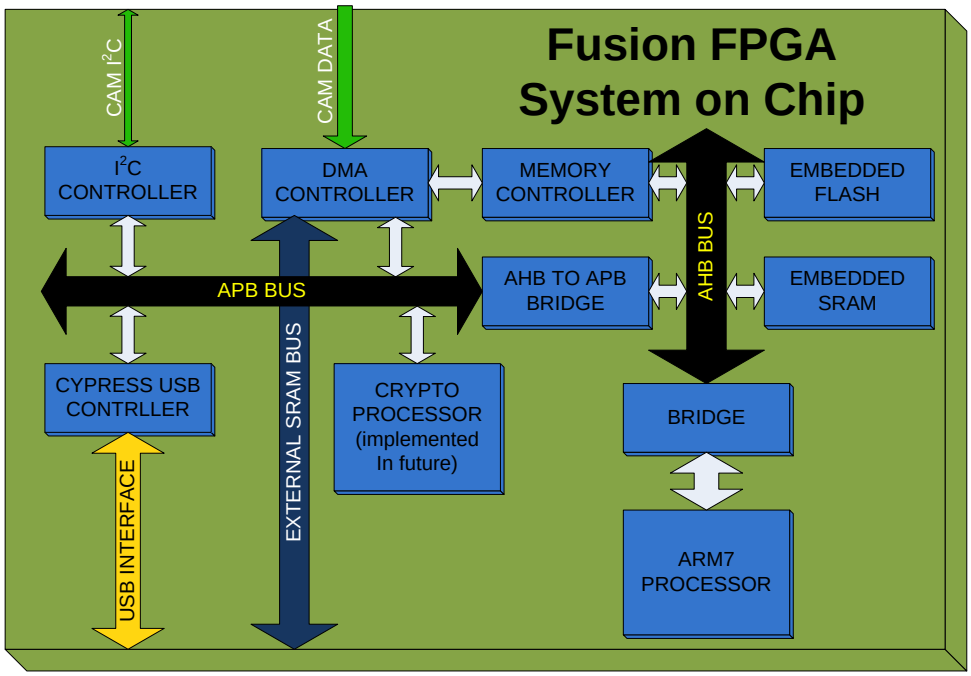

Fig. 4 – 2 System on Chip based on ARM7 processor

The main part of the SoC is the ARM7 processor [12], [13]. The processor represents the main control unit of the SoC. AHB and APB buses represent convenient solution for interconnecting peripherals with the ARM7. Bandwidth critical peripherals (i.e. SRAM block) have to be connected to the AHB bus. In contrast, peripherals with lower bandwidth demands, can benefit from the simplicity of the APB protocol.

The program for ARM7 processor is stored in an embedded flash memory. The embedded SRAM is used for storing temporary results of the calculation. For interfacing with an external SRAM memory, a memory controller block is necessary. Operation of the camera is controlled via the I^2C interface. If more than one device requires access to the external memory, DMA controller has to be used. In this case external memory is shared by the image acquisition system (part of the DMA controller) and the ARM7's memory controller. Cypress USB controller block enables ARM7 to communicate with

the PC via Cypress USB module [10].

All data traffic before being transmitted out of the FPGA has to be encrypted. For this purpose, cryptoprocessor has to be included in the design in the future.

4.3 Image acquisition

Once the image acquisition is initiated, image data are sent from the external camera into the FPGA. Inside the ES, DMA controller redirects the traffic outside the FPGA into the external SRAM memory. Before a 32-bit word can be transferred into the external SRAM memory, it has to be concatenated from a four 8-bit words (each pixel is encoded in one byte). This concatenation is performed in 8/32-bit converter (concatenator) block.

Since incoming data and control signals from the camera have to be synchronized, the design contains another PLL, which is synchronized to the external clock signal from the camera. Afterwards, all control signals are registered within this clock domain.

Before the data can be transferred into the external SRAM memory, an address has to be generated. For this purpose Address Generator is included in the project. This generator is initialized by the controller, which communicates with the ARM7 processor via the APB bus. Since ARM7 is the master of the DMA controller, acquisition of the image is initiated by ARM7 by switching multiplexers in the DMA controller towards the acquisition part. If the acquisition is not performed, multiplexers are switched towards the memory controller, thus ARM7 is capable of accessing external SRAM memory. The DMA controller is illustrated in the Fig. 4 – 3. Zones enclosed within the red line belong to the ARM7's clock domain and zones enclosed within the green line belong to the camera synchronized clock domain.

I have chosen this solution as the only feasible one. If the data were acquired by the ARM7 processor and subsequently transferred into the memory, it could consume all processor time and no other control and processing would be possible.

State diagram of the main image acquisition state machine is illustrated in the Fig. 4 – 4. After the acquisition is initiated by ARM7, state machine abandons IDLE state and

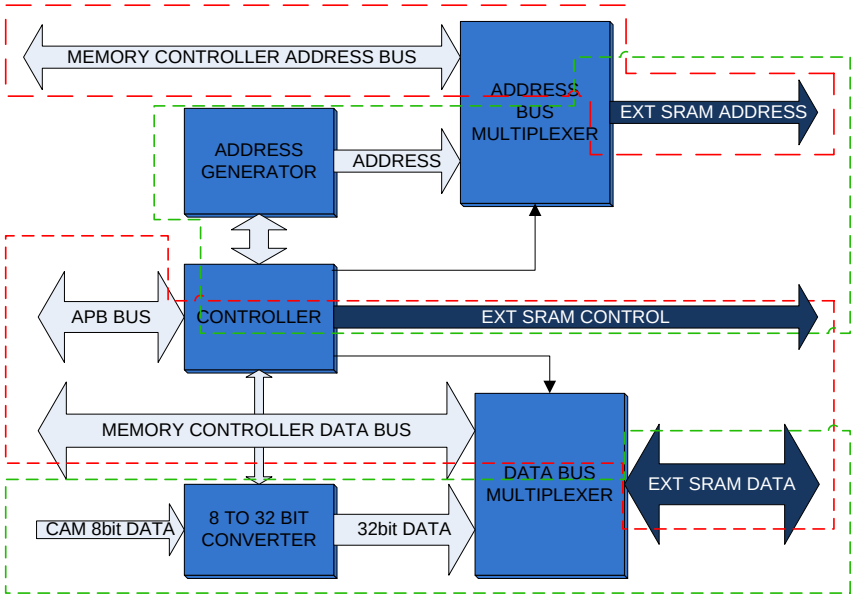

Fig. 4 – 3 Illustration of the DMA controller with two clock domains

waits until vertical image synchronization signal (VSYNC) is LOW. In the next state, state machine waits until vertical image synchronization signal changes to HIGH. If it happens it signalizes the beginning of the image. State machine immediately enters capture state where an acquisition is performed.

After the image is acquired, VSYNC signal changes to LOW and state machine enters IDLE state. Status signals indicate successfully performed acquisition. ARM7's task is to monitor these signals.

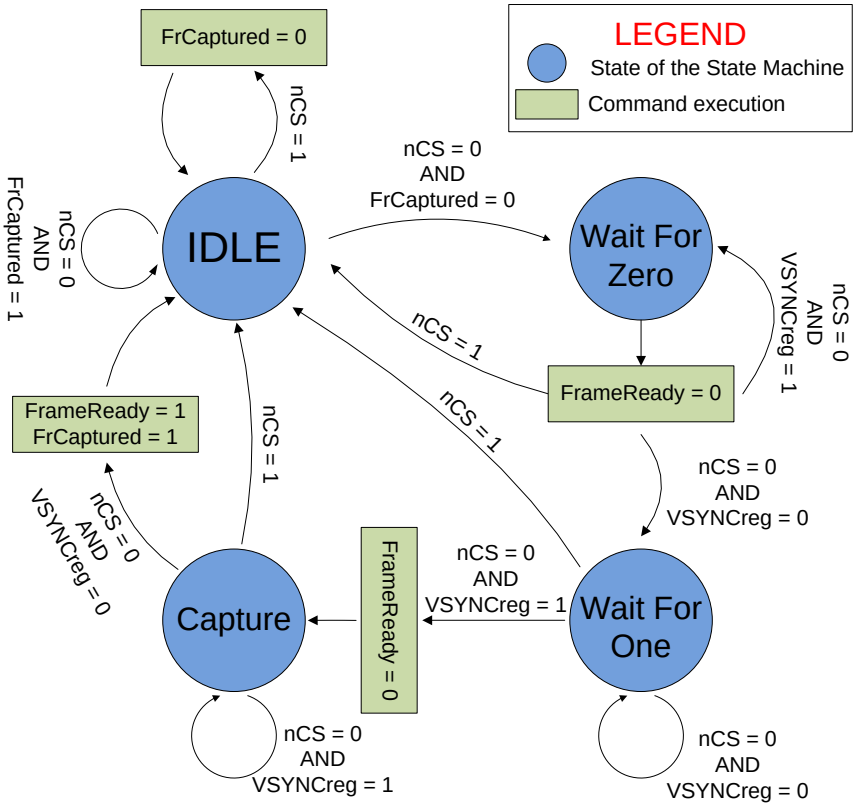

Fig. 4 – 4 State diagram of the main image acquisition state machine

5 Optical signature extraction algorithm

One of the most important steps during secured traceability of the product is the comparison of the optical signature of the product with the optical signature of the original product. It is very important to evaluate optical signatures of two scanned images as **different** when these images were printed in **two different printers**. If images were printed with **the same printer**, optical signatures of the scans have to be evaluated as **matching**. Algorithm for the optical signature extraction and subsequent comparison with the optical signature of the original product is described in next section.

5.1 Description of the image processing, signature extraction and comparison algorithm

Before the algorithm can be applied, special uncopyable marker has to be recognized in one of the acquired images. Afterwards, image has to be processed to be suitable for extraction of the signature. Example of the marker is shown in the Fig. 5 – 1.

