A Reconfigurable Crypto Sub System for the Software Communication Architecture

Michael Grand, Lilian Bossuet, Bertrand Le Gal and Dominique Dallet
Laboratoire IMS
University of Bordeaux
first_name.last_name@ims-bordeaux.fr

Guy Gogniat
Laboratoire LabSTICC
University of Bretagne Sud
guy.gogniat@univ-ubs.fr

***Abstract*—Upgradeability and interoperability are main concerns of Software Defined Radio (SDR). But in the case of military applications, security is also a relevant aspect of SDR. The Secure Software Communication Architecture (SSCA) is a standardized solution to secure SDR. This architecture needs a cryptographic processor for security purposes. However, currently available SSCA compliant ASIC-based cryptoprocessors limit radio set upgradeability and interoperability. In this paper, we describe the first non-confidential reconfigurable cryptoprocessor architecture for SSCA. We provide some area estimation of processor main parts on Xilinx Virtex 4 FPGA.**

I. Introduction

The concept of *Software Defined Radio* appears in 70's in USA and Europe. Currently, radio set development faces at least two major issues. Firstly radio set shall support a multitude of different waveforms, secondly waveform specifications have a very high renewal rate (e.g. different versions of WIMAX based waveforms). As a result, SDR development was mainly motivated by the need to ensure radio set upgradeability and interoperability. Additionally, software components are often built from proprietary interfaces, as a consequence portability and compatibility are issues frequently encountered in SDR designs.

In 1997, the U.S. government launched the *Joint Tactical Radio Software Program* (JTRS). The main goal of this program was to standardize a new software architecture for SDR in order to improve software component portability. This architecture, named *Software Communication Architecture* (SCA) [1], is an open set of object interfaces compliant with the *Common Object Request Broker Architecture* (CORBA) which is a software architecture for distributed computing.

Even if the SCA was mainly developed for military purposes, first drafts did not take security into account. To solve this issue, the U.S. government published the SCA Security Supplement. This supplement introduces a new set of interfaces and specifications which allows securing SDR design. Currently, the last draft (V. 2.2.2) of this document is kept confidential by the U.S. government. The last publicly available draft (2.2.1) [2] was published in 2004. Recently, a new program, managed by the EDA (European Defence Agency), was launched. This program, named *European Secure SOftware Radio* (ESSOR), intends to complete the SCA standard and to propose an European alternative to the U.S. Security Supplement.

The secure architecture proposed by the Security Supplement is mainly designed with a *Crypto Sub System* (CSS) module. The CSS must provide functional and electrical isolation between red (confidential data) and black (unconfidential or encrypted data) radio parts. The CSS acts as a cryptographic boundary. Data streams between these two parts are encrypted and authenticated by the CSS.

Currently available ASIC-based solutions lack of flexibility. This significant drawback reduces the global architecture upgradeability and interoperability. In this paper, we introduce a new kind of CSS based on reconfigurable chips such as FPGA. Reconfigurability gives us the possibility to improve and correct hardware components. By this way, product vulnerability is reduced, radio interoperability and radio upgradeability are improved.

This paper is organized as follows. Section 1 introduces the concept of Secure SDR. Then in Section 2, the *Secure SCA* (SSCA) is presented. Section 3 describes the CSS architecture and Section 4 lists available related works on CSS architecture. Our new architecture is detailed in Section 5. Section 6 details expected characteristics of our architecture. Section 7 concludes this paper and lists remaining work.

II. Secure SCA

A. Attack Types

A SDR must be protected against several kinds of attacks. Two different kinds of attacks can be identified:

1) Attacks Against Radio Transmissions: Electromagnetic waves move freely on the air so they can be intercepted by an attacker. The attacker could be passive or active. A passive attacker intercepts radio transmissions and tries to decode them while an active attacker not only intercepts but also alters radio transmissions.

Passive attacks are prevented by the use of message encryption whereas active attacks are prevented by the use of message authentication and message integrity checking. Cryptographic components usually provide these three services.

