



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΗΛΕΚΤΡΟΝΙΚΗΣ
ΠΙΣ ΗΛΕΚΤΡΟΝΙΚΗ & ΕΠΙΚΟΙΝΩΝΙΕΣ

ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΚΑΙ FPGA ΥΛΟΠΟΙΗΣΕΙΣ ΤΟΥ ΠΡΟΤΥΠΟΥ ΚΡΥΠΤΟΓΡΑΦΙΑΣ P1619

ΕΙΔΙΚΗ ΕΡΕΥΝΗΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΣΑΚΟΥΛΗΣ ΘΑΝΑΣΗΣ

A.M. 460

Επιβλέπων: Επίκ. Καθηγητής, Γ. Θεοδωρίδης

Πάτρα, Νοέμβριος 2015

ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΚΑΙ FPGA ΥΛΟΠΟΙΗΣΕΙΣ ΤΟΥ ΠΡΟΤΥΠΟΥ ΚΡΥΠΤΟΓΡΑΦΙΑΣ P1619

ΕΙΔΙΚΗ ΕΡΕΥΝΗΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΣΑΚΟΥΛΗΣ ΘΑΝΑΣΗΣ

Εξεταστική επιτροπή : Γ. Θεοδωρίδης

Κ. Ψυχάλινος

Ο. Κουφοπαύλου

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 2 Νοεμβρίου 2013.

Γ. Θεοδωρίδης
Επίκ. Καθηγητής

Κ. Ψυχάλινος
Καθηγητής

Ο. Κουφοπαύλου
Καθηγητής

Πάτρα, Νοέμβριος 2015

Περίληψη

Στη σημερινή εποχή, με την εξέλιξη της τεχνολογίας, η ανάγκη για προστασία των προσωπικών μας δεδομένων έχει γίνει επιτακτική. Μια σειρά από ευαίσθητες πληροφορίες είναι αποθηκευμένες στους υπολογιστές μας όπως, προσωπικές φωτογραφίες, σημαντικά αρχεία, μελλοντικά σχέδια κ.α. Αυτές οι πληροφορίες θα πρέπει να διασφαλιστούν τόσο από απλές επιθέσεις αλλά ακόμη και στην ακραία περίπτωση όπου ένας μη εξουσιοδοτημένος χρήστης αποκτήσει τα προσωπικά μας δεδομένα.

Το πρόβλημα αυτό καλείται να λύσει το πρότυπο P1619. Το 2007 ο SISWG (Security in Storage Working Group) πρότεινε το πρότυπο P1619 το οποίο εγκρίθηκε από την IEEE το ίδιο έτος. Το πρότυπο P1619 θεωρείται κατάλληλο για κρυπτογράφηση δεδομένων σε sector-based αποθηκευτικά μέσα. Βασική μονάδα του P1619 αποτελεί ο αλγόριθμος AES ο οποίος καλείται να διεκπεραιώνει τις διαδικασίες κρυπτογράφησης και αποκρυπτογράφησης. Η κατάσταση λειτουργίας που έχει επιλεγεί είναι το ECB (Electronic CodeBook). Βασικό μειονέκτημα της κατάστασης αυτής είναι ότι για ένα ίδιο plaintext παράγεται το ίδιο ciphertext. Αυτό το μειονέκτημα καλύπτεται από το P1619 το οποίο ενσωματώνει στη κρυπτογράφηση τη λογική θέση των δεδομένων. Ακόμη, μια μονάδα πολλαπλασιασμού GF(Galois Field (2^{128})) είναι απαραίτητη ούτως ώστε να μην παραχθεί ποτέ το ίδιο ciphertext. Ο αλγόριθμος που χρησιμοποιεί το P1619 ονομάζεται XTS-AES και διακρίνεται ανάλογα με το κλειδί κρυπτογράφησης που έχει επιλεγεί, XTS-AES-128 για κλειδί 128 bits και XTS-AES-256 για κλειδί 256 bits.

Σε αυτήν τη διπλωματική εργασία ασχολούμαστε με τον XTS-AES ο οποίος εφαρμόζεται σε μπλοκ δεδομένα σταθερού μήκους (128-bits). Αρχικά, κάνουμε μια διερεύνηση του προτύπου P1619 σύμφωνα με ένα συγκεκριμένο χρονοπρογραμματισμό που ακολουθείται και στον οποίο βασίζονται οι αρχιτεκτονικές που υλοποιούνται. Στη συνέχεια παρουσιάζονται τα μειονεκτήματα αυτής της προσέγγισης και προτείνεται ένας διαφορετικός χρονοπρογραμματισμός στον οποίο βασίζονται και οι νέες αρχιτεκτονικές. Ακόμη, πραγματοποιείται μια σύγκριση, μεταξύ των αντίστοιχων υλοποιημένων αρχιτεκτονικών, η οποία θεωρείται αναγκαία ούτως ώστε να διαπιστωθεί ποιος χρονοπρογραμματισμός παράγει καλύτερα αποτελέσματα.

Στο 1^ο κεφάλαιο αυτής της διπλωματικής εργασίας εξηγούνται βασικές έννοιες κρυπτογραφίας και κρυπτογράφησης δίσκου. Στο 2^ο κεφάλαιο περιγράφεται αναλυτικά το πρότυπο P1619 ενώ στο 3^ο κεφάλαιο γίνεται αναλυτική περιγραφή των χρονοπρογραμματισμών και των προτεινόμενων αρχιτεκτονικών. Στη συνέχεια στο κεφάλαιο 4 φαίνονται όλες οι προσομοιώσεις από όλες τις αρχιτεκτονικές και στο κεφάλαιο 5 παρουσιάζονται τα αποτελέσματα και γίνεται η απαραίτητη σύγκριση μεταξύ των δύο χρονοπρογραμματισμών. Τέλος στο κεφάλαιο 6 παρουσιάζονται τα συμπεράσματα από τη διπλωματική εργασία και οι προτάσεις για μελλοντική έρευνα.

Λέξεις κλειδιά: P1619, XTS-AES, Κρυπτογραφία, Κρυπτογράφηση δίσκου, FPGA

Abstract

Nowadays, with the development of technology, the need for protection of our personal data has become imperative. A number of sensitive information are stored on our computers such as personal photos, important files, future plans etc. This information must be safeguarded both from simple attacks, but even in the extreme case where an unauthorized user obtains our personal data.

This problem is called to solve by the P1619 standard. In 2007, SISWG (Security in Storage Working Group) proposed the P1619 standard which was approved by the IEEE in the same year. The P1619 standard is considered suitable for data encryption in sector-based storage media devices. Basic unit of this standard is the AES algorithm which is mainly handled the processes of encryption/decryption. The operation mode that has been chosen is the ECB (Electronic CodeBook). A key disadvantage of this operation mode is that if the same plaintext is encrypted twice or more times, the same ciphertext will be generated. This drawback is solved by the P1619 which incorporates the logical block address in the encryption/decryption process. Furthermore, a GF (*Galois Field* (2^{128})) multiplication unit is necessary so as never being produced the same ciphertext. The algorithm that is being used by the P1619 standard is called XTS-AES and is distinguished depending on the encryption that has been selected, XTS-AES-128 means that the key length is 128 bits and XTS-AES-256 for key length 256 bits respectively.

In this master thesis, we deal with the XTS-AES which is applied in fixed-length data blocks (128 bits). First, we investigate the P1619 standard, following a specific scheduling, which is adopted in many works, and implement the respective architectures. Subsequently, a different scheduling approach is presented and new architectures, that are based on this scheduling, are implemented. Furthermore, a comparison is being made between the respective implemented architectures in order to establish the scheduling which produce better results.

In first chapter of this master thesis, are explained basic concepts of cryptography such as modes of operation, block ciphers, disk encryption etc. In second chapter, the P1619 standard is explained analytically while in the third chapter, the two types of scheduling and the proposed architectures are presented. Then, in fourth chapter, simulations from all the implemented architectures are showed thoroughly and in chapter five, are presented the results of this master thesis and the comparison between the two types of scheduling is being made. Finally, in sixth chapter, we discuss about the conclusions and future research proposals are presented.

Key Words: P1619, XTS-AES, Cryptography, Disk encryption, FPGA

Ευχαριστίες

Η παρούσα μεταπτυχιακή διπλωματική εργασία εκπονήθηκε στα πλαίσια του ΠΜΣ Ηλεκτρονική και Επικοινωνίες (Ραδιοηλεκτρολογία) που διοργανώνεται από το τμήμα Φυσικής του Πανεπιστημίου Πατρών.

Θα ήθελα να ευχαριστήσω το κ. Γεώργιο Θεοδωρίδη, επιβλέπων καθηγητή αυτής της εργασίας, που μου έδωσε την ευκαιρία να μάθω πολύ σημαντικά πράγματα σχετικά με τον χώρο της ηλεκτρονικής και συγκεκριμένα των Ψηφιακών Κυκλωμάτων. Ακόμη, θα ήθελα να ευχαριστήσω το κ. Αθανάσιο Κακαρούνα, για την εξαιρετική συνεργασία που είχαμε κατά την υλοποίηση αυτής της προσπάθειας. Ακόμη, θα ήθελα να ευχαριστήσω το κ. Κωνσταντίνο Ψυχάλινο, που με έχει βοηθήσει και συμβουλέψει όσο κανένας σε όλη τη διάρκεια των σπουδών μου. Επίσης, ευχαριστώ τον κ. Οδυσσέα Κουφοπαύλου που μου έκανε τη τιμή να είναι μέλος της τριμελούς επιτροπής.

Ακόμη, είμαι ευγνώμον σε όλους του καθηγητές μου επειδή όλοι τους μου δίδαξαν και από κάτι σημαντικό. Επίσης, θέλω να ευχαριστήσω όλους τους υποψήφιους διδάκτορες του τομέα καθώς με βοήθησαν σημαντικά όποτε τους χρειάστηκα.

Επίσης, θέλω να ευχαριστήσω τις «Υποτροφίες Ανδρέας Μεντζελόπουλος του Πανεπιστημίου Πατρών» που με βοήθησαν να ολοκληρώσω επιτυχώς τις σπουδές μου.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου η οποία όλα αυτά τα χρόνια με στηρίζει και με βοηθά στις σπουδές μου.

Πάτρα, 2015

*Η παρούσα εργασία υποστηρίχθηκε οικονομικά από τις
«Υποτροφίες Ανδρέας Μεντζελόπουλος του Πανεπιστημίου Πατρών»*

Πίνακας περιεχομένων

Περίληψη	iii
Abstract	v
Ευχαριστίες	vii
Λίστα σχημάτων	xii
Λίστα Πινάκων	xvii
1. Εισαγωγή – Κρυπτογραφία	1
1.1 Κρυπτογραφία	2
1.1.1 Βασικές έννοιες	2
1.1.2 Βασικές λειτουργίες	4
1.1.3 Αλγόριθμοι και κλειδιά	5
1.2 Κρυπτογράφηση δίσκου	9
1.2.1 Πλήρης κρυπτογράφηση δίσκου	10
1.2.2 Κρυπτογράφηση συστημάτων αρχείων	10
1.2.3 Full disk encryption VS File-system-level encryption	11
1.3 Υλοποίηση σε υλικό έναντι λογισμικού	11
2. Το πρότυπο κρυπτογράφησης IEEE P1619	13
2.1 Ο αλγόριθμος AES	14
2.1.1 Μαθηματικό υπόβαθρο	14
2.1.2 Η κρυπτογράφηση	15
2.1.3 Η αποκρυπτογράφηση	18
2.1.4 Key expansion routine	21
2.2 Πολλαπλασιασμός με το πρωταρχικό στοιχείο α	23
2.3 Η XTS-AES κρυπτογράφηση	24
2.4 Η XTS-AES αποκρυπτογράφηση	25
2.5 Εφαρμογή του XTS-AES	27
2.6 Υφιστάμενη κατάσταση του IEEE P1619	27
3. Χρονοπρογραμματισμός και προτεινόμενες αρχιτεκτονικές	29

3.1 Χρονοπρογραμματισμός	30
3.1.1 Υπάρχον χρονοπρογραμματισμός	30
3.1.2 Προτεινόμενος χρονοπρογραμματισμός	33
3.2 Προτεινόμενες αρχιτεκτονικές	35
3.2.1 Αρχιτεκτονική δύο πυρήνων	35
3.2.2 Αρχιτεκτονική τριών πυρήνων	42
3.2.3 Αρχιτεκτονική τεσσάρων πυρήνων	45
4. Προσομοιώσεις.....	49
4.1 Προσομοίωση αρχιτεκτονικής δύο πυρήνων	50
4.1.1 Προσομοίωση κρυπτογράφησης XTS-AES-128	51
4.1.2 Προσομοίωσης αποκρυπτογράφησης XTS-AES- 128.....	53
4.1.3 Προσομοίωση κρυπτογράφησης XTS-AES-256	54
4.1.4 Προσομοίωση αποκρυπτογράφησης XTS-AES-256	55
4.2 Προσομοίωση αρχιτεκτονικής τριών πυρήνων	56
4.2.1 Προσομοίωση κρυπτογράφησης XTS-AES-128.....	56
4.2.2 Προσομοίωση αποκρυπτογράφησης XTS-AES-128	58
4.2.3 Προσομοίωση κρυπτογράφησης XTS-AES-256	59
4.2.4 Προσομοίωση αποκρυπτογράφησης XTS-AES-256.....	60
4.3 Προσομοίωση αρχιτεκτονικής τεσσάρων πυρήνων.....	62
4.3.1 Προσομοίωση κρυπτογράφησης XTS-AES-128.....	62
4.3.2 Προσομοίωση αποκρυπτογράφησης XTS-AES-128	64
4.3.3 Προσομοίωση κρυπτογράφησης XTS-AES-256	65
4.3.4 Προσομοίωση αποκρυπτογράφησης XTS-AES-256.....	67
5. Αποτελέσματα	69
6. Συμπεράσματα-Προτάσεις.....	83
Βιβλιογραφία.....	87

Λίστα σχημάτων

Σχήμα 1.1 Διαδικασία κρυπτογράφησης/αποκρυπτογράφησης.....	3
Σχήμα 1.2 Διαδικασία κρυπτογράφησης/αποκρυπτογράφησης με το ίδιο κλειδί.....	5
Σχήμα 1.3 Διαδικασία κρυπτογράφησης/αποκρυπτογράφησης με το διαφορετικό κλειδί.....	6
Σχήμα 1.4 Block cipher κρυπτογράφηση/αποκρυπτογράφηση.....	7
Σχήμα 2.1 Μετατροπή των bits εισόδου σε bytes.	15
Σχήμα 2.2 Μεταβάσεις από και προς το πίνακα <i>state</i>	15
Σχήμα 2.3 S-BOX του μετασχηματισμού SubBytes.	16
Σχήμα 2.4 Κυκλική ολίσθηση των τριών τελευταίων γραμμών του πίνακα <i>state</i>	17
Σχήμα 2.5 Μετασχηματισμός AddRoundKey	18
Σχήμα 2.6 Ο μετασχηματισμός InvShiftRows	19
Σχήμα 2.7 S-BOX του μετασχηματισμού InvSubBytes.	20
Σχήμα 2.8 Ψευδοκώδικας για τη ρουτίνα Key Expansion.....	22
Σχήμα 2.9 Είσοδοι και έξοδοι του πολλαπλασιασμού με το <i>a</i>	23
Σχήμα 2.10 Πολλαπλασιασμός με το πρωταρχικό στοιχείο <i>a</i>	23
Σχήμα 2.11 Διάγραμμα XTS-AES κρυπτογράφησης.....	25
Σχήμα 2.12 Διάγραμμα XTS-AES αποκρυπτογράφησης	26
Σχήμα 3.1 Γενική αρχιτεκτονική.....	36
Σχήμα 3.2 Μονάδα κρυπτογράφησης/αποκρυπτογράφησης.	37
Σχήμα 3.3 Μονάδα διαχείρισης κλειδιών	38
Σχήμα 3.4 Σχεδιάγραμμα της μονάδας ελέγχου για την αρχιτεκτονική δύο πυρήνων.....	39
Σχήμα 3.5 Μονάδα διαχείρισης <i>tweak values</i>	40
Σχήμα 3.6 Αρχιτεκτονική δύο πυρήνων	41
Σχήμα 3.7 Μονάδα διαχείρισης <i>tweak values</i>	43
Σχήμα 3.8 Σχεδιάγραμμα της μονάδας ελέγχου για την αρχιτεκτονική τριών πυρήνων	43
Σχήμα 3.9 Αρχιτεκτονική τριών πυρήνων	44
Σχήμα 3.10 Μονάδα διαχείρισης <i>tweak values</i>	45
Σχήμα 3.11 Αρχιτεκτονική τεσσάρων πυρήνων	46
Σχήμα 3.12 Σχεδιάγραμμα της μονάδας ελέγχου για την αρχιτεκτονική τεσσάρων πυρήνων.....	47
Σχήμα 4.1 Είσοδοι/Έξοδοι XTS-AES-2CORES.....	50
Σχήμα 4.2 Προσομοίωση κρυπτογράφησης XTS-AES-128 α' μέρος.	52
Σχήμα 4.3 Προσομοίωση κρυπτογράφησης XTS-AES-128 β' μέρος.	52
Σχήμα 4.4 Προσομοίωση κρυπτογράφησης XTS-AES-128 γ' μέρος.	52
Σχήμα 4.5 Προσομοίωση κρυπτογράφησης XTS-AES-128 δ' μέρος.	52
Σχήμα 4.6 Προσομοίωση κρυπτογράφησης XTS-AES-128 ε' μέρος.....	53
Σχήμα 4.7 Προσομοίωση κρυπτογράφησης XTS-AES-128 στ' μέρος.....	53

Σχήμα 4.8 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 α' μέρος.....	53
Σχήμα 4.9 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 β' μέρος.....	54
Σχήμα 4.10 Προσομοίωση κρυπτογράφησης XTS-AES_256 α' μέρος.....	54
Σχήμα 4.11 Προσομοίωση κρυπτογράφησης XTS-AES_256 β' μέρος.....	54
Σχήμα 4.12 Προσομοίωση κρυπτογράφησης XTS-AES_256 γ' μέρος.....	55
Σχήμα 4.13 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 α' μέρος.....	55
Σχήμα 4.14 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 β' μέρος.....	55
Σχήμα 4.15 Είσοδοι/έξοδοι του XTS-AES-3CORES.....	56
Σχήμα 4.16 Προσομοίωση κρυπτογράφησης XTS-AES-128 α' μέρος.....	57
Σχήμα 4.17 Προσομοίωση κρυπτογράφησης XTS-AES-128 β' μέρος.....	57
Σχήμα 4.18 Προσομοίωση κρυπτογράφησης XTS-AES-128 γ' μέρος.....	57
Σχήμα 4.19 Προσομοίωση κρυπτογράφησης XTS-AES-128 δ' μέρος.....	57
Σχήμα 4.20 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 α' μέρος.....	58
Σχήμα 4.21 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 β' μέρος.....	58
Σχήμα 4.22 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 γ' μέρος.....	58
Σχήμα 4.23 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 δ' μέρος.....	59
Σχήμα 4.24 Προσομοίωση κρυπτογράφησης XTS-AES-256 α' μέρος.....	59
Σχήμα 4.25 Προσομοίωση κρυπτογράφησης XTS-AES-256 β' μέρος.....	60
Σχήμα 4.26 Προσομοίωση κρυπτογράφησης XTS-AES-256 γ' μέρος.....	60
Σχήμα 4.27 Προσομοίωση κρυπτογράφησης XTS-AES-256 δ' μέρος.....	60
Σχήμα 4.28 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 α' μέρος.....	61
Σχήμα 4.29 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 β' μέρος.....	61
Σχήμα 4.30 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 γ' μέρος.....	61
Σχήμα 4.31 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 δ' μέρος.....	61
Σχήμα 4.32 Είσοδοι/Έξοδοι του XTS-AES-4CORES.....	62
Σχήμα 4.33 Προσομοίωση κρυπτογράφησης XTS-AES-128 α' μέρος.....	63
Σχήμα 4.34 Προσομοίωση κρυπτογράφησης XTS-AES-128 β' μέρος.....	63
Σχήμα 4.35 Προσομοίωση κρυπτογράφησης XTS-AES-128 γ' μέρος.....	63
Σχήμα 4.36 Προσομοίωση κρυπτογράφησης XTS-AES-128 δ' μέρος.....	63
Σχήμα 4.37 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 α' μέρος.....	64
Σχήμα 4.38 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 β' μέρος.....	64
Σχήμα 4.39 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 γ' μέρος.....	65
Σχήμα 4.40 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 δ' μέρος.....	65
Σχήμα 4.41 Προσομοίωση κρυπτογράφησης XTS-AES-256 α' μέρος.....	66
Σχήμα 4.42 Προσομοίωση κρυπτογράφησης XTS-AES-256 β' μέρος.....	66
Σχήμα 4.43 Προσομοίωση κρυπτογράφησης XTS-AES-256 γ' μέρος.....	66
Σχήμα 4.44 Προσομοίωση κρυπτογράφησης XTS-AES-256 δ' μέρος.....	66
Σχήμα 4.45 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 α' μέρος.....	67
Σχήμα 4.46 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 β' μέρος.....	67
Σχήμα 4.47 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 γ' μέρος.....	68
Σχήμα 4.48 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 δ' μέρος.....	68

Σχήμα 5.1 Throughput(Mbps), Virtex6	72
Σχήμα 5.2 Area(Slices), Virtex6.....	72
Σχήμα 5.3 Throughput/ Area(Mbps/Slice).....	73
Σχήμα 5.4 Throughput(Mbps)	75
Σχήμα 5.5 Throughput/ Area(Mbps/Slice).....	75
Σχήμα 5.6 Throughput(Mbps)	77
Σχήμα 5.7 Throughput/ Area(Mbps/Slice).....	78
Σχήμα 5.8 Throughput(Mbps), Virtex6	79
Σχήμα 5.9 Throughput/ Area(Mbps/Slice), Virtex6	80

Λίστα Πινάκων

Πίνακας 3.1 Υπάρχον χρονοπρογραμματισμός αρχιτεκτονικών 2, 3 και 4 πυρήνων αντίστοιχα.....	31
Πίνακας 3.2 Προτεινόμενος χρονοπρογραμματισμός αρχιτεκτονικών 2, 3 και 4 πυρήνων αντίστοιχα.....	34
Πίνακας 4.1 Περιγραφή εισόδων/εξόδων.....	51
Πίνακας 4.2 Δεδομένα Κρυπτογράφησης XTS-AES-128.....	51
Πίνακας 4.3 Δεδομένα αποκρυπτογράφησης XTS-AES-128.....	53
Πίνακας 4.4 Δεδομένα κρυπτογράφησης XTS-AES-256.....	54
Πίνακας 4.5 Δεδομένα αποκρυπτογράφησης XTS-AES-256.....	55
Πίνακας 4.6 Δεδομένα κρυπτογράφησης XTS-AES-128.....	56
Πίνακας 4.7 Δεδομένα αποκρυπτογράφησης XTS-AES-128.....	58
Πίνακας 4.8 Δεδομένα κρυπτογράφησης XTS-AES-256.....	59
Πίνακας 4.9 Δεδομένα αποκρυπτογράφησης XTS-AES-256.....	61
Πίνακας 4.10 Δεδομένα κρυπτογράφησης XTS-AES-128.....	62
Πίνακας 4.11 Δεδομένα αποκρυπτογράφησης XTS-AES-128.....	64
Πίνακας 4.12 Δεδομένα κρυπτογράφησης XTS-AES-256.....	65
Πίνακας 4.13 Δεδομένα αποκρυπτογράφησης XTS-AES-256.....	67
Πίνακας 5.1 Λεπτομέρειες των FPGAs.....	70
Πίνακας 5.2 Αποτελέσματα Virtex6.....	70
Πίνακας 5.3 Αποτελέσματα Virtex5.....	74
Πίνακας 5.4 Αποτελέσματα Spartan6.....	74
Πίνακας 5.5 Αποτελέσματα Virtex6.....	76
Πίνακας 5.6 Αποτελέσματα Virtex5.....	76
Πίνακας 5.7 Αποτελέσματα Spartan6.....	76
Πίνακας 5.8 Αποτελέσματα σύνθεσης, Virtex6.....	81

Κεφάλαιο 1

Εισαγωγή – Κρυπτογραφία

Στην σημερινή εποχή, η ανάγκη για προστασία των προσωπικών μας δεδομένων έχει γίνει επιτακτική. Τα δεδομένα και γενικά οι πληροφορίες από τη φύση αυτής έχουν την έννοια του “απορρήτου”. Ο βαθμός κρισιμότητας αυτής και η ανάγκη προστασίας αυτής αποτελούν μεγέθη ανάλογα. Όσο πιο σημαντικά είναι τα δεδομένα τόσο πιο επιτακτική είναι η ανάγκη να διαφυλαχτεί το περιεχόμενο αυτής. Η ανάγκη λοιπόν για τη διαφύλαξη των δεδομένων, είτε αυτά πρόκειται να αποσταλούν είτε πρόκειται να αποθηκευτούν, οδήγησε τον άνθρωπο να ψάξει να βρει τις μεθόδους εκείνες που θα του επιτρέψουν να διασφαλίσει αυτές πληροφορίες και τα δεδομένα του. Σήμερα, στην εποχή της πληροφορίας και του διαδικτύου, η προστασία των δεδομένων είναι αναπόσπαστο κομμάτι των πληροφοριακών συστημάτων και αποτελεί έναν ολόκληρο επιστημονικό κλάδο που ονομάζεται κρυπτογραφία.

Στο πρώτο κεφάλαιο αυτής διπλωματική εργασίας θα εξηγήσουμε βασικές αρχές και έννοιες που σχετίζονται με τη κρυπτογραφία. Ακόμη, θα αναφερθούμε στην ανάγκη κρυπτογράφησης των αποθηκευτικών μέσων (σκληροί δίσκοι, USB drives κ.α.) καθώς και στα πλεονεκτήματα υλοποίησης αυτής σε υλικό (hardware).

1.1 Κρυπτογραφία

Ο όρος κρυπτογραφία προέρχεται από τα συνθετικά “κρυπτός” + “γράφω” και είναι ένα διεπιστημονικό πεδίο που ασχολείται με τη μελέτη, την ανάπτυξη και τη χρήση τεχνικών κρυπτογράφησης και αποκρυπτογράφησης με σκοπό την απόκρυψη του περιεχόμενου των δεδομένων[1]. Η κρυπτογραφία είναι ο ένας από τους δύο κλάδους της κρυπτολογίας (ο άλλος είναι η κρυπτανάλυση), η οποία ασχολείται με τη μελέτη της ασφαλούς μετάδοσης ή αποθήκευσης δεδομένων. Η σημασία της κρυπτολογίας είναι τεράστια στους τομείς της ασφάλειας υπολογιστικών συστημάτων και των τηλεπικοινωνιών. Ο κύριος στόχος της είναι να παρέχει μηχανισμούς ώστε δύο ή περισσότερα άκρα επικοινωνίας (π.χ. άνθρωποι, προγράμματα υπολογιστών κλπ.) να ανταλλάζουν μηνύματα, χωρίς κανέναν τρίτο να είναι ικανός να διαβάσει την περιεχόμενη πληροφορία εκτός από τα δύο κύρια άκρα.

Ιστορικά, η κρυπτογραφία χρησιμοποιήθηκε για τη μετατροπή της πληροφορίας μηνυμάτων από μια κανονική, κατανοητή μορφή σε έναν “γρίφο”, που χωρίς τη γνώση του κρυφού μετασχηματισμού θα παρέμενε ακατανόητος. Κύριο χαρακτηριστικό των παλαιότερων μορφών κρυπτογράφησης ήταν ότι η επεξεργασία γινόταν πάνω στη γλωσσική δομή του μηνύματος. Στις νεότερες μορφές, η κρυπτογραφία κάνει χρήση του αριθμητικού ισοδύναμου, ενώ η έμφαση έχει μεταφερθεί σε διάφορα πεδία των μαθηματικών, όπως διακριτά μαθηματικά, θεωρία αριθμών, θεωρία πληροφορίας, υπολογιστική πολυπλοκότητα, στατική και συνδυαστική ανάλυση.

1.1.1 Βασικές έννοιες

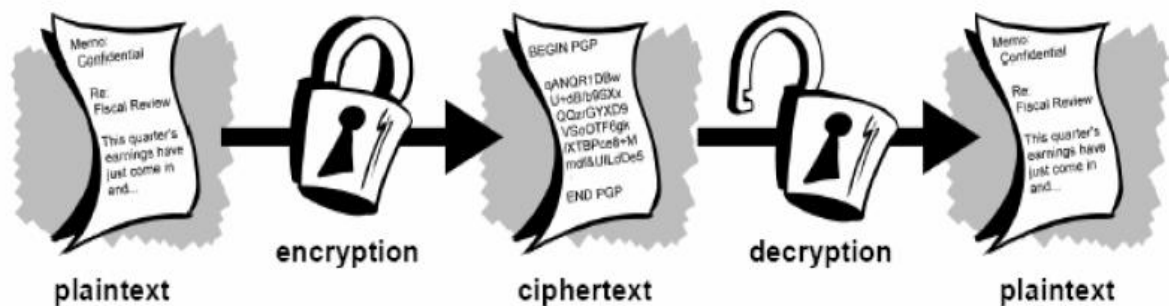
Ο πρωταρχικός στόχος της κρυπτογραφίας είναι η επικοινωνία δύο κρυπτοσυστημάτων μέσω ασφαλούς διαύλου επικοινωνίας ώστε να καθίσταται αδύνατη η παρεμβολή, στην επικοινωνία αυτών, οποιουδήποτε μη εξουσιοδοτημένου μέσου.

Η ορολογία, η οποία είναι απαραίτητη και χρησιμοποιείται για να περιγράψουμε έναν αλγόριθμο ή μια διαδικασία κρυπτογράφησης/αποκρυπτογράφησης, περιγράφεται συνοπτικά στις ακόλουθες παραγράφους[2]

- **Κρυπτογράφηση (*encryption*)** ονομάζεται η διαδικασία μετασχηματισμού ενός μηνύματος σε μία ακατανόητη μορφή με τη χρήση κάποιου κρυπτογραφικού αλγορίθμου ούτως ώστε να μην μπορεί να διαβαστεί από κανέναν εκτός του νόμιμου παραλήπτη. Η αντίστροφη διαδικασία όπου από το κρυπτογραφημένο κείμενο παράγεται το αρχικό μήνυμα ονομάζεται **αποκρυπτογράφηση (*decryption*)**.

- **Κρυπτογραφικός αλγόριθμος (*cipher*)** είναι η μέθοδος μετασχηματισμού δεδομένων σε μία μορφή που να μην επιτρέπει την αποκάλυψη των περιεχομένων τους από μη εξουσιοδοτημένα μέρη. Κατά κανόνα ο κρυπτογραφικός αλγόριθμος είναι μία πολύπλοκη μαθηματική συνάρτηση.
- **Αρχικό κείμενο (*plaintext*)** είναι το μήνυμα το οποίο αποτελεί την είσοδο σε μία διεργασία κρυπτογράφησης.
- **Κλειδί (*key*)** είναι ένας αριθμός αρκετών bits που χρησιμοποιείται ως είσοδος στη συνάρτηση κρυπτογράφησης.
- **Κρυπτογραφημένο κείμενο (*ciphertext*)** είναι το αποτέλεσμα της εφαρμογής ενός κρυπτογραφικού αλγόριθμου πάνω στο αρχικό κείμενο.
- **Κρυπτανάλυση (*cryptanalysis*)** είναι μία επιστήμη που ασχολείται με το "σπάσιμο" κάποιας κρυπτογραφικής τεχνικής ούτως ώστε χωρίς να είναι γνωστό το κλειδί της κρυπτογράφησης, το αρχικό κείμενο να μπορεί να αποκωδικοποιηθεί.

