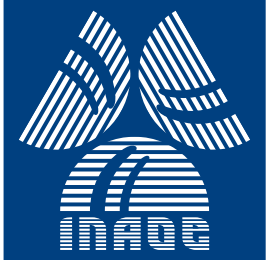

# A Software-Radio Platform with Reconfigurable Architecture on MAC Layer for Security Systems


por

**Ignacio Algredo Badillo**


Tesis sometida como requisito parcial para obtener el grado de

**DOCTORADO EN CIENCIAS EN LA ESPECIALIDAD DE CIENCIAS COMPUTACIONALES**

en el

**Instituto Nacional de Astrofísica, Óptica y Electrónica**

Diciembre 2008

Tonantzintla, Puebla

Supervisada por:

**Dra. Claudia Feregrino Uribe, INAOE**

**Dr. René Armando Cumplido Parra, INAOE**



# A Software-Radio Platform with Reconfigurable Architecture on MAC Layer for Security Systems

*Ignacio ALGREDO-BADILLO*

December, 2008

# Abstract

The development, analysis and evaluation of architectures of high performance, as well as new methodologies of design hardware are useful tools in the area of security systems based on cryptography, where recently it is required flexibility to change different functionalities. It is important to highlight two key points, on the one hand, the cryptographic algorithms utilize complex and iterative processes with many operations, and their application in communications networks causes a decreased speed of the data transmission. On the other hand, there is a great amount of communications networks, which establish standards or security architectures based on communication protocols. These last ones have functionalities that are independent of the algorithm, originating several possible combinations between different types networks, protocols and algorithms. So, the flexibility is an important characteristic because an ideal device of digital communications must be connected and establish interchange of data in any type of network. This idea is considered by the software-radio concepts, where a basic radio system with modifiable operational elements can provide different functionalities, which are controlled by external elements. These last elements have configurations that allow to make intelligent decisions, providing diverse software radios such as radios programmed by the user, radios controlled by software, and cognitive radios. In this point, the reconfigurable hardware architectures are an important element, because they can change their functionality by modifying their configuration, providing high performance and high flexibility. So, both the reconfigurable cryptographic architectures and the software radios are motivation for topics of research works focused to obtain optimal systems. The goal of this work is to design and develop a reconfigurable architecture for the secure communication of digital information, focused towards the development of a software-radio platform. For purposes of validation, testing and comparison, the proposed architectures are implemented on FPGA devices, where the hardware implementations report high performance and efficiency when implementation results are compared with similar works.

# Resumen

El desarrollo, análisis y evaluación de arquitecturas de alto desempeño, así como nuevas metodologías de diseño hardware son herramientas útiles en el área de sistemas de seguridad basados en criptografía, donde actualmente se requiere flexibilidad del sistema para ofrecer diferentes funcionalidades. Hay dos puntos críticos importantes, por un lado, los algoritmos criptográficos realizan procesos complejos e iterativos con demasiadas operaciones, y su uso en redes de comunicación provocan que la velocidad de la transmisión de datos se decremente. Por otro lado, hay una gran cantidad de redes de comunicaciones, las cuales establecen estándares o arquitecturas de seguridad basados en protocolos de comunicación que son independientes del algoritmo, originando una gran cantidad de combinaciones posibles. La flexibilidad es un característica importante porque un dispositivo ideal de comunicaciones digitales debe conectarse y establecer intercambio de datos en cualquier tipo de red. Esta idea es considerada por los conceptos radio software, donde se establece un sistema radio básico, al que se modifican sus elementos operacionales para obtener diferentes funcionalidades. Estas funcionalidades son controladas por un elementos externo con decisiones inteligentes, encontrando, por ejemplo, radios programados por el usuario, radios controlados por software, y *cognitive radios*. En este punto, las arquitecturas hardware reconfigurables son un elemento clave, porque permiten la capacidad de cambiar su configuración de funcionamiento, combinando un alto desempeño con una alta flexibilidad. Tanto los radio software como las arquitecturas reconfigurables y criptográficas son motivo de investigación para obtener sistemas óptimos. La meta de este trabajo es diseñar y desarrollar una arquitectura reconfigurable para el procesamiento seguro de información, enfocada hacia la aplicación de una plataforma radio software. Para propósitos de validación, comprobación y comparación, las arquitecturas son implementadas en dispositivos FPGA, midiendo la eficiencia de la implementación hardware y comparando con trabajo similar, donde las arquitecturas propuestas reportan hasta ahora el más alto desempeño y la mejor eficiencia.

# Acknowledgements

I am grateful to all those people who without their participation this research would have been impossible. My special thanks to my mother Gisela Badillo Díaz, who has supported my brothers Uriel and Edilberto Jr, my father Edilberto and me, allowing us to reach our personal goals devoting herself completely to us. Finally, I am also grateful to my grandfather Ignacio, who supports my parents in different tasks.

# Dedication

Particular thanks are owed to Dr. Claudia Feregrino Uribe and Dr. René Cumplido Parra for providing guidance and resources during the course of this project. I am also grateful for Dr. Paraskevas Kitsos, Dr. Andrés David García García, Dr. Saúl Pomares Hernández, Dr. Leopoldo Altamirano Robles and Dr. Miguel Octavio Arias Estrada for being part of the examining committee.

This work is dedicated to my children Yexalén and Máximo Algredo Corona and my wife Adriana Corona Hernández, a total and important motivation for reaching my goals.

# Abbreviations

2G : Second Generation mobile telecommunication

3-DES: Triple DES algorithm

3G : Third Generation mobile telecommunication

4G : Fourth Generation mobile telecommunication

AAD : Additional Authentication Data

AES: Advanced Encryption Standard algorithm

AES-CCM : AES in CCM mode

ASIC : Application-Specific Integrated Circuit

AwR : Aware Radio

AdR : Adaptable Radio

BRAM : Block Random Access Memory

CaR : Software Capable Radio

CAST : Carlisle Adams and Stafford Tavares algorithm

CBC : Cipher Block Chaining mode

CBC-MAC : Cipher Block Chaining - Message Authentication Code mode

CCM : CTR with CBC-MAC mode

CDMA : Code Division Multiple Access

CLB : Configurable Logic Block

CFB : Cipher FeedBack mode

CR : Cognitive Radio

CRC : Cyclic Redundancy Check

CTR : Counter mode

DES : Data Encryption Standard algorithm

DH : Diffie-Hellman

DSA : Digital Signature Algorithm

DSP : Digital Signal Processor

DSS : Digital Signature Standard

ECB : Electronic CodeBook mode

ECC : Elliptic Curve Cryptography

FIR : Finite Impulse Response filter

FF : Flip-Flop

FFT : Fast Fourier Transform

FPGA : Field Programmable Gate Array

Gbps : Gigabits per second

GMACH : Generic MAC Header

GPP : General-Purpose Processor

GPRS : General Packet Radio Service

IEEE : Institute of Electrical and Electronics Engineers

IDEA : International Data Encryption Algorithm

IKE : Internet Key Exchange

IP : Internet Protocol

IPSec : Secure IP

IR : Infrared

IV : Initialization Vector

LAN : Local Area Network

LUT : LookUp Table

MAC : Medium Access Control sub-layer

MAN : Metropolitan Area Network

Mbps : Megabits per second

MD4/5 : Message Digest 4/5

MIC : Message Authentication Code

OCB : Offset Code Book

OSI : Open System Interconnection model

OFB : Output FeedBack mode

PAN : Personal Area Network

PHY : Physical layer

PKI : Public Key Infrastructure

RC2/4/5 : Rivest's Code 2/4/5 algorithms

RF : Radio Frequency

RIPEMD-160 : RACE Integrity Primitives Evaluation Message Digest of 160 bits

RSA : Rivest-Shamir-Adleman algorithm

SHA-1/224/256/384/512 : Secure Hashing Algorithm 1/224/256/384/512

SDR : Software Defined Radio

SIC : Segmented Integer Counter (other name for CTR mode)

SIMD : Single Instruction Multiple Data

SPR : Software Programmable Radio

SR : Software Radio

TTA : Transport Triggered Architecture

UMTS : Universal Mobile Telecommunications System

VLSI : Very Large Scale Integration

WAN : Wide Area Network

WBAN : Wireless Body Area Network

WCDMA : Wideband Cell-Division Multiple Access

Wi-Fi : Wireless Fidelity

WLAN : Wireless LAN

WMAN : Wireless MAN

WPAN : Wireless PAN

WRAN : Wireless Regional Area Network

WWAN : Wireless WAN

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In the modern digital communications, many applications such as cellular telephony, Bluetooth service, Internet, television, multimedia, e-commerce, and bank transfer are present. All of them have a wide market, but they use specific devices, requiring a device for each application. For example, when a cellular phone operates in a cellular network and transfers data by Bluetooth or IR (infrared), related systems contained by the cellular phone should be activated. This means that multiple devices should be linked to have different behaviors operating on several communication networks where there is a number of networks operating in the communications environment, and the wireless systems, which allow mobility, are the most demanded. Hardly, new features, such as using a printer or a scanner, can be added, requiring suitable systems.

For each network, specific devices execute one or more defined tasks. What is the ideal solution? The answer is a device that can operate in the different networks. This idea is conceptualized by the software radio, that can be programmed to operate in different networks. In general, they are based on a basic radio with characteristics of multi-functionality, with an intelligent element controlling their program. For example, there are radios programmed by the user, by software or self-programmable radios. This evolution has occurred according to the technological advances.

Here, flexibility is required, but we found two main security problems. On the one hand, software radios can enter into any network, and they can be attackers. On the other hand, given that the transmission channel is the air, they can be attacked at the same time. There are several ways to protect the transmissions of data, and one of the most important is the cryptography. By

using cryptographic algorithms with iterative and complex processes, bottlenecks can appear, so, implementations with high performance are required. It is reported that hardware architectures have better performance than the software ones. It is vital to offer security through cryptographic security based on standards, where implementations report high performance and flexibility.

Reconfigurable hardware architectures are an option because they have inherent characteristics of high performance and flexibility. Therefore pertinent analysis should be performed to review the feasibility of a reconfiguration scheme, to check the type of reconfiguration that can be used, and to review the type of tasks partition that can be obtained by considering different networks and security architectures.

## 1.1 Motivation

The multi-functionality is required to an ideal radio device, and software radio is a concept that establishes mobile devices providing flexibility. Software-radio systems can change their programs or functions to operate on different networks. In the OSI reference model, SR research is focused on lower layers, closing to the physical layer [Bucknell, 2000]. These systems can enter/leave the communications networks, and security problems can come out, because an SR system can be considered like an attacker/attacked, which transmits data in its channel that is the air. There are several mechanisms to secure a given network, and cryptography is widely used.

Cryptography uses algorithms to provide security, executing complex computations in different modes of operations. Constantly, new computational technologies emerge, and the cryptographic research should be updated, focusing on the design, evaluation, and implementation of cryptographic algorithms and protocols, on the development of security architectures for information and communication systems and on the development of security mechanisms for embedded systems. And not only cryptographic algorithms have been proposed, but modes of operation that increase the security, such as CCM mode [Dworkin, 2004]. These algorithms execute iterative process, using multiple operations and special control, which reduces the speed in the transmissions of data. Furthermore, cryptographic hardware architectures have reported better performance than the software ones. In this way, this work focused to provide high efficiency for security hard-

ware architectures based on communication protocols just standardized. By using hardware architectures, the lack of flexibility becomes a big problem, because several communication protocols support a number of cryptographic algorithms.

Reconfiguration is a feature that enables hardware architecture to change its configurations in certain times. It is necessary to highlight that reconfigurable hardware design methodologies are necessary, because reconfiguration is a new way of designing and modeling hardware architectures.

A reconfigurable architecture combines efficiency with flexibility to offer cryptographic security to software radios based on standardized protocols.

In this work, a software-radio platform to support security protocols for wireless communication networks is proposed as well as hardware design methodologies, featuring high flexibility. The last ones include reconfigurable and non-reconfigurable architectures focused on high hardware efficiency.

## 1.2 Description of the Problem

Nowadays, in the communication and computer sciences, the software radios and reconfigurable computing have become important research areas. New design methodologies and hardware architectures are necessary, which provide systems that satisfy modern and future requirements, such as high data transmission rate, security and functionality in several operational environments. Although certain devices allow the designers to modify dynamically their configurations even in portions of the chip, the problem is a lack of satisfying design methodologies, which enable to optimally implement a high-level specification into a software-radio platform with dynamic reconfiguration.

SR research is focused on lower layers, but an SR system can be an attacker/attacked, here security is necessary, and cryptographic algorithms execute complex operations, requiring architectures with high efficiency and flexibility. Hardware architectures report certain performance, but if they are used for a reconfigurable architecture they report a lower performance, but their performances are better than the other ones reported by the GPPs. Reconfiguration enables hardware architectures to report flexibility, and in this work, the aim is to reach high throughput/area ratio, an efficiency metric that is close to the non-reconfigurable hardware architectures.

## 1.3 Research Questions

This research aims to answer the following research questions:

Is it possible to provide an SR reconfigurable architecture to satisfy security requirements of several wireless environments?

Which is the best way to partition protocol-specific tasks to efficiently use the proposed reconfigurable architecture?

Which reconfiguration scheme is more suitable for wireless communication security applications?

Which reconfiguration scheme for software radio functions has the best flexibility and performance?

What modules must be reconfigurable to improve the throughput?

## 1.4 General Objective

The aim of this project is to apply the software radio concept to the wireless communication networks, specifically to the cryptographic processing on the MAC (Medium Access Control) sub-layer, providing a design methodology to the development of a software-radio platform, which present high hardware efficiency. This platform will be designed and developed to evaluate the proposed hardware design suitable for modern applications that require a high speed of cipherdata transmission. This includes the development of new design methodologies for reconfigurable and configurable hardware architectures.

## 1.5 Specific Objectives

Next, specific objectives are described:

To revise and select security protocols of wireless communication networks to test reconfiguration schemes.

To design and propose hardware architectures for MAC cryptographic processing of the selected wireless communication networks.

To analyze selected security architectures and to identify common and particular tasks.

To propose design methodologies to develop hardware architectures and design optimized modules to provide high throughput cryptographic functions focused on MAC sub-layer processing.

To develop a flexible high-performance SR processing platform and a configuration library for the platform.

To propose a reconfiguration scheme to support security functions for the selected protocols. Also, to design a reconfiguration model to control the proposed SR platform.

## 1.6 Methodology

The methodology focuses on the reconfigurable hardware design for a software radio architecture, using optimization hardware techniques to improve the performance of these security architectures, which enables to transmit data at high speed in the modern applications of the communication networks. This methodology is based on revision and analysis, identification of the main tasks, design and development of high-efficiency hardware architectures, proposing hardware design methodologies, and evaluating and proposing the SR platform to support reconfigurable security architectures and to report high efficiency (see Fig 1.1).

From now on, it is assumed that hardware architecture is a model, representing a block diagram of an algorithm or an process, which is hierarchically conformed by modules, sub-modules, and components. Implementations are descriptions in high-level language of the architectures, which are supported by FPGA devices. Also, in the standards, such as in the IEEE 802.11i-2004 and IEEE 802.16e-2005, the security architectures are defined, which are named security schemes in this document.

## 1.7 Evaluation

The test of the hypothesis is proved by evaluating the proposed SR platform and its cryptographic hardware architectures. The performance is measured in terms of: 1) flexibility, 2) hardware efficiency and 3) throughput.

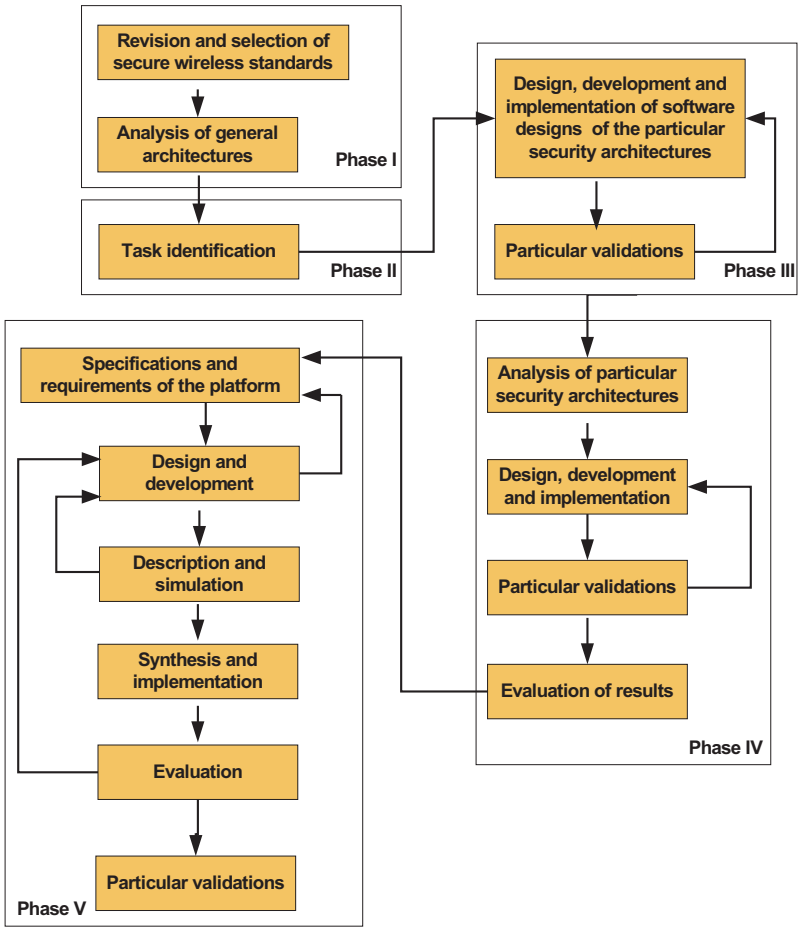


Figure 1.1: Methodology

The flexibility is evaluated by changing cryptographic schemes of the communication standards on the SR platform and by changing modules or blocks on these hardware architectures of the same protocols.

The hardware implementation efficiency (bps/slices) is a measurement of this type of cryptographic hardware implementations and it is defined as the ratio between the reached throughput and the number of slices that each implementation consumes [Kitsos, 2006].

Design methodologies are necessary for trade-off studies between the used hardware resources and throughput and to reach high throughput and efficiency. The focus is on reducing critical path, using few hardware resources,

and designing specialized modules without wasting advantages such as parallelization or unrolling.

## 1.8 Structure of this Document

Fundamentals on communication networks, SR concepts, security, cryptography and reconfigurable hardware are revised in Chapter 2. Related works are described in Chapter 3. The design and development of the SR platform is divided in four chapters. According to the methodology (see Section 1.6), in Chapter 4, initial steps such as requirements and particular methodology are proposed, needing the design and development of the hardware architectures for the AES and AES-CCM algorithms (Section 4.4.1), and for the IEEE 802.11i and IEEE 802.11e-2005 security schemes (Chapter 5). Implementation results are depicted in Chapter 6.The last two security architectures are the key element for designing SR platform, and the data input to select them are used to develop the SR module. In Chapter 7, final steps are described to propose the SR platform. Finally, in Chapter 8, conclusions, contributions and future work are depicted.

# Chapter 2

# Fundamentals

Recently, the global telecommunications industry and market have had many new technological developments, where digitalization of communications, wireless cellular telephony, and the Internet are the most significant, see Fig 2.1 [Pashtan, 2006]. These developments have enabled the addition of voice, data and video services using different protocols, which are based in communication networks, offering to consumers a number of applications and services, irrespective of location and operating environment.

This continuous development of digital communications and the Internet have played a role in redefining the way people communicate, handling multiple communication networks with diverse characteristics and advantages. Any given network is used for a specific range and data rate, using a corresponding communication standard. To access network services, a user must select among the available standards as shown in Fig 2.2, and this ensures reliable interchange of data between communication systems on different platforms.

In this scenario of a great variety of standards there are several problems and one of them is that makes difficult for a single architecture to incorporate the functions needed to manage a number of protocols and networks, however, it is desired that this device operates in different networks, for example, transmitting data to a Bluetooth notebook, handling a printer in a local network, watching videoconferences from a WiMAX server, sending messages to cellular phones, or receiving television channels from (Wireless Regional Area Network) WRAN base stations.

Another problem is that new markets for services and devices will be created, as well as new versions of established standards (for example IEEE

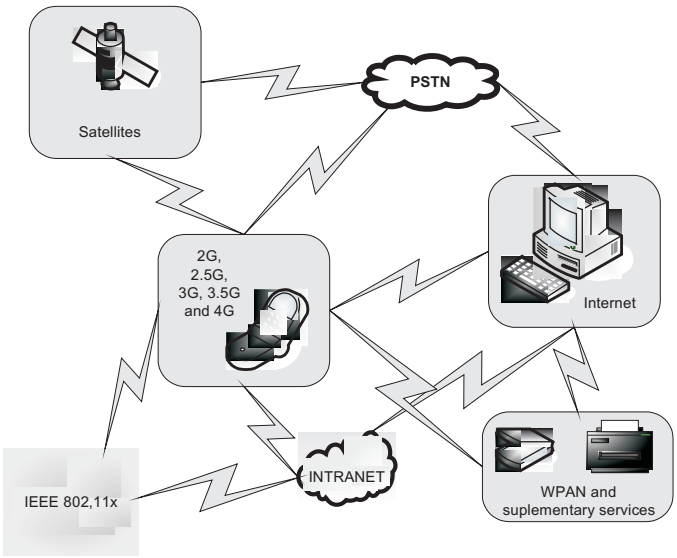


Figure 2.1: Example of a communication environment

802.11 standards and cellular standards, Fig 2.3 and Fig 2.4 [National-Instruments-Corporation, 2006], [Walke, 2001].

This emergence of new standards, and future applications will demand higher data rates, more security and/or less power consumption by using applications such as simultaneous videoconferences, TV or DVD transmissions, remote videogames for multiplayer, great amount of digital information, ultra wideband and WLAN for 4G (fourth generation) communications. These applications running on current communication networks can not be possible, but they will be benefited by transference of data at 1 Gbps, see Table 2.1. This technology about 1-Gbps wireless link is on development for future applications [ICT-Centre, 2008].

Finally the lack of security in wireless communications, because the air is the transmission channel, which is insecure, the transmissions of data can be attacked by a third party. New communication technologies also motivate the development of more demanding applications that require exchanging of secure information. In these applications, network devices allow exchanging digital information using wireless channels which make them more vulnerable to security problems like interruption, modification, interception and fabrication, see Fig 2.5. Furthermore, a software-radio system is able to

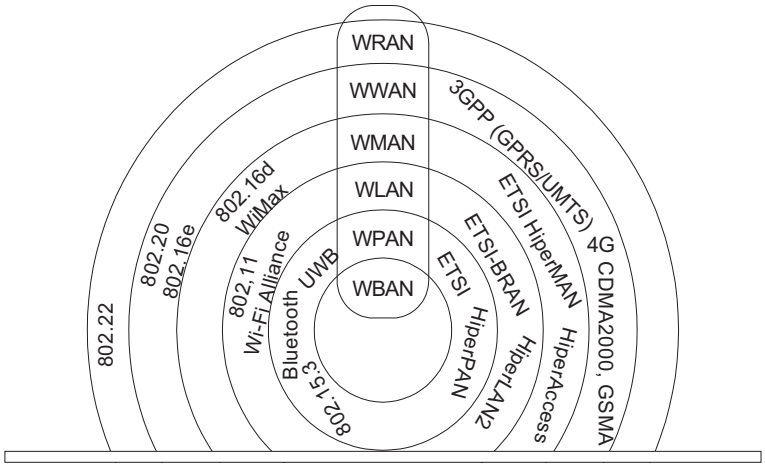


Figure 2.2: Multiple standards in different type of communication networks

Table 2.1: Required time in data transference for modern applications in current networks and a 1-Gbps wireless link [ICT-Centre, 2008]

| Item | Size | Dial-up | Broadband | WLAN | Gigabit |
|---|---|---|---|---|---|
| Digital files | 5 Mb | 12 min | 27 sec | 3.6 sec | 0.04 |
| CD-ROM | 650 Mb | 1.1 days | 58 min | 7.8 min | 5.2 sec |
| Quality TV-120 min | 1 Gb | 1.7 days | 1.5 hrs | 12 min | 8 sec |
| DVD-120 min | 4.2 Gb | 6.9 days | 6.2 hrs | 50 min | 34 sec |

enter to several networks, because it configures its system at a given time to connect to a certain network, and in this way, it can be considered as an potential attacker, or if has not protection, it can be attacked. Security services are necessary to protect the communications, where current wireless communication protocols are based on cryptography.

In the communication networks, hardware architectures with high performance and flexibility are required to provide security services without causing bottlenecks. Next, more details are described.

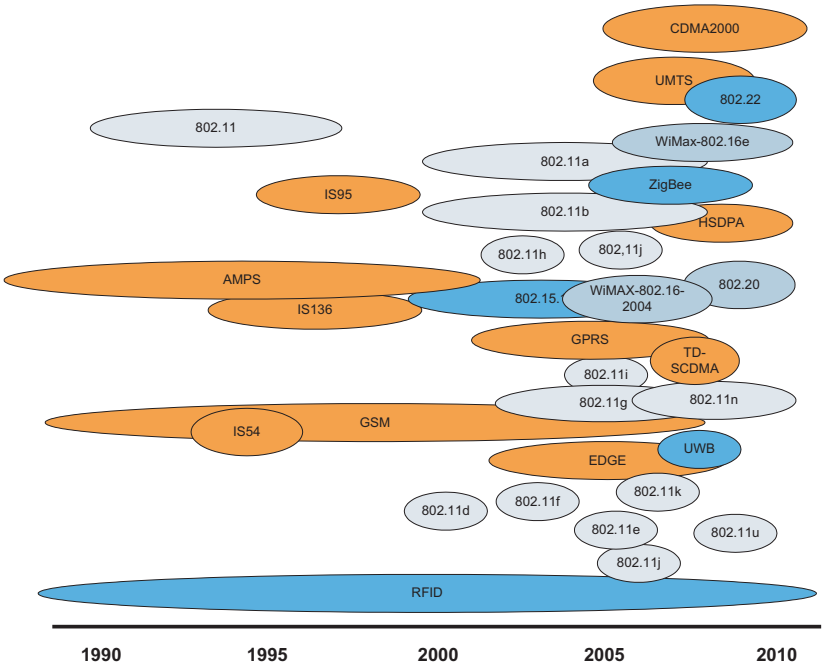


Figure 2.3: New communication standards are being created [National-Instruments-Corporation, 2006]

## 2.1 Communication Networks

In communication networks, two or more computing devices are linked, having the purpose of sharing digital information, and combining software and hardware architectures. The way in that these devices establish communication is through a language called protocol. A reference model is developed for the networks based on seven-layer architecture, and it is in each layer where a set of standard protocols specifies the behavior of the layer [Hekmat, 2005].

Networks can be categorized in several different ways (for example geographic area, used protocols, wireless or not), and an approach is based on the geographic area, according to their range of operation [Agrawal, 2008].

**Wireless Body Area Network (WBAN).** This network is conformed of sensors to monitor a user within a short range (up to 1 meter), providing the communication to other systems of wearable computers whose components are distributed on the body.

**Wireless Personal Area Network (WPAN).** It is used for inter-

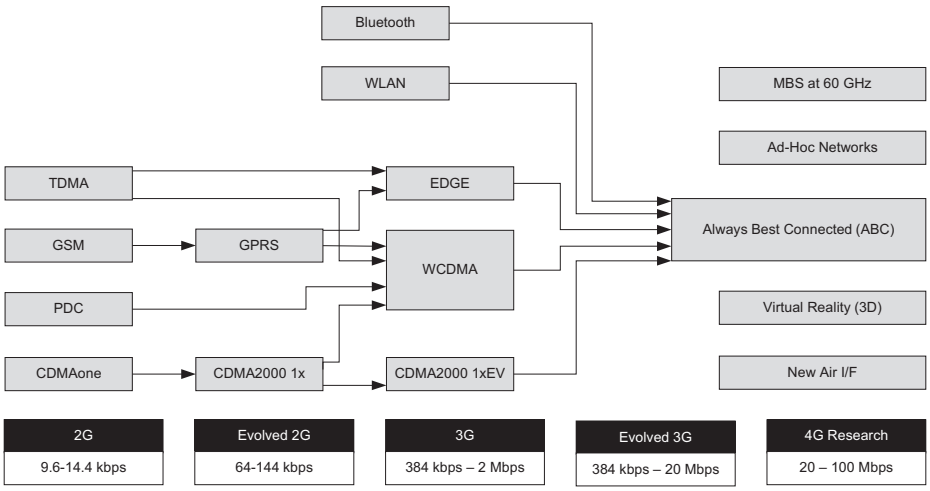


Figure 2.4: Evolution of cellular networks [Walke, 2001]

connecting devices centered on an individual person's workspace or house. Typically, it permits communication within 10 meters.

**Wireless Local Area Network (WLAN).** Extending out from the personal area networks, WLAN is used in an outdoor workspace, with range up to 20 kilometers.

**Wireless Metropolitan Area Network (WMAN).** It is utilized for supplying broadband wireless access, linking base stations in metropolitan areas up to a range of 30 kilometers.

**Wireless Wide Area Network (WWAN).** This network provides access across cities covering a range up to 50 kilometers.

**Wireless Regional Area Network (WRAN).** The main application for these networks is the wireless broadband access for dispersed regions, with a transmission range up to 100 kilometers.

This classification is based on the wired networks, although LAN and WAN are the original categories, and the remaining type of networks have gradually emerged, and wired networks such as Storage Area Network, System Area Network, Server Area Network, Small Area Network, Desk Area Network, Campus Area Network, Controller Area Network, or Cluster Area Network are new elements of this geographic category. For example, System Area Network connects high-performance computers, links based on clusters with high-speed data transference, whereas Campus Area Network is a link for network spanning multiple LANs (smaller than a MAN), which is used

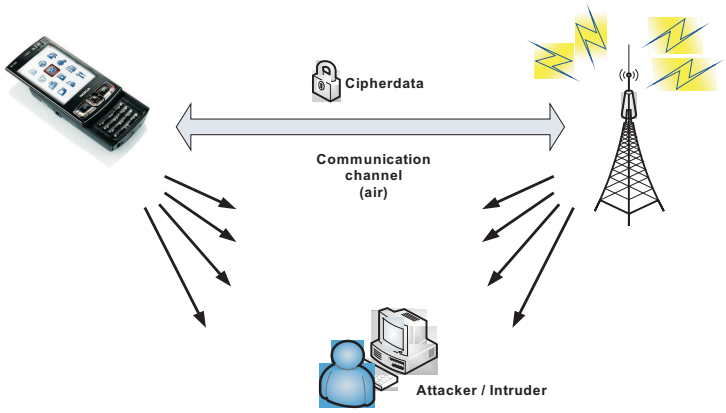


Figure 2.5: Insecurity in the wireless communications

in universities or local business campus.

This great amount of types of networks requires multiple communication protocols, and each one has a given application, resulting that a system running in different networks enables an ideal element, saving for example multiple devices, costs and power consumption. This characteristic of flexibility is provided by using software-radio concepts.

## 2.2 Software-Radio Concepts

As these new technologies evolve, more efforts are made to integrate new architectures and services using the software radio concept, which tries to integrate multiple communication functions and operations into a common platform. This convergence is made possible by recent advances in VLSI (Very Large Scale Integration), design tools and software design. These advances have also made feasible hardware reconfiguration, which in turn, allows designing processing platforms that offer high performance and flexibility.

The software radio concepts merge radio technology with capability of flexible communication systems, which can handle signals from various systems using the same hardware device. These concepts require hardware architectures with possibility of changing their programmable structure, and reconfigurable computing involves manipulation of the device resources, ex-

Figure 2.6: OSI model

ecuting different functional operations at established times.

Software radio will become enabler for developing multi-standard, multi-band devices with reduced effort and costs. As technology advances, the software radio concept has added new functionality, going from a non-programmable radio, passing by a radio controlled by software, to an aware, adaptive or cognitive radio, see Fig 2.7. This evolution has been benefited by the increasing of the technology as hardware elements as software ones [Polson, 2004].

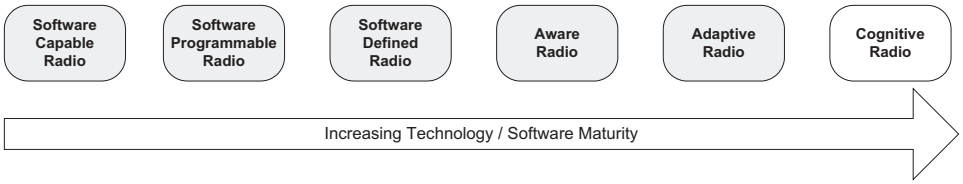


Figure 2.7: Evolution of the software-radio concepts

**Software Capable Radio (CaR).** It varies its communication conditions and requirements with minimum user parameters on dedicated hardware.

**Software Programmable Radio (SPR).** Radio architectures implemented in dedicated hardware, which is programmed by the user.

**Software Defined Radio (SDR).** SDRs can be reprogrammed and/or reconfigured on the fly to handle multiple communication protocols. For example, a SDR implementation could handle IEEE 802.11a, IEEE 802.11b and CDMA (Code Division Multiple Access) protocols, and also acts as a cell phone, cordless phone, wireless Internet device, GPS (Global Positioning System), and garage-door opener.

Software radio architectures have traditionally taken a hardware-oriented approach, driven by the signal processing capabilities of the underlying processing platform. SDR systems have typically been designed around a combination of processing technologies such as DSPs (Digital Signal Processors), FPGAs (Field Programmable Gate Arrays) and other specialized reconfigurable processing devices. The other aspects of the system architecture, such as the RF (Radio Frequency), analog-to-digital conversion and the system I/O (Input/Output), are then designed around the capabilities and limitations of the processing systems.

**Aware Radio (AwR).** This radio senses the environment and modifies its operation to adapt to new conditions. It is flexible and capable of

responding or conforming to changing or new situations, and it is dynamic, because its agility enables real-time configuration of the system.

**Adaptive Radio (AdR).** It monitors its own performance, varies operating characteristics, such as frequency, power or data rate, monitors the path quality and optimizes its performance by automatically selecting frequencies or channels.

**Cognitive Radio (CR).** This radio adapts to its environment by analyzing the RF environment and appropriately adjusting the spectrum use. It is an adaptive, multi-dimensionally aware, autonomous radio system that learns from its experiences to reason, plan and decide future actions to meet user needs.

The recent research works about software-radio systems focused on the lower layers of the OSI model, see Fig 2.8, which consider hardware architectures, but providing security services is primordial.

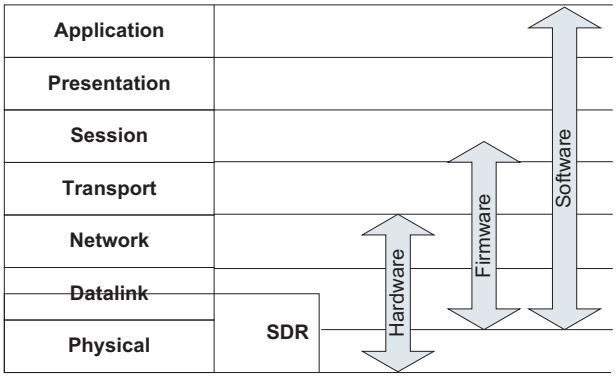


Figure 2.8: Research works about SR systems on the OSI model, considering implementation platforms

## 2.3 Security

Recently, there is a proliferation of the digital communications, but insecurity is a problem that affects the development of the communication systems, and situations that tend to be very popular as bank transfers are considered insecure. Insecurity should be prevented, checking weakness in the security of the system, avoiding threats to exploit vulnerabilities, using security services to protect the system, and applying countermeasures to provide a security

service [Ollikainen, 2004]. A bundle of security services can be provided to ensure the security of the system. These services are:

1. authentication: the receiver and sender identity should be verified,
2. secrecy or confidentiality: only the authenticated user is able to interpret the transmitted data,
3. integrity: the content of the communicated data is assured to be non-modifiable,
4. non-repudiation,
5. availability, and
6. countermeasures of routing data

There are many mechanisms to secure a system, such as cryptography, antivirus, firewalls, intrusion detectors, secure routing, and security policy management, where cryptography is one of the most important, and in the communication networks, security at all layers is mostly based on cryptographic protocols (see Fig 2.9).

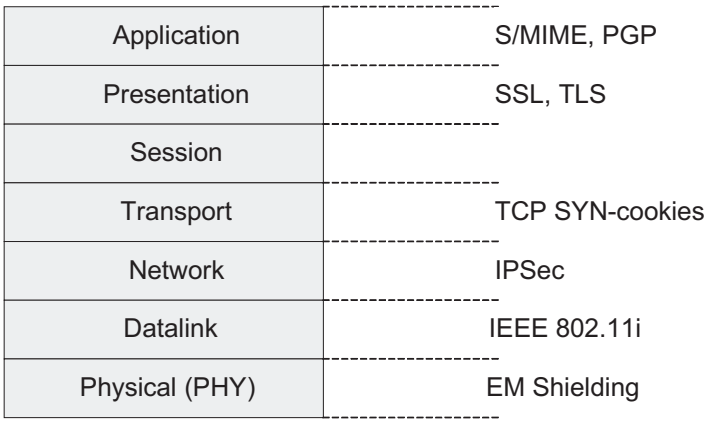


Figure 2.9: Examples of security protocols on different layers of the OSI model

## 2.4 Cryptography

Cryptography provides powerful mechanisms to protect data with high cost in terms of computing power. These mechanisms are mathematic algorithms which execute complex operations on iterative processes. Traditionally, cryptographic algorithms are used to provide a given security services, but nowadays, there are modes of operation that altering the service offering by an algorithm, into one or more services. The cryptographic algorithms can be classified in three types [Kessler, 1998], hash functions, symmetric, and asymmetric algorithms, that compute cryptographic operations (for example ciphering, messages code authentication or hashing) to provide security services (such as authentication, non-repudiation, integrity and privacy), and in the communication networks, security protocols use cryptographic algorithms, see Fig 2.10.