978-1-4244-5239-2/09/$26.00 ©2009 IEEE

2) Attacks Against Radio Sets: If radio transmissions are encrypted and authenticated, it may be very difficult for an attacker to obtain useful information from such signals. Sometimes, radio set is the most interesting target for an attacker. Secret keys, certificates and other sensitive information can be stolen from the radio. Therefore, radio transmissions become insecure until keys are changed.

Radio set may be protected against such type of attacks in four complementary ways:

- *Use of standard cryptographic services.*
- *Use of hardware security.*
- *Use of security policy.*
- *Use of anti-virus software.*

B. The Secure SCA Architecture

The SCA Security Supplement is based on the red/black isolation concept. A set of interfaces and specifications standardizes the process used to secure a SCA compliant radio. The security supplement does not give implementation details to developers. A secure SDR must include features like:

- *Standard cryptographic services*
- *Security policy enforcement*
- *Keys and certificates management*
- *Access control*
- *Configuration management*
- *Alarms and auditing*
- *Memory management*

The security supplement requires some of these features to be implemented as hardware components. Practically hybrid implementations (software with hardware accelerators) are used to improve the performance-cost trade-off.

Communication through the Red/Black isolation is controlled from the SSCA interfaces. Following sections present these two concepts:

1) Security Interfaces: The SCA Security Supplement provides a set of security interfaces which are described with the *Interface Definition Language* (IDL) introduced by the *Object Management Group* (OMG). These interfaces allow the management of security components embedded in a SSCA compliant radio. If security hardware component is CORBA compliant (like *General Purpose Processor* (GPP)), then interfaces can be directly implemented on it. If not, these interfaces act like software proxies which allow a standard mean of communication between GPPs and non-CORBA compliant security devices.

2) Red/Black Isolation: A Red/Black isolated circuit is split in two parts:

- A **Red** part manages confidential data which is generated by the data source (e.g. voice, video, ...).
- A **Black** part manages unconfidential or encrypted data sent to or received from the unsecured physical communication channel.

Red/Black isolation involves electric and functional isolation. Electric isolation between red and black area limits side-channel attacks from radio black parts. Functional isolation is provided by the cryptographic boundary which is named *Crypto Sub System* (CSS) in the Security Supplement. Data streams between red and black radio areas are encrypted and authenticated by the CSS.

III. The Crypto Sub System

A. The CSS as a Secure Bridge

The security supplement describes the CSS like a gateway between red and black radio parts and all exchanged data pass through it. Data pass through one or more channels, each channel has its own cryptographic key and protocol and is identified by an unique ID. CORBA messages encapsulate data streams, these messages are composed of headers and data blocks. When a CORBA message goes through a channel, only payload have to be encrypted, headers must remain in plain text in order to be checked by the bypass unit. Messages are filtered according to a set of security rules. Such behaviour needs the CSS to be split in two parts, a bypass unit and a cryptographic unit, as shown in Fig. 1.

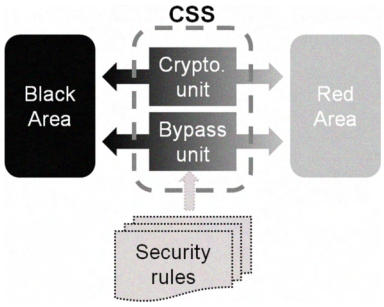

Fig. 1. CSS as Red/Black Bridge

B. Message Typology

CSS has to deal with three different kinds of messages which use the CORBA *Object Request Broker* (ORB) [3] as software bus and the *General Inter-ORB Protocol* (GIOP) as communication protocol.

1) Internal CORBA Messages (Type 1): They are used for communication between CORBA objects. For example, the CORBA *NamingService* is used as a server which lists all available components in a CORBA architecture. When a CORBA client tries to establish a connection to a remote server, the client needs to ask to the *NamingService* a reference to the server. *NamingService* may be located on red area whereas client is in black area.