Η διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης φαίνεται συνοπτικά στο παρακάτω σχήμα.



Σχήμα 1.1 Διαδικασία κρυπτογράφησης/αποκρυπτογράφησης

Το *plaintext* συμβολίζεται συνήθως με το γράμμα P και είναι μια ακολουθία από bits η οποία μπορεί να αναπαριστά ένα αρχείο κειμένου, ήχου, εικόνας ή οτιδήποτε άλλο. Στους ηλεκτρονικούς υπολογιστές το P είναι δεδομένα σε δυαδική μορφή. Το *plaintext* μπορεί να προορίζεται είτε για μετάδοση είτε για αποθήκευση[3]. Σε κάθε περίπτωση πάντως, το *plaintext* είναι το αρχείο που πρόκειται να κρυπτογραφηθεί. Το *ciphertext* είναι το αποτέλεσμα κρυπτογράφησης του *plaintext* και συμβολίζεται με C . Το *ciphertext* είναι και αυτό μια ακολουθία από bits. Το μέγεθος του *ciphertext* μπορεί να είναι ίσο, μικρότερο ή μεγαλύτερο από μέγεθος του *plaintext*. Αν η συνάρτηση κρυπτογράφησης που χρησιμοποιούμε συμβολίζεται ως E , τότε μπορούμε να παραστήσουμε μαθηματικά τη κρυπτογράφηση ως

$$E(P) = C \quad (1.1)$$

Αντίθετα, αν συμβολίζουμε με D τη συνάρτηση αποκρυπτογράφησης, τότε η αποκρυπτογράφηση μπορεί να συμβολιστεί ως

$$D(C) = P \quad (1.2)$$

Εφόσον η διαδικασία αποκρυπτογράφησης, έχει ως στόχο τη παραγωγή του αρχικού *plaintext*, τότε θα πρέπει να ισχύει η ισότητα

$$D(E(P)) = P \quad (1.3)$$

1.1.2 Βασικές λειτουργίες

Οι βασικές λειτουργίες στην αλληλεπίδραση των υπολογιστών είναι αντίστοιχες με εκείνες των διαπροσωπικών σχέσεων. Συγκεκριμένα, οι βασικές λειτουργίες που θα πρέπει να εξυπηρετούνται στην επιστήμη της κρυπτογραφίας είναι οι ακόλουθες

- **Εμπιστευτικότητα** (Confidentiality) είναι η υπηρεσία που χρησιμοποιείται για να προφυλάσσεται το περιεχόμενο της πληροφορίας από τους μη εξουσιοδοτημένους ενδιαφερόμενους. Η μυστικότητα (secrecy) αποτελεί όρο συνώνυμο με την εμπιστευτικότητα.
- **Ακεραιότητα Δεδομένων** (Data integrity) είναι η υπηρεσία που διευθετεί τη μη εξουσιοδοτημένη αλλαγή δεδομένων. Για να διασφαλιστεί η ακεραιότητα δεδομένων, κάποιος πρέπει να έχει τη δυνατότητα να ανιχνεύει την αλλαγή δεδομένων από μη εξουσιοδοτημένα μέλη. Η αλλαγή των δεδομένων περιλαμβάνει ενέργειες όπως εισαγωγή, αφαίρεση-διαγραφή και αντικατάσταση.
- **Πιστοποίηση** (Authentication) είναι η υπηρεσία που σχετίζεται με την αναγνώριση- ταυτοποίηση. Τα δύο μέρη που θέλουν να επικοινωνήσουν πρέπει να αναγνωρίσουν το ένα το άλλο. Αυτή η διεργασία εφαρμόζεται και στις δύο οντότητες που θέλουν να επικοινωνήσουν αλλά και στη πληροφορία. Για την πληροφορία η οποία μεταφέρεται λοιπόν, απαιτείται πιστοποίηση οντότητας και πιστοποίηση προέλευσης δεδομένων. Η πιστοποίηση προέλευσης δεδομένων μπορεί να παρέχει την ακεραιότητα δεδομένων.
- **Μη άρνηση** (Non-repudiation) είναι η υπηρεσία που αποτρέπει μια οντότητα από το να αρνηθεί ότι διέπραξε προηγούμενες συμφωνίες ή ενέργειες. Όταν προκύπτουν διαφωνίες λόγω της άρνησης μιας οντότητας ότι ορισμένες ενέργειες έχουν πραγματοποιηθεί τότε είναι απαραίτητοι οι μηχανισμοί εκείνοι ώστε να επιλύονται τέτοια προβλήματα. Γι παράδειγμα, μια οντότητα μπορεί να εγκρίνει μια αγορά από μια άλλη οντότητα και αργότερα να αρνηθεί αυτή την ενέργεια. Επομένως, μια διαδικασία που περιλαμβάνει ένα εμπιστευμένο τρίτο (third party) απαιτείται για να επιλύσει το θέμα.

1.1.3 Αλγόριθμοι και κλειδιά

Περιγράφοντας έναν αλγόριθμο κρυπτογράφησης (*cipher*), στην πραγματικότητα αναφερόμαστε σε μια πολύπλοκη μαθηματική συνάρτηση. Παλαιότερα, βασική απαίτηση αυτού του αλγορίθμου ήταν, για λόγους ασφαλείας, να είναι γνωστός μόνο στα μέλη που τον χρησιμοποιούν. Τέτοιου είδους αλγόριθμοι, που η ασφάλεια τους βασίζεται στο τρόπο λειτουργίας του ονομάζονται **Περιορισμένοι** (*restricted*). Αυτού του είδους οι αλγόριθμοι θεωρούνται ξεπερασμένοι για τις σημερινές απαιτήσεις και έχουν μόνο ιστορικό ενδιαφέρον.

Τα σύγχρονα προβλήματα άπτονται στο πεδίο της σύγχρονης κρυπτογραφίας και αντιμετωπίζονται με τη χρήση κλειδιών (*keys*) το οποίο συμβολίζεται με K . Το εύρος που μπορεί να πάρει ένα κλειδί ονομάζεται *keyspace*. Τόσο οι διαδικασίες κρυπτογράφησης όσο και διαδικασίες αποκρυπτογράφησης χρησιμοποιούν το κλειδί και μαθηματικά μπορούν να περιγραφούν ως

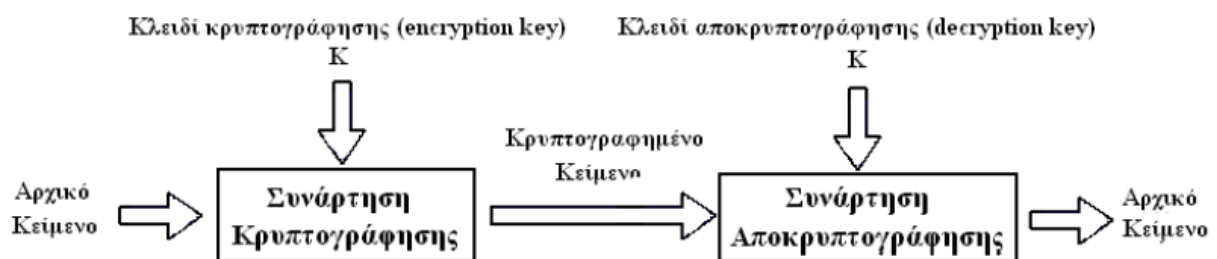
$$E_k(P) = C \quad (1.4)$$

$$D_k(C) = P \quad (1.5)$$

Οι παραπάνω συναρτήσεις προφανώς έχουν την ιδιότητα

$$D_k(E_k(P)) = P \quad (1.6)$$

Η χρήση κλειδιών στη διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης αποικτά τη μορφή που απεικονίζεται στο σχήμα 1.2.



Σχήμα 1.2 Διαδικασία κρυπτογράφησης/αποκρυπτογράφησης με το ίδιο κλειδί

Ακόμη, υπάρχουν αλγόριθμοι που χρησιμοποιούν διαφορετικό κλειδί κατά τη διαδικασίες κρυπτογράφησης και αποκρυπτογράφησης. Συμβολίζουμε με K το κλειδί κρυπτογράφησης και με K' το κλειδί της αποκρυπτογράφησης.

Η παραπάνω διαδικασία περιγράφεται μαθηματικά ως

$$E_k(P) = C \quad (1.7)$$

$$D_{k'}(C) = P \quad (1.8)$$

$$D_{k'}(E_k(P)) = P \quad (1.9)$$

Η διαδικασία κρυπτογράφησης/αποκρυπτογράφησης με διαφορετικά κλειδιά φαίνεται στο σχήμα 1.3



Σχήμα 1.3 Διαδικασία κρυπτογράφησης/αποκρυπτογράφησης με το διαφορετικό κλειδί

Η ασφάλεια των παραπάνω αλγορίθμων βασίζεται αποκλειστικά στο κλειδί και όχι στις λεπτομέρειες του αλγόριθμου. Αυτό σημαίνει ότι ο αλγόριθμος μπορεί να δημοσιοποιηθεί και να παραχθεί μαζί για διάφορα προϊόντα.

Όπως αναφέραμε και προηγουμένως, κρυπτογράφηση (*encryption*) ονομάζεται ο μετασχηματισμός των δεδομένων σε μορφή που δεν μπορεί να διαβαστεί από κανένα παρά μόνο από αυτόν που διαθέτει ένα κατάλληλο κλειδί. Υπάρχουν δύο μεγάλες οικογένειες αλγορίθμων κρυπτογράφησης, οι συμμετρικοί αλγόριθμοι (ή αλγόριθμοι μυστικού κλειδιού) και οι ασύμμετροι (ή αλγόριθμοι δημόσιου κλειδιού)[4].

1.1.3.1 Συμμετρικοί αλγόριθμοι κρυπτογράφησης

Στους συμμετρικούς αλγόριθμους το κλειδί κρυπτογράφησης μπορεί να υπολογιστεί από το κλειδί που χρησιμοποιείται για την αποκρυπτογράφηση και το ανάποδο. Μάλιστα στις περισσότερες περιπτώσεις τα κλειδιά κρυπτογράφησης και αποκρυπτογράφησης είναι τα ίδια. Αυτοί οι αλγόριθμοι χρειάζονται την συμφωνία μεταξύ του αποστολέα και του παραλήπτη για το κλειδί που θα χρησιμοποιηθεί, για να μπορέσουν να επικοινωνήσουν με ασφάλεια.

Η ασφάλεια των αλγορίθμων βασίζεται στην μυστικότητα αυτού του κλειδιού. Για όσο καιρό επιθυμούμε η επικοινωνία να παραμείνει μυστική, για τον ίδιο καιρό πρέπει και το κλειδί να παραμείνει μυστικό.

Οι συμμετρικοί αλγόριθμοι μπορούν να διαιρεθούν σε δύο υποκατηγορίες:

- αλγόριθμοι ροής (stream ciphers) οι οποίοι λειτουργούν bit προς bit και
- μπλοκ αλγόριθμοι (block ciphers) οι οποίοι λειτουργούν πάνω σε τμήματα δεδομένων (συνήθως των 64 ή 128 bit).

Παραδείγματα συμμετρικών αλγορίθμων είναι οι AES, DES, IDEA, RC5 και SAFER.

1.1.3.2 Ασύμμετροι αλγόριθμοι κρυπτογράφησης

Οι ασύμμετροι αλγόριθμοι ή αλγόριθμοι δημόσιου κλειδιού είναι σχεδιασμένοι έτσι ώστε το κλειδί που χρησιμοποιείται για την κρυπτογράφηση να είναι διαφορετικό από το κλειδί που χρησιμοποιείται για την αποκρυπτογράφηση. Πέρα από αυτό, το κλειδί αποκρυπτογράφησης δεν μπορεί να υπολογιστεί από το κλειδί κρυπτογράφησης. Οι αλγόριθμοι αυτοί καλούνται και "δημόσιου κλειδιού" γιατί το κλειδί κρυπτογράφησης μπορεί να δημοσιοποιηθεί. Ο καθένας μπορεί να κρυπτογραφήσει ένα μήνυμα με το δημόσιο κλειδί αλλά μόνο αυτός που διαθέτει το αντίστοιχο ιδιωτικό κλειδί μπορεί να το αποκρυπτογραφήσει. Παραδείγματα ασύμμετρων αλγορίθμων είναι οι RSA, ElGamal και DSA.

Οι συμμετρικοί αλγόριθμοι είναι πολύ πιο γρήγοροι, εφαρμοσμένοι είτε σε υλικό είτε σε λογισμικό από τους ασύμμετρους αλγόριθμους. Ως εκ τούτου οι συμμετρικοί αλγόριθμοι χρησιμοποιούνται για την κρυπτογράφηση του κυρίου μέρους των δεδομένων, ενώ οι αλγόριθμοι δημόσιου κλειδιού βρίσκουν κατάλληλη εφαρμογή σε πρωτόκολλα ανταλλαγής κλειδιών και ψηφιακών υπογραφών.

1.1.3.3 Block ciphers

Στη κρυπτογραφία, *block cipher* ονομάζεται ένας αλγόριθμος συμμετρικού κλειδιού που λειτουργεί για μεγέθη δεδομένων σταθερού μήκους, εφαρμόζοντας έναν συγκεκριμένο μετασχηματισμό. Όταν εκτελείται η κρυπτογράφηση, ο *block cipher* παίρνει ένα μπλοκ του *plaintext* ή μιας καθορισμένης εισόδου και παράγει ένα μπλοκ *ciphertext* ίσου μεγέθους. Αυτός ο μετασχηματισμός λαμβάνει χώρα εφόσον ένας εξουσιοδοτημένος χρήστης έχει παράσχει το μυστικό κλειδί. Η αποκρυπτογράφηση είναι μια αντίστοιχη διαδικασία. Πλέον όμως, στην είσοδο του αλγορίθμου τοποθετείται το *ciphertext* και στην έξοδο παράγεται το αντίστοιχο *plaintext*.



Σχήμα 1.4 Block cipher κρυπτογράφηση/αποκρυπτογράφηση

Οι μπλοκ αλγόριθμοι διαφέρουν από τους αλγόριθμους ροής επειδή τροποποιούν τα bit του *plaintext* μεμονωμένα, το ένα μετά το άλλο, και ο μετασχηματισμός ποικίλει κατά τη διαδικασία κρυπτογράφησης. Η διαφορά των δύο τύπων αλγορίθμων είναι κάπως ασαφής, αφού ένας μπλοκ αλγόριθμος μπορεί να λειτουργήσει σε μια κατάσταση και ως αλγόριθμος ροής

Το Data Encryption Standard (DES)[5] ήταν ένα σχέδιο μπλοκ αλγόριθμου που είχε μεγάλη επιρροή. Αναπτύχθηκε και δημοσιεύθηκε από την IBM και κυκλοφόρησε το 1977 ως πρότυπο. Το Advanced Encryption Standard (AES) είναι ο διάδοχος του DES ο οποίος υιοθετήθηκε το 2001.

Για να κρυπτογραφηθούν μηνύματα τα οποία είναι μεγαλύτερα από το μέγεθος ενός μπλοκ, διάφορες καταστάσεις λειτουργίας έχουν προταθεί και χρησιμοποιούνται. Οι πιο συνηθισμένες και ευρέως διαδεδομένες είναι οι ακόλουθες[6]

- **Electronic Code Book (ECB)** είναι η πιο εύκολη κατάσταση λειτουργίας. Επικεντρώνεται στο “σπάσιμο” της συνολικής πληροφορίας σε n ίσα τμήματα ($P_1, P_2, \dots, P_n$) και ύστερα κρυπτογραφεί το κάθε ένα ξεχωριστά.

$$C_i = E_k(P_i) \quad (1.10)$$

Η μεγάλη αδυναμία αυτής της κατάστασης λειτουργίας έγκειται στο γεγονός ότι αν το ίδιο *plaintext* κρυπτογραφηθεί δύο ή περισσότερες φορές τότε θα παραχθεί το ίδιο *ciphertext*. Η τυπική εφαρμογή του ECB είναι για μικρού μήκους δεδομένα όπως προσωρινά κλειδιά κρυπτογράφησης

- **Cipher Block Chaining (CBC)**. Όπως και στην ECB κατάσταση λειτουργίας, το *plaintext* χωρίζεται σε ίσα μπλοκ δεδομένων ($P_1, P_2, \dots, P_n$). Βασική διαφορά του όμως με το ECB είναι ότι το αποτέλεσμα κρυπτογράφησης ενός *plaintext* εξαρτάται από το προηγούμενο *ciphertext*. Αυτό που συμβαίνει στην ουσία είναι ότι το *plaintext* που οδεύει προς κρυπτογράφηση γίνεται πρώτα XOR με το τελευταίο *ciphertext*. Για να κρυπτογραφηθεί το πρώτο *plaintext*, χρησιμοποιείται ένα αρχικό διάνυσμα (*initial vector, IV*).

$$C_i = E_k(C_{i-1} \oplus P_i) \quad (1.11)$$

$$C_0 = IV \quad (1.12)$$

Το βασικό πλεονέκτημα του CBC είναι η κρυπτογράφηση του συγκεκριμένου μπλοκ εξαρτάται από όλα προηγούμενα, έτσι δύο ή περισσότερα ίδια *plaintext* θα κρυπτογραφηθούν διαφορετικά.

- **Cipher Feed Back (CFB)** είναι μια κατάσταση λειτουργίας που διαφέρει αρκετά από αυτές που μόλις αναφέραμε επειδή το κλειδί είναι διαφορετικό για κάθε *plaintext*. Το πρώτο κλειδί παράγεται από το αρχικό διάνυσμα (*initial vector*) και τα υπόλοιπα από το προηγούμενο *ciphertext*

$$C_i = P_i \oplus K_i \quad (1.13)$$

Τα πλεονεκτήματα του CFB είναι ότι ο μπλοκ αλγόριθμος μπορεί να λειτουργεί και ως αλγόριθμος ροής αν τα δεδομένα φθάνουν κατά τμήματα. Τα μειονεκτήματα είναι ότι ένα κατεστραμμένο τμήμα *ciphertext*, κατά τη διάρκεια της μετάδοσης, θα επηρεάσει το παρών και ακόμη περισσότερα *plaintext* όταν επιχειρηθεί η αποκρυπτογράφηση.

- **Output FeedBack(OFB)** είναι μια κατάσταση λειτουργίας παρόμοια με το CFB, με τη διαφορά ότι το κάθε κλειδί παράγεται από το προηγούμενο. Με την αλλαγή

αυτή, επιτυγχάνεται η αποτροπή της διάδοσης ενός κατεστραμμένου τμήματος *ciphertext* προς αποκρυπτογραφημένα τμήματα που ακολουθούν.

- **Counter Mode Operation (CTR)** η βασική ιδέα του *Counter mode* είναι η δημιουργία μιας ροής κλειδιών από ένα απλό κλειδί ούτως ώστε να κρυπτογραφηθούν όλα τα plaintext με διαφορετικό κλειδί.

$$T_1 = IV \quad (1.14)$$

$$T_i = T_{i-1} + 1 \quad (1.15)$$

$$C_i = P_i \oplus E_k(T_i) \quad (1.16)$$

$$C = (IV, C_1, C_2 \dots) \quad (1.17)$$

Το κύριο πλεονέκτημα αυτού του τρόπου λειτουργίας είναι η γρήγορη κρυπτογράφηση και αποκρυπτογράφηση επειδή τα μπλοκ επεξεργάζονται παράλληλα. Το αρχικό διάνυσμα (initial value, IV) δε θα πρέπει να χρησιμοποιηθεί αρκετές φορές

1.2 Κρυπτογράφηση δίσκου

Η Κρυπτογράφηση δίσκου είναι μια ειδική περίπτωση προστασίας των δεδομένων σε χρήση, όταν το μέσο αποθήκευσης είναι *sector-base* συσκευές, όπως είναι ο σκληρός δίσκος. Στην ενότητα αυτή αναφέρεται η θεωρία του *Disk Encryption*[7] και η κατάσταση στην οποία βρίσκεται. Δεδομένου ότι η επιλογή του τρόπου κρυπτογράφησης (*cipher mode*) είναι ουσιώδης για την ασφάλεια των δεδομένων του σκληρού δίσκου, θα αναφερθούν οι εξελίξεις στον οργανισμό τυποποίησης *IEEE* πάνω στα προτεινόμενα πρότυπα.

Η *SISWG*[8] έχει τυποποιήσει ένα *narrow cipher mode* και ένα *wide cipher mode*. Ένα *narrow cipher mode* λειτουργεί με ένα block μεγέθους ίσου με το μέγεθος του block του υποκειμένου κρυπτογράφησης. Ένα *wide cipher mode* περιλαμβάνει περισσότερα *cipher block* και αναπτύσσει μια ολόκληρη μονάδα στο μέγεθος ενός τομέα (*sector*), επιτρέποντας στο πρότυπο να διαχέει την πληροφορία σε όλο το *sector*.

Τα βασικότερα πρότυπα που παρουσιάστηκαν είναι το CBC, LRW, XEX, XTS, CMC, EME και το ESSIV. Το LRW ήταν το προτιμώμενο πρότυπο για *narrow cipher mode* το οποίο τελικά εγκαταλείφθηκε λόγω των αδυναμιών του και τη θέση του πήρε το XTS, με το οποίο θα ασχοληθούμε σε αυτήν τη διπλωματική εργασία, ενώ για *wide cipher mode* προτιμήθηκε το πρότυπο EME.

Μέχρι σήμερα, η παρουσίαση των τρόπων κρυπτογράφησης, έχει περισσότερο γενικό χαρακτήρα. Το *Disk Encryption* γίνεται σε ένα πλαίσιο διαφορετικό από την κρυπτογράφηση επικοινωνίας (*communication encryption*). Θεωρείται ότι στην κρυπτογράφηση δίσκου δεν υπάρχει κανένας τρόπος για να αποθηκευθούν πληροφορίες στο ενδιαμέσο, εκτός από το μυστικό κλειδί. Μετρητές ή ενδιάμεσες τιμές δεν μπορούν να αποθηκευτούν σε αντίθεση με την κρυπτογράφηση επικοινωνίας. Στην ενότητα αυτή θα γίνει αναφορά στην Κρυπτογράφηση Συστημάτων Αρχείων και στη Πλήρη Κρυπτογράφηση Δίσκου.

1.2.1 Πλήρης κρυπτογράφηση δίσκου

Πλήρης κρυπτογράφηση δίσκου είναι ένα είδος *disk encryption (software ή hardware)*, κατά το οποίο κρυπτογραφείται κάθε bit δεδομένων που πηγαίνει σε ένα δίσκο. Ο όρος "πλήρης" χρησιμοποιείται για να δηλώσει ότι τα πάντα σε ένα δίσκο συμπεριλαμβανομένου και του λειτουργικού συστήματος είναι κρυπτογραφημένα.

Η Πλήρης κρυπτογράφηση δίσκου έχει πολλά πλεονεκτήματα σε σύγκριση με τα συνήθη μοντέλα κρυπτογράφησης αρχείων ή φακέλων. Τα παρακάτω είναι κάποια οφέλη του FDE.

- Τα πάντα, συμπεριλαμβανομένου των προσωρινών αρχείων αλλά και του χώρου swap, κρυπτογραφούνται. Η κρυπτογράφηση αυτών των αρχείων είναι σημαντική, δεδομένου ότι μπορούν να αποκαλύψουν σημαντικές εμπιστευτικές πληροφορίες.
- Η απόφαση για το ποια αρχεία θα κρυπτογραφηθούν δεν εναπόκειται στους χρήστες.
- Παρέχουν υποστήριξη πιστοποίησης πριν από την εκκίνηση του λειτουργικού.

Η Πλήρης κρυπτογράφηση δίσκου ορισμένες φορές χρησιμοποιείται σε συνδυασμό με *file-system-level encryption*, έτσι ώστε να επιδιώκεται μεγαλύτερη ασφάλεια. Στην περίπτωση του FDE από την στιγμή που χρησιμοποιείται ένα μόνο κλειδί για όλα τα δεδομένα αυτά είναι αποκρυπτογραφημένα όταν το σύστημα λειτουργεί. Αν ο επιτιθέμενος αποκτήσει πρόσβαση στο υπολογιστικό σύστημα την ώρα που βρίσκεται σε λειτουργία τότε έχει πρόσβαση σε όλα τα αρχεία του δίσκου. Η κρυπτογράφηση φακέλων και αρχείων συμπληρώνει το FDE μιας και απαιτεί διαφορετικό κλειδί για κάθε κρυπτογραφημένο αρχείο, με αποτέλεσμα ο εισβολέας να έχει λιγότερες πιθανότητες να συλλέξει πληροφορίες, ακόμα και αν αποκτήσει πρόσβαση στο υπολογιστικό σύστημα.

1.2.2 Κρυπτογράφηση συστημάτων αρχείων

Η κρυπτογράφηση συστημάτων αρχείων[9] (*file-system-level encryption*) περιγράφει τις μεθόδους και τις τεχνικές κρυπτογράφησης που χρησιμοποιούνται με σκοπό την απόκρυψη του περιεχομένου, δηλαδή των αρχείων και καταλόγων, ενός συστήματος αρχείων. Τα ιδιωτικά δεδομένα που περιέχονται στο σύστημα αρχείων πρέπει να προστατευθούν από πιθανές επιθέσεις κάποιου αντιπάλου ο οποίος ίσως να αποκτήσει πρόσβαση μέσω δικτύου στον υπολογιστή ή μπορεί να αποκτήσει το ίδιο το φυσικό μέσο (π.χ. το σκληρό δίσκο) στο οποίο βρίσκεται το σύστημα αρχείων. Ο αντίπαλος θα πρέπει να μην μπορεί να διαβάσει τα ιδιωτικά δεδομένα (ή ακόμα καλύτερα δεν θα πρέπει να μπορεί να διαπιστώσει καν την ύπαρξή τους).

Η κρυπτογράφηση συστημάτων αρχείων διακρίνεται από την κρυπτογράφηση σε επίπεδο δίσκου, στην οποία κάθε bit δεδομένων, συμπεριλαμβανομένων όλων των προγραμμάτων, που αποθηκεύεται σε έναν δίσκο κρυπτογραφείται.

Οι αρχές που διέπουν τη κρυπτογράφηση συστημάτων αρχείων είναι οι παρακάτω:

- Τα ιδιωτικά δεδομένα δεν πρέπει ποτέ να αποθηκεύονται στο σύστημα αρχείων σε μορφή καθαρού κειμένου.
- Η κρυπτογράφηση και η αποκρυπτογράφηση γίνονται διαρκώς κάθε φορά που τα δεδομένα μεταφέρονται από τη μνήμη στο δίσκο και αντίστροφα.
- Η διαδικασία αυτή γίνεται διάφανα, χωρίς καμιά παρέμβαση από το χρήστη, εκτός από την αρχή της χρήσης του συστήματος αρχείων, όταν του ζητείται να δώσει τα κλειδιά.
- Η πιο ασφαλής πρακτική είναι τα κλειδιά της κρυπτογράφησης να φυλάσσονται σε κάποιο φυσικό μέσο διαφορετικό από αυτό που φιλοξενεί τα κρυπτογραφημένα δεδομένα.
- Σε έναν αντίπαλο παρατηρητή, τα κρυπτογραφημένα δεδομένα πρέπει να φαίνονται σαν τυχαιοποιημένα δεδομένα.

Ενδεικτικά αναφέρουμε ότι η κρυπτογράφηση συστημάτων αρχείων χρησιμοποιείται ήδη σε λειτουργικά συστήματα Unix(π.χ. Mac OS) και Windows με τη χρήση εύκολων προγραμμάτων. Ο αλγόριθμος που χρησιμοποιείται συνήθως είναι ο AES.

1.2.3 Full disk encryption VS File-system-level encryption

Στη κρυπτογράφηση συστημάτων αρχείων συνηθίζεται να είναι κρυπτογραφημένο ένα διαμέρισμα του σκληρού δίσκου (π.χ. ο κατάλογος /home στα συστήματα Unix), ενώ κάποια άλλα διαμερίσματα δεν είναι. Ένας επιτιθέμενος αντίπαλος μπορεί να αντλήσει πληροφορίες για τα περιεχόμενα του κρυπτογραφημένου καταλόγου (π.χ. μια λίστα με τα ονόματα των αρχείων) αν διαβάσει ένα άλλο, μη κρυπτογραφημένο σύστημα αρχείων στον ίδιο υπολογιστή (όπως /var, /tmp, /swap).

Ακόμη, μπορεί να διαβάσει τα δεδομένα κάποιων συστημάτων αρχείων, που μερικές φορές αποθηκεύονται σε μορφή καθαρού κειμένου. Μία λύση είναι να κρυπτογραφούνται και τα υπόλοιπα διαμερίσματα του δίσκου, ή ακόμα και ολόκληρος ο δίσκος. Έτσι η λύση της πλήρους κρυπτογράφησης δίσκου προκρίνεται έναντι της κρυπτογράφησης συστημάτων αρχείων.

Σημειώνεται ότι παρόμοιο πρόβλημα υπάρχει και στο EFS της Microsoft, το οποίο δεν μπορεί να κρυπτογραφήσει τους καταλόγους συστήματος.

1.3 Υλοποίηση σε υλικό έναντι λογισμικού

Για την ασφαλή κρυπτογράφηση δεδομένων σε USBs, σκληρούς δίσκους και SSDs, μόνο δύο μέθοδοι είναι διαθέσιμες. Η κρυπτογράφηση βασισμένη σε λογισμικό (*Software encryption*) και η κρυπτογράφηση βασισμένη σε υλικό(*Hardware encryption*)[10]. Η *software encryption* βασίζεται σε λογισμικό, όπου η κρυπτογράφηση του δίσκου παρέχεται μέσω ενός εξωτερικού λογισμικού για να διασφαλίσει τα δεδομένα. Διάφορες επιλογές κρυπτογράφησης

βασισμένες σε λογισμικό είναι διαθέσιμες στην αγορά ως μια φθηνότερη εναλλακτική έναντι της *hardware encryption*, αλλά τα μειονεκτήματά τείνουν να εκμηδενίσουν τα οφέλη αυτής της επιλογής. Συχνά, παρατηρούνται απαιτήσεις για αρκετές ενημερώσεις ούτως ώστε να αποτρέπονται νέου είδους επιθέσεις.

Επιπλέον, η *software encryption* μπορεί να είναι αρκετά αργή και να απαιτεί πολύπλοκούς *driver* και εγκαταστάσεις λογισμικού. Η *software encryption* μπορεί επίσης να μην παρέχει τη πλήρη ασφάλεια του απαιτούν οι επιχειρήσεις, ούτως ώστε να μην διαρρεύσουν ευαίσθητες πληροφορίες. Συμπερασματικά, η *software encryption* είναι καλύτερη από τη μη κρυπτογράφηση δεδομένων. Ακόμη, η *software encryption* είναι ευάλωτη σε σφάλματα του χρήστη καθώς δεδομένα μπορεί να μείνουν απροστάτευτα επειδή ο χρήστης ξέχασε ή αγνόησε κάποιες συγκεκριμένα βήματα της διαδικασίας κρυπτογράφησης.

Η *hardware encryption* σε USBs και σκληρούς δίσκους χειρίζεται αυτόματα τη διαδικασία κρυπτογράφησης καθώς αυτή είναι ενσωματωμένη ακριβώς μέσα στο ίδιο το *chip*. Εφόσον, τα δεδομένα έχουν κρυπτογραφηθεί, γίνονται μη αναγνώσιμα και αποθηκεύονται σε κρυπτογραφημένη μορφή μέσα στο δίσκο. Αν ένας μη εξουσιοδοτημένος χρήστης προσπαθήσει να αποκτήσει πρόσβαση στα δεδομένα χωρίς να έχει το απαιτούμενο κλειδί, τότε θα διαπιστώσει ότι τα δεδομένα που λαμβάνει δεν αποδίδουν καμία ουσιαστική πληροφορία. Εφόσον όμως αποκτήσει, με κάποιο τρόπο το κλειδί, τότε όλα τα δεδομένα που βρίσκονται στο δίσκο γίνονται διαθέσιμα στο χρήστη.