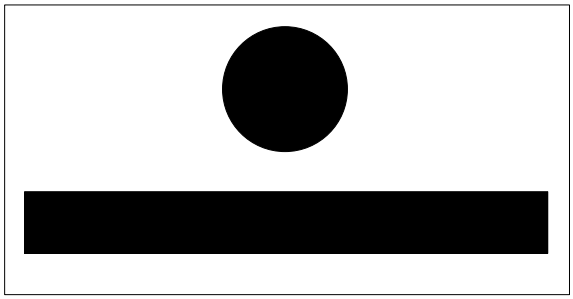

Fig. 5 – 1 Special uncopyable marker for optical signature extraction

After the marker is found in the image, image processing operations are performed as follows:

1. Computation of optimal global image threshold using Otsu's method
2. Binarization of the image
3. Application of 2D median filter

4. Localization of the circle and rectangle objects in the image
5. Compututation of the angle between horizontal plain and rectangle's longer edge
6. Rotation of the image
7. Trimming of the image to contain only circle object
8. Move of the circle object to the center of the image

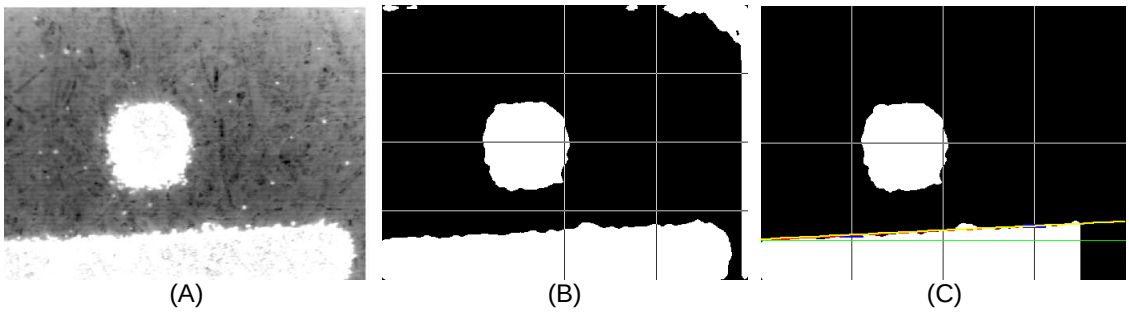

Fig. 5 – 2 A) Scan of original print, B) Binarized and filtered, C) Angle estimation

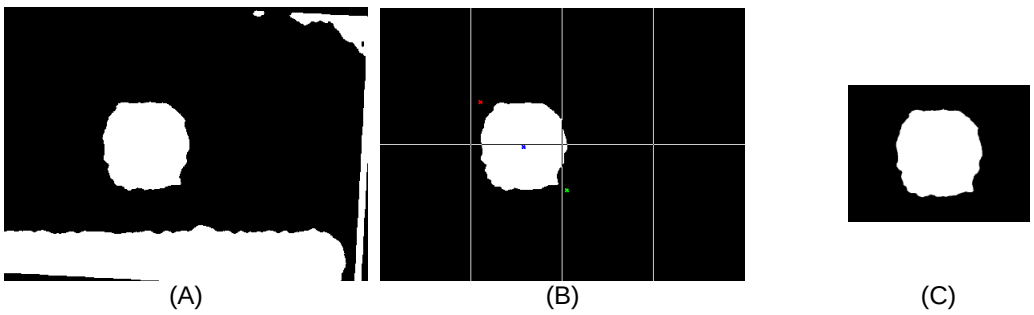

Fig. 5 – 3 A) Rotated, B) Dot localized, C) Centered and trimmed

Examples of image processing steps applied on the scan of original print are shown in the Fig. 5 – 2 and Fig. 5 – 3.

After the image is processed, optical signature can be extracted. For this purpose, computation of the Freeman code is favorable. Freeman code describes an outline of the object. At the beginning, the first pixel of the object is located. Subsequently the next pixels in the edge are located. Each pixel is described as one of eight possible directions

from the previous pixel in the edge. Algorithm is finished when all of the pixels in the edge are described by the Freeman code.

As soon as the optical signature is extracted, it has to be compared with the signature of the original product acquired from the encrypted data area on the surface (i.e. encrypted data in barcode). The best way to compare two Freeman codes with different lengths is to compute the Levenshtein distance. Big advantage of the Levenshtein distance is the resistance to different lengths of the compared Freeman codes.

Fig. 5 – 4 Original print: scan A and scan B

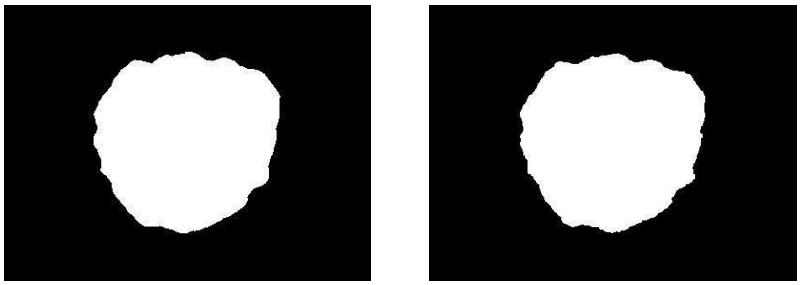

Fig. 5 – 5 Fake print: scan A and scan B

Example is illustrated in the Fig. 5 – 4 and Fig. 5 – 5. Two scanns of the original image are shown in the Fig. 5 – 4. Two scanns of the fake image are shown in the Fig. 5 – 5. Levenshtein distances of the different prints and their scans are summarized in the Tab. 5 – 1. As shown in the table, Levenshtein distances between scans of original prints and scans of fake prints are all over 200. In contrary, Levenshtein distances between two scans of the same print are less than 150. It is important to note that images were scanned in laboratory conditions, therefore Levenshtein distances are very different and it is easy to

Tab. 5 – 1 Levenshtein distance of different prints and corresponding scans

		Original print		Fake print	
		Scan A	Scan B	Scan A	Scan B
Original print	Scan A	0	140	222	231
	Scan B	140	0	220	222
Fake print	Scan A	222	220	0	134
	Scan B	231	222	134	0

recognize the original from fake one. Recognition can be a questionable issue when light conditions are worse and the difference between the original and fake ones are smaller.

5.2 Hardware implementation of image processing, signature extraction and comparison algorithms

One of the system's constraints was the acquisition speed of at least 4 frames per second. Therefore, system has to be able to capture the image, carry out the image processing, extract a Freeman code, compute the Levenshtein distance and decide whether the product is original or fake before the next image is captured. This gives the system approximately 250 ms to perform all the mentioned steps.

As explained in the previous section, algorithm is very robust. There are three possible solutions for computation of algorithm:

- in ARM7 processor
- in coprocessor connected to ARM7
- part in the coprocessor and part in the ARM7

However, algorithm has a serial structure, thus parallelization can become an issue. This

limits the advantage of the FPGA which is its parallelism. And what is more, some operations are carried out on every pixel of the image, or small blocks of the image. These operations are very time consuming.

The whole algorithm has been developed in Matlab. The goal of this work is to estimate the number of ARM7 cycles required to carry out all operations of the algorithm and to estimate the resource count in case of hardware implementation. As long as used functions in Matlab haven't been written in the c-code yet, cycle accurate estimation in Keil μVision 3 couldn't have been performed. Estimation issue was even greater when an operation of the algorithm used several robust Matlab functions. Due to these issues and lack of time, in some cases I wasn't able to estimate required number of cycles. These estimations have to be carried out in greater detail in the future.

Tab. 5 – 2 summarizes the estimations for the particular operations of the algorithm. Size of the image is 320x240 pixels. Operations that couldn't have been estimated are denoted as N/A (Not Available). However, the provided estimations can be used to predict the complexity of calculations. However, only three estimated operations require about 10 000 000 clock cycles. If the processor operated in the frequency of 40 MHz and the algorithm consisted only from these three estimated operations, processing of the image would take 250 ms. These estimations have to be proved by simulations in the future.

Tab. 5 – 2 Basic estimation of required ARM7 cycles as well as required logic resources for implementation of particular operations within the algorithm.

	ARM7 [cycles]	VHDL [tiles]
Computation of global image threshold	1 200 000	N/A
Binarization of the image	1 152 000	N/A
Application of 2D median filter	N/A	120
Localization of circular and rectangular objects	N/A	N/A
Computation of rotation angle	N/A	N/A
Rotation of the image	N/A	N/A
Trimming of the image + relocation to the center	N/A	N/A
Computation of the Freeman code	60 000	N/A
Computation of the Levenshtein distance	8 000 000	N/A
Total	**N/A**	**N/A**

6 Analysis of the cryptoprocessor core

Every data before being transferred out the FPGA have to be encrypted. Strong encryption will assure perfect protection of confidential data against attacks from outside. When data enters the FPGA from outside, they have to be authenticated too. As long as big amounts of data are processed in 128-bit blocks, a block cipher mode have to be used. According to NIST specifications, the following block cipher modes are recommended:

- Confidentiality modes: ECB, CBC, CFB, OFB, CTR [28]
- Authentication mode: CMAC [29]
- Authenticated encryption mode: CCM [30]
- High-throughput authenticated encryption mode: GCM [31]

One of the most flexible ways to implement these block modes is to build a cryptoprocessor [4]. The main part of the cryptoprocessor is the encryptor/decryptor block. Other operations required by a block mode can be performed in special ALU and intermediate results can be temporarily stored in internal registers. To sustain security of the system, it is preferable to implement two sets of registers: Data registers and key registers. Key registers are special registers where main key as well as session keys are stored. These key registers have to be inaccessible from outside of the cryptoprocessor [4].

6.1 AES Encryption/Decryption module

Main part of the cryptoprocessor is the AES module for encryption and decryption. AES is a new encryption standard approved by NIST as FIPS-197 [1]. Robustness of the algorithm assures sufficient security of the confidential data after being encrypted. AES is symmetric cipher. Size of the plaintext is limited to 128 bits and size of the key can be 128, 192 or 256 bits. While sufficient security can be assured with 128-bit key, AES module used in this project works only with 128-bit. If needed, the project can be expanded in the future.

