Design of a Secure Microprocessor

A. Toker, T. Ayav

***Abstract* — This work presents the prototype design of a secure microprocessor that executes encrypted programs. The aim of such a secure processor is to prevent programs from being copied; otherwise the encrypted programs will not run on ordinary platforms. The idea is demonstrated through a concrete design of a 16-bit soft processor, namely CryptOdin, along with the simulations and physical implementation on FPGA. This processor is expected to easily hide critical algorithms/programs without sacrificing the performance by the help of a dedicated decryption unit implemented in the processor.**

***Index Terms*—VHDL, Encryption/Decryption, Soft Processor, Embedded Design.**

I. Introduction

It is quite common in many embedded system designs that the developed algorithms generally contain the vital results of research and need to be hidden against potential reverse engineering problems. For instance, many military applications involve high quality research and the resulting algorithms or programs must be protected.

There might be several methods to protect programs in embedded systems such as:

(i) Storing the program in the memory-on-chip
(ii) Implementing the algorithm in the hardware
(iii) Encrypting the program

Method (i) suffers from the lack of memory size since internal memory size is very limited in almost all microcontrollers. Method (ii) proposes implementation of algorithm in an FPGA or ASIC design. This truly dedicated chip has many advantages such as compactness and efficiency yet it is not flexible, *i.e.* the implemented algorithm cannot be changed, hence it is most suitable for mass production. Finally, method (iii) imposes encrypting the program to store in an external memory.

In this project, physically a secure processor has been designed and simulated. The secure coprocessors contain special-purpose hardware in the CPU and memory. CryptOdin features an on-chip block cipher hardware between the cache and the bus interface. Code and data are decrypted on-the-fly while being fetched from RAM. Even someone with physical access to the printed circuit board cannot observe the executed software and its data flow.

The prototype hardware architecture for Secure Microprocessor is developed targeting Spartan3e - xc3s500e Field Programmable Gate Array (FPGA) development platforms.

The main features of CryptOdin can be listed as follows:

16 bit address and 16 bit data bus
Harvard Architecture
32 general purpose registers
38 instructions
16-bit and 32-bit long instructions
Pipelining
Operation at maximum 68 MHz clock frequency
Execution of encrypted programs

Cryptodin provides utmost security by executing the programs that are previously encrypted and stored in the external memory. Please note that the power of this method depends on the encryption algorithm provided that the necessary precautions against reverse engineering are taken in the fabrication of integrated circuit [1].

The organization of the text is as follows: Section II summarizes the background. In particular, we mention the secure processors, FPGA and soft processor design issues. We thoroughly explain the key points in the design of our secure processor CryptOdin in Section III, followed by the simulation results given in Section IV. Finally we conclude the paper with Section V, giving further discussion about limitations and future research.

II. Background

The classification of secure processor designs is based on the location of the crypto-engine necessary to implement encryption or decryption of instructions and data. The crypto-engine can be placed on-chip, in a single monolithic architecture or it can be placed off-chip, like a field programmable gate array (FPGA)-based co-processor. Bus

Manuscript received March 30, 2010.

A. T. is with Izmir University, 35350-Izmir, Turkey. (phone: +90-232-2464949; fax: +90-232-2240909; e-mail: atilla.toker@izmir.edu.tr).

T. A. is with Izmir Institute of Technology, Izmir, Turkey. He is now with the Department of Computer Engineering (e-mail: tolgaayav@iyte.edu.tr).

encryption technique was first introduced 25 years ago. The CPU was considered as secure and consequently all data and addresses are in decrypted form inside the CPU and encrypted outside the processor chip. A cipher unit is implemented onchip and a secret cipher key is located in an on-chip register [1][2][3].

When we use a secure code, we can easily protect executables from being copied and illegally utilized by attackers. The code to be protected can be distributed and stored in encrypted form, so copying it without obtaining the code decryption key is futile. Our design prevents the execution of a program on other machines, and also protects software code from any access, keeps unauthorized reverse engineers from observing the memory and the execution of program instructions.

A. *What is a FPGA?*

Field-programmable gate arrays (FPGAs) fill a need in the design space of digital systems, complementary to the role played by microprocessors [4].

FPGA has become the main stream in complex logic circuit design due to its flexibility, ease of use, and short time to market [5]. The area filled by FPGAs has grown enormously in the past twenty years since their introduction [4]. FPGAs take advantage of complex chips to improve the chip design in several ways: FPGAs can be quickly programmed, their performance characteristics are predictable, and the same part can be used for several different logic designs [4].