Κεφάλαιο 2

Το πρότυπο κρυπτογράφησης IEEE P1619

Σε αυτό το κεφάλαιο θα παρουσιάσουμε αναλυτικά το πρότυπο κρυπτογράφησης P1619 [11] καθώς και τα επιμέρους βήματά του. Επιπλέον, γίνεται σύντομη αναφορά στον αλγόριθμο AES (Advanced Encryption Standard) ο οποίος αποτελεί βασική μονάδα του προτύπου P1619. Στο πρότυπο αυτό ορίζεται η αρχιτεκτονική κρυπτογραφίας για την προστασία των δεδομένων που είναι αποθηκευμένα σε *sector-base* συσκευές με σταθερού μήκους μπλοκ. Το πρότυπο ορίζει το μετασχηματισμό κρυπτογράφησης και τη μέθοδο εισαγωγής και εξαγωγής των κρυπτογραφικών κλειδιών.

Βασική αρχή του P1619[12] είναι ότι βασίζεται στη κρυπτογράφηση μεταβλητού block (*tweakable block*) και χρησιμοποιείται σε μονάδες δεδομένων (*data units*) των 128-bits και πάνω, χρησιμοποιώντας την κρυπτογράφηση AES ως υπορουτίνα. Το βασικό χαρακτηριστικό του XTS είναι ότι και αυτός χρησιμοποιεί ένα μεταβλητό κλειδί (*tweak value*) το οποίο ορίζει τη λογική θέση των δεδομένων στον δίσκο. Το πρότυπο XTS αντιμετωπίζει απειλές όπως είναι η αντιγραφή-και-επικόλληση (*copy-and-paste attack*), ενώ επιτρέπει παραλληλισμό αλλά και *pipelining* επιτυγχάνοντας υψηλό throughput σε κρυπτογραφημένες υλοποιήσεις.

Η κρυπτογράφηση των δεδομένων που είναι σε μεταφορά δεν υποστηρίζεται από το πρότυπο, αλλά και σε αυτή την περίπτωση προσφέρεται μέγιστη ασφάλεια τόσο από τις περιπτώσεις κλοπής των δεδομένων μέσω του δικτύου όσο και από την κλοπή του ίδιου του σκληρού δίσκου.

2.1 Ο αλγόριθμος AES

Ο AES είναι ένας συμμετρικός κρυπτογραφικός αλγόριθμος *τμήματος (block)* δεδομένων και μπορεί να υλοποιηθεί σε λογισμικό, firmware, υλικό ή οποιοδήποτε συνδυασμό επ' αυτού[13]. Αποτελείται από σύνθετες συναρτήσεις, οι οποίες επαναλαμβάνονται πολλές φορές ξεκινώντας από την είσοδο του *plaintext*. Κάθε επανάληψη καλείται *κύκλος (round)*. Η επαναληπτική διαδικασία τροποποιείται ελαφρώς για τον τελικό κύκλο και υπάρχει ένα επιπρόσθετο στάδιο προ-επεξεργασίας κατά την έναρξη κάθε κρυπτογράφησης.

Ο αριθμός των κύκλων που απαιτούνται εξαρτάται από το επιλεγμένο μέγεθος κλειδιού. Για ένα κλειδί μεγέθους 128-bits απαιτούνται 10 κύκλοι, για ένα κλειδί 192-bits απαιτούνται 12 κύκλοι, και για ένα 256-bit κλειδί απαιτούνται 14 κύκλοι. Η επιλογή μεγαλύτερου μήκους κλειδιού έχει ως άμεση συνέπεια μεγαλύτερο επίπεδο ασφαλείας αλλά και ελαφρώς μεγαλύτερο χρόνο επεξεργασίας. Κάθε κύκλος του AES απαιτεί ένα μοναδικό 128-bit κλειδί (*round key*), το οποίο παράγεται από το αρχικό 128-bit, 192-bit ή 256-bit κλειδί, χρησιμοποιώντας μια ρουτίνα επέκτασης (*key expansion routine*).

2.1.1 Μαθηματικό υπόβαθρο

Στην υποενότητα αυτή, είναι σημαντικό να εξηγήσουμε τον τρόπο εκείνο με τον οποίο υλοποιούνται οι μαθηματικές πράξεις, πρόσθεση και πολλαπλασιασμός, για τις ανάγκες του αλγορίθμου AES. Οι πράξεις αυτές γίνονται στο πεπερασμένο πεδίο, $GF(2^8)$, και ορίζονται ως εξής.

Η πρόσθεση στο πεπερασμένο πεδίο γίνεται με τη βοήθεια της πράξης XOR ($\oplus$ -modulo 2) ούτως ώστε, $1 \oplus 1 = 0$, $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$. Για παράδειγμα,

$$\text{Δυαδική αναπαράσταση : } \{01101111\} \oplus \{11011010\} = \{10110101\}$$

$$\text{Δεκαεξαδική αναπαράσταση : } \{6F\} \oplus \{DA\} = \{B5\}$$

Η πράξη του πολλαπλασιασμού στο πεπερασμένο πεδίο αντιστοιχεί στον πολλαπλασιασμό των πολυωνύμων modulo ένα irreducible πολώνυμο 8^{ου} βαθμού. Για παράδειγμα,

$$\text{Δεκαεξαδική αναπαράσταση : } \{57\} \cdot \{83\} = \{C1\}$$

Irreducible καλείται το πολυώνυμο το οποίο δεν έχει άλλους διαιρέτες εκτός από τη μονάδα και τον εαυτό του. Στην περίπτωση του AES επελέγη το πολυώνυμο $m(x) = x^8 + x^4 + x^3 + x + 1$ ή $\{000100011011\}$ σε δυαδική ή $\{01\}\{1B\}$ σε δεκαεξαδική μορφή.

2.1.2 Η κρυπτογράφηση

Η κρυπτογράφηση, όπως και η αποκρυπτογράφηση, λαμβάνει χώρα πάνω σε ένα πίνακα *state*, όπου εφαρμόζονται όλες οι συναρτήσεις που αποτελούν τον αλγόριθμο AES. Τα αρχικά bits εισόδου μετατρέπονται σε bytes σύμφωνα με τη λογική του σχήματος 2.1 και τοποθετούνται σειριακά σε έναν πίνακα εισόδου.

Input bit sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
Byte number	0								1								2								...
Bit numbers in byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	...

Σχήμα 2.1 Μετατροπή των bits εισόδου σε bytes.

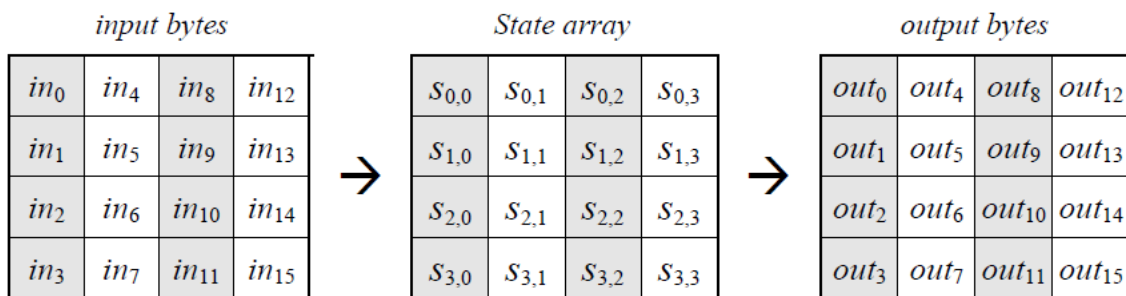
Πριν ξεκινήσει η κρυπτογράφηση, τα bytes του πίνακα εισόδου αντιγράφονται στο πίνακα *state* σύμφωνα με τη συνάρτηση

$$s[r, c] = in[r + 4c] \quad \text{όπου } 0 \leq r < 4 \text{ και } 0 \leq c < 4 \quad (2.1)$$

Εφόσον η διαδικασία έχει ολοκληρωθεί, τα bytes του πίνακα *state* αντιγράφονται σε ένα πίνακα εξόδου σύμφωνα με τη συνάρτηση

$$out[r + 4c] = s[r, c] \quad \text{όπου } 0 \leq r < 4 \text{ και } 0 \leq c < 4 \quad (2.2)$$

Η παραπάνω διαδικασία φαίνεται στο σχήμα 2.2 αναλυτικά.



Σχήμα 2.2 Μεταβάσεις από και προς το πίνακα *state*

Η διαδικασία της κρυπτογράφησης περιλαμβάνει μια σειρά από μετασχηματισμούς και μπορούν να χωριστούν σε τρεις φάσεις.

- *Initial round*
 - *AddRoundKey*

- *Rounds*
 - *SubBytes*
 - *ShiftRows*
 - *MixColumns*
 - *AddRoundKey*
- *Final round*
 - *SubBytes*
 - *ShiftRows*
 - *AddRoundKey*

Όπως αναφέρθηκε και προηγουμένως ο AES-128 ολοκληρώνεται σε 10 γύρους. Ο αρχικός κύκλος ονομάζεται κύκλος αρχικοποίησης και εφαρμόζεται στο πίνακα *state* μόνο η συνάρτηση *AddRoundKey*. Στη συνέχεια ακολουθούν 9 γύροι όπου πάνω στο πίνακα *state* εφαρμόζονται σειριακά οι μετασχηματισμοί *SubBytes*, *ShiftRows*, *MixColumns* και *AddRoundKey*. Έπειτα, ακολουθεί ο τελικός κύκλος όπου παραλείπεται ο μετασχηματισμός *MixColumns*. Τέλος, τα δεδομένα μεταβαίνουν στο πίνακα εξόδου όπως περιγράφεται στην εξίσωση 2.2. Ο αριθμός των γύρων(δεύτερη φάση) γίνεται 11 στη περίπτωση του AES-192 και 13 στην περίπτωση του AES-256.

Ο μετασχηματισμός SubBytes

Ο μετασχηματισμός αυτός εφαρμόζεται σε κάθε ένα προς ένα byte του πίνακα *state*. Χρησιμοποιεί έναν πίνακα που ονομάζεται S-Box και περιέχει αντιστοιχίες σε νέα bytes για κάθε πιθανό byte εισόδου. Έτσι, όλα τα byte του πίνακα *state* αντικαθίστανται σύμφωνα με το σχήμα 2.3

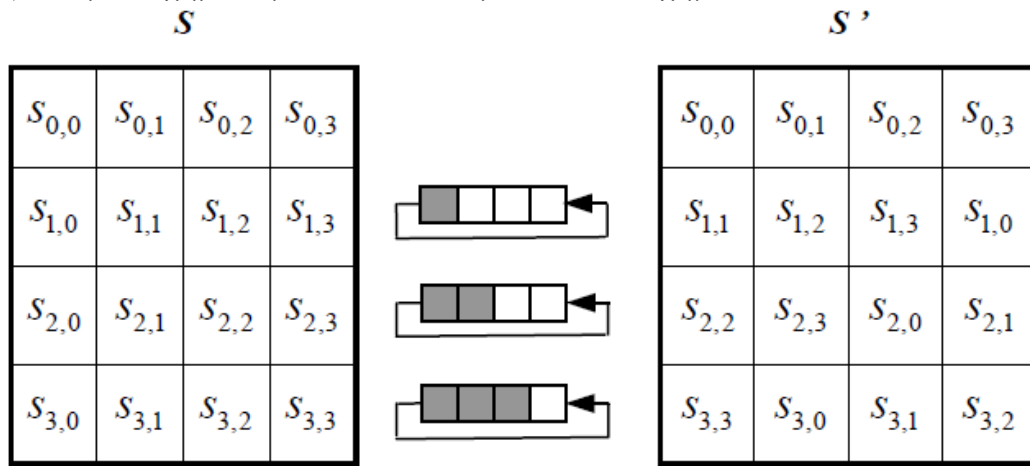
		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Σχήμα 2.3 S-BOX του μετασχηματισμού SubBytes.

Για παράδειγμα αν το $S_{0,0}$ του πίνακα state έχει τη τιμή {b9}, τότε αυτή θα άλλαζε σε {56} σύμφωνα με το παραπάνω σχήμα.

Ο μετασχηματισμός ShiftRows

Ο μετασχηματισμός ShiftRows προκαλεί κυκλική ολίσθηση των bytes στις 3 τελευταίες γραμμές του πίνακα state. Τα bytes στην δεύτερη γραμμή κινούνται κατά ένα κελί αριστερά, στην τρίτη γραμμή κατά δύο κελιά αριστερά και στην τέταρτη γραμμή κατά τρία κελιά αριστερά. Ο μετασχηματισμός ShiftRows φαίνεται στο σχήμα 2.4 πιο αναλυτικά.



Σχήμα 2.4 Κυκλική ολίσθηση των τριών τελευταίων γραμμών του πίνακα state

Ο μετασχηματισμός MixColumns

Στο μετασχηματισμό MixColumns, οι στήλες του πίνακα state αντιμετωπίζονται ως πολώνυμα. Αυτός ο μετασχηματισμός παίρνει μία προς μία τις στήλες του πίνακα state και τις πολλαπλασιάζει με ένα καθορισμένο πολώνυμο $a(x)$ modulo $x^4 + 1$. Το πολώνυμο $a(x)$ ορίζεται ως:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (2.3)$$

Ο πίνακας που προκύπτει μπορεί να εκφραστεί ως $s'(x) = a(x) \oplus S(x)$:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

όπου $0 \leq c < 4$.

Ως αποτέλεσμα αυτού του πολλαπλασιασμού, τα τέσσερα bytes της στήλης αντικαθίστανται από τα ακόλουθα:

$$S'_{0,c} = (\{02\} \cdot S_{0,c}) \oplus (\{03\} \cdot S_{1,c}) \oplus S_{2,c} \oplus S_{3,c} \quad (2.4)$$

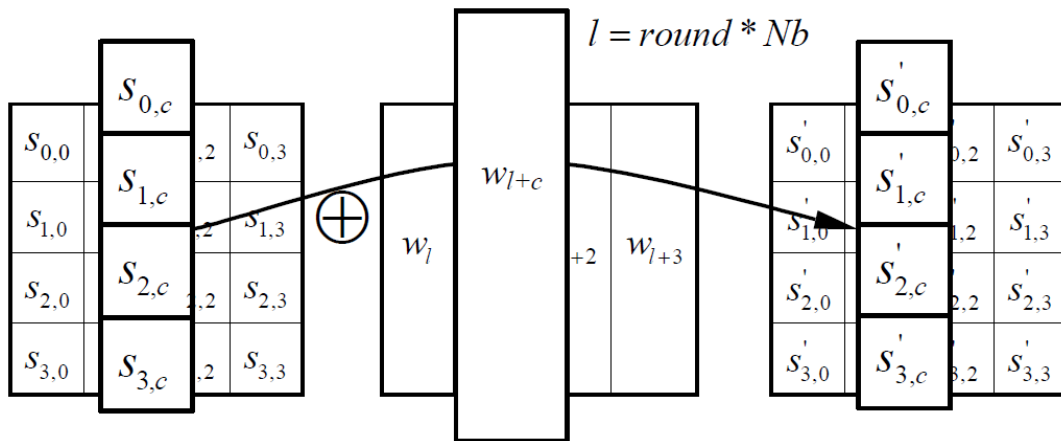
$$S'_{1,c} = S_{0,c} \oplus (\{02\} \cdot S_{1,c}) \oplus (\{03\} \cdot S_{2,c}) \oplus S_{3,c} \quad (2.5)$$

$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \cdot S_{2,c}) \oplus (\{03\} \cdot S_{3,c}) \quad (2.6)$$

$$S'_{3,c} = (\{03\} \cdot S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \cdot S_{3,c}) \quad (2.7)$$

Ο μετασχηματισμός AddRoundkey

Κατά τον μετασχηματισμό AddRoundKey κάθε στήλη του πίνακα *state* γίνεται XOR με την αντίστοιχη στήλη ενός RoundKey (κάθε 4 στήλες του πίνακα Key Schedule αποτελούν ένα RoundKey). Το κάθε RoundKey παράγεται από το προηγούμενο. Σε επόμενη υποενότητα θα εξηγήσουμε τη ρουτίνα παραγωγής κλειδιών. Η λειτουργία του μετασχηματισμού φαίνεται στο σχήμα 2.5



Σχήμα 2.5 Μετασχηματισμός AddRoundKey

2.1.3 Η αποκρυπτογράφηση

Η διαδικασία της αποκρυπτογράφησης περιλαμβάνει μια σειρά από αντίστροφους μετασχηματισμούς, από εκείνους που περιλαμβάνει η κρυπτογράφηση, και μπορούν επίσης να χωριστούν σε τρεις φάσεις.

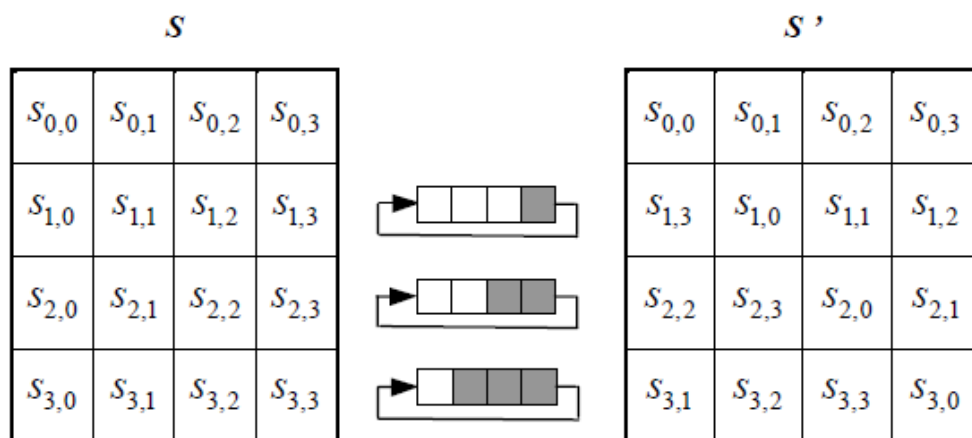
- *Initial round*
 - *AddRoundKey*

- *Rounds*
 - *InvShiftRows*
 - *InvSubBytes*
 - *AddRoundKey*
 - *InvMixColumns*
- *Final round*
 - *InvShiftRows*
 - *InvSubBytes*
 - *AddRoundKey*

Όπως και στη περίπτωση της κρυπτογράφησης, αρχικά εκτελείται το *Initial round*, στη συνέχεια ακολουθούν τα *rounds* για 9, 11 ή 13 επαναλήψεις ανάλογα με την κρυπτογράφηση που έχει επιλεγεί AES-128, AES-192 ή AES-256 αντίστοιχα. Τέλος, η διαδικασία ολοκληρώνεται με το *final round*.

Ο μετασχηματισμός InvShiftRows

Ο μετασχηματισμός InvShiftRows είναι ο αντίστροφος του μετασχηματισμού ShiftRows. Τα bytes των τριών τελευταίων γραμμών ολισθαίνουν προς τα δεξιά. Τα bytes της δεύτερης γραμμής ολισθαίνουν κατά μία θέση, εκείνα της τρίτης κατά δύο θέσεις ενώ εκείνα της τρίτης γραμμής ολισθαίνουν κατά τρεις. Ο μετασχηματισμός InvShiftRows φαίνεται καλύτερα στο σχήμα 2.6



Σχήμα 2.6 Ο μετασχηματισμός InvShiftRows

Ο μετασχηματισμός InvSubBytes

Ο μετασχηματισμός InvSubBytes αποτελεί τον αντίστροφο του μετασχηματισμού SubBytes. Ακολουθώντας την ίδια προσέγγιση λοιπόν με το μετασχηματισμό SubBytes, ο InvSubBytes αντικαθιστά κάθε byte του πίνακα *state* σύμφωνα με το S-BOX του σχήματος 2.7. Αν για παράδειγμα, έχουμε το byte $S_{0,0}$ του πίνακα *state* είναι το {56}, τότε το byte που θα το αντικαταστήσει, σύμφωνα με το σχήμα 2.7, είναι το {b9}.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Σχήμα 2.7 S-BOX του μετασχηματισμού InvSubBytes.

Ο μετασχηματισμός InvMixColumns

Ο μετασχηματισμός InvMixColumns είναι ο αντίστροφος του μετασχηματισμού MixColumns. Ο μετασχηματισμός αυτός παίρνει, μία προς μία, τις στήλες του πίνακα *state* (ως πολυώνυμα) και τις πολλαπλασιάζει με το σταθερό πολυώνυμο $a^{-1}(x)$ modulo $x^4 + 1$. Όπου $a^{-1}(x)$ έχει τη τιμή

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \quad (2.8)$$

Η παραπάνω πράξη μπορεί να γραφεί ως πολλαπλασιασμός πινάκων $S'(x) = a^{-1} \oplus S(x)$.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Ως αποτέλεσμα του παραπάνω πολλαπλασιασμού, τα τέσσερα bytes της στήλης αντικαθίστανται από τα ακόλουθα

$$S'_{0,c} = (\{0e\} \cdot S_{0,c}) \oplus (\{0b\} \cdot S_{1,c}) \oplus (\{0d\} \cdot S_{2,c}) \oplus (\{09\} \cdot S_{3,c}) \quad (2.9)$$

$$S'_{1,c} = (\{09\} \cdot S_{0,c}) \oplus (\{0e\} \cdot S_{1,c}) \oplus (\{0b\} \cdot S_{2,c}) \oplus (\{0d\} \cdot S_{3,c}) \quad (2.10)$$

$$S'_{2,c} = (\{0d\} \cdot S_{0,c}) \oplus (\{09\} \cdot S_{1,c}) \oplus (\{0e\} \cdot S_{2,c}) \oplus (\{0b\} \cdot S_{3,c}) \quad (2.11)$$

$$S'_{3,c} = (\{0b\} \cdot S_{0,c}) \oplus (\{0d\} \cdot S_{1,c}) \oplus (\{09\} \cdot S_{2,c}) \oplus (\{0e\} \cdot S_{3,c}) \quad (2.12)$$

Ο μετασχηματισμός AddRoundKey

Ο μετασχηματισμός AddRoundKey δεν διαφέρει από εκείνον που χρησιμοποιήθηκε στο πεδίο της κρυπτογράφησης δεδομένου ότι περιλαμβάνει μόνο τη πράξη XOR.

2.1.4 Key expansion routine

Ο αλγόριθμος AES χρησιμοποιεί ένα πίνακα *Cipher Key*, στον οποίο έχει τοποθετηθεί το αρχικό κλειδί K , για να παράγει όλα τα υπόλοιπα κλειδιά που χρησιμοποιούνται σε κάθε *round*. Ο πίνακας *Cipher Key* είναι ένας πίνακας 128,192 ή 256 bits. Έχει 4 γραμμές και N_k στήλες. Δηλαδή $N_k = 4, 6$ ή 8 , ανάλογα με τον αριθμό των bits του *Cipher Key* μιλάμε για τον AES-128, τον AES-192 ή τον AES-256. Ο αριθμός των επαναλήψεων του αλγορίθμου (N_r) είναι άμεση συνάρτηση του N_k . Έτσι έχουμε για $N_k = 4 \Rightarrow N_r = 10$, για $N_k = 6 \Rightarrow N_r = 12$, και για $N_k = 8 \Rightarrow N_r = 14$. Η ρουτίνα *Key Expansion* κάνοντας χρήση του *Cipher Key* παράγει τον πίνακα *Key Schedule*, δηλαδή έναν αριθμό *Round Keys*. Το *Key Schedule* είναι ένας πίνακας 4 γραμμών και $N_b \cdot (N_r + 1)$ στηλών. Κάθε 4 στήλες του πίνακα *Key Schedule* αποτελούν ένα *Round Key*. Ο πίνακας *Cipher Key* αντιμετωπίζεται από τον αλγόριθμο ως μονοδιάστατος πίνακας bytes ενώ ο *Key Schedule* ως μονοδιάστατος πίνακας λέξεων. Στο σχήμα 2.8 φαίνεται ο ψευδωδικός για τη ρουτίνα *Key Expansion*.

Οι μετασχηματισμοί που χρησιμοποιεί η ρουτίνα key expansion είναι :

- *SubWord()* → Η συνάρτηση *SubWord()* παίρνει μια λέξη τεσσάρων bytes και με εφαρμογή του S-box (που περιγράφηκε στην υποενότητα 2.1.2) σε κάθε ένα από τα 4 bytes της μας δίνει μια άλλη λέξη.
- *RotWord()* → Η συνάρτηση *RotWord()* παίρνει μια λέξη τεσσάρων bytes έστω $[\alpha_0, \alpha_1, \alpha_2, \alpha_3]$ και κάνοντας μια κυκλική μετάθεση μας δίνει τη λέξη $[\alpha_1, \alpha_2, \alpha_3, \alpha_0]$.
- *Rcon(i)* → Η μεταβλητή *Rcon(i)* είναι μια λέξη (word) της οποίας τα τρία τελευταία bytes είναι μηδενικά, και το πρώτο byte είναι x^{i-1} , όπου x είναι το $\{02\}$ στο πεπερασμένο πεδίο $GF(2^8)$. Άρα, η λέξη που αποτελεί το *Rcon(i)* κάθε φορά μπορεί να συμβολιστεί ως $[\{x^{i-1}\}, \{00\}, \{00\}, \{00\}]$.

KeyExpansion (byte key[4*Nk], word[Nb*(Nr+1)], Nk)

begin

word temp

i = 0

while (i < Nk)

w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])

i = i+1

end while

i = Nk

while (i < Nb * (Nr+1))

temp = w[i-1]

if (i mod Nk = 0)

temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]

else if (Nk > 6 and i mod Nk = 4)

temp = SubWord(temp)

end if

w[i] = w[i-Nk] xor temp

i = i + 1

end while

end

Σχήμα 2.8 Ψευδοκώδικας για τη ρουτίνα Key Expansion

Αρχικά, οι πρώτες N_k λέξεις της ρουτίνας *Key Expansion* γεμίζονται με το αρχικό κλειδί κρυπτογράφησης. Στη συνέχεια, κάθε λέξη που ακολουθεί, $w[i]$, προκύπτει από τη πράξη XOR της λέξης $w[i]$ με την λέξη $w[i-N_k]$. Για τις λέξεις που είναι σε θέσεις πολλαπλάσιες του N_k εφαρμόζονται μια σειρά από μετασχηματισμούς. Οι μετασχηματισμοί αυτοί είναι μια κυκλική ολίσθηση των bytes της λέξης (*RotWord()*) ακολουθούμενη από την αντικατάσταση των τεσσάρων bytes της λέξης (*SubWord()*). Τέλος, το αποτέλεσμα που έχει προκύψει γίνεται XOR με τη σταθερά *Rcon(i)*.

2.2 Πολλαπλασιασμός με το πρωταρχικό στοιχείο α

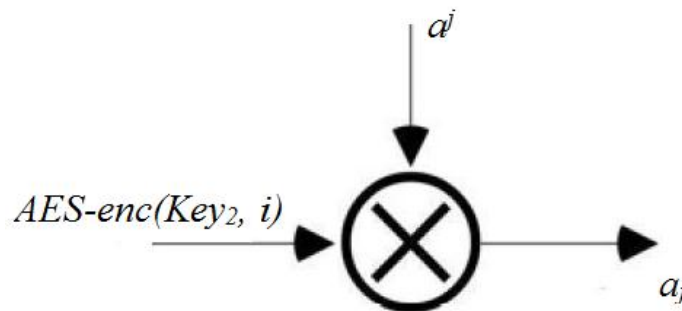
Κατά τη διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης χρησιμοποιείται το αποτέλεσμα της λογικής θέσης του *sector* (*i value*). Το αποτέλεσμα της από τη κρυπτογράφηση τη λογικής θέσης του *sector*, χρησιμοποιώντας το Key_2 ($AES-enc(Key_2, i)$), πολλαπλασιάζεται ακολούθως με την j -οστή δύναμη του a , όπου a ένα πρωταρχικό στοιχείο του πεπερασμένου πεδίου $GF(2^8)$.

Πριν γίνει ο πολλαπλασιασμός, το αποτέλεσμα της κρυπτογράφησης διαμορφώνεται ως ένας πίνακας από bytes $a_0[k]$, όπου $k = 0, 1, 2, \dots, 15$ και $a_0[0]$ είναι το πρώτο byte που προέκυψε από τη κρυπτογράφηση της λογικής θέσης.

Η διαδικασία του πολλαπλασιασμού καθορίζεται από τους επόμενους παράγοντες :

Inputs	j είναι η δύναμη του a $a_0[k], k = 0, 1 \dots 15$
Outputs	$a_j[k], k = 0, 1 \dots 15$

Σχήμα 2.9 Είσοδοι και έξοδοι του πολλαπλασιασμού με το a



Σχήμα 2.10 Πολλαπλασιασμός με το πρωταρχικό στοιχείο a .

Ο πίνακας εξόδου καθορίζεται κατ' επανάληψη από τους ακόλουθους τύπους όπου το i επαναλαμβάνεται από το 0 έως το j :

$$\alpha_{i+1}[0] \leftarrow (2(a_i[0] \bmod 128)) \oplus (135 \lfloor a_i[15]/128 \rfloor) \quad (2.13)$$

$$\alpha_{i+1}[k] \leftarrow (2(a_i[k] \bmod 128)) \oplus (\lfloor a_i[k-1]/128 \rfloor), k = 0, 1 \dots 15 \quad (2.14)$$

2.3 Η XTS-AES κρυπτογράφηση

Στην ενότητα αυτή θα περιγράψουμε την XTS-AES διαδικασία κρυπτογράφησης για ένα μπλοκ δεδομένων 128-bits. Η διαδικασία κρυπτογράφησης του XTS-AES για block των 128-bits διαμορφώνεται από την παρακάτω εξίσωση

$$C \leftarrow AES - Enc(Key, P, i, j) \quad (2.15)$$

Όπου

- Key , είναι το 256 ή 512 *bits* κλειδί του XTS-AES
- P , είναι το κείμενο(μεγέθους 128 bits) που πρόκειται να κρυπτογραφηθεί, *plaintext*
- i , είναι η 128-bits *tweak value*
- j , είναι ο ακολουθιακός αριθμός του 128-bits μπλοκ της data unit
- C , το 128-bits κρυπτογραφημένο κείμενο που προκύπτει από τη παραπάνω διαδικασία, *ciphertext*

Το κλειδί μπορεί να μοιραστεί σε δύο κλειδιά ίσου μεγέθους με όνομα Key_1 και Key_2 αντίστοιχα, έτσι ώστε: $Key = Key_1 \parallel Key_2$.