2) SCA Dependant Messages (Type 2): They are used for communication between SCA components. For example, the *DomainManager* [1] needs to communicate with the *DeviceManager* [1] while radio is being initialized.

3) Waveform Dependant Messages (Type 3): They correspond to data streams in the SDR (e.g. video, voice data stream).

The Security Supplement does not precisely explain how Type 1&2 messages are transmitted through the CSS. However, SSCA interfaces provide several functions (Encrypt, Decrypt and Transform) which standardize Type 3 message transmissions between software waveform components and the CSS. Anyway, each message type requires different bypass policies and processing steps (Tab. I) because it carries different data types.

TABLE I
PROCESSING UNIT PER MESSAGE TYPE

Message Type	Bypass Unit	Cryptographic Unit
Type 1	Yes	No
Type 2	Yes	No
Type 3	Yes	Yes

C. CSS Functional Architecture

The Security Supplement suggests a functional architecture for the CSS. This architecture is designed with three main units, one bypass unit, one or more cryptographic units and one security manager. Fig. 2 shows a schematic representation of the architecture proposed by [2].

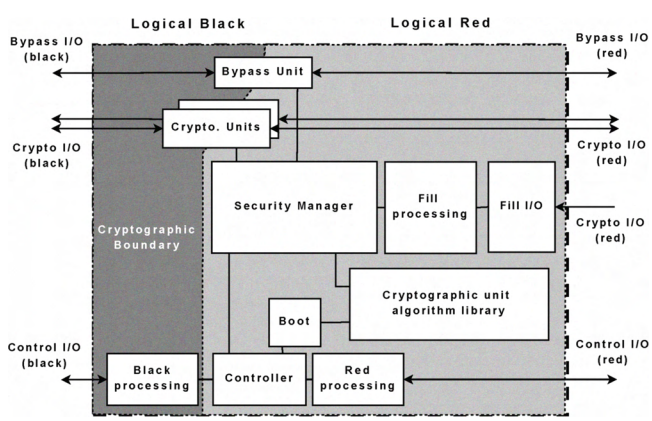

Fig. 2. CSS Functional Architecture

1) The Bypass Unit: All data, which have not to be encrypted, must use the bypass unit to go through the CSS. Bypassed data may be status and/or control data (e.g. GIOP headers) or user defined bypassable data which are embedded in type 3 messages. The bypass unit filters incoming data according to a set of security rules. Security rules shall define:

- *Valid bypassable data type*
- *Valid bypassable data source*
- *Valid bypassable data destination*

The bypass unit must reject unusual and unauthorized incoming data by closing the corrupted channel and informing the user.

2) The cryptographic unit: The cryptographic unit is used to encrypt or decrypt message bodies. It manages message authentication and integrity checking. It can decrypt and authenticate software components used in the SCA architecture or in the instantiated waveform. This unit needs to embed a wide range of symmetric and asymmetric algorithms (public or not). The needed cryptographic keys and protocols are selected from security interfaces provided by the Security Supplement.

3) The security manager: This is the CSS manager. The security manager (SM) provides keys and algorithms management, security policy enforcement and alarm processing. Encrypted keys and algorithms are loaded into the CSS from the fill port with DS-101 or DS-102 confidential protocols [2]. Then SM loads keys and algorithms on cryptographic unit and bypass unit. The SM is controlled by Security Supplement Interfaces through the control port.

