2010 International Conference on Reconfigurable Computing

HCrypt: A Novel Concept of Crypto-processor with Secured Key Management

Lubos GASPAR, Viktor FISCHER, Florent BERNARD, Lilian BOSSUET
Université de Lyon
CNRS, UMR 5516, Laboratoire Hubert Curien
F-42000, Saint-Etienne, France
{*lubos.gaspar, fischer, florent.bernard, lilian.bossuet*}*@univ-st-etienne.fr*

Pascal COTRET
Université de Bretagne-Sud
CNRS, UMR 3192, Laboratoire Lab-STICC
56321 Lorient, France
pascal.cotret@univ-ubs.fr

***Abstract*—The paper presents a novel concept of processor aimed at symmetric-key cryptographic applications. Its architecture is optimized for implementation of common cryptography tasks. The processor has 128-bit separated data and key registers, dedicated instruction set optimized for key generation and management, embedded cipher, and embedded random number generator. From an architectural point of view, the most important characteristic of the proposed crypto-processor is the physical separation of data and key registers and buses, insuring that confidential keys will never leave the system in clear. This way, the processor enables to separate protected and unprotected security zones easily and also achieve complete physical isolation of key management and data zones inside the single FPGA. The first version of the processor implemented in Xilinx Virtex 5 FPGA device achieves the frequency of 160 MHz and it occupies 1343 configurable logic blocks and 21 embedded memory blocks.**

***Keywords*-HCrypt; secured crypto-processor; register separation; key management;**

I. INTRODUCTION

Hardware cryptographic systems must fulfill contradictory requirements: fast parallel structures implementing computationally extensive cryptographic functions must co-exist with complex sequential structures used to implement cryptographic algorithms such as cipher modes, key management operations and cryptographic protocols. Implementation of cryptographic algorithms and protocols in hardware necessitates employing many complex state machines that make the logic very unstable and vulnerable. Furthermore, upgrades of a hardwired logic can become complicated, long and expensive. Because of this, cryptographic systems based on processor/coprocessor architectures were frequently employed in the past.

The most common solution employed in the past decade consists in the use of a general-purpose processor featuring one or more cryptographic co-processors. This solution permits to implement sequential algorithms (that evolve very frequently as a consequence of attacks and/or algorithm standard modifications) by the processor program, while the tasks that can be executed in parallel are implemented in the co-processor placed inside the same logic device. The biggest part of this research has been devoted to elliptic curve cryptography [1], [2]. A lot of attention has also been given to optimizations in symmetric key cryptography (block cipher coprocessors) [3], [4]. Combined cryptographic system based on NIOS II processor with ECC, SHA-1, RSA and AES co-processors has been proposed by Hani et al. [5].

Some architectures employ a general-purpose processor that is modified and optimized for implementation of cryptographic algorithms. Wu et al. [6] and Pereira et al. [7] used several processors based on Very-Long-Instruction-Word (VLIW) architecture, while profiting from parallelism to some extent. Other way for optimizing execution of cryptographic algorithms is to use a configurable architecture. The most common reconfigurable processors are COBRA [8], PipeRench [9], MorphoSys [10], GARP [11] and Xtensa [12].

The use of processors brings a security weakness in all of the above mentioned solutions: the processor manipulates the keys as ordinary data and modification (intentional or unintentional) of the program memory contents can enable reading the keys in clear outside the system. In [13] authors use two processors with different security level and different tasks. They create virtual zones inside a physical memory that is shared by both processors, in order to be able to separate red and black zones dedicated to key and data storage, respectively. The secure specific-purpose processor can access both private (red) and public (black) virtual zones and the general-purpose processor is allowed to access only public (black) virtual zone. However, both zones are located in the same physical memory thus still allowing certain types of remote attacks, such as protocol attacks [14].

In order to face protocol attacks and fault injection attacks, we propose a new architectural concept of the crypto-processor that allows physical separation of data and key registers and buses. The data flow between the embedded cipher and key and data registers is designed so that the confidential keys generated inside the system will never leave it in clear, even if the processor was subject of a protocol (modification of the program code) or fault injection attack (modification of the data flow).

The paper is organized as follows: Section II introduces application background of the global multi-processor system on chip containing the crypto-processor. Section III presents

978-0-7695-4314-7/10 $26.00 © 2010 IEEE
DOI 10.1109/ReConFig.2010.38

the novel architecture of the crypto-processor. Section IV describes the implementation of the processor in FPGA. Results are presented in Section V. Section VI discusses the results and Section VII concludes the paper.