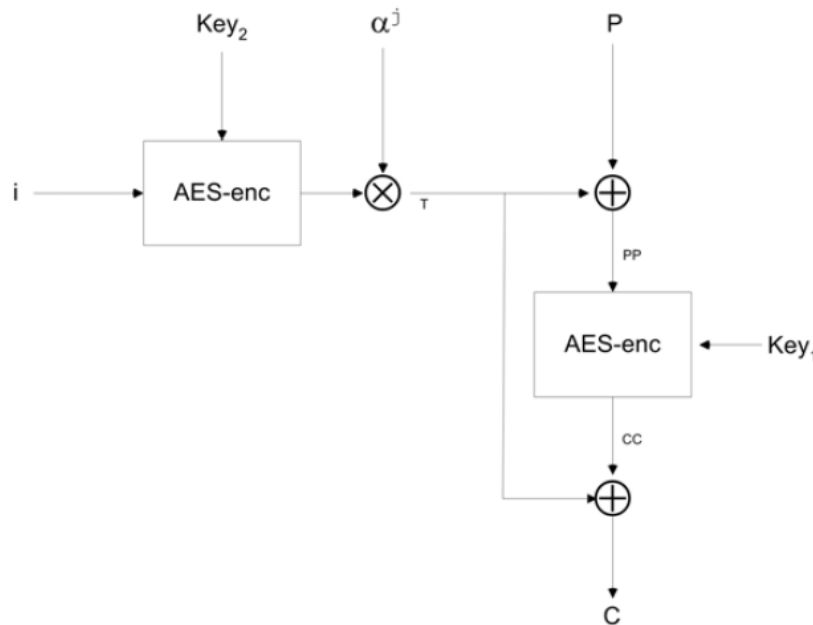
Σύμβολα

- $\oplus$ Αποκλειστικό-OR (XOR)
- $\otimes$ Modular πολλαπλασιασμός 2 πολυωνύμων στο δυαδικό πεδίο $GF(2)$, modulo $x^{128} + x^7 + x^2 + x + 1$, όπου GF το Galois Field.
- $|$ Ακολουθία (Concatenation) (π.χ., αν $K_1=0001$ και $K_2=1010$, τότε $K_1|K_2=00011010$).

Τα βήματα που πρέπει να ακολουθηθούν ούτως ώστε η διαδικασία XTS-AES κρυπτογράφησης να ολοκληρωθεί αποτελεσματικά είναι :

1. $T \leftarrow AES - enc(Key_2, i) \otimes \alpha^j$
2. $PP \leftarrow P \oplus T$
3. $CC \leftarrow AES - enc(Key_1, PP)$
4. $C \leftarrow CC \oplus T$

Το *ciphertext* υπολογίζεται με βάση το παρακάτω σχήμα :



Σχήμα 2.11 Διάγραμμα XTS-AES κρυπτογράφησης

Στο πρώτο βήμα κρυπτογραφείται η λογική θέση του *sector* και παράγεται η *tweak value*. Στο δεύτερο βήμα γίνεται μια πράξη XOR της *tweak value* με το *plaintext*. Στη συνέχεια, το αποτέλεσμα του βήματος δύο κρυπτογραφείται και το αποτέλεσμα αυτού με τη σειρά του γίνεται XOR με τη *tweak value* ούτως ώστε να παραχθεί το *ciphertext*

2.4 Η XTS-AES αποκρυπτογράφηση

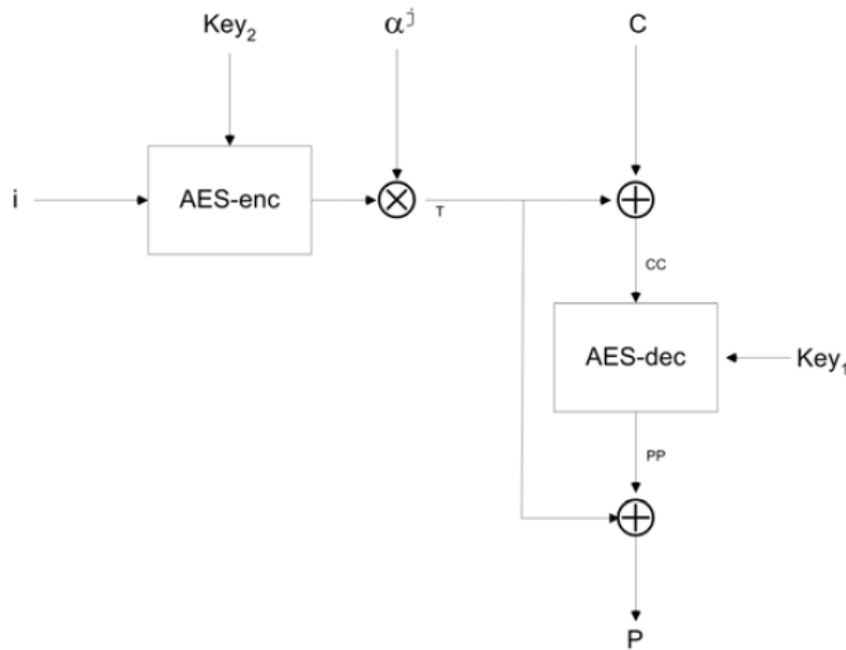
Στην ενότητα αυτή θα περιγράψουμε την XTS-AES διαδικασία αποκρυπτογράφησης για ένα μπλοκ δεδομένων 128-bits. Η διαδικασία αποκρυπτογράφησης του XTS-AES για block των 128-bits διαμορφώνεται από την παρακάτω εξίσωση

$$P \leftarrow AES - Dec(Key, C, i, j) \quad (2.16)$$

Όπου

- *Key*, είναι το 256 ή 512 *bits* κλειδί του XTS-AES
- *C*, είναι το κείμενο(μεγέθους 128 bits) που πρόκειται να αποκρυπτογραφηθεί, *ciphertext*
- *i*, είναι η τιμή της 128-bits *tweak value*
- *j*, είναι ο ακολουθιακός αριθμός του 128-bits μπλοκ της data unit
- *P*, το 128-bits αποκρυπτογραφημένο κείμενο που προκύπτει από τη παραπάνω διαδικασία, *plaintext*

Το κλειδί, Key , που χρησιμοποιείται και εδώ, μπορεί να θεωρηθεί ως μια ακολουθία δύο κλειδιών, Key_1 και Key_2 , όπου $Key = Key_1 \parallel Key_2$. Η διαδικασία της XTS-AES αποκρυπτογράφησης μπορεί να γίνει περισσότερο κατανοητή αν παρατηρήσουμε το σχήμα 2.12.



Σχήμα 2.12 Διάγραμμα XTS-AES αποκρυπτογράφησης

Η διαδικασία της XTS-AES αποκρυπτογράφησης περιγράφεται αναλυτικά ακολουθώντας κατά σειρά τα παρακάτω βήματα.

1. $T \leftarrow AES - enc(Key_2, i) \otimes \alpha^j$
2. $CC \leftarrow C \oplus T$
3. $PP \leftarrow AES - dec(Key_1, CC)$
4. $P \leftarrow PP \oplus T$

Η διαδικασία της XTS-AES αποκρυπτογράφησης είναι ουσιαστικά η αντίστροφη διαδικασία από αυτήν που περιγράψαμε στη προηγούμενη ενότητα. Αρχικά, υπολογίζεται η *tweak value*, η οποία είναι ουσιαστικά η κρυπτογραφημένη τιμή της λογικής θέσης του *sector* που είναι αποθηκευμένη η πληροφορία. Στη συνέχεια, η *tweak value* γίνεται XOR με το *ciphertext* και έπειτα κρυπτογραφείται. Το αποτέλεσμα της κρυπτογράφησης γίνεται XOR με τη *tweak value* και παράγεται το αρχικό *plaintext*.

2.5 Εφαρμογή του XTS-AES

Οι διαδικασίες κρυπτογράφησης και αποκρυπτογράφησης που περιγράφονται στις υποενότητες 2.3 και 2.4 έχουν ως βάση τον αλγόριθμο *AES*. Εάν το *XTS-AES* key είναι 256-bits, οι διαδικασίες χρησιμοποιούν τον 128-bit *AES*. Εάν το *XTS-AES* key είναι 512-bits, οι διαδικασίες χρησιμοποιούν τον 256-bit *AES*. Ο πρώτος τρόπος θα αναφέρεται ως *XTS AES-128* και ο δεύτερος ως *XTS-AES-256*. Για να υπάρχει συμβατότητα με το πρότυπο, οποιαδήποτε υλοποίηση θα πρέπει να υποστηρίζει τουλάχιστον μία από τις δύο λειτουργίες που μόλις αναφέρθηκαν.

Το *Key scope* (βλέψη ενός κλειδιού) καθορίζει το εύρος των δεδομένων που κρυπτογραφούνται με ένα *XTS-AES* κλειδί. Το *Key scope* αντιπροσωπεύεται από τρεις ακέραιους: Πρώτον, την τιμή της *tweak value* η οποία συσχετίζεται με την 1^η *data unit* (από την ακολουθία των *data units* που κρυπτογραφήθηκαν από αυτό το κλειδί), δεύτερον, το μέγεθος σε bits που διαθέτει η κάθε *data unit* και τρίτο, τον αριθμό των *data units* που κρυπτογραφούνται/αποκρυπτογραφούνται με αυτό το κλειδί.

Μια υλοποίηση αυτού του προτύπου μπορεί (χωρίς να είναι υποχρεωτικό) να υποστηρίζει πολλαπλάσια μεγέθη *data units*. Σε μια εφαρμογή αυτού του προτύπου σε *sector-level* κρυπτογράφηση ενός δίσκου, η *data unit* αντιστοιχεί σε ένα *logical block* του δίσκου. Το *Key scope* περιλαμβάνει ένα εύρος από διαδοχικά *logical blocks* του δίσκου, και η τιμή της *tweak value* που συσχετίζεται με την πρώτη *data unit* αντιστοιχεί στο *Logical Block Address* (LBA) το οποίο είναι το πρώτο *logical block*. Ένα *XTS-AES* κλειδί δεν θα πρέπει να αντιστοιχεί σε περισσότερα από ένα *key scope*.

Ο λόγος του ανωτέρω περιορισμού είναι ότι η κρυπτογράφηση περισσότερων από ένα block δεδομένων με το ίδιο κλειδί και τον ίδιο δείκτη εισάγει κενά ασφαλείας που ενδεχομένως να χρησιμοποιηθούν σε μια επίθεση στο σύστημα. Ειδικότερα, με αυτό τον τρόπο αποφεύγουμε επιθέσεις του τύπου *cut-and-paste*.

2.6 Υφιστάμενη κατάσταση του IEEE P1619

Την στιγμή που γράφεται το παρόν κείμενο η SISWG αναφέρει τέσσερις εκδοχές του προτύπου IEEE P1619

1. IEEE P1619 Narrow-Block Encryption.

Η ομάδα επιστημόνων που έχει αναλάβει το IEEE P1619 εργάζεται στον αλγόριθμο κρυπτογράφησης *XTS* και στη διαμόρφωση ενός προτύπου εξαγωγής των κλειδιών βασισμένο στη γλώσσα XML.

2. IEEE P1619.1 Authenticated Encryption. (GCM, CCM, CBC-HMAC, XTS-HMAC). Για το IEEE P1619.1 αναπτύσσεται ένα πρότυπο κρυπτογράφησης με αλγορίθμους που χρησιμοποιούν AES-256 κρυπτογράφηση:

- CCM-128-AES-256
- GCM-128-AES-256
- CBC-AES-256-HMAC-SHA.
- XTS-AES-256-HMAC-SHA.

3. IEEE P1619.2 Wide-Block Encryption (TET, EME*, XCB...).

Σε αυτό το πρότυπο προδιαγράφεται μια αρχιτεκτονική κρυπτογράφησης για δεδομένα σε τυχαίας προσπέλασης συσκευές αποθήκευσης με block size μεγαλύτερο ή ίσο με 512 bytes

4. IEEE P1619.3: Key Management

Αυτό το πρότυπο προδιαγράφει μια αρχιτεκτονική για τη διαχείριση του κλειδιού. Καθορίζονται μέθοδοι για την αποθήκευση, τη διαχείριση και την διανομή των κρυπτογραφικών κλειδιών

Κεφάλαιο 3

Χρονοπρογραμματισμός και προτεινόμενες αρχιτεκτονικές

Ένα από τα πιο σημαντικά σημεία της σχεδίασης ψηφιακών κυκλωμάτων συναντάται τη στιγμή που σχεδιάζεται ο χρονοπρογραμματισμός (*scheduling*) ή διαφορετικά η χρονοδρομολόγηση του κυκλώματος καθώς και οι αρχιτεκτονικές που τον υιοθετούν ούτως ώστε να παραχθεί το τελικό κύκλωμα.

Ο χρονοπρογραμματισμός παρουσιάζει αναλυτικά όλες εκείνες τις διεργασίες που πρέπει να ακολουθηθούν κατά σειρά και με συνέπεια καθώς και στον σωστά ορισμένο χρόνο έτσι ώστε το κύκλωμα να λειτουργεί αποτελεσματικά. Κύριο μέλημα του χρονοπρογραμματισμού είναι η βελτιστοποίηση των ήδη υπαρχουσών που έχει ως στόχο την αποτελεσματικότερη και πιο γρήγορη υλοποίηση της όλης διαδικασίας ούτως ώστε να επιτευχθούν καλύτερα αποτελέσματα (μεγαλύτερο *throughput*, μικρότερη επιφάνεια, κ.α.).

Επίσης, πολύ σημαντικό ρόλο διαδραματίζουν οι αρχιτεκτονικές που υιοθετήθηκαν μέσω του χρονοπρογραμματισμού αυτού και κατά πόσο αξιοποιούν τον ίδιο τον χρονοπρογραμματισμό ούτως ώστε να παράγουν στο μικρότερο δυνατό χρόνο το σωστό αποτέλεσμα. Ακόμη, σημαντικό παράμετρο σε μια αρχιτεκτονική, πέρα από το *throughput* και την επιφάνεια, παίζει το *latency* (αρχικός χρόνος προετοιμασίας του κυκλώματος για να λειτουργεί αποτελεσματικά), καθώς και οι ενδιάμεσες καθυστερήσεις οι οποίες ιδανικά θα πρέπει να είναι μηδενικές.

Σε αυτό το κεφάλαιο θα ασχοληθούμε αναλυτικά με τον χρονοπρογραμματισμό εκείνο που έχουν υιοθετήσει οι περισσότεροι που έχουν ασχοληθεί με το πρότυπο P1619[14], θα προτείνουμε τη δικιά μας εκδοχή και άποψη πάνω στο συγκεκριμένο θέμα και ακόμη θα προβούμε σε μια ποιοτική σύγκριση μεταξύ των δύο προσεγγίσεων ούτως ώστε να δούμε ποια υπερτερεί και για ποιους λόγους. Στη συνέχεια, θα μιλήσουμε για τη λογική των αρχιτεκτονικών που υλοποιήθηκε καθώς και για τις επιμέρους αρχιτεκτονικές με βάση των αριθμό των πυρήνων AES που εμπεριέχουν. Ακόμη, θα αναφερθούμε στη λογική των μονάδων ελέγχου που χρησιμοποιήθηκαν και στο τρόπο με τον οποίο σχεδιάστηκαν. Τέλος, γίνεται μια ανάλυση των στοιχείων που χρησιμοποιήθηκαν και εξηγούμε κατά πόσο είναι εύκολη η μετάβαση από την αρχιτεκτονική δύο πυρήνων σε εκείνη των τριών και των τεσσάρων.

3.1 Χρονοπρογραμματισμός

Όπως αναφέρθηκε και στην εισαγωγή του κεφαλαίου 3 προηγουμένως, ο χρονοπρογραμματισμός αναφέρεται σε όλες εκείνες τις διεργασίες που πρέπει να ακολουθηθούν πιστά ούτως ώστε να επιτευχθεί η αποτελεσματική λειτουργία του κυκλώματος. Σε αυτή την ενότητα παρουσιάζονται ο υπάρχον χρονοπρογραμματισμός που έχει υιοθετηθεί σε πολλές εργασίες που ασχολήθηκαν με το πρότυπο P1619 καθώς και ο προτεινόμενος χρονοπρογραμματισμός που εμείς υιοθετήσαμε θεωρώντας ότι είναι ικανός να παραγάγει καλύτερα αποτελέσματα.

3.1.1 Υπάρχον χρονοπρογραμματισμός

Με τον όρο υπάρχον χρονοπρογραμματισμός αναφερόμαστε στο χρονοπρογραμματισμό εκείνο που έχει ακολουθηθεί κατά κύρια βάση στις περισσότερες προσπάθειες που έχουν προταθεί όσον αφορά το πρότυπο P1619[15]. Σε αυτήν την υποενότητα εξετάζεται και μελετάται αναλυτικά η λειτουργία της συγκεκριμένης λογικής και παρουσιάζονται τα μειονεκτήματα που εμπεριέχει. Τα μειονεκτήματα αυτά κατά κύριο λόγο μας οδήγησαν στη προσπάθεια να υιοθετήσουμε μια διαφορετική προσέγγιση η οποία ως στόχο είχε να καλυφθούν αυτά τα μειονεκτήματα και να μας οδηγήσουν σε μια πιο αποδοτική υλοποίηση.

Στο πίνακα 3.1 παρουσιάζονται αναλυτικά ο χρονοπρογραμματισμός, τόσο της λογικής εκείνης που υιοθετεί δύο πυρήνες AES (3.1α) όσο και εκείνης που υιοθετεί τέσσερις (3.1β). Ας δούμε πρώτα τι σημαίνουν τα σύμβολα που παρουσιάζονται στο πίνακα 3.1 και στη συνέχεια θα παρουσιάσουμε βήμα προς βήμα τη λογική που ακολουθείται και στις δύο περιπτώσεις.

Arc. 2 cores		
τ	C1	C2
1	T1	
2	1/32	2/32
3	3/32	4/32
.	.	.
.	.	.
17	31/32	32/32
18	T2	
19	1/32	2/32
.	.	.
.	.	.
34	31/32	32/32
35	T3	
36	1/32	2/32
.	.	.
.	.	.
51	31/32	32/32
52	T4	
53	1/32	2/32
.	.	.
.	.	.
.	.	.

3.1α

Arc. 4 cores				
τ	C1	C2	C3	C4
1	T1			
2	1/32	2/32	3/32	4/32
3	5/32	6/32	7/32	8/32
.	.	.	.	.
.	.	.	.	.
9	29/32	30/32	31/32	32/32
10	T2			
11	1/32	2/32	3/32	4/32
.	.	.	.	.
.	.	.	.	.
18	29/32	30/32	31/32	32/32
19	T3			
20	1/32	2/32	3/32	4/32
.	.	.	.	.
.	.	.	.	.
27	29/32	30/32	31/32	32/32
28	T4			
29	1/32	2/32	3/32	4/32
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

3.1β

Πίνακας 3.1 Υπάρχον χρονοπρογραμματισμός αρχιτεκτονικών 2, 3 και 4 πυρήνων αντίστοιχα.

Οι πόροι οι οποίοι είναι διαθέσιμα για την αρχιτεκτονική δύο πυρήνων είναι οι C1 και C2 (*core1* και *core2*) οι οποίοι, όπως γίνεται εύκολα αντιληπτό, αναλαμβάνουν να κρυπτογραφήσουν/αποκρυπτογραφήσουν τα μπλοκ δεδομένα. Σε αυτό το σημείο θα πρέπει να αποσαφηνιστεί ότι το σύμβολο τ δεν αποτελεί κάποιο σήμα χρονισμού (*clock*) αλλά το χρόνο εκείνο που χρειάζεται ένα μπλοκ δεδομένων για να κρυπτογραφηθεί. Αποτελεί δηλαδή μια σειρά από σήματα *clock*.

Όπως έχει αναφερθεί και στο προηγούμενο κεφάλαιο, η κρυπτογράφηση ή η αποκρυπτογράφηση γίνεται σε επίπεδο τομέα (*sector-based*). Απαραίτητη προϋπόθεση για να

ξεκινήσει η κρυπτογράφηση/αποκρυπτογράφηση είναι να έχει πρώτα υπολογιστεί η *tweak value*. Με το σύμβολο T ονοματίζονται οι *tweak values*, T_1 η πρώτη *tweak value*, T_2 δεύτερη *tweak value* και αναλόγως ονοματίζονται και οι υπόλοιπες. Η κάθε *tweak value* τροποποιείται μέσω ενός *GFMC* (*galois field multiplication core*) ούτως ώστε σε κάθε μπλοκ του *sector* να αντιστοιχίζεται μια μοναδική τιμή. Η λογική του *GFMC* και των *tweak values* θα περιγραφεί αναλυτικά σε επόμενο κεφάλαιο. Ως *tweak value* αναφερόμαστε στο αποτέλεσμα της κρυπτογράφησης της λογική θέσης του *sector* (*i value*).

Κάθε *tweak value* ανατίθεται σε ένα και μοναδικό *sector* και μπορεί να χρησιμοποιηθεί στη καλύτερη περίπτωση για τριάντα δύο (32) μπλοκ δεδομένων (για ολόκληρο δηλαδή το *sector*). Γνωρίζοντας ότι το μέγεθος του κάθε μπλοκ δεδομένων έχει οριστεί στα 128 bits, ο μέγιστος αριθμός δεδομένων που μπορεί να εμπεριλάβει η κάθε *tweak value* ορίζεται στα 512 bytes ή 4096 bits.

Η διαδοχική διαδικασία της κρυπτογράφησης/αποκρυπτογράφησης των μπλοκ δεδομένων φαίνεται αναλυτικά στο πίνακα 3.1, όπου η αρίθμηση των μπλοκ δεδομένων ξεκινούν από το 1 και φτάνουν μέχρι το 32. Το πρώτο μπλοκ δεδομένων συμβολίζεται ως 1/32, το δεύτερο ως 2/32 και ομοίως συμβολίζονται και τα υπόλοιπα. Η κρυπτογράφηση/αποκρυπτογράφηση που περιγράφεται στο πίνακα 3.1 αφορά αρκετά *sectors* και αποτελεί το γενικό χρονοπρογραμματισμό.

Στο πίνακα 3.1α παρατηρούμε ότι ο χρόνος που χρειάζεται να ολοκληρωθεί η διαδικασία της κρυπτογράφησης/αποκρυπτογράφησης ενός *sector* είναι σταθερός για όλα τα *sector*. Άρα, το *throughput* μπορεί να υπολογιστεί για ένα *sector*. Ο χρόνος που απαιτείται για να ολοκληρωθεί είναι 17τ . Η αριθμητική τιμή του *throughput* (ρυθμός διαμεταγωγής) μπορεί να υπολογιστεί με βάση τη παρακάτω εξίσωση

$$Throughput = \frac{(bits \text{ που επεξεργάστηκαν}) \cdot (Μέγιστη \text{ συχνότητα λειτουργίας})}{Κύκλοι \text{ ρολογιού που χρειάστηκαν}} \quad 3.1$$

Τα δεδομένα που επεξεργάστηκαν σε ένα *sector* είναι 4096 bits και οι κύκλοι ρολογιού που χρειάστηκαν είναι $17\tau = 17 \times 12 = 204$. Αν υποθέσουμε ότι η μέγιστη συχνότητα λειτουργίας είναι 100MHz, τότε το *throughput* υπολογίζεται περίπου στα 2Gbps το οποίο κρίνεται ικανοποιητικό και αφορά την αρχιτεκτονική δύο πυρήνων. Στη περίπτωση των τεσσάρων πυρήνων ο απαιτούμενος χρόνος μειώνεται σε $9\tau = 9 \times 12 = 108$ κύκλους ρολογιού και το *throughput* ανέρχεται στα 3.8Gbps. Θεωρητικά λοιπόν υπάρχει περίπου 90% αύξηση στο *throughput* αλλά ανάλογη θα πρέπει να θεωρείται και η αύξηση της επιφάνειας καθώς οι πυρήνες έχουν διπλασιαστεί. Το βασικό μειονέκτημα αυτής της προσέγγισης, τόσο της αρχιτεκτονικής δύο πυρήνων όσο και εκείνης των τεσσάρων, είναι ότι υπάρχουν διαστήματα κατά τα οποία οι περισσότεροι πόροι (μονάδες κρυπτογράφησης) μένουν ανενεργοί.

Υπενθυμίζουμε σε αυτό το σημείο ότι ο συγκεκριμένος χρονοπρογραμματισμός επιλέγει τον αριθμό των πυρήνων AES σύμφωνα με τις δυνάμεις δύο. Οι αρχιτεκτονικές δηλαδή που μπορεί να υλοποιηθούν με τη συγκεκριμένη λογική είναι των 2, 4, 8, 16 και στην πιο εξεζητημένη περίπτωση εκείνη των 32 πυρήνων. Αυτό προκύπτει από το γεγονός ότι ο

μέγιστος αριθμός μπλοκ που μπορεί να κρυπτογραφηθεί με την ίδια *tweak value* δε μπορεί να υπερβεί τα τριάντα δύο (32) μπλοκ όπου και αυτός ο αριθμός αποτελεί δύναμη του δύο. Κατά συνέπεια, όπως γίνεται εύκολα κατανοητό, αυτή η λογική προσφέρει μια κάποιου είδους ομοιομορφία που μπορεί, κατά μια έννοια, να μεταφραστεί και ως μια προσπάθεια να γίνει ευκολότερη η μετάβαση από τη λογική στην υλοποίηση.

3.1.2 Προτεινόμενος χρονοπρογραμματισμός

Στην υποενότητα αυτή θα ασχοληθούμε με τη δικιά μας προσέγγιση και άποψη για το θέμα του χρονοπρογραμματισμού. Θα μιλήσουμε αναλυτικά για τη λογική που επιλέξαμε και θα προβούμε στην απαραίτητα σύγκριση, η οποία είναι αναγκαία, ώστε να καταλήξουμε ποιος χρονοπρογραμματισμός υπερτερεί και για ποιους λόγους.

Όπως φαίνεται στο πίνακα 3.2, περιγράφονται τρεις αρχιτεκτονικές. Στο πίνακα 3.2α φαίνεται η αρχιτεκτονική δύο πυρήνων, στο πίνακα 3.2β η αρχιτεκτονική τριών πυρήνων και στο πίνακα 3.2γ η αρχιτεκτονική τεσσάρων πυρήνων. Όπως έγινε γρήγορα αντιληπτό στη δικιά μας προσέγγιση έχει υιοθετηθεί και αρχιτεκτονική τριών πυρήνων και αυτό συμβαίνει διότι δεν υπάρχει κάποιος περιορισμός στον αριθμό των πυρήνων που μπορούν να χρησιμοποιηθούν. Αυτή η προσέγγιση μπορεί να υιοθετήσει N -πυρήνες, όπου N ο αριθμός των πυρήνων που κυμαίνεται από 1 έως 32 στη καλύτερη περίπτωση. Ένα σημαντικό πλεονέκτημα αυτού του χρονοπρογραμματισμού λοιπόν, σε σύγκριση με το προηγούμενο, είναι ότι μπορεί να υιοθετεί N αριθμό πυρήνων.

Το βασικό μειονέκτημα του προηγούμενου χρονοπρογραμματισμού ήταν ότι υπήρχαν κάποια “νεκρά” χρονικά διαστήματα κατά τη λειτουργία του σε κάποιους πυρήνες. Αυτά τα διαστήματα παρουσιάζονται κατά τη διάρκεια του υπολογισμού των *tweak values*. Αυτό συμβαίνει διότι σε κάθε *sector* ανατίθεται μια *tweak value* η οποία υπολογίζεται μόλις ολοκληρωθεί το προηγούμενο *sector*. Αυτό συνέβαινε διότι κατά τη διάρκεια παραγωγής των *tweak values* έχει γίνει η προσέγγιση ότι δε μπορούν να επεξεργάζονται δεδομένα

Η δικιά μας προσέγγιση έχει υιοθετήσει τη λογική εκείνη που αξιοποιεί καλύτερα τις *tweak values* υπολογίζοντας αυτές νωρίτερα. Ας δούμε όμως αναλυτικά τον τρόπο με τον οποίο υπολογίζονται οι *tweak values* και στις τρεις αρχιτεκτονικές. Στην αρχιτεκτονική των δύο πυρήνων οι *tweak values* υπολογίζονται ανά δύο ούτως ώστε να καλύπτονται τα κενά πλήρως. Στην αρχιτεκτονική τριών πυρήνων, όπου υπάρχει ένα κενό, μπορεί να χρησιμοποιηθεί για να υπολογίσει την *tweak value* που θα ανατεθεί στο επόμενο *sector*. Τέλος, στην αρχιτεκτονική τεσσάρων πυρήνων, οι *tweak values* παράγονται ανά τέσσερις ούτως ώστε όλοι διαθέσιμοι πόροι να παραμένουν σε λειτουργία.

Με αυτόν τον τρόπο καλύπτονται τα κενά διαστήματα του προηγούμενου χρονοπρογραμματισμού. Αυτή η προσέγγιση έχει άμεσο αντίκτυπο στο *throughput* καθώς στον ίδιο χρόνο επεξεργάζονται περισσότερα μπλοκ δεδομένων. Για παράδειγμα, αν δούμε αναλυτικά τη διαδικασία, στις αντίστοιχες αρχιτεκτονικές των δύο πυρήνων, σε ένα βάθος 20

sector τότε η δικιά μας προσέγγιση έχει επεξεργαστεί αυτά τα δεδομένα σε $331\tau = 331 \times 12 = 3972$ κύκλους ρολογιού ενώ, σύμφωνα με την προηγούμενη προσέγγιση, τα ίδια δεδομένα έχουν επεξεργαστεί σε $340\tau = 340 \times 12 = 4080$ κύκλους ρολογιού. Τα δεδομένα που αντιστοιχούν σε 20 *sectors* είναι 81.920 bits ή αλλιώς 10,24 KBytes. Αν υποθέσουμε ότι η μέγιστη συχνότητα λειτουργίας και των δύο προσεγγίσεων είναι τα 100MHz τότε το νέο *throughput* που επιτυγχάνεται είναι περίπου 2.06Gbps, ενώ σύμφωνα με την υπάρχουσα το *throughput* μπορεί να φτάσει περίπου στα 2Gbps.

Αυτή η εικόνα που παρουσιάζει η δικιά μας προσέγγιση μπορεί να θεωρηθεί ως μια βελτίωση κατά 3% της προηγούμενης. Η βελτίωση αυτή είναι πολύ σημαντική καθώς με μια αλλαγή στο χρονοπρογραμματισμό επιτυγχάνεται μια αξιοσημείωτη αύξηση στο *throughput*.

Arc. 2 cores			Arc. 3 cores				Arc. 4 cores				
τ	C1	C2	τ	C1	C2	C3	τ	C1	C2	C3	C4
1	T1		1	T1			1	T1			
2	1/32	2/32	2	1/32	2/32	3/32	2	1/32	2/32	3/32	4/32
.	.	.	.	.	.	.	.	.	.	.	.
16	29/32	30/32	12	T2	31/32	32/32	7	T2	21/32	22/32	23/32
17	T2	31/32	13	1/32	2/32	3/32	8	T3	24/32	25/32	26/32
18	T3	32/32	14	4/32	5/32	6/32	9	T4	27/32	28/32	29/32
19	1/32	2/32	15	7/32	8/32	9/32	10	T5	30/32	31/32	32/32
.	.	.	.	.	.	.	.	.	.	.	.
34	31/32	32/32	22	28/32	29/32	30/32	19	29/32	30/32	31/32	32/32
35	1/32	2/32	23	T3	31/32	32/32	20	1/32	2/32	3/32	4/32
36	3/32	4/32	24	1/32	2/32	3/32	21	5/32	6/32	7/32	8/32
.	.	.	.	.	.	.	.	.	.	.	.
50	T4	31/32	34	T4	31/32	32/32	27	29/32	30/32	31/32	32/32
51	T5	32/32	35	1/32	2/32	3/32	28	1/32	2/32	3/32	4/32
53	1/32	2/32	36	4/32	5/32	6/32	29	5/32	6/32	7/32	8/32
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.

3.2α

3.2β

3.2γ

Πίνακας 3.2 Προτεινόμενος χρονοπρογραμματισμός αρχιτεκτονικών 2, 3 και 4 πυρήνων αντίστοιχα.