The two most common languages used for FPGA design are VHDL and Verilog. The acronym VHDL stands for the VHSIC Hardware Description Language. The acronym VHSIC, in turn, refers to the Very High Speed Integrated Circuit program [6]. We have chosen to use VHDL for the design.

Xilinx Integrated Software Environment (ISE) is an integrated design environment that allows you to design Xilinx FPGA. ModelSim is a tool that integrates with Xilinx ISE to provide simulation and testing. Two kinds of simulation are used for testing a design: functional simulation and timing simulation [7].

B. *Soft Processor Design*

A soft processor is a microprocessor fully described in software, usually in VHDL or Verilog, which can be synthesized in programmable hardware, such as FPGAs. While a soft processor cannot easily match the performance/area/power of a hard processor, soft processors do have several compelling advantages. Using a generic FPGA chip, a designer can implement the exact number of soft processors required by the application, and the CAD tools will automatically place them within the design to ease routing. Since it is implemented in configurable logic, a soft processor can be tuned by varying its implementation and complexity to match the exact requirements of an application. While these benefits have resulted in wide deployment of soft processors in FPGA-based embedded systems, the architecture of soft processors has yet to be studied in depth [9].

III. CRYPTODIN: 16-BIT SECURE MICROPROCESSOR

In single-chip FPGA implementation of our secure processor, we present an implementation of a secure Blowfish algorithm, and standard microcontroller units on a single Xilinx Virtex-Spartan3E FPGA.

The schematic of the designed Secure Microprocessor is shown in Fig. 1 in the Appendix. The secure processor consists of single chip microprocessor and a decryption circuit External Memory Access Unit with Decrypter (EMAD) for incoming instructions. The EMAD is placed between the Program Counter logic and the external main memory as seen in Figure 1.

External memory and peripherals are assumed to be untrusted; they may be observed and tampered with at will by an adversary [1]. In our design, the cryptographic keys are stored in EMAD. User can not modify cryptographic key as the key is hardcoded in the EMAD unit. Decryption enable/disable behavior is selected by the *CRYPTON* and *CRYPTOFF* assembly commands.

The instructions are stored as encrypted or plaintext in the ROM. The ROM has opcodes, addresses and constants for each operation. EMAD receives the data and instructions from external ROM, and sends them to internal units to control the operations. The Control Unit (CU) is used to control the internal units. The Arithmetic-Logic-Unit (ALU) is needed for arithmetic operations. 32x16 bit general purpose registers are also implemented.

The external instructions from ROM are decrypted by the Blowfish algorithm. The detailed explanation about the EMAD and Blowfish is given in the following subsection.

The datapath consists of ALU, Multiplier, Program Counter, Instruction Register, Stack external ROM/RAM modules and Shifter logic with 32x16-bit registers as shown in Figure 1 in the Appendix.

The secure processor has two levels of code. One of them is encrypted code using the defined encryption key, the other one is plain text. The defined CryptOdin opcodes are given the Table I.