AES algorithm is divided into 10 rounds, while the following operations can be found in individual rounds [1]:

- Substitute bytes
- Shift rows
- Mix columns
- Add round key

When decryption is performed, inverse operations are carried out [1]:

- Inverse shift rows
- Inverse substitute bytes
- Add round key
- Inverse mix columns

While the key length is only 128 bits, key has to be expanded [1]. After expansion ten new 128-bit round keys are generated. For more detailed description refer to AES standard FIPS-197 [1].

While steps of the cipher (later denoted as direct cipher) and decipher (later denoted as inverse cipher) are reversed to one another, implementation of encryptor and decryptor consumes many logic resources. If some modifications are made (some operations in inverse cipher are swapped) resource sharing between encryptor and decryptor is possible [1].

6.1.1 Implementation of the shared AES encryptor/decryptor core

Implementation of shared AES Encryptor/Decryptor core is shown in the Fig. 6 – 1. User has a possibility to select Encryptor, Decryptor, or shared Encryptor/Decryptor core. Red components in the Fig. 6 – 1 are Encryptor core specific and green components are Decryptor core specific. Black components are used in all cores. As shown in the Fig. 6 – 1

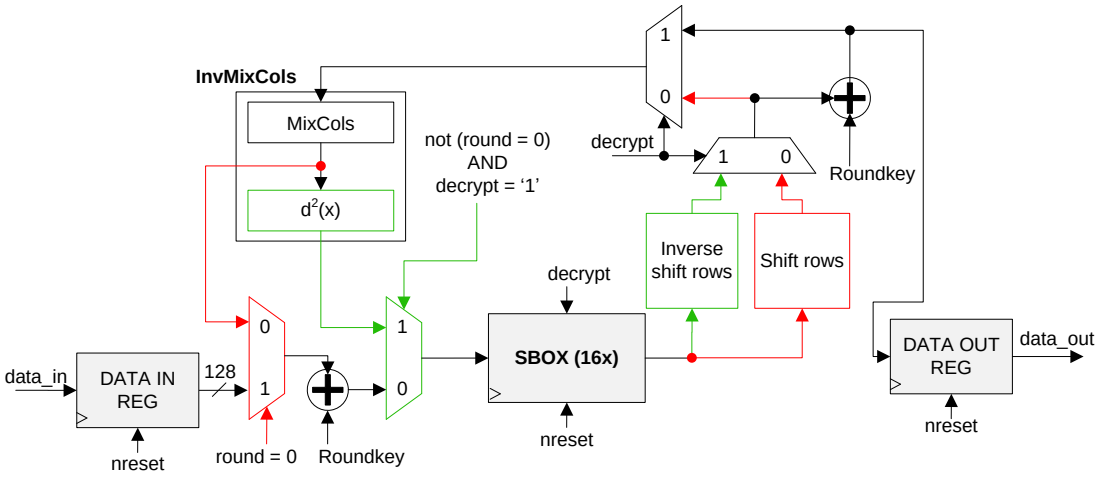

Fig. 6 – 1 Shares AES Encryption/Decryption core

the following logic parts are shared:

1. Add round key A – Operation of XORing data with round key. It is implemented after round zero multiplexer.

   - Encryption:
     - o Round 0: XORing input data with original (0^{TH}) round key
     - o Rounds 1 to 9: XORing output of MixCols with round key
     - o Round 10: not used, Add round key B used instead (reason: key has to be added before MixCols operation)
   - Decryption:
     - o Round 0: XORing input data with original (0^{TH}) round key
     - o Rounds 1 to 10: not used

2. S-boxes – implemented as LUTs in embedded dual port memories. Each S-box is switchable between being direct or inverse. Each embedded memory block can operate as two direct S-boxes or two indirect S-boxes [3]. S-boxes will be discussed in section 6.1.2 in more detail.

3. Add round key B – Operation of XORing data with round key. It is implemented after Shift rows multiplexer.

   - Encryption:
     - Round 0 to 9: not used
     - Round 10: XORing data from shift rows multiplexer with last round key before registration and output
   - Decryption:
     - Round 0: not used, Add round key A used instead
     - Round 1 to 10: XORring data from shift rows multiplexer with all round keys except original key.

4. InvMixColumn – MixColumn block is shared by both encryption and decryption. InvMixColumn was implemented according to V. Fischer & M. Drutarovský proposed serial decomposition of InvMixCols [32].

5. Data_in_reg, Data_out, Key_in_reg, Key_out registers are shared

In the Fig. 6 – 2 implementation of the key expansion block is shown. In key expansion block, many resources are shared between forward (used during encryption) and backward (used during decryption) expansion of the key. RCON and Inv_RCON are generated by the RCON generator illustrated in the Fig. 6 – 3.

6.1.2 Implementation of S-boxes

One of the most important parts of the AES cipher are S-boxes. S-box represents complex non-linear byte transformation [1], therefore it is difficult to implement it. Byte substitution according to FIPS-197 [1] composes of the following operations:

1. Computation of the multiplicative inverse in GF(2^8)

2. Affine transformation [1]

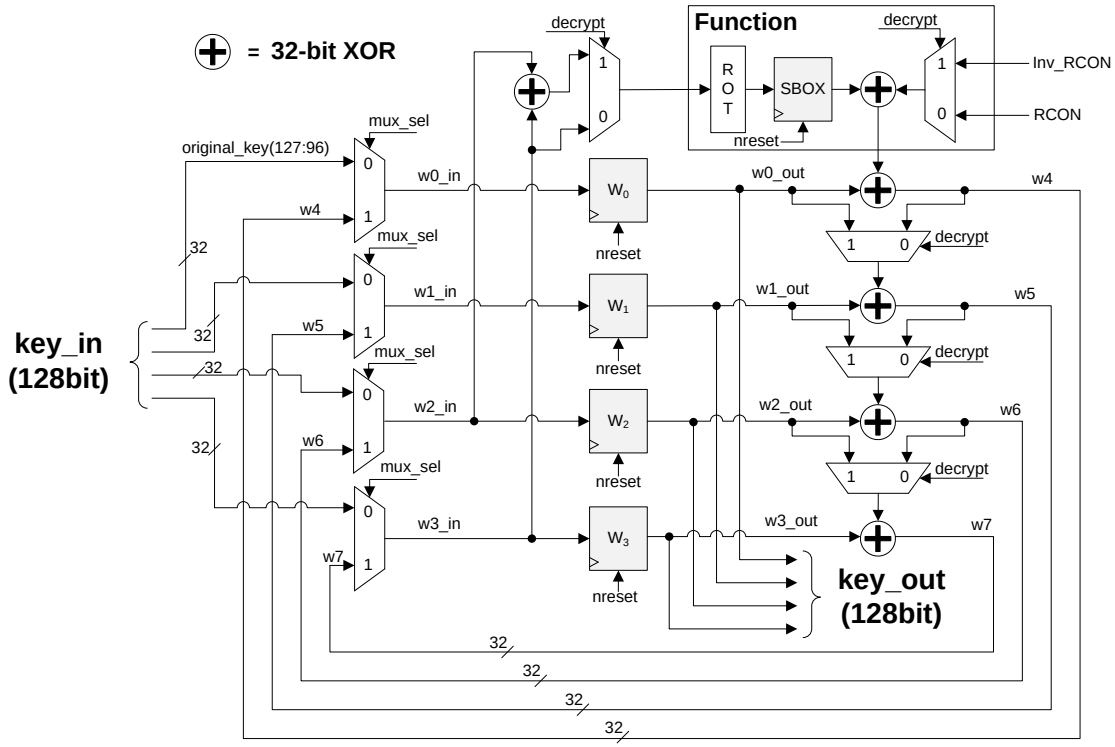

Fig. 6 – 2 Key expansion block for forward and backward expansion

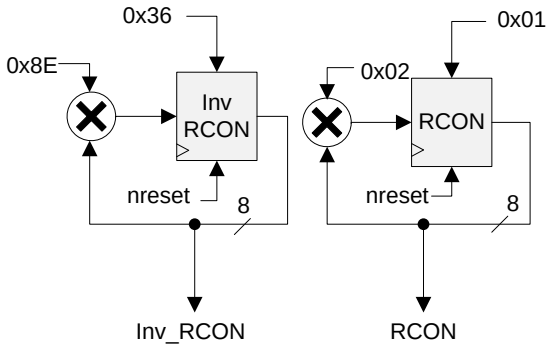

Fig. 6 – 3 RCON and Inv_RCON constants generator

Most usual implementations are [3]:

- Decomposition of GF(2^8) and mapping into two GF(2^4) – implemented in combinatorial logic
- Look-Up Table (LUT) – implemented in embedded dual port RAM

The decomposition method [33] is preferably used in ASIC circuits, where implementation of RAM memory is more expensive [3]. In contrary, LUT implementation is used mainly in FPGAs where RAM memories are available and logic resources can be utilized for the rest of the AES core [3]. For this reason, LUT implementation was chosen for the particular project [3].

6.1.2.1 LUT based S-box implementation

Before the LUT based S-box can be used, LUT has to be created inside the embedded RAM. In case of volatile FPGAs (Altera, Xilinx), LUT is created during initial configuration of the FPGA. In case of non-volatile FPGAs, the configuration stage during initialization is omitted. However, LUT has to be created inside the memory. For this purpose, S-box generator has to be implemented in logic, thus memory can be initialized immediately after power-up [3]. However, Actel Fusion FPGA contains true dual port RAMs (TDPRAM) [2]. With TDPRAM both S-box and S-box^{-1} can be generated and saved in TDPRAM simultaneously. And what is more, TDPRAM can be used as two S-boxes operating simultaneously [3]. Combination of S-box generator and TDPRAM is illustrated in the Fig. 6 – 4. One idea of generating addresses and corresponding S-box values is to

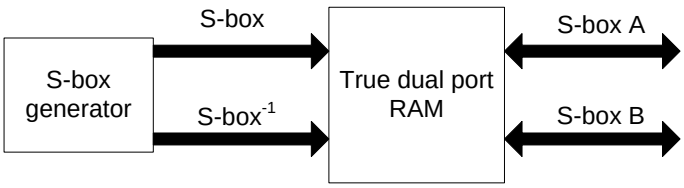

Fig. 6 – 4 S-box and S-box^{-1} generation and connection to TDPRAM

implement one combinatorial S-box and source it with counter [3]. Counter value can be considered as the memory address and output of the combinatorial S-box as the value to be written to memory [3]. After 256 cycles, whole LUT is created in the memory [3]. However, implementation of combinatorial S-box utilizes about 200 tiles [3].