Η βελτίωση του *throughput* μπορεί να φτάσει σε ακόμη μεγαλύτερο ποσοστό αν κάνουμε την ίδια προσέγγιση και σύγκριση για την αρχιτεκτονική των τεσσάρων πυρήνων. Η δικιά μας προσεγγίζει μπορεί να φτάσει το ρυθμό διαμεταγωγής στο επίπεδο του 4.1Gbps ενώ η υφιστάμενη φτάνει περίπου τα 3.8Gbps. Η θεωρητική αύξηση που επετεύχθη αγγίζει το ποσοστό του 7.8%. Το ποσοστό αυτό θεωρείται κάτι παραπάνω από ικανοποιητικό και δείχνει θεωρητικά την υπεροχή της δικιά μας πρότασης σε σύγκριση με την υφιστάμενη, τόσο στην αρχιτεκτονική των δύο πυρήνων όσο και σε εκείνη των τεσσάρων.

Επίσης, η αρχιτεκτονική των τριών πυρήνων, η οποία δεν έχει μέτρο σύγκρισης, μπορεί να φτάσει ρυθμούς διαμεταγωγής της τάξης των 3.08Gbps. Πετυχαίνει δηλαδή μια αύξηση του ρυθμού διαμεταγωγής δεδομένων της τάξης του 50% περίπου σε σύγκριση με εκείνη των δύο πυρήνων. Η επιφάνεια της όμως αυξάνεται και εκείνη με τη σειρά της κατά 50% περίπου. Η αρχιτεκτονική των τριών πυρήνων προκρίνεται ως μια λύση ισορροπίας ανάμεσα σε ταχύτητα και επιφάνεια.

Καταλήγοντας, παρατηρούμε ότι ο χρονοπρογραμματισμός που προτείνεται από τη δική μας πλευρά μπορεί να πετύχει καλύτερα αποτελέσματα (θεωρητικά πάντα) σε σύγκριση με την υφιστάμενη πρόταση. Αυτό που μένει πλέον, είναι να επιβεβαιωθεί αυτή η υπεροχή και στα πειραματικά αποτελέσματα τα οποία παρουσιάζονται στο πέμπτο κεφάλαιο.

3.2 Προτεινόμενες αρχιτεκτονικές

Στην προηγούμενη ενότητα περιγράψαμε αναλυτικά τους δύο χρονοπρογραμματισμούς, υπάρχον και προτεινόμενος, και προβήκαμε σε μια ποιοτική σύγκριση αυτών. Στη παρούσα ενότητα θα ασχοληθούμε με τις αρχιτεκτονικές που υλοποιήθηκαν με βάση πάντα τον προτεινόμενο χρονοπρογραμματισμό. Ακόμη περιγράφονται αναλυτικά όλα εκείνα τα στοιχεία που είναι απαραίτητα για τη σύνθεση της κάθε αρχιτεκτονικής καθώς η γενική δομή που ακολουθούν όλες οι αρχιτεκτονικές. Τέλος, γίνεται ιδιαίτερη μνεία στην ανάλυση των μονάδων ελέγχου της κάθε αρχιτεκτονικής.

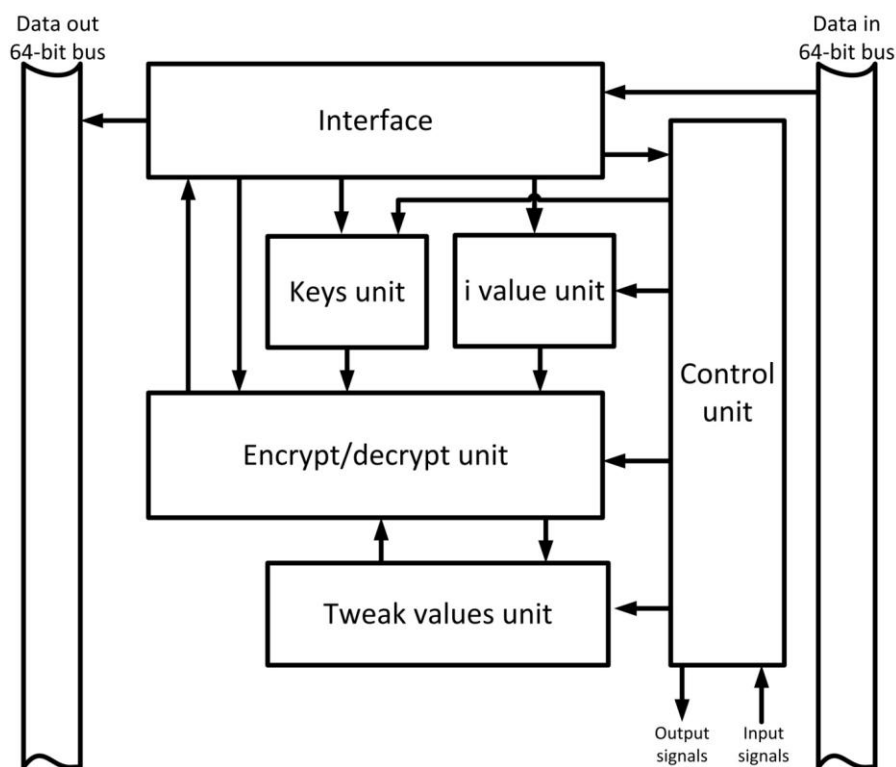
3.2.1 Αρχιτεκτονική δύο πυρήνων

Η αρχιτεκτονική ενός κυκλώματος αποτελεί ένα από τα σημαντικότερα στάδια, αν όχι το σημαντικότερο, στη σύνθεση ψηφιακών κυκλωμάτων. Σε αυτήν την υποενότητα αναλύεται το γενικό σχεδιάγραμμα όλων των αρχιτεκτονικών καθώς και η αρχιτεκτονική που χρησιμοποιεί δύο πυρήνες.

Στο σχήμα 3.1 παρουσιάζεται η γενική δομή που ακολουθούν όλες οι αρχιτεκτονικές που υλοποιήθηκαν στα πλαίσια αυτής της διπλωματικής εργασίας. Ξεκινώντας την ανάλυση του παραπάνω σχήματος, σημαντικό είναι να αναφέρουμε μερικές παραδοχές που έχουν γίνει. Η φόρτωση των δεδομένων, είτε αυτά δεδομένα προς κρυπτογράφηση ή

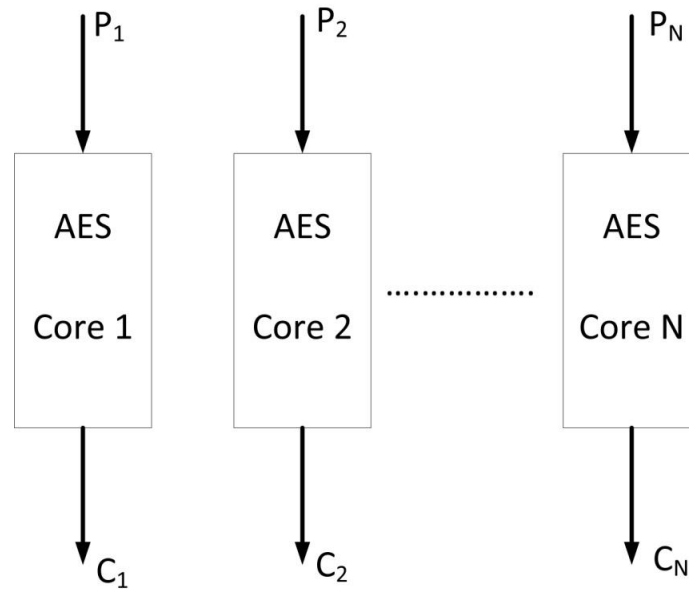
αποκρυπτογράφηση (*plaintext/ciphertext*), είτε είναι τα κλειδιά (*key1/key2*) τα οποία είναι απαραίτητα, είτε τέλος είναι η λογική θέση των *sectors* (*i values*) από τις οποίες παράγονται οι *tweak values*, γίνεται αποκλειστικά από ένα διαυλο δεδομένων μήκους 64-bits. Όπως φαίνεται στο σχήμα 3.1, μπορεί να γίνει κατακερματισμός της παραπάνω αρχιτεκτονικής σε έξι επιμέρους στοιχεία (components) συν τους διαύλους δεδομένων, εισόδου και εξόδου, των 64-bits.

Τα στοιχεία εκείνα τα οποία συνθέτουν όλες τις αρχιτεκτονικές που υλοποιήθηκαν είναι το *interface*, η μονάδα διαχείρισης κλειδιών, η μονάδα διαχείρισης των *tweak values*, η μονάδα κρυπτογράφησης/αποκρυπτογράφησης, η μονάδα διαχείρισης της *i value* καθώς και η μονάδα ελέγχου του κυκλώματος. Η μονάδα κρυπτογράφησης/αποκρυπτογράφησης φαίνεται στο σχήμα 3.2.



Σχήμα 3.1 Γενική αρχιτεκτονική

Όπως είναι λογικό στη μονάδα αυτή, το μέγεθος της είναι απολύτως ανάλογο του αριθμού των πυρήνων που χρησιμοποιεί η κάθε αρχιτεκτονική. Οι πυρήνες αυτοί μπορούν να τοποθετηθούν σε σειρά δίχως να δημιουργείται κάποιο ιδιαίτερο πρόβλημα καθώς ο κάθε πυρήνας αναλαμβάνει να κρυπτογραφήσει/αποκρυπτογραφήσει τα δεδομένα που του αναλογούν. Γίνεται εύκολα κατανοητός λοιπόν ο τρόπος με τον οποίο τοποθετούνται οι πυρήνες σε όλες τις αρχιτεκτονικές.



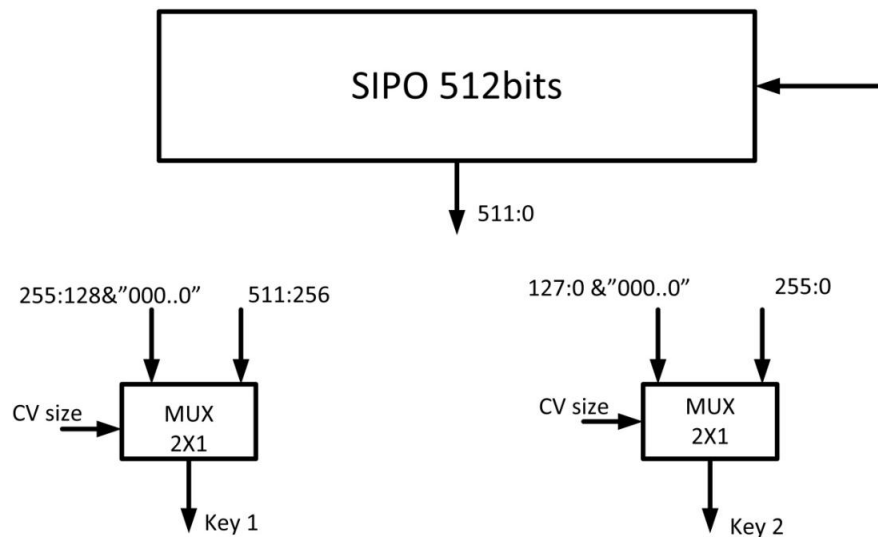
Σχήμα 3.2 Μονάδα κρυπτογράφησης/αποκρυπτογράφησης.

Η μονάδα διαχείρισης της *i value* είναι ουσιαστικά ένας *sipo* (*serial in - parallel out*) register μεγέθους 128 bits ο οποίος φορτώνει σειριακά ανά 64 bit τη πληροφορία. Ακόμη, πριν βγάλει τα δεδομένα στην έξοδο του, αναλαμβάνει τη διαμόρφωση τους κατά *little endian*. Αυτό σημαίνει ότι όταν παίρνει μια πληροφορία π.χ. *18191a1b1c1d1e1f* τότε η έξοδος θα έχει την εξής μορφή *1f1e1d1c1b1a1918*.

Η μονάδα διαχείρισης κλειδιών αναλαμβάνει τη τροφοδοσία των πυρήνων AES με τα κατάλληλα κλειδιά. Η μονάδα αυτή δίνει έμφαση σε ένα σημαντικό πρόβλημα που παρουσιάζεται. Το κύκλωμα θα πρέπει να λειτουργεί είτε επιλέγονται κλειδιά μήκους 128 bits είτε κλειδιά μήκους 256 bits. Ανάλογα με την κρυπτογράφηση/αποκρυπτογράφηση (XTS-128 ή XTS-256) λοιπόν που έχει επιλεγεί πρέπει να εξάγονται και τα κατάλληλα μήκη κλειδιών. Το πρόβλημα αυτό έχει λυθεί χρησιμοποιώντας έναν καταχωρητή σειριακής εισόδου – παράλληλης εξόδου μήκους 512 bits, όπου με τη βοήθεια δυο κατάλληλων πολυπλεκτών (δύο σε ένα, 2X1) αναλαμβάνουν να φέρουν εις πέρας την όλη διαδικασία της μονάδας διαχείρισης κλειδιών.

Στη περίπτωση του XTS-AES 256 bits, το σήμα *in size* έχει τεθεί στο υψηλό δυναμικό (“high”) και τα κλειδιά έχουν εξαχθεί κανονικά χωρίς προβλήματα. Στη περίπτωση όμως του XTS-AES 128 bits υπήρχε ένα πρόβλημα καθώς το μήκος των κλειδιών δε μπορεί να αλλάξει κατά περίπτωση όταν πρόκειται για υλοποίηση ψηφιακού κυκλώματος. Το πρόβλημα αυτό λύνεται με τη μέθοδο του γεμίσματος (*padding*) με μηδενικά ούτως ώστε το μήκος να είναι σταθερό στα 256 bits. Σε αυτή τη περίπτωση, όπως είναι προφανές, η τιμή του *in size* έχει τεθεί στο χαμηλό δυναμικό (“low”). Χρησιμοποιώντας αυτήν την απλή τεχνική λοιπόν, η μονάδα διαχείρισης κλειδιών λειτουργεί αποτελεσματικά

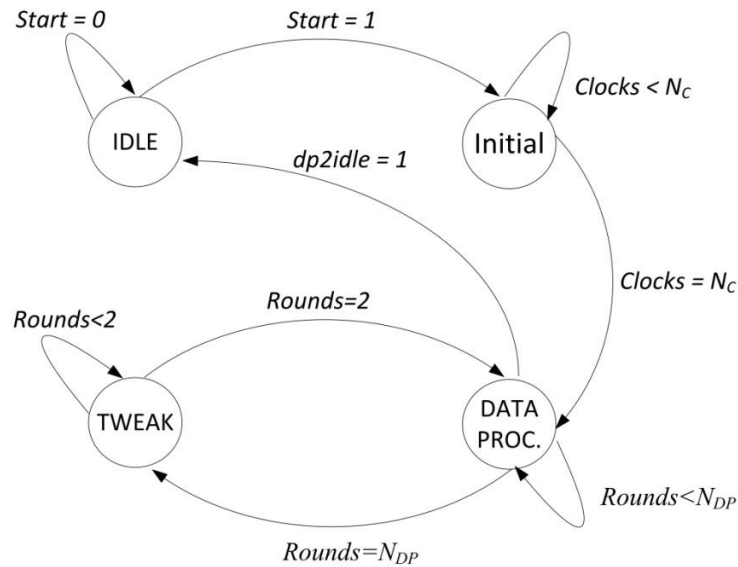
Η διαδικασία που μόλις περιγράφηκε συνοπτικά παραπάνω φαίνεται καλύτερα στο σχήμα 3.3 που ακολουθεί



Σχήμα 3.3 Μονάδα διαχείρισης κλειδιών

Το *interface* περιλαμβάνει όλες εκείνες τις επιμέρους μονάδες οι οποίες είναι απαραίτητες για τη κατάλληλη φόρτωση/εξαγωγή του *plaintext/ciphertext* (κείμενο το οποίο πρόκειται να υποβληθεί σε κρυπτογράφηση/αποκρυπτογράφηση αντίστοιχα). Οι μονάδες αυτές είναι δύο καταχωρητές. Ο πρώτος, που τοποθετείται στην είσοδο, είναι ένας καταχωρητής σειριακής εισόδου – παράλληλης εξόδου ενώ ο δεύτερος, ο οποίος τοποθετείται στην έξοδο, είναι ένας καταχωρητής παράλληλης εισόδου – σειριακής εξόδου. Η σειριακή είσοδος/έξοδος, στις αντίστοιχες περιπτώσεις, γίνεται ανά 64-bits. Το μέγεθος και των δύο καταχωρητών έχει άμεση σχέση με την κάθε επιμέρους αρχιτεκτονική και συγκεκριμένα με τον αριθμό των πυρήνων που χρησιμοποιεί αυτή. Ο γενικός τύπος που ορίζει το μέγεθος των καταχωρητών είναι $N \cdot 128bits$, όπου N ο αριθμός των πυρήνων. Άρα, η αρχιτεκτονική δύο πυρήνων που εξετάζεται σε αυτήν την υποενότητα διαθέτει καταχωρητές μήκους 256 bits. Όλα αυτά που περιγράφονται παραπάνω φαίνονται καλύτερα στα σχηματικά των επιμέρους αρχιτεκτονικών που δείχνονται στη συνέχεια. Η μονάδα ελέγχου καθώς και η μονάδα διαχείρισης των “*tweak values*” διαφέρουν αρκετά μεταξύ των επιμέρους αρχιτεκτονικών και γίνεται ειδική μνεία σε αυτές στη συνέχεια.

Η μονάδα ελέγχου αποτελεί το στοιχείο εκείνο το οποίο είναι υπεύθυνο για τη διαχείριση όλων των σημάτων του κυκλώματος που πρέπει να ενεργοποιούνται και να απενεργοποιούνται στις κατάλληλες χρονικές στιγμές. Η *control unit* λοιπόν αποτελείται από ένα *fsm* (*finite state machine*)[16] το σχέδιο λειτουργίας του οποίου φαίνεται αναλυτικά στο παρακάτω σχήμα 3.4



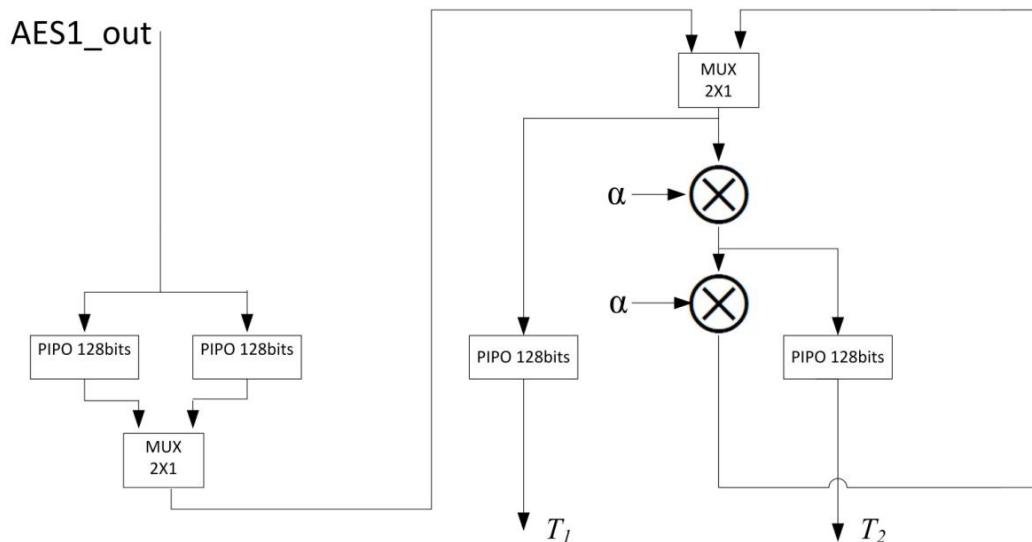
Σχήμα 3.4 Σχεδιάγραμμα της μονάδας ελέγχου για την αρχιτεκτονική δύο πυρήνων

Το *fsm* που χρησιμοποιήθηκε για την αρχιτεκτονική δύο πυρήνων αποτελείται από τέσσερις βασικές καταστάσεις. Αυτές είναι η *idle* (κατάσταση κατά την οποία το κύκλωμα βρίσκεται σε αδράνεια), *initial* (κατάσταση κατά την οποία γίνονται οι απαραίτητες προετοιμασίες για την αρχικοποίηση του κυκλώματος), *data proc.* (κατά την οποία επεξεργάζονται τα δεδομένων) και τέλος η *tweak* (κατάσταση κατά την οποία παράγονται δεδομένα και οι *tweak values* ταυτόχρονα).

Όσο το σήμα “*start*” παραμένει στο χαμηλό δυναμικό, η μονάδα ελέγχου βρίσκεται στη κατάσταση “*idle*”. Όταν έρθει μια θετική ακμή πυροδότησης στο σήμα *start* η μονάδα ελέγχου μεταβαίνει στην κατάσταση “*initial*” για την αρχικοποίηση του κυκλώματος (φόρτωση κλειδιών, φόρτωση δεδομένων κ.α.). Στη κατάσταση αυτή θα παραμείνει για 21 *clocks*, όταν χρησιμοποιείται κλειδί μήκους 128 bits, και για 25 *clocks*, όταν χρησιμοποιείται κλειδί μήκους 256 bits, τα οποία είναι απαραίτητα ούτως ώστε να γίνει η κατάλληλη προετοιμασία του κυκλώματος. Στη συνέχεια, μεταβαίνει στη κατάσταση “*data proc.*” για να επεξεργαστεί τα δεδομένα όπου θα παραμείνει για N_{DP} *rounds*. Ο αριθμός N_{DP} στα 15 *rounds* για τη πρώτη φορά που εκτελείται η κατάσταση “*data proc.*” και στη συνέχεια θα αλλάξει σε 31 *rounds*. Υπενθυμίζεται σε αυτό το σημείο ότι το κάθε *round* αποτελείται από μια σειρά από *clocks*, ο αριθμός των οποίων είναι ίσος με τον αριθμό που χρειάζονται ούτως ώστε να κρυπτογραφηθούν/αποκρυπτογραφηθούν τα δεδομένα. Μετά από αυτήν τη κατάσταση, η μονάδα ελέγχου μεταβαίνει στην “*tweak*” για τη παραγωγή *tweak values* και δεδομένων. Στη κατάσταση αυτή θα παραμείνει για δύο *rounds* καθώς, όπως αναφέρθηκε και στην υποενότητα 3.1.2, οι *tweak values*, όσον αφορά την αρχιτεκτονική δύο πυρήνων, παράγονται ανά δύο. Η κατάσταση που ακολουθεί είναι η “*data proc.*” για την επεξεργασία δεδομένων.

Γενικά, μετά από αυτό το στάδιο, οι καταστάσεις “*data proc.*” και “*tweak*” εναλλάσσονται μεταξύ τους. Το εσωτερικό σήμα *dp2idle* είναι εκείνο που τερματίζει τη λειτουργία του κυκλώματος όταν ενεργοποιηθεί. Το *dp2idle* σχετίζεται άμεσα με το σήμα *data_ld* καθώς το τελευταίο πρέπει να ενεργοποιηθεί τακτικά ώστε να εισέρχονται νέα δεδομένα στο κύκλωμα

προς επεξεργασία. Αν το σήμα *data_ld* μείνει ανενεργό για αρκετά *clocks* τότε δεν υπάρχουν νέα δεδομένα προς επεξεργασία και η μονάδα ελέγχου μεταβαίνει στη κατάσταση “idle” μέσω του σήματος *dp2idle*.

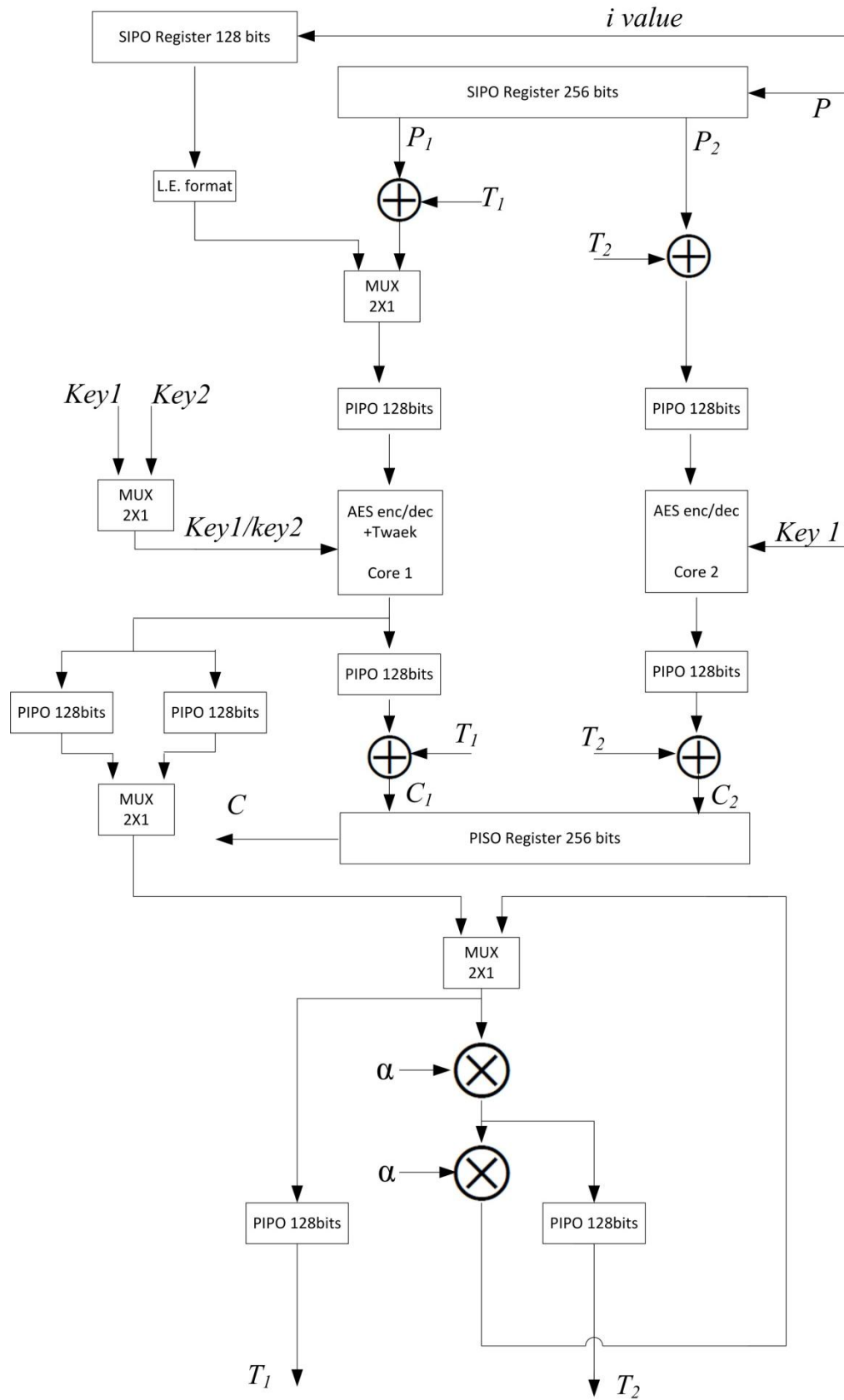


Σχήμα 3.5 Μονάδα διαχείρισης *tweak values*.

Τέλος, η μονάδα διαχείρισης των “*tweak values*” φαίνεται στο σχήμα 3.5 αναλυτικά. Τα στοιχεία εκείνα τα οποία είναι απαραίτητα για την υλοποίηση της είναι δύο πολλαπλασιαστές *GFMC* (*Galois Field Multiplication Core*) οι οποίοι αναλαμβάνουν τη παραγωγή όλων των επιμέρους “*tweak values*” που αντιστοιχούν σε κάθε ένα από τα μπλοκ του *sector*. Υπενθυμίζουμε ότι ο πολλαπλασιασμός και η πρόσθεση γίνονται στο πεδίο *Galois*, $GF(2^{128})$, οι οποίες έχουν εξηγηθεί αναλυτικά στο κεφάλαιο 2.

Ακόμη, απαραίτητοι είναι τέσσερις καταχωρητές, παράλληλης εισόδου – παράλληλης εξόδου (*parallel in – parallel out, pipo*) των 128 bits. για την προσωρινή αποθήκευση των “*tweak values*”. Οι δύο πρώτοι καταχωρητές, που τοποθετούνται στην έξοδο της πρώτης μονάδας AES, κρατάνε τις κρυπτογραφημένες *i values* και οι υπόλοιποι δύο παράγουν όλες τις υπόλοιπες μέσω αυτών. Τέλος, απαραίτητοι είναι και δύο πολυπλέκτες, δύο σε ένα (2X1), ώστε να χειρίζονται τη προτεραιότητα των “*tweak values*” που πρέπει να ακολουθείται.

Τέλος, στο σχήμα 3.6 παρουσιάζεται αναλυτικά σε επίπεδο RTL η αρχιτεκτονική δύο πυρήνων. Στο κάτω μέρος του σχήματος φαίνεται η μονάδα διαχείρισης των “*tweak values*” που μόλις αναλύθηκε.



Σχήμα 3.6 Αρχιτεκτονική δύο πυρήνων

Οι καταχωρητές *SIPO Register 256 bits* και *PISO Register 256 bits* αποτελούν τη διεπαφή (*interface*) του κυκλώματος ενώ ο καταχωρητής *SIPO Register 128 bits* μαζί με την μονάδα *L.E. format* αποτελούν την “*i value unit*”. Επίσης φαίνονται οι *xor-units* (μονάδες που αναλαμβάνουν τη πρόσθεση). Ακόμη, απαραίτητοι ήταν κάποιοι επιπλέον καταχωρητές παράλληλης εισόδου – παράλληλης εξόδου (*parallel in – parallel out, rifo*) των 128 bits οι οποίοι διασφαλίζουν την ομαλή λειτουργία του κυκλώματος. Κάποιοι πολυπλέκτες ήταν αναγκαίοι ούτως ώστε να δημιουργούνται οι σωστές ροές δεδομένων αναλόγως της χρονικής κατάστασης που βρίσκεται το κύκλωμα. Τέλος, παρατηρούνται και οι δύο πυρήνες AES που ενσωματώνει αυτή η αρχιτεκτονική.

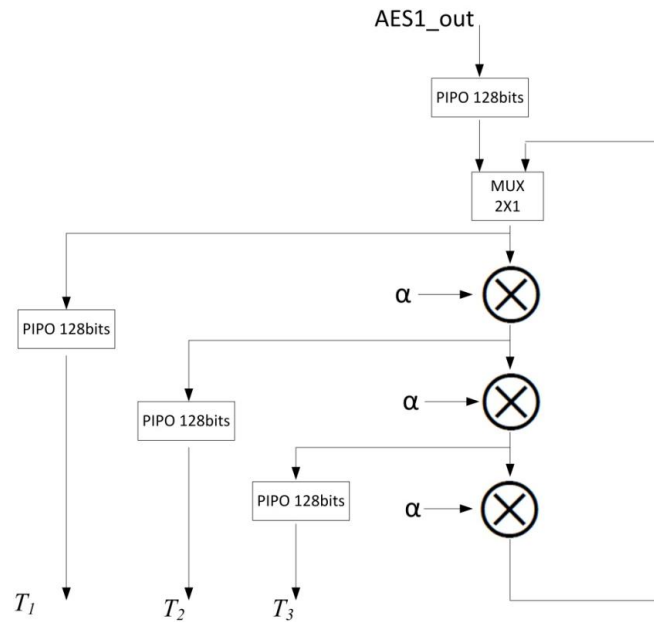
3.2.2 Αρχιτεκτονική τριών πυρήνων

Η αρχιτεκτονική τριών πυρήνων που παρουσιάζεται στη παρούσα υποενότητα δεν έχει πολλές διαφορές σε σύγκριση με εκείνη των δύο πυρήνων. Άρα, δεν υπάρχει σοβαρός λόγος να γίνει αναλυτική περιγραφή όλων των στοιχείων που περιλαμβάνει. Αυτό που πρέπει να τονιστεί όμως είναι τα σημεία που έχουν διαφοροποιηθεί και με ποιον τρόπο. Στο σχήμα 3.9 παρουσιάζεται αναλυτικά η αρχιτεκτονική τριών πυρήνων.