II. General Context and Communication Needs

Embedded systems including Multi-Processor Systems on Chip (MPSoC) are employed increasingly in applications needing higher security levels [15], [16]. The crypto-processor described in this paper is proposed as a construction element of such a Secured Multi-Processor System-on-Chip (MPSoC). This system is studied with the objective of improving the security shield of reconfigurable technologies such as FPGAs at logical, architectural and system levels.

A. Architecture of the System

It is supposed that the proposed MPSoC will contain at least three types of processors (see Fig. 1): one or more general-purpose application processors, one or more crypto-processors and one security processor responsible for reconfiguration of the system. Besides processors, the system will include internal and external memories and one or more data interfaces such as Ethernet, USB, serial interface, etc. The architecture should be modular and sufficiently general in order to fulfill various security and performance requirements in numerous applications such as Set-top boxes, software radios, security video systems etc..

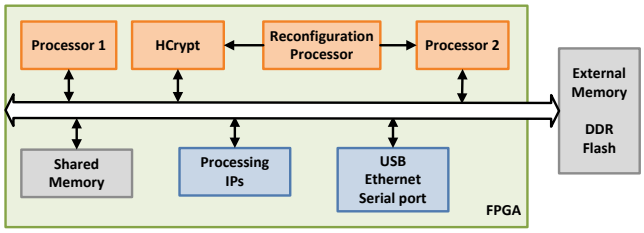

Figure 1: General system architecture

B. Communication Protocol

In order to maintain high order of modularity and connectivity to external networks, the embedded processors are supposed to communicate between them using network protocol and packets. The crypto-processor is responsible for ciphering (or deciphering) internal and external data frames coming via the communication ports. The ports for plaintext and ciphertext packets are separated.

III. HCrypt: A Novel Concept of Crypto-processors

Next, we will describe our novel concept of a secure crypto-processor called HCrypt.

A. Processor Structure and Data Flow

General principle of the processor is illustrated in Fig. 2. Dedicated architecture is composed of two sets of registers and buses: common data register set connected to common data bus and dedicated key data register set connected to dedicated key data bus.

The keys are saved in clear in key data registers. When reading the key from a key register, the key data is enciphered before being saved in general-purpose data registers. In the same way, key blocks manipulated as cipher-text are deciphered before being saved in the key memory. This way, the key data can never leave the processor in clear (even by modifying anyhow the program of the processor). Session key management is illustrated in the Fig. 3.

The key data memory is a true dual port memory. In this memory, the keys are accessible as data using one bi-directional port for reading and writing after ciphering/deciphering (as mentioned before), but also as effective encryption/authentication keys (in clear) during encryption/decryption process via one dedicated one-directional port. Note that this second port is independent from the data buses (common or key data bus).

At least, two session keys and master keys are available in the proposed processor: one for data (and key) enciphering/deciphering and one for data (and key) authentication. While manipulating ordinary data, the session keys will be used. When manipulating the session keys, the master keys will be used.

The processor contains a true-random number generator that is used to generate session keys. Note that before saving the key to the key data registers, the generated random data is post-processed by deciphering the generated block using a master key.

The processor contains also two blocks guaranteeing the flexibility: the program memory block and the cipher/decipher block. The contents of the program memory could be eventually changed using the security processor from Section II that is responsible for the system reconfiguration. We recall that the cipher/decipher is considered as a black box in this architecture and that it does not have a direct impact on the control logic and the data flow. The processor can thus use any encryption/decryption algorithm, with selected side-channel attack countermeasures.

B. Instruction Set

Besides common instructions, the instruction set takes advantage of the processor specific structure. Namely, it uses different instructions for handling ordinary data and keys. Also, arithmetic and logic instructions are optimized for implementing common cryptographic functions and modes.

The instruction set is be divided into four main groups:

- Data transfer instructions: data manipulation between data registers and input/output FIFOs