However, more convenient solution based on LFSRs exists and will be described in the next section.

6.1.2.1.1 S-box generator based on LFSRs

One of the design constraints was the size of the S-box generator. Number of used resources of the generator is very important when implementing in smaller FPGAs. Therefore, S-box generator based on two LFSRs was designed [3].

Before the S-box pairs (address and value) could be created, multiplicative inversed elements have to be generated simultaneously in LFSRs [3]. Implementation of both LFSRs [3] is shown in the Fig. 6 – 5. Due to the limited number of pages of this thesis, the mathematical description of the LFSRs will not be discussed and only main results will be presented. Mathematical description is explained in article, which was submitted to FPL'09 conference [3] and CryptArchi workshop [3].

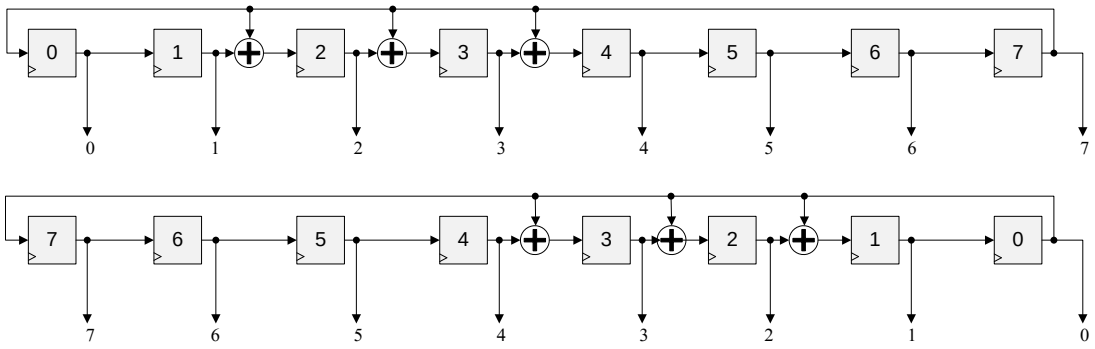

Fig. 6 – 5 LFSRs for generation of multiplicative inversed pairs

Unfortunately, GF(2^8) generated by LFSRs is expressed by non-AES primitive polynomial (6.2). Since GF(2^8) suitable for AES is expressed by AES irreducible polynomial (6.1), all elements of the generated GF(2^8) have to be transformed into AES GF(2^8) by the basis transformation.

$$\mathrm{m(x)} = x^8 + x^4 + x^3 + x + 1 \tag{6.1}$$

$$\mathrm{m'(x)} = x^8 + x^4 + x^3 + x^2 + 1 \tag{6.2}$$

Let assume d is the element in the GF(2^8) expressed by primitive polynomial m'(x) (6.2). Its polynomial representation [34] is

$$\mathrm{d(x)} = d_7x^7 + d_6x^6 + d_5x^5 + d_4x^4 + d_3x^3 + d_2x^2 + d_1x + d_0 \tag{6.3}$$

Let assume c is the element in the GF(2^8) expressed by the AES irreducible polynomial m(y) (6.1). Its polynomial representation is

$$\mathrm{c(y)} = c_7y^7 + c_6y^6 + c_5y^5 + c_4y^4 + c_3y^3 + c_2y^2 + c_1y + c_0 \tag{6.4}$$

If A is the root of the polynomial m(y) and B is the root of the polynomial m'(x) [34] than it holds that

$$\mathrm{m(A)} = A^8 + A^4 + A^3 + A + 1 = 0 \tag{6.5}$$

$$\mathrm{m'(A)} = B^8 + B^4 + B^3 + B^2 + 1 = 0 \tag{6.6}$$

Therefore the basis of both polynomials (6.1) and (6.2) can be expressed as [3]

$$[A^7, A^6, A^5, A^4, A^3, A^2, A^1, A^0] \tag{6.7}$$

$$[B^7, B^6, B^5, B^4, B^3, B^2, B^1, B^0] \tag{6.8}$$

The smallest root of the m(x) polynomial (6.1) is A = 0x02 [34]. The smallest root of the m'(x) polynomial (6.2) is B = 0x03 [34]. Therefore, we can find the relationship between A and B as

$$B = A \oplus 1 \tag{6.9}$$

where $\oplus$ repesents logic operation exclusive-or (XOR). In order to find the relationship between elements c and d, variables x and y in polynomials (6.3) and (6.4) have to be substituted by the roots of the irreducible polynomials:

$$\mathrm{c(A)} = c_7A^7 + c_6A^6 + c_5A^5 + c_4A^4 + c_3A^3 + c_2A^2 + c_1A + c_0 \tag{6.10}$$

$$\mathrm{d(B)} = d_7B^7 + d_6B^6 + d_5B^5 + d_4B^4 + d_3B^3 + d_2B^2 + d_1B + d_0 \tag{6.11}$$

After substitution according to equation (6.9)

$$\begin{aligned} d(A) \quad = \quad & d_7(A \oplus 1)^7 + d_6(A \oplus 1)^6 + d_5(A \oplus 1)^5 + d_4(A \oplus 1)^4 + \\ & d_3(A \oplus 1)^3 + d_2(A \oplus 1)^2 + d_1(A \oplus 1) + d_0 \end{aligned} \tag{6.12}$$

After simplification

$$\begin{aligned} d(A) &= d_7A^7 + (d_7 \oplus d_6)A^6 + (d_7 \oplus d_5)A^5 + (d_7 \oplus d_6 \oplus d_5 \oplus d_4)A^4 + \\ &\quad (d_7 \oplus d_3)A^3 + (d_7 \oplus d_6 \oplus d_3 \oplus d_2)A^2 + (d_7 \oplus d_5 \oplus d_3 \oplus d_1)A + \\ &\quad (d_7 \oplus d_6 \oplus d_5 \oplus d_4 \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0) \end{aligned} \tag{6.13}$$

If c(A) = d(A) than the relationship between coefficients can be described as

$$c = BT(d) = \begin{bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} d_7 \\ d_6 \\ d_5 \\ d_4 \\ d_3 \\ d_2 \\ d_1 \\ d_0 \end{bmatrix} = \begin{bmatrix} d_7 \\ d_7 \oplus d_6 \\ d_7 \oplus d_5 \\ d_7 \oplus d_6 \oplus d_5 \oplus d_4 \\ d_7 \oplus d_3 \\ d_7 \oplus d_6 \oplus d_3 \oplus d_2 \\ d_7 \oplus d_5 \oplus d_3 \oplus d_1 \\ d_7 \oplus d_6 \oplus d_5 \oplus d_4 \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \end{bmatrix} \tag{6.14}$$

where c = BT(d) describes the basis transformation and 8x8 matrix is the basis transformation matrix. In some cases it is beneficial to combine basis transformation with affine transformation forming compact combined transformation. Affine transformation [1] can be described as

$$AT(c) = \begin{bmatrix} s_7 \\ s_6 \\ s_5 \\ s_4 \\ s_3 \\ s_2 \\ s_1 \\ s_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \tag{6.15}$$

After substitution c = BT(d)

$$AT\_BT(d) = \begin{bmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{bmatrix} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \right) \begin{bmatrix} d_7 \\ d_6 \\ d_5 \\ d_4 \\ d_3 \\ d_2 \\ d_1 \\ d_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \tag{6.16}$$

where $\times$ operation represents matrix multiplication. After matrix multiplication

$$AT\_BT(d) = \begin{bmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} d_7 \\ d_6 \\ d_5 \\ d_4 \\ d_3 \\ d_2 \\ d_1 \\ d_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} d_7 \oplus d_4 \oplus d_3 \\ (d_7 \oplus d_6 \oplus d_4 \oplus d_2)' \\ (d_7 \oplus d_5 \oplus d_4 \oplus d_3 \oplus d_2 \oplus d_1)' \\ d_7 \oplus d_6 \oplus d_5 \oplus d_0 \\ d_7 \oplus d_4 \oplus d_0 \\ d_7 \oplus d_6 \oplus d_4 \oplus d_3 \oplus d_0 \\ (d_7 \oplus d_5 \oplus d_4 \oplus d_2 \oplus d_0)' \\ (d_7 \oplus d_6 \oplus d_5 \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0)' \end{bmatrix} \tag{6.17}$$

where ()' operation represents logic negation.

Complete S-box generator is shown in the Fig. 6 – 6. Detailed implementation of the S-box is shown in the Fig. 6 – 7 [3].