IV. RELATED WORKS

There are few concrete implementations of SSCA compliant CSS [4]. Harris Corporation, General Dynamics and Raytheon provide such cryptographic chips:

- ***Sierra II:*** In [5], Harris Corp. presents a Secure SDR radio which uses the NSA certified Sierra II ASIC. Key management and cryptographic algorithms are embedded in the ASIC device whereas bypass unit, policy enforcement, alarms and secure MMU are implemented with additional logic. Both hardware and software are used to implement cryptographic algorithms like SHA, AES, etc. According to the datasheet [6], this chip may handle data rates up to 300 Mbps.
- ***AIM:*** In [7], a secure SDR using the General Dynamics (Motorola) Advanced INFOSEC Machine is presented. This NSA certified chip handles data rates up to 20 Mbps and multiple channels (up to 1024). This chip integrates a key management unit, a specific fill interface and three cryptoprocessors [8].
- ***Cornfield:*** The Raytheon security chip [9] can process data rates up to 40 Mbps per channel. Eight algorithms and sixty four keys can be concurrently stored on the chip. Keys and algorithms can be loaded from DS-102 and DS-102 compliant fill port interface.

All these cryptographic chips are ASIC based chip which allow low consumption and high performances. However such technology leads to invariable architecture, so discovered product vulnerabilities cannot be corrected. In the next section, this paper introduces a new kind of reconfigurable CSS that overcomes this problem.

V. A Reconfigurable Crypto Sub System

A. CSS Architecture

There is no publicly available CSS detailed implementation. Proprietary architectures of chips presented in the previous section are kept confidential. This paper intends to propose the first open CSS architecture. This new architecture is especially designed for FPGA technologies but concept could be reused for ASIC chip. This architecture benefits from FPGA reconfiguration ability in order to provide an highly upgradeable and reconfigurable CSS.

Upgradeability is a consequence of the FPGA reconfiguration ability. The radio set could upgrade the CSS bitsream each time an update is available on a remote server. Partial reconfiguration technology, which is available on Xilinx SRAM FPGA, allows the design of dynamically reconfigurable CSS.

The CSS provides cryptographic and security services to the SSCA radio set. It handles three different processing modes:

- *Red/Black mode for red/black communication.*
- *Red/Red mode for red area cryptographic needs.*
- *Black/Black mode for black area cryptographic needs.*

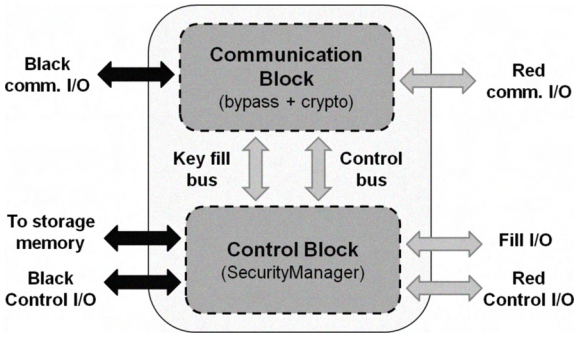

Fig. 3. CSS Architecture

This architecture is divided in two blocks, a *Communication Block* (CommB) and a *Control Block* (ContB), as shown in Fig. 3. The CommB is made of bypass and cryptographic units whereas the ContB is used as *Security-Manager*. ContB is managed through red and black *Control Bus* from Security Supplement interfaces. Red and black communication I/O convey data streams through the CSS. The ContB controls and configures the CommB through the key fill bus and the internal control bus. The specific internal key fill bus permits to enforce keys isolation from other data in the CSS. The internal control bus transmits bypass rules and software to the communication block.

A storage memory (e.g. flash memory) is used as permanent storage device. All data (e.g. keys, certificate, algorithms and programs) are encrypted and stored in this memory. Furthermore cryptographic integrity checking is used while data is loaded from this memory.

B. The Communication Block

1) The Communication Block Architecture: As shown in Fig. 4, the CommB is built on a 32 bits CPU. A *Dynamically Reconfigurable Cryptographic Accelerator* (DRCA) is used by the CPU for cryptographic processing. DRCA configuration is set on-the-fly by the ContB through the *Internal Reconfiguration Access Port* (ICAP) [10] embedded in Xilinx FPGA. As a consequence only one Cryptographic Accelerator is needed by the CommB. To use another cryptographic accelerator the ContB reconfigures the DRCA with a specific partial bitstream which achieves the CommB algorithm requirements.