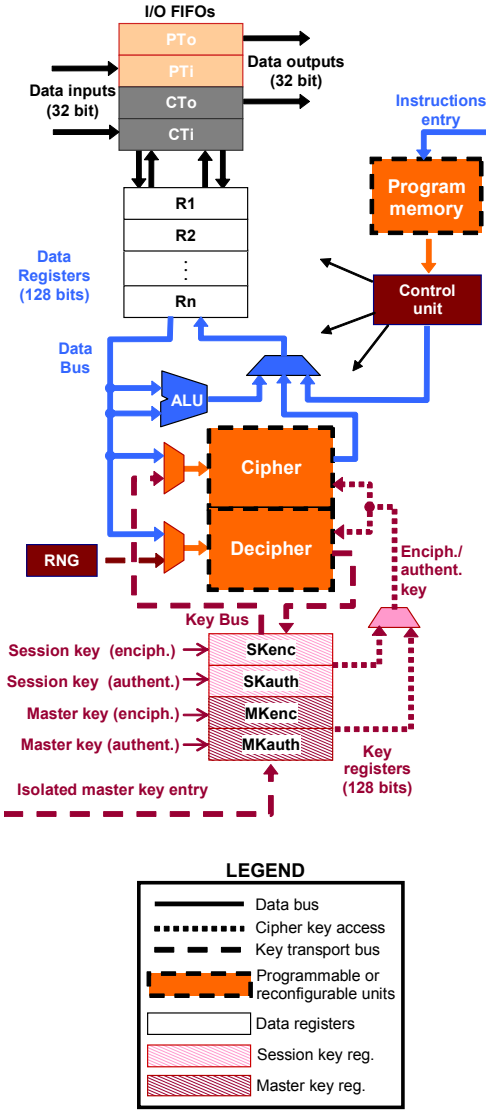

Figure 2: Concept of HCrypt

- Instructions for key generation and transport between ordinary data registers and key data registers
- Data processing instructions: optimized for implementation of cipher modes and key management
- Control instructions: aimed at the program control (branching)

The proposed instruction set with respect to this grouping is shown in Tab. I.

C. Assembler - Compiler

Complex instruction set makes writing programs in machine code more difficult. Long machine code is only hardly readable. In order to address this issue, we developed a processor-specific assembler (compiler) tool. In the case of the processor in development, frequent changes in hardware architecture (changes in instruction set) impose frequent changes in assembler. For this reason, we decided to define the instruction set of the assembler in an easy to read separate text file that is taken into account during compilation.

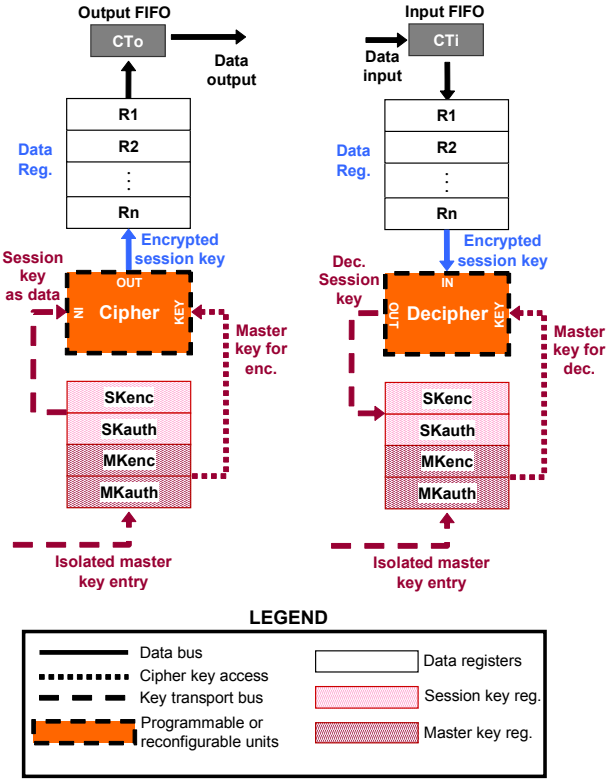

Figure 3: Session key management (left: encryption and transmission, right: reception and decryption)

D. Examples of Execution of Basic Modes

The operation of the processor can be illustrated on the program examples in Tab. II that realize various basic encryption modes. In this table, PTI means plaintext input, PTO plaintext output, CTI ciphertext input and CTO ciphertext output.

1) CBC Encryption Block Cipher Mode Example: The code illustrated in Tab. II (part A) enables to encrypt the packet consisting of N 128-bit data blocks (N is given in the first data block in the packet) in the CBC mode (see NIST SP 800-38A, p. 10 [17]).

2) CFB Encryption Block Cipher Mode Example: The code illustrated in Tab. II (part B) enables to generate the session key that is then used to encrypt the packet consisting of N 128-bit data blocks (N is given in the first data block in the packet) in the CFB mode (see NIST SP 800-38A, p. 12 [17]).