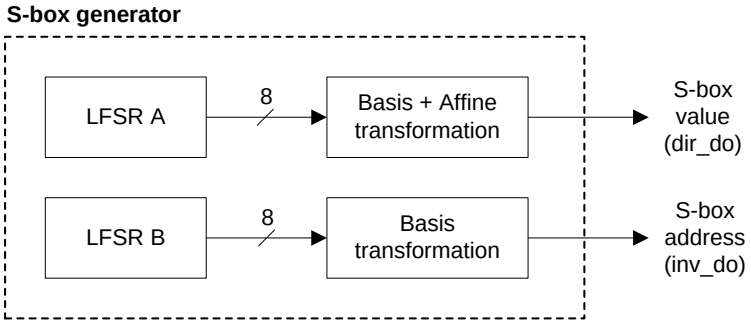

Fig. 6 – 6 S-box generator with basis transformation

6.1.3 Implementation of AES enc/dec core in FPGA

AES core has been implemented and tested in Actel Fusion AFS600, Actel Igloo AGL600 and Altera Cyclone III C5E144 FPGAs [3]. Both S-boxes in logic and in memory were tested. When implemented in Actel Fusion device, it turned out that AES with S-boxes in memory utilized 2987 tiles and highest achieved frequency was 58.5 MHz and AES with S-boxes in logic utilized 5499 tiles and highest achieved frequency was 28.7 MHz [3]. In case of Altera Cyclone III, only AES with S-boxes in memory was measured. It utilized 1705 logic elements (one logic element can implement one combinatorial gate with register in series, therefore it can be compared with 2 tiles) and the highest achieved

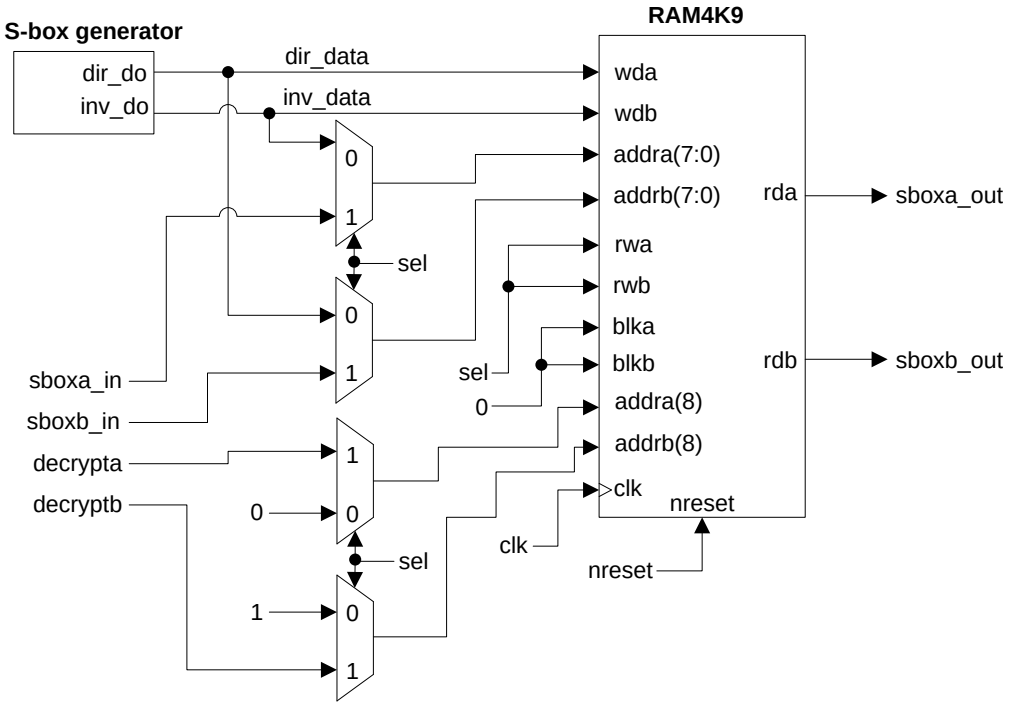

Fig. 6 – 7 Detailed implementation of the S-box in TDPRAM

frequency was 111.4 MHz [3]. AES with S-boxes in memory was the only tested in Actel Igloo device. Highest achieved frequency was 24.5 MHz [3]. It is important to notice that Actel Igloo device is targeted for low power applications and therefore its performance was reduced [3]. When measuring power consumption, Actel Igloo outperformed both Actel Fusion and Altera Cyclone III devices [3].

6.2 Analysis of cryptoprocessor core

There are two possible ways to design cryptoprocessor [4]. First way is to make analysis of all regular available open-source processors if they are modifiable to cryptoprocessor. Other way is to start building the cryptoprocessor from scratch. During the design time, designer can inspire in or even utilize control parts from open-source processors and design new datapath. Nevertheless the cryptoprocessor topic is completely new and no research has been carried out in this field before [4]. The cryptoprocessor hasn't been

implemented yet and is just in the design stage.

6.2.1 Modifiability analysis of available open-source processors

All processors except the Leon 3 [35] were acquired from opencores.org [36]. Main criteria for selecting the best processor to be reused were the following:

- suitable for implementing in FPGA
- easily modifiable to cryptoprocessor
- VHDL code should be readable and well commented/documented
- size of implementation has to be reasonable

According to these criteria most considered properties of processors were:

- synthesizability
- type of architecture (Harvard vs. von Neumann)
- implementation of registers (in logic vs. in RAM)
- readability and good modifiability of the VHDL code
- size of circuit (in tiles) and number of used RAM modules
- reusability of a control part of the processor
- number of stages in pipeline (preferred not more than 3)

Observed results are summarized in Tab 6 – 1 and Tab 6 – 2.

According to the results of the analysis, the following 10 processors (see Tab. 6 – 3) have been chosen to be reusable:

Only 5 processors were optimized for FPGA (registers in RAM, Harvard architecture ...). For more appropriate decision, every chosen processor has to be reviewed more carefully and tested on simulations. Most reusable part is the control logic. Datapaths have to be usually modified completely.

Tab. 6 – 1 Architecture and implementation comparison of open-source processors

Name of CPU	num of bits	architecture	pipeline	registers in
Lightweight 8080	8	Von Neumann	2	16x in logic
68HC08	8	Von Neumann	3	In logic
LEON3	32	Harvard	7	In RAM
Ax8	8	Harvard	N/A	In RAM(2x)
HPC-16	16	Von Neumann	N/A	In logic
McAdam's Marca	16	Von Neumann	4	16x in RAM
MiniMIPS	32	Von Neumann	5	In logic
PAVR	32	Von Neumann	6	In RAM(10x)
Plasma - MIPS	32	Von Neumann	3	In RAM
Ppx16	8	Harvard	N/A	In RAM, except acc.
RISC5x	8	Harvard	1	In RAM
Tiny64	32	Von Neumann	3	8x in logic
System68	8	Von Neumann	N/A	In logic
System11	8	Von Neumann	N/A	In logic
T48	8	Harvard	N/A	In logic
T65	8	Von Neumann	N/A	In logic
T80	8	Harvard	N/A	In RAM(2x)
T400	4	Harvard	3	In logic

Tab. 6 – 2 Comparison of code properties of open-source processors

Name of CPU	synthesizability	code length	readability	modifiability
Lightweight 8080	2007	1120	good, many comments	very good
68HC08	5517	2430	very poor, poor comments	poor
LEON3	9674	N/A	very good, perfect doc.	poor, too complex
Ax8	1315	1900	good, many comments	good
HPC-16	2894	3759	good, good document.	good
McAdam's Marca	N/A	N/A	very good, good doc.	very good
MiniMIPS	16171	N/A	medium, good doc.	good
PAVR	7876	7900	good, medium doc.	medium
Plasma - MIPS	4180	2000	medium, medium comments	medium
Ppx16	1172	970	medium, poor comments	good
RISC5x	2983	1700	good, good comments	medium
Tiny64	3529	530	good, good comments	good
System68	2802	3962	medium, medium comm.	poor
System11	3529	4792	medium, medium comm.	poor
T48	1547	5500	good, medium comments	medium
T65	1603	1700	medium, poor comments	good
T80	3229	3100	good, poor comments	good
T400	1216	3550	medium, medium comm.	medium

Tab. 6 – 3 Selected open-source processors to be reused in cryptoprocessor

Not very suitable for FPGA	More suitable for FPGA
Lightweight 8080	Ax8
HPC-16	McAdam's Marca
Tiny64	Ppx16
T48	RISC 5x
T65	T80

6.2.2 Design of new cryptoprocessor architecture

The other way to develop the cryptoprocessor is to create a new design. However, there are many issues that need to be considered. Control logic has to be very stable, because it directly influences the performance of the datapath. Number of used logic resources also has to be considered. In this case datapath is the most expensive one. Thus it is very important to decide to base the architecture on either 128-bit datapath or 32-bit datapath.

Block diagram of the cryptoprocessor is shown in the Fig. 6 – 8 [4]. As shown in the Fig. 6 – 8, data and key registers are physically separated in different RAM modules. This precaution helps to preserve security of the key inside the FPGA [4]. Even if it doesn't seem critical at first sight, it can turn to catastrophe if an unencrypted key is transferred out of the cryptoprocessor via the data bus [4]. How is this possible? If an attacker intentionally applies very strong magnetic field to the critical parts of the FPGA (i.e. select signal of the multiplexer) key can be redirected to output data bus. However it is not sufficient to physically separate data and key registers. The whole architecture (buses, multiplexers, ...) has to be designed carefully, otherwise the security is reduced. For these reasons, parts of the architecture that operate with keys need to be specially treated. In this case, resource optimization is not as important as the security of keys. Some key paths can be doubled to assure safe transfer of the key from one block to another. In the

Fig. 6 – 8 critical key bus is illustrated in red color [4].

Despite the security of the AES cipher there is a possibility to extract a key from big amount of encrypted data. For this reason it is recommended to change key every 64 kB of data [4]. This issue has been solved by using two sets of registers for keys: Main key register (for permanent key) and Session key register (for temporary key) [4]. At the beginning session key is generated by the RNG. After being stored in session registry, it is encrypted by the main key and sent out to receiver. Afterwards receiver decrypts the encrypted session key with the same main key. Subsequently, the sender can encrypt data with generated session key and send them to the receiver. The receiver decrypts the data with the previously received and decrypted session key. For each 64 kB of data new session key is generated [4].