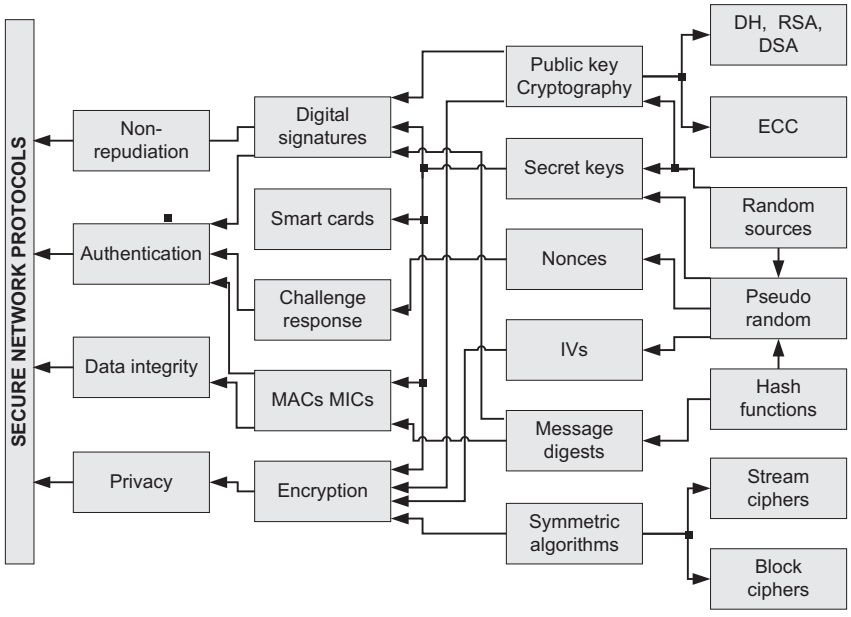


Figure 2.10: Security services of the secure protocols

The security standards used in the secure network protocols have been listed as optional methods different cryptographic algorithms and modes of operation. These will be selected based on security requirements, and parameters. For example, if security service of privacy is necessary, a symmetric

algorithm based on cipher block can be selected, and a cipher process will be used to provide privacy. Next, cryptographic algorithms and modes of operation are described.

**Hash Functions.** These functions do not cipher the complete message, but they are based on compression functions that generate blocks of length mm from blocks of length nn, see Fig 2.11. It is computationally infeasible that a plain text can be calculated from a hash value. Examples of hash functions are MD4/5, SHA-1/224/256/384/512, RIPEMD-160, and Whirlpool.

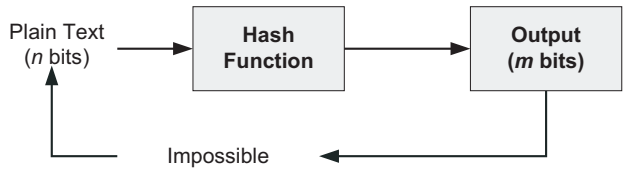


Figure 2.11: Hash functions

**Secret-key or symmetric algorithms.** They cipher the complete message, where sender and receiver share same key for ciphering or deciphering, see Fig 2.12. Distribution and storage of their keys presents major problems, as well as key management for multiple participants. Example of symmetric algorithms are DES, 3DES, AES, Blowfish, RC2/4/5, IDEA and CAST.

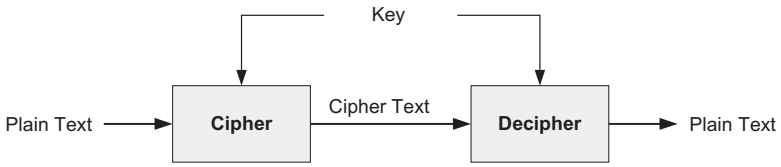


Figure 2.12: Symmetric algorithms

**Public-key or asymmetric algorithms.** They cipher the complete message, where sender and receiver have different keys for ciphering or deciphering, see Fig 2.13. Key pairs are mathematically dependent, and messages ciphered by one key can only be deciphered by other key. Example of asymmetric algorithms are RSA, DSA, ElGamal, DH, and ECC.

Modes of operation allow block ciphers to provide other services security or different levels of security. For example, CCM mode changes confidentiality service offering by a symmetric algorithm into authentication and confidentiality services. Other example is about level of security, CBC mode

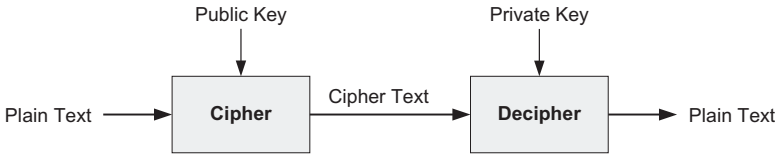


Figure 2.13: Asymmetric algorithms

can produce different cipherdata, considering a same plaintext, whereas ECB mode ciphers separately data blocks, producing the same cipherdata for a same plaintext, although this last one is chaining in a set of data blocks. Analyzing different attacks to the algorithm in these modes of operation, CBC mode provides more security than ECB mode.

Several modes of operation can be used, and these are so important that, currently, new modes have been defined. Examples of these modes of operation are ECB, CBC, CFB, OFB, OCB, CTR and CCM [Dworkin, 2001].

**ECB (Electronic Code Book).** Each block is independently ciphered in this mode, see Fig 2.14, showing patterns and repetitions.

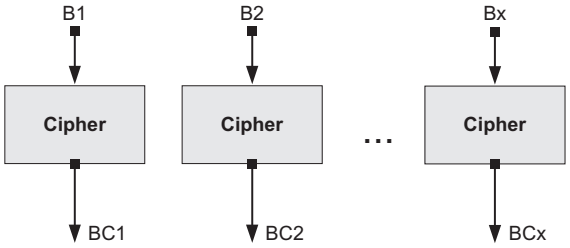


Figure 2.14: ECB

**CBC (Chaining Block Cipher).** In this mode, each block of plaintext is modified by computing an XOR operation before being ciphered, to mean that each ciphertext depends on all previous plaintext blocks. The other input for computing the XOR operation is the previous ciphertext or an initialization vector if the block of the plaintext is unique or it is the first. It can hide patterns and repetitions.

**CFB (Ciphertext Feed Back).** This mode provides a self-synchronous stream ciphering, where ciphertext depends o all previous plaintext blocks, see Fig 2.16. Using different IVs, patterns and repetitions are hidden. An IV is a value to provide randomization for the ciphering, and this is value is

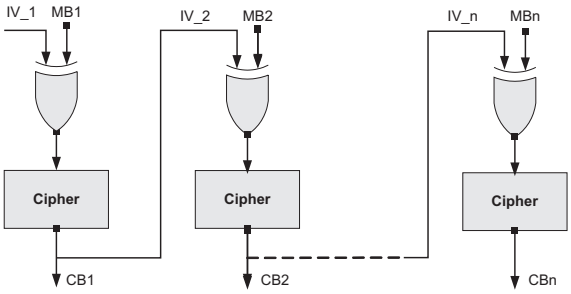


Figure 2.15: CBC

never reused with the same key, and it does not need be secret.

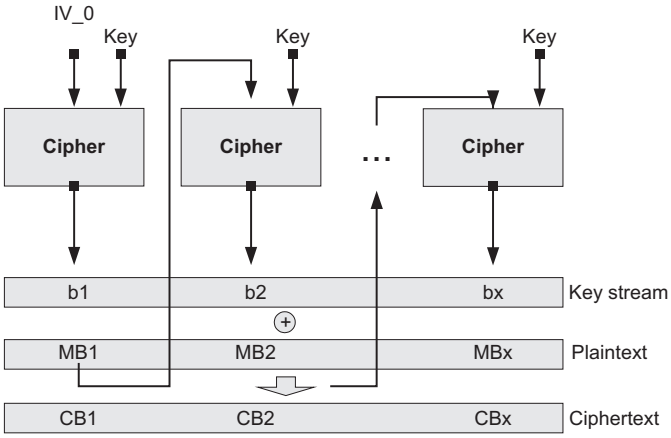


Figure 2.16: CFB

**OFB (Output Feed Back).** Different IVs are necessary to maintain a high level of security, and there is no linking between subsequent blocks, see Fig 2.17. This mode provides a synchronous stream ciphering, generating keystream blocks, which are computed by an XOR operation with the plaintext to get the ciphertext.

**OCB (Offset Code Book).** It is focused on parallelization to provide high speeds in data transmissions. It ciphers as CBC, but only the output of the last block is considered as a tag. Privacy and authentication are provided without using two modes or systems for providing these two security services, thus it requires lower computational cost.

**CTR (Counter).** This mode is also known as Segmented Integer Counter

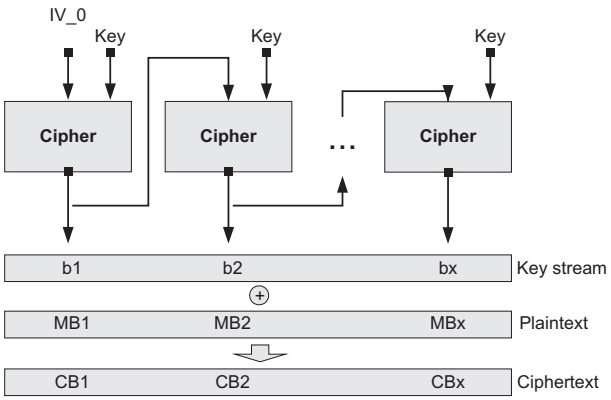


Figure 2.17: OFB

(SIC), and it provides a stream cipher, generating the next keystream block by ciphering successive values of the counter block. Nonce and IV can be the same value, and counter should guarantee a different value and do not repeat it for a long time.

**CCM (CTR with CBC-MAC).** This mode offers similar functionality as OCB mode, but requiring two modes of operation, ciphering each block of ciphertext and each block of associated authenticated data for providing privacy and authentication, respectively. Thus, it is computationally expensive, adds overhead to the ciphering and can use a same key for the two modes.

Both security services provided by the cryptographic algorithms and their modes of operation, security communication protocols handle these algorithms can increase the services security. For example, IEEE 802.11i standard, see Chapter 5, uses hash functions and methods for providing digital signatures and asymmetric algorithms for the key exchange, and further, this protocol adds characteristics for providing replay attack detection. IEEE 802.11i and IEEE 802.16e are modern standards, which are based on security architectures that use cryptographic algorithms on the MAC sub-layer, see Fig 2.21, to mean that cryptographic processing is executing in the lower layers, focusing to the hardware architectures.

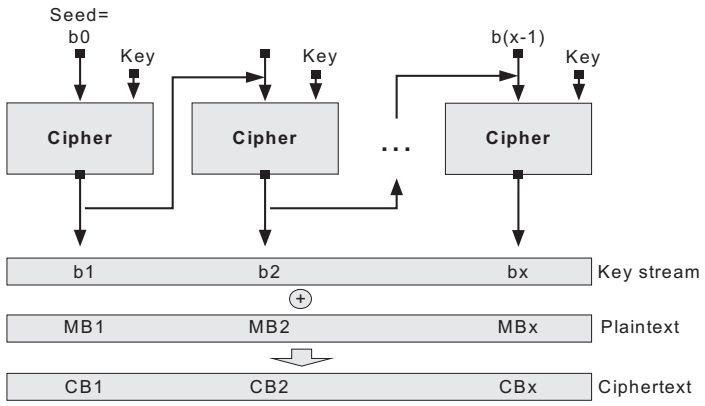


Figure 2.18: OCB

## 2.5 Security in Wireless Communications

Different cryptographic systems in diverse applications, like WWW servers, multimedia, the Transport Layer Security (TLS) protocol and secure mail protocols such as S/MIME, have provided a safe way for storing and transmitting information. These systems offer security based on complex architectures by adding cryptographic algorithms that may be hash functions, symmetric key algorithms and asymmetric key algorithms [Kocher et al., 2004], each one can be used for multiple and different services. Security is a primordial element in data transmissions for both wired and wireless communications, because communication networks offer a number of appliances that enable people to communicate and to use several applications such as: bank transfers, videoconferences and multimedia applications, among others.

Due to the cryptanalysis, the cryptographic algorithms are continuously attacked, and to the emergence of security problems, modern cryptographic algorithms are not sufficient to secure communications networks. In this way, security based in cryptography is a key element in the recent standards of the communication networks, requiring modern enhancements, such as new cryptographic algorithms, new modes of operation of these algorithms and, considering these networks, their new security schemes that should solve other issues. CCM (Counter with Cipher block chaining-message authen-

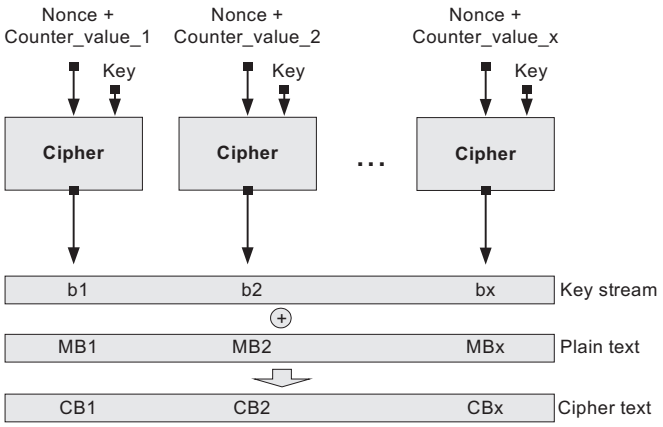


Figure 2.19: CTR

tication code Mode) is a mode or method of operation for cryptographic algorithms that allows authentication and confidentiality using a same block cipher. CCM is defined for use with 128-bit block ciphers. AES-CCM algorithm has a large number of applications, for example, the IEEE 802.11i standard formally replaces Wired Equivalent Privacy in the original IEEE 802.11 standard with a protocol using AES-CCM [Manral, 2007]. It is important to highlight that complex mathematical operations are used in AES-CCM algorithm, being necessary to propose hardware architectures with high performance and high throughput/area ratio.

For purposes of this project, AES and AES-CCM algorithms, and security schemes of the IEEE 802.11i-2004 and IEEE 802.16e-2005 standards are revised in the next sub-sections, which are established for the most important wireless networks.

### 2.5.1 AES Algorithm

The AES algorithm is a symmetric block cipher that can process data blocks of 128 bits, and it uses cipher keys of 128, 192, and 256 bits [FIPS-197, 2001]. In this project, a hardware architecture of the AES algorithm is developed for ciphering 128-bit data with 128-bit keys, see Fig 2.22. These lengths are selected, because these are required in the IEEE 802.11i-2004 and IEEE 802.16e-2005.

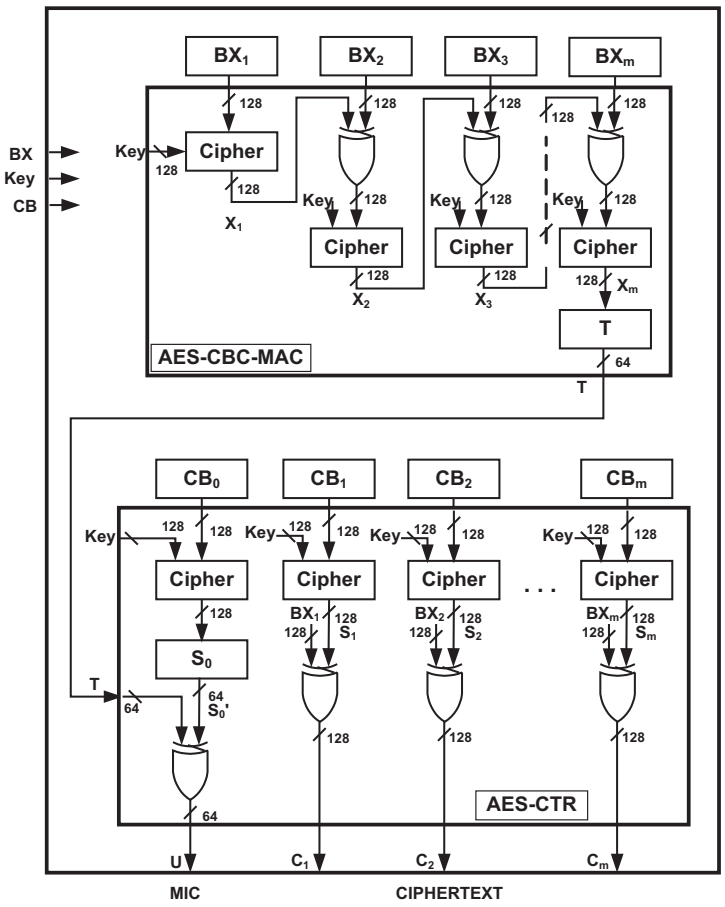


Figure 2.20: CCM

Each input block is grouped and operated as an array of bytes, termed the state array, which changes during the eleven rounds. The state array has a dimension of 4x4 (4 rows and 4 columns) or 128 bits. The basic unit is a byte and all bytes are interpreted as finite field elements, which are added and multiplied. In polynomial representation, these operations are computed in Galois Field $GF(2^8)$. In this context, results of the operations are ensured as a binary polynomial of degree less than 8, and are represented by a byte [Chaves et al., 2006]. In general, to cipher a data block, firstly, an initial round is executed by computing an XOR operation between key and data block, next, nine rounds are computed by executing four transformations, and finally, the last round is executed omitting the third transformation.

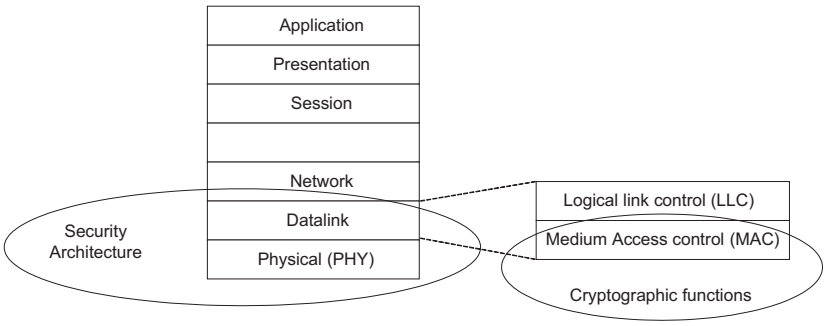


Figure 2.21: Security protocols executing cryptographic operations on the MAC sub-layer

The four transformations are, see Fig 2.23:

1. byte-to-byte substitution (SubBytes),
2. rotation of rows (ShiftRows),
3. mixing of columns (MixColumns), and
4. addition of round key (AddRoundKey).

In SubBytes transform, each byte of the array is replaced by its substitute, which is selected from an S-box. In the ShiftRows transform, the byes are re-arranged and shifted in a specified form. In MixColums transform, sixteen multiplications are computed by two constant elements: 02 and 03. The state array and a constant matrix are computed in multiplication of matrices, requiring additions in $GF(2^8)$. The final transform is AddRoundKey, which computes XOR operation between state array and key for the current round.

Additionally, an important operation is Key expansion, which generates a round key for every round. The round key is re-arranged, left rotated, and transformed using the S-boxes. Finally, an XOR is performed between this result and a round-dependent constant. The key expansion operation computes a key schedule or a 128-bits key in each round. The non-linear byte substitution and key expansion operations require S-box substitution, where one byte is substituted and determined by the intersection of the row and the column. These substitution values for the byte $xy$ are defined in [FIPS-197, 2001]. After these rounds, the ciphertext is obtained.

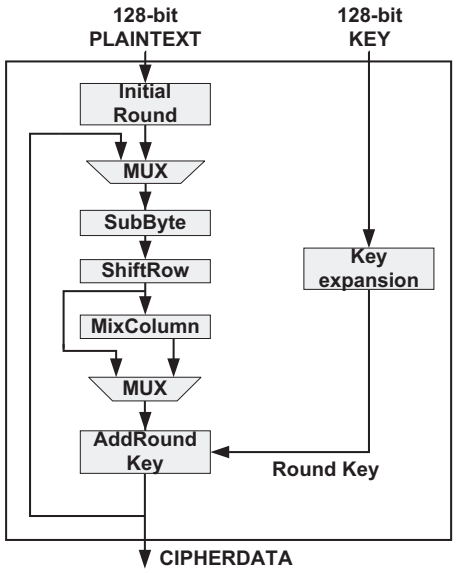


Figure 2.22: Block diagram of the AES algorithm

All AES processing in CCM encryption uses AES with a 128-bit key and a 128-bit block size, because AES-CBC-MAC and AES-CTR are constituted by this common algorithm.

### 2.5.2 AES-CCM Algorithm

In cryptography, diverse types of algorithms are focused on offering different security services, where each algorithm operates a set of mathematical rules are used for ciphering and deciphering. Traditionally, communication applications use two different cryptographic algorithms for authentication and for confidentiality, each cryptographic algorithm performing a given security service. AES algorithm in CCM mode is proposed to provide authentication and confidentiality at the same time with high security levels. Current analysis and collisions on security problems of these algorithms have led to propose new algorithms and modes of operation. Recently, CCM mode is defined and used in security schemes for wireless communication networks, such as in the IEEE 802.11i-2004 and IEEE 802.16e-2005 standards.

A number of algorithm parameters are defined in the NIST CCM specification [Dworkin, 2004]. These parameters have a fixed value and are

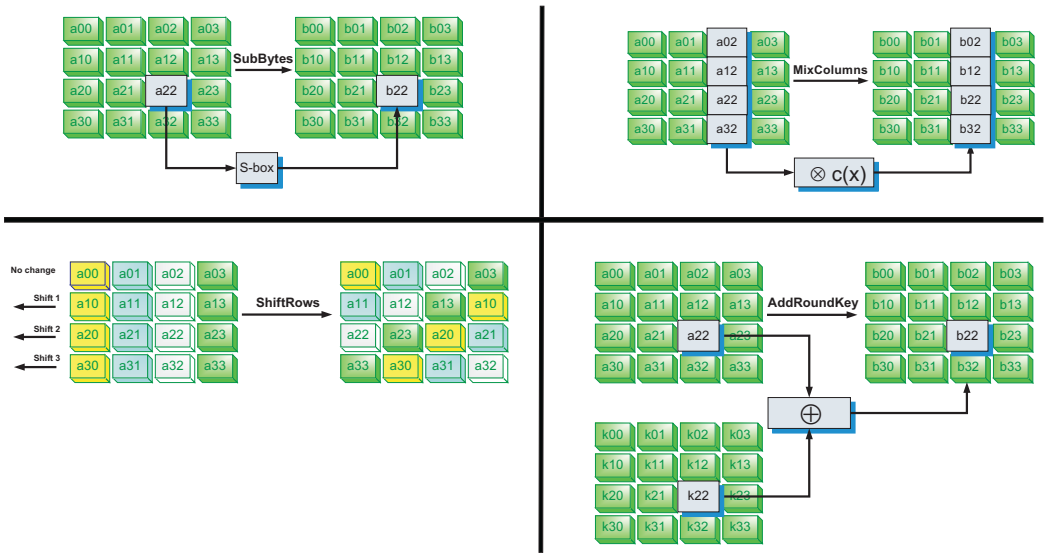


Figure 2.23: Transformations on the AES algorithm

established by each standard. For example, in IEEE 802.16e-2005, these parameters shall be fixed to specific values: i) the number of bytes in the Message Authentication Code field shall be set to 8, ii) the size of the length field Q (bit string representation of the octet length of the payload) shall be set to 2, and iii) the length of the additional authenticated associated data string shall be set to 0.

AES-CCM operates on the MAC Protocol Data Unit (MPDU) of the security schemes of the standards, which are constituted by several fields, including, for example, the payload, the length of payload, and the generic MAC header. In general, the security scheme ciphers data input (plaintext MPDU), using AES-CCM algorithm, and resulting the data output cipher MPDU. AES-CCM executes two related processes: generation-encryption and decryption-verification. For the purposes of this work, which focused to the development of a transmission platform, the generation-encryption process is considered to design the architecture.

AES-CCM is based on two modes of operation, see Fig 2.24: CBC-MAC (Cipher Block Chaining - Message Authentication Code) and CTR (Counter). CBC-MAC process is applied to the plaintext payload, the data associated AAD, and the nonce to generate a MIC (Message Integrity Code) whereas CTR mode is applied to the MIC and the plaintext payload to obtain

the ciphertext payload (cipher MPDU).

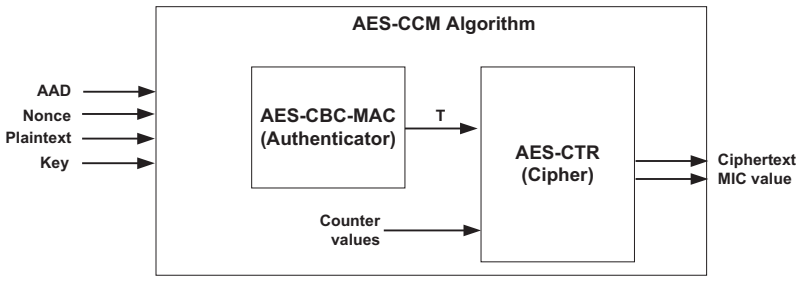


Figure 2.24: Block diagram of the AES-CCM algorithm

In general, AES-CBC-MAC takes the first block and ciphers it using AES. An XOR operation is made using the previous result with second block, and this result is ciphered. This process is applied for the remaining blocks. CBC-MAC works sequentially and it cannot be parallelized. CBC-MAC is used if there are an exact number of blocks and hence requires padding. To calculate a MIC value (T), see Fig 2.25, AES-CBC-MAC algorithm parses data input into 128-bit blocks and uses the following process: 1) Ciphers an initial 128-bit block (Block 1) with AES block cipher and the data integrity key (TK). This produces a 128-bit result or cipherdata output (X1). 2) Performs an exclusive OR (XOR) operation between the result of step 1 and the next 128-bits block over which the MIC is being calculated. 3) Ciphers the result of step 2 with the AES algorithm and TK, resulting in a cipherdata of 128 bits. 4) Performs an XOR operation between the result of step 3 and the next 128-bit block. 5) Repeats steps 3-4 for the remainder 128-bit blocks. The high-order 64 bits (T) of the final result are the MIC value.

When ciphering two identical input blocks, CTR mode produces different cipher blocks, which is based on a nonce value rather than starting it from a fixed value. This mode provides authentication by adding extra capabilities. Some properties of CTR is that ciphering can be done in parallel, decryption is the same process as encryption, and the message need not break into an exact number of blocks [Dworkin, 2004]. The AES-CTR algorithm uses the following process, see Fig 2.26, where it is necessary to parse the data input: 1) Ciphers a starting 128-bit counter (A0) with AES and TK. This produces a 128-bit result or cipherdata output S0. 2) Performs an XOR operation between the result T of the CBC-MAC process, and the first 64-bit block of the data (S0'). This produces the 128-bit cipherdata block U. 3) Increments

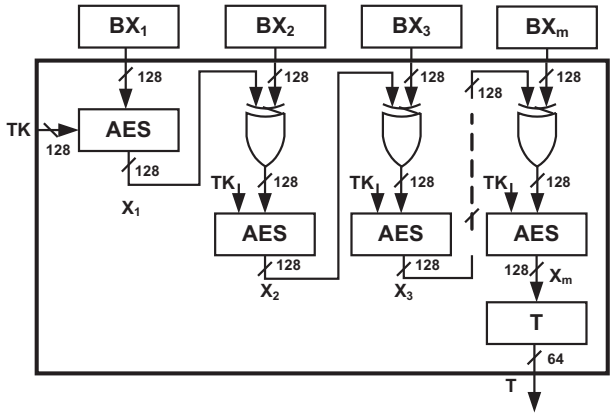


Figure 2.25: Block diagram of the AES-CBC-MAC algorithm

the counter and ciphers the next 128-bit counter value with AES and TK. This produces a 128-bit result S1. 4) Performs XOR between the result of the step 3 and the next 128 bits of the data (B1). This produces the second 128-bit encrypted block (C1). 5) Repeats steps 3-4 for the remainder 128-bit blocks. AES-CTR repeats steps 3-4 for the additional 128-bit blocks in the data until the final block is processed. Additionally, for the final block the ciphered counter is XORed with the remaining bits, producing cipherdata of the same length as the last block of data. If the last input block is smaller than 128 bits, the XOR operation is performed with the same number of bits as the block size.

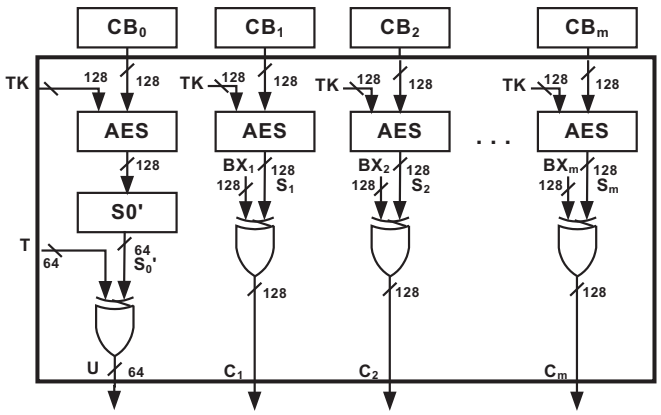


Figure 2.26: Block diagram of the AES-CTR algorithm

It is important to highlight that due to the ciphering used in a communication line or in a transmission channel, the CBC operation mode does not permit pipelined architectures, because feedback operations are performed after ciphering a block [Menezes et al., 1996], see Fig 2.27.

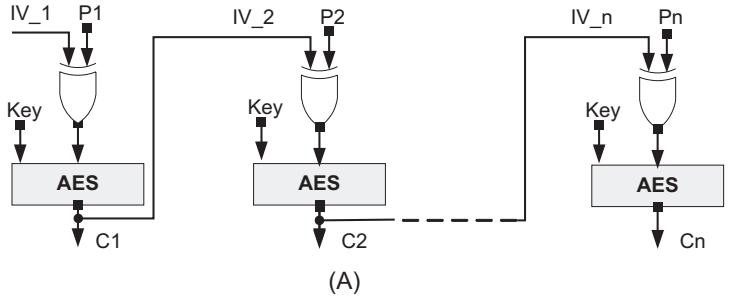


Figure 2.27: AES algorithm in the CBC mode

Security schemes of the standards add new security features, which are mainly based on the AES-CCM algorithm.

### 2.5.3 IEEE 802.11i-2004 Security Scheme

The security characteristics of the AES block cipher in CCM mode are used in AES-CCMP [LAN/MAN-Standards-Committee, 2004] to provide data confidentiality, integrity, authentication and replay-attack protection, operating on the MAC Protocol Data Unit (MPDU), see Fig 2.28. MPDU contains several fields, including, the payload, the length of payload, and header of MAC layer.

In general, the security scheme based on AES-CCMP ciphers data input (plaintext MPDU), using the AES-CCM algorithm, to produce the data output Cipher MPDU. AES-CCMP disassembles each packet in KeyID, packet number (PN), and plaintext MPDU (Medium Access Control Protocol Data Unit). Reuse of a PN with the same temporal key voids all security guarantees. A temporal key (TK) is required for every ciphering session. MPDU is expanded in three parts, 1) payload DataP, 2) Address 2 (A2) and a priority octet, and 3) MAC Header. With the elements mentioned, a CCMP Header is constructed as well as a Nonce value (unique for each frame protected by a given TK and a 48-bit PN) and additional authentication data (AAD). The payload, TK, Nonce value and ADD are input to the AES-CCM. It outputs

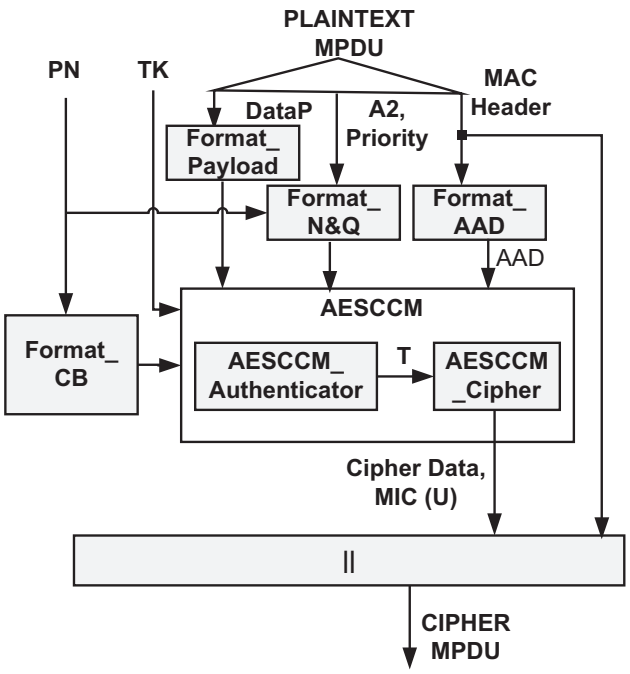


Figure 2.28: Security architecture based on the AES-CCM Protocol for IEEE 802.11i networks

the cipher data and message integrity code (MIC) that are used together with the CCMP and MAC headers to build the Cipher MPDU.

Certain main functions operating the input data should be executed before the ciphering:

1. Additional Authentication Data (AAD) construction, see Fig 2.29, provides integrity protection. Fields in the MPDU header are used to construct this value. Several bits in the fields are masked to 0. The length of the AAD is 22 octets when no A4 field and no QC filed cist, and it is 28 octets long when the MPDU includes the A4 field.

2. Nonce construction, see Fig 2.30, is formed from address 2 (A2) field, Priority field, and the packet number (PN) field. The Priority field has a reserved value set to 0.

3. Formatting of the payload makes exactly 128-bit data blocks from plaintext payload, if it is necessary more bits, these are set to 0. When

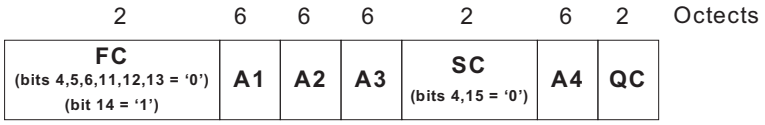


Figure 2.29: AAD construction

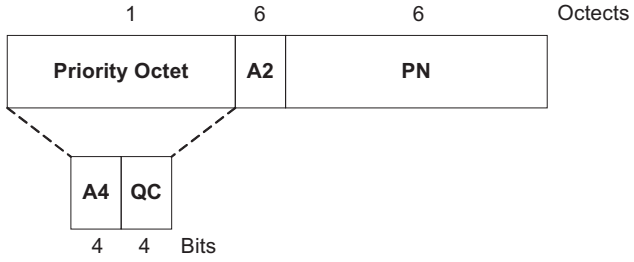


Figure 2.30: Nonce construction

ciphertext payload is obtained, the length of this ciphertext should be equal to the plaintext payload.

4. Formatting of the counter blocks, consistent with the NIST CCM specification the counter blocks are formed as shown in Fig 2.31, where Flag field is set to a fixed value and nonce is added. The last field is a counter value of 16 bits, and it is updated according to the number of data blocks formed from plaintext payload.

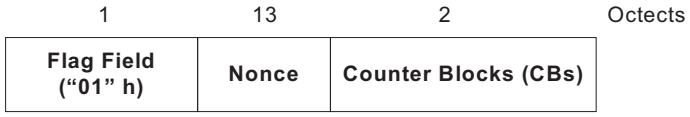


Figure 2.31: Formatting of the counter blocks (CBs)

These main functions make 128-bit data blocks that are used as input for AES-CBC-MAC and AES-CTR algorithms, excepting the AAD and Nonce constructions. These main functions and the AES-CCM algorithm are part of the proposed hardware architecture, see Chapter 5 .

AES-CCM has four inputs, see Fig 2.32: the payload, temporal key (TK), nonce value and additional authenticated data (AAD). CCM can be seen as a mode of operation of the AES algorithm, combining two cryptographic primitives: counter-mode ciphering and cipher block chaining-based authentication. It outputs the cipher data and message integrity code (MIC U). CBC-MAC mode is applied to the payload, nonce and ADD using a TK to generate an internal MIC value (T), whereas CTR mode is applied to the T, the payload and TK to obtain the ciphertext.

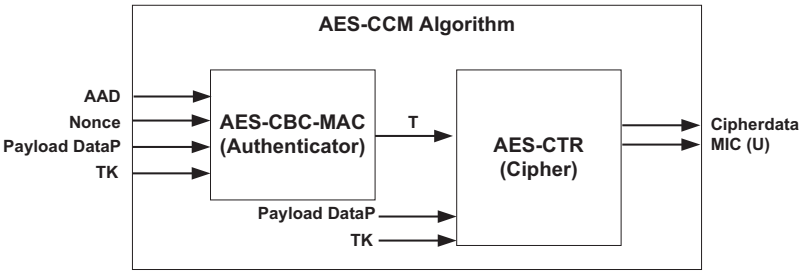


Figure 2.32: AES-CCM algorithm used in IEEE 802.11i-2004 standard

### 2.5.4 IEEE 802.16e-2005 Standard using AES-CCM

IEEE 802.16 e-2005 standard establishes several cryptographic methods, and the security is defined in a sub-layer, enabling that communication networks provide privacy, authentication, and confidentiality. These security services are based on cryptographic algorithms. These algorithms use several iterative mathematic operations and protect data transmissions with high computational costs.

The security scheme of the IEEE 802.16e-2005 networks has two component protocols: key management (PKM) and encapsulation. Different cryptographic algorithms in several modes of operations, such RSA, AES in CCM mode, and DES in CBC mode, are established as a set of capabilities within the MAC security sublayer for providing the security services. In the general operation of the encapsulation protocol, ciphering is applied to the MAC PDU payload for privacy service, whereas in the PKM, this protocol allows for authentication. In the encapsulation, data are protected by ciphering the information or plaintext payload, and by providing a value for the message integrity. For the purposes of this work and considering the high

level of security, the aim is to provide a hardware architecture based in the AES-CCM algorithm, which is widely used with high levels of security.

Ciphering payload requires that two values shall be appended: packet number (PN) and message authentication code (MIC), and AES-CCM algorithm shall be applied to the plaintext payload, see Fig 2.33.

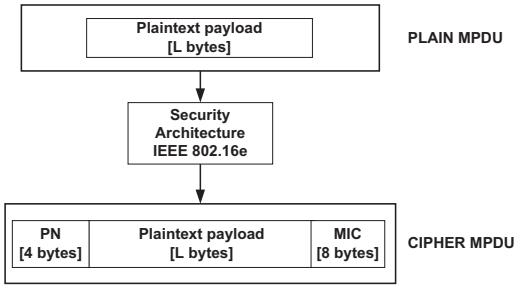


Figure 2.33: Ciphertext payload using AES-CCM algorithm in the security scheme

For applying AES-CCM algorithm, other related main functions are executed, formatting data input such as plaintext payload, counter blocks, initial block, nonce value, packet number (PN), and generic MAC header (GH-MAC), see Fig 2.34. These functions are described in the security scheme of the standard.

Certain main functions operating the input data should be executed before the ciphering:

1. Nonce construction, see Fig 2.35, is formed from the generic MAC header (omitting HCS field), the reserved bytes (set to 0) and the packet number (PN) field.