TABLE I
CRPYTODIN OPCODES

Mnemonic	Operands	Comments
Nop		NOP: instruction does nothing.
Add	Ri,Rj;	Addition: Adds Ri to Rj and stores the results back in Ri
Addi	Ri, #im	Addition (immediate): Adds a word value to the Ri and stores the results back in the Ri.
Sub	Ri,Rj;	Subtaction: Subtracts Rj from the Ri and stores the results back in Ri.
Subi	Ri, #im	Subtraction (immediate): Subtracts a word value from the Ri and stores the results back in the Ri.
Mul	Ri,Rj	Multiplication: returned in the Ri. The high-order byte of the product is returned in Rj.
Muli	Ri, #im	Multiplication (immediate): multiplies the signed 16-bit integer in Ri
Mulu	Ri,Rj	Multiplication (unsigned): multiplies the unsigned 16-bit integer in Ri
Inc	Ri	Increment: Increments Ri by 1.
Dec	Ri	Decrement: Decrements Ri by 1.
Cmp	Ri	Compare: compares two operands
Andr	Ri,Rj	AND:performs a bitwise logical AND operation
Andi	Ri, #im	AND (immediate): performs a bitwise logical AND operation
Orr	Ri,Rj	OR: performs a bitwise logical OR operation
Ori	Ri, #im	OR (immediate): performs a bitwise logical OR operation
Notr	Ri ; Rxi	NOT: logically complements the value of the specified destination operand
Xorr	Ri,Rj	XOR:performs a bitwise logical XOR operation between the specified operands
Xori	Ri, #im	XOR (immediate): performs a bitwise logical XOR operation operand.
Sllr	Ri	Logical shift left: shifts the 16-bits in the register left one bit position.
Srlr	Ri	Logical shift right: shifts the 16-bits in the register right one bit position.
Slar	Ri	Arithmetic shift left:
Srar	Ri	Arithmetic shift right:
Lw	Rxi,address	Load word: Load address word to register
Sw	Rxi,address	Store word: Store address word from register
Syscal	address	System call (SW interrupt):
Hlt		Halt:
Beq	offset	Branch if equal to 0: Branch to the specified address if the value in the accumulator is 0.
Bne	offset	Branch if not equal to 0: Branch to the specified address if the value in the accumulator is not 0.
Ba	offset	Branch always near: Branch to the target address specified by offset. PC+offset
Bf	offset+extension	Branch always far: Branch to the target address specified by offset and extension.
BL	address	Branch and Link: Branches to an instruction specified by the branch target address.
Movi	Ri, #im	Move data (immediate):
Movdi	@Ri,Rj	Move indirect/direct data between registers:
Movsi	Ri,@Rj	Move direct/indriect data between registers:
Mov	Ri,Rj	Move direct/direct data between registers:
Movii	@Ri,@Rj	Move indriect/indriect data between registers:
Clr	Ri	Clear register
Clr	@Ri	Clear indirect register
Ret		Return
Crypton		Turns On the opcode decrypter unit
Cryptoff		Turns Off the opcode decrypter unit

A. External Memory Access Unit with Decrypter (EMAD)

External Memory Access Unit with Decrypter (EMAD) plays the role of buffer between the external memory and instruction register. The second and more important duty of EMAD is to perform decryption and encryption on the program code and data. The external instructions from ROM are decoded in Blowfish algorithm. Similarly, the internal data are encoded before writing them to the external memory.

We implement the Blowfish algorithm for encryption/decryption in EMAD. Blowfish is a symmetric block cipher that can be used as a drop-in replacement for DES or IDEA. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic and exportable use. Blowfish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms. Since then it has been analyzed considerably, and it is slowly gaining acceptance as a strong encryption algorithm [10][11].

Blowfish is a block cipher that encrypts data in 8-byte blocks. The algorithm consists of two parts: a key-expansion part and a data-encryption part. Key expansion converts a variable-length key of at most 56 bytes (448 bits) into several subkey arrays totaling 4168 bytes. Blowfish has 16 rounds. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round [11].

TABLE II
EXAMPLE ASSEMBLY PROGRAM

Mnemonic	Comment
…	
CRYPTON	Turn on the encryption in EMAD
%ENCRYPT KEY	Assembly directive to encrypt the following machine code using the key “KEY”.
DEC R1	Decrement R1 register
SW R1, 0003	Store word
MOVI R0, 0005	Move data (immediate)
NOP	No operation
INC R1	Increment R1
CRYPTOFF	Turn off the encryption in EMAD
%PLAINTEXT	Assembly directive to disable code encryption
BEQ +02	Branch if equal to accumulator is 0
NOP	No operation
NOP	No operation
INC R31	Increment R31 register
HTL	Halt

The reason why we have chosen Blowfish is twofold. First, it is unpatented and license-free, and is entirely available for all intents and purposes. Second, both the

enciphered text and plain text are the same in size, which is an important design specification of processors.

IV. Hardware Realization And Simulation Results

The secure processor core described in VHDL using ModelSim simulator and compiled using Xilinx ISE8.2. Our secure processor CrpytOdin was targeted at the Xilinx's Spartan3e-xc3s500e development platform. XILINX ISE -The Xilinx Integrated Software Environment- used for development. ISE provides the HDL and schematic editors, logic synthesizer, fitter, and bitstream generator software [7]. After successfully compiling an FPGA design using the Xilinx development software, the design can be downloaded using the iMPACT programming software and the USB cable.