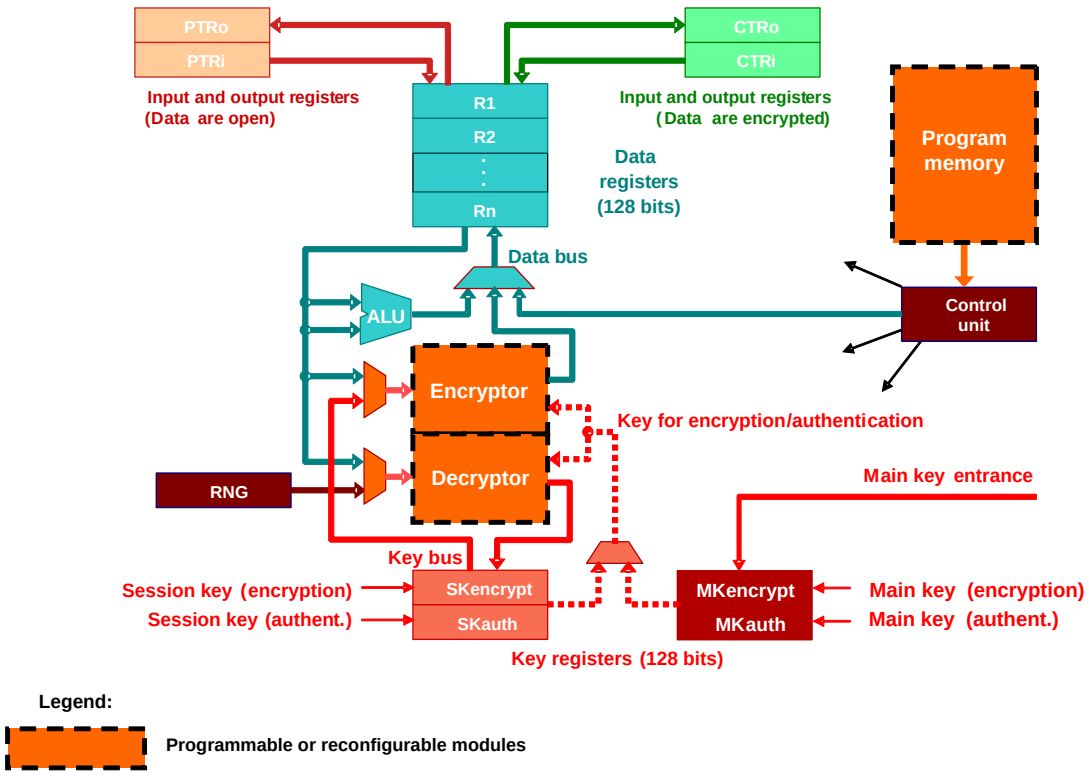

Fig. 6 – 8 Block diagram of the cryptoprocessor

6.2.2.1 Architecture with 128-bit datapath

Implementation of the 128-bit datapath of the cryptoprocessor is shown in the Fig. 6 – 9. Arithmetic logic unit is broken into separate parts: XOR, increment, compare and flow-through operations. For most of the block cipher modes XOR operation is required. Increment can be used in the CTR block mode (counter mode) and comparison is usually used during authentication. In the center of the architecture is the cipher block. Archi-

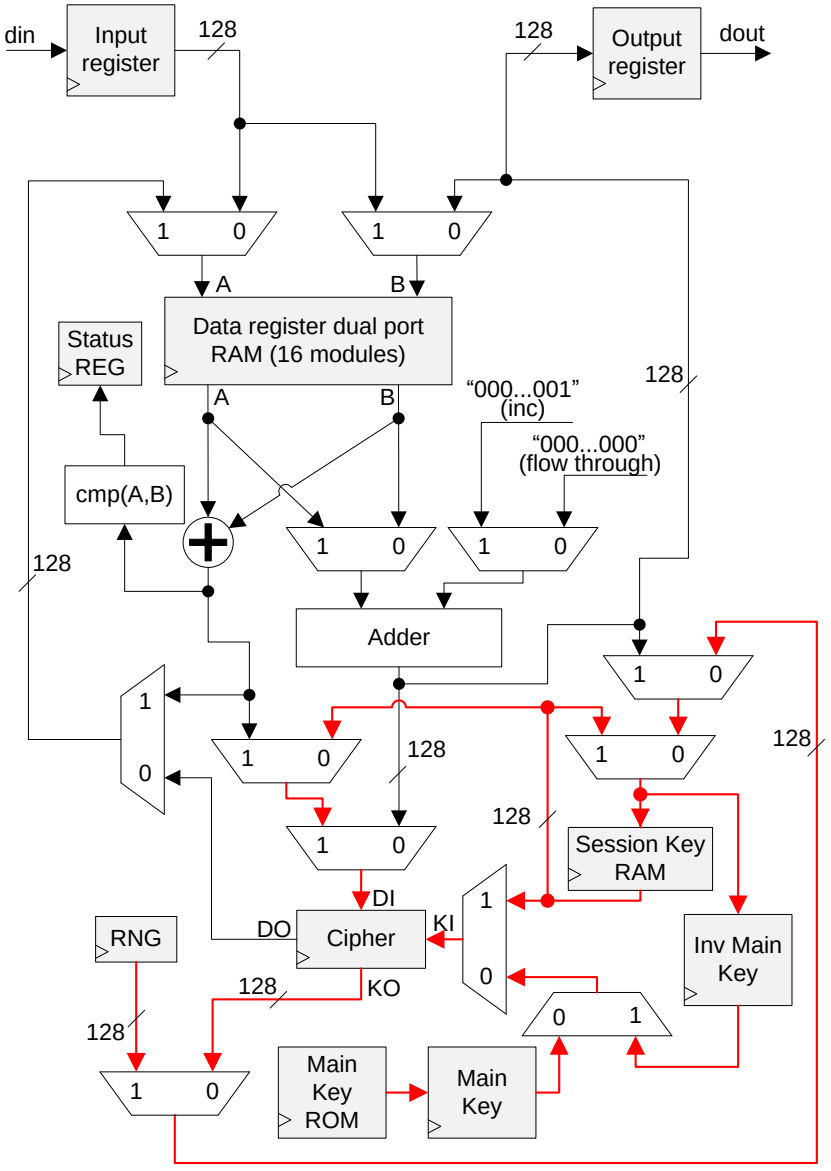

Fig. 6 – 9 Implementation of cryptoprocessor with 128-bit datapath

tecture allows implementing any type of symmetric cipher with 128-bit data length and 128-bit key length. For the particular project AES was chosen as the one recommended by NIST [1].

One may notice red colored buses in Fig. 6 – 9. These buses represent paths where keys are transferred and require special treatment. All these paths could be doubled to secure the integrity of transferred keys. Furthermore, all the red buses are designed in the way that no key can be multiplexed to the data bus without being encrypted first.

In case of decryption, generated session key for encryption has to be expanded first, the last expanded key has to be stored in session key register and only then it can be used for decryption. In case of decryption with the main key, the last expanded key has to be stored in inverse main key register.

As shown in the Fig. 6 – 9, 128-bit architecture is very convenient but is very expensive in terms of FPGA resources. If one decides to implement cryptoprocessor with 128-bit datapath, 16 memory modules for data registers have to be included as well as 8 memory modules for session key registers. AES cipher itself uses 10 memories for storing S-boxes. In this case all 24 memory modules available in Fusion FPGA [2] wouldn't be enough. This allows the implementation of the system in bigger FPGA families only. The second disadvantage of this robust 128-bit architecture lies in number of used logic resources. Each 128-bit multiplexer utilizes 128 tiles. In this case all multiplexers utilize about 1400 tiles. Further, it is important to mention the critical path. The architecture was designed to be as symmetric as possible in terms of critical paths' delays. However, the biggest delay is introduced in the right path because of the Adder. For all mentioned issues it could be advantageous to design cryptoprocessor capable of working with 128-bit words but using only 32-bit datapath.

6.2.2.2 Architecture with 32-bit datapath

Implementation of the 128-bit cryptoprocessor operating with 32-bit datapath is shown in the Fig. 6 – 10. Despite the shift to 32-bit datapath, each operation has to be carried out over 128-bit words. Therefore, each operation has to be performed in 4 steps where in

each step a sub-operation is carried out over 32-bit sub-words. However, computational performance is 4 times lower.

When comparing number of used FPGA resources, number of tiles necessary for multiplexers decreases rapidly. In 32-bit datapath only about 360 tiles are needed to implement multiplexers when compared to about 1400 tiles needed in 128-bit datapath. All other parts of datapath are smaller. For XOR operation only 32 tiles are needed. Adder benefits from less resource count but the most important is the reduction of a delay in the critical path because of a shorter carry chain within the Adder.

The most important are savings in the number of used memory modules. In contrast to 128-bit data registers (using 16 RAM modules) 32-bit data registers require only 4 RAM modules. The session key register requires only 2 RAM modules when compared to 8 RAM modules needed in 128-bit datapath. This saving allows implementing cryptoprocessor in Actel Fusion FPGA [2] used in this project.

However, cipher module operates with 128-bit words. For this reason, word length conversion blocks have to be added. The convertor block from 32-bit to 128-bit is shown in the Fig. 6 – 11. One may notice three sets of 32-bit registers implemented in logic. These registers are controlled by the enable signals E0, E1 and E2 and provide temporary storage of successive 32-bit sub-words. Last 32-bit sub-word is not registered and is transferred directly to 128-bit output bus. This solution helps to save one clock cycle, which would have been needed in case of registration of the last 32-bit sub-word. The whole conversion requires 4 clock cycles.

Inversed convertor is shown in the Fig. 6 – 12. One may notice that no registration is necessary and all 32-bit sub-words are successively multiplexed into 32-bit bus. The only necessity is the correct control of select signals S1, S2 and S3. The whole conversion requires 4 clock cycles.