2. Formatting of the initial block, consistent with the CCM specification [Dworkin, 2004], the initial block B0 is built as shown in Fig 2.36, where Flag field is set to a fixed value, and nonce and length of plaintext payload in bytes are added.

3. Formatting of the payload and Formatting of the counter blocks are executed consistent with the NIST CCM. These processes are similar to those described in the Sub-section 2.5.3.

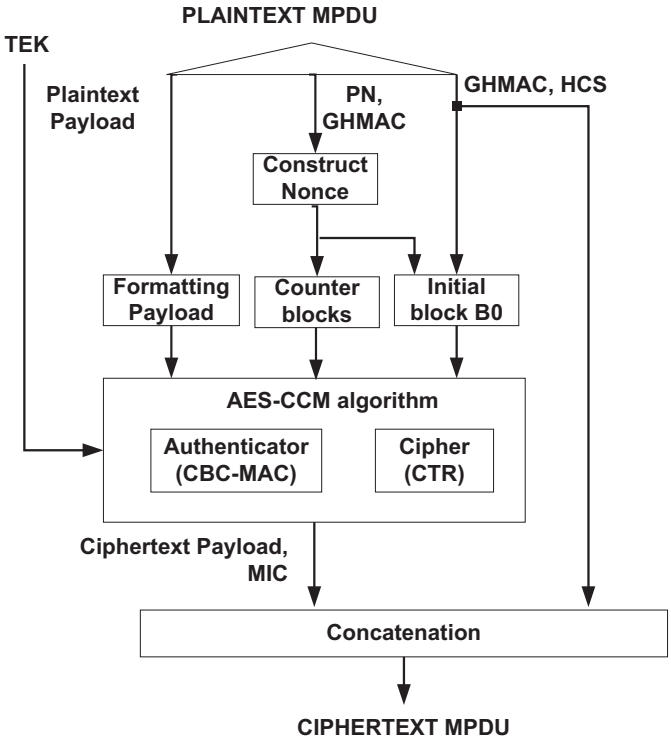


Figure 2.34: Related processes for ciphering in the IEEE 802.16e-2005 standard

Excepting the Nonce construction, the other main functions make 128-bit data blocks that are used as input for AES-CBC-MAC and AES-CTR algorithms. These main functions and the AES-CCM algorithm are part of the proposed hardware architecture, see Chapter 5.

In AES-CCM algorithm, a number of parameters are defined in the NIST CCM specification [Dworkin, 2004]. In IEEE 802.16e-2005, these parameters shall be fixed to specific values: i) the number of bytes in the Message Authentication Code field shall be set to 8, ii) the size of the length field Q (bit string representation of the octet length of the payload) shall be set to 2, and iii) the length of the additional authenticated associated data string shall be set to 0.

AES-CCM operates on the MAC Protocol Data Unit (MPDU), see Fig 2.37. MPDU contains several fields, including, for example, the payload, the length of payload, and the generic MAC header. In general, the security

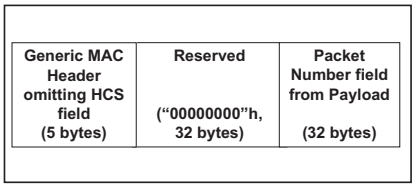


Figure 2.35: Nonce construction

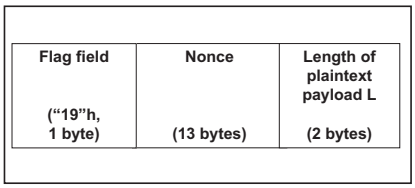


Figure 2.36: Formatting of the initial block

scheme ciphers data input (plaintext MPDU), using AES-CCM algorithm, and resulting the data output cipher MPDU. The payload, traffic encryption key (TEK), and nonce value are input to the AES-CCM. It outputs the ciphertext payload and message integrity code (MIC) that are used together with modified generic MAC headers to build the cipher MPDU.

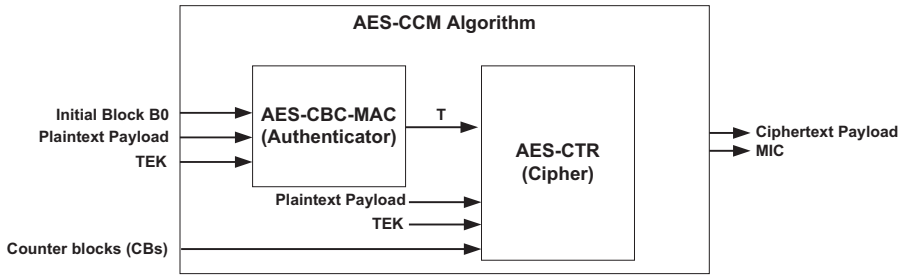


Figure 2.37: Block diagram of the AES-CCM algorithm

Additionally, security protocols have not a fixed cryptographic algorithm, but they are independent of the algorithm, describing a set of algorithms to be used. Here, considering that cryptographic algorithms execute multiple operations in iterative processes and use them causes bottlenecks in transference of data, in most of the cases, hardware implementations report better performance than software ones. However, software architectures have better flexibility than hardware ones.

## 2.6 Reconfiguration and Hardware Architectures

Recently, the hardware architectures have a fixed configuration, which provide one or more functions or tasks. These architectures for hardware implementations can not be upgraded or updated, but offer a high performance. The architectures for software implementations, which are based-processor systems, present a high flexibility with a poorer performance, where several functions or tasks can be operated. Cryptographic algorithm agility, or the capability to switch between several encryption algorithms, is a desirable feature of new communication systems due to the recent security protocols defined as algorithm independent [Paar, 2000]. A promising answer to algorithm agility in hardware is the reconfigurable logic with advantages due to the hardware characteristics. Architectures of cryptographic algorithms in hardware have several advantages over software architectures; the main one is the high processing speed. Hardware architectures on ASICs lack flexibility, whereas software architectures have a low throughput, see Fig 2.38. Here, reconfigurable architectures equilibrate the disadvantages of both software and hardware architectures, delivering high throughput with high flexibility, by establishing a correct hardware design methodology to provide high-performance hardware architectures.

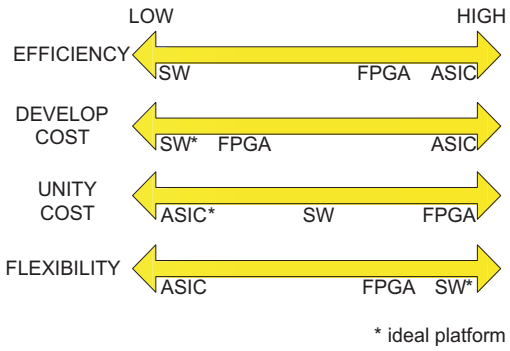


Figure 2.38: Advantages of the reconfigurable architectures [Paar, 2000]

Throughput is an important feature of the cryptographic implementations, and in the communication networks, like Gigabit Ethernet require processing speeds of 1 Gbps and it is expected that also future wireless personal area networks perform at these data rates [Quinn et al., 2005]. These

networks require flexible, high throughput systems which compute cryptographic algorithms that are more efficiently implemented in custom hardware that in software running on general-purpose processors (GPPs) [Umamaheshwari and Shanmugan, 2004].

In this project, cryptographic hardware architectures based on the AES algorithm are proposed, which combine parallelized structures with minimal hardware resources. This is reached by making an analysis to reduce critical path by developing specialized modules, proposing compact control units, identifying parallelization of the data buses and modules, and balancing paths formed by the combinational and sequential elements. For evaluation purposes, these architectures are implemented in FPGA devices, reporting high hardware implementation efficiency. Further of the operations of this algorithm, other iterative processes should be made to the data input in the security scheme. This set of operations causes bottlenecks in the data transmissions, and architectures with high throughput are required, considering future data transmissions of 1 Gbps, such as in the wireless networks [Guo, 2007], with application to transmit high-quality TV, movies in DVD, and great amount of digital files using personal computers, among others.

Also, the hardware implementations offer more security than software ones because they cannot be as easily read or modified by an outside attacker [Bertoni et al., 2004]. Implementing cryptography in FPGA devices provides a good alternative to custom and semi custom ASICs (Application Specific Integrated Circuits), which have an expensive and time-consuming fabrication, and more inflexibility or parameter switching [Gaj and Chodowiec, 2000], and GPPs and special-purpose processors, like DSPs (Digital Signal Processors) [Li et al., 2000], that offer lower performance. The advantages of the FPGA reprogrammable devices are especially prominent in security applications, where new security protocols decouple the choice of cryptographic algorithms from the design of the protocol, and users select the cryptographic standard to start a secure session.

There are two metrics to evaluate a hardware architecture, which focused to the cryptographic implementation: throughput and efficiency. The throughput is computed by the Eq. 2.1 [Gaj and Chodowiec, 2000], and it indicates how many plaintext blocks can be processed by time unit (bits per second, bps).

$$Throughput = \frac{Plain\_data\_block\_size}{Clock\_period * Clock\_cycles} \tag{2.1}$$

The other metric is the implementation efficiency, and it is a measurement of this type of cryptographic hardware implementations, which is defined as the ratio between the reached throughput and the number of slices that each implementation consumes. The efficiency is computed by the Eq. 2.2 [Kitsos, 2006].

$$Efficiency = \frac{Throughput}{Used\_hardware\_resources} \tag{2.2}$$

## 2.7 Summary

Hardware architectures have been proposed as an ideal element to offer security and to support software-radio systems. These architectures report high performance and flexibility, moreover, the reconfigurable ones present characteristics of flexibility with specialized configurations. The modern research works in these areas focus on developing systems and tools to support reconfigurable hardware architectures for software radio, which mainly operates for the lower layers of the OSI model. Wireless communications and mobile commerce have motivated the developing of new security network technologies, using multiple specialized cryptographic functions and increasing the complexity of network systems. Existing architectures execute one or several tasks for a particular protocol by means of a fixed platform, making difficult to update or modify its functionality for new applications or protocols. It seems that architectures based on the SR concepts may offer the possibility of modifying its functionality to operate with different existing and future protocols. This can be achieved by providing a reconfigurable hardware platform that can be reprogrammed to perform previously defined tasks.

# Chapter 3

# State of the Art

Various applications in the wireless communications need diverse characteristics in the data transmissions, requiring devices with different technologies and capabilities, see Fig 3.1 [Telecommunications-Technology-Association, 2004].

Considering the ideal radio, this great number of characteristics should be considered and supported by the software radios. In this way, research focuses on different topics such as proposing platforms with characteristics of software-radio functionality, high throughput, optimal architectural designs of the wireless standards, high flexibility, high spectral efficiency, support for the MIMO (Multiple-Input Multiple Output) systems, or multiprocessing of several channels.

To develop works on the previous topics, where the typical wireless communication system is composed of diverse main blocks (see Fig 3.2), it is necessary to explore in the different elements of the typical system, examining configurations and algorithms to reach optimal results to provide the detailed characteristics.

For example, in the transmitter side, the main operations typically consist on taking data from the medium access control (MAC) sub-layer and then scrambling, ciphering, encoding, modulating and pre-compensating. In the receiver side, the operations are more complex; it estimates, demodulates, detects errors and decodes to recover the received data. The algorithms used by the receiver involve sophisticated signal processing. Other important elements are the digital intermediate frequency (IF) standards, which define the physical-layer interface, and higher-level protocols necessary for moving digitized signals between the radio frequency (RF) front-end and the baseband

Figure 3.1: Capabilities for supporting applications of the wireless communications

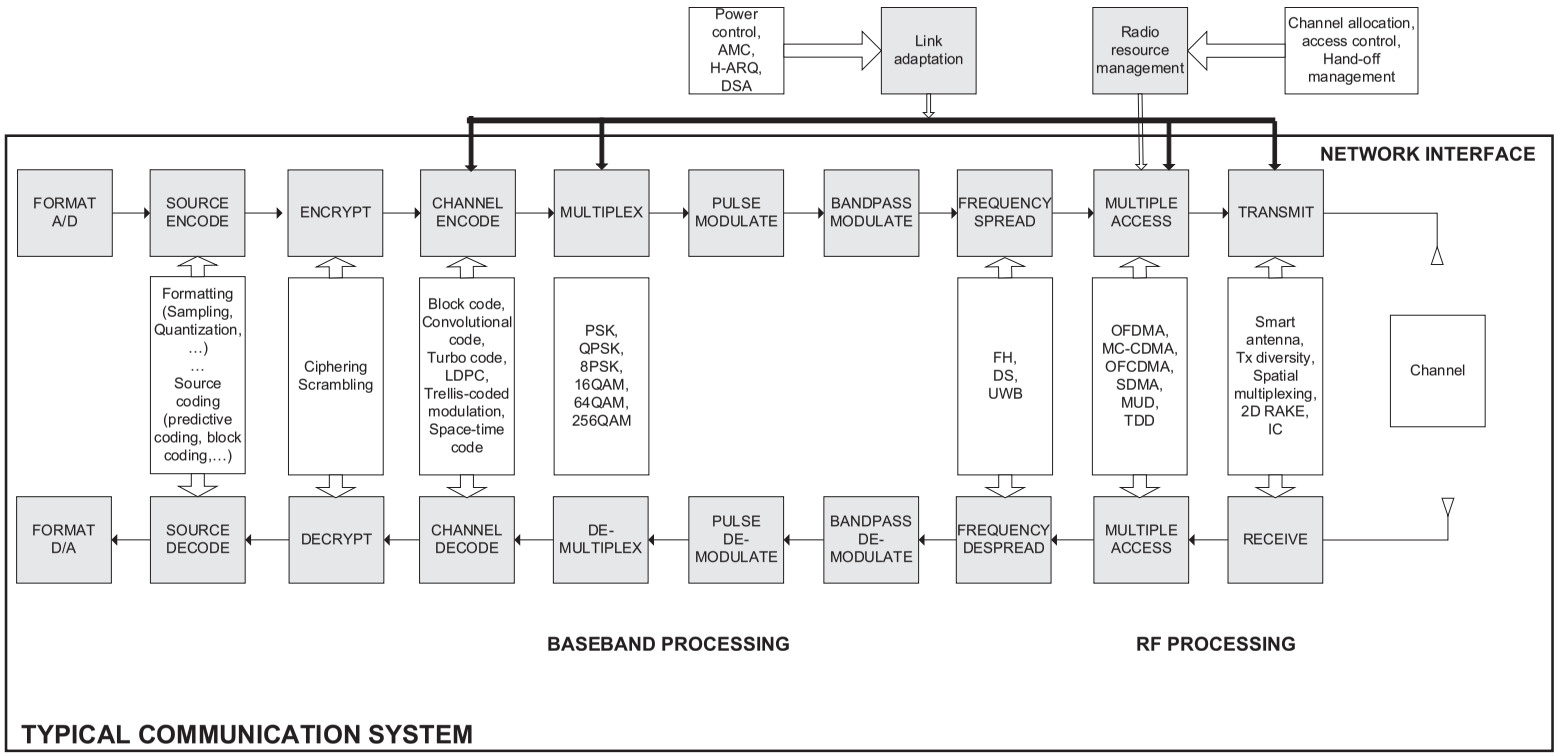


Figure 3.2: Block diagram of the typical communication systems

processing in the digital radio architecture.

Moreover, software defined radio can be viewed simply as an implementation technique in which signal processing hardware is replaced by programmable devices such as DSPs or FPGAs. In the broader perspective, software defined radio is a collection of hardware and software technologies that enable reconfigurable wireless infrastructure and user terminals. It is an enabling technology that is applicable across a wide range of areas within the wireless industry.

Several works have proposed specific solutions for software-radio systems (see Section 3.1) and security with cryptographic multi-functionality in software/hardware architectures (see Section 3.2).

## 3.1 SR Systems

Software-based systems will be capable of replacing traditional hardware systems such as FM radio, TV broadcast or cell based phone systems. Architectures have many real-time requirements for several algorithms in new emerging wireless standards, where efficiency and flexibility are main features. Due to the diverse layers and sub-layers, the next methodologies and architectures report results about different research sub-areas (see Fig 3.3), and in this project, the next works are classified and selected due to their multi-functionality. They report architectures supporting multiple functions for communication system with software-radio applications.

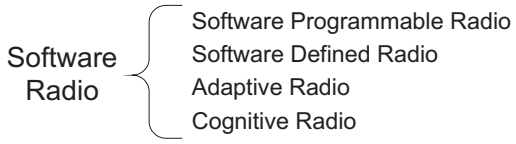


Figure 3.3: Related works on software radios

Several architectures show multi-functionality, where only certain modules can change their operation, considering given parameters or their complete operation. For example, the WF flexible modem architecture in [Bogucka et al., 2002] proposes that some baseband operations are adjustable with adaptive transmission parameters such as modulation size, code rate, code block size, transmission power and number of active sub-carriers. On the receiver side, advanced signal processing includes adaptive algorithms for the

equalization, the phase noise/residual frequency offset compensation and the iterative decoding. This architecture focuses on minimizing the power consumption of user equipment, changing different baseband operations. Also, in a similar goal but in a specific application: i) [Lin et al., 2006] reports a reprogrammable architecture, which supported multiple protocols focusing on low power consumption and is based on the SIMD parallelism, and ii) [Panigrahi et al., 2002] presents a multimedia radio architecture that reconfigures the source coder, the channel coder, the RF modulator and the power amplifier modules for adaptive wireless image communication over the JPEG image compression algorithm. The obtained results are the minimization of the energy consumption. However, [Komara, 2004] reports an adaptive base station with multiple carriers that handles several cellular standards, where its architecture is composed of modules for MAC, PHY, and RF/IF (Radio Frequency/Intermediate Frequency) processing. An important point is that the architectures are being supported in the FPGA devices, which enables to change configurations, such as in i) [Cavallaro and Radosavljevic, 2004], which reports an ASIP architecture based on TTA (Transport Triggered Architecture) and presented for wireless applications and synthesized for Xilinx FPGA boards, ii) [Bhatia, 2004] that presents an approach for the implementation of a baseband radio architecture, which is utilized on wireless data transfer applications, considering a modular design to reconfigure a part of FPGA in order to support different modulation and demodulation schemes, iii) [Brodersen et al., 2004] that proposes to design and build a multi-purpose computing platform, which will offer an FPGA-based hardware architecture and software design methodology that target a range of real-time radio telescope signal processing applications, and iv) [Tikkanen et al., 2000] reports a platform with several hardware modules, executing certain tasks for MAC and PHY processes. It uses elements such as FPGA, DSP, a radio board, and a computer, and its main function is to adapt non-synchronized data streams by changing the modem configuration.

These last works report architectures that can be modified to have reconfigurable structures, and their functions are based on baseband operations, which can be applied to the software-radio applications, and specifically, for cognitive radios. However, reconfigurable architectures for communication networks are being developed, such as in [Berlemann et al., 2005], that focuses on a generic protocol stack. This wireless communication system changes its configuration for applications of software radios. In [Tuan et al., 2001], it is introduced a design methodology for wireless protocol processor

design, with the focus on exploring a reconfigurable platform.

These reconfigurable architectures are interesting due to their associated design and methodology, moreover, they can be used for the software-radio concepts, but, for purposes of this project, architectures focused on software radio are also interesting, for example, in [Jackson et al., 2004], a methodology for designing an Application - Specific Integrated Processor (ASIP) architecture in FPGA with SDR application is described. Here, the FFT (Fast Fourier Transform) and FIR (Finite Impulse Response) modules are considered for the reconfiguration. Other example is [Medina et al., 2006] that presents a model with IF stage tasks and processes such as beam switching that are performed by a programmable processor. These architectures change the configuration of certain modules for obtaining different functions, but they are based on a basic radio platform, which does not happen in the results of the works reported in [Carpenter et al., 2005] and [Ryser, 2005]. The Software GPS Receiver reported in [Carpenter et al., 2005] operates at different frequencies using up to eight Digital Antenna Elements, with other basic communication waveforms for SDR applications. It has a large functionality in software and uses a reprogrammable hardware platform that can also be configured to perform other communication functions. Further, the sensors compatible with this system can provide GPS, wireless, inertial and image information for a diverse set of applications. On the other hand, the architecture [Ryser, 2005] is of a Software Defined Radio system running on embedded Linux in a single FPGA, which can be adapted to different standards using different external outputs or inputs. The author reports an example of a digital TV broadcast receiver. The result is a working SDR application using hardware, software, and a fully embedded operating system, which is constituted of an FPGA with two integrated PowerPC CPUs. The partial reconfiguration is used to adapt different standards, such as PAL, NTSC and HDTV video signal formats or/and AC-3 and MPGE-2 audio formats or/and ATSC 8-VSB, DVB-T COFDM and ISDB-T BST-OFDM video broadcasting. The SDR concept is applied to satellite payloads.

In the previous works, CaR, SPR and SDR architectures have been described for applications of the software-radio concepts, and the last two works with more capabilities to be adapted like AwR, AdR or CR architecture, due to their structure with sensors or with multiple inputs/outputs. For example, [Clancy, 2006] examines the problem of dynamic spectrum access, for implementation of a cognitive radio system.

In general, the described works focused on different baseband operations

that allow multi-functionality based on a simple or complex radio architecture, but the security is not considered. Due to the security characteristics according to the OSI model, see Section 2.3, the commercial and research works focused on cryptographic solutions. Next, cryptographic architectures are presented, which report high flexibility.

## 3.2 Security Systems

Research on hardware/software architectures for diverse security purposes for software radio is being developed, and in this project, the revision focuses on the cryptographic multi-functionality, see Fig 3.4. A great amount of works that execute multiple cryptographic operations have been designed. In Section 3.2.1 some works are described; and a small number of works that operate on multiple security protocols are described in Section 3.2.2.

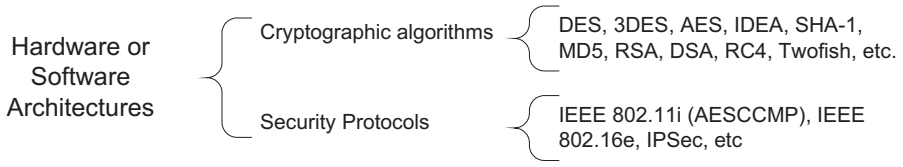


Figure 3.4: Related works on security architectures

### 3.2.1 Works on Multiple Cryptographic Algorithms

In this section and for purposes of this project, the cryptographic architectures with reconfigurable structures are described, but without application in software-radio systems.

Reconfigurable computing provides hardware cryptographic architectures with high performance and flexibility, reporting multi-functionality processor [Kim and Lee, 2004], [Wu et al., 2001], [Taylor and Goldstein, 1999] and with a specialized architecture [Elbirt, 2002], [Algredo-Badillo, 2004]. [Kim and Lee, 2004] reports a reconfigurable private and public key crypto-processor for a security system. In [Wu et al., 2001], an architecture with four specialized units and four stages is described. It executes 3-DES and AES cryptographic processes. The reconfigurable cryptographic processor, reported in [Taylor and Goldstein, 1999], is based on the pipelined architecture, implementing

Crypton, IDEA, RC6 and Twofish cryptographic algorithms. [Elbirt, 2002] reports a reconfigurable architecture, based on specialized modules to compute different cryptographic operations. RC6, Rijndael, and Serpent cryptographic algorithms were implemented. [Algredo-Badillo, 2004] describes a reconfigurable architecture for IPSec applications, with a specialized complete structure for each algorithm.

These works are revised because they focus on design methodologies when developing cryptographic solutions. This selection is due to the cryptographic algorithms that are considered complex, and detailed revisions are required when they are used for hardware implementations and software implementations. In this way, a special design methodology is necessary when the cryptographic algorithms are mapped into reconfigurable architectures.

The key element of these architectures is the performance, where the throughput reported by the processor-type architectures is lower, and the flexibility depends on the specialized functional units to compute new algorithms. The systems with specialized modules provide architectures with high throughput and flexibility, but these require changing completely its configuration to obtain different cryptographic functions. One of the objectives is to reach high performance, because the second ones present higher throughput, being the most important to be examined.

Finally, the research about software-radio architectures focuses on several topics, most of them related to the lower layers, considering the OSI model. The security is a very important topic, which is not completely developed in the related works, requiring a hardware architecture with high throughput and flexibility, which can be part of a software-radio system, based on security schemes of the communication networks. Each network establishes different protocols and security schemes, and so, the hardware architecture should operate between different networks and protocols. This architecture should achieve high efficiency and high performance for data transference without provoking bottlenecks, and related works, which solve this problem, are described in the next section. Research work in the development of this hardware architecture includes exploring, analyzing and evaluating the different types of reconfiguration to support different cryptographic implementations.

### 3.2.2 Works on Multiple Security Protocols

Considering software radios, security and reconfigurable schemes, research works in these networks focused on different topics, some works focused on the same network. On the one hand, in relation to the WLAN networks, the base-band architecture reported in [Tell, 2005] is reprogrammable, with a structure of parallel heterogeneous processors for multiple standard such as IEEE 802.11a, b and g with accelerators for front-end operations, demapping, scrambling, CRC, interleaving, channel coding and modified Walsh transform. In addition, [Cheung, 2006] presents a commercial solution for WLAN on cellular platforms, which is focused on MAC processing, including some PHY processes and interfaces. These works are software-defined or programmable radios, but adaptive radios are also being developed, such as [Wouters et al., 2002] that reports a WLAN receiver with an adaptive architecture that modifies bit rate according to the channel conditions. On the other hand, in the WMAN networks, [Boppana, 2005] reviews the capabilities of FPGA platforms to implement WMAN standards and presents information on processing speed, flexibility, integration, and time-to-market, resulting that FPGAs offer a good alternative to implement several processes for different layers when compared to DSPs. Research in commercial products, [Altera-Corporation, 2004] presents a platform of a WMAN protocol over a FPGA platform, which stresses the need to have platforms capable of changing their configuration to adapt to new protocols.

The results of these works focus on the fact that the same platform can be used to design a complete radio with characteristics to be used in different radio concepts, although those works that explore different types of networks are more interesting. For example, [Zwart et al., 2002] reports a platform for embedded systems with hardware-software co-design focused on mobile multimedia applications, which has several peripheral interfaces, and provides a methodology for application mapping, supporting some WPAN and WLAN applications by changing its configurations of the hardware modules. Other work, [Burbank and Kasch, 2006] presents an overview of WLAN and WMAN architectures and concludes that future network devices should operate with protocols on both types of networks. Finally, in [Nilsson, 2007], a reprogrammable architecture based on SIMD (Single Instruction Multiple Data) is reported, which executes baseband processing tasks, independently of the main processor, focusing to the multi-standard functionality to get WiMAX, WLAN and WCDMA systems. The last two works are based on

the same two networks used in this project. The advantage is that these works can be used to develop a complete SR system, executing one or more specific tasks, the disadvantage is that security is not provided.

Few works are related to security architectures for wireless protocols and reconfigurable platforms [Sklavos et al., 2005], [Hi/fn-Inc., 2008], and [Gehrmann and Sthl, 2006]. About security for wireless systems, [Sklavos et al., 2005] review cost and performance for FPGA implementations of WLAN standards. Two cryptographic algorithms were considered: WEP (Wired Equivalent Privacy) and AES algorithm, which are separately implemented and compared in terms of throughput, hardware resources, operating frequency, and power consumption are other options but they do not run in a common platform. [Hi/fn-Inc., 2008] presents a commercial WiMax security processor which supports the cryptographic algorithms AES, DES, 3DES, RC4, AES-CCMP, SHA-1 and MD5, and also some compression algorithms. [Gehrmann and Sthl, 2006] report a security platform for the access and application layers. They included a crypto accelerator, which executes different cryptographic algorithms such as cipher algorithms (DES, 3DES, and AES), hash functions (MD5 and SHA-1), and PKI (Public Key Infrastructure) engine. This architecture has application on 2G (second generation) standards, performing several operations for UMTS (Universal Mobile Telecommunications System) and GPRS (General Packet Radio Service).

These works present architectures operating in a same type of network, and an important idea is to explore security schemes of different networks, proposing architectures with optimal characteristics with the aim of being used on software radios.

## 3.3 Outline of the Thesis Project

There are many research works focused on SR and cryptographic multifunctionality systems, but just a few works focus on the security for SR systems. Most authors present specific solutions for software radio concepts, from baseband processing to MAC operations using software defined radio and cognitive radio architectures. Some authors designed application specific processors while others use general purpose processors with accelerators. In some cases, several devices such as processors, FPGA, DSP or/and ASIC, containing reconfigurable, reprogrammable or selectable modules are used. Although, it is generally accepted that having processing platforms to sup-

port security functions for several wireless network protocols is required, just a few works have been reported.

In these related works, it is important to highlight that architectures based on processors report high flexibility but poor performance, contrary to the application-specific hardware architectures which report better performance with a poor flexibility; the reconfigurable architectures equilibrate these characteristics, although design methodologies are necessary to provide architectures with high efficiency based on reconfiguration models. In this project, the software-radio platform explores reconfiguration schemes based on security schemes just standardized and evaluates hardware architectures for these security schemes. The proposed methodology in this work plans an examination of two communications standards, then an analysis to evaluate the best reconfigurable platform, the selection of the best reconfiguration scheme for a module with software radio functions and finally, the development of the complete platform.

Considering initial specialized hardware architectures (Chapters 4 and 5), which reports high implementation efficiency, this platform focused in providing hardware architectures with high throughput (Chapters 6), taking into account that designing reconfigurable architectures reports smaller performance than the specialized architectures.

# Chapter 4

# Initial Phases of the Design

In the current wireless communication world, there are many applications such as 3G cellular phones, e-commerce solutions, Wi-Fi (Wireless Fidelity), Bluetooth, Wi-Max, Internet access, which use different networks and communication protocols. Several of these applications can be supported by a device, for example a cellular phone with Bluetooth. It has specific hardware to execute these applications, running at the same time, without possibilities to increase other applications. This flexibility is an ideal characteristic of a complete device. In this way, software-radio concepts propose to use a radio with additional intelligence, operating in multiple environments. A radio with this flexibility can enter into diverse networks, presenting security issues, because it can be an attacker. Furthermore, wireless communications use air like transmission channel, which is insecure, therefore this radio can be attacked by intercepting, modifying, fabricating or interrupting the transmitted data.

Software radios will be very useful, but they should prevent these security issues. In this work, a software-radio platform is proposed to support two security schemes. The aim is to offer security services, based on communication standards, through a reconfigurable hardware architecture with high throughput and flexibility.

The design and development of the reconfigurable platform is divided conforming to the phases of the methodology (see Section 1.6). The design methodology makes use of a modular approach for the design of the reconfigurable platform, based on reusable modules and focused on completing the general and particular objectives. The revision and analysis of the wireless communication networks are part of the methodology in order to select an

initial set of security schemes, which enables to probe the idea of a reconfiguration scheme for a software-radio platform, focusing in high-efficiency hardware architectures. In this Chapter, Phases I, II and III are described, revision and analysis of the communication networks and their protocols are detailed, and also selection and software design of the two communication protocols considering their security schemes are revised.

The Phase I is to select a set of security wireless standards. The selection is based on the analysis of the most representative standards for two types of communication networks. After selecting the standards, in the Phase II, their security schemes are analyzed to identify the cryptographic operations. The Phase III according to the previous phase enables to design security architectures. These software implementations will be developed with modular designs, simulating the proposed hardware architectures. In the Phase IV, security hardware architectures will be developed with modular designs for the selected standards, evaluating the high hardware implementation efficiency. At this phase, results such as throughput, performance, hardware resources, and critical path time will be obtained. These results will allow the evaluation of the reconfigurable security architecture for the software-radio platform. In the final phase, a reconfiguration scheme will be proposed and evaluated by implementing a processing platform to support the security architectures of the selected standards using the SR concept. Several parameters such as execution time, area occupied, throughput and reconfiguration delay will be obtained.

## 4.1 Phase I: Revision and Analysis

In the Phase I, review of the state of the art, and analysis of the protocols and wireless communication networks are made. There are several communication networks, which have emerged, evolved, coexisted in the communications environment. These have many protocols, see Fig 4.1. For example, Bluetooth, HiperPAN, and UWB (Ultra-Wide Band) are protocols for WPANs, but an ideal platform should change its functions between different types of networks (vertical arrows) or different types of protocols (horizontal arrows). The proposed reconfigurable architectures of this project should change their configurations to provide an ideal function, operating in several type of networks and several particular cryptographic functions, and considering security schemes based on standards.

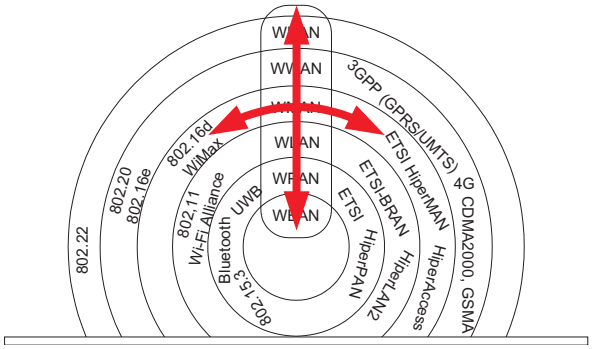


Figure 4.1: Each type of wireless network has several communication protocols

In Table 4.1, some representative communication protocols for certain networks are shown to have different applications and characteristics. For the development of the software-radio platform, it is considered secure communication protocols of the most important wireless communication networks, which have already been standardized.

Table 4.1: Communication protocols

| | |
|---|---|
| IEEE 802.22 - WRAN<br>[Weissberger, 2005] | 2004, 40Km<br>20 Mbps |
| IEEE 802.16e - WMAN<br>[LAN/MAN-Standards-Committee, 2005] | 2005, Hundreds of meters<br>75 Mbps |
| IEEE 802.11a - WLAN<br>[LAN/MAN-Standards-Committee, 1999] | 1999, 35-120 m<br>54 Mbps |
| IEEE 802.11i - WLAN<br>[LAN/MAN-Standards-Committee, 2004] | 2004, up to 100 meters<br>54 Mbps |
| IEEE 802.15.1 - WPAN<br>[LAN/MAN-Standards-Committee, 2005] | 2005,1-100 m<br>11-55 Mbps |

Due to the variety of the protocols, the revised protocols are based on an important characteristic (to select an initial set of protocols): their security architectures. Security features of the 802.22 standard are undefined (authentication, authorization, message integrity and data encryption) and they are expected to be included in the final draft of the standard. For IEEE 802.16a/d standards, they are based on the 56-bit Data Encryption Standard (DES), but they do not provide adequate protection against data forgery or replies. For the new standard, IEEE 802.16e-2005, it is implemented 128-bit Advanced Encryption Standard (AES) in CCM mode, which is generally considered a strong standard. An explicit packet numbering scheme is also implemented to prevent replay attacks. The data sent in IEEE 802.20 networks are encrypted with public keys generated by the AES 128-bit algorithms; data-integrity and authentication services will be included in the standard.

For 802.11a/b/e/f/g/h networks, their security is based on IEEE 802.10 standard, which specifies security association management and key management, data confidentiality and data integrity. For example, IEEE 802.11b provides access control and ciphering services based on WEP (Wired Equivalent Privacy). For data ciphering, RC4 algorithm is used. For IEEE 802.11i networks, AES-CCM algorithm is considered, which is stronger than RC4, further the AES-CCM provides both authentication and privacy, based on 128-bit AES algorithm. Finally, IEEE 802.15.1 can provide ciphering and authentication at the link layer. For ciphering, stream ciphers are used based on modulo-2 additions, whereas authentication on challenge-response schemes.

The revised security schemes describe mechanisms based on cryptographic algorithms to provide security services. On the one hand, some of these security schemes are proposing certain mechanisms. On the other hand, protocols already standardized use certain algorithms, which some of them are considered stronger than the others. In this project, security schemes already standardized with modern cryptographic algorithms are considered.

For the purposes of the research project, two wireless communication networks were selected: WMAN (Wireless Metropolitan Area Network) and WLAN (Wireless Local Area Network), considering the IEEE 802.11i-2004 and IEEE 802.11e-2005 standards, respectively. The selected security schemes are prominent with a flexible communication system, using advanced cryptographic technologies, which are based on AES algorithm, using it the modern mode of operation: CCM. The security schemes are different, detailing individual processes, data formats and cryptographic operations. Task iden-

tification is part of the design methodology to identify functional elements, allowing the best selection of the reconfiguration scheme. Particular and common functional elements should address several aspects for the problem of designing a reconfigurable platform.

The selected security schemes are prominent with a flexible communication system. Multiple cryptographic algorithms can be used, see Table 4.2, and in this project, security schemes using advanced cryptographic technologies which are based on AES-CCM algorithm, are considered.

Table 4.2: Cryptographic algorithms used on communication protocols

| Service/Protocol | IEEE 802.16e | IEEE 802.11i |
|---|---|---|
| Confidentiality | DES-CBC<br>AES-CCM<br>DES (TEK) | TKIP (RC4)<br>WEP (RC4)<br>AES-CCM<br>NIST key WRAP |
| Integrity | Any algorithm<br>HMAC-SHA-1<br>HMAC-MD5 | HMAC - SHA-1<br>HMAC - MD5<br>TKIP (Michael MIC)<br>AES-CCM |
| Data key exchange | AES<br>1024-bit RSA<br>3DES | IEEE 802.1X<br>and manual |
| Key generation | | HMAC-SHA-1<br>RFC-1750<br>Propietary |

To explore an adequate reconfiguration scheme of the hardware architecture, the security schemes of the two communication protocols will be analyzed and implemented. Firstly, the task identification, Phase II, will be made (see Section 4.2) to group particular functions, and next, the design and the development of the software implementations, Phase III, are executed (see Section 4.3) to simulate each particular function and to validate the general process of the security schemes of the standards.

## 4.2 Phase II: Task Identification

From the selected communication protocols in Phase I, two security schemes were chosen to design and develop the software-radio platform: IEEE 802.11i [LAN/MAN-Standards-Committee, 2004] and IEEE 802.16e-2005 [LAN/-MAN-Standards-Committee, 2005]. The aim of this phase is to identify functional groups that enable designs of the hardware architectures with high performance. The two security schemes are executed in MAC sub-layer. For example, Fig 4.2 shows a block diagram of the security scheme for the IEEE 802.11i networks.