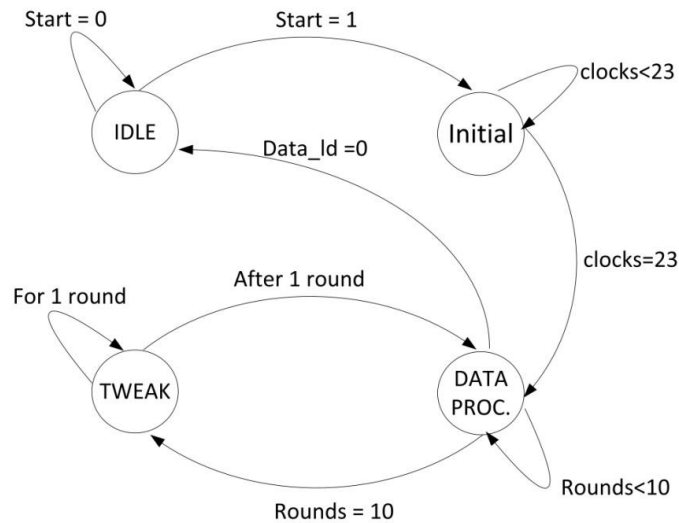
Μια πρώτη αλλαγή, που ήταν απαραίτητη να γίνει, ήταν στο μέγεθος των καταχωρητών (*SIPO 384-bits, PISO 384-bits*) το οποίο μεταβλήθηκε από 256 σε 384 bits καθώς τώρα υπάρχουν τρεις πυρήνες AES που πρέπει να τροφοδοτούνται σε κάθε *round*. Επίσης μια σημαντική αλλαγή φαίνεται και στη “*tweak values unit*” όπως φαίνεται και στο σχήμα 3.7.

Στην αρχιτεκτονική τριών πυρήνων, παράγεται μια “*tweak value*” η οποία χρησιμοποιείται για ένα ολόκληρο *sector* και στη συνέχεια αντικαθίσταται από την επόμενη που αντιστοιχεί στο επόμενο *sector*. Σημειώνεται ότι για κάθε μπλοκ εντός του *sector*, η “*tweak value*” που του ανατίθεται είναι διαφορετική λόγω των GFMC που έχουν αυξηθεί σε τρεις. Όπως είναι προφανές ο δεύτερος καταχωρητής, στην έξοδο της πρώτης μονάδας AES είναι περιττός και μπορεί να παραλειφθεί από την αρχιτεκτονική. Κατά συνέπεια, ο πολυπλέκτης, που είχε τοποθετηθεί στην αρχιτεκτονική δύο πυρήνων ούτως ώστε να διαχειρίζεται τους δύο καταχωρητές, μπορεί και αυτός να παραλειφθεί με τη σειρά του. Θυμίζουμε ότι έξοδοι, T_1 , T_2 και T_3 , της “*tweak values unit*”, αποτελούν τις εισόδους των XOR-units που υπάρχουν πριν και μετά τους AES.

Καθώς έχει αλλάξει ο τρόπος και η σειρά παραγωγής των “*tweak values*”, η μονάδα ελέγχου έχει και αυτή με τη σειρά της υποστεί κάποιες αλλαγές ούτως ώστε να μπορεί να υποστηρίξει τη παρούσα αρχιτεκτονική. Οι αλλαγές που έχουν γίνει στη λειτουργία της μονάδας ελέγχου φαίνονται στο σχήμα 3.8 αναλυτικά

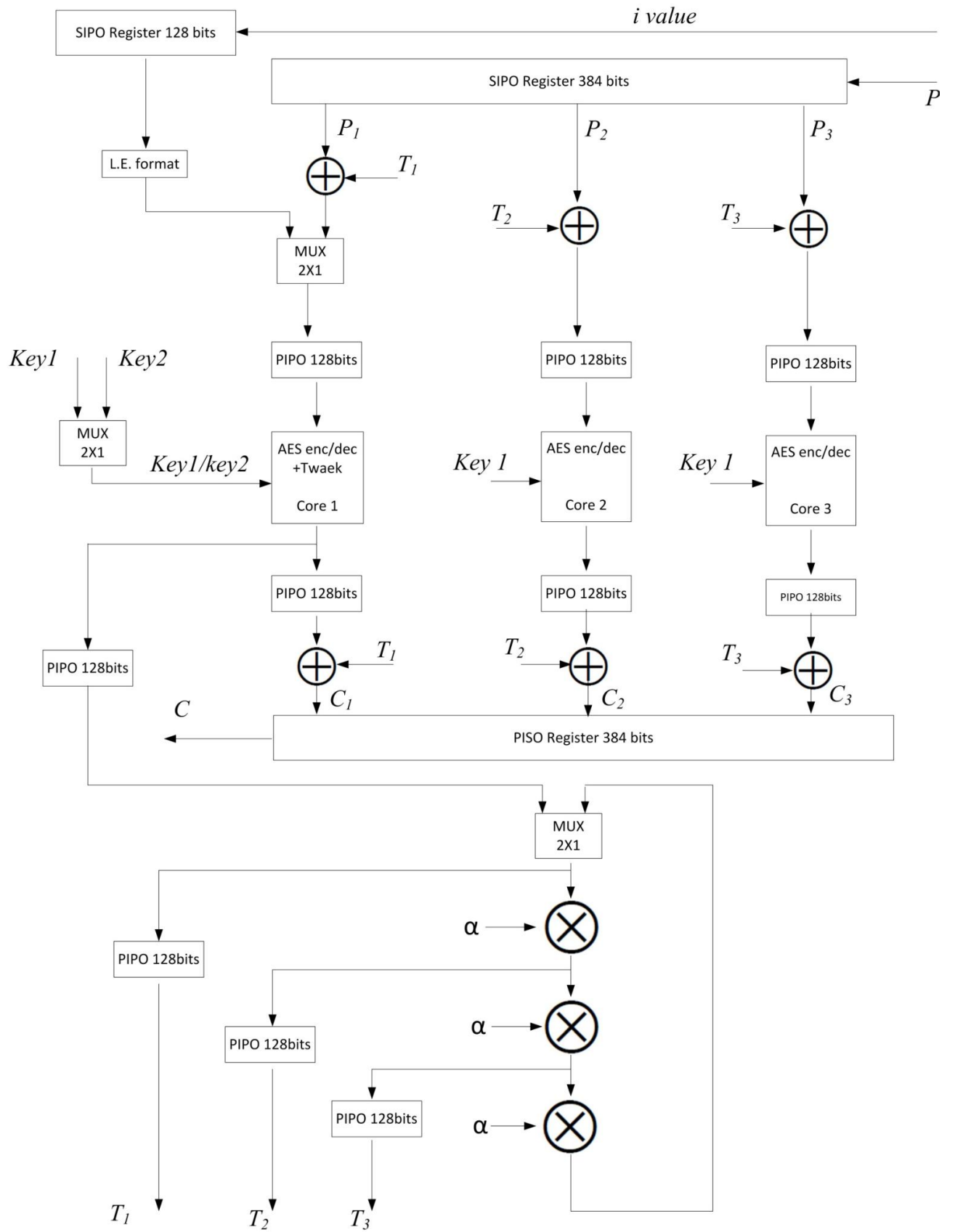


Σχήμα 3.7 Μονάδα διαχείρισης *tweak values*.



Σχήμα 3.8 Σχεδιάγραμμα της μονάδας ελέγχου για την αρχιτεκτονική τριών πυρήνων

Οι δυνατές καταστάσεις είναι παρόμοιες (“idle”, “initial”, “data proc.”, “tweak”) με τη μονάδα ελέγχου της αρχιτεκτονικής δύο πυρήνων. Το βασικό στοιχείο το οποίο έχει αλλάξει είναι οι χρονικές στιγμές μετάβασης μεταξύ των καταστάσεων “data proc.” και “tweak”. Όταν γίνει η μετάβαση από την κατάσταση “initial” στη κατάσταση “data proc.”, η μονάδα ελέγχου θα παραμείνει σε αυτήν για 11 *rounds*. Στη συνέχεια, θα μεταβεί στη κατάσταση “tweak” όπου θα παραμείνει για ένα *round* και θα ξανά επιστρέψει στην “data proc.”. Αυτή η διαδικασία (μεταξύ “data proc.” και “tweak”) θα επαναλαμβάνεται μέχρις ότου το κύκλωμα τερματίσει τη λειτουργία του

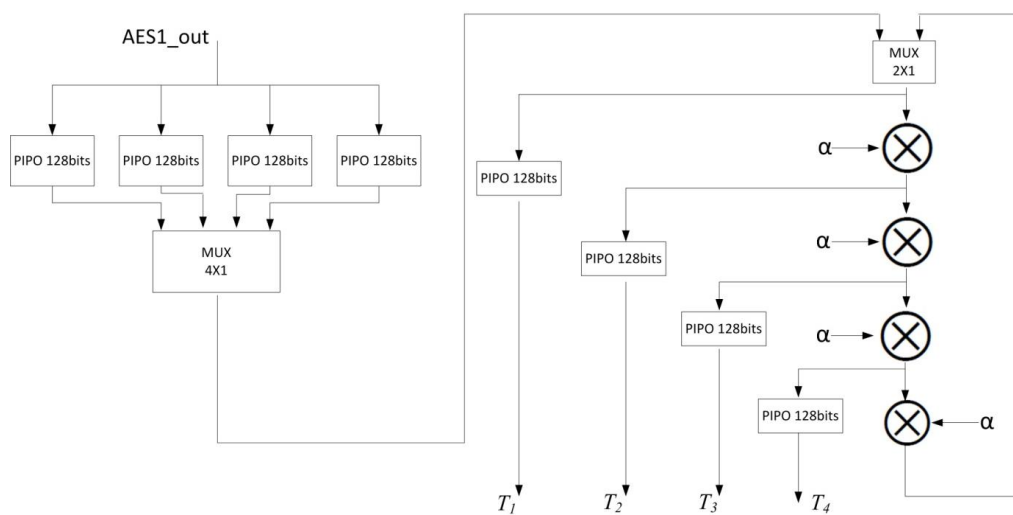


Σχήμα 3.9 Αρχιτεκτονική τριών πυρήνων

3.2.3 Αρχιτεκτονική τεσσάρων πυρήνων

Σε αυτήν την υποενότητα παρουσιάζεται η αρχιτεκτονική τεσσάρων πυρήνων η οποία και αυτή με τη σειρά της, όπως και η αρχιτεκτονική τριών πυρήνων, δε παρουσιάζει μεγάλες διαφορές σε σύγκριση με εκείνη των δύο πυρήνων.

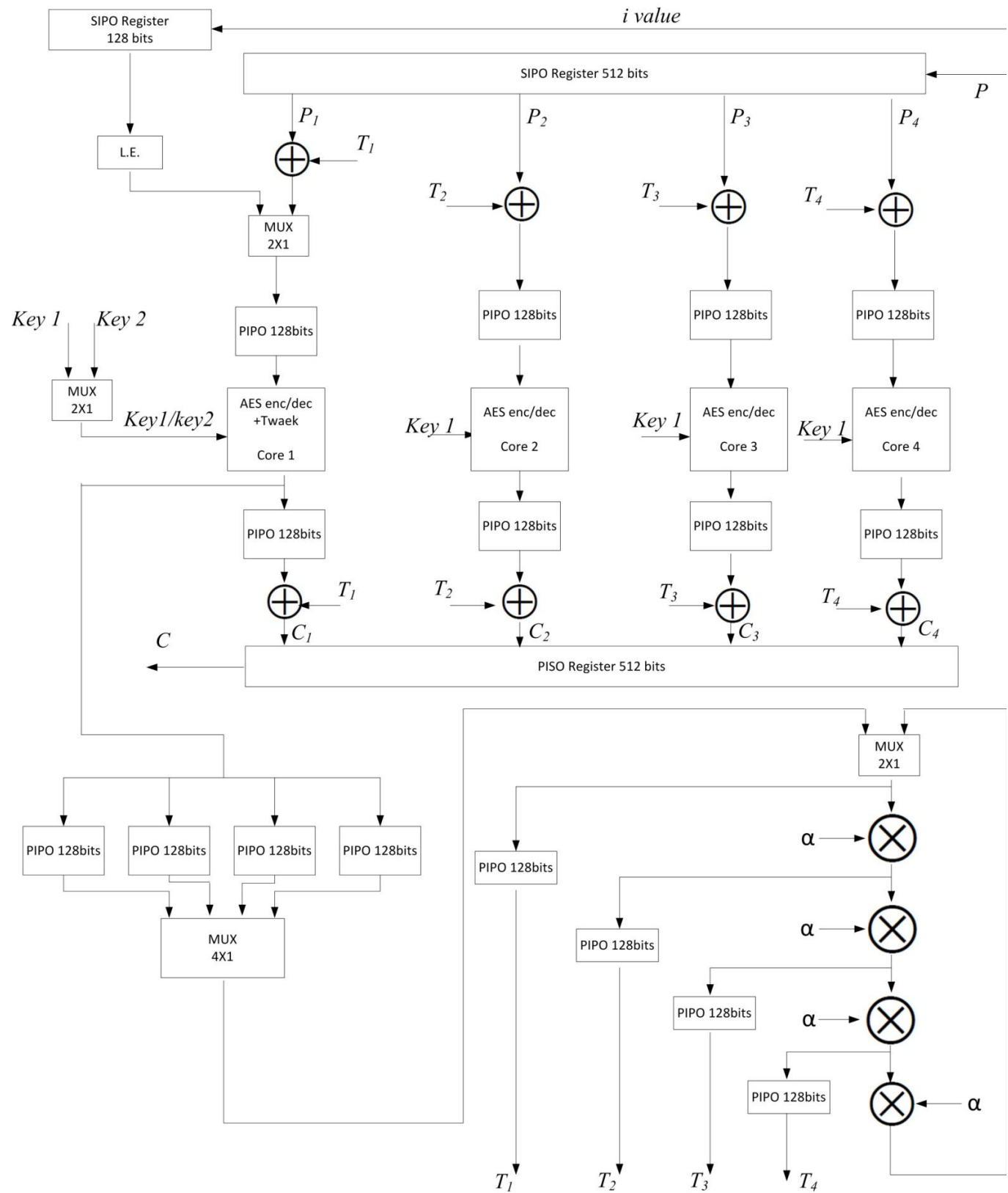
Μια σημαντική διαφορά παρατηρείται στο τρόπο και τη λογική παραγωγής των *tweak values*. Η μονάδα διαχείρισης των *tweak values* αναλαμβάνει να υπολογίσει τέσσερις κατά σειρά *tweak values* οι οποίες αντιστοιχούν στα τέσσερα επόμενα *sectors* στα οποία πρόκειται να αποθηκευτούν δεδομένα. Στο σχήμα 3.10 φαίνεται αναλυτικά η δομή, σε επίπεδο RTL, της μονάδας διαχείρισης των “*tweak values*”.



Σχήμα 3.10 Μονάδα διαχείρισης *tweak values*

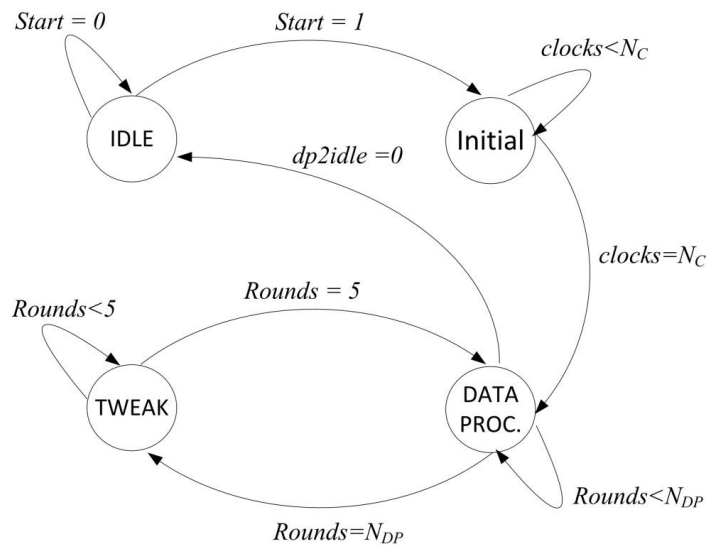
Όπως είναι λογικό, εφόσον οι *tweak values* υπολογίζονται ανά τέσσερις, οι καταχωρητές έχουν γίνει και αυτοί με τη σειρά τους τέσσερις. Κατά συνέπεια, ο πολυπλέκτης, που χειρίζεται ποια από αυτές θα χρησιμοποιείται κάθε φορά, έχει αλλάξει και αυτός με τη σειρά του σε έναν πολυπλέκτη τέσσερα σε ένα (4X1). Ακόμη, ένας πολλαπλασιαστής GFMC έχει προστεθεί ούτως ώστε οι επί μέρους *tweak values* που αντιστοιχούν σε κάθε τέσσερα μπλοκ μέσα στο *sector* να είναι διαθέσιμοι ταυτόχρονα.

Μια ακόμη σημαντική διαφορά υπάρχει και στο μέγεθος των καταχωρητών εισόδου και εξόδου (*SIPO 512-bits*, *PISO 512-bits*). Το μέγεθος τους έχει αυξηθεί στα 512 bits καθώς η αρχιτεκτονική τεσσάρων πυρήνων πρέπει τροφοδοτεί με δεδομένα τέσσερις πυρήνες AES. Μπορούμε να δούμε καλύτερα όσα περιγράφηκαν παραπάνω στο αμέσως επόμενο σχήμα το οποίο δείχνει την αρχιτεκτονική τεσσάρων πυρήνων σε επίπεδο RTL. Υπενθυμίζουμε σε αυτό το σημείο ότι τα κλειδιά (*key1* και *key2*) προέρχονται από το κύκλωμα εκείνο που απεικονίζεται στο σχήμα 3.3.



Σχήμα 3.11 Αρχιτεκτονική τεσσάρων πυρήνων

Τέλος, θα μιλήσουμε για τις διαφορές που υπάρχουν στη λογική λειτουργίας της μονάδας ελέγχου σε σύγκριση με εκείνη της αρχιτεκτονικής δύο πυρήνων. Η λογική λειτουργίας φαίνεται στο σχήμα 3.12 αναλυτικά.



Σχήμα 3.12 Σχεδιάγραμμα της μονάδας ελέγχου για την αρχιτεκτονική τεσσάρων πυρήνων

Οι διαφορές που έχει υποστεί η μονάδα ελέγχου ούτως ώστε να ανταποκρίνεται στη παρούσα αρχιτεκτονική αφορούν κυρίως αλλαγές που έχουν γίνει στο πεδίο των *rounds* καθώς έχουν αυξηθεί οι πυρήνες και έχει αλλάξει η διαδικασία παραγωγής των *tweak values*. Οι δυνατές καταστάσεις παραμένουν οι ίδιες (“idle”, “initial”, “data proc.”, “tweak”). Τη πρώτη φορά που η *control unit* θα μεταβεί στη κατάσταση “data proc.” θα παραμείνει μόνο για 4 *rounds*, $N_{DP} = 5$, και στη συνέχεια θα μεταβεί στη κατάσταση “tweak” όπου παραμένει πάντα για 4 *rounds*. Στη συνέχεια, η μονάδα ελέγχου επιστρέφει στη κατάσταση “data proc.” όπου θα παραμείνει για 36 *rounds*, $N_{DP} = 37$. Η διαδικασία αυτή (εναλλαγή μεταξύ “tweak” για 4 *rounds* και “data proc.” για 36 *rounds*) θα συνεχίζεται μέχρι τη στιγμή που το κύκλωμα θα τερματίσει τη λειτουργία του. Υπενθυμίζεται εδώ ότι κάθε *round* αντιστοιχεί στο χρόνο που είναι απαραίτητος ούτως ώστε να κρυπτογραφηθούν/αποκρυπτογραφηθούν τα δεδομένα.

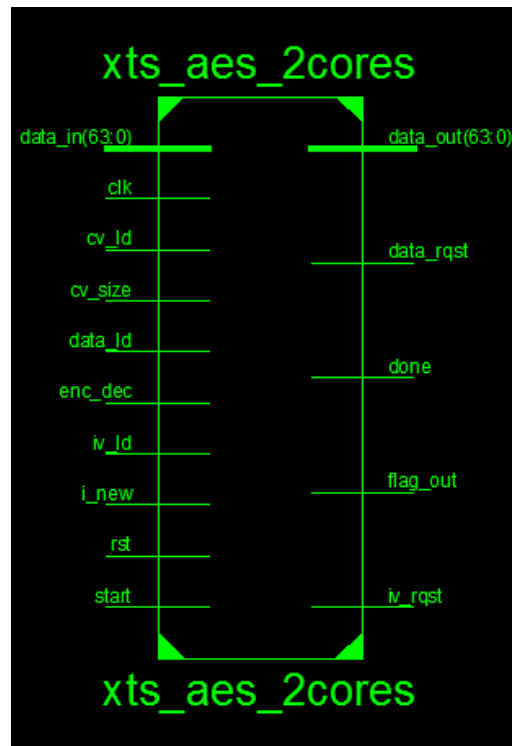
Κεφάλαιο 4

Προσομοιώσεις

Σε αυτό το κεφάλαιο θα παρουσιάζονται οι υλοποιήσεις που έγιναν και κυρίως οι προσομοιώσεις. Οι υλοποιήσεις πραγματοποιήθηκαν σε γλώσσα VHDL[17] με χρήση του λογισμικού Xilinx ISE 14.2[18] και οι προσομοιώσεις έγιναν με τη βοήθεια του λογισμικού modelsim 10.4a[19]. Οι υλοποιήσεις αφορούν τη παρουσίαση όλων εκείνων των εισόδων και εξόδων οι οποίες είναι απαραίτητες για τη σωστή και ομαλή λειτουργία των κυκλωμάτων (XTS-AES-2CORES, XTS-AES-3CORES και XTS-AES-4CORES). Οι προσομοιώσεις από την άλλη μεριά αποτελούν την απόδειξη ότι τα κυκλώματα υλοποιήθηκαν και λειτουργούν σύμφωνα με τα *test vectors* που προτείνει το πρότυπο. Γίνεται αναλυτική παρουσίαση τόσο της κρυπτογράφησης όσο και της αποκρυπτογράφησης όλων των αρχιτεκτονικών αλλά και όλων των τρόπων λειτουργίας (XTS-AES-128, XTS-AES-256).

4.1 Προσομοίωση αρχιτεκτονικής δύο πυρήνων

Στην ενότητα αυτή παρουσιάζονται η *top-level* αρχιτεκτονική δύο πυρήνων καθώς και οι προσομοιώσεις της για όλες τις περιπτώσεις (enc/dec XTS-AES-128, enc/dec XTS-AES-256). Στο σχήμα 4.1 μπορούμε να δούμε τις εισόδους και τις εξόδους του κυκλώματος XTS-AES-2CORES και στο πίνακα 4.1 περιγράφεται η λειτουργία τους



Σχήμα 4.1 Είσοδοι/Εξοδοι XTS-AES-2CORES.

Σε αυτό το σημείο σημαντικό είναι να γίνει μια περαιτέρω επεξήγηση κάποιων σημάτων. Η είσοδος των δεδομένων (*ciphertext/plaintext*), των κλειδιών (*keys*) και της *i value* γίνεται αποκλειστικά από το *bus* εισόδου δεδομένων και η έξοδος (*ciphertext/plaintext*) γίνεται στο *bus* εξόδου δεδομένων. Τα αιτήματα *iv_rqst* και *data_rqst* προστέθηκαν για την ομαλή φόρτωση των δεδομένων. Επίσης, ένα σήμα *start* ήταν απαραίτητο για την εκκίνηση του κυκλώματος. Τα σήματα *cv_ld*, *data_ld* και *iv_ld* ήταν αναγκαία για την σωστή φόρτωση των κλειδιών, των δεδομένων και της *i value* αντίστοιχα. Ακόμη, τα σήματα *enc/dec* και *cv_size* παίζουν καθοριστικό ρόλο στη λειτουργία του κυκλώματος καθώς ανάλογα με τη τιμή τους επιλέγεται ο αντίστοιχος τρόπος λειτουργίας. Τέλος, για τα σήματα *clk* και *rst* δε χρειάζονται ιδιαίτερη αναφορά καθώς είναι απαραίτητα για τη λειτουργία ενός ψηφιακού κυκλώματος. Σε αυτό το σημείο να σημειωθεί ότι τόσο η αρχιτεκτονική τριών πυρήνων όσο και η αρχιτεκτονική τεσσάρων πυρήνων δεν απαιτούν επιπλέον σήματα σε σύγκριση με την αρχιτεκτονική δύο πυρήνων. Συνεπώς, ο πίνακας 4.1 αποτελεί το γενικό πίνακα εισόδων εξόδων για όλες τις αρχιτεκτονικές που υλοποιήθηκαν.

Όνομα σήματος	Μέγεθος	I/O	Περιγραφή
Clk	1	I	Αποτελεί το ρολόι που συγχρονίζει το κύκλωμα
Rst	1	I	Αρχικοποιεί το κύκλωμα
Enc/dec	1	I	Κρυπτογράφηση('1') ή αποκρυπτογράφηση('0')
Data_in	64	I	Bus δεδομένων εισόδου
Cv_size	1	I	XTS-AES128('0'), XTS-AES256('1'),
Cv_ld	1	I	Σήμα φόρτωσης του καταχωρητή των κλειδιών
Data_ld	1	I	Σήμα φόρτωσης του καταχωρητή δεδομένων
Iv_ld	1	I	Σήμα φόρτωσης του καταχωρητή της i value
I_new	1	I	Δηλώνει ότι υπάρχει διαθέσιμη i value
Start	1	I	Έναρξη λειτουργίας του κυκλώματος
Data_out	64	O	Bus δεδομένων εξόδου.
Data_rqst	1	O	Αίτημα για νέα δεδομένα Plaintext/ciphertext
Iv_rqst	1	O	Αίτημα για νέα i value
Flag_out	1	O	Στο bus εξόδου υπάρχουν έγκυρα δεδομένα
Done	1	O	Η διαδικασία ολοκληρώθηκε

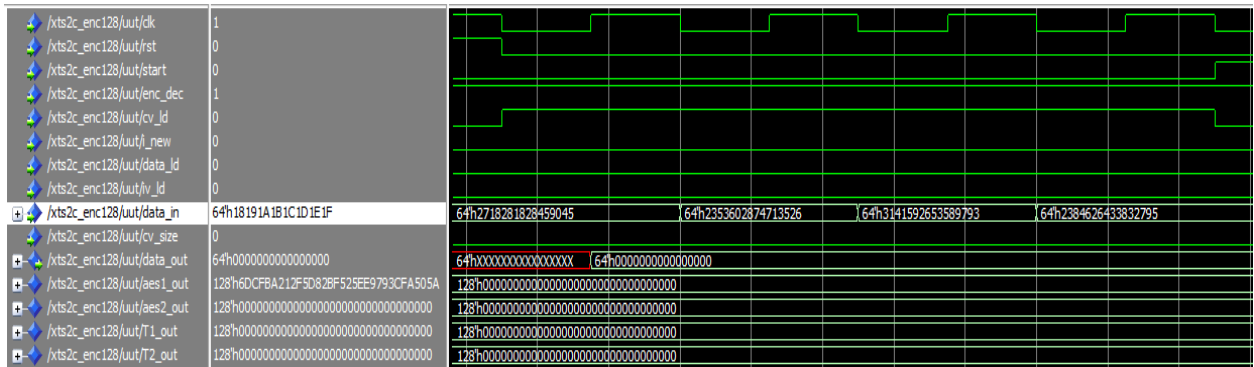
Πίνακας 4.1 Περιγραφή εισόδων/εξόδων

4.1.1 Προσομοίωση κρυπτογράφησης XTS-AES-128

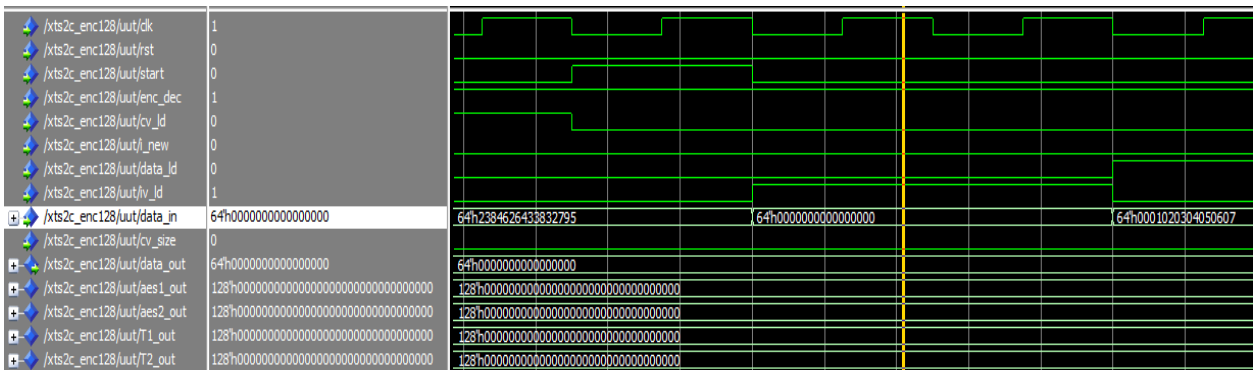
Στην υποενότητα 4.1.1 παρουσιάζονται οι προσομοιώσεις τις αρχιτεκτονικής δύο πυρήνων για την κρυπτογράφηση με μήκος κλειδιού 128-bits. Στο πίνακα 4.2 φαίνονται καθαρά το κείμενο που πρόκειται να κρυπτογραφηθεί (*plaintext*) μήκους 256-bits, η τιμή της λογικής θέσης του *sector* (*i value*) μήκους 128-bits και το κλειδί (*key*) μήκους 256-bits. Ακόμη, αναγράφεται η τιμή του κρυπτογραφημένου κειμένου(*ciphertext*) που λαμβάνουμε. Τέλος, στα σχήματα φαίνονται και οι *tweak values* που αντιστοιχούν σε κάθε block με χαρακτηριστικό δείκτη (T1_out για το 1^ο block, T2_out για το 2^ο block κ.ο.κ). Στη συνέχεια, ακολουθούν οι προσομοιώσεις

Κρυπτογράφηση XTS-AES-128	
Data_in	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f
i value	00000000000000000000000000000000 00000000000000000000000000000000
Key	27182818284590452353602874713526 31415926535897932384626433832795
Data_out	27a7479befa1d476489f308cd4cfa6e2 a96e4bbe3208ff25287dd3819616e89c

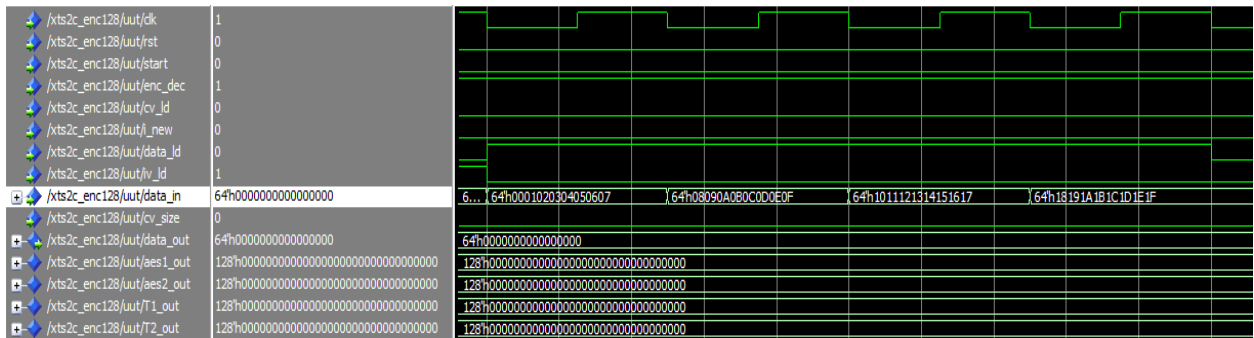
Πίνακας 4.2 Δεδομένα Κρυπτογράφησης XTS-AES-128



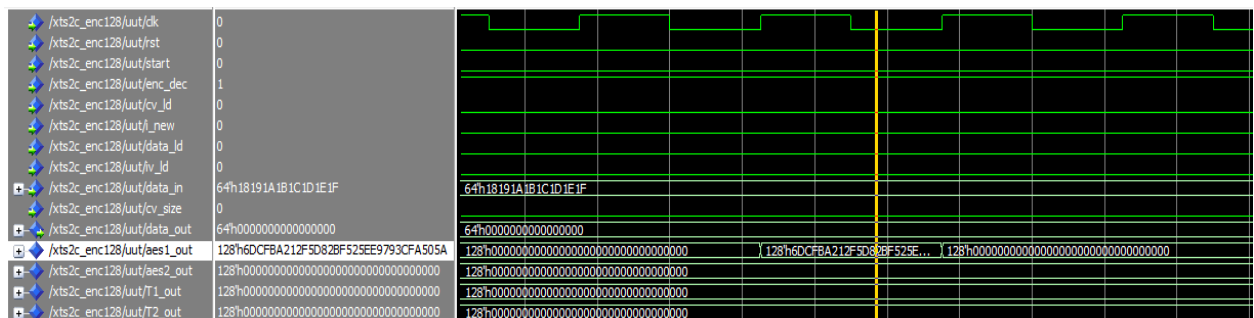
Σχήμα 4.2 Προσομοίωση κρυπτογράφησης XTS-AES-128 α' μέρος.