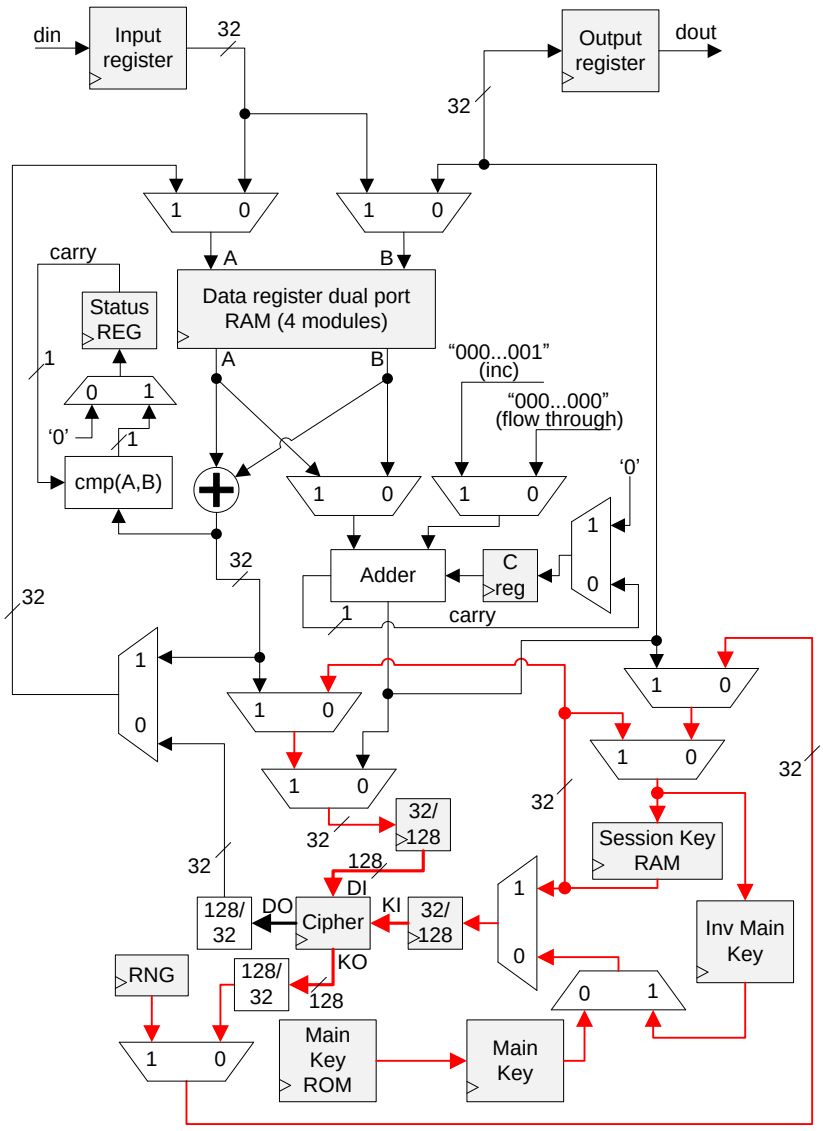

Fig. 6 – 10 Implementation of cryptoprocessor with 32-bit datapath

6.3 Interconnection of cryptoprocessor with ARM7

After the whole cryptoprocessor is completed, it has to be connected to the main data bus of the system. This interconnection is illustrated in the Fig. 6 – 13. The cryptoprocessor should behave as a black box which would read data from 32-bit input FIFO, carry out its task and write results to 32-bit output FIFO. The status should indicate accomplishment of the task and APB access controller could read this status and send it to ARM7.

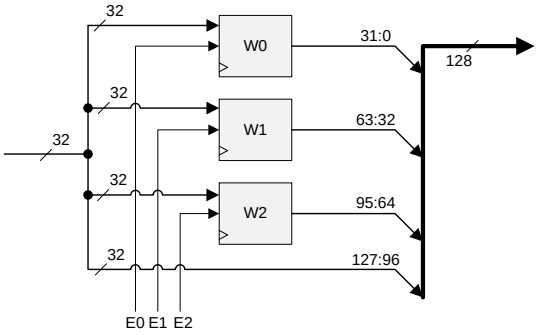

Fig. 6 – 11 Conversion of 4 successive 32-bit words into 128-bit word

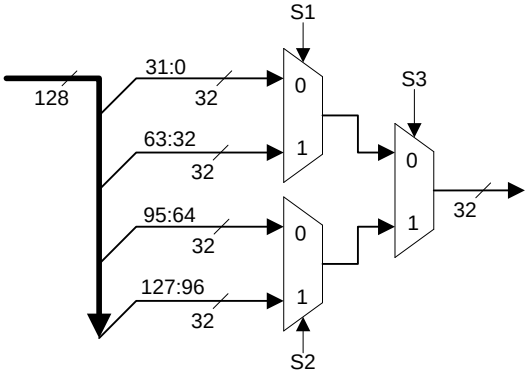

Fig. 6 – 12 Conversion of 128-bit word into 4 successive 32-bit words

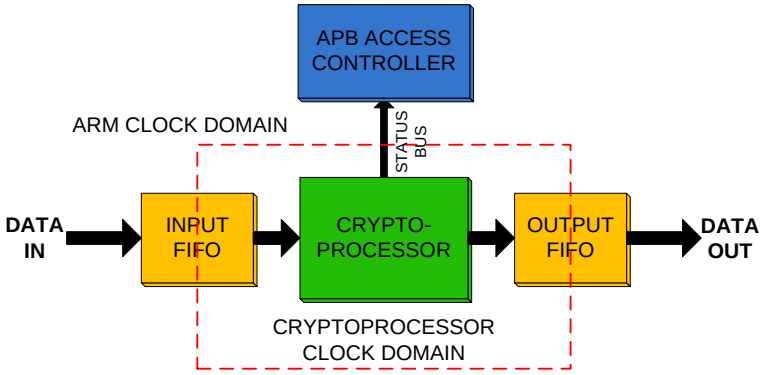

Fig. 6 – 13 Interconnection of cryptoprocessor with ARM7

Since cryptoprocessor is separated from the rest of the system, special protocol has to be created and applied on the data. First 32 bits of data should contain header. In this header the whole necessary configuration should by present. Cryptoprocessor should access this header, compare it with masks stored in special registers and decide which

action to perform. Afterwards data could be read from 32-bit input FIFO and processed. This study is the first step and further work is necessary to design exact protocol for data and control packets transfer and processing.

7 Software implementation

7.1 ARM7 firmware design

Firmware was written in c-code in Keil μVision3 development environment [19] using external GNU compiler. Its complexity lies in close-to-hardware (low-level) programming. At the first step, simple functions had to be created. These functions were directly accessing registers in peripherals in the ARM7's address space (peripherals connected to APB bus). Functions were grouped in libraries according to peripheral they were written for. Finally, Main function with an infinite loop was written. This function uses all low-level functions to access ARM7's peripherals.

ARM7 core [12], [13] has its firmware stored in the non-volatile flash memory inside FPGA [2]. This memory is accessed by CoreAhbNvm component (flash controller), which translates communication between the processor and the flash memory. Because the processor executes instructions from the beginning of the memory space, CoreAhbNvm's address space (range: 0x00000000 - 0x0FFFFFFF) is in the beginning of the ARM7's address space.

Firmware is divided into these parts:

- ***Main function*** - including infinite loop
- ***I^{2}C library*** - containing functions for I^{2}C bus control
- ***CamDMA library*** - containing functions for image acquisition and external memory sharing
- ***Cyp_usb library*** - containing functions for USB communication

Firmware for ARM7 is responsible for image acquisition, memory sharing, camera initialization and USB communication. This firmware is just the first version. In future it has to be expanded to include optical signature extraction and cryptoprocessor control.

7.2 USB transport protocol

To transport data via USB, protocol had to be created. The idea was to divide data into packets with headers. The transport protocol postulated asymmetric data rates in different directions. Packets transferred from ARM7 to PC had bigger header and were capable to contain more data. Packets transferred from PC to ARM7 had smaller header and smaller data space. This asymmetry was due to requirements of the system. ARM7 had to send big amount of image data while PC had to send only command with little data to specify request.

The communication model is illustrated in the Fig. 7 – 1. Communication is carried out by sending a pair of physical packets. First packet represents the logical header and the other packets the logical data.

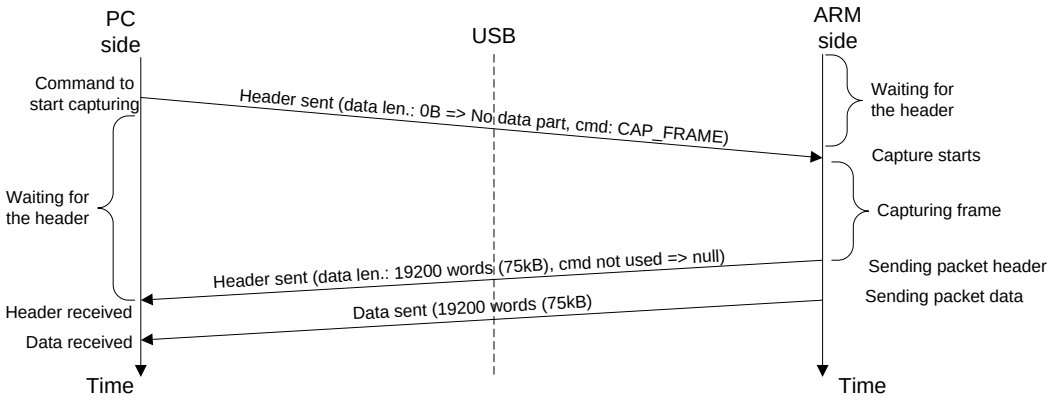

Fig. 7 – 1 Communication model between ARM7 and a PC

7.3 PC software implementation

PC software was programmed in the Borland C++ Builder 6.0 [20]. Its purpose is to visualize the results of the capturing process on the PC. Even if embedded system was able to capture frames at the speed of 30 fps, USB interface slowed down the process to approximately 4 fps. As long as PC serves for visualization purposes only, 3-4 fps is considered as sufficient. This visualization is used just to prove that captured frame inside the memory was captured correctly.

7.3.1 PC interface from user's point of view

PC interface is very easy to be used. It has a lot of comments displayed at the hint line and the small glyphs for the user to intuitively understand the meaning of the buttons. The interface contains two windows:

- *Main* window
- *About* window

7.3.1.1 The Main window

Main window is shown in the Fig. 7 – 2.

Fig. 7 – 2 Main window of the PC interface

Control panel is located in the bottom part of the window. To initialize camera, the user has to click to the Camera initialization button. After the initialization, Start Capture button displays. After clicking on the Start Capture button, the Screen starts to display the captured picture. FPS will be dynamically measured and displayed in their FPS display area. Time display area displays actual system time. After performing any user action, it will be displayed in the console panel. The console can be cleared with the button located in the lower right corner of the window. The last button in the lower right corner shows the About window.