The Spartan-3E includes several types of interconnect as local, general-purpose, I/O, global, and clock. The key features of the Spartan-3E Starter Kit board are: Xilinx XC3S500E Spartan-3E FPGA, Up to 232 user-I/O pins, 320-pin FBGA package, over 10,000 logic cells, Xilinx 4 Mbit Platform Flash configuration PROM, x16 data interface, 100 MHz, 16 MByte (128 Mbit) of parallel NOR Flash (Intel StrataFlash), FPGA configuration storage, 16 Mbits of SPI serial Flash (STMicro), 2-line, 16-character LCD screen.

The designed secure processor CryptOdin has 16 bit address and 16 bit data bus and Harvard Architecture. It has 32 general purpose registers and 38 instructions. An instruction pipeline technique is used during design. It operates at maximum 68 MHz clock frequency and can execute encrypted programs.

The sample assembly program is shown in the Table II. Please note that the assembler of Cryptodin has built-in encryption algorithm, hence it can produce the encrypted code that will be loaded into the external program memory. The programmer tells the assembler to enable and disable the encryption using assembly directives %ENCRYPT *KEY* and %PLAINTEXT respectively. The programmer does not have to store the whole program as enciphered in the memory. By using these assembly directives accompanied by assembly commands CRYPTON and CRYPTOFF, he/she can use the crypto feature just for the critical part of the program. Please also note that assembly commands CRYPTON and CRYPTOFF must be put right before their associating assembly directives as can be seen in Table II.

Design statistics and resource consumption of the design are shown in Table III.

TABLE III
FPGA Device Utilization for Crypto Processor

Logics	Amount	Percentage
Number of Slices	1020 out of 4656	21%
Number of Slice Flip Flops	815 out of 9312	8%
Number of 4 input LUTs	891 out of 9312	20%
Number of IOs	70	
Number of bonded IOBs	70 out of 92	76%
IOB Flip Flops	1	
Number of MULT18XSIOs	1 out of 20	5%
Number of GCLKs	1 out of 24	4%

We report here that the hardware implementation of CryptOdin utilizes only 21% of configurable slices, and 5% of the dedicated multipliers and 20% of the available LUTs on the Spartan3e-xc3s500e platform.

The total execution time of the sample program is 34 clock cycles that is 34 * 14.511 ns. The same design mapped to an ASIC is guaranteed to give even better rates.

V. Limitations, Conclusions and Future Work

A secure processor implementation was presented. The hardware realization of our secure processor CryptOdin was implemented using FPGAs, and verified with Xilinx simulation and co-simulation tools. The design was targeted at the Xilinx's Spartan3e-xc3s500e FPGA and simulated using ISE/ModelSim tool. The CryptOdin secure processor is best suited for applications with frequency in the range of 50 ~80 MHz.

Although CryptOdin was originally designed as 16-bit, it is quite straightforward and convenient to adapt the code as either 32-bit for higher performance or 8-bit for lightweight applications.

The design has several deficiencies. There is no internal code ROM. This makes it easy for the attacker to rewrite control programs. Also there is no internal cryptographically secure random number generator. A Blowfish Authentication Code modification method should be included for this purpose.