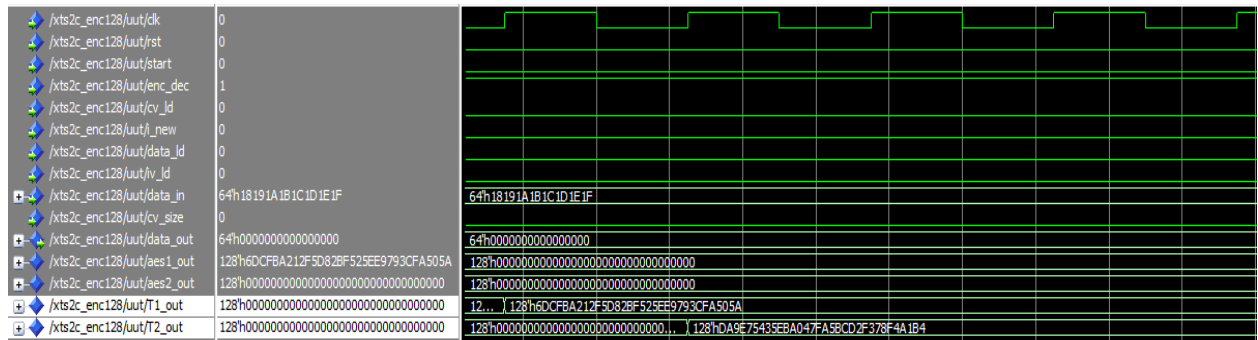
Σχήμα 4.3 Προσομοίωση κρυπτογράφησης XTS-AES-128 β' μέρος.



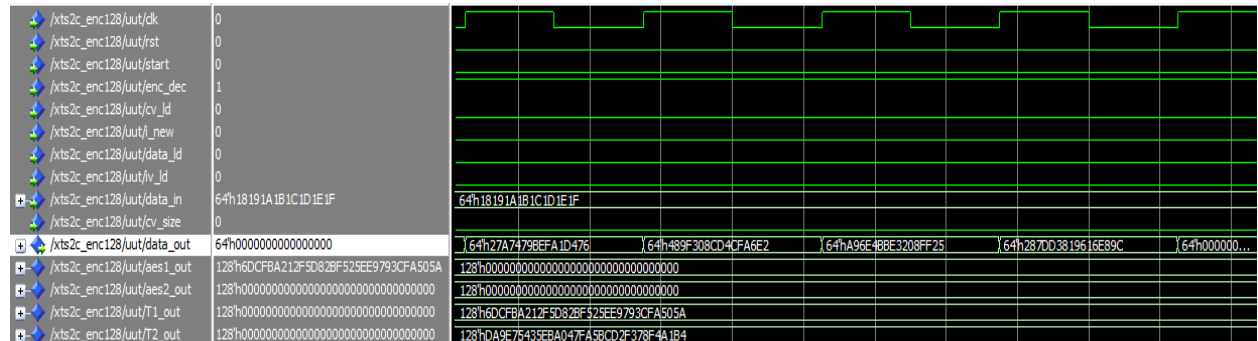
Σχήμα 4.4 Προσομοίωση κρυπτογράφησης XTS-AES-128 γ' μέρος.



Σχήμα 4.5 Προσομοίωση κρυπτογράφησης XTS-AES-128 δ' μέρος.



Σχήμα 4.6 Προσομοίωση κρυπτογράφησης XTS-AES-128 ε' μέρος.



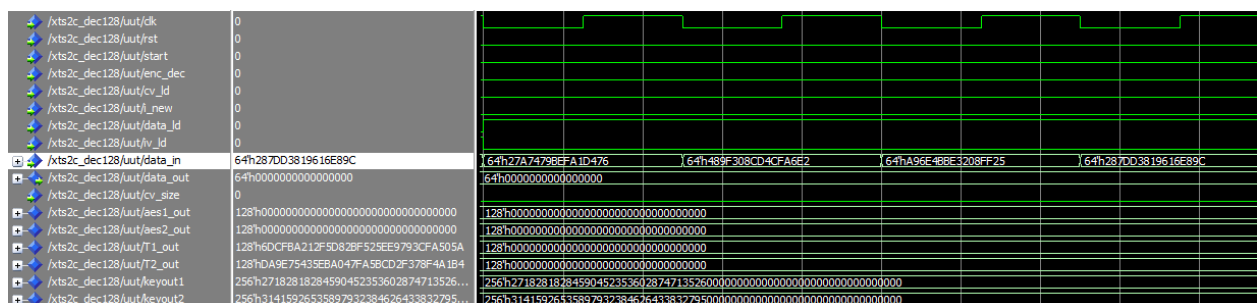
Σχήμα 4.7 Προσομοίωση κρυπτογράφησης XTS-AES-128 στ' μέρος.

4.1.2 Προσομοίωσης αποκρυπτογράφησης XTS-AES-128

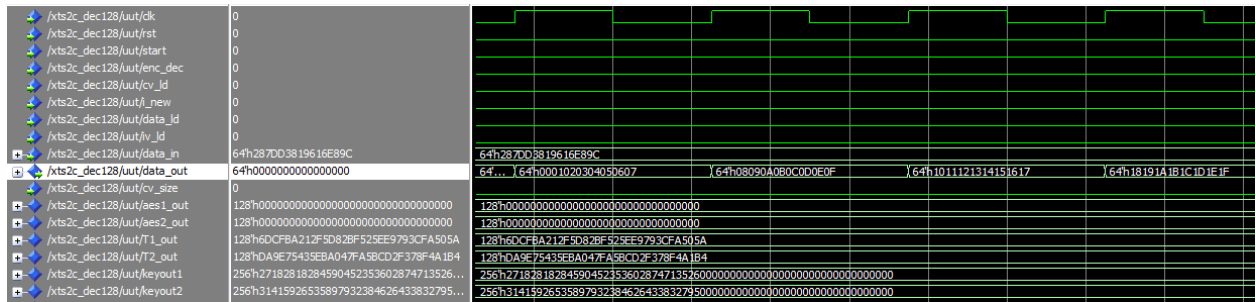
Στην υποενότητα αυτή παρουσιάζεται η XTS-AES-128 αποκρυπτογράφηση της αρχιτεκτονικής δύο πυρήνων. Στην ουσία πρόκειται για αντίστροφη διαδικασία από αυτήν που μόλις παρουσιάστηκε

Αποκρυπτογράφηση XTS-AES-128	
Data_in	27a7479bfa1d476489f308cd4cfa6e2 a96e4bbe3208ff25287dd3819616e89c
i value	00000000000000000000000000000000 00000000000000000000000000000000
Key	27182818284590452353602874713526 31415926535897932384626433832795
Data_out	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f

Πίνακας 4.3 Δεδομένα αποκρυπτογράφησης XTS-AES-128



Σχήμα 4.8 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 α' μέρος.



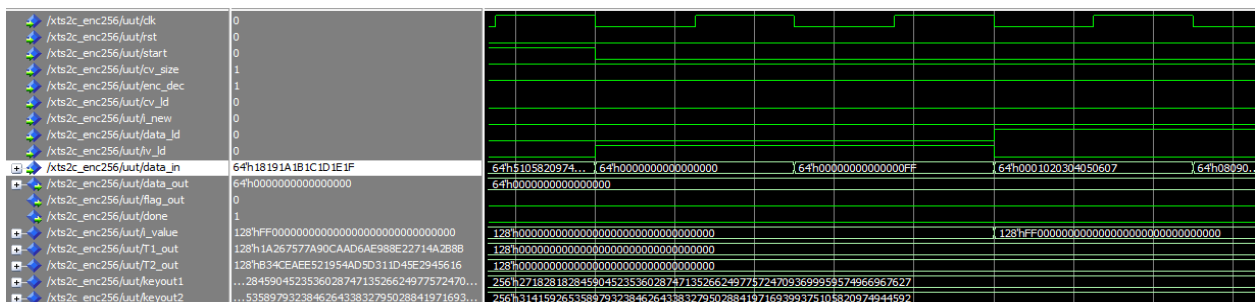
Σχήμα 4.9 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 β' μέρος.

4.1.3 Προσομοίωση κρυπτογράφησης XTS-AES-256

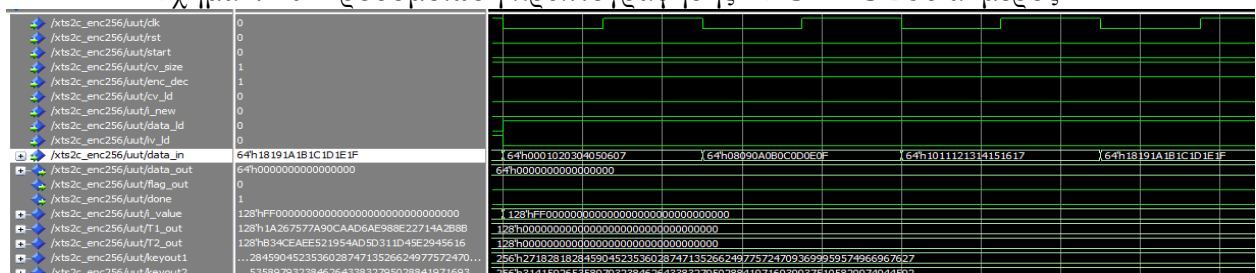
Στη παρούσα υποενότητα παρουσιάζεται η προσομοίωση XTS-AES-256 κρυπτογράφηση με μήκος κλειδιού 256-bits. Να σημειώσουμε εδώ ότι τα $T1_out$ και $T2_out$ περιέχουν τις κατάλληλες *tweak values* για τα αντίστοιχα block.

Κρυπτογράφηση XTS-AES-256	
Data_in	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f
<i>i value</i>	00000000000000000000000000000000 000000000000000000000000000000ff
Key	27182818284590452353602874713526 62497757247093699959574966967627 31415926535897932384626433832795 02884197169399375105820974944592
Data_out	1c3b3a102f770386e4836c99e370cf9b ea00803f5e482357a4ae12d414a3e63b

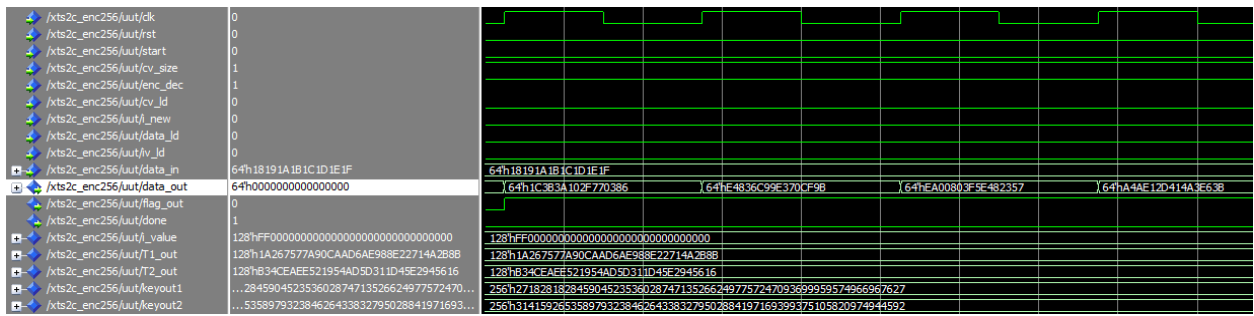
Πίνακας 4.4 Δεδομένα κρυπτογράφησης XTS-AES-256



Σχήμα 4.10 Προσομοίωση κρυπτογράφησης XTS-AES-256 α' μέρος



Σχήμα 4.11 Προσομοίωση κρυπτογράφησης XTS-AES-256 β' μέρος



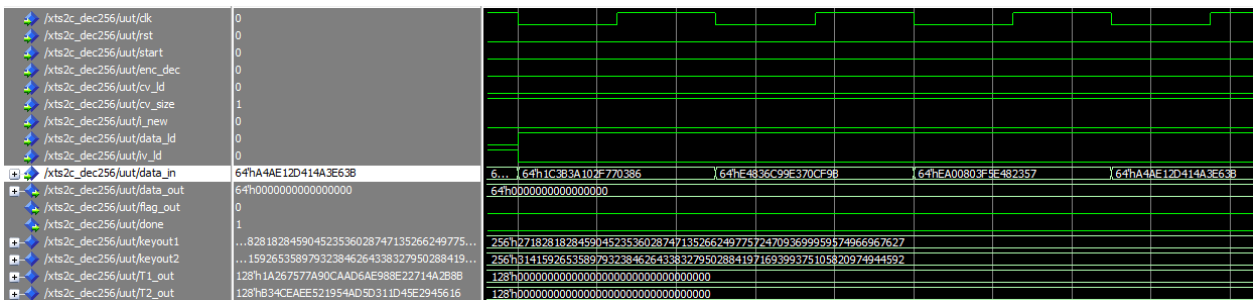
Σχήμα 4.12 Προσομοίωση κρυπτογράφησης XTS-AES-256 γ' μέρος

4.1.4 Προσομοίωση αποκρυπτογράφησης XTS-AES-256

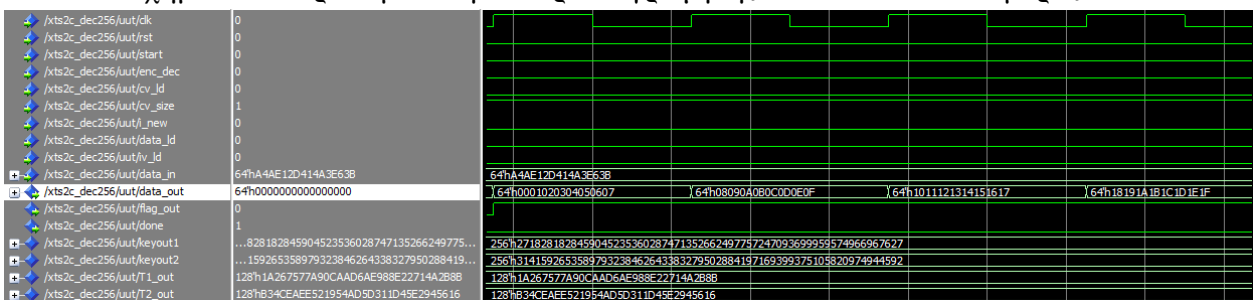
Στην υποενότητα αυτή παρουσιάζεται η προσομοίωση XTS-AES-256 αποκρυπτογράφηση με μήκος κλειδιού 256-bits.

Αποκρυπτογράφηση XTS-AES-256	
Data_in	1c3b3a102f770386e4836c99e370cf9b ea00803f5e482357a4ae12d414a3e63b
i value	00000000000000000000000000000000 000000000000000000000000000000ff
Key	27182818284590452353602874713526 62497757247093699959574966967627 31415926535897932384626433832795 02884197169399375105820974944592
Data_out	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f

Πίνακας 4.5 Δεδομένα αποκρυπτογράφησης XTS-AES-256



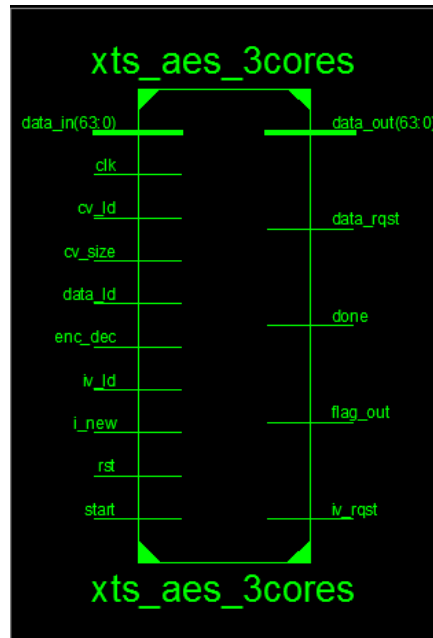
Σχήμα 4.13 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 α' μέρος



Σχήμα 4.14 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 β' μέρος

4.2 Προσομοίωση αρχιτεκτονικής τριών πυρήνων

Στην ενότητα αυτή παρουσιάζονται η top-level αρχιτεκτονική τριών πυρήνων καθώς και οι προσομοιώσεις της για όλες τις περιπτώσεις (enc/dec XTS-AES-128, enc/dec XTS-AES-256). Στο σχήμα 4.15 φαίνονται οι είσοδοι και τις έξοδοι του κυκλώματος. Η λειτουργία τους περιγράφεται στο πίνακα 4.1



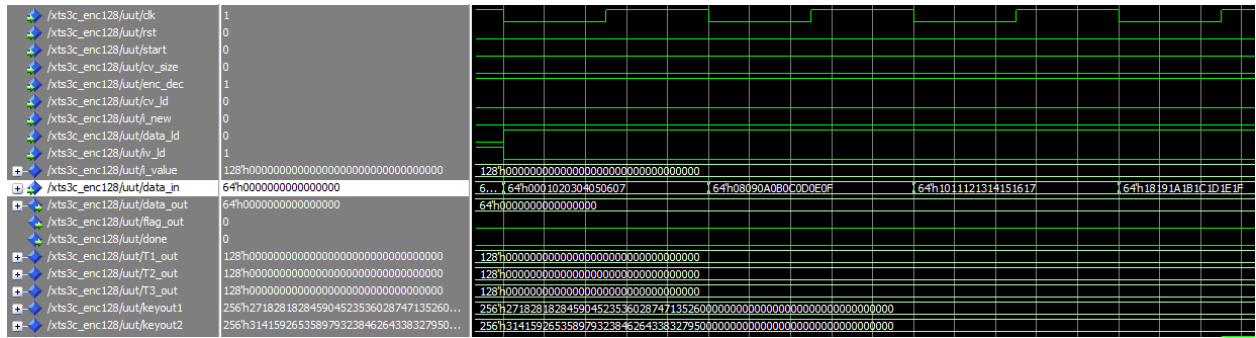
Σχήμα 4.15 Είσοδοι/έξοδοι του XTS-AES-3CORES

4.2.1 Προσομοίωση κρυπτογράφησης XTS-AES-128

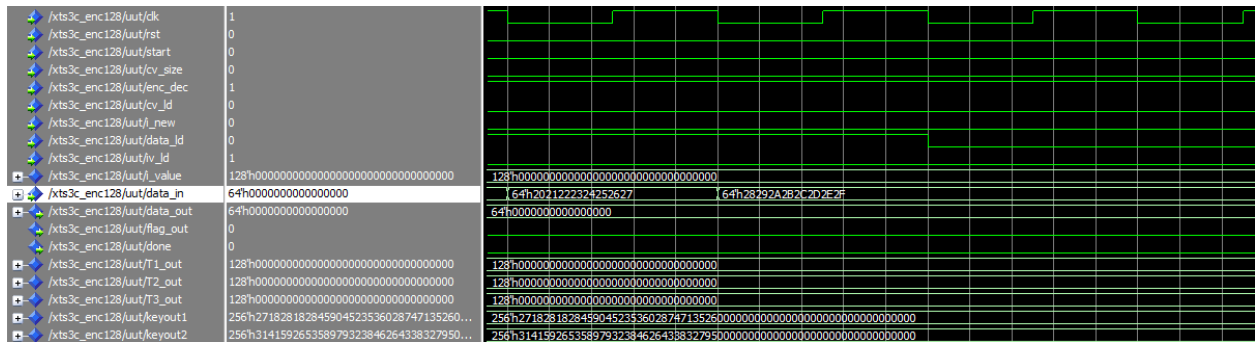
Στην ενότητα αυτή παρουσιάζεται η προσομοίωση της αρχιτεκτονικής τριών πυρήνων για τη κρυπτογράφηση με μήκος κλειδιού 128-bits.

Κρυπτογράφηση XTS-AES-128	
Data_in	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f 202122232425262728292a2b2c2d2e2f
<i>i value</i>	00000000000000000000000000000000 00000000000000000000000000000000
Key	27182818284590452353602874713526 31415926535897932384626433832795
Data_out	27a7479befa1d476489f308cd4cfa6e2 a96e4bbe3208ff25287dd3819616e89c c78cf7f5e543445f8333d8fa7f560000

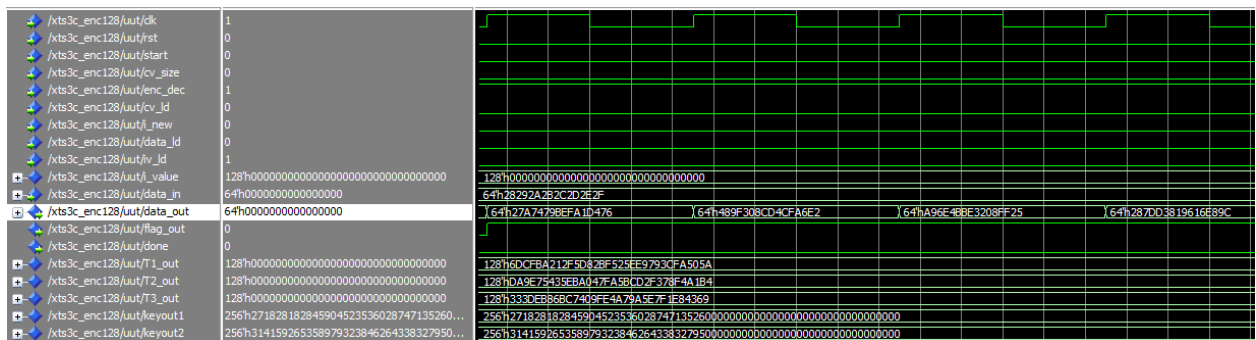
Πίνακας 4.6 Δεδομένα κρυπτογράφησης XTS-AES-128



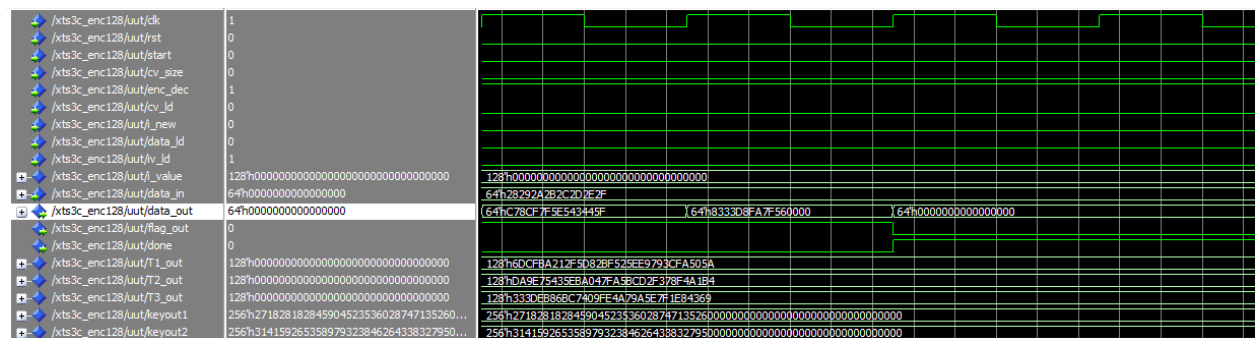
Σχήμα 4.16 Προσομοίωση κρυπτογράφησης XTS-AES-128 α' μέρος



Σχήμα 4.17 Προσομοίωση κρυπτογράφησης XTS-AES-128 β' μέρος



Σχήμα 4.18 Προσομοίωση κρυπτογράφησης XTS-AES-128 γ' μέρος



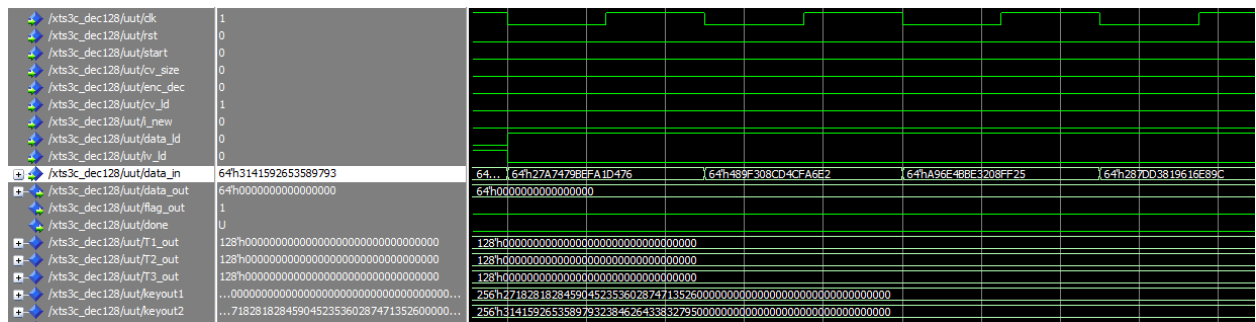
Σχήμα 4.19 Προσομοίωση κρυπτογράφησης XTS-AES-128 δ' μέρος

4.2.2 Προσομοίωση αποκρυπτογράφησης XTS-AES-128

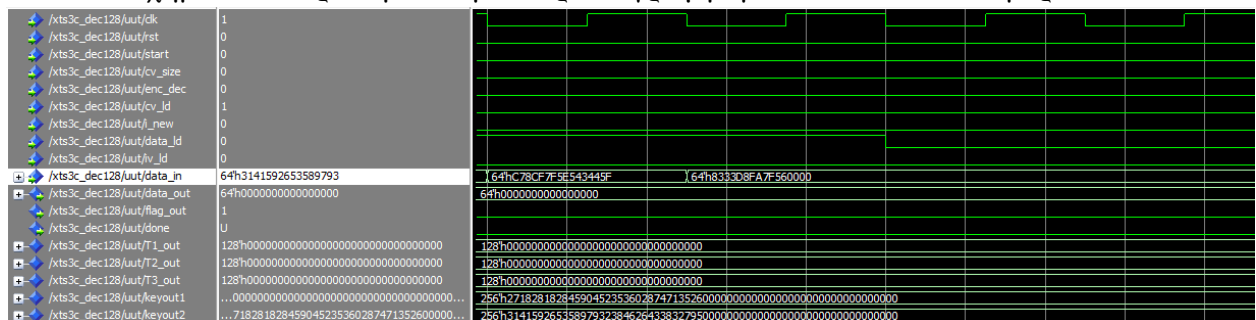
Στην ενότητα αυτή παρουσιάζεται η προσομοίωση της αρχιτεκτονικής τριών πυρήνων για την αποκρυπτογράφηση με μήκος κλειδιού 128-bits.

Αποκρυπτογράφηση XTS-AES-128	
Data_in	27a7479bfa1d476489f308cd4cfa6e2 a96e4bbe3208ff25287dd3819616e89c c78cf7f5e543445f8333d8fa7f560000
i value	00000000000000000000000000000000 00000000000000000000000000000000
Key	27182818284590452353602874713526 31415926535897932384626433832795
Data_out	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f 202122232425262728292a2b2c2d2e2f

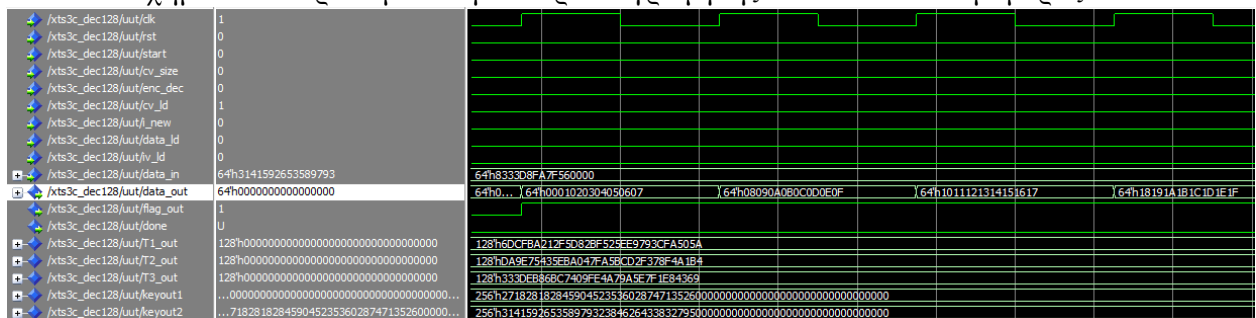
Πίνακας 4.7 Δεδομένα αποκρυπτογράφησης XTS-AES-128



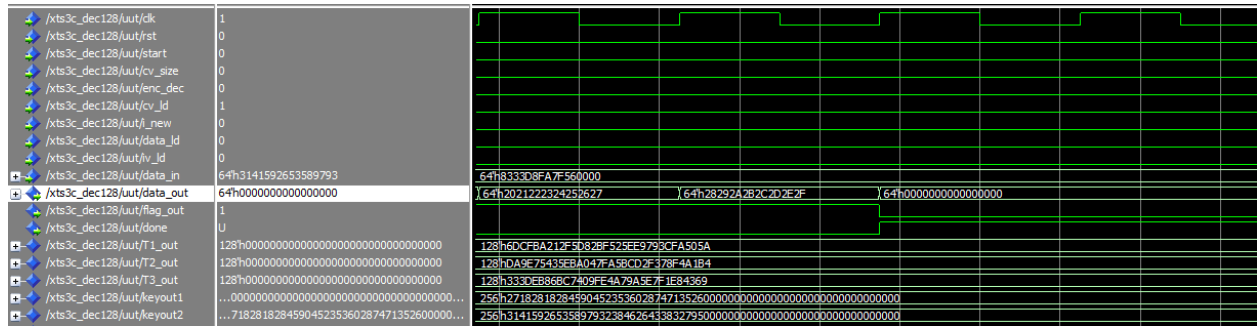
Σχήμα 4.20 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 α' μέρος



Σχήμα 4.21 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 β' μέρος



Σχήμα 4.22 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 γ' μέρος



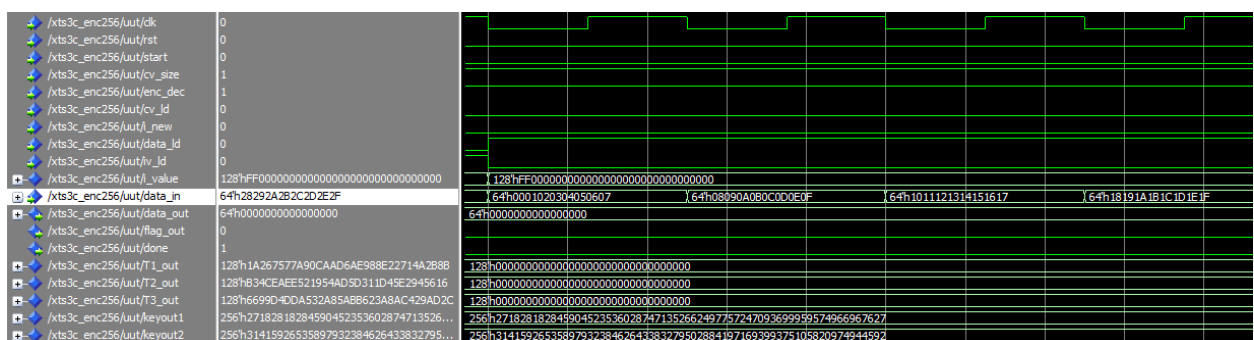
Σχήμα 4.23 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 δ' μέρος

4.2.3 Προσομοίωση κρυπτογράφησης XTS-AES-256

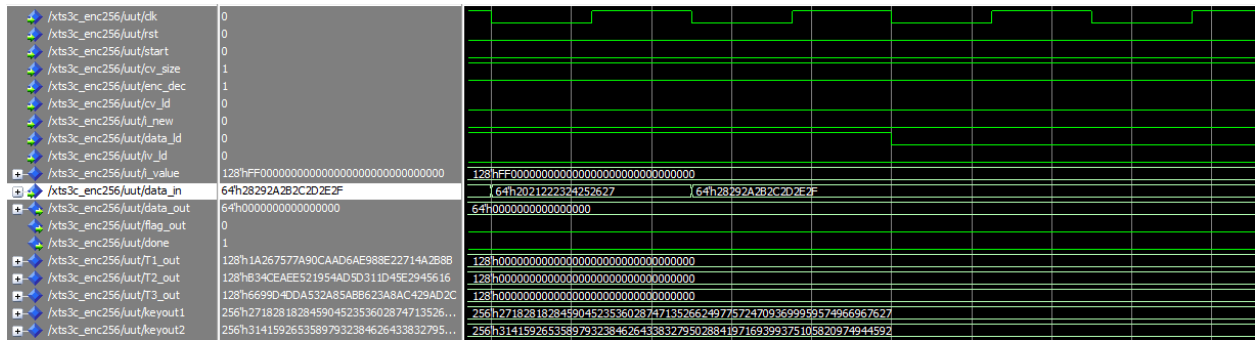
Στην ενότητα αυτή παρουσιάζεται η προσομοίωση της αρχιτεκτονικής τριών πυρήνων για τη κρυπτογράφηση με μήκος κλειδιού 256-bits.