If there was a need to display the frame errors, the interface can be switched to the debug mode by pressing the Ctrl+Shift+F12 key combination. When in debug mode, small red "D" letter will be added in the left part of the hint line. If there was an error during capturing process, it would be displayed in the console with the time when the error occurred. Fig. 7 – 3 shows interface in debug mode.

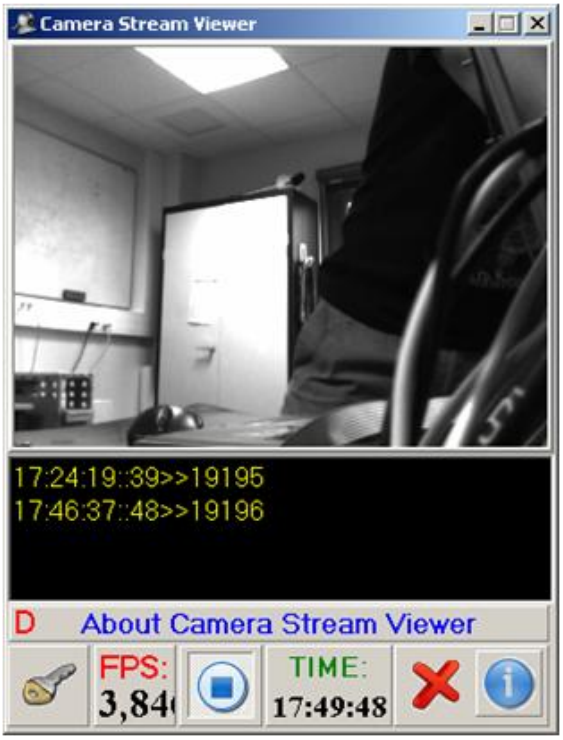

Fig. 7 – 3 Application in the Debug mode

7.3.1.2 The About window

About window displays the basic information about the application. Fig. 7 – 4 shows the About window.

Fig. 7 – 4 About window in the PC interface

8 Conclusion

As it has been stated in the introduction, one of the biggest problems of the modern world is faking of original products and thus enabling to gain profit illegally from items being in a declarable ownership of their sole authors. This research has been carried out to establish hardware platform to install the procedures of distinguishing original and fake products from each other with the highest possible accuracy.

From the technical point of view, this project dealt with the realization of the first steps important for this new technology, particularly with the image acquisition, encryption, analysis of cryptoprocessor and feasibility study about implementation of optical signature extraction algorithm to hardware. Image acquisition subsystem together with ARM7 subsystem have been successfully implemented in hardwate and tested. Prototype has been created. AES cipher/decipher subsystem have been implemented in FPGA and basic parameters have been measured. It hasn't been tested yet, since block cipher modes are required for testing in hardware and cryptoprocessor for performing these block cipher modes hasn't been implemented yet. Architecture of cryptoprocessor has been proposed in structural level. Future development lies in the implementation and testing of the cryptoprocessor. Product authentication algorithm has been designed and tested in Matlab. Analysis of the complexity of this algorithm has been carried out. Other algorithms are developed too and finally the best algorithm will be chosen.

The work has dealt with several questions and difficulties concerning the particular project, whereas the most significant obstacles for the research were the stability of camera operation with easy configuration, optimized VHDL design inside the Actel FPGA and setup of the USB interface. During the design of AES cipher module, mathematic basics in the field of Galois fields had to be studied and applied in the design. All these difficulties have been overcome with success and the project has met all the requirements needed for its realization.

The most discussed and researched parts were the VHDL optimization with frame corruption issues, optimized generation of S-boxes inside the FPGA and study of third-

party processors' VHDL codes. Embedded system was able to capture frames with the frame rate of 30 fps. The only part that slowed down the frame rate was the USB part. Because of the USB module and the USB driver obstacles, PC frame rate was rapidly decreased to 3-4 fps. Because visualization served for frame-check only, speed of frames per second in the PC is not important as long as embedded system is capable of real-time capturing. A lot of mathematic calculations and simplifications were necessary to derive transformation matrices. These matrices were successfully implemented in transformation blocks. Basic theory in the field of LFSRs, NLFSRs and generation of maximum length sequences was studied too.

To sum it up, the first steps required for developing the above described technologies has been carried out with success. Further development and the final realization lies on the subsequent research and on the final technical realization. It can be concluded, that this project has been finished successfully and it has fulfilled all the expectations connected with its practical realization.

Bibliography

[1] NIST, *FIPS 197: Advanced Encryption Standard*, 2001, available online: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[2] ACTEL, *Fusion Handbook*, 2009, available online: http://www.actel.com/documents/Fusion_HB.pdf

[3] L. GASPAR, M. DRUTAROVSKY, V. FISCHER AND N. BOCHARD, *Efficient AES S-boxes implementation in non-volatile FPGAs*, submitted for FPL 2009 conference, Prague, Aug. 31-Sep. 2, 2009

[4] ANR, *SecReSoC project*, France, 2009

[5] ST MICROELECTRONICS, *VS6524: VGA single-chip camera module*, datasheet, 2006, available online: http://www.stdistributiondemandcreation.com/steval/datasheets/vs6524.pdf

[6] ST MICROELECTRONICS, *X24 Camera integration kit: User manual*, 2007

[7] ACTEL, *Actel System Management Board: User's Guide*, 2006

[8] GSI TECHNOLOGY, *GS88018/32/36BT-xxxV: 9Mb Sync Burst SRAMs*, datasheet, 2007, available online: http://www.gigasemi.com/880xxB_V.pdf

[9] ST MICROELECTRONICS, *M29W800DT: 8Mbit 3V Supply Flash Memory*, datasheet, 2004, available online: http://www.btdesigner.com/pdfs/M29W800D.pdf

[10] MICRONIC, *USB board*, datasheet, 2006

[11] CYPRESS, *CY7C68013A: EZ-USB FX2LP USB Microcontroller*, datasheet,2008, available online: http://www.cypress.com/?docID=5485

[12] ACTEL, *CoreMP7*, datasheet, 2007, available online: http://www.actel.com/documents/CoreMP7_DS.pdf

[13] ARM LTD., *ARM7TDMI-S: Technical Reference Manual*, 2001, available online: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0234b/DDI0234.pdf

[14] WIKIPEDIA, *ARM7TDMI*, 2009, available online: http://en.wikipedia.org/wiki/ARM7TDMI

[15] ARM LTD., *AMBA Specification*, 2001,

[16] NXP, *The I^2C-Bus Specification*, 2000, available online: http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf

[17] ACTEL, *Libero IDE Quick Start Guide: for Software v8.4*, 2008, available online: http://www.actel.com/documents/gettingstarted_ug.pdf

[18] MENTOR GRAPHICS, *ModelSim User's Manual: software version 6.3g*, 2008, available online: http://www.actel.com/documents/modelsim_ug.pdf

[19] KEIL, *µVision User's Guide*, 2009, available online: http://www.keil.com/support/man/docs/uv3/

[20] BORLAND, *Borland C++ Builder reference guide*, 2002

[21] SYNPLICITY, *Synplify FPGA Synthesis*, 2008, available online: http://www.actel.com/documents/synplify_ref_ug.pdf

[22] ACTEL, *Designer v8.4 User's Guide*, 2008

[23] ACTEL, *FlashPro User's Guide v8.5*, 2008, available online: http://www.actel.com/documents/flashpro_ug.pdf

[24] ACTEL, *SmartTime v8.5 User's Guide*, 2008, available online: http://www.actel.com/documents/smarttime_ug.pdf

[25] ACTEL, *SmartPower v8.5 User's Guide*, 2008, available online: http://www.actel.com/documents/smartpower_ug.pdf

[26] ACTEL, *SoftConsole: Quick Start Guide*, 2006, available online: http://www.actel.com/documents/SoftConsole_QS_UG.pdf

[27] ACTEL, *CoreConsole v1.4 User's Guide*, 2007, available online: http://www.actel.com/documents/CoreConsole_ug.pdf

[28] NIST, *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*, 2001, available online: http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[29] NIST, *Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, 2005, available online: http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

[30] NIST, *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*, 2004, available online: http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf

[31] NIST, *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, 2007, available online: http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

[32] V. FISCHER, M. DRUTAROVSKÝ, P. CHODOWIEC AND F. GRAMAIN, *InvMix-Column Decomposition and Multilevel Resource Sharing in AES Implementations*, *Very Large Scale Integration (VLSI) Systems, IEEE Transactions* on Volume 13, Issue 8, Aug. 2005, pp: 989-992. ISSN 10638210

[33] A. SATOH, S. MORIOKA, K. TAKANO, AND S. MUNETOH, *A compact Rijndael hardware architecture with S-Box optimization*, *Proc. Theory and Application of Cryptology and Information Security* (ASIACRYPT'01), ser. LNCS, vol. 2248. Gold Coast, Australia: Springer-Verlag, Dec. 9-13, 2001, pp. 239–254.

[34] D. CANRIGHT, *A very compact s-box for AES*, *Proc. International Workshop on Cryptographic Hardware and Embedded Systems* (CHES'05), ser. LNCS, vol. 3659. Edinburgh: Springer-Verlag, Aug. 29–31, 2005, pp. 441–455.

[35] JIRI GAISLER, SANDI HABINC AND EDVIN CATOVIC, *GRLIB IP Library User's Manual: LEON3 quick-start guide Quick Start Guide*, 2009, available online: http://www.gaisler.com/products/grlib/grlib.pdf

[36] OPENCORES, *OpenCores database: Processors*, 2009, available online: http://www.opencores.org/

Appendices

CD containing the following:

Appendix A electronic version of this document

Appendix B VHDL source codes of:

- ARM7 subsystem
- Image acquisition subsystem
- S-box with generator
- AES cipher/decipher core

Appendix C ARM7's firmware souce codes for Keil μVision 3

Appendix D PC software source codes for Borland C++ Builder