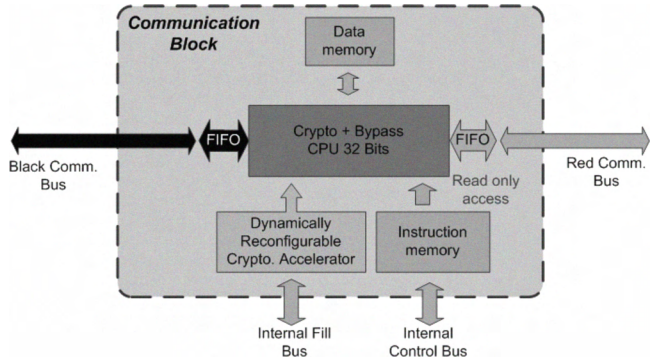

Fig. 4. CSS Communication Block

An hybrid implementation has been chosen for cryptographic algorithms. For example, in the case of the AES algorithm, key diversification and AES mode (CBC, CCM, ...) are implemented in software whereas AES cipher block is implemented in hardware. Only intensive computing is made by hardware. As a result hardware occupation on FPGA is reduced. Moreover software implemented cipher mode can be easily changed according to user needs.

Data are sent through red and black communication buses to the radio red and black parts. This specific architecture needs a custom CPU which has two specific ports, a red one and a black one. CPU design shall restrict access to these ports in order to enforce red/black data isolation. Instructions and bypass rules are read from a dedicated memory. This memory is in read only access from the CommB and in read/write access from the ContB. Keys are directly loaded into the DRCA by the internal fill

bus, the CommB CPU cannot have access to this bus, so keys are completely isolated from it. A specific memory is used by the CommB CPU for software processing. SSCA specifications suggest the use of secure MMU for memory access in order to enforce space memory isolation between CommB CPU processes.

2) Message Transport and Processing: Usually CORBA ORBs use the *Internet Inter Orb Protocol* (IIOP) which is a specialization of the GIOP communication protocol for TCP/IP transport layer. However CSS is not necessarily TCP/IP enabled. Additionally, TCP/IP is not suitable for communications between several processors on the same chip. This issue may be solved in two different ways, either by the use of an *Environnement Specific Inter Orb Protocol* (ESIOP) developed in compliance with OMG standards, or by the use of proxies as explained in the SCA security supplement.

Regardless which transport layer is used, message processing can be broken down to several steps. For Red to Black exchange, steps consist in:

1) *Message Unpacking:* Message has to be unpacked from the transport protocol.
2) *Message Parsing:* In this step, headers and body are extracted from the message content.
3) *Headers Validation:* Then headers have to be validated according to security policy rules. If CommB detects unusual or invalid headers, it rejects the message and closes the communication channel. Security policy rules are stored in a read only memory to avoid rules corruption.
4) *Body Encryption or Validation:* If message is a Type 3 message and if headers are validated then the message body is encrypted and/or signed according to properties of the selected channel. In the case of message Type 1 or 2, body message has to be validated before message forwarding.
5) *Message Packing:* Message is packed and sent to the final destination.

For Black to Red, Red to Red or Black to Black exchanges, only the fourth step has to be adapted to specific needs.

C. The Control Block

1) The Control Block Architecture: The CommB is controlled via the ContB through security interfaces. ContB is built on a 32 bits CPU with several peripherals plugged on a red or black bus according to their security level needs, as shown in Fig. 5. Keys are stored in a special secure register allowing key isolation from other data. The ContB has its own secret key stored in the key register. Key, algorithms and software are read from the Fill I/O or the red control bus. Then, they are authenticated and integrity