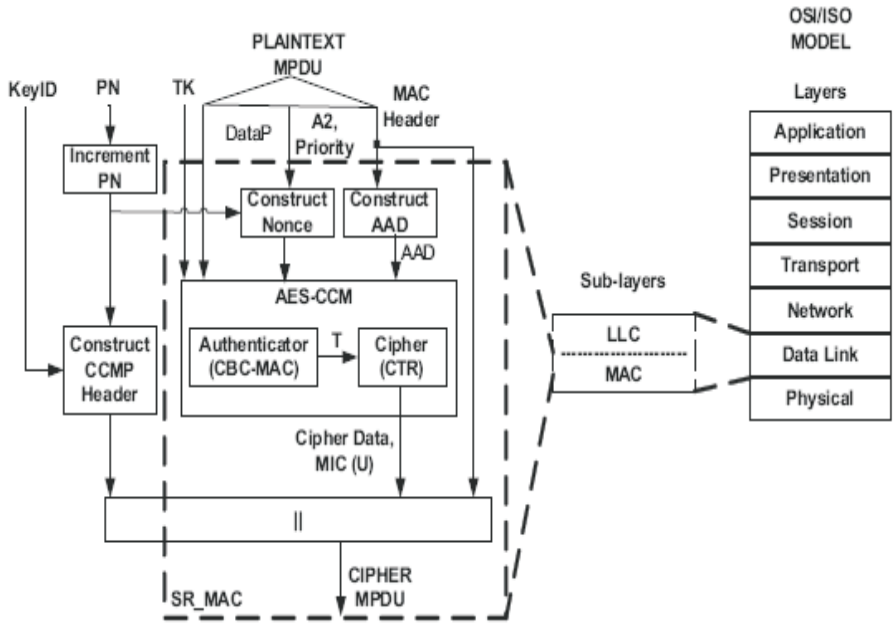


Figure 4.2: Security scheme of the IEEE 802.11i networks

For the IEEE 802.11i security scheme, five tasks were identified:

1. Format Nonce and Counter,
2. Format Payload,
3. Format Additional Authentication Data (AAD),
4. Format Counter Blocks, and

5. AES-CCM algorithm.

Whereas for the IEEE 802.16e-2005 security scheme, six tasks were identified:

1. Format Initial Block (B0),
2. Format Counter Blocks,
3. Construct Nonce,
4. Modifying GHMAC,
5. Format Payload, and
6. AES-CCM algorithm.

These tasks were identified by examining the security schemes. The analysis starts by applying a task decomposition, which offers basic functions. These were selected and grouped to the identified tasks. Thus, each identified task can be designed, implemented and simulated. Finally, examinations on these tasks (considering functional blocks and identified data buses) together with approaches of hardware design enable to propose design methodologies and to develop the proposed hardware architectures. IEEE 802.11i and IEEE 802.16e-2005 have a common part: the AES-CCM algorithm. Each of the security schemes has processes to format data and to use the AES-CCM algorithm. To design the proposed AESCCM architecture, hardware architectures of two basic algorithms will be developed and analyzed:

1. AES algorithm,
2. AES-CCM algorithm.

In this project, simulations by software and hardware implementations of the selected security schemes are proposed, and the design of these architectures is based on task identification, to develop modules, processes and data buses.

The next phase is to design and develop simulations by software. These provide functional simulation and ideas to design hardware architectures. Data buses, functional modules, and complete hardware architectures are simulated, trying to find similar modules and likeness between two proposed security hardware architectures, carrying out the design and development work of the software-radio platform.

## 4.3 Phase III: Software Implementation and Validation

Software simulations of the proposed security schemes were designed and developed based on diagram blocks of the models of the security schemes, considering modular hardware design. For example, Fig 4.3 shows the model for the IEEE 802.16e-2005 security scheme, where these software simulations were used to validate the hardware blocks and data bus. This evaluation checks functional blocks using test vectors, which are provided by the particular standards [LAN/MAN-Standards-Committee, 2005] and [LAN/MAN-Standards-Committee, 2004]. These modules parse and format data blocks, execute AES-CCM algorithm, and control the dataflow. Also, security architecture of the IEEE 802.11i-2004 was modeled and simulated.

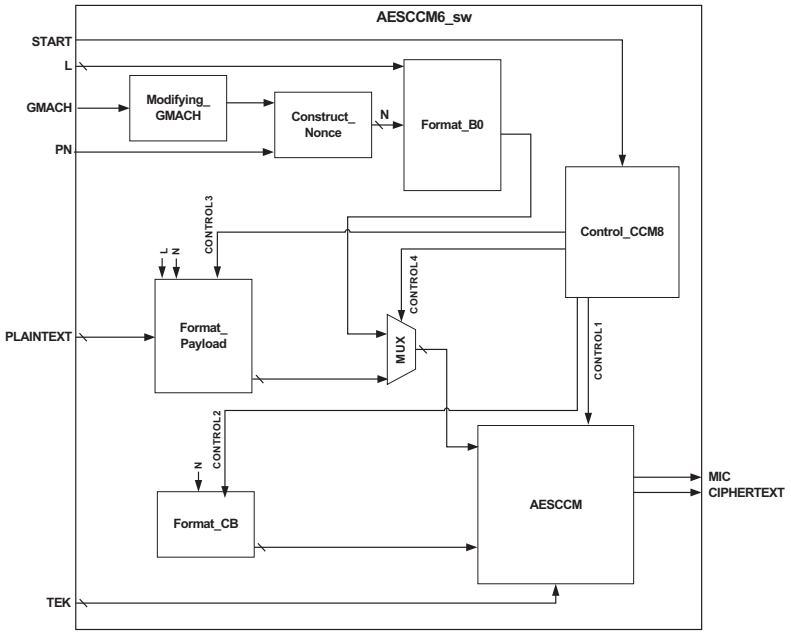


Figure 4.3: Diagram block of the model of the IEEE 802.11e-2005 security scheme

The software simulations of the proposed hardware architectures verify data outputs of each functional module and final values such as MIC and

ciphertext payload. The security schemes are different, detailing individual processes, data formats and cryptographic operations. It is necessary to identify functional elements, allowing the best selection of the reconfiguration scheme. Particular and common functional elements should address several aspects for the problem of designing a reconfigurable platform. These simulation results enable to design the particular hardware architectures by means of a high level programming language, focusing in the high throughput, because parallelization and specialization can be applied. In this way, the design and development of the proposed hardware architectures and hardware design methodologies can be accelerated, considering modular designs and identifying common and particular modules.

## 4.4 Phase IV: Modular Hardware Architectures

Hardware architectures achieve higher performance than software ones, although architectural models should be designed considering hardware benefits. In this phase, the proposed general basic hardware design methodology (each particular hardware architecture uses a modified design methodology) is to develop basic modular hardware architectures, to evaluate and examine characteristics and parameters. These hardware architectures will be analyzed and modified, proposing new efficient hardware architectures. The implementations of the efficient hardware architectures provide initial data such as used hardware resources, critical path time, functional elements, parallelization and specialization of modules, and an idea to design the reconfigurable architecture.

To provide high-performance architectures, analyzing implementation results enable to design and develop the software-radio platform, due to the evaluation of the particular hardware architectures. A particular objective of this project is to implement fast iterative hardware architectures with low FPGA resource requirements. In the next chapters, hardware architectures and implementation results by using design methodologies shows that the developed hardware architectures have advantages compared to related work. The design of these hardware architectures was written in VHDL and simulated using FPGA-Advantage 6.3. The hardware architectures were implemented by using ISE Xilinx tools.

Next, the hardware architectures for the AES and AES-CCM algorithms and their implementations will be described.

### 4.4.1 *AES* and *AESCCM* Hardware Architectures

In this sub-section, the non-pipelined *AES* hardware architecture is described, see Section 4.4.2, which focuses on the design with high hardware implementation efficiency. This architecture is a main part of the AES-CBC-MAC and AES-CTR algorithms for the *AESCCM* architecture, see Section 4.4.3. The AES-CCM algorithm lies at the core of the security schemes used by important communication networks such as IEEE 802.11i and IEEE 802.16e-2005 standards. The algorithm, based on the special operating modes of the Advanced Encryption Standard, provides authentication and ciphering services.

The design methodology is based on simulating AES and AES-CCM algorithm, designing diagram blocks and evaluating test vectors from their specifications, see Fig 4.4. After this simulation, initial hardware architectures are developed, applying parallelization of modules and data buses, modular specialization. Next, trade-off studies on throughput and resources are made, focusing on decreasing critical path to improve the performance and efficiency, developing high-efficiency hardware architectures. Implementation results validating efficient architectures with high throughput are showed in Chapter 6.

### 4.4.2 *AES* Hardware Architecture

The proposed architecture is based on the AES standard algorithm specified in the Federal Information Processing Standards Publication 197 [FIPS-197, 2001] of the National Institute of Standards and Technology. The aim is to implement a fast and simple iterative *AES* architecture with low FPGA resource requirements.

The main modules of the architecture are (see Fig 4.5):

1. *AES_Control*, which outputs control signals and organizes the dataflow,
2. *AES_GenKey*, which outputs the round keys, and
3. *AES_Round*, which ciphers the data.

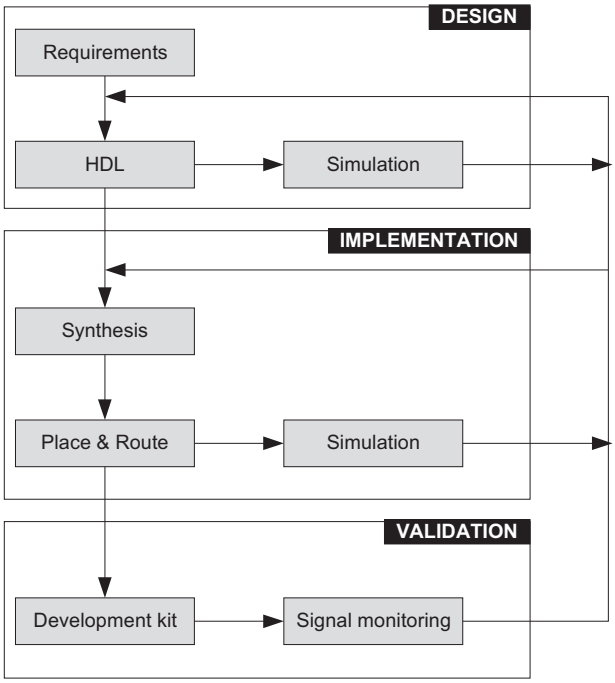


Figure 4.4: Particular design methodology for the hardware architectures

The initial round is computed by the $XOR$ gate, and the following ten rounds are executed by the *AES_Round* module. The round keys are added in *AES_Round* module and the intermediate cipher data are feedback to the same module until the final cipher data are obtained. The selection of the initial round data and the intermediate cipher data is made by the multiplexer. After several clock cycles, the final cipher data are addressed from multiplexer output. *AES_Round* is the main module, it covers the four transformations defined in [FIPS-197, 2001], see Fig 4.6. This module calculates the ten round functions, whereas the initial round operation and the key generation are externally operated.

The general architecture of the basic modular implementation is iterative, and the S-boxes are implemented using twenty memories. *AES_Control* module is a 12-state FSM (Finite State Machine). The state diagram and FSM initial values are shown in Fig 4.7.

If the system ciphers data, and it is maintained in the ROUND0-ROUND10 loop, its output value will offer 128-bit cipher data every ten clock cycles for

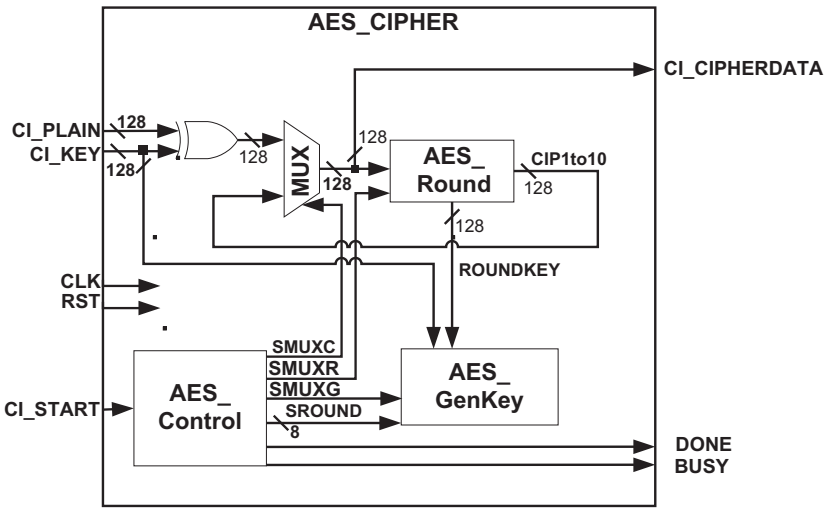


Figure 4.5: Block diagram of the proposed *AES* hardware architecture

128-bit plain data and 128-bit key data. The throughput of the iterative architecture is given by Eq. 2.1.

*AES_GenKey* module is the key-expansion operation, which outputs a 128-bits key every round (see Fig 4.8). Internally, S-boxes and XOR gates compute the round keys; a register stores the round key. All S-boxes are used at the same time (parallel form). In the first clock cycle, the key input is stored, which is used to compute the first round key, and in the next clock cycles, previous round key is feedback to compute the current round key. In the LOAD and ROUND10 states the key input is stored, and from ROUND1 to ROUND10 states, round keys are stored. In the ROUND10 state, the key input is stored because it is used by the ROUND1-ROUND10 loop, when the system ciphers data successively.

The general structure of the *AES_Round* module is shown in the Fig 4.9. This module computes the four transformations defined in [FIPS-197, 2001]. The SubByte transformation is performed by S-boxes implemented in 16 distributed memories. The ShiftRow transformation is made by readdressing the *ByteSub* bus to the $ShiftRow$ bus. This has the effect of cyclically shifting over different number of bytes.

The MixColumn transformation operates $GF(2^8)$ multiplications over $ShiftRow$ bus, and it is performed by the *AES_MixCol* sub-module, see Fig 4.10, which outputs the $MIXColumn$ bus. Finally, in the AddRound-

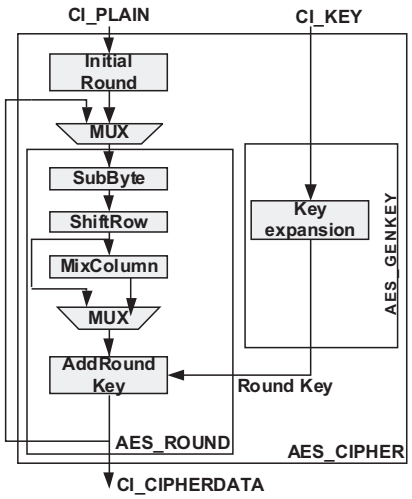


Figure 4.6: The four transformations of the AES algorithm are integrated on the *AES_Round* module of the general architecture

Key transformation, the round key is added by a simple XOR operation. The multiplexer selects in the first nine rounds the $MIXColumn$ bus, whereas in the last round it selects the $ShiftRow$ bus. The multiplexer output is added to the $Ikey$ bus (round key).

The *AES_MixCol* sub-module computes multiplications and additions over $GF(2^8)$. In [FIPS-197, 2001] it is described a matrix multiplication with the fixed polynomial:

$$a(x) = \{3\}x^3 + \{1\}x^2 + \{1\}x + \{2\} \tag{4.1}$$

The equation computes multiplications $\{1\}$ and $\{3\}$, and additions. The $GF(2^8)$ addition is the XOR operation and the $GF(2^8)$ multiplication is special since it is only necessary to multiply by some constants [McCaffrey, 2003]. Constant multiplicands permit to implement XOR operations and multiplexors, and these substitute the multiplication described in [FIPS-197, 2001]. For example, a section of AES_MixCol sub-module is shown in Fig 4.11, where a MixColumn transformation is performed for the $OMIX(127:120)$ byte, or

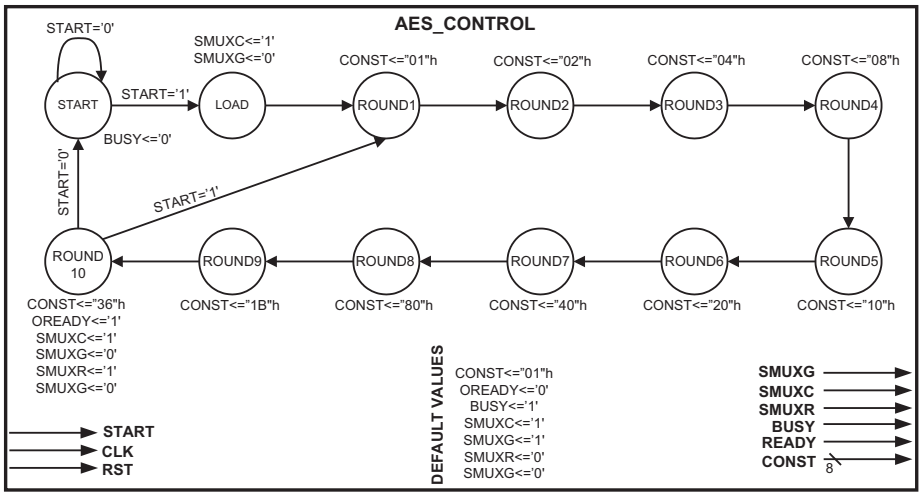


Figure 4.7: State diagram of the *AES_Control* module

$$OMIX(127:120) <= \{2\} * IMIX(127:120) \oplus \\ \oplus \{3\} * IMIX(119:112) \oplus \{1\} * IMIX(111:104) \\ \oplus \{1\} * IMIX(103:96) \tag{4.2}$$

The {1}, {2} and {3} constant coefficients in 3 are multiplied in $GF(2^8)$. In multiplication by {1}, the result is equal to the non-one factor, for example, $IMIX(111:104)$ and $IMIX(103:96)$ bytes are added by the XOR gate.

The multiplication by {2} is a conditional 1-bit left shift implemented by a multiplexer. Its selector, $IMIX(127)$, controls the overflow in $GF(2^8)$. If the value being multiplied is less than "10000000", the result is the value itself left-shifted by 1 bit, $IMIX(126:120)\&'0'$. If the value is greater than or equal to "10000000", the result is the value left-shifted by 1 bit added with "00011011", $IMIX(126:120)\&'0'$ XOR "00011011". This prevents overflowing and keeps the product of multiplication in $GF(2^8)$.

The multiplication by {3} is reduced to additions and multiplications by {2}, where the last multiplications are conditional 1-bit left-shifts [McCaffrey, 2003]. Multiplication by {3} can be decomposed as {3} = {2} + {1}. Thus:

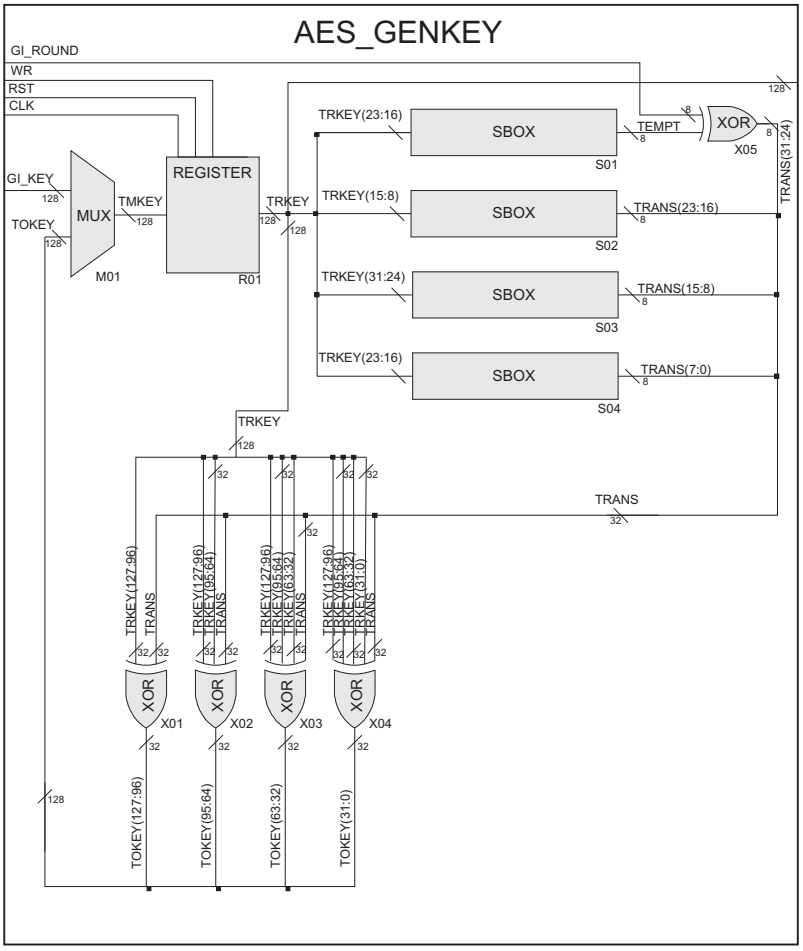


Figure 4.8: Diagram of the $AES\_GenKey$ module

$$\{3\} * IMIX(119:112) <= \{2+1\} * IMIX119:112$$
$$<= \{2\} * IMIX(119:112) + \{1\} * IMIX(119:112)$$

The multiplication by $\{3\}$ is implemented by two multiplexors, three XOR gates, and multiplications by $\{1\}$ and $\{2\}$ implemented as mentioned in the above paragraph.

Initially, parallelization of modules and data buses is used for designing the $AES$ hardware architecture, and after, a trade-off study on throughput/area ratio for decreasing the critical path is made. This path is located

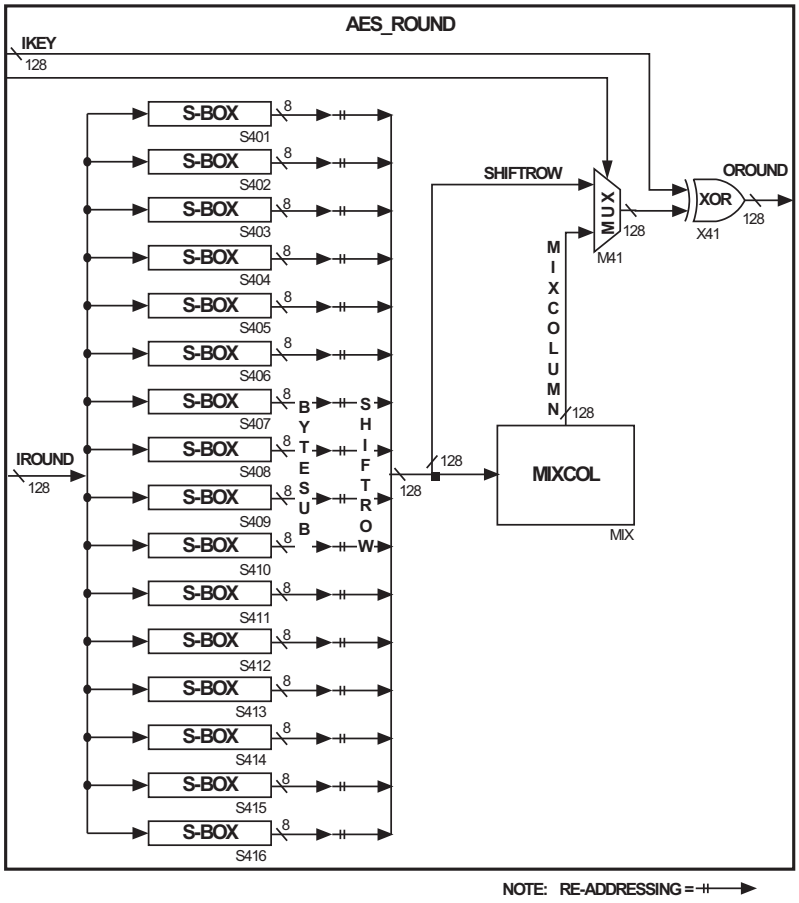


Figure 4.9: Diagram of the *AES_Round* module

in the input of the plaintext and the key. The modification is made by adding two registers, which reduces the critical path. The input data and key are stored in registers to be processed later on in parallel (see Fig 4.12). So, the final architecture multiplexes the $CI\_Plain$ and $CI\_Key$ 128-bit buses, and the $AES\_Control$ enables ciphering without requiring additional clock cycles, since the data are stored in the processing time. In a given clock cycle, a bus is registered, and in the next clock cycle, the other bus. By successively ciphering data, the key and plain data are stored in run time, and each ten clock cycles, an $AO\_CIP$ output or cipher data are obtained.

The implementation of twenty memories for twenty S-boxes requires pro-

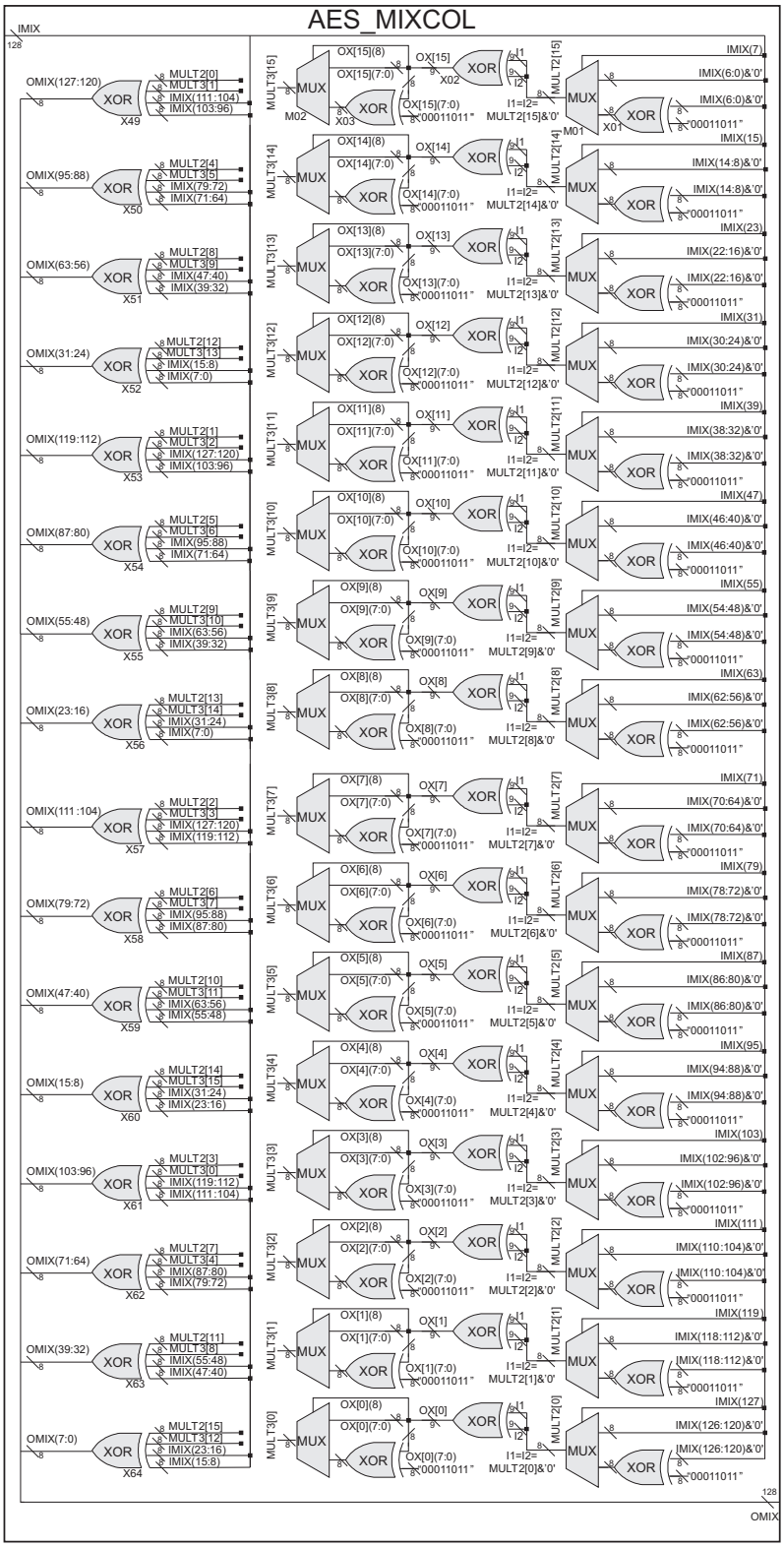


Figure 4.10: Diagram of the *AES_MixCol* sub-module

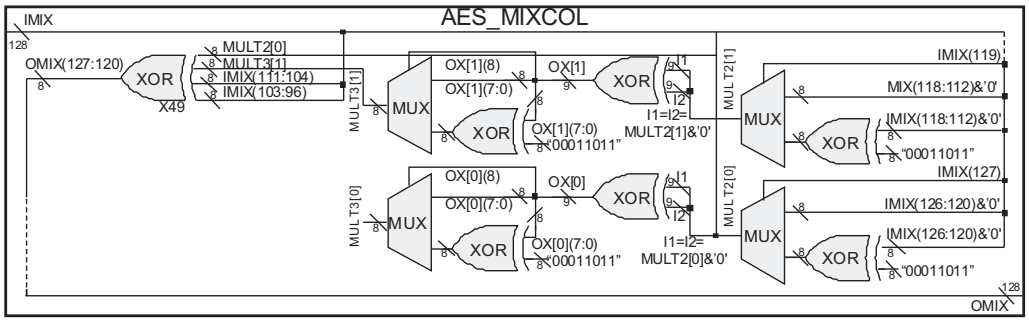


Figure 4.11: Diagram of the operation in Eq. 4.1, which is part of the *AES_MixCol* sub-module

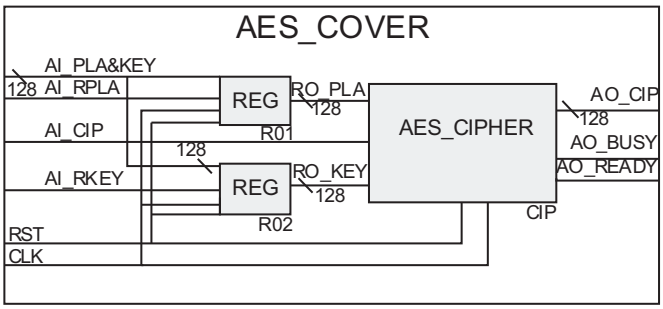


Figure 4.12: Final general *AES* architecture

portional FPGA resources to place and route them, which results in a critical path time proportional to the FPGA utilized logic. Designing this *AES* hardware architecture enables to develop an efficient *AESCCM* hardware architecture, providing design ideas to improve the performance, which are described in the Section 4.4.3.

### 4.4.3 Proposed *AESCCM* Hardware Architecture

The used hardware design methodology is based on designing an initial hardware architecture (considering specific parameters), exploiting hardware advantages such as loop unrolling, pipelining, and using embedded hardware resources [Chaves et al., 2006]. By designing *AES* hardware architecture, see Section 4.4.2, several details are identified, which affect the performance: 1) a straightforward hardware architecture design from an algorithm does not

offer efficient hardware architectures, 2) altering latency can improve the throughput, and 3) parallelization of functional modules does not necessarily improve the throughput. The design methodology followed in this work is to get architectures with higher throughput and lower hardware resources, and an analysis is made to improve the throughput and to reduce hardware resources, proposing the improved hardware architecture.

According to the requirements of the $AESCCM$ architecture (a) to be iterative, fast and simple, and (b) to use low hardware resources, and following the methodology described previously, two architectures are obtained. The first one, $AESCCM$, is based on a straightforward implementation of the architecture shown in the standard, balancing and paralleling to decrease the critical path (increased performance). By making studies on the AES-CCM main process in the initial hardware architecture, the second one, $AESCCMv2$, is an efficient and compact architecture that simplifies a common component used in the two main blocks of $AESCCMv1$, reporting a high throughput/area ratio. The two next sub-sections show the $AESCCM$ hardware architectures. Implementation results are presented and compared to other related works in Chapter 6 to highlight how initial hardware architecture is modified and improved.

#### 4.4.3.1 $AESCCM$ Initial Hardware Architecture

Normally designing an iterative architecture, such as $AES$, considers adding functional elements according to the requirements, for example, a register for storing an output, distributed or embedded memory for storing the values of the S-boxes, multiplexors for selecting internal values or inputs. However, after designing hardware architectures in this way, large critical paths can decrease the performance, requiring applying design techniques to improve it. To design the $AESCCM$ initial hardware architecture, modules and data buses are parallelized.

The development of the initial hardware architecture for the AES-CCM algorithm considers that data blocks, counter, and packet number are constructed by an upper layer, such as in the 802.11i standard. Other consideration is that the M and L CCM parameters, have values of 8 and 2, respectively. The $AESCCM$ hardware architecture is based on the IEEE 802.11i specifications [LAN/MAN-Standards-Committee, 2004], where security operations are defined.

Next, the $AESCCM$ initial hardware architecture and its two main mod-

ules (AES-CBC-MAC and AES-CTR) are detailed. *AES_Cipher* architecture structure uses low FPGA resources and achieves high throughput, useful characteristics for the AES-CBC-MAC and AES-CTR modules. Firstly, main blocks are described because these are key elements of the initial hardware architecture, and because the improved hardware architecture is depicted immediately, which improves implementations results of the first one.

**Module for AES-CBC-MAC**

The *AESCCM_Authenticator* module executes the CBC-MAC process to compute the authentication field $T$. The hardware architecture of this module is mainly constituted of an *AES_Cipher*, see Fig 4.13. This last one has an iterative and non-pipelined architecture. Each 128-bit block is fed to the data input $BX$, *AES_Cipher* processes it during ten clock cycles and outputs 128-bit blocks, $YK$, which are feedback to cipher the next data block. When all 128-bit blocks have been processed, the output T is obtained by selecting eight bytes from the last output $YK$ of the *AES_Cipher*.

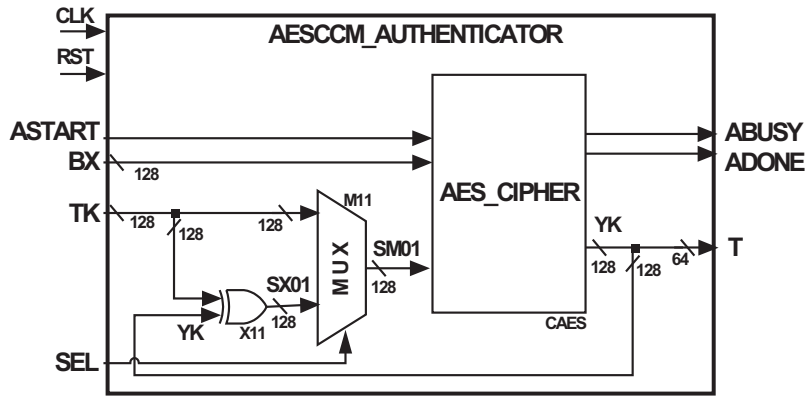


Figure 4.13: Block diagram of the *AESCCM_Authenticator* module

The control signals (*SE*L and *AStart*) organize the dataflow, choosing the appropriate input for *AES_Cipher*, and initializing the main operation. The flag signals (*ABusy* and *ADone*) indicate the status of the *AESCCM_Authenticator* module, processing or valid output $T$.

**Module for AES-CTR**

*AESCCM_Cipher* executes the CTR process to compute the cipherdata *Cipher_MPDU*. Fig 4.14 shows the block diagram of the *AESCCM_Cipher*

hardware architecture, which also uses the $AES\_Cipher$. The CTR process of this module executes AES process for each 128-bit counter value with 128-bit key $TK$. Firstly, for the initial counter value $CB$, the input $T$ together with the first 64 bits of the $AES\_Cipher$ output are used to generate the main output of the $AESCCM\_Cipher$ module, named $U$. After that, the next counter values are processed, and their outputs of the $AES\_Cipher$ and the $BX$ are used to generate the $Cipher\_MPDU$. Registers are added for data synchronization.

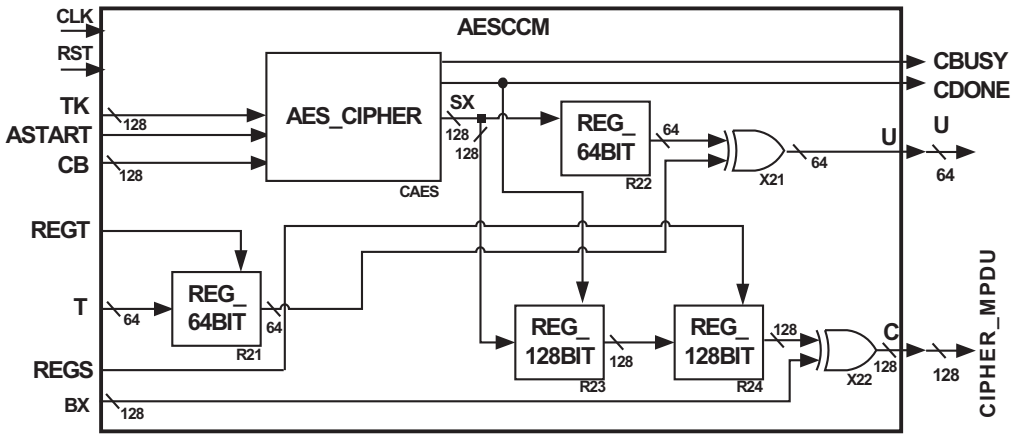


Figure 4.14: Block diagram of the $AESCCM\_Cipher$ module

The control signals ($RegT$ and $RegS$) enable register writing. $CSTART$ signal initiates process in the $AES\_Cipher$. The flag signals ($CBusy$ and $CDone$) indicate the status of the $AESCCM\_Cipher$ module, indicating processing or valid outputs $U$ and $C$.

These two modules, $AESCCM\_Authenticator$ and $AESCCM\_Cipher$, constitute the main two blocks of the $AESCCMv1$ architecture, which serves as a first work platform according to the design methodology mentioned in Section 4.4.1. These two modules and used data buses are parallelized, proposing AESCCMv1 architecture.

**Initial hardware architecture:** $AESCCMv1$

The $AESCCMv1$ architecture is a straightforward implementation of AES-CCM algorithm that besides providing a basic work platform is optimized to achieve better performance. The block diagram of this architecture is shown in Fig 4.15. Further, considering the design methodology, the

throughput is improved by using modular parallelization, defining data buses and designing specialized functional modules.