Table I: The dedicated instruction set

A) Data transfer instructions

Instruction	Description
init #,rx	Initialize Rx with a constant
getpt rx	Get one block from plain-text input FIFO and put it to Rx
putpt rx	Put data from data reg. Rx to the plain-text output FIFO
getct rx	Get one block from cipher-text input FIFO to data register Rx
putct rx	Put data from data register Rx to cipher-text output FIFO
move rx, ry	Move the contents of data register Rx to data register Ry

B) Instructions for key generation and manipulation

Instruction	Description
genk kx	Generate the key and save it in the key data register Kx
getk rx, ky	Decipher encrypted key from data reg. Rx to key reg. Ky
putk kx, ry	Encipher the key from key reg. Kx to data reg. Ry

C) Data processing instructions

Instruction	Description
enc rx,ry	Encipher the contents of Rx and save it in Ry
xor rx,ry	Rx XOR Ry and save the result in Ry
dcr rx	Decrement Rx
clr rx	Clear Rx
cmp rx, ry	Compare Rx with Ry, set the flag Z accordingly

D) Control instructions

Instruction	Description
goto displ	Unconditional branching
bneq displ	Conditional branching if not equal (Z = 0)

Table II: Block encryption modes

A) CBC encryption mode

```
      getpt   r3      ;Get N from PTI→R3
      getpt   r1      ;Get IV from PTI→R1
      putct   r1      ;Send IV to CTO
cbce: getpt   r2      ;Get data block from PTI→R2
      xor     r1, r2  ;R1 XOR R2→R2
      enc     r2, r1  ;Encipher R2→R1
      putct   r1      ;R1→CTO
      dcr     r3      ;Decrement R3 (R3 - 1→R3)
      bneq    cbce    ;Branch to cbce if not zero
```

B) CFB encryption mode

```
      genk    kx      ;Generate new session key
      putk    kx, r1  ;Encipher session key and put it to R1
      putct   r1      ;Send session key to CTO
      getpt   r3      ;Get N from PTI→R3
      getpt   r1      ;Get IV from PTI→R1
      putct   r1      ;Send IV from R1 to CTO
cfbe: enc     r1, r2  ;Encipher R1→R2
      getpt   r1      ;Get plain-text block from PTI→R1
      xor     r2, r1  ;R2 XOR R1→R1
      putct   r1      ;Put R1 to CTO
      dcr     r3      ;Decrement R3 (R3 - 1→R3)
      bneq    cfbe    ;Branch to cfbe if not zero
```

IV. Implementation of HCrypt

A. Specifications

As previously mentioned, the objective is to physically separate registers and buses carrying keys from those carrying data. For this reason, two sets of registers are implemented, so that keys and data are stored separately. In order to achieve higher performance, the processor has a 128-bit datapath.

HCrypt exchanges data with external/internal environment using input/output FIFOs. The processor is able to implement a serial communication protocol using packets. It permits to analyze and create packets efficiently. Data processing operations needed for implementation of cipher modes are carried out in a special-purpose arithmetic logic unit (ALU) that constitutes an important part of the processor.

For security reasons, the key lifetime should be limited (the same key should be used only for a limited amount of data). One of adopted solutions is to use two hierarchical key levels: master keys and session keys. The session keys are used to encrypt data and they are generated inside the system. In order to enable data decryption, the session key has to be exchanged with the communication party. It is therefore encrypted by the master key (shared with the party) and sent together with encrypted data. In order to generate keys inside the system, true random number generator (TRNG) is embedded in the processor.

Although the cipher/decipher blocks should be included in the processor's datapath, they are considered as black boxes in our project. We used AES cipher and decipher for testing purposes.

B. Design of the HCrypt Processor

The crypto-processor consists of the following parts:

- Datapath
  - Data manipulation part
  - Key management part
  - Key generation part - TRNG
- Control logic
- Input/Output FIFOs

1) Datapath: The datapath is the most area- and performance-critical path of the crypto-processor. It consists of data bus, data registers and ALU. Data registers are implemented in four 32 bits wide true dual-port RAM blocks. They are accessible through two 128-bit data ports. The ALU has a 128-bit datapath. It carries out all arithmetic and logic operations (XOR, AND, NOT, CMP, ROR, ROL, CLR, INC, DEC, ADD, SUB) necessary for implementation of most encryption/decryption modes. The authentication modes are not taken into account, yet.

Key data bus and key data registers for the key management are shown in Fig. 2. The Master key (aimed at

Session key encryption/decryption) is stored in the Master key register. This register is capable of storing two 128-bit keys: one for encryption and one for decryption. This register is implemented in the logic area of the FPGA. Before the operation, the master key has to be transferred to the crypto-processor via a 32-bit separated data bus by the Security processor from Session II.