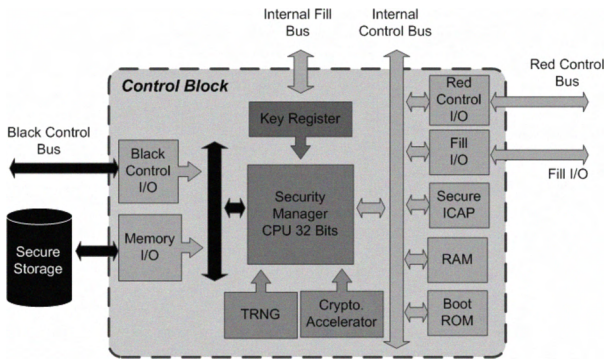

Fig. 5. Control Block Architecture

checked by the CPU. A *True Random Number Generator* (TRNG) and a cryptographic accelerator are used by the ContB CPU for key generation and cryptographic services. ContB red peripherals are:

- *Red Control I/O:* This unit is used to receive order from radio set red side.
- *Fill I/O*: Algorithms, key and certificates can be loaded into the CSS through this interface. Security Supplement suggests the use of DS-101 and DS-102 confidential protocols. All data incoming from fill I/O must be decrypted and authenticated.
- *Boot ROM:* The security manager boot loader is stored in this ROM, its integrity must be guaranteed. The boot loader must decrypt and authenticate the main software loaded on the ContB CPU.
- *RAM:* It corresponds to internal FPGA RAM. If external RAM is needed, it must be plugged to the black bus. In this case, encryption and integrity checking must be used.
- *Secure ICAP:* This peripheral enables reconfiguration of the CommB cryptographic accelerator. Secure ICAP allows write on FPGA according to a security policy describing which FPGA area can be reconfigured. Secure ICAP also frequently checks if FPGA bitstream is corrupted by the use of a hash algorithm in order to prevent fault injection attacks or natural *Single Event Upset*.

ContB black peripherals are:

- *Black Control I/O:* This unit is used to receive order from radio set black side.
- *Memory controller:* A memory controller is used to enforce secure storage of red data into non volatile memories (e.g. flash memories).

2) Channel Configuration: A channel is a specific connection through the CommB. There are two kinds of channel, system channels and user channels. Only **one** *system channel* is active at any time, this channel is used to

convey Type 1 and Type 2 messages. The system channel is initialized at CSS boot time according to radio set system channel bypass rules. Type 3 messages are sent through the user channels. One or more user channels may be initialized at the same time according to CSS capabilities. SSCA interfaces enable to set user channel properties like:

- *Mode (r/r, b/b, simplex, half-duplex, full-duplex)*
- *cryptographic algorithms*
- *Keys*
- *Certificate*
- *Bypass rules*

When creating a new channel, the ContB uploads bypass rules to the CommB memory and attributes an ID to the channel.Then system returns the channel ID. When the CommB receives a Type 3 message, the channel ID is extracted. This ID is used to select proper keys and certificates on the key register. If necessary, the CommB asks the ContB to reconfigure the DRCA to match needed algorithms.

VI. CSS Expected Characteristics

A. Communication Block Expected Characteristics

CommB is built on a 32 bits CPU and a DRCA. A synthesized CPU core like Xilinx Microblaze [11] uses less than 500 slices (a Virtex 4 slice is composed by two 4-inputs Look Up Table, two D-Flip-Flop and one carry chain). Three 16Kb block RAM are needed for CPU volatile data storage, instruction storage and bypass rule storage. It is not easy to estimate size of needed FIFOs. Each CORBA message must be processed entirely in one time (message may be rejected), so FIFOs must be able to contain at least one CORBA message. As shown in [12], packet as large as 32000 bytes may be used in SCA radio in order to sustain high data rates. However FPGA internal RAM is limited. In [13], a secured network architecture using the Sierra II chip is presented, such architecture handles data rates up to 4 Mbps and payload are limited to 576 bytes. So in a first approach, four FIFOs are allocated on FPGA block RAM for CommB red and black buses, so entire message size is limited to block RAM size on Virtex 4. In further developments, FIFO could be embedded in external RAM to extend maximum message size and as a consequence to increase data rates.