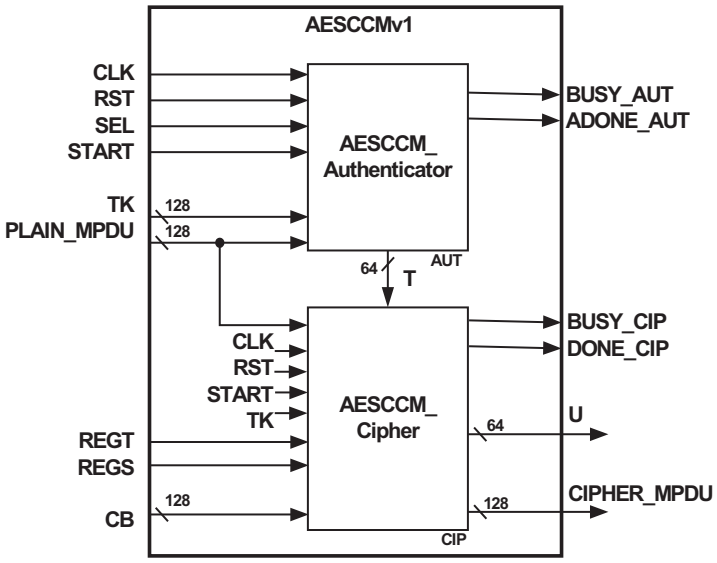


Figure 4.15: Block diagram of the $AESCCMv1$ Architecture

Also, balancing of combinational and sequential elements and altering latency are performed to decrease the critical path. Balancing is based on equilibration of the different paths, by moving sequential elements through combinational elements. Altering of the latency is made by adding or eliminating sequential elements such as registers or the S-boxes implemented in memories. In the next section, the $AESCCM$ hardware architecture is modified to increase the performance.

#### 4.4.3.2 $AESCCM$ Improved Hardware Architecture

The analysis is based on revising the two $AES\_Cipher$ architectures (used in both $AESCCM\_Authenticator$ and $AESCCM\_Cipher$) of $AESCCMv1$, because particular $AES\_Round$ of each module computes different data blocks with a round key, but the input key and generated keys are the same for the two $AES\_Round$.

In the $AESCCMv1$ hardware architecture, $AES\_GenKey$ computes and outputs, with the same result or round keys, at different times, which should be synchronized for the two main modules, $AESCCM\_Authenticator$ and

$AESCCM\_Cipher$. In this way, reduction of used hardware resources is made and critical path is decreased, improving the throughput and efficiency. This idea is accomplished to propose improved $AESCCM$ hardware architecture: $AESCCMv2$.

**Improved hardware architecture:** $AESCCMv2$

The goal of $AESCCMv2$ is to increase throughput/area ratio by analyzing and obtaining a common component, $AES\_GenKey$, and thus simplifying the $AESCCMv1$ hardware architecture to reduce critical path time (improving throughput) of $AESCCMv1$ and to reduce the hardware resources requirement.

The block diagram of the $AESCCMv2$ hardware architecture is shown in Fig 4.16. As it can be seen, the $AES\_GenKey$ module is only one, working for both modules, $AESCCM\_Authenticator\_v2$ and $AESCCM\_Cipher\_v2$. It is worth to mention that these modules are different from the ones in $AESCCMv1$, since they have been modified by extracting from them the $AES\_GenKey$ module. The two main blocks of $AESCCMv2$ have been synchronized, using the control signals $RegT$ and $RegS$, in order to allow $AES\_GenKey$ works properly for both of them.

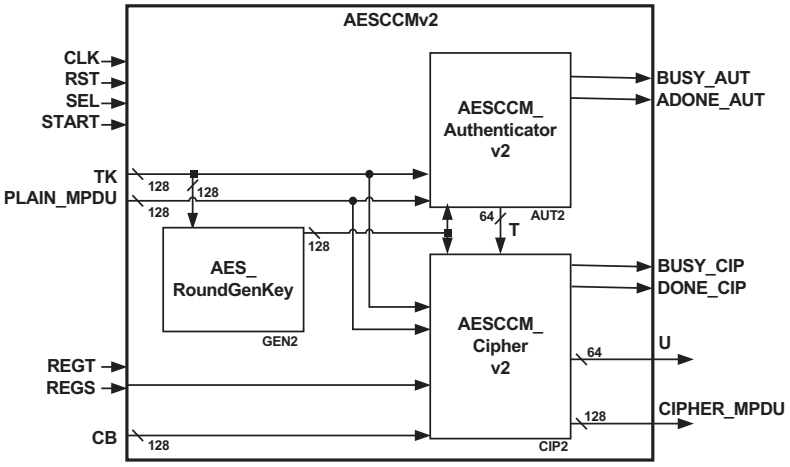


Figure 4.16: Block diagram of the $AESCCMv2$ hardware architecture

# Chapter 5

# Efficient Hardware Architectures

Hardware Architectures for the Security Schemes of the IEEE 802.11i-2004 and IEEE 802.16e-2005 Standards

In this Chapter, custom hardware architectures for the security schemes of the standards are presented: 1) AES-CCM Protocol (AES-CCMP) which is the basis for the security scheme of the IEEE 802.11i standard, see Section 5.1, and 2) scheme based on the AES-CCM algorithm for the IEEE 802.16e-2005 standards, see Section 5.2. Also, AES-CCMP is based on the AES-CCM algorithm that performs the Advanced Encryption Standard in CTR with CBC-MAC mode (CCM mode). These two standards have specified different security mechanisms, using mainly the AES-CCM algorithm to provide better security services, although it is required to execute a great number of operations, several iterations, and multiple processes.

## 5.1 Architecture for the 802.11i-2004 Security Scheme

The security scheme of the IEEE 802.11i standard is based on the AES-CCM Protocol (AES-CCMP), which in turn is based on the AES-CCM algorithm that performs the Advanced Encryption Standard in CCM mode. The IEEE 802.11i standard replaces Wired Equivalent Privacy in the original IEEE 802.11 standard with the AES-CCM. Traditionally, two different cryptographic algorithms are used to provide privacy and authentication,

but AES-CCM algorithm provides these two security services with the same algorithm, using the *AES* block cipher and the same key. The privacy is provided by the AES algorithm in CTR mode, requiring a value that ensures uniqueness. The authentication is performed by the AES algorithm in CBC-MAC mode and additional capabilities; CBC-MAC is an integrity method that ensures that every cipher block depends on every preceding part of the plain text, where ciphering two identical blocks results in different cipher blocks.

The use of cryptographic algorithms in demanding applications that transmit great amounts of data requires computing complex operations, which may result in system bottlenecks. Based on the fact that hardware implementations of cryptographic algorithms usually have better performance than their corresponding software implementations, this project presents a custom hardware architecture for the AES-CCM Protocol. A careful analysis of the algorithm allowed exploiting parallelization of some processes and the design of highly specialized processing modules in order to achieve the highest throughput/area ratio when compared against similar works.

### 5.1.1 Proposed Hardware Architecture

The $AESCCMP$ hardware architecture is illustrated in Fig 5.1. It supports several blocks of the security scheme of the IEEE 802.11i standard, see Fig 5.2. It is assumed that Increment PN and Construct CCMP Header blocks are executed in a processing upper layer. The $AESCCMP$ hardware architecture is constituted by specialized modules to format data ($Format\_N\&Q$, $Format\_AAD$, $Format\_Payload$, and $Format\_CB$), to compute AES-CCM algorithm ($AESCCM$), and main control ($Control\_CCMP$). Each module for formatting data has its own control sub-module. The main control and the control of each sub-module are based on Finite State Machines (FSMs). By distributing some control tasks, the main control module is simplified. Modules $AESCCM\_Authenticator$ and $AESCCM\_Cipher$ compute AES-CBC-MAC and AES-CTR algorithms in parallel.

The general operation consists on processing two types of data, parsed in 128-bit data blocks, and the same 128-bit key block through the $AESCCM$ architecture. The first data block is taken from three different data sources ($PAY\_N\&Q$, $PAY\_AAD$, and $PAY\_PAY$) to compute the MIC value in the $AESCCM\_Authenticator$ module, whereas the second data block is taken from the same sub-module ($Format\_CB$) to compute cipher data in the

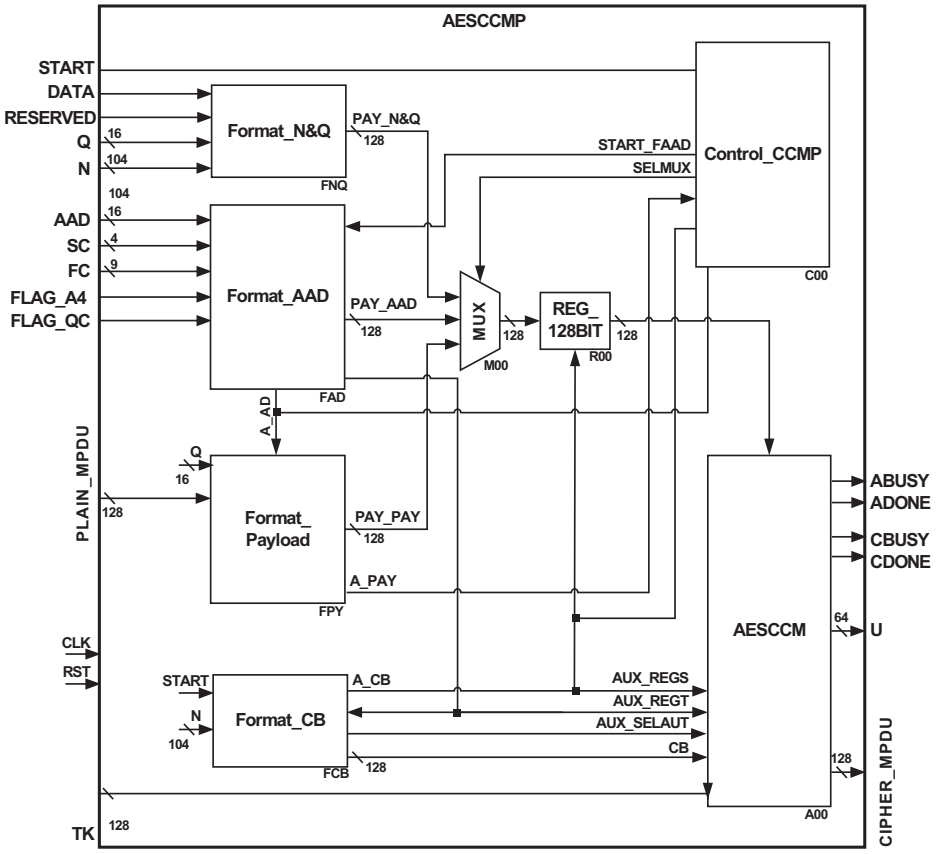


Figure 5.1: Block diagram of the *AESCCMP* architecture

*AESCCM_Cipher* module. After processing all data blocks, the *AESCCM* architecture generates the cipherdata Cipher_MPDU and U value. Next, the specialized modules of the *AESCCMP* hardware architecture are described: *AESCCM* architecture, modules for the construction of data blocks, and the main control module. Implementation results are presented and compared against related works in Chapter 6.

**AESCCM**

The *AESCCM* architecture, see Fig 5.3, has an efficient and compact architecture, see details in Section 4.4.3. According to [Dworkin, 2004], the two CCM parameters (M and L) take values of 8 and 2, respectively. The *AESCCM* architecture has two modules: *AESCCM_Authenticator* and *AESCCM_Cipher*. The general operation of this architecture is divided

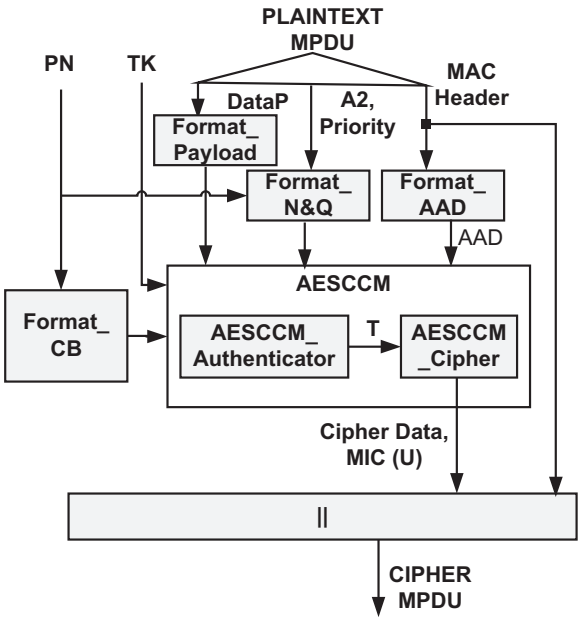


Figure 5.2: Functions supported by the $AESCCMP$ hardware architecture

into two modules, where the first module is required to calculate the authentication field value T, and the second module computes the $cipher\_MPDU$, considering the CCMP encapsulation.

Computing of the $T$ value is done by an iterative non-pipelined $AES$ architecture. Firstly, a sequence of 128-bit blocks is fed to the data input ($BX$), and then each block is processed by the $AES\_Cipher$ during ten clock cycles. Finally, the output $T$ is obtained by selecting eight bytes from the output $YK$ of $AES\_Cipher$. In the first clock cycle, multiplexer component $M01$ selects the input $BX$, considering the control signal $SEL$. If there are not more 128-bit blocks, the output $T$ is obtained from selecting 64 bits of output $YK$, if not, the output $YK$ of the first block and the next block $BX$ are inputs of the $XOR$ gate. Thus, the output $SX01$ is selected from component $M01$, and it is processed by the $AES\_Cipher$. After of processing each block $BX$, value $T$ is finally obtained by selecting 64 bits of the output $YK$ computes a key schedule or a 128-bits key in each round.

To compute the cipherdata $Cipher\_MPDU$, $AESCCM\_Cipher$ module executes AES-CTR process for each 128-bit counter value, and 128-bit data block with 128-bit key ($TK$). The input $T$, and the cipher output $SX$ of the

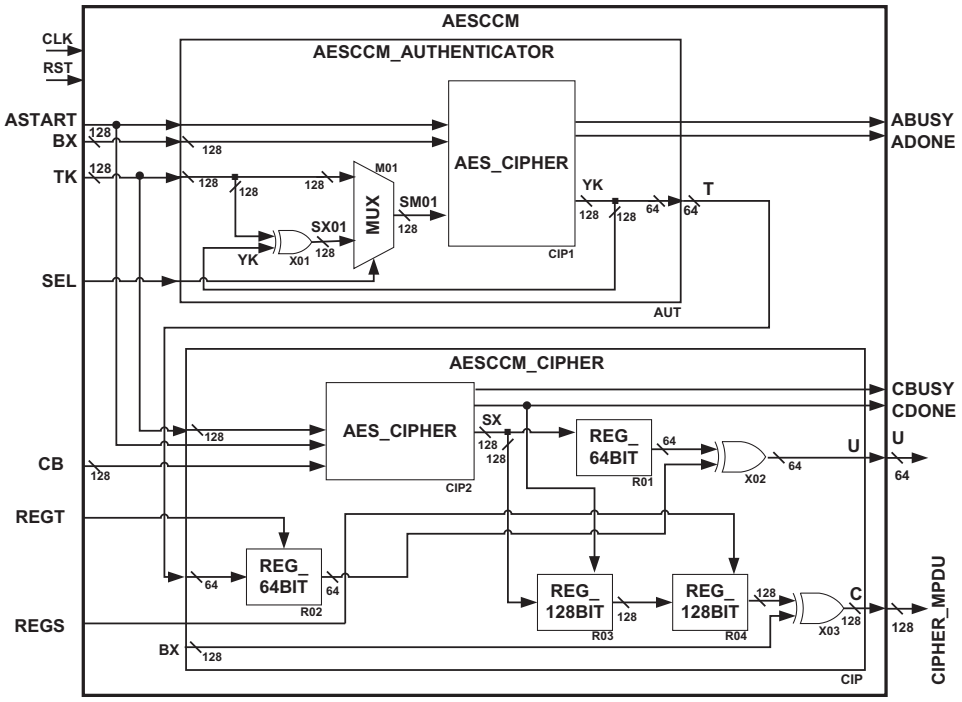


Figure 5.3: Block diagram of the $AESCCM$ used in the $AESCCMP$ architecture

first block $BX$ are inputs for gate $X02$, and the output of this gate is the main output of the $AESCCM\_Ciphermodule$ ($U$). The next counter values are processed, and their outputs of the $AES\_Cipher$ and the data block ($CB$) are inputs of the XOR gate $X03$. The output of this gate is the other main output of the $AESCCM\_Cipher$ ($Cipher\_MPDU$). Registers are added for synchronizing the data. Both, the $AESCCM\_Authenticator$ and the $AESCCM\_Cipher$ modules have a common component named $AES\_Cipher$, which computes the AES algorithm. All processing used within CCM uses AES with 128-bit key and 128-bit block size, more details of this $AES$ hardware architecture in Section 4.4.2.

**Data-input Formatting**

In general terms, for the CBC-MAC process, IEEE 802.11i specifies that AES-CCMP increments PN, obtaining a fresh PN for each MPDU, and the CCM initial block is constructed from this PN, from the MPDU data length (Q), and from other defined bits. The next two data blocks are formatted and

constructed from AAD, whereas the remainder data blocks are constructed from the payload. For this last process, a counter is initialized, and with the nonce, the counter blocks (CBs) are constructed. These, the payload, and the value T are the inputs for the CTR process that obtains the ciphered data and the final value MIC (U).

The $AESCCMP$ architecture constructs data blocks through four specialized modules:

i) $Format\_N\&Q$, which generates the initial data block (Fig 5.4),

ii) $Format\_AAD$, which formats the AAD (Fig 5.5),

iii) $Format\_Payload$, which constructs data blocks from payload (Fig 5.6), and finally,

iv) $Format\_CB$, which constructs counter blocks from the counter and nonce for the CTR process (Fig 5.7).

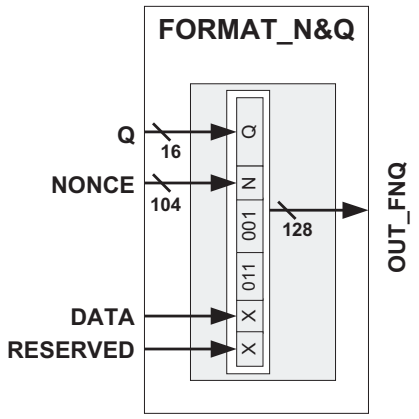


Figure 5.4: Block diagram of the $Format\_N\&Q$ module

$AESCCMP$ architecture executes the CBC-MAC and CTR processes in parallel form, resulting in less hardware resources. $AESCCM\_Authenticator$ takes data input from three different sources, formatting and multiplexing to calculate the value $T$. $Format\_N\&Q$ generates the initial data block that is the input for the $AESCCM\_Authenticator$, and is processed during ten clock cycles. The next data blocks are obtained from the $Format\_AAD$ module. These data blocks are formed by two 128-bit data blocks from ADD. In this module, the key element is the control that enables data formatting from a variable source. For this, it is important to consider that ten clock cycles are used for processing an initial data block; therefore, the input bus for AAD

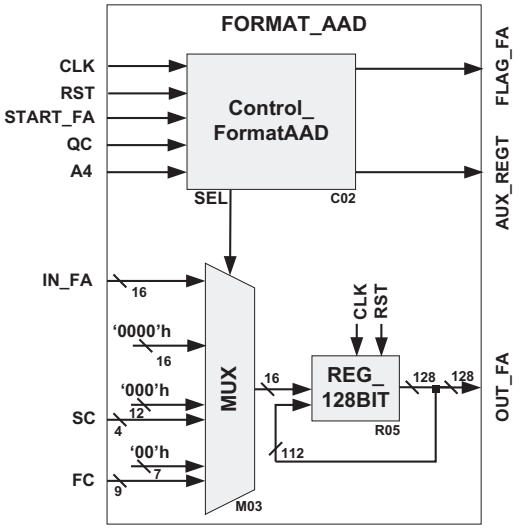


Figure 5.5: Block diagram of the *Format_AAD* module

is selected with a 16-bit size. This decision considers the data rate in the other modules, such as *AESCCM*, enabling to use less hardware resources and forming 128-bit data blocks.

The variable source is due to the AAD is constituted by several fields with length in octets, see Fig 5.8. The construction of the data block *PAY_AAD*, see Fig 5.5, requires to modify certain fields (FC and SC) of the data input *AAD*, and other fields (A4 and QC) can be in attendance or not. This generates a variable size of the AAD, hence two 128-bit data blocks including padding bits are required.

Some bits of the FC and SC fields are modified, and A4 and QC fields are optional. This produces the variable length of the AAD, presenting the particular control unit a complex operation based on a FSM, see Fig 5.9. This control considers four possible inputs of *AESCCM_Authenticator*, see Fig 5.5: i) modified FC, ii) modified SC, iii) zeros (padding bits), and iv) the AAD (*IN_FA* input). Moreover, *Format_AAD* considers inclusion or not of the A4 and/or QC fields, according to signals *Flag_A4* and *Flag_QC*, respectively, which indicates the presence of these fields. For example, if A4 and QC are present, then AAD has a length of 240 bits, see Table 5.1.

*Format_Payload* module constructs 128-bit data blocks from the payload. The data blocks generated by *Format_Payload* are processed during ten clock cycles in *AESCCM* architecture, enabling a 16-bit bus for produc-

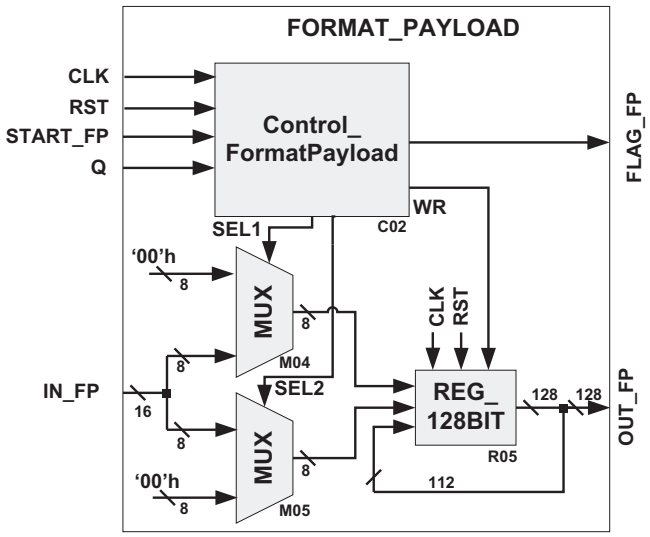


Figure 5.6: Block diagram of the *Format_Payload* module

Table 5.1: Possible values of the AAD

| Condition | AAD (bits) | Padding Bits |
|---|---|---|
| A4 & QC ($F\_A4$=1, $F\_QC$=1) | 240 | 16 |
| A4 ($F\_A4$=1, $F\_QC$=0) | 224 | 32 |
| QC ($F\_A4$=0. $F\_QC$=1) | 192 | 64 |
| Neither ($F\_A4$=0, $F\_QC$=0) | 176 | 80 |

ing and formatting a 128-bit data block. The length Q has a variable value, and it indicates the number of 128-bit data blocks. The padding bits are set to zeros. The 16-bit input $In\_FA$ is divided into two bytes, see Fig 5.6. The particular control unit $Control\_FormatPayload$, see Fig 5.10, selects these data input for constructing of the data blocks.

Finally, $AESCCM\_Cipher$ module takes data input from $Format\_CB$ module, initializing the counter $Counter\_16Bit$, and using the nonce see Fig 5.7. The counter blocks are generated, and these, together with the value $T$ and the formatted payload, are data inputs for $AESCCM\_Cipher$. They are ciphered, obtaining the *cipherdata* and the final value MIC ($U$). The associated control unit is simple, see Fig 5.11. It is a FSM with four states,

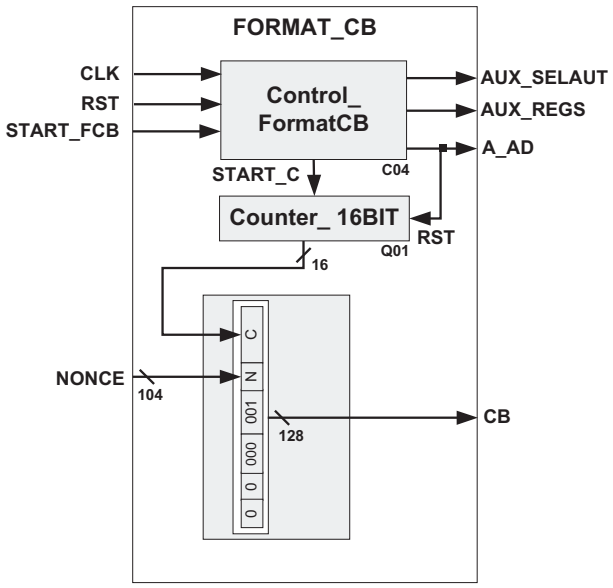


Figure 5.7: Block diagram of the $Format\_CB$ module

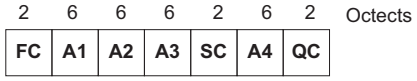


Figure 5.8: AAD Construction [LAN/MAN-Standards-Committee, 2004]

which counts ten clock cycles to indicate that $Format\_CB$ has a valid output.

### Main Control Module

$AESCCMP$ architecture has specialized modules with their own control units, with two processes ($AESCCM\_Authenticator$ and $AESCCM\_Cipher$) executed in parallel. These processes use a common component to generate keys for the AES algorithm. The $Control\_CCMP$ module allows controlling this dataflow, by managing the parallelization and synchronization of the processes that execute both, the CTR and CBC-MAC processes. The control is based on a FSM, see Fig 5.12.

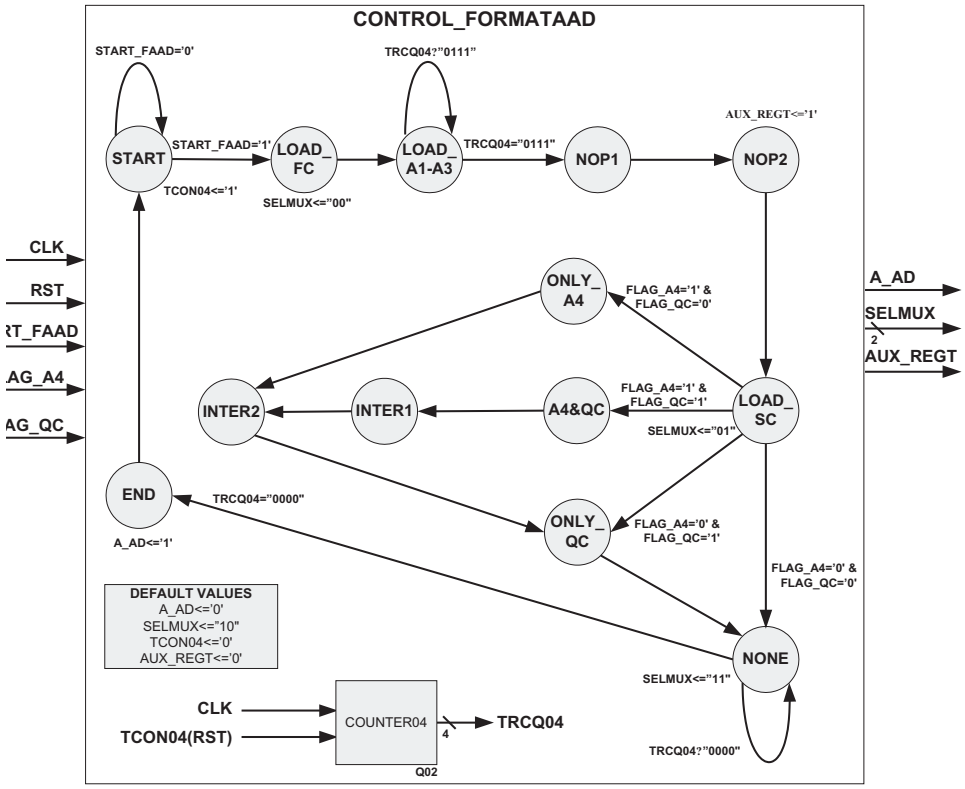


Figure 5.9: Finite State Machine used for the *Control_FormatAAD* control unit

## 5.2 Architecture for the 802.16e-2005 Security Scheme

For the IEEE 802.16e-2005 security standard, a hardware architecture based on the AES-CCM algorithm is proposed. The proposed hardware architecture, named $AESCCM6$, is illustrated in Fig 5.13. It supports several blocks of the security scheme of the IEEE 802.16e-2005 standard, see Fig 5.14. This is based on the security scheme of this standard. This architecture is constituted by specialized modules to format data ($Modifying\_GHMAC$, $Construct\_Nonce$, $Format\_Payload$, $Format\_B0$, and $Format\_CB$), to compute AES-CCM algorithm ($AESCCM$ architecture), more details in Section 2.5.4. The dataflow is managed by the main control. $Format\_Payload$ executes a complex process due to the variable length L6 of the plain-

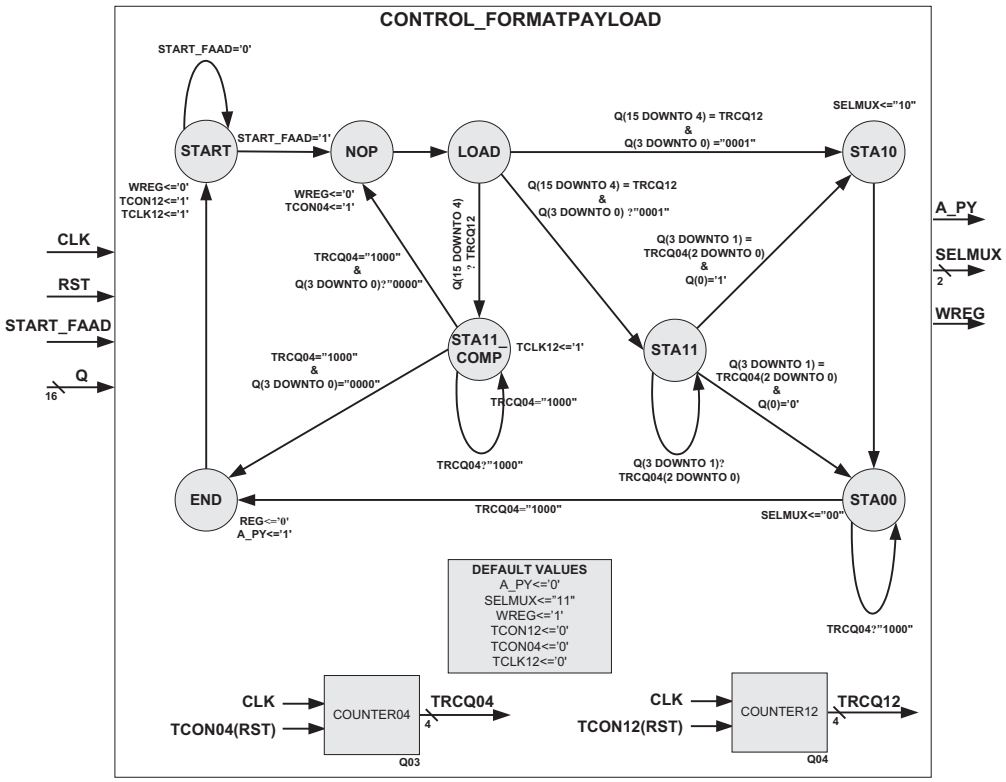


Figure 5.10: Finite State Machine used for the *Control_FormatPayload* control unit

text payload, so, this module has a particular control sub-module. The main control and the particular control sub-modules are based on Finite State Machines (FSMs). Considering and using the states of the particular control sub-module, the main control module is simplified, generating flag and control signals to the dataflow. *AESCCM* architecture is based on the *AESCCM_Authenticator* and *AESCCM_Cipher* modules, working in parallel, which compute AES-CBC-MAC and AES-CTR algorithms, respectively.

The general operation consists on processing two types of data, parsed in 128-bit data blocks, and the same 128-bit key block through *AESCCM* architecture. The first data block is taken from two different data sources (*Format_Payload* and *Format_B*0 modules) to compute the MIC value in the *AESCCM_Authenticator* module, whereas the second block is taken

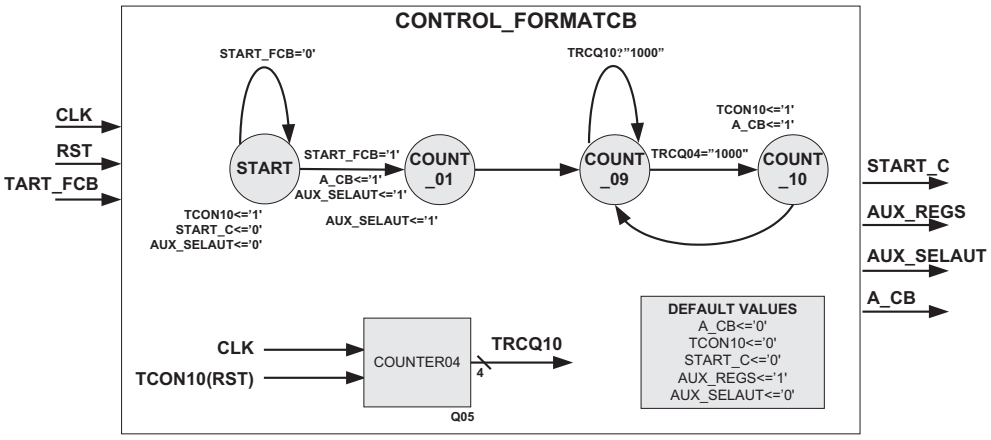


Figure 5.11: Finite State Machine used for the *Control_FormatCB* control unit

from the *Format_CB* module to compute ciphertext in the *AESCCM_Cipher* module. After processing all data blocks, *AESCCM* generates the *Ciphertext* and *MIC* value.

Next, the specialized modules of the *AESCCM*6 hardware architecture are described: *AESCCM* architecture, modules for the construction of data blocks, and the main control module. Implementation results are presented and compared against related works in Chapter 6.

*AESCCM*

The design of the *AESCCM* architecture is based on a straightforward architecture, which is balanced and parallelized to decrease the critical path (increased performance). An additional analysis is made to decrease both the critical path and used hardware resources, improving efficiency hardware implementation. The *AESCCM* architecture, see Fig 5.15, has an efficient and compact architecture that simplifies a common component used in the two main blocks of *AESCCM*, presenting a high throughput/area ratio.

AES-CCM uses AES-CBC-MAC in conjunction with AES-CTR to produce a MIC (Message Integrity Code) for authentication purposes, linking together encryption and authentication under a single key. According to [Dworkin, 2004], the two CCM parameters (length of M and L) take values of 8 and 2 bytes, respectively. The *AESCCM* architecture has two modules: *AESCCM_Authenticator* and *AESCCM_Cipher*. The general operation of this architecture is based on the modules, where the first module is required

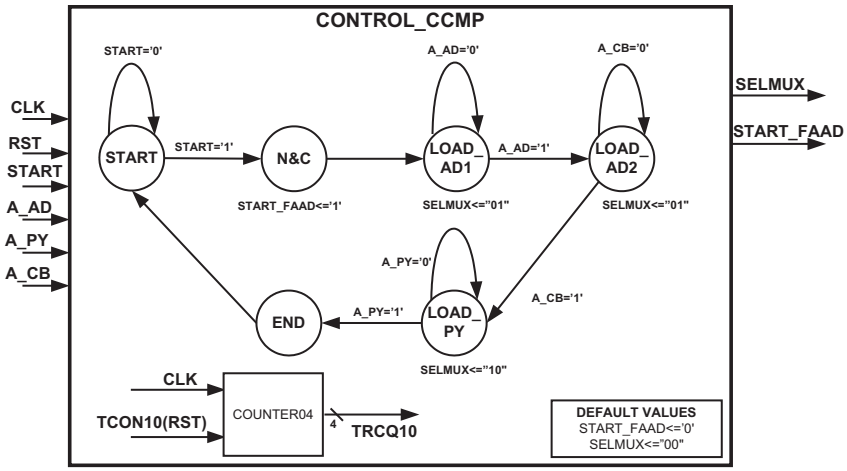


Figure 5.12: Finite State Machine used in the $Control\_CCMP$ module

to calculate the authentication field value MIC, and the second module computes the output $CiphertextPayload$, considering the encapsulation defined in IEEE 802.16e-2005.

Computing of the MIC value is done by an iterative non-pipelined $AES$ architecture. Firstly, a sequence of 128-bit blocks is fed to the data input ($B$), and then each block is processed by the $AES\_Cipher$ during ten clock cycles. After processing all data blocks, the output $MIC$ is obtained by selecting eight bytes from the output $YK$ of $AES\_Cipher$, and by executing XOR operation with $T$ 64-bit bus, value from $AESCCM\_Cipher$. In the first clock cycle, multiplexer component selects the input $B$, and after the feedback input is taken into the $AES\_Cipher$.

To compute the *ciphertextpayload*, $AESCCM\_Cipher$ module executes AES-CTR process for each 128-bit counter value, and 128-bit data block with 128-bit key ($TEK$). The output $T$ is generated by 64 bits, which are the most significant bits of the cipherdata from the first counter blocks, when the counter is 0. After, the next counter values are processed by the $AES\_Cipher$, and their cipher outputs are computed using XOR operation with data block $B$ to generate the $CipherdataPayload$. This output is the other main output of the $AESCCM\_Cipher$. Registers are added for synchronizing the data, because it is necessary to store data blocks of the $T$, which is computed by the cipherdata of the first value $CB$, whereas $B$ is stored to compute the Ciphertext Payload, because $B$ changes its value after

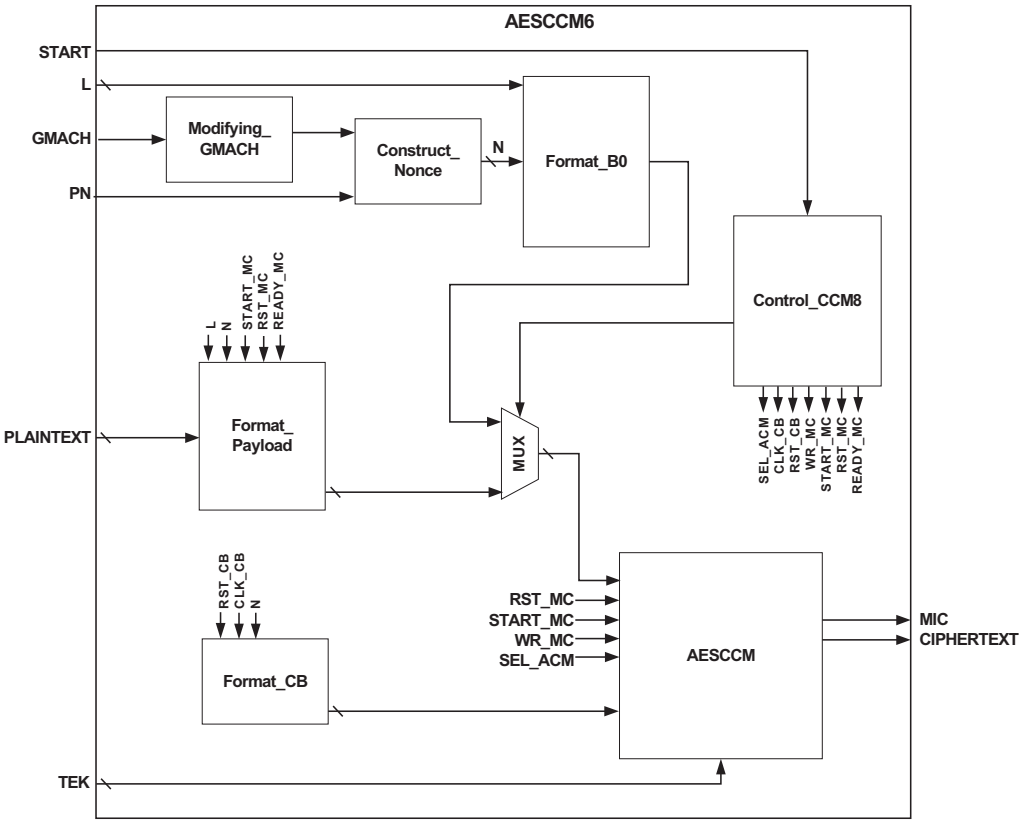


Figure 5.13: Block diagram of the $AESCCM6$ architecture

ten clock cycles.