Session key registers are implemented using two 128-bit dual-port RAM blocks. Session keys can be generated inside the processor using embedded TRNG or they can be received by the crypto-processor in an encrypted form and subsequently decrypted using the master key. If generated inside the processor, the session key has to be encrypted using a master key and sent out according to the protocol.

Note that multiplexers included in the datapath are organized in such a way, that unintentional or intentional transfers of unencrypted keys outside the crypto-processor would never be possible.

The cipher and decipher blocks that implement the AES encryption algorithm in their current version, are connected to the processor from outside. The keys and data are transported to them by two separate 128-bit buses. Encrypted/decrypted data are registered in cipher/decipher blocks in 128-bit registers accessible by the processor.

This first version of the datapath is partially optimized for the Xilinx Virtex 5 FPGA. In the future, further optimizations will be carried out and authentication will be implemented. In addition, the crypto-processor will be optimized for other FPGA technologies as well.

2) Control Logic: The first version of the control logic is responsible for instruction fetching, decoding and execution. Instructions are carried out in 2, 3, 6 or more clock cycles, depending on the type and complexity of the operation. The pipelining is not implemented. However, it is supposed to be implemented in the future. Instructions are 32-bit wide. Current instruction set consists of 32 instructions.

The program is stored in the program memory that is initialized by an external 32-bit interface. The program memory is implemented as one dual port RAM block capable of storing 1024 instructions. The program memory can be expanded if necessary (by concatenating two or more memory blocks).

3) Input/Output FIFOs: There are four input/output FIFOs in the current version of the processor. They serve as input/output data ports, which convert 32-bit data words into 128-bit words used inside the crypto-processor and vice versa. Additionally, the FIFOs allow interfacing external clock domains with the internal clock domain of crypto-processor.

V. Results

HCrypt was described in VHDL language using Xilinx ISE Web ver. 11.1 and implemented in Xilinx Virtex 5 XC5VLX110T FPGA. System utilizes just fine-grain FPGA resources and embedded RAMs/FIFOs as illustrated in Tab. III. Functionality was simulated using Xilinx ISim tool. Subsequently, timing simulation was carried out. The period estimated from the longest critical path was 6.45 ns, thus the highest clock frequency was 155 MHz.

Table III: Utilization of resources in XC5VLX110T

	Slices		RAM(36kb)/FIFO	
Total	17280	100.0%	148	100.0%
Crypto-Proc.	1343	7.7%	21	14.2%
AES Cipher	286	1.6%	9	6.1%
AES Decipher	361	2.1%	9	6.1%

The crypto-processor was then tested in Virtex 5 XC5VLX110T FPGA using the XUPII development board. In the first stage of tests, the processor was initialized by the master key and the program. Afterwards, data packets were written into the input plain-text FIFO. Next, the crypto-processor carried out encryption in the CFB128 AES encryption mode and the resulting packets were written to the ciphertext output FIFO. A stable operation has been achieved at the clock frequency up to 160 MHz, thus confirming estimations.

Assembler compiler tool with configurable instruction set was also developed. Various short programs were written for testing purposes. Assembler was tested by compiling source code for HCrypt. No compilation errors were observed.

VI. Discussion

The main difficulty concerning the portability of the proposed VHDL code concerns embedded memory employment. As mentioned above, current processor was mapped to Xilinx Virtex 5 device where each true dual port memory block has two 32-bit input and output ports. However, in Altera Stratix III FPGA each true dual port memory block has two 16-bit input and output ports, thus twice more memory blocks have to be instantiated in order to maintain the same width of the datapath (128-bit). Moreover, when adapting to Actel Fusion or SmartFusion devices, where each embedded true dual port memory block has two 8-bit input and output ports, 16 memory block have to be instantiated to maintain the same width of the datapath.

It has been observed that current cipher and decipher blocks constitute the longest critical path, therefore optimization of the cipher can result in better performance. Some long critical paths were also reported in the ALU, thus this part has to be optimized in the future as well.

Resource distribution report showed that the most logic resources are dedicated to ALU. On the contrary, the most embedded RAM blocks were used for Input/Output FIFO instantiations, in order to interconnect 32-bit external buses with 128-bit internal datapath with different clocks.

In the current version, only basic encryption block cipher modes are supported [17]. Flexibility of the architecture al-

lows its further extension, in order to implement new authentication modes (such as CMAC) and combined encryption and authentication modes (such as CCM and GCM).