DRCA has to accelerate a large amount of different cryptographic algorithms. In order to keep the DRCA as small as possible, only the most time consuming parts of the cryptographic algorithms are implemented in hardware. Three kinds of cryptographic algorithms may be defined:

- *Symmetric encryption algorithms (e.g. AES)*
- *Asymmetric signature algorithms (e.g. RSA or ECC)*
- *Integrity check algorithms (e.g. SHA or MD5)*

All algorithm types are used for Type 3 messages, so DRCA needs to be sufficiently large to run these algorithms at the same time.

In the case of symmetric algorithms, only one round may be implemented in a sequential architecture. In [14] a compact implementation of AES is described. This AES IP core only uses 222 slices and three block RAM on a Virtex II and reaches a throughput of 139 Mbps at 50 MHz. IP slice occupation may be reduced again implementing key diversification algorithms as software. AES implementation without key diversification on Virtex 4 uses roughly 170 slices at 139 MHz.

Like symmetric algorithms, only one round of integrity check algorithms may be implemented. Heliontech made an SHA-256 core which computes one 256 bits SHA block in 66 cycles. This core uses 758 slices and one block RAM.

An hardware architecture of *Elliptic Curve Digital Signature Algorithm* (ECDSA) is proposed by [15]. This 256 bits NIST implementation takes 1715 slices and 32 DSP blocks on a Virtex 4. A full ECDSA signature is generated in 495µs at 490 MHz. Heliontech provides an 128 bits modular exponentiation IP core which takes 1025 slices and one BRAM. This core can compute 17.8 1024-bits operations per seconds at 148 MHz.

TABLE II
Communication Block FPGA Occupation on Virtex 4

IP core	Slice	RAMB16	Max Freq. (MHz)
CPU system	500	3	150
FIFO	X	4	X
AES-128	170	3	139
SHA-256	758	1	160
ECDSA 256	1715 + 32 dsp	X	490
RSA 128	1025	1	228
TOTAL	3140 + 32 dsp	12	150

Table II summarizes some information about CommB IP core expected FPGA occupation. CommB implementation on last FPGA devices like Virtex 6 might give us better results. Expected total data rates is impossible to evaluate due to unknown algorithm time execution on the CPU.

B. Control Block Expected Characteristics

ContB uses one 32 bits CPU which requires less than 500 slices. Red and black control I/O are protocol specific, so maximum frequency and FPGA occupation cannot be inferred before a protocol has been chosen. Key register size will only depend on maximum supported key number. The *True Random Number Generator* uses few slices, an implementation of such IP core is described in [16].

The cryptographic accelerator is expected to use pretty much the same resources as the CommB cryptographic accelerator. In further works cryptographic accelerator may be shared by CommB and ContB, however ContB and CommB isolation enforcement must be preserved.

An UART controller is used as Fill I/O port because DS-101 and DS-102 key fill protocols are kept confidential. As it is not possible to evaluate easily software size, ROM and RAM size cannot be inferred. A secure ICAP should include features like liability enhancement and write access management. [17] describes a fault tolerant ICAP which could be used against natural fault occurrence and fault attacks on FPGA bitstream. This ICAP uses 1308 slices and 6 block RAM on Virtex 4. [18] describes an ICAP that prevents unauthorized write access on sensitive FPGA region (e.g. ICAP itself) and a secure reconfiguration process using bitstream encryption is presented in [19].

VII. Conclusion and Further Work

This paper describes a new *Crypto Sub System* reconfigurable architecture for SCA compliant SDR radio set. This architecture tends to solve issues like CSS upgradeability and interoperability by the use of reconfigurable FPGA chip. There is two units, one *Configuration Block* and one *Communication Block*. The CommB can be reconfigured from the ContB according to user needs. This solution is far more flexible than already existing CSS solution, however use of FPGA raises some issues. CSS circuit must enforce its own integrity against unauthorized FPGA bitstream modification from natural radiation or fault attacks. Moreover there is no means, at this moment, to ensure integrity and authenticity of the first loaded bitstream. Today, manufacturers only provide bitsream encryption on FPGA.