Both, the Authenticator and the Cipher modules have a common component named *AES_Cipher*, which computes the AES algorithm. All processing used within CCM uses AES with 128-bit key and 128-bit block size. More details about the *AES_Cipher* hardware architecture are given in Section 4.4.2.

**Data-Input Formatting**

In general terms for the CCM process, IEEE 802.16e-2005 specifies that certain values should be rearranged, such as GHMAC and PN, see Fig 5.16. So, construction of the nonce uses the modified values of the PN and GHMAC, see Fig 5.17. The nonce is used to compute the CBC-MAC and CTR processes, because from it is formed the block initial and the counter blocks.

The initial block $B0$ is constructed from the nonce, see Fig 5.18, and

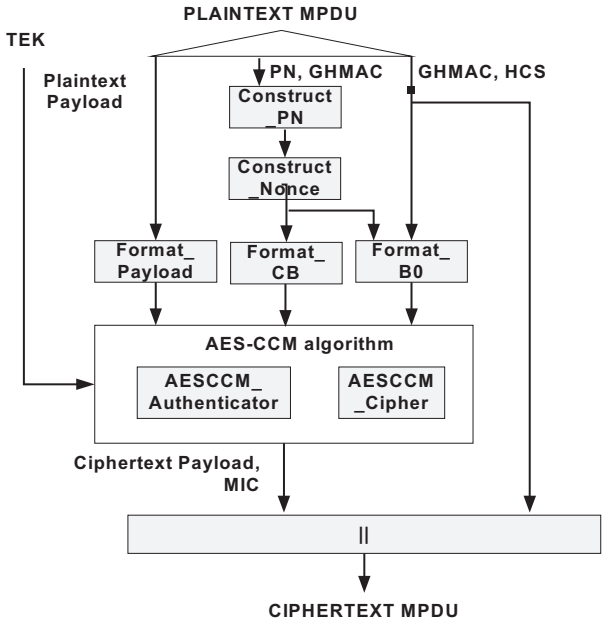


Figure 5.14: Functions supported by the hardware architecture of the security scheme based AES-CCM

the length in bytes of the plaintext payload, and from other defined fixed bits, see Section 2.5.4. $B0$ is a data source together with the obtained data blocks of plaintext payload to input to the CBC-MAC process.

The design of the $Format\_Payload$ module takes input from a 128-bit bus, containing plaintext payload, which has a variable length L in bytes. Only 128-bit data blocks should be generated, and if it is necessary addition of more bits, these will be set to 0. The block diagram of this module is shown in Fig 5.19. This module is the second source of the AES-CBC-MAC process, and constructs 128-bit data blocks from plaintext payload. An important consideration of design of this module is that the data block generated by $Format\_B0$ is processed during ten clock cycles by the $AESCCM$ architecture, enabling to evaluate two bytes by clock cycles for producing and formatting a 128-bit data block.

An important consideration is that plaintext payload has a minimal length L equal to 1 byte. At the level of bytes, and due to the possible filling of $0s$, two different cases are identified: i) L $\leq$ 15 and ii) L $>$ 15. In the first case, 128-bit data blocks are completely taken from plaintext payload, whereas in

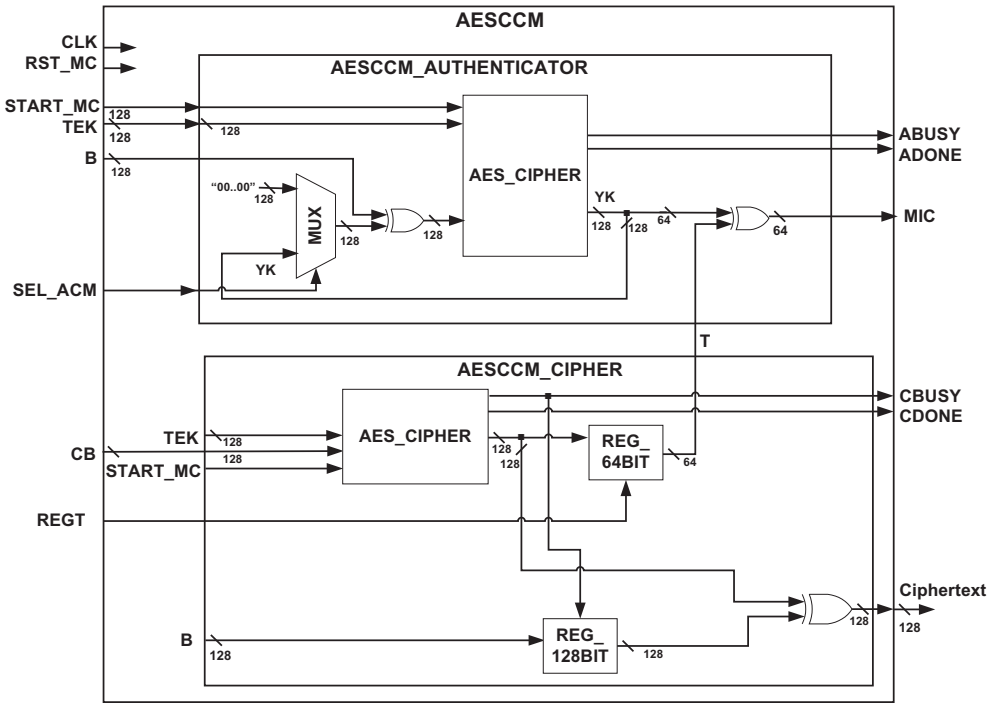


Figure 5.15: Block diagram of the *AESCCM* architecture used in *AESCCM*6

the second case, it is possible that some bytes should be set to 0. The value of each byte is managed by the multiplexors, and the input of these multiplexors is obtained from a circular register, which stores the 128-bit data block. The circular register is for putting each byte of the data block into the multiplexor. This complex operation was separately designed for main control, and each multiplexor is controlled by the particular $Control\_FormatPayload$ sub-module, see Fig 5.20.

The counters (Counter12 and Counter04) are important, because the first one is used to count the complete 128-bit data blocks, whereas the second one is used to count the 128 bits of each block. This particular control considers three cases based on the variable length of the payload, see Table 5.2. The case $A$ considers that all bytes of the input $PlaintextPayload$ are valid, so, they are selected by the multiplexors and stored in the final register. Considering the state diagram, the case $A$ takes its path on the states: $pre1\_Caaioa$, $pre2\_Caaioa$, $pre3\_Caaioa$ and $Caaioa$. The cases $B$

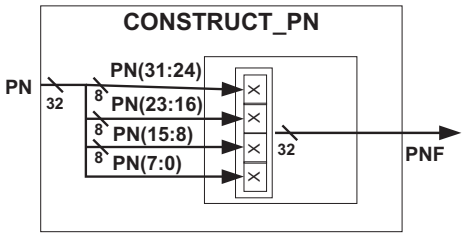


Figure 5.16: Block diagram of the *Construct_PN* module

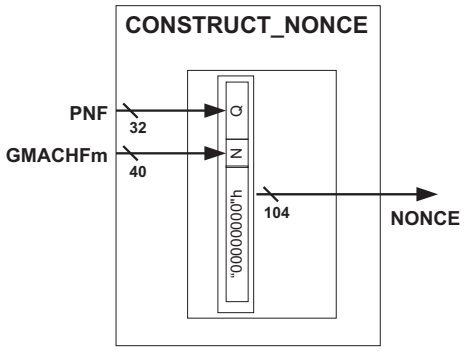


Figure 5.17: Block diagram of the *Construct_Nonce* module

and $C$ consider that some bytes of the input $PlaintextPayload$ are valid, and the remaining bytes should be set to 0. The case $B$ indicates that one multiplexor, in a certain moment, should select one byte from the input and the other multiplexor should set $0s$, whereas the case $C$, in a certain moment, indicates that the two multiplexors outputs $0s$. These last two cases fill of $0s$ the remaining bytes of the register. Considering the state diagram, the case $B$ and $C$ take a similar path, where the state $NO\_0s$ is the difference.

Table 5.2: Selection of the multiplexors of the *Format_Payload*

| Case | Length of Payload (bytes) | Selection |
|---|---|---|
| $A$ | L > 15 | Mux1=Mux2='1' |
| $B$ | L $\leq$ 15 and L is odd | Mux1='1',Mux2='0' |
| $C$ | L $\leq$ 15 and L is even | Mux1=Mux2='0' |

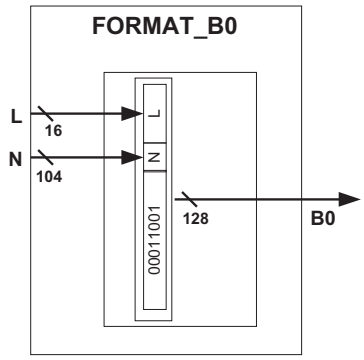


Figure 5.18: Block diagram of the $Format\_B0$ module

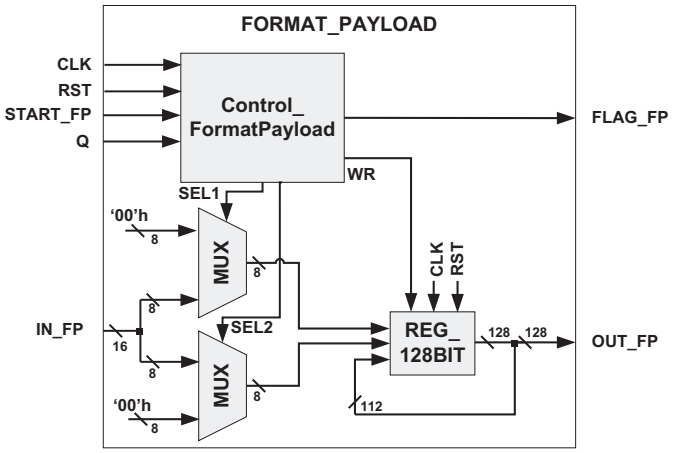


Figure 5.19: Block diagram of the $Format\_Payload$ module

Finally, $AESCCM\_Cipher$ module takes data input from $Format\_CB$ module, initializing the counter $Counter\_16Bit$, and using the nonce. The counter blocks are generated, and these, together with the value $T$ and the formatted payload, are data inputs for $AESCCM\_Cipher$. They are ciphered, obtaining the cipherdata and the final value $MIC$. For this last process, the nonce is used, some bits have a fixed value, and a counter is initialized, see Fig 5.21. This counter is incremented according to the number of 128-bit blocks (built from plaintext payload). The generated $CBs$ are used to input to the CTR process. The control signals are utilized to increment or to reset the counter values. These values should be synchronized to execute additions with 128-bit plaintext payload blocks, generating the

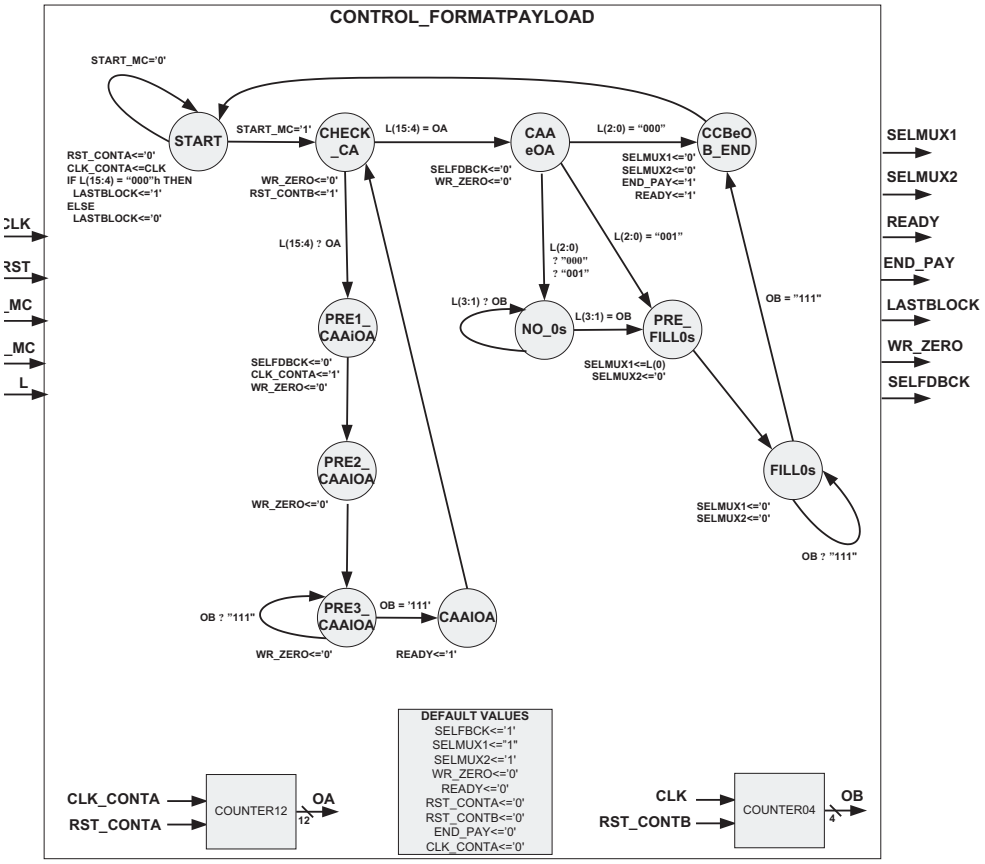


Figure 5.20: State diagram of the *Control_FormatPayload* sub-module

*ciphertextpayload*.

The general structure of the *AESCCM* architecture is iterative, and the S-boxes are implemented using twenty embedded RAM memories, because two *AES_Cipher* architectures are used. This enables to use them at the same time (parallel form).

**Main Control Module**

*AESCCM*6 architecture has specialized modules, generating 128-bit data blocks, and the *MainControl* module produces the control signals. The two processes, *AESCCM_Authenticator* and *AESCCM_Cipher*, are executed in parallel form. To handle this dataflow, and to manage the parallelization and synchronization of the processes executing both CTR and CBC-MAC, the *MainControl* module is developed, which is based on a FSM, see Fig

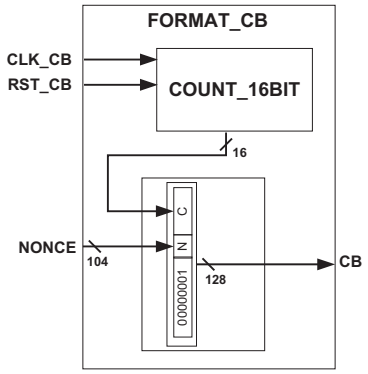


Figure 5.21: Block diagram of the $Format\_CB$ module

5.22.

Each module was independently designed and developed, and connected to the $AESCCM6$ Architecture. Diverse control signals are required, and these are analyzed, checking when they should be enabled. At this point, states of the previous control module is analyzed, which allow save states in the main control. In this way, $Control\_FormatPayload$ particular control is taken into account, using its states and transitions to set some control signals of the $AESCCM6$ architecture. This new scheme facilitates a simple main control unit.

The operation of the $MainControl$ module is determined by processing the initial block $B0$ and the first $CB$ in parallel, which are computed in ten clock cycles. After, it is necessary to process the remaining data blocks Bx and $CBs$, excepting the last block $Bx$ if it should be filled of $0s$.

The hardware architectures are developed and based on modular design, considering parallelization of the data buses and operational blocks to process several data buses at the same time. Also, each block has a specialized structure, executing only the necessary operations. Finally, sequential elements are minimized, obtaining a smaller latency.

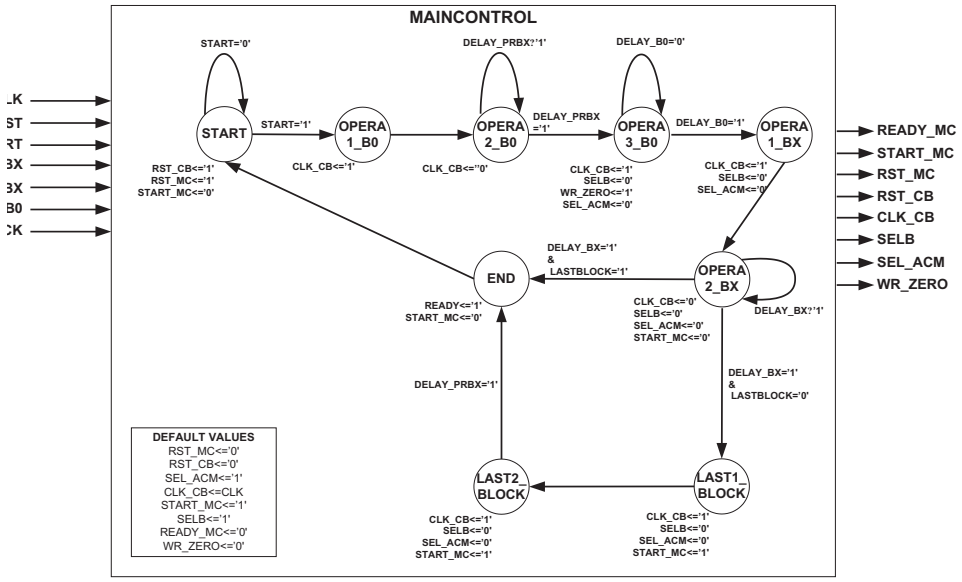


Figure 5.22: State diagram of the *MainControl* module

# Chapter 6

# Implementation Results

In this Chapter, the proposed non-pipelined hardware architectures with low resource requirements are implemented and described. The result implementations are compared against related works of software and hardware implementations.

## 6.1 *AES* Implementation

The implementation results of the *AES* architecture, see Fig. 4.12, are shown in Table 6.1, and these are taken from the post-Place & Route reports. Implementation results show that changing the twenty distributed memories by ten dual-port memories decreases the critical path time from 11.50 ns to 8.80 ns, reduces the FPGA resources, and eliminates some intermediate registers. Also, the implementation results indicate that, in terms of required FPGA resources, the S-box substitution is the dominant element of the *AES* implementation.

Table 6.1: Implementation results of the AES algorithm, which has a non-pipelined iterative architecture

| Period (ns) | Clock (MHz) | IOBs | Slices | LUTs | Clock cycles | Throughput (Gbps) |
|---|---|---|---|---|---|---|
| 8.808 | 96.42 | 263 | 586 | 847 | 10 | 1.452 |

These results show less wired and logic FPGA resources, which are 586 slices and 847 LUTs. This leads to a compact architecture with a lower critical path time, providing a higher clock frequency (96.42 MHz). This *AES* hardware architecture can cipher data at transmission speeds of 1.45 Gbps, due to the Eq. 2.1, where *Plain_data_block_size* has a constant value of 128 bits, and *Clock_cycles* in this design has a value of 10 and *Clock_period* of (96.42 MHz)-1. Consequently, a reduction of *Clock_cycles* implies an architecture with unrolled rounds, which increases the use of FPGA resources and the critical path time or decrease the clock frequency. A reduction of critical path time, by the modification of *Clock_period* is the more practical option.

Recently, several algorithm-specific hardware architectures of AES algorithm have been reported in the literature, involving commercial and research works, with different design techniques, architectures and FPGA resources. Among these are iterative architectures on Virtex-II, and Virtex-4 suitable for CBC mode implementation, and a pipelined architecture on Virtex-II. Different device families will yield different performance results. Important measurements of hardware *AES* implementations to consider are FPGA utilized resources, clock frequency, latency and throughput, see Table 6.2.

The commercial implementation in [Barco-Silex, 2005] has an iterative architecture, 128-bit data input, data output and key input buses. The datasheet presents FPGA implementations, where the "Fast version" has a throughput of 0.58 Gbps. The work in [Liu et al., 2005] presents a partitioning architecture without using the BRAMs (Blocks RAM). The architecture is designed in two parts: 1) implementation of the Key Expansion, which calculates the round keys, and 2) implementation of the functional rounds to cipher 128-bit data. The implementation results of the second part show a throughput of 0.20 Gbps. In [Lu and Lockwood, 2005], AES implementation synthesis results are reported with three different key lengths, and the best throughput is 1.19 Gbps with 128-bit data buses. [Algotronix-Ltd., 2004] describes an AES commercial product, which offers diverse operation modes and key lengths. The architecture uses 4 BRAMs in CBC mode to cipher data; the implementation needs 44 clock cycles at 93 MHz, performing at 0.27 Gbps. The next two works are included for comparing their throughputs and FPGA resource utilization. The work in [Hodjat and Verbauwhede, 2004] reports four AES pipelined architectures, where two of them use BRAMs. The 7-stage AES architecture shows the highest throughput of 21.64 Gbps, at the expense of FPGA resources. [Limited, 2004] presents commercial im-

Table 6.2: Result comparison of the *AES* hardware implementations

| Work-device | FPGA resources | Through-put (Gbps) | Efficiency (Gbps/Slices x$10^{-3}$) |
|---|---|---|---|
| [Liu et al., 2005] XCV200E-6 | 425 CLB | 0.205 | - |
| [Algotronix-Ltd., 2004] XCV250-5 | 791 4 BRAM | - | - |
| [Barco-Silex, 2005] XCV400E-8 | 1672 LUT 175 FF | 0.584 | - |
| [Lu and Lockwood, 2005] Virtex-II Pro | 2703 LUT 44 BRAM | 1.197 | - |
| [Limited, 2004] Virtex-II | 1125 Slices 18 BRAM | 1.400 | 1.244 |
| This work-XC2V1000 | 586 Slices, 847 LUT 10 BRAM | 1.452 | 2.477 |

plementations on the Xilinx Virtex-II FPGA, the main characteristics are a throughput of about 1.40 Gpbs, using 18 BRAMs and 1,125 slices.

The previous works demonstrate that implementing S-box on internal memory improves the throughput, decreases the used FPGA resources, and reduces the critical path time. Current architectures with greater throughput use pipelined structures are mentioned only as a reference, because they cannot be applied to the CBC-MAC mode. This project reports an *AES* architecture with the excellent performance and low resource requirements. The general methodology design used in this work aimed to obtain an iterative architecture with low hardware resources utilization. The modular design was optimized in a way that the algorithm functionality was not altered (e.g. eliminating basic components like registers or multiplexors). Distributed memories were replaced by dual-port memories to handle data in parallel and registers were added for data multiplexing and key storage in order to reduce the critical path, resulting in less hardware that in turn results in a more efficient Place & Route process and higher throughput. In terms

of area requirements, throughput and hardware efficiency, this architecture exhibits excellent abilities compared to the most recent AES architectures, implemented in the Virtex families.

Its performance results and low resource requirements make the architecture suitable as a module for the SR platform, which handles several cryptographic algorithms and is applicable in secure communication systems, where devices or networks require cryptographic solutions with high flexibility and high throughput.

## 6.2 *AESCCM* Implementation

The AES-CCM algorithm lies at the core of the security architectures used by important communication networks such as IEEE 802.11i and IEEE 802.16e-2005 standards. The algorithm, based on the special operating modes of the Advanced Encryption Standard, provides authentication and ciphering services. The proposed architecture supports the two operating modes of the AES-CCM algorithm, the CTR (Counter) mode and the CBC-MAC (Cipher Block Chaining - Message Authentication Code) mode. The design methodology is based on the use of specialized functional modules that results in a highly efficient implementation. Results of implementing the architecture on a FPGA device are presented and compared against similar architectures

The two versions of the *AESCCM* hardware architecture, see Figs. 4.15 and 4.16, were synthesized, mapped and routed for Spartan-3 and Virtex-4 FPGA devices by using Xilinxs ISE 8.2 design tools. Results of implementing the AES-CCM algorithm in a Spartan-3 device were obtained in order to make a fair comparison with other works that are based on these devices.

Implementation results such as the execution time, IOBs, the hardware resources, latency, throughput, and throughput/area ratio are shown in Table 6.3. The implementation efficiency enables to measure the reached throughput and the number of slices that each implementation consumes.

For both Virtex-4 and Spartan-3 implementations, diverse advantages are reached. One of them is that the reported period, used hardware resources (slices, LUTs and BRAMs) and clock cycles are decreased from $AESCCMv1$ to $AESCCMv2$. Furthermore, by Eq. 2.1, the throughput is improved due to the use of a small period and few clock cycles. For example, $AESCCMv2$ on Virtex-4 needs ten clock cycles to process a 128-bit data block, working at 145.11 MHz; the combinational logic implementation provides 1.857 Gbps.

Table 6.3: FPGA resources and characteristics of the *AESCCM* hardware architecture

| Parameter/ Architecture | AESCCMv1 Virtex-4 XC4VLX100 -11 | AESCCMv1 Spartan-3 XC3S4000 -4 | AESCCMv2 Virtex-4 XC4VLX100 -11 | AESCCMv2 Spartan-3 XC3S4000 -5 |
|---|---|---|---|---|
| Period (ns) | 7.229 | 17.474 | 6.891 | 12.611 |
| Clock (MHz) | 138.33 | 57.22 | 145.11 | 79.29 |
| IOBs | 581 | 581 | 581 | 581 |
| Slices | 1364 | 1341 | 1229 | 1191 |
| LUTs | 2106 | 1975 | 1816 | 1756 |
| BRAMs | 20 | 20 | 19 | 19 |
| Clock cycles | 10 | 10 | 10 | 10 |
| Throughput (Gbps) | 1.770 | 0.732 | 1.857 | 1.015 |
| Throughput/Area (Gbps/Slices x$10^{-3}$) | 1.297 | 0.545 | 1.511 | 0.852 |

Throughput/area ratio of the $AESCCMv2$ implementation is high compared against the other proposed $AESCCM$ implementations. Using few hardware resources in the hardware architecture with a high throughput increases the hardware efficiency. 581 IOBs are required for the three proposed $AESCCM$ hardware architectures.

For both FPGA devices, $AESCCMv1$ hardware architecture uses more hardware resources than $AESCCMv2$, due to the two complete $AES\_Cipher$ modules. Parallelization of these modules on the design of the $AESCCMv1$ hardware architecture has the aim of increasing the performance, but using more hardware resources for placing and routing increases the clock period or the critical path. By simplifying $AES\_Cipher$ modules using $AES\_GenKey$ as a common module, $AESCCMv2$ requires less hardware resources and BRAMs than the $AESCCMv1$ implementation. So, placing and routing connect fewer components, the critical path of $AESCCMv2$ is shorter than the other implementation, overcoming the reported throughput.

A performance comparison in terms of frequency, throughput and implementation efficiency among the $AESCCM$ hardware implementations is shown in Table 6.4.

Compared with previous published $AESCCM$ implementations [López-Trejo et al., 2005], [Aziz and Ikram, 2007], [ductor Inc., 2008], the proposed $AESCCMv2$ implementation on Virtex-4 achieves the highest throughput. It can be used on applications that transmit data up to 1.857 Gbps. Furthermore, this implementation reports the highest throughput/area ratio compared against the related work, which indicates that it has a high throughput by using few hardware resources. The proposed iterative architecture achieves higher throughput/area ratio than the rest of the related works. This is due to its high clock frequency (or short critical path), few hardware resources and its low latency.

For $AESCCM$ hardware implementations on Spartan-3, $AESCCMv2$ implementation reports similar and in most cases better clock frequency. This implementation is slightly slower (about 3.5%) in terms of throughput compared against [López-Trejo et al., 2005], which reports the highest throughput, although at the expense of almost twice the number of slices and three times the number of BRAMs. The proposed $AESCCMv2$ implementation uses the FPGA hardware resources in an efficient manner, and it uses 44.70% fewer slices, utilizes 66.03% fewer BRAMs, and has 20.79 slower clock frequency. Considering implementation efficiency, the $AESCCMv2$ implementation has the second implementation efficiency or throughput/area

Table 6.4: *AESCCM* hardware implementations

| Work-device | Clock (MHz)-Clock cycles | Slices - BRAM | Through-put (Gbps) | Throughput /Area (Gbps/Slices x$10^{-3}$) |
|---|---|---|---|---|
| [López-Trejo et al., 2005] XC3S4000 | 100.08 12 | 2154 106 | 1.051 | 0.488 |
| [Aziz and Ikram, 2007] XC3S50 | 247.00 46 | 487 4 | 0.687 | 1.411 |
| [ductor Inc., 2008] ASIC | 73.00 12 | - - | 0.800 | - |
| This work-XC4VLX100 | 145.11 10 | 1229 19 | 1.857 | 1.511 |
| XC3S4000 | 79.29 10 | 1191 19 | 1.015 | 0.852 |

ratio compared against [Aziz and Ikram, 2007], which has the biggest ratio, although the proposed $AESCCMv2$ implementation achieves 47.75% higher throughput, and reports 20.77% slower clock frequency than the implementation in [Aziz and Ikram, 2007].

The hardware design methodology allows providing an iterative hardware architecture with low hardware resources utilization and high throughput, obtaining a balanced implementation. Several BRAMs are used to handle data in parallel and registers were added for data multiplexing and key storage in order to reduce the critical path, resulting in a small hardware architecture that in turn results in a more efficient place and route process and higher throughput. Hardware architectures report a better performance than software ones, but using more hardware resources do not necessary increment the performance, it is necessary to use a good hardware design methodology. In this work, trade-off studies between throughput and area are made to reach a good performance of the proposed hardware design. Throughput/area ratio is improved by designing a compact architecture, using parallelization of data buses and components, identification of common processing elements, and specialization of modules, focused to design iterative hardware architecture with a good throughput/area ratio.

Important issues in secure communication systems are that systems or networks requiring AES-CCM cryptographic solutions need upgrades or new elements of hardware, which stimulate design and development of new hardware architectures such as $AESCCM$ with high performance. This architecture is used to implement the security schemes of the IEEE 802.11i and IEEE 802.16e-2005 standards. AES-CCM is considered a secure algorithm as it is based on AES cryptographic algorithm, and it can be used for both, authentication and ciphering.

## 6.3 $AESCCMP$ Implementation

This project presents a custom hardware architecture for the AES-CCM Protocol (AES-CCMP) which is the basis for the security scheme of the IEEE 802.11i standard. AES-CCMP is based on the AES-CCM algorithm that performs the Advanced Encryption Standard in CTR with CBC-MAC mode (CCM mode). The general approach used in this work aimed to obtain an iterative architecture with low hardware resources utilization. Results of implementing the proposed architecture targeting FPGAs devices are pre-

sented and discussed. Comparisons against similar works show significant improvements in terms of both throughput and area.

The results of synthesizing the proposed architecture for the AES-CCMP are presented in this section. For the purpose of validation, comparison and prototyping, the *AESCCMP* architecture was synthesized, mapped, placed and routed for the Xilinx Virtex-4 LX FPGA device. Also, in order to have a fair comparison against similar works, the architecture was also implemented targeting Xilinx Virtex-II and Spartan-3 FPGA devices. The implemented architecture was simulated and verified considering real-time operation condition by using the design conformance test data, provided with the IEEE 802.11i standard [LAN/MAN-Standards-Committee, 2004]. In operating mode, the *AESCCMP_Architecture* is able to cipher a 128-bit input block with a 128-bit key to produce a 128-bit output block of ciphered data every ten clock cycles.

Table 6.5 shows implementation results of the *AESCCMP_Architecture* for the three selected FPGA devices, achieving the maximum operating frequency and throughput in the Virtex-4 FPGA. The hardware implementation reports better results depending on the technology. The important part is the comparisons, which are made against related works, see Table 6.6.

Table 6.5: Implementation results of the proposed *AESCCMP* hardware architecture for three different technologies

| Parameter<br>Architecture | *AESCCMP*<br>Virtex-4<br>XC4LX100-12 | *AESCCMP*<br>Virtex-II<br>XC2V1000-6 | *AESCCMP*<br>Spartan-3<br>XC3S4000-5 |
|---|---|---|---|
| Period (ns) | 5.799 | 7.437 | 14.111 |
| Clock (MHz) | 172.44 | 134.46 | 70.86 |
| Slices | 2001 | 1679 | 1730 |
| BRAMs | 20 | 20 | 20 |
| Clock cycles | 10 | 10 | 10 |
| Throughput (Gbps) | 2.207 | 1.721 | 0.907 |
| Throughput/Area<br>(Gbps/Slices x$10^{-3}$) | 1.103 | 1.025 | 0.524 |

The architecture presented in [Aziz et al., 2005] reports an *AESCCMP* architecture with a throughput of 0.127 Gbps, running at 63.7 MHz. This

architecture has been designed focusing on efficiency and low power consumption. In [Shim et al., 2004], authors present an AES architecture with CCMP and OCB modes, which is implemented in a Xilinx VirtexE FPGA, reporting 0.243 Gbps at 50 MHz. To process 2312 bytes, MIC generation needs 1 clock cycle; counter initialization needs 1 clock cycle, and CTR encryption needs 145 clock cycles. [Smyth et al., 2006] reports a security processor, which is designed to offload cryptographic processing from the host microprocessor achieving a throughput of 0.275 Gbps. [Bae et al., 2006] reports a hardware architecture operating with a clock frequency of 50 MHz, which ciphers a block every 44 clock cycles. Commercial platforms are reported in [Quatech-Inc., 2007], [Hi/fn-Inc., 2008], [RadiSys-Corporation, 2006]. [Quatech-Inc., 2007] presents a radio platform, which supports AES/CCMP, WPA, and WEP security mechanisms with a maximum data rate of 0.054 Gbps. [Hi/fn-Inc., 2008] reports a security processor, reporting a maximum data rate of 0.275 Gbps. [RadiSys-Corporation, 2006] presents a processor with multiple security mechanisms, encountering AES-CCMP. Specific characteristics are not provided, and it is reported that the processor has the capability of cipher/decipher data up to 2 Gbps, but this datasheet does not specify if AESCCMP process reaches 2 Gbps. Finally, an additional work in [Sivakumar and Velmurugan, 2007] reports an AESCCMP hardware architecture, although details about the design, its implementation and the comparison are only presented for the AES hardware architecture, thus, it is not included in Table 6.6.

As seen in Table 6.6, there are few reported implementations of AES algorithm in mode CCMP. These have a throughput inferior to 1 Gbps, where [Smyth et al., 2006] and [Hi/fn-Inc., 2008] reports the higher throughput of 0.275 Gbps. [Aziz et al., 2005] uses fewer slices with a low throughput, although a design using more FPGA logic rarely has a proportional throughput. The other works report processors, which have a low throughput with a high flexibility. The proposed $AESCCMP$ architecture implementation in this project has an improved structure with the highest performance, which offers high hardware implementation efficiency.

Hardware implementations are required to support new wireless communication applications, which have high data transmission rates. In the design of hardware architectures, there is rarely a proportional throughput when using more hardware resources, but these should apply a complementary design, utilizing hardware advantages and certain approaches. An efficient iterative hardware architecture of the IEEE 802.11 security architecture is reported in

Table 6.6: Implementation results of the $AESCCMP$ hardware architectures

| Work-device | FPGA resources - Clock (MHz) | Throughput (Gbps) | Efficiency |
|---|---|---|---|
| [Aziz et al., 2005] - Spartan-3 | 523, 63.70 | 0.127 | 0.243 Mbps/Slices |
| [Shim et al., 2004] - VirtexE | 3750, 50.00 | 0.243 | 0.064 Mbps/Slices |
| [Smyth et al., 2006] - Virtex-II | 3474, 15 BRAM, 80.30 | 0.275 | 0.079 Mbps/Slices |
| [Bae et al., 2006] - Stratix | 5605, 50.00 | 0.258 | 0.046 Gbps/logic cells |
| [Quatech-Inc., 2007] - ASIC module | - | 0.054 | - |
| [Hi/fn-Inc., 2008] - Processor | 66.00 | 0.275 | - |
| [RadiSys-Corporation, 2006] - Processor | - | $>$ 1500.00 2 | - |
| This work - Virtex4-LX | 2001, 38 BRAM, 172.42 | 2.207 | 1.102 Mbps/Slices |
| This work - Virtex-II | 1679, 38 BRAM, 134.46 | 1.721 | 1.025 Mbps/Slices |
| This work - Spartan-3 | 1730, 38 BRAM, 70.85 | 0.907 | 0.524 Mbps/Slices |

this project. The proposed architecture supports AES-CCM Protocol. The general approach used in this work aimed to obtain an iterative architecture with low hardware resources utilization, obtaining an $AESCCMP$ hardware implementation with a high performance and a balanced ratio of hardware resources and throughput. Trade-off analysis about parallelization was made, and a common module was identified, enabling the proposed designs to achieve both a major hardware resources reduction and a high throughput. The specialized modules are based on computing AES algorithm in different modes of operation and formatting of data input, parallelizing data buses and modules. Custom particular control sub-modules are designed for each specialized module, having a simple main control and a short critical path. The implementation results were presented and compared with related designs, showing that the proposed $AESCCMP$ design reports the highest efficiency.

## 6.4 $AESCCM6$ Implementation

IEEE 802.16e-2005 Standard has specified security mechanisms, using the AES-CCM algorithm to provide better security services, although it is required to execute a great number of operations, several iterations, and multiple processes. In this project, a hardware architecture based on AES-CCM for this standard is described, this architecture reports a high throughput/area ratio. The proposed architecture uses parallelization and specialization modular, and reduces critical path without increasing the execution latency that is required by the AES algorithm.

The VLSI synthesis results of the $AESCCM6$ hardware architecture are presented in this section. For the purpose of validation and comparison, $AESCCM6$ architecture was synthesized, mapped, placed and routed for different FPGA technologies: Xilinx Virtex and Xilinx Spartan-3 devices. The synthesized architecture was simulated and verified considering real-time operation condition by using the design conformance test data, provided by the IEEE 802.16e-2005 standard.