To support a future discussion, HCrypt project is available on the SecReSoC project webpage: *http://labh-curien.univ-st-etienne.fr/secresoc/*

VII. Conclusion

We proposed a novel architecture of the crypto-processor with secured key management. The architecture is based on physical separation of registers and buses aimed at key management from registers and buses aimed at data exchange and data processing. The physical separation of the processor blocks increases its robustness against protocol and fault injection attacks. Cipher and decipher blocks are independent from the processor structure and can be easily upgraded or secured by including side-channel countermeasures. Internal structure of the processor and its arithmetic-logic unit are optimized for cipher modes implementation enabling all the necessary mathematical operations. Moreover, the processor can easily analyze and create packets needed in communication protocols and it is thus suitable as a component in multi-processor networks. In order to achieve higher level of data security, only two-level symmetric-key management have been considered up to now. Nevertheless, the structure of the processor will be soon extended in order to support also the asymmetric key cryptography.

Acknowledgment

The work presented in this paper was realized in the frame of the SecReSoC project number ANR-09-SEGI-013, supported by the French National Research Agency (ANR).

References

[1] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, "Multicore curve-based cryptoprocessor with reconfigurable modular arithmetic logic units over GF (2^n)," *IEEE Transactions on Computers*, pp. 1269–1282, 2007.

[2] M. Machhout, Z. Guitouni, K. Torki, L. Khriji, and R. Tourki, "Coupled FPGA/ASIC Implementation of Elliptic Curve Crypto-Processor," *International Journal*, vol. 2.

[3] F. Crowe, A. Daly, T. Kerins, and W. Marnane, "Single-chip FPGA implementation of a cryptographic co-processor," in *2004 IEEE International Conference on Field-Programmable Technology, 2004. Proceedings*, 2004, pp. 279–285.

[4] Y. Eslami, A. Sheikholeslami, P. Gulak, S. Masui, and K. Mukaida, "An area-efficient universal cryptography processor for smart cards."

[5] M. Hani, H. Wen, and A. Paniandi, "Design and implementation of a private and public key crypto processor for next-generation it security applications," *Malaysia Journal of Computer Science*, vol. 19, no. 1, pp. 29–45, 2006.

[6] L. Wu, C. Weaver, and T. Austin, "CryptoManiac: a fast flexible architecture for secure communication," *ACM SIGARCH Computer Architecture News*, vol. 29, no. 2, pp. 110–119, 2001.

[7] F. Pereira, E. Ordonez, and R. Chiaramonte, "VLIW Cryptoprocessor: Architecture and Performance in FPGAs," *IJCSNS*, vol. 6, no. 8A, p. 151, 2006.

[8] A. Elbirt and C. Paar, "An instruction-level distributed processor for symmetric-key cryptography," *IEEE Transactions on Parallel and Distributed Systems*, pp. 468–480, 2005.

[9] S. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. Taylor, and R. Laufer, "PipeRench: a co/processor for streaming multimedia acceleration," in *Proceedings of the 26th annual international symposium on Computer architecture*. IEEE Computer Society, 1999, pp. 28–39.

[10] H. Singh, M. Lee, G. Lu, F. Kurdahi, N. Bagherzadeh, and E. Chaves Filho, "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications," *IEEE Transactions on Computers*, vol. 49, no. 5, pp. 465–481, 2000.

[11] J. Hauser and J. Wawrzynek, "Garp: A MIPS processor with a reconfigurable coprocessor," in *IEEE Symposium on FPGAs for Custom Computing Machines*, vol. 33. Citeseer, 1997.

[12] R. Gonzalez, "Xtensa: A configurable and extensible processor," *IEEE Micro*, vol. 20, no. 2, pp. 60–70, 2000.

[13] A. Ashkenazi and D. Akselrod, "Platform independent overall security architecture in multi-processor system-on-chip integrated circuits for use in mobile phones and handheld devices," *Computers & Electrical Engineering*, vol. 33, no. 5-6, pp. 407–424, 2007.

[14] R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov, "Cryptographic processors - a survey," *Proceedings of the IEEE*, vol. 94, no. 2, pp. 357–369, 2006.

[15] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a new dimension in embedded system design," in *Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, p. 760.

[16] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 3, pp. 461–491, 2004.

[17] M. Dworkin, "NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation-Methods and Techniques," *National Institute of Standards and Technology/US Department of Commerce*, 2001.