APPENDIX

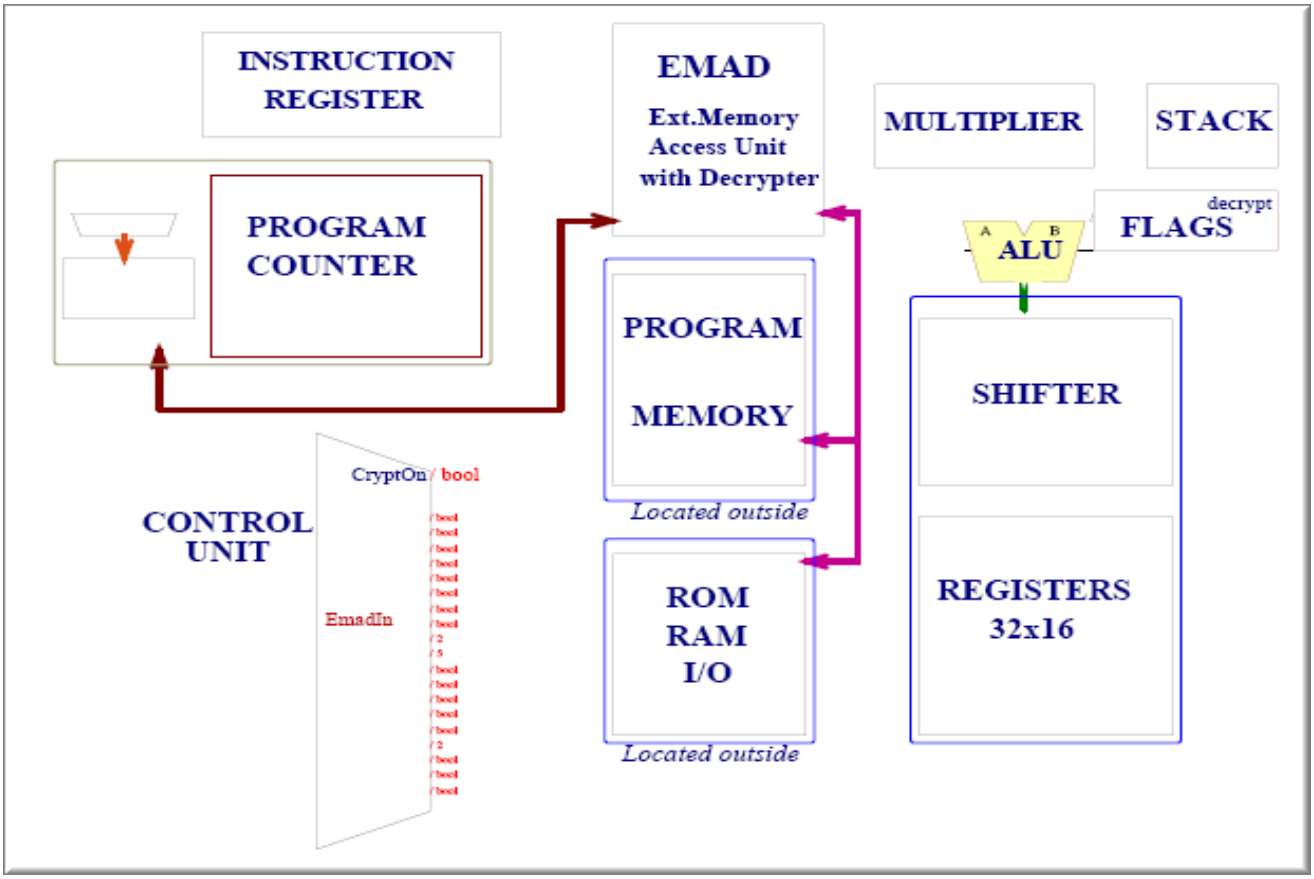

Fig. 1. The simplified schematic of the designed Microprocessor.