If $AESCCM6$ architecture ciphers data, and it is maintained in the ciphering loop, its output bus will offer 128-bit cipher data every ten clock cycles for 128-bit plain data and 128-bit key data. Table 6.7 shows implementation results of the $AESCCM6$ architecture in three different FPGAs, and the implementations on Virtex support more than 1 Gbps, whereas the

implementation on Spartan-3 is close to 1 Gbps. The design reported in this work satisfies the current data transmission of the standards, and the architecture is aimed to meet new standards, which will be used in applications more demanding of cryptographic computational power.

Table 6.7: Implementation results of the $AESCCM6$ architecture for three different technologies

| Parameter<br>Architecture | AESCCM6<br>Virtex-4<br>XC4LX100-12 | AESCCM6<br>Virtex-II<br>XC2V2000-6 | AESCCM6<br>Spartan-3<br>XC3S4000-5 |
|---|---|---|---|
| Period (ns) | 6.644 | 11.071 | 14.219 |
| Clock (MHz) | 150.51 | 90.32 | 70.77 |
| Slices | 1823 | 1476 | 1485 |
| LUTs | 3024 | 2310 | 2320 |
| BRAMs | 20 | 20 | 20 |
| Clock cycles | 10 | 10 | 10 |
| Throughput (Gbps) | 1.926 | 1.156 | 0.905 |
| Throughput/Area<br>(Gbps/Slices x$10^{-3}$) | 1.056 | 0.783 | 0.609 |

Next, comparisons against related work of the $AESCCM$ implementations for security scheme of the IEEE 802.16e-2005 are made, considering that different device families will yield different performance results. Important measurements of hardware $AES$ implementations to consider are FPGA utilized resources, clock frequency, latency and throughput. The performance measurements of the $AESCCMP$ implementations of this work and other designs are shown in Table 6.8 for comparison.

Related works are about processors executing security schemes for IEEE 802.16e-2005, which are commercial products and support AES-CCM algorithm. [Fujitsu Microelectronics America, 2007] is a processor with a set of features for WiMax certification, supporting a security implementation based on AES-CCM algorithm. More details on implementation results are not described. [Hi/fn-Inc., 2005] presents a security processor, which supports different types of cryptographic algorithms, such AES, DES, SHA and RSA. The throughput for the AES-CCM process is of 0.275 Gbps. [Elliptic-Semiconductor-Inc., 2008] reports a processor, supporting several crypto-

Table 6.8: Implementation results of the hardware architectures for IEEE 802.16e-2005 based on AES-CCM algorithm

| Work - Device | Clock (MHz) | Throughput (Gbps) |
|---|---|---|
| [Fujitsu Microelectronics America, 2007]-Processor | - | - |
| [Hi/fn-Inc., 2005]-Processor | - | 0.275 |
| [Elliptic-Semiconductor-Inc., 2008]-Processor | 200.00 | 0.250 |
| [Jetstream-Media-Technologies, 2006]-Spartan-3 | 93.00 | - |
| [Jetstream-Media-Technologies, 2006]-Virtex-4 | 197.00 | - |
| [IPCores-Products-Inc., 2006]-ASIC | 150.00 | 0.960 |
| This work - Spartan-3 | 70.77 | 0.905 |
| This work - Virtex-4 | 150.51 | 1.926 |

graphic algorithms (AES in CBC-MAC, CTR and CCM modes and DES in CBC mode), focused to the IEEE 802.16e-2005 standard. This makes cryptographic processing at the MPDU level, using specialized modules for AES and DES processes. The throughput for the AES-CCM process is about 0.250 Gbps. [Jetstream-Media-Technologies, 2006] presents a hardware implementation for the AES-CCM algorithm, which is specialized in the IEEE 802.16e. This architecture is compact, using few hardware resources: 775 and 793 slices for the Spartan-3 and Virtex-4 devices, respectively, and 3 BRAMs for both devices. Required clock cycles and throughput results are not provided. Finally, [IPCores-Products-Inc., 2006] presents an implementation for the AES-CCM algorithm focused on the IEEE 802.16e, and implementation results are reported for ASIC technology.

Details about these hardware architectures are not provided. These works are generally based in processors, which report low throughput with high flexibility. The implementation on ASIC reports the higher throughput than the other works, running at the frequency of 150 MHz. In this work, it is proposed the design and development of a high-performance architecture, which reports a high throughput/area ratio.

As seen in Table 6.8, there are few reported implementations of the AES-CCM algorithm for IEEE 802.16e applications. These have a through-

put inferior to 1 Gbps, where [IPCores-Products-Inc., 2006] reports the higher throughput of 0.960 Gbps on an ASIC device. Considering FPGA implementations, [Jetstream-Media-Technologies, 2006] uses few hardware resources (slices and BRAMs) but the presented throughput is not reported. The other works report processors, which have a low throughput with a high flexibility. In this project, proposed $AESCCM6$ hardware architecture implementation has an improved structure with the highest performance, which offers high hardware implementation efficiency.

## 6.5 Discussion

Security enhancements are required to support new wireless communication applications, which have a high data transmission. Furthermore, the new security schemes add safety features based on the AES algorithm in CCM mode, which executes several iterations on diverse set of operations, using a complex control and decreasing the speed of the data transmissions. In this point, the hardware architectures present a high performance, but in the design of hardware architectures, there is rarely a proportional throughput when using more hardware resources. This detail was identified when parallelizing generators of keys for the AES ciphers, the throughput of the architecture is lower than when a common generator was used. Also, it is useful balancing sequential and combinational elements to equilibrate paths. An additional design characteristic is adding sequential elements if the critical path is very large, it is to provide a short critical, and it is better if they do not add clock cycles to the latency.

Design methodologies and trade-off studies should be made to improve the performance of these hardware architectures. The general methodology used in this project aimed to obtain an iterative architecture with low hardware resources utilization, taking advantage of the hardware characteristics such as parallelization and specialization of modules, obtaining hardware implementations with a high performance and a balanced ratio of hardware resources and throughput. The specialized modules are based on computing AES algorithm in CBC-MAC, CTR and CCM modes of operation and formatting of data input, parallelizing data buses and modules. Finally, custom particular control sub-modules are designed for specialized modules, having a simple main control and a short critical path.

In this way, efficient iterative hardware architectures of the AES, AES-

CCM, IEEE 802.11 and IEEE 802.16e-2005 security schemes are presented, and result implementations are described to validate a high implementation efficiency and a high throughput.

# Chapter 7

# Software Radio Platform

In this Chapter, the fundamental research of this work results in the proposal of a high-performance reconfigurable architecture. Several schemes are evaluated, considering the proposed security architectures for the IEEE 802.11i and IEEE 802.16e standards presented in the Chapter 5. The proposed hardware architectures have high levels of parallelization, defined-size buses and specialized modules. These architectures are mapped into reconfigurable schemes and controlled by an external module.

These schemes are based on the different types of reconfiguration [Bloget and James-Roxby, 2003]:

1. Full. All devices require a full configuration of the device resources at start time

2. Partial. Configuration of a subset of the device resources

3. Dynamic. Configuration of a subset of the resources by an external device, while rest of device maintains correct operation

4. Self-reconfiguration. Configuration of a subset of the resources by another subset of the chip resources, while rest of chip maintains correct operation

In the next section, the design methodology of the reconfigurable architectures is described.

## 7.1 Design Methodology for the Reconfigurable Architectures

In general terms, the design methodology, see Fig. 7.1, is based on dividing tasks of the hardware security architectures, designing reconfigurable schemes and evaluating their hardware implementations by using test vectors. After that, reconfiguration methodology is applied [Xilinx, 2006], which is based for Virtex devices.

The design and development based on this methodology is a cyclic process. Initially, each HDL design is captured and simulated, next, this design is implemented and a post-place & route model is generated to simulate. After confirming a correct operation, applying reconfiguration methodology is the next step, where several elements are required to insert bus macros, synthesize and establish constraints to the placing and the routing of the reconfigurable architecture. Lastly, validation of the final architecture is executed by evaluating different test vectors.

## 7.2 Reconfigurable Architectures

Several reconfiguration schemes will be evaluated. The two hardware security architectures of the protocols, IEEE 802.11i and IEEE 802.16e, enable to probe the idea of a reconfiguration scheme for the software-radio platform.

Four versions of the reconfigurable platform are designed, considering three types of reconfiguration:

1. $ReconfigurableArchitecture0$, where a full configuration is made, see Fig. 7.2.

   Due the great number of IOBs, the hardware security architectures are modified to decrease the number of input/output blocks, see Figs. 7.3 and 7.4, and it is reached by multiplexing data input and output, reducing hardware resources for the IOBs. This modification helps to place the designs and to partition the special regions for the static and reconfigurable parts.

   These extended hardware architectures are implemented and their configurations are used to completely configure the device.

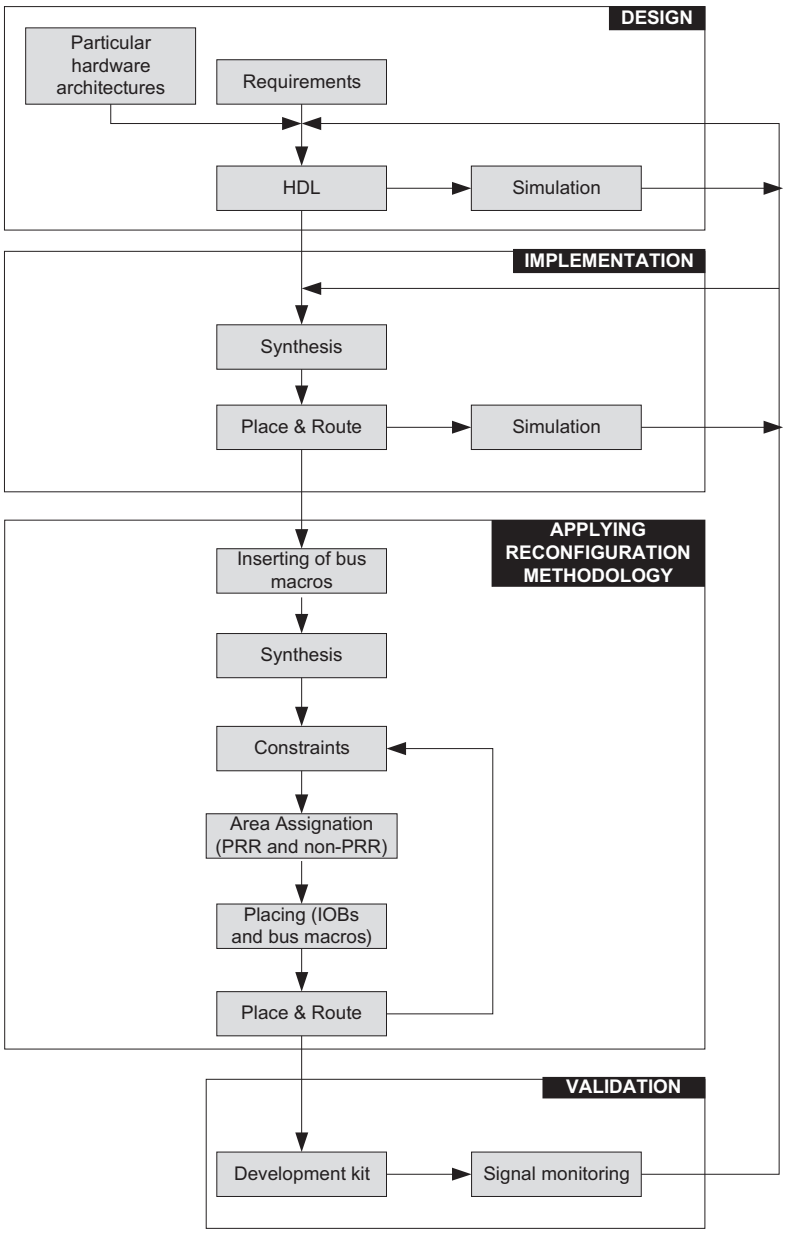


Figure 7.1: Particular design methodology for the reconfigurable architectures

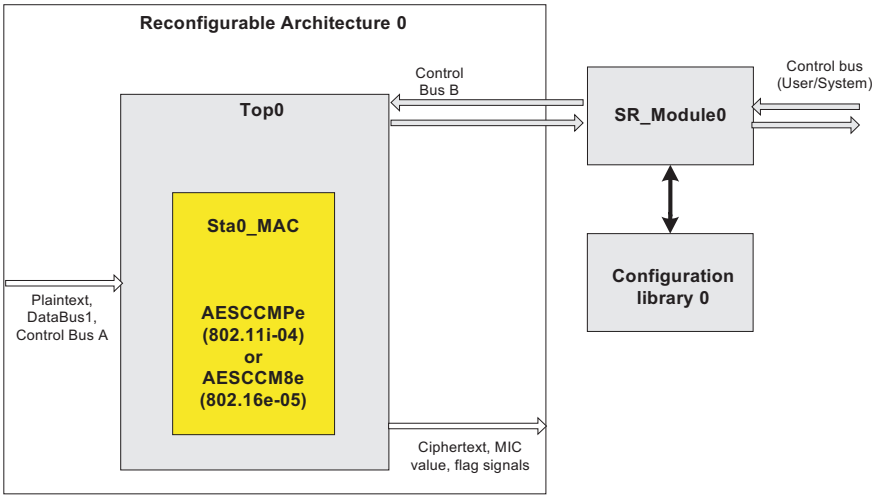


Figure 7.2: Block diagram of the fully configurable architecture

The next two reconfiguration schemes explore the partial reconfiguration, where a part of the device is reconfigured.

2. $ReconfigurableArchitecture1$, where design using partial reconfiguration is developed, see Fig. 7.5, considers a subset of common elements of the hardware security architectures.

   In this scheme, the hardware security architectures are completely used as a reconfigurable module. Unlike the first reconfiguration scheme, the $ReconfigurableArchitecture1$ configures a subset of hardware resources, requiring a shorter time for the reconfiguration of the device. The next scheme uses a smaller area for the reconfiguration.

3. $ReconfigurableArchitecture2$, where a partial configuration is made, see Fig. 7.6, considering the complete hardware security architectures as elements for the reconfiguration.

   In this scheme, several modules are used like common parts, the remaining ones are distributed on reconfigurable regions. The advantage is reached by changing the configuration of few hardware elements. In the next scheme, a dynamic reconfigurable scheme is presented.

4. $ReconfigurableArchitecture3$, where a dynamic configuration is made,

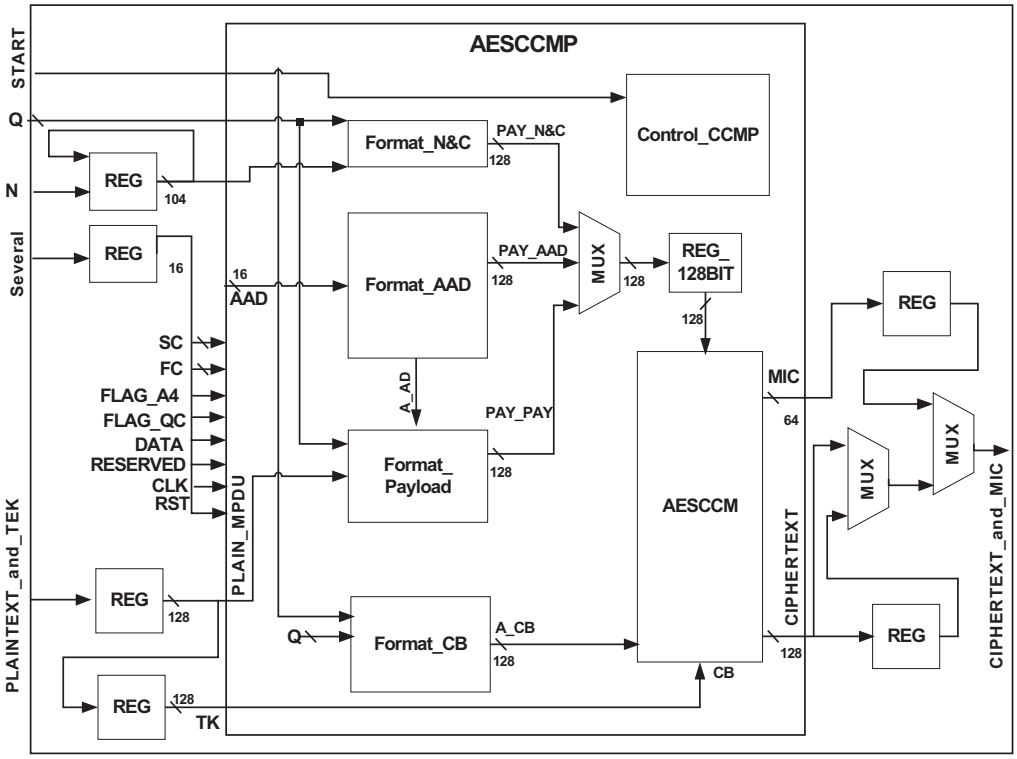


Figure 7.3: Block diagram of the *AESCCMP* extended hardware architecture

Fig. 7.7. In this architecture, two reconfigurable modules support a complete hardware security architecture. The function of the device is maintained by an operative module, whereas the other module can be reconfigured. After, the function can change by using the new configuration, and the other module can change its configuration and use it when this one is required.

In each of these schemes, there are modules ($SR\_Module1/2/3/4$) that control the reconfiguration. The decision of changing the configuration comes from external signals of an upper-layer system or user. For example, a cognitive radio sensing its environment decides the operation of its platform, and can generate these control signals. In a similar way, a radio, such as a SDR or a radio controlled by user, can generate control signals to change the configuration of the proposed architecture.

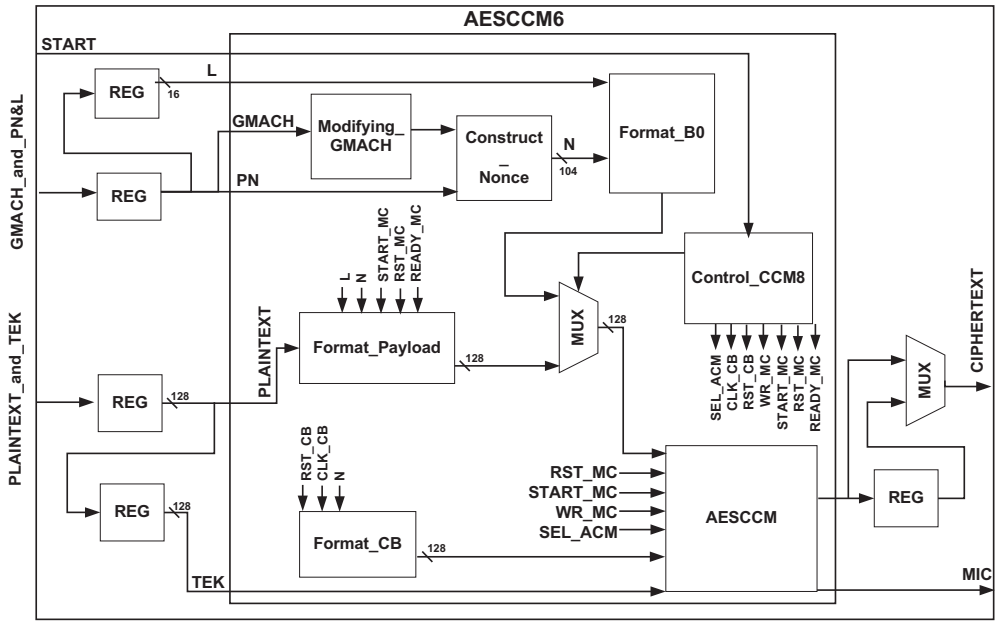


Figure 7.4: Block diagram of the *AESCCM*6 extended hardware architecture

In these schemes, the bus macros are necessary because they are used as fixed data paths for signals going between a reconfigurable part and a static part. The bus macros are defined for each of the devices families.

Virtex-II/4/5 families were used to explore the reconfiguration methodologies [Xilinx, 2006]. In general, the difference is when inserting the bus macros, which depends on the technology. The bus macros of the Virtex-II can be inserted and are directional, they go from left to right or from right to left. In the Virtex-4, bus macros can go in the same way, and additionally, they can go from top to bottom and from bottom to top. Finally, bus macros in the Virtex-5 are not directional. Implementation details are reported in the next section.

## 7.3 Analysis of the Implementation Results

Several tools have been used along the development of the platform. Considering the methodology (see Section 7.1), the HDL design was captured by the FPGA Advantage tool 6.3. These designs are implemented and tested

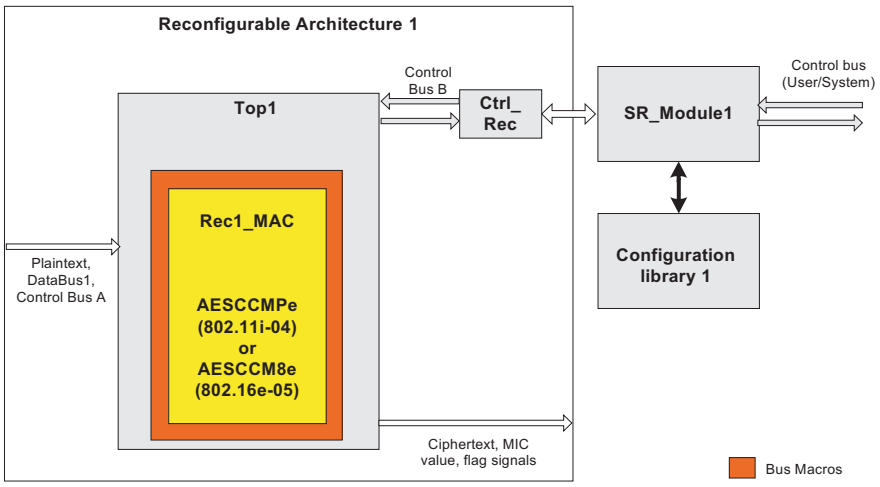


Figure 7.5: Block diagram of the partially reconfigurable architecture 1

using the Xilinx ISE 9.2 and ModelSim 5.8 tools, respectively, see Fig. 7.8. Finally, the reconfiguration methodology is applied by inserting bus macros through FPGA Advantage 6.3, synthesizing these new designs through Xilinx ISE 9.2, establishing constraints, and placing IOBs and bus macros through PlanAhead 9.2 tool, designing the reconfiguration controls through EDK 9.2 and implementing reconfigurable designs through patched-version Xilinx ISE 9.2.04i_PR8, see Fig. 7.9.

A special feature is that applying reconfiguration methodology requires that bus macros are placed on a particular border of the region between static and reconfigurable parts for Virtex-II and 4, whereas bus macros can be placed anywhere within reconfigurable part for Virtex-5.

The previously-described reconfigurable architectures are implemented. In general, each reconfigurable architecture supports two protocols (IEEE 802.11i-2004 and IEEE 802.16e-2005) with two configurations each one. These two configurations are based on: 1) the highly-parallelized *AESCCM* module and 2) the *AESCCM* module with a common element. These experiments are carried out by analyzing and comparing implementation results using highly-parallelized modules versus optimized modules, where their configurations are part of a reconfigurable environment, which allows to evaluate reconfigurable modules.

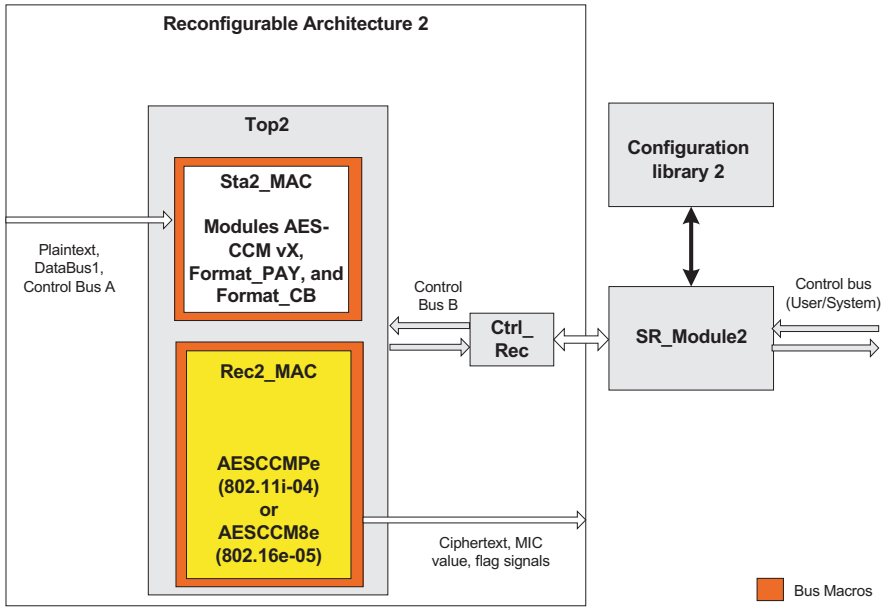


Figure 7.6: Block diagram of the partially reconfigurable architecture 2

The two metrics, throughput and implementation efficiency, are computed by Eq. 2.1 and Eq. 2.2, respectively. To set values of *Plain_data_block_size* and *Clock_cycles*, it is important to point out that computing MIC and cipherdata are executed in parallel by the proposed reconfigurable architectures, and the details considered for computing them are:

1. In the security architectures, the message has a maximum size of 1024 bytes, thus 64 data blocks of 128 bits are obtained.

2. AAD has a size of 0 bytes for the IEEE 802.16e-2005 standard, and none data block is formed. In the IEEE 802.11i-2004, two data blocks are formed from AAD.

3. There is an initial data block, which is formed by using the Nonce value.

4. These architectures need 10 clock cycles more for loading initial data input blocks, 10 clock cycles for processing each data block, and 3 clock cycles for downloading the last data output blocks.

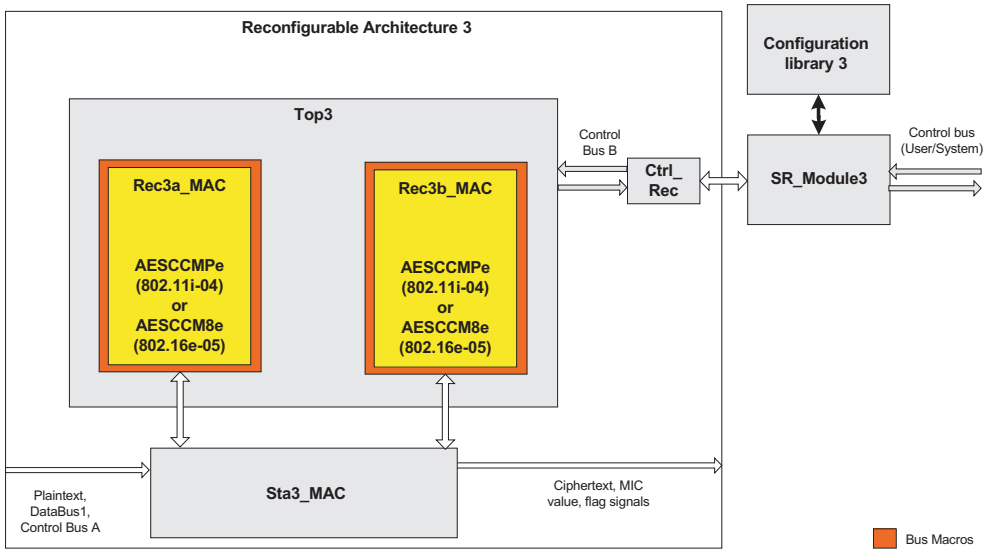


Figure 7.7: Block diagram of the dynamic reconfigurable architecture

So, the reconfigurable architectures for the IEEE 802.16e-2005 process 65 data blocks (64 for the message, and 1 for the initial data block), but the initial data block is considered overhead, so only 64 data blocks have effective bits, i. e., (64) (128 bits) = 8192 bits = $Plain\_data\_block\_size$. These data blocks for authentication (initial block, and message) and ciphering (CBs and message) are processed in parallel, so, it is necessary to process 65 data blocks, requiring (65) (10 clock cycles) + 10 clock cycles + 3 clock cycles = 663 clock cycles. In the same way, the reconfigurable architectures for the IEEE 802.11i-2005 process 67 data blocks (64 for the message, one for the initial data block and two for the AAD) in 683 clock cycles. The value of the $Plain\_data\_block\_size$ is (66) (128 bits) = 8448 bits.

The ratio 8192/663 = 12.356 $\approx$ 12.368 = the ratio 8448/683, and by Eq. 2.1, similar results can be reached, depending in great manner of the Clock_period reported by each one of the implemented architectures.

Next, the implementation results and analysis are presented, considering hardware resources, efficiency and throughput for the reconfigurable architectures. Several FPGA families were used, but the next results are reported for the same device: Virtex-5 xc5vlx155t-1ff1136. This FPGA family contains different hardware elements, classifying the slices such as slice-registers (SRs) and slice-LUTs (SLs), where flip-flops (FFs) are related to the SRs. So, the

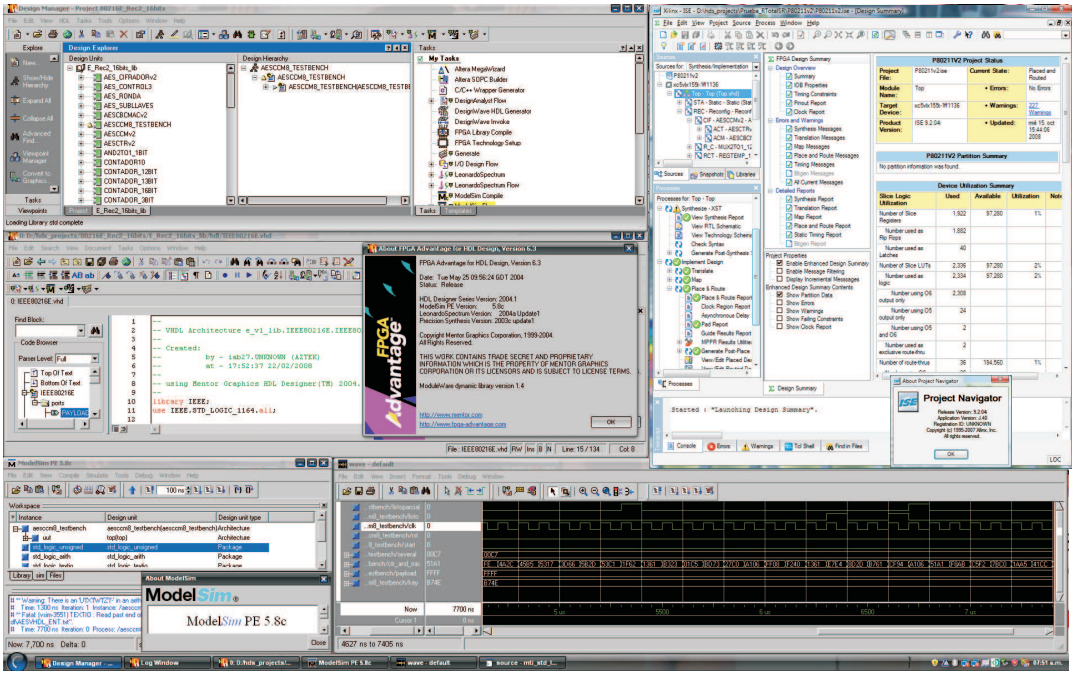


Figure 7.8: FPGA Advantage 6.3, ModelSim 5.8 and Xilinx ISE 9.2 tools

efficiency of the reconfigurable architectures is computed according to the number of SLs, SRs and the SL-FF pairs required in the implementations.

1. $ReconfigurableArchitecture0$

   Implementation results are showed in Table 7.1. Four versions are supported, where versions 1 (v1) have a common module and versions 2 (v2) are highly parallelized. This is reflected in the use of BRAMs, because using a highly-parallelized module requires two S-boxes or a double-port BRAM more.

   The configurations for the IEEE 802.11i-2004 standard ($P80211iv1$ and $P80211iv2$) use more hardware resources than the configurations for the IEEE 802.16e-2005, which is consequence of the specialized module to format the AAD data input. In spite of this, the first configurations report a shorter critical path than the second ones, resulting in a higher throughput. Using fewer hardware resources by the common module and reporting a shorter critical path helps to improve the efficiency.

Table 7.1: Implementation results of the *ReconfigurableArchitecture0*

| Parameter - Configuration | *P*80211*iv*1 | *P*80211*iv*2 | *P*80216*ev*1 | *P*80216*ev*2 |
|---|---|---|---|---|
| RAMB18X2s | 9 | 10 | 9 | 10 |
| Slice-Registers (SRs) | 1838 | 1962 | 1527 | 1651 |
| Flip-flops (FFs) | 1798 | 1922 | 1490 | 1614 |
| Latches | 40 | 40 | 37 | 37 |
| Slice-LUTs (SLs) | 2132 | 2336 | 1700 | 1832 |
| SL-FF pairs | 2719 | 3177 | 2154 | 2422 |
| Clock period (ns) | 6.528 | 6.758 | 7.454 | 7.501 |
| Clock frequency (MHz) | 153.18 | 147.97 | 134.15 | 133.31 |
| Throughput (Gbps) | 1.894 | 1.830 | 1.657 | 1.647 |
| Efficiency (Mbps/SR) | 1.030 | 0.932 | 1.085 | 0.997 |
| Efficiency (Mbps/SL) | 0.888 | 0.783 | 0.974 | 0.899 |
| Efficiency (Mbps/SL-FF) | 0.696 | 0.576 | 0.769 | 0.680 |

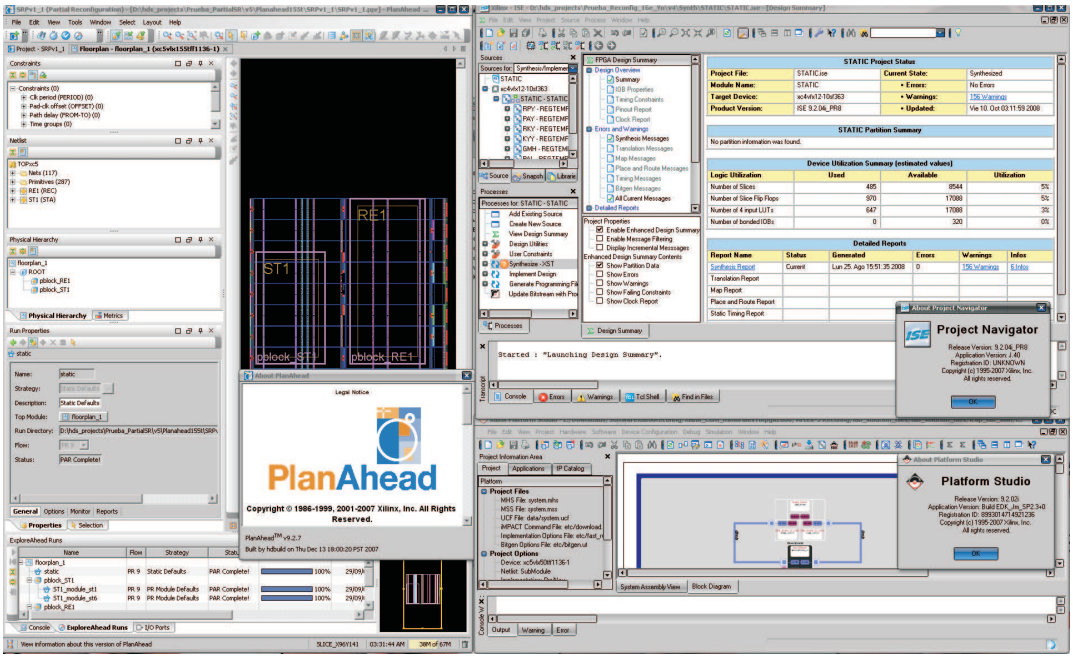


Figure 7.9: PlanAhead 9.2, Xilinx ISE 9.2.04i_PR8 and EDK 9.2 tools

By comparing architectures with a common module against the highly-parallelized ones, it can be seen that although versions 2 have high levels of parallelization -which should increase the performance- the $ReconfigurableArchitecture0$ reports highest throughput and efficiency using versions 1. Fewer hardware resources and a shorter critical path are reached due to the use of the common module. This architecture can support other cryptographic operations or security standards, where configurations should be placed and routed in the reconfigurable part.

The $ReconfigurableArchitecture0$ reports a better efficiency by using the IEEE 802.16e-2005 configurations instead of the IEEE 802.11i-2004 configurations, and the opposite situation happens with the throughput.

2. $ReconfigurableArchitecture1$

   The $ReconfigurableArchitecture1$ is based on a partially reconfigurable scheme, and the implementation results are showed in Table 7.2.

The reconfigurable part has the complete configurations for the hardware security architectures of the IEEE 802.11i-2004 and the IEEE 802.16e-2005 standards.

The static part, which has the routing resources between the bus macros and the IOBs of the device, is the basic element for each of these configurations. This part only uses 72 SLs (72 SL-FF pairs), where the $Clock\_period$ of the architecture is mainly determined by the clock period of the reconfigurable part. For computing efficiency, the hardware resources of both reconfigurable and static parts are taken into account.

The implementation results show that the configurations for IEEE 802.11i-2004 utilize half of BRAMs compared to the configurations for IEEE 802.16e-2005. These memories were mapped and placed by the tool. Using the double of BRAMs requires to connect more hardware elements, provoking a greater critical path. However, using the half of BRAMs entails to use a smaller amount of hardware resources with a better throughput and efficiency.

Considering versions 1 against the versions 2, the configuration for IEEE 802.11i-2004 using a common module reports a better throughput than the highly-parallelized one, which occurs in the same manner that in the implementation results of the $ReconfigurableArchitecture0$. This does not happen in the configurations for the IEEE 802.16e-2005, which require the double of the BRAMs, needing more hardware resources for routing these elements. This produces greater critical paths, decreasing the throughput and, because there are more hardware elements, the efficiency is very reduced. In addition, comparing version 1 against version 2, the first one reports worse throughput but better efficiency due to the fewer hardware resources for placement and routing.