A complete development of a SSCA compliant radio set is a challenge. Our architecture is the first non confidential CSS architecture and we hope that it could help secure SDR designing.

Acknowledgement

This work was in part supported by the French DGA (part of the French Department of Defence) and by the University of Bordeaux. The views expressed in this paper are those of the authors and cannot be regarded as stating an official position of the DGA or the French DoD.

References

[1] *Software Communications Architecture Specification*, JTRS Std. 2.2.2, Rev. FINAL, Mai 2006. [Online]. Available: http://sca.jpeojtrs.mil/

[2] *Security Supplement to the Software Communications Architecture Specification*, April 30, 2004, JTRS Std. 2.2.1, April 2004. [Online]. Available: http://sca.jpeojtrs.mil/

[3] *Common Objetc Request Broker Specification - Part 2: CORBA Interoperability*, Object Management Group Std. 3.1, January 2008.

[4] D. K. Murotake, "An open architecture sca reference plateform," in *Proceeding of the SDR 07 Technical Conference and Product Exposition*, 2007.

[5] M. Kurdziel, J. Beane, and J. J. Fitton, "An sca security supplement compliant radio architecture," in *Proc. IEEE Military Communications Conference MILCOM 2005*, 17–20 Oct. 2005, pp. 2244–2250.

[6] *Sierra II Datasheet*, Harris Corp., 2005. [Online]. Available: http://www.rfcomm.harris.com/

[7] D. Murotake and A. Martin, "A high assurance wireless computing system (hawcs) for software defined radio," in *Proceeding of the SDR 06 Technical Conference and Product Expositi*, 2006.

[8] *Advanced INFOSEC Machine Datasheet*, General Dynamics, 2006. [Online]. Available: http://www.gdc4s.com/

[9] Cornfield multi-chip module. Raytheon. [Online]. Available: http://www.fas.org/irp/program/security/_work/cornfld.html

[10] *Virtex-4 FPGA Configuration User Guide*, Xilinx, April 2008.

[11] *MicroBlaze Processor Reference Guide*, Xilinx, 2008.

[12] G. Lind and C. Littke, "Software communication architecture (sca) for above 2 ghz satcom," in *Proceeding of the SDR 04 Technical Conference and Product Exposition*, 2004.

[13] B. C. Boorman, C. D. Mackey, and M. T. Kurdziel, "A scalable hardware architecture to support applications of the haipe 3.1 standard," in *Proc. IEEE Military Communications Conference MILCOM 2007*, 29–31 Oct. 2007, pp. 1–8.

[14] P. Chodowiec and K. Gaj, "Very compact fpga implementation of aes algorithm," in *CHES 2003*. Berlin Heidelberg: Springer-Verlag, 2003, pp. 319–333.

[15] T. Gneysu and C. Paar, "Ultra high performance ecc over nist primes on commercial fpgas," in *Cryptographic Hardware and Embedded Systems CHES 2008*, S. B. . Heidelberg, Ed., vol. 5154/2008, 2008, pp. 62–78.

[16] V. Fischer and M. Drutarovsky, "True random number generator embedded in reconfigurable hardware," in *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002), Springer-Verlag, LNCS 2523*. Springer-Verlag, 2002, pp. 415–430.

[17] J. Heiner, N. Collins, and M. Wirthlin, "Fault tolerant icap controller for high-reliable internal scrubbing," in *Proc. IEEE Aerospace Conference*, 1–8 March 2008, pp. 1–10.

[18] R. J. F. Chaves, "Secure computing on reconfigurable systems," Ph.D. dissertation, Technical University of Lisbon, 2007.

[19] L. Bossuet, G. Gogniat, and W. Burleson, "Dynamically configurable security for sram fpga bitstreams," *International Journal of Embedded Systems*, vol. 2, pp. 73–85, 2006.