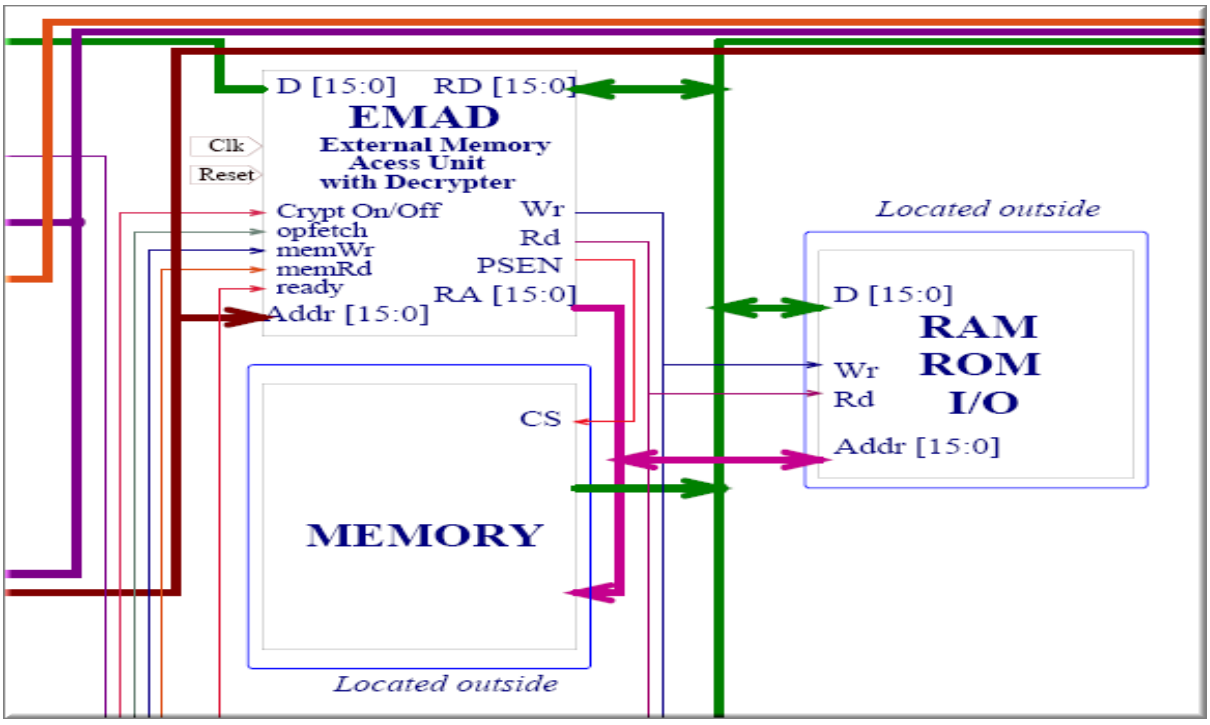

Fig. 2. The detailed schematic showing the connections between EMAD and external memory elements.

REFERENCES

[1] Kannavara, R.; Bourbakis, N.G, "Surveying secure processors", Potentials, IEEE, 2009, Volume: 28 , Issue: 1, pp. 28-34.
[2] Fan Mingyu, Wang Jinahua, and Wang Guangwei, "A design of hardware cryptographic co-processor," IEEE Syst., Man Cybern., June 2003, pp. 234–236.
[3] F. Crowe et al., "Single-chip FPGA implementation of a cryptographic co-processor," Proc. IEEE Int. Conf. Field-Programmable Tech., 2004 pp. 279–285.
[4] Carter W. S., "The future of programmable logic and its impact on digital system design," in IEEE Int. Conf. Computer Design: VLSI in Computers and Processors, 1994, pp. 10–16.
[5] Wolf W. "FPGA Based System Design", Prentice Hall PTR, New Jersey, 2004
[6] Yalamanchili S. 2001. Introductory VHDL, From Simulation to Synthesis, Prentice Hall.
[7] Xilinx. www.xilinx.com., 2010
[8] Anderson C. Z., d'Amore R., "A Low-cost FPGA Implementation of the Advanced Encryption Standard Algortihm", 15th symposium on Integrated Circuits and Systems Design (SBCC1'02)
[9] Morris K., "Embedded Dilemma". http://www.fpgajournal.com/articles/embedded.htm, November 2003.
[10] Michael C.-J. Lin, Youn-Long Lin, "A VLSI Implementation of the Blowfish Encryption/Decryption Algorithm," Asia and South Pacific Design Automation Conference, p. 1, Asia and South Pacific Design Automation Conference 2000 (ASP-DAC'00), 2000.
[11] Finch, P. J., "A Study of the Blowfish Encryption Algorithm", Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-21269., City University of New York, 1995.
[12] Schneier, Bruce, "Applied Cryptography Second Edition: Protocols, algorithms, and source code in C.", John Wiley & Sons, Inc., 1996
[13] Ashenden Peter J., "The Designer's Guide To VHDL", Academic Press, 2002
[14] Maxfield, C., "The Designer Warrior's Guide to FPGAs, Elsevier, 2004

Kadir Atilla Toker – was born in 1964 in Izmir. He has a BSc and MSc in Electrical and Electronics Engineering. He has worked for the Telecom-Izmir/Turkey-Telephone Exchange Engineer-, Alcatel SEL-Stuttgart/Germany-R&D Engineer-, Alcatel Telecom Norway-Oslo/Norway - R&D Engineer-, Kitron Development-Oslo/Norway-R&D Engineer-, Izmir Institute of Technology (IYTE) Izmir/Turkey-expert at the Communication and Information Technologies Research Centre- and he joined Izmir University in 2008, where he is working as an instructor.

He has been involved in various hardware and software development projects. His research interests include FPGA implementation of various systems like MIMO communication systems/Led TV, ASIC design, wind turbines, computer architecture and firmware/software design.

Tolga Ayav – was born in 1974 in Izmir. He has a BSc in Electrical and Electronics Engineering, MSc and PhD in Computer Engineering. He worked for EN-KO Ltd. Co. as an industrial automation engineer and Computer Application and Research Center at Izmir Institute of Technology as system analyst. He is currently assistant professor in the Department of Computer Engineering in Izmir Institute of Technology.