3. $ReconfigurableArchitecture2$

   This architecture is also based on the partially reconfigurable scheme that consists of two reconfigurable modules and a static part. Moreover, it supports the same four configurations: two for IEEE 802.11i-2004 and two for IEEE 802.16e-2005. The associated control module manages the four configurations, where $ST1$ reconfigurable module can only work with the $P80211iv1$ and $P80211iv1$ reconfigurable modules, and $ST6$ reconfigurable module can only work with the $P80216ev1$

Table 7.2: Implementation results of the *ReconfigurableArchitecture*1

| Parameter - Configuration | *P*80211*iv*1 | *P*80211*iv*2 | *P*80216*ev*1 | *P*80216*ev*2 |
|---|---|---|---|---|
| RAMB18X2s | 9 | 10 | 18 | 20 |
| Slice-Registers (SRs) | 1693 | 1703 | 1527 | 1655 |
| Flip-flops (FFs) | 1553 | 1663 | 1490 | 1618 |
| Latches | 40 | 40 | 37 | 37 |
| Slice-LUTs (SLs) | 4194 | 4398 | 6297 | 6954 |
| SL-FF pairs | 5559 | 5975 | 7708 | 8491 |
| Clock period (ns) | 8.057 | 8.388 | 11.016 | 10.166 |
| Clock frequency (MHz) | 124.11 | 119.21 | 90.77 | 90.82 |
| Throughput (Gbps) | 1.535 | 1.474 | 1.121 | 1.215 |
| Efficiency (Mbps/SR) | 0.906 | 0.865 | 0.734 | 0.734 |
| Efficiency (Mbps/SL) | 0.365 | 0.335 | 0.178 | 0.174 |
| Efficiency (Mbps/SL-FF) | 0.276 | 0.246 | 0.145 | 0.143 |

and $P80216ev2$ reconfigurable modules. For computing efficiency, the hardware resources of the $STx$ module together with the ones of its relative modules are considered. Moreover, the clock period of the $STx$ module is shorter than the relative modules, and for computing the throughput, the reported clock for the relative module is considered. The implementation results are showed in Table 7.3.

Similarly to the first partially reconfigurable architecture, the static part of the $ReconfigurableArchitecture2$ uses 72 SLs (72 SL-FF pairs) and 69 IOBs that are taken into account to compute the efficiency. In this architecture, the configurations for IEEE 802.11i-2004 and IEEE 802.16e-2005 use the same number of BRAMs, depending on the versions 1 or 2.

In this reconfigurable architecture, the critical path is due to the communication between the two reconfigurable modules, and these paths are larger than the ones of the previous reconfigurable architectures, then more hardware resources are required and connected by larger paths in the configurations for the IEEE 802.11i-2004. In this way, these configurations report poorer throughput and efficiency than the configurations for the IEEE 802.16e-2005.

The configurations v1 report a better throughput and efficiency than the versions 2. In this reconfigurable scheme, the configurations using a common module present a shorter critical path than the configurations with a highly-parallelized structure. Moreover, the architectures for IEEE 802.11i-2004 use more hardware resources for formatting the AAD input, which affects the throughput and the efficiency of the implementations.

4. $ReconfigurableArchitecture3$

   This reconfiguration scheme presents a dynamic reconfigurable architecture, where its functionality is maintained while other part of the architecture changes. This fact affects the configuration and a new function is provided. In this case, two reconfigurable modules are chosen, which contain a complete configuration conformed by the hardware security architectures of the IEEE 802.11i-2004 and IEEE 802.16e-2005 standards. Considering the same two functions for each standard, these four configurations should be implemented in each left ($L$) or right ($R$)

Table 7.3: Implementation results of the *ReconfigurableArchitecture*2

| Parameter - Configuration | *ST*1 | *P*80211*iv*1 | *P*80211*iv*2 | *ST*6 | *P*80216*ev*1 | *P*80216*ev*2 |
|---|---|---|---|---|---|---|
| RAMB18X2s | 0 | 18 | 20 | 0 | 18 | 20 |
| Slice-Registers (SRs) | 1315 | 587 | 715 | 1007 | 520 | 648 |
| Flip-flops (FFs) | 1275 | 587 | 715 | 970 | 520 | 648 |
| Latches | 40 | 0 | 0 | 37 | 0 | 0 |
| Slice-LUTs (SLs) | 2326 | 6078 | 6735 | 1915 | 6077 | 6734 |
| SL-FF pairs | 3515 | 6657 | 7442 | 2824 | 6589 | 7374 |
| Clock period (ns) | 1.758 | 16.091 | 16.633 | 1.670 | 13.719 | 14.792 |
| Clock frequency (MHz) | - | 62.14 | 60.12 | - | 72.89 | 67.60 |
| Throughput (Gbps) | - | 0.768 | 0.743 | - | 0.900 | 0.835 |
| Efficiency (Mbps/SR) | - | 0.426 | 0.366 | - | 0.589 | 0.504 |
| Efficiency (Mbps/SL) | - | 0.090 | 0.081 | - | 0.111 | 0.095 |
| Efficiency (Mbps/SL-FF) | - | 0.074 | 0.067 | - | 0.094 | 0.081 |

side of the reconfigurable architecture. The implementation results are showed in Table 7.4.

In the previous reconfigurable architectures, 69 IOBs are required, but in the $ReconfigurableArchitecture3$, its control module requires to select the outputs from the left or right side through multiplexors. This causes that the static part has 216 SLs (216 SL-FF pairs) and 70 IOBs.

The critical path size is affected by several elements, originated by a complex situation due to several details: i) the amount of hardware resources for placement and routing for supporting two hardware security architectures, ii) the input and output blocks required for external communication that cross the whole device containing the $ReconfigurableArchitecture3$, and iii) the number of BRAMs used like double-port and simple modes. The throughput is directly dependent of the implementation clock period, and this situation greatly increases or decreases the critical path of each configuration, resulting in a decreased or increased throughput, respectively. Both configurations for the IEEE 802.11i-2004, left and right sides, are implemented by the tool using the BRAMs like double-port memories, whereas in the configurations for IEEE 802.16e-2005, the BRAMs are implemented like simple memories. In this way, the use of the double of BRAMs requires more hardware resources for placement and routing, provoking a larger critical path.

Furthermore, the configurations of the right side require a smaller critical path for outputting than the other configurations of the left side, because output blocks are placed in the left side. The input blocks do not affect the critical path since their placement is located in the middle of the selected device. This supports the $ReconfigurableArchitecture3$, but this characteristic can not be available in other devices, affecting the critical path. The static part should select from the outputs of the two reconfigurable modules for externally outputting data, requiring that the output blocks be placed on the right lateral edge of the device. If the placement of these blocks is in the left lateral edge, larger critical path are reported by the configurations of the right reconfigurable module.

Considering the right configurations in $ReconfigurableArchitecture3$,

Table 7.4: Implementation results of the *ReconfigurableArchitecture3*

| Parameter - Configuration | P80211 *iv1L* | P80211 *iv2L* | P80216 *ev1L* | P80216 *ev2L* | P80211 *iv1R* | P80211 *iv2R* | P80216 *ev1R* | P80216 *ev2R* |
|---|---|---|---|---|---|---|---|---|
| RAMB18X2s | 9 | 10 | 18 | 20 | 9 | 10 | 18 | 20 |
| Slice-Registers (SRs) | 1693 | 1703 | 1527 | 1655 | 1693 | 1703 | 1527 | 1655 |
| Flip-flops (FFs) | 1553 | 1663 | 1490 | 1618 | 1553 | 1663 | 1490 | 1618 |
| Latches | 40 | 40 | 37 | 37 | 40 | 40 | 37 | 37 |
| Slice-LUTs (SLs) | 4266 | 4470 | 6369 | 7026 | 4266 | 4470 | 6369 | 7026 |
| SL-FF pairs | 5584 | 6041 | 7777 | 8566 | 5584 | 6056 | 7782 | 8565 |
| Clock period (ns) | 12.078 | 12.336 | 18.073 | 20.672 | 10.615 | 13.895 | 16.030 | 15.354 |
| Clock frequency (MHz) | 82.79 | 81.06 | 55.33 | 48.37 | 94.20 | 71.96 | 62.38 | 65.12 |
| Throughput (Gbps) | 1.024 | 1.002 | 0.683 | 0.597 | 1.162 | 0.890 | 0.770 | 0.804 |
| Efficiency (Mbps/SR) | 0.604 | 0.588 | 0.447 | 0.350 | 0.684 | 0.522 | 0.504 | 0.486 |
| Efficiency (Mbps/SL) | 0.240 | 0.213 | 0.103 | 0.082 | 0.247 | 0.189 | 0.116 | 0.111 |
| Efficiency (Mbps/SL-FF) | 0.183 | 0.160 | 0.085 | 0.067 | 0.185 | 0.141 | 0.096 | 0.091 |

they should present a similar critical path to the one of the configurations of the other reconfigurable architectures. This does not happen because the static part of the $ReconfigurableArchitecture3$ is completely combinational, multiplexing the data buses of its reconfigurable modules and requiring a larger path for outputting. These paths can be decreased by adding registers which delays the data buses in a clock cycle. At the end, only a clock cycle is added to the latency which slightly affects the throughput and efficiency, but with a shorter critical path, increases considerably the performance and competes against the other proposed reconfigurable architectures in this work.

Except for configuration $P80216ev1R$, the versions 1 with a common module report a better throughput configurations with a common module against highly-parallelized configurations (versions 2). The performance of these configurations is very close, which is originated by the placement of hardware resources and the static part of the reconfigurable architecture 3.

Analysis and evaluations of the different proposed reconfigurable architectures have been presented, and in the next section, these architectures are compared against related works.

## 7.4 Comparisons

The design and evaluation of these reconfigurable schemes are based on IEEE 802.11i-2004 and IEEE 802.16e-2005 standards, which are established for the WLAN and WMAN, respectively. Few works are related to security architectures for wireless protocols and reconfigurable platforms [Sklavos et al., 2005], [Hi/fn-Inc., 2008], and [Gehrmann and Sthl, 2006]. In the Table 7.5, the proposed architectures are compared against the related works.

The architecture described in [Sklavos et al., 2005] reports implementation results about of two algorithms to cipher data and to examine a platform that supports both cryptographic algorithms. The last two are commercial architectures based on processor, which support a large set of cryptographic algorithm. These works implement different algorithms to be operated in different protocols. Compared with [Sklavos et al., 2005] and [Hi/fn-Inc., 2008], the implementations proposed in this project allocate more area and higher operation frequency, with a higher performance. The reached throughput is

Table 7.5: Comparisons of implementation results

| Work | Supported protocols | Cryptographic Algorithms & Implementation results |
|---|---|---|
| [Sklavos et al., 2005] | IEEE 802.11 | AES - 323 CLBs, 57 MHz, 177 Mbps<br>WEP - 750 CLBs, 40 MHz, 2.2 Mbps |
| [Hi/fn-Inc., 2008]<br>-7956 Processor | IPSec, SSL/TLS | AES/SHA-1, 554 Mbps<br>88 DH exchanges per second<br>66 Mhz |
| [Gehrmann and Sthl, 2006] | Several on the cellular networks | UMTS (f8 and f9 using Kasumi)<br>GSM, A5/1, A5/2, A5/3, SHA-1, MD5<br>GPRS, GEA1, GEA2, GEA3, and GEA4<br>DES, 3DES, AES, RSA and DH |
| This work -<br>*Reconfigurable*<br>*Architecture*0 | IEEE 802.11i-2004<br><br>IEEE 802.16e-2005 | AES - 1838 SRs, 2132 SLs, 9 BRAMs, 1.894 Gbps<br><br>AES - 1527 SRs, 1700 SLs, 9 BRAMs, 1.657 Gbps |

better compared with the implementations of the related works. Since these works do not present any efficiency results, an efficiency comparison can not be made.

There are many works specialized in some algorithm (see Sections 6.1 and 6.2) or some security protocol (see Sections 6.3 and 6.4), but different results are obtained when the architectures support complete protocols. The final implementation results of this reconfigurable architecture, which also executes formatting of data, are compared against specialized related work, for the IEEE 802.11i-2004 security architectures (see Table 7.6) and for the IEEE 802.16e-2005 security architectures (see Table 7.7).

For the IEEE 802.11i-2004 standard, the reconfigurable architectures excepting the number 2 report higher throughput than the software implementation on processor and hardware implementation on ASIC. The hardware implementation on FPGA does not present results about of throughput and efficiency, but the proposed reconfigurable architectures use more hardware resources. The results of the reconfigurable architecture 2 show that by making new enhancements, this architecture can reach a similar performance.

For IEEE 802.16e-2005, the processor in [RadiSys-Corporation, 2006] reports the highest throughput based on ASIC, but with a higher clock frequency, which is many times greater. Comparing against the architectures implemented on FPGAs, the proposed reconfigurable architectures report higher throughput and the highest efficiency. The reconfigurable architectures require more hardware resources due to the distribution of elements to the configuration of the reconfigurable schemes.

The proposed reconfigurable architectures in this work present a competitive throughput and efficiency, when they are compared against the related works. This is achieved due to the design methodology used, see Section 4.4, which enables to implement modules with high performance that can be supported and evaluated in the development of the different reconfigurable architectures. And although the throughput and efficiency decrease when the modules are mapped into the reconfigurable architectures, the analysis allows to take decisions for improving the performance by adding new branches to the design methodology (see Section 7.1) that allow decreasing the critical path by adding sequential elements.

Table 7.6: Implementation results of the hardware architectures for the IEEE 802.11i-2004 networks

| Work - Device | FPGA resources | Clock (MHz) | Throughput (Gbps) | Efficiency |
|---|---|---|---|---|
| [Hi/fn-Inc., 2005]-Processor | - | - | 0.275 | - |
| [Elliptic-Semiconductor-Inc., 2008]-Processor | - | 200.00 | 0.250 | - |
| [Jetstream-Media-Technologies, 2006]-Virtex-5 | 238 Slices, 3 BRAM | 316.00 | - | - |
| [IPCores-Products-Inc., 2006]-ASIC | 150.00 | - | 0.960 | - |
| This work - Virtex-5 | | | | |
| *ReconfigurableArchitecture*0 | 2719 SL-FF, 9 BRAM | 153.18 | 1.894 | 0.696 Mbps/SL-FF |
| *ReconfigurableArchitecture*1 | 5559 SL-FF, 9 BRAM | 124.11 | 1.535 | 0.276 Mbps/SL-FF |
| *ReconfigurableArchitecture*2 | 10172 SL-FF, 18 BRAM | 62.14 | 0.768 | 0.074 Mbps/SL-FF |
| *ReconfigurableArchitecture*3 | 5584 SL-FF, 9 BRAM | 94.20 | 1.162 | 0.185 Mbps/SL-FF |

Table 7.7: Implementation results of the hardware architectures for the IEEE 802.16e-2005 networks

| Work-device | FPGA resources | Clock (MHz) | Throughput (Gbps) | Efficiency |
|---|---|---|---|---|
| [Aziz et al., 2005] - Spartan-3 | 523 | 63.70 | 0.127 | 0.243 Mbps/Slices |
| [Shim et al., 2004] - VirtexE | 3750 | 50.00 | 0.243 | 0.064 Mbps/Slices |
| [Smyth et al., 2006] - Virtex-II | 3474, 15 BRAM | 80.30 | 0.275 | 0.079 Mbps/Slices |
| [Bae et al., 2006] - Stratix | 5605 | 50.00 | 0.258 | 0.046 Gbps/logic cells |
| [Quatech-Inc., 2007] - ASIC module | - | - | 0.054 | - |
| [Hi/fn-Inc., 2008] - Processor | - | 66.00 | 0.275 | - |
| [RadiSys-Corporation, 2006] - Processor | - | $> 1500.00$ | 2.000 | - |
| This work - Virtex-5 | | | | |
| $ReconfigurableArchitecture0$ | 2154, 9 BRAM | 133.31 | 1.647 | 0.680 Mbps/SL-FF |
| $ReconfigurableArchitecture1$ | 8491, 20 BRAM | 90.82 | 1.215 | 0.143 Mbps/SL-FF |
| $ReconfigurableArchitecture2$ | 9413, 18 BRAM | 72.89 | 0.900 | 0.094 Mbps/SL-FF |
| $ReconfigurableArchitecture3$ | 8565, 20 BRAM | 65.12 | 0.804 | 0.091 Mbps/SL-FF |

## 7.5 Discussion

The design methodology of the reconfigurable hardware allows the development of architectures based on different types of reconfiguration, considering that high-performance hardware modules have been previously designed.

Also, highly-parallelized modules were compared against modules with a common element. Their performances were revised and analyzed on the different proposed reconfigurable architectures, showing that in most of the cases, the second ones report better throughput and efficiency. Although the highly-parallelized modules are designed to improve the performance.

Reconfigurable architectures affect the performance of the hardware implementations due to the larger critical path that can be generated, which in certain cases have a double of the size ($Architecture0$ versus $Architecture3$). The presented analysis of the throughput and efficiency on the reconfigurable architectures allows to decide new branches to increase their performance, and to highlight key points to design reconfigurable architectures focused on decreasing their critical path.

# Chapter 8

# Conclusions and Contributions

## 8.1 Conclusions

During this research, it was emphasized that the software radios focus on lower layers (hardware) and security services have better performance and higher levels of security when they are implemented in hardware. There is a great amount of architectures being developed, and it is necessary to offer flexible systems that operate in the different types of networks. A reconfigurable architecture is considered to be a key element for these research works to offer several advantages due to its characteristics such as high performance and flexibility. In this work, different types of reconfiguration are analyzed to propose architectures that provide high performance as long as they can be part of any type of software radio and provide security services for two communication networks.

To guarantee a right application of the reconfigurable schemes, this research is based on secure protocols already established and revising their security standards. The flexibility of these schemes is evaluated and presented to operate in two different types of networks, moreover, each configuration enables to change between two different cryptographic modules. Final implementation results carry out ideas to design and develop a more general software-radio platform, which operates in multiple wireless networks.

The design methodology for the reconfigurable architectures and it associated analysis were focused on providing a reconfigurable architecture featuring high hardware implementation efficiency, which is achieved by reducing critical path time of the cryptographic hardware architectures, and by

trade-off studies (of hardware resources, throughput and reconfiguration) to evaluate different reconfiguration schemes. An important point is that the reconfiguration time does not affect the functionality of a complete system, because the selected types of networks need establishing security association that provides sufficient time to change the operation of the proposed reconfigurable architectures. Implementation results are showed for comparisons and validation, which enable to decide that security reconfigurable architectures for software-radio operations are capable to work in different networks and to offer security services, reporting a high efficiency that allows high data transmissions required in modern applications.

The revision of the wireless communication networks and the selection, simulation and studies of the security architectures, which are based on AES algorithm in the CCM mode, allows to propose the design methodology of the hardware security architectures. This methodology focuses on improving the throughput and efficiency aiming to provide short critical paths and using few hardware resources, taking advantage of the parallelization and modular specialization.

Results about the particular implementations of these cryptographic architectures (for AES and AES-CCM algorithms) and security hardware architectures (for the IEEE 802.11i-2004 and IEEE 802.16e-2005 security architectures) compared against the related works show that the proposed architectures report high performance. The throughput of the hardware architectures is improved by taking advantage of the hardware characteristics such as parallelization and specialized modules. These proposed hardware security architectures enable to test the reconfiguration scheme, and their hardware implementations are part of the configuration library of the software-radio platform.

In general, the obtained results show an improvement in performance by using configurations with a common module compared against the configurations with highly-parallelized structures. Technique of parallelization should help to increase the performance, but there are other elements that should be taken into account, because using more hardware resources in the highly-parallelized structures for placement and routing can provoke larger critical paths, decreasing the throughput. Furthermore, since a greater amount of resources is required and a decreased throughput is reported, the efficiency is smaller.

Finally, the combinational and sequential elements of reconfigurable architectures should be balanced with the aim of reducing their critical path.

The balanced hardware security architectures supported in the reconfigurable architecture 0 report a high performance, while reconfigurable architectures 1-3 report a decreased throughput and efficiency. These characteristics are affected by the distribution and implementation on the reconfigurable structures. The integral design and development that are considered to reduce the critical path when architectures are mapped on the reconfigurable structures can help to improve their performance.

## 8.2 Contributions

This research is aimed to develop a reconfiguration architecture for a system architecture with a set of customizable modules. There are different types of reconfiguration so that several reconfigurable architectures be analyzed and evaluated. These architectures satisfy the MAC security processing of two different wireless network standards, providing a security solution to multi-standard wireless environment and software radio concept. The following contributions are reached:

1. A methodology to design reconfigurable hardware architectures for security operations considering the software radio concept. This methodology will enable to design reconfigurable hardware architectures with high performance, high throughput and low hardware resources that run over a flexible reconfigurable platform.

2. An SR platform for the MAC processing, this enables to operate in different networks and to secure the data transmissions. The security architectures of the important wireless communication networks are recommended and standardized on the MAC sub-layer. The proposed platform may act as a complement of an SR system that provides security for different network wireless standards in this sub-layer.

3. Design and development of high-efficiency specialized hardware architectures, which execute security schemes of the modern communication protocols. Trade-off studies between throughput and usage of hardware resources are key elements to design and implement cryptographic hardware architectures, which reports high performance.

4. Hardware design methodologies to provide cryptographic hardware architectures for the security architecture IEEE 802.11i-2004 and IEEE

802.16e-2005 and AES-CCM algorithms. These hardware architectures for security protocols are based on modern algorithms and operation modes, which offer a high level of security, and proposed cryptographic hardware implementations are reported.

5. A flexible reconfigurable system architecture. The platform should support the addition of new security protocols as well as additional cryptographic operations for existing protocols.

6. A complete software radio for applications such as the cognitive radios or software-defined radio, where these reconfigurable schemes can be totally applied.

## 8.3 Future Work

Future work focuses on the reconfigurable architecture to provide high throughput and efficiency. To achieve that, the following activities can be explored:

1. Designing and developing of a self-reconfigurable architecture, see Fig. 8.1, evaluating throughput and efficiency, and analyzing and comparing the configurations with common-module against the configurations with highly-parallelized structure.

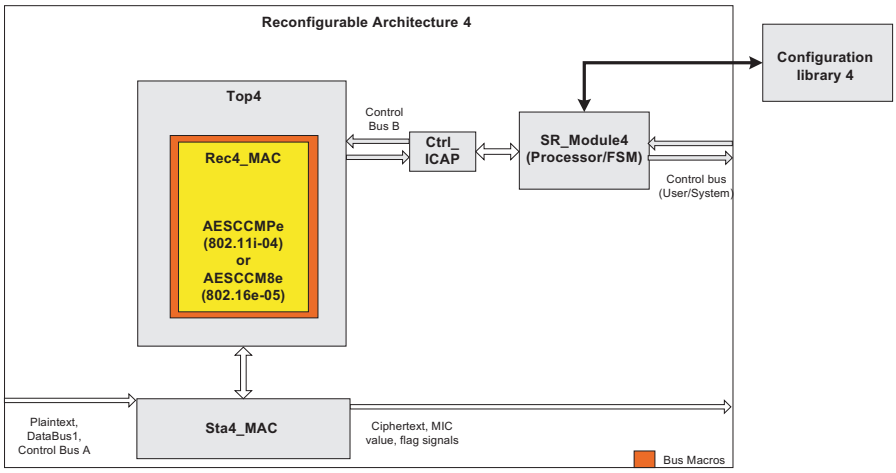


Figure 8.1: Block diagram of the self-reconfigurable architecture

2. Adding elements to the design methodology for the reconfigurable architectures that allow to improve the throughput and efficiency, considering the reconfigurable scheme, so that the critical path can be reduced.

3. Analyzing and proposing new enhancements that improve the performance of the proposed reconfigurable architectures in this work. For example, the critical path of the reconfigurable architecture 3 can be decreased by adding sequential elements in the static part.

4. Evaluating other structures and distributions for the area assignation of the proposed reconfigurable architectures focused on the high throughput.

5. Proposing other hardware security architectures of different standards, to design the modules of other cryptographic algorithms. Obtaining the configurations of the implementations, which enables to support them on the diverse proposed reconfigurable architectures.

# Bibliography

D. P. Agrawal. Ubiquitous mobility and multi-service in 3g and beyond. *The Third International Conference on COMmunication System softWAre and middlewaRE (COMSWARE 2008)*, January 2008.

Algotronix-Ltd. Aes core product description. *Datasheet. Available at: http://www.algotronix.com/*, November 2004.

I. Algredo-Badillo. Arquitectura reconfigurable para la implementación de algoritmos estándares de criptografía aplicados a comunicaciones. Master's thesis, Instituto Nacional de Astrofísica, Óptica y Electrónica, October 2004.

Altera-Corporation. Accelerating wimax system design with fpgas. *Altera Corporation. White Paper*, October 2004.

A. Aziz and N. Ikram. An fpga-based aes-ccm crypto core for ieee 802.11i architecture. *International Journal of Networks Security*, 5(2):224–232, 2007.

A. Aziz, A. Samiah, and N. Ikram. A secure framework for robust secure wireless network (rsn) using aes-ccmp. *4th International Bhurban Conference on Applied Sciences and Technology*, June 2005.

D. Bae, G. Kim, J. Kim, S. Park, and O. Song. An efficient design of ccmp for robust security network. *ICISC 2005, Lecture Notes in Computer Science 3935, Springer-Berlin*, 3935(325–361), 2006.

Barco-Silex. Aes encryption and decryption ba411aes factsheet. *Datasheet. Available at: http://www.barco.com/*, 2005.

L. Berlemann, R. Pabst, and B. Walke. Multimode communication protocols enabling reconfigurable radios. *EURASIP Journal on Wireless Communications and Networking 2005*, pages 390–400, 2005.

G. Bertoni, J. Guajardo, and C. Paar. Architectures for advanced cryptographic systems. *Idea Group Inc*, 2004.

N. S. Bhatia. A physical layer implementation of reconfigurable radio. Master's thesis, Virginia Polytechnic Institute, 2004.

B. Bloget and B. James-Roxby. A self-reconfiguring platform. *Research Seminar on Reconfigurable Hardware*, 2003.

H. Bogucka, A. Polydoros, and G. Razzano. Wf- a reconfigurable radio system on the path to sdr. *SDR Forum Technical Conference*, November 2002.

D. Boppana. Fpga-based wimax system design. *FPGA and Structured ASIC Journal. Available at: http://www.altera.com/literature/cp/cp gspx wimax.pdf*, October 2005.

R. Brodersen, C. Chang, J. Wawrzynek, and D. Werthimer. Bee2: a multipurpose computing platform for radio telescope signal processing applications. *International SKA Conference 2004 (ISKA'04)*, July 2004.

P. Bucknell. Software radio and reconfiguration. *London Communications Symposium. The Annual London Conference on Communication*, 2000.

J. L. Burbank and W. T. Kasch. An ieee 802.16/802.11 hybrid tan architecture for the next-generation nas. *NASA ICNS Conference and Workshop*, May 2006.

F. Carpenter, S. Srikanteswara, and A. Brown. Software defined radio test bed for integrated communications and navigation applications. *Proceedings of ION GNSS 2005*, September 2005.

J. R. Cavallaro and P. Radosavljevic. Asip architecture for future wireless systems: Flexibility and customization. *Wireless World Research Forum (WWRF)*, 2004.

R. Chaves, G. K. Kuzmanov, S. Vassiliadis, and L. A. Sousa. Reconfigurable memory based aes co-processor. *International Parallel and Distributed Processing Symposium 2006 (IPDPS 2006), IEEE Computer*, pages 446–455, 2006.

P. Cheung. Wlan on cellular platforms: Freescale lp170x wlan solution. *2006 Freescale Semiconductor Inc, Freescale Technology Forum 2006 (FTF 2006). Available at: http://www.freescale.com/files/abstract/overview/FTF2006 PM111.pdf*, May 2006.

T. C. Clancy. *Dynamic Spectrum Access in Cognitive Radio Networks*. PhD thesis, University of Maryland, 2006.

Elliptic-Semicon ductor Inc. Clp-20 high performance aes-ccm core. *Datasheet. Available at: www.ellipticsemi.com*, 2008.

M. Dworkin. Nist special publication 800-38c, recommendation for block cipher modes of operation: The ccm mode for authentication and confidentiality. *National Institute of Standards and Technology (NIST)*, 2004.

M. Dworkin. Recommendation for block cipher modes of operation: Methods and techniques. *National Institute of Standards and Technology (NIST), NIST Special Publication 800-38*, December 2001.

A. J. Elbirt. *Reconfigurable Computing for Symmetric-key Algorithms*. PhD thesis, Worcester Polytechnic Institute, April 2002.

Elliptic-Semiconductor-Inc. Llp-02: Pdu processor for 802.16/wimax. *Datasheet. Available at: www.ellipticsemi.com*, 2008.

FIPS-197. Announcing the advanced encryption standard (aes). *Federal Information Processing Standards Publication*, November 2001.

Inc. Fujitsu Microelectronics America. Mb86k21: The fujitsu 802.16e-2005 mobile wimax soc. *Product Brief. Available at: http://us.fujitsu.com/micro*, 2007.

K. Gaj and P. Chodowiec. Comparison of the hardware performance of the aes candidates using reconfigurable hardware. *Proceedings of the 3rd Advanced Encryption Standard (AES) Candidate Conference*, April 2000.

C. Gehrmann and P. Sthl. Mobile platform security: Ericsson review no.2. *Sony Ericsson Mobile Communications*, 2006.

Y. J. Guo. Ubiquitous gigabit wireless networks. *2nd International Conference on Wireless Broadband and Ultra Wideband Communications (AusWireless '07)*, 2007.

S. Hekmat. *Communication Networks*. Available at: www.pragsoft.com, digital edition, 2005.

Hi/fn-Inc. 7955/7956: The wimax security processor. *Datasheet. Available at: www.hifn.com*, 2005.

Hi/fn-Inc. Hifn-class security for robo/sme applications: Applied services processors - 7954/7955/7956. *Datasheet. Available at: www.hifn.com*, 2008.

A. Hodjat and I. Verbauwhede. A 21.54 gbits/s fully pipelined aes processor on fpga. *IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2004.

ICT-Centre. Multi gigabit millimetre wave wireless. *Innovative ICT transforming Australian industries*, 2008.

IPCores-Products-Inc. Ccm6 802.16e (wimax) aes core. *Datasheet. Available at: www.ipcores.com/IEEE802.16e_CCMCore.htm*, 2006.

R. Jackson, S. Hettiaratchi, M. Fitton, and S. Perry. Reconfigurable radio with fpga-based application-specific processors. *2004 Software Defined Radio Technical Conference and Product Exposition (SDR'04)*, November 2004.

Jetstream-Media-Technologies. Jetccm-6: 802.16e wimax aes-ccm core. *Datasheet. Available at: www.security-cores.com*, October 2006.

G. C. Kessler. An overview of cryptography. *Electronic Book. Available at: http://www.garykessler.net/library/crypto.htm*, May 1998.

H. W. Kim and S. Lee. Design and implementation of a private and public key crypto processor and its application to a security system. *IEEE Transactions on Consumer Electronics*, 50(1):214–224, 2004.

P. Kitsos. Hardware implementations for the iso/iec 18033-4:2005 standard for stream ciphers. *International Journal of Signal Processing*, 3(1):66–73, 2006. ISSN 1304-4478.

P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi. Security as a new dimension in embedded system design. *DAC 2004*, June 2004.

M. Komara. Sdr architecture ideally suited for evolving 802.16 wimax standards. *Airnet Communications Corporation, Datasheet. Available at: www.airnetcom.com*, 2004.

LAN/MAN-Standards-Committee. Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Computer Society*, 1999.

LAN/MAN-Standards-Committee. Part 16: Air interface for fixed and mobile broadband wireless access systems. *IEEE Computer Society and the IEEE Microwave Theory and Techniques Society*, February 2005.

LAN/MAN-Standards-Committee. Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Computer Society*, July 2004.

LAN/MAN-Standards-Committee. Part 15.1: Wireless medium access control (mac) and physical layer (phy) specifications for wireless personal area networks (wpans). *IEEE Computer Society*, 2005.

Y. Li, T. Callahan, E. Darnell, R. Harr, U. Kurkure, and J. Stockwood. Hardware-software co-design of embedded reconfigurable architectures. *ACM*, 2000.

Helion Technology Limited. Overview datasheet - helion cores. *Datasheet. Available at: www.heliontech.com/*, 2004.

Y. Lin, H. Lee, M. Woh, Y. Harel, S. A. Mahlke, T. N. Mudge, C. Chakrabarti, and K. Flautner. Soda: A low-power architecture for software radio. *33rd International Symposium on Computer Architecture (ISCA '06)*, 2006.

T. Liu, C. Tanougast, P. Brunet, Y. Berviller, H. Rabah, and S. Weber. An optimized fpga implementation of an aes algorithm for embedded applications. *International Workshop on Applied Reconfigurable Computing 2005 (ARC2005)*, February 2005.

E. López-Trejo, F. Rodríguez-Henríquez, and A. Díaz-Pérez. An efficient fpga implementation of ccm using aes. *The 8th International Conference on Information Security and Cryptology (ICISC'05), Lecture Notes in Computer Science 3935, pp.208-215, Springer-Verlag 2005*, (208–215), 2005.

J. Lu and J. Lockwood. Ipsec implementation on xilinx virtex-ii pro fpga and its application. *Reconfigurables Architectures Workshop (RAW)*, April 2005.

V. Manral. Cryptographic algorithm implementation requirements for encapsulating security payload (esp) and authentication header (ah). *Request for Comments 4835, Network Working Group*, 2007.

J. McCaffrey. You're your data secure with the new advanced encryption standard. *MSDN Magazine. Available at: http://msdn.microsoft.com*, November 2003.

S. Medina, E. Astaiza, and P. Vera. Reconfigurable satellite payload model based on software radio technology. *3rd IEEE International Congress of the Andean Region (ANDESCON2006)*, 2006.

A. Menezes, P. V. Oorschot, and S. Vanstone. Handbook of applied cryptography. *CRC Press*, 1996.

National-Instruments-Corporation. Designing next generation test systems: An in-depth developers guide. 2006.

A. Nilsson. *Design of Programmable Multi-Standard Baseband Processors*. PhD thesis, Linkping University, 2007.

H. Ollikainen. Routing security - an overview. *Telecommunications Software and Multimedia Laboratory*, 2004.

C. Paar. Reconfigurable hardware in modern cryptography. *4th Workshop on Elliptic Curve Cryptography (ECC 2000)*, 2000.

D. Panigrahi, C. N. Taylor, and S. Dey. A hardware/software reconfigurable architecture for adaptive wireless image communication. *Proceedings of the 2002 VLSI Design Conference/ASP-DAC*, 2002.

A. Pashtan. Wireless terrestrial communications: Cellular telephony. *Telecommunication Systems and Technologies, Encyclopedia of Life Support Systems (EOLSS), Electronic Book. Available at: http://www.eolss.net/ebooks/home.aspx*, 2006.

J. Polson. Cognitive radio applications in sofwtare defined radio. *SDR'04 Technical Conference and Product Expositions, SDR Forum*, 2004.

Quatech-Inc. Airbone: Embedded radio modules (802.11b/g). *Datasheet. Available at: www.quatech.com*, November 2007.

L. Quinn, P. Mehta, and A. Sicher. Wireless communications technology landscape. *White Paper, Dell*, February 2005.

RadiSys-Corporation. Promentum amc-8201/2: Mpc8641d powerpc advanced mezzanine card. *Datasheet. Available at: www.radisys.com*, 2006.

P. Ryser. Software defined radio with reconfigurable hardware and software: A framework for a tv broadcast receiver. *Embedded Systems Conference 2005*, 2005.

J. H. Shim, T. W. Kwon, D. W. Kim, J. H. Suk, Y. H. Choi, and J. R. Choi. Compatible design of ccmp and ocb aes cipher using separated encryptor and decryptor for ieee 802.11i. *Proceedings of the International Symposium on Circuits and Systems (ISCAS '04)*, 3(645–8), 2004.

C. Sivakumar and A. Velmurugan. High speed vlsi design ccmp aes cipher for wlan (ieee 802.11i). *International Conference on Signal Processing, Communications and Networking (IEEE-ICSCN 2007)*, (22–24), 2007.

N. Sklavos, G. Selimis, and O. Koufopavlou. Fpga implementation cost and performance evaluation of ieee 802.11 protocol encryption security schemes. *Second Conference on Microelectronics, Microsystems and Nanotechnology, Journal of Physics: Conference Series 10 (2005), Institute of Physics Publishing Ltd, Doi 10.1088/1742-6596/10/1/088*, pages 361–364, 2005.

N. Smyth, M. McLoone, and J. V. McCanny. Wlan security processor. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 53(7):1506–1520, 2006. ISSN 1057-7122.

R. R. Taylor and S. C. Goldstein. A high-performance flexible architecture for cryptography. *Cryptographic Hardware and Embedded Systems 1999 (CHES'99)*, August 1999.

Telecommunications-Technology-Association. Proposed working methods on phase 1 collaboration. *The 7th CJK B3G Standards Meeting, Beijing, China*, December 2004.

E. Tell. *Design of Programmable Baseband Processors*. PhD thesis, Linkping University, 2005.

K. Tikkanen, M. Hnnikinen, T. Hmlinen, and J. Saarinen. Advanced prototype platform for a wireless multimedia local area network. *10th European Signal Processing Conference (EUSIPCO2000)*, pages 2309–2312, September 2000.

T. Tuan, S. F. Li, and J. Rabaey. Reconfigurable platform design for wireless protocol processor. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2001.

G. Umamaheshwari and A. Shanmugan. Efficient vlsi implementation of the block cipher rijndael algorithm. *Academic Open Internet Journal. Available at: http://www.acadjournal.com/*, 12, 2004.

B. H. Walke. On the importance of wlans for 3g cellular radio to become a success. *Proceedings 10th Aachen Symposium on Signal Theory 9/2001*, pages 13–24, September 2001.

A. J. Weissberger. Ieee 802.22 wireless regional area network (wran). March 2005.

M. Wouters, G. Vanwijnsberghe, P. Van Wesemaeland T. Huybrechts, and S. Thoen. Real time implementation on fpga of an ofdm basedwireless lan modem extended with adaptive loading. In *Proceedings of the 28th European Solid-State Circuits Conference 2002 (ESSCIRC 2002)*, pages 531–534, September 2002. ISBN 2-86332-180-3.

L. Wu, C. Weaver, and T. Austin. Cryptomaniac: A fast flexible architecture for secure communication. *28th Annual International Symposium on Computer Architecture (ISCA-2001)*, June 2001.

Inc Xilinx. Early access partial reconfiguration user guide: For ise 8.1.01i. *User Guide UG208, Xilinx, Inc. Available at: www.xilinx.com*, March 2006.

W. Zwart, J. Eilers, G. Gaydadjiev, and S. Cotofana. Damp - delft altera-based multimedia platform. In *Proceedings of the Program for Research on Integrated Systems and Circuits 2002 (ProRISC 2002)*, pages 587–594, November 2002. ISBN 90-73461-33-2.