Κρυπτογράφηση XTS-AES-256	
Data_in	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f 202122232425262728292a2b2c2d2e2f
i value	00000000000000000000000000000000 000000000000000000000000000000ff
Key	27182818284590452353602874713526 62497757247093699959574966967627 31415926535897932384626433832795 02884197169399375105820974944592
Data_out	1c3b3a102f770386e4836c99e370cf9b ea00803f5e482357a4ae12d414a3e63b 5d31e276f8fe4a8d66b317f9ac683f44

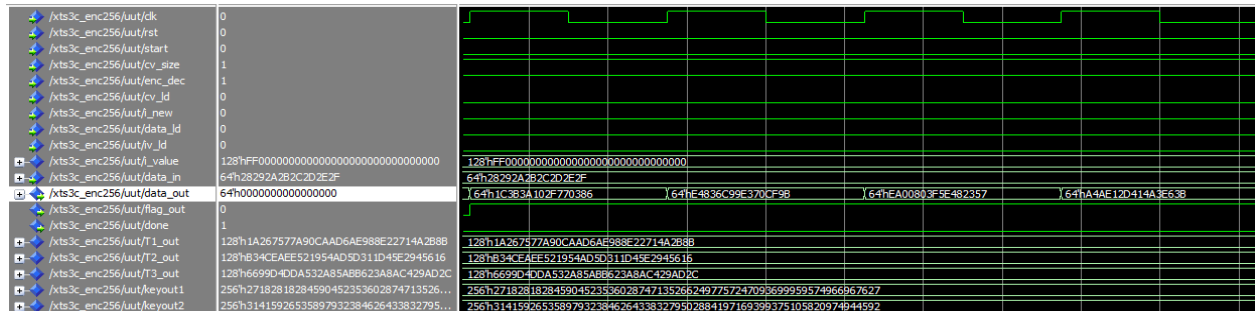
Πίνακας 4.8 Δεδομένα κρυπτογράφησης XTS-AES-256



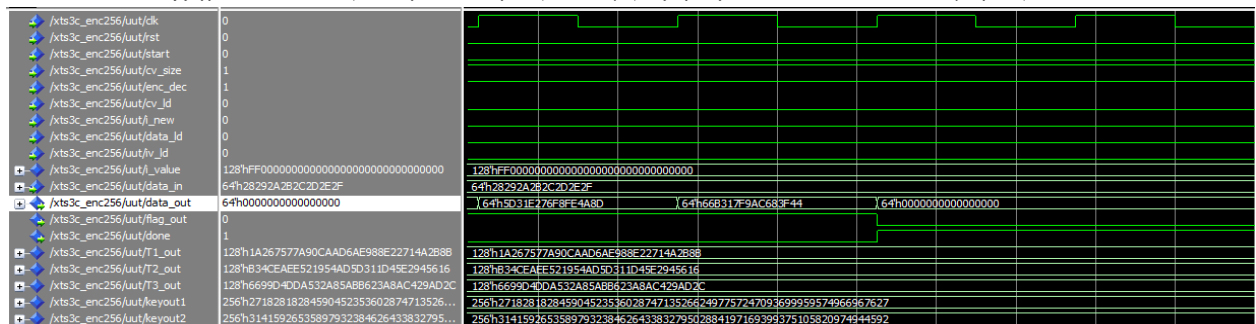
Σχήμα 4.24 Προσομοίωση κρυπτογράφησης XTS-AES-256 α' μέρος



Σχήμα 4.25 Προσομοίωση κρυπτογράφησης XTS-AES-256 β' μέρος



Σχήμα 4.26 Προσομοίωση κρυπτογράφησης XTS-AES-256 γ' μέρος



Σχήμα 4.27 Προσομοίωση κρυπτογράφησης XTS-AES-256 δ' μέρος

4.2.4 Προσομοίωση αποκρυπτογράφησης XTS-AES-256

Στην ενότητα αυτή παρουσιάζεται η προσομοίωση της αρχιτεκτονικής τριών πυρήνων για την αποκρυπτογράφηση με μήκος κλειδιού 128-bits.

Αποκρυπτογράφηση XTS-AES-256	
Data_in	27a7479befa1d476489f308cd4cfa6e2 a96e4bbe3208ff25287dd3819616e89c c78cf7f5e543445f8333d8fa7f560000
i value	00000000000000000000000000000000 00000000000000000000000000000000
Key	27182818284590452353602874713526 31415926535897932384626433832795
Data_out	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f 202122232425262728292a2b2c2d2e2f

Πίνακας 4.9 Δεδομένα αποκρυπτογράφησης XTS-AES-256

[illegible]

Σχήμα 4.28 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 α' μέρος

[illegible]

Σχήμα 4.29 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 β' μέρος

[illegible]

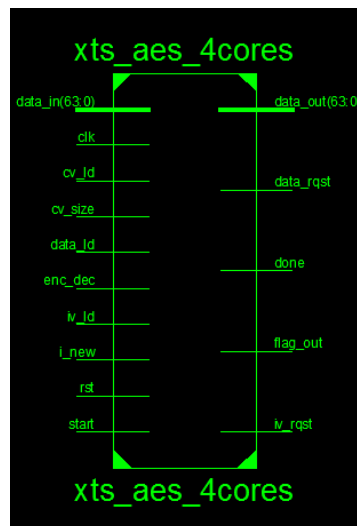
Σχήμα 4.30 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 γ' μέρος

[illegible]

Σχήμα 4.31 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 δ' μέρος

4.3 Προσομοίωση αρχιτεκτονικής τεσσάρων πυρήνων

Στη τελευταία ενότητα του κεφαλαίου αυτού παρουσιάζονται οι προσομοιώσεις, τόσο της κρυπτογράφησης όσο και της αποκρυπτογράφησης με μήκους 128 και 256 bits, της αρχιτεκτονικής τεσσάρων πυρήνων (XTS-AES-4CORES). Στο σχήμα 4.31 φαίνονται οι είσοδοι και τις έξοδοι του κυκλώματος των οποίων η λειτουργία έχει περιγραφεί στο πίνακα 4.1



Σχήμα 4.32 Είσοδοι/Έξοδοι του XTS-AES-4CORES

4.3.1 Προσομοίωση κρυπτογράφησης XTS-AES-128

Στην υποενότητα αυτή παρουσιάζεται η προσομοίωση της αρχιτεκτονικής τεσσάρων πυρήνων για τη κρυπτογράφηση με μήκος κλειδιού 128-bits.

Κρυπτογράφηση XTS-AES-128	
Data_in	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f 202122232425262728292a2b2c2d2e2f 303132333435363738393a3b3c3d3e3f
i value	00000000000000000000000000000000 00000000000000000000000000000000
Key	27182818284590452353602874713526 31415926535897932384626433832795
Data_out	27a7479befa1d476489f308cd4cfa6e2 a96e4bbe3208ff25287dd3819616e89c c78cf7f5e543445f8333d8fa7f560000 05279fa5d8b5e4ad40e736ddb4d35412

Πίνακας 4.10 Δεδομένα κρυπτογράφησης XTS-AES-128

4.3.2 Προσομοίωση αποκρυπτογράφησης XTS-AES-128

Στην υποενότητα αυτή παρουσιάζεται η προσομοίωση της αρχιτεκτονικής τεσσάρων πυρήνων για την αποκρυπτογράφηση με μήκος κλειδιού 128-bits.

Αποκρυπτογράφηση XTS-AES-128	
Data_in	27a7479befa1d476489f308cd4cfa6e2 a96e4bbe3208ff25287dd3819616e89c c78cf7f5e543445f8333d8fa7f560000 05279fa5d8b5e4ad40e736ddb4d35412
<i>i value</i>	00000000000000000000000000000000 00000000000000000000000000000000
Key	27182818284590452353602874713526 31415926535897932384626433832795
Data_out	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f 202122232425262728292a2b2c2d2e2f 303132333435363738393a3b3c3d3e3f

Πίνακας 4.11 Δεδομένα αποκρυπτογράφησης XTS-AES-128

[illegible]

Σχήμα 4.37 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 α' μέρος

[illegible]

Σχήμα 4.38 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 β' μέρος

[illegible]

Σχήμα 4.39 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 γ' μέρος

[illegible]

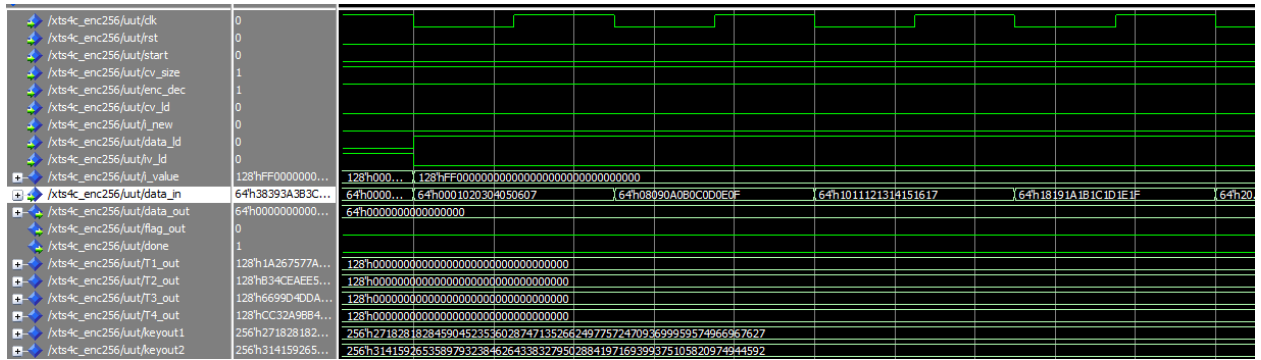
Σχήμα 4.40 Προσομοίωση αποκρυπτογράφησης XTS-AES-128 δ' μέρος

4.3.3 Προσομοίωση κρυπτογράφησης XTS-AES-256

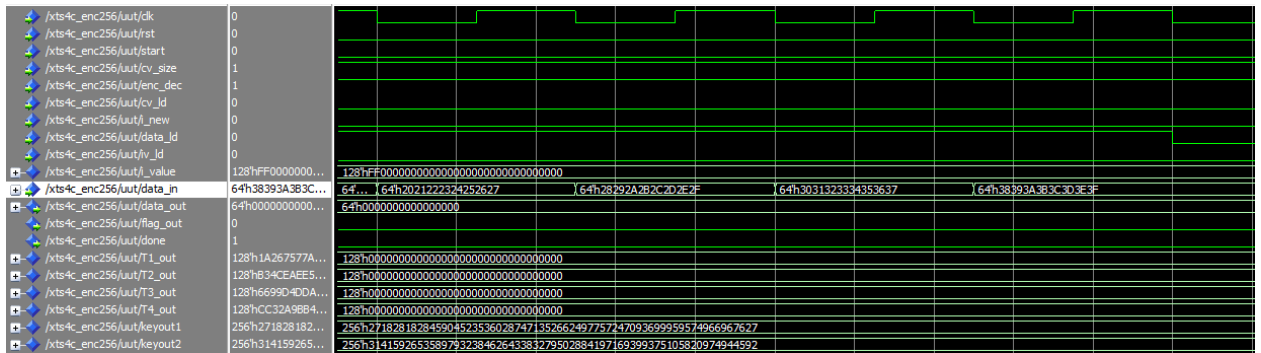
Στην ενότητα αυτή παρουσιάζεται η προσομοίωση της αρχιτεκτονικής τριών πυρήνων για τη κρυπτογράφηση με μήκος κλειδιού 256-bits.

Κρυπτογράφηση XTS-AES-256	
Data_in	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f 202122232425262728292a2b2c2d2e2f 303132333435363738393a3b3c3d3e3f
<i>i value</i>	00000000000000000000000000000000 000000000000000000000000000000ff
Key	27182818284590452353602874713526 62497757247093699959574966967627 31415926535897932384626433832795 02884197169399375105820974944592
Data_out	1c3b3a102f770386e4836c99e370cf9b ea00803f5e482357a4ae12d414a3e63b 5d31e276f8fe4a8d66b317f9ac683f44 680a86ac35adfc3345befecb4bb188fd

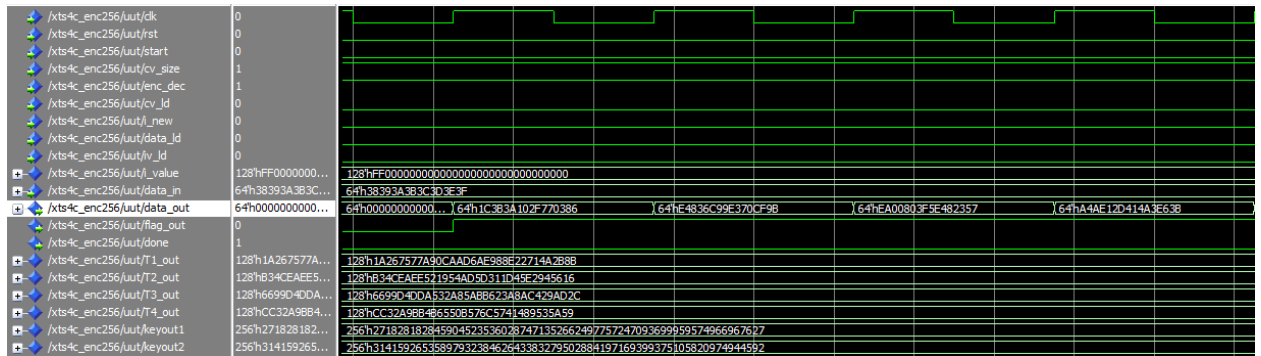
Πίνακας 4.12 Δεδομένα κρυπτογράφησης XTS-AES-256



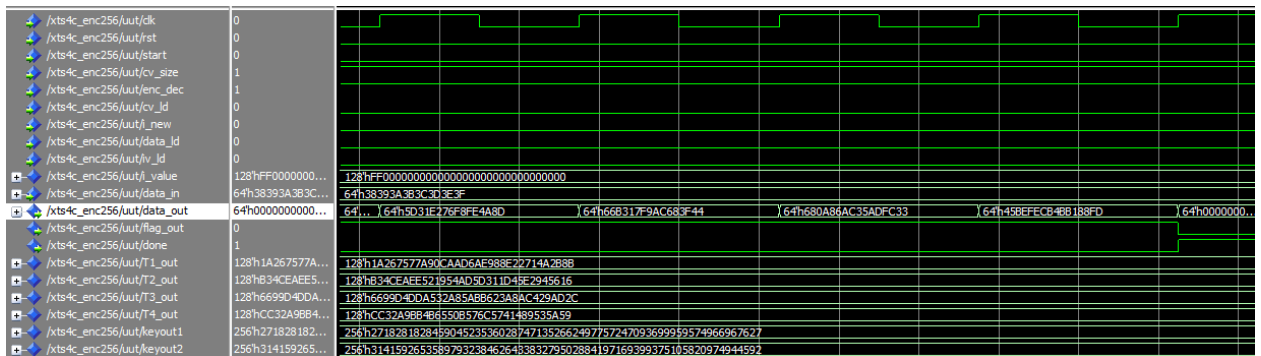
Σχήμα 4.41 Προσομοίωση κρυπτογράφησης XTS-AES-256 α' μέρος



Σχήμα 4.42 Προσομοίωση κρυπτογράφησης XTS-AES-256 β' μέρος



Σχήμα 4.43 Προσομοίωση κρυπτογράφησης XTS-AES-256 γ' μέρος



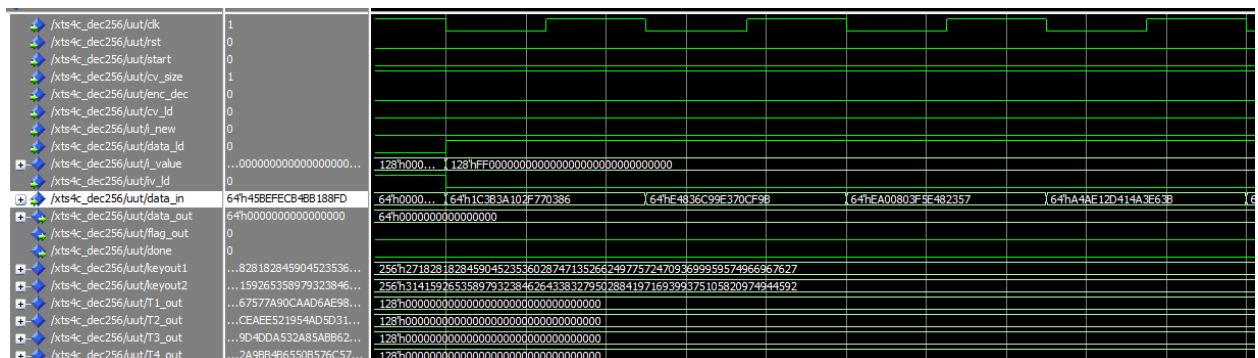
Σχήμα 4.44 Προσομοίωση κρυπτογράφησης XTS-AES-256 δ' μέρος

4.3.4 Προσομοίωση αποκρυπτογράφησης XTS-AES-256

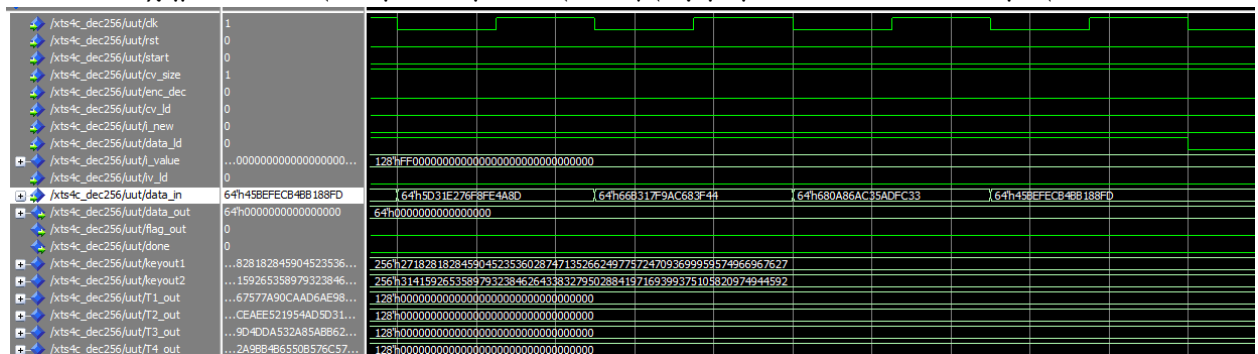
Στην τελευταία υποενότητα παρουσιάζεται η προσομοίωση της αρχιτεκτονικής τεσσάρων πυρήνων για τη αποκρυπτογράφηση με μήκος κλειδιού 256-bits.

Αποκρυπτογράφηση XTS-AES-256	
Data_in	1c3b3a102f770386e4836c99e370cf9b ea00803f5e482357a4ae12d414a3e63b 5d31e276f8fe4a8d66b317f9ac683f44 680a86ac35adfc3345befecb4bb188fd
i value	00000000000000000000000000000000 000000000000000000000000000000ff
Key	27182818284590452353602874713526 62497757247093699959574966967627 31415926535897932384626433832795 02884197169399375105820974944592
Data_out	000102030405060708090a0b0c0d0e0f 101112131415161718191a1b1c1d1e1f 202122232425262728292a2b2c2d2e2f 303132333435363738393a3b3c3d3e3f

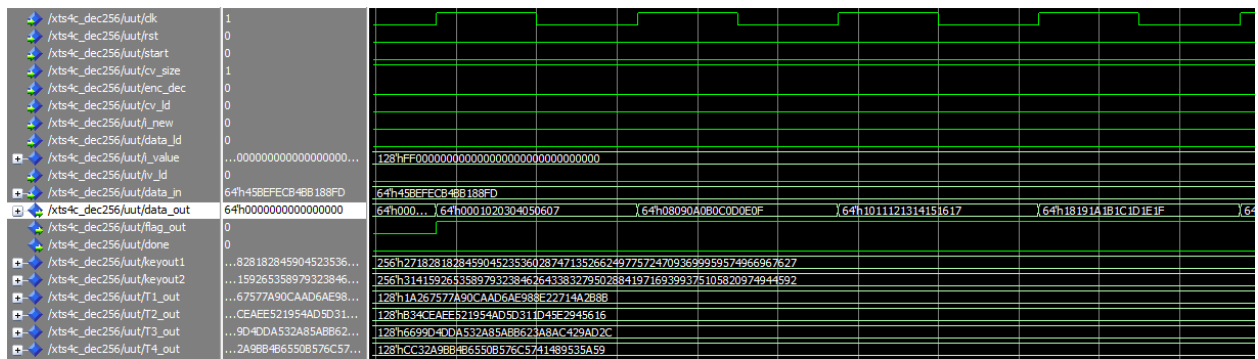
Πίνακας 4.13 Δεδομένα αποκρυπτογράφησης XTS-AES-256



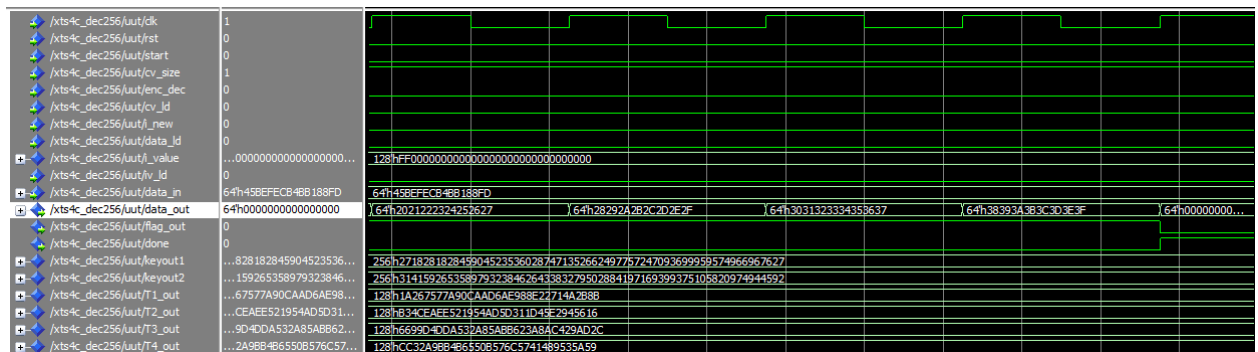
Σχήμα 4.45 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 α' μέρος



Σχήμα 4.46 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 β' μέρος



Σχήμα 4.47 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 γ' μέρος



Σχήμα 4.48 Προσομοίωση αποκρυπτογράφησης XTS-AES-256 δ' μέρος

Όπως προκύπτει από όλες τις επιμέρους προσομοιώσεις, οι αρχιτεκτονικές που παρουσιάστηκαν λειτουργούν αποτελεσματικά τόσο στη κρυπτογράφηση όσο και στην αποκρυπτογράφηση είτε πρόκειται για 128-bits κλειδί είτε για 256-bits.

Κεφάλαιο 5

Αποτελέσματα

Στο πέμπτο κεφάλαιο αυτής της διπλωματικής εργασίας παρουσιάζονται τα αποτελέσματα, όπως αυτά παρήχθησαν μέσω του λογισμικού Xilinx ISE Design Suite 14.2. Τα αποτελέσματα αφορούν τόσο τις αρχιτεκτονικές που προκύπτουν με βάση το προτεινόμενο χρονοπρογραμματισμό όσο τις αρχιτεκτονικές που υιοθετούν τον υπάρχον χρονοπρογραμματισμό. Αυτό έγινε ούτως ώστε να μπορεί να υπάρξει και μια πραγματική σύγκριση μεταξύ των δύο τύπων αρχιτεκτονικών πέραν από τη ποιοτική σύγκριση που κάναμε στο κεφάλαιο 3.

Στις πίνακες και στα γραφήματα αυτού του κεφαλαίου έχουμε επικεντρωθεί σε τρεις βασικές παραμέτρους αυτές είναι α) ο ρυθμός διαμεταγωγής δεδομένων (*throughput*) που δείχνει τον όγκο δεδομένων που μπορεί να επεξεργαστεί το κύκλωμα μέσα σε ένα δευτερόλεπτο, β) η επιφάνεια που καταλαμβάνει το κάθε κύκλωμα (*Area*) και γ) ο ρυθμός διαμεταγωγής δεδομένων ανά επιφάνεια (*throughput/area*) ο οποίος ουσιαστικά είναι ένας ποιοτικός δείκτης που δείχνει τη συσχέτιση μεταξύ των δύο παραμέτρων. Εννοείται ότι αν

ενδιαφερόμαστε για κυκλώματα που απαιτούν μια ισορροπία ανάμεσα σε *throughput* και *area* τότε μας ενδιαφέρει η παράμετρος *throughput/area* να έχει όσο το δυνατόν μεγαλύτερη τιμή γίνεται.

Οι μονάδες που μετριοούνται τα παραπάνω μεγέθη είναι *Mbps* για το *throughput*, *slices* για το *area* και *Mbps/slices* για το *throughput/area*. Τα *Mbps*, όπως είπαμε και προηγουμένως, δείχνουν πόσα *MegaBit* επεξεργάζεται το κύκλωμα ανά μονάδα χρόνου. Οι μονάδες στις οποίες χωρίζεται ένα FPGA είναι τα CLBs(Configurable Logic Blocks). Τα CLBs από τη άλλη μεριά αποτελούνται από *slices*. Τα *slices* με τη σειρά τους αποτελούνται από LUTs(look-up tables), πολυπλέκτες και D-FFs ούτως ώστε το FPGA να μπορέσει να δημιουργήσει οποιοδήποτε κύκλωμα απαιτηθεί από το χρήστη. Τέλος, το *throughput/area* που μετριέται σε *Mbps/slices*, δείχνει το μέσο ρυθμό διαμεταγωγής που μπορεί να αποδώσει το κάθε *slice*. Να σημειώσουμε εδώ ότι το *throughput/area* είναι ο λόγος των δύο παραμέτρων που μόλις αναφέρθηκαν και δεν αποτελεί κάποια ανεξάρτητη παράμετρο.

Οι μετρήσεις και τα αποτελέσματα που παρουσιάζονται σε αυτό το κεφάλαιο προέκυψαν από διαφορετικά FPGAs[20]. Οι οικογένειες και τα FPGA που χρησιμοποιήθηκαν φαίνονται στο παρακάτω πίνακα.

Family	Virtex 6	Virtex 5	Spartan 6
Device	xc6vlx130t	xc5vlx220	xc6slx100t
Package	ff1156	ff1760	fgg484
Speed	2	2	4

Πίνακας 5.1 Λεπτομέρειες των FPGAs

Αρχικά, παρουσιάζουμε το πίνακα των αποτελεσμάτων όπως αυτά προέκυψαν από την οικογένεια Virtex 6 μετά από *place & route* διαδικασία. Τα αποτελέσματα αυτά αφορούν τις αρχιτεκτονικές που ακολουθούν τον προτεινόμενο χρονοπρογραμματισμό. Οι αρχιτεκτονικές αυτές θα ονομάζονται *arc2_2cores*, *arc2_3cores* και *arc2_4cores*. Αντίστοιχα, εκείνες που ακολουθούν τον υπάρχον χρονοπρογραμματισμό θα ονομάζονται *arc1_2cores* και *arc1_4cores* ούτως ώστε να γίνεται αντιληπτό ποια αρχιτεκτονική σχολιάζεται κάθε φορά.

	Cycles	F _{max} (MHz)	Throughput (Mbps)	Area (Slices)	Throughput/Area (Mbps/Slices)
Arc2_2cores	3972	161,394	3328,65	5233	0.636
Arc2_3cores	2652	159,515	4927,4	6421	0.767
Arc2_4cores	2040	157,456	6322,94	9090	0.696

Πίνακας 5.2 Αποτελέσματα Virtex6

Ας δούμε αναλυτικά τι περιγράφει η κάθε στήλη και πως προκύπτουν τα παραπάνω δεδομένα. Καταρχήν, έχουμε κάνει τη προσέγγιση ότι τα δεδομένα που πρόκειται να

κρυπτογραφηθούν είναι ίσα με τη χωρητικότητα που έχουν είκοσι *sectors* ενός σκληρού δίσκου. Αυτό σημαίνει ότι ο αριθμός των δεδομένων που πρόκειται να επεξεργαστεί το κύκλωμα ανέρχεται σε 81920 bits ή αλλιώς 10.24 Kbytes εφόσον το κάθε *sector* έχει χωρητικότητα 4096 bits. Οι κύκλοι επεξεργασίας, για έναν τέτοιο όγκο δεδομένων, ανέρχονται στους 331, όσον αφορά την *arc2_2cores*, 221 όσον αφορά την *arc2_3cores* και 170 την αρχιτεκτονική *arc2_4cores*. Κάθε κύκλος επεξεργασίας όμως αποτελείται από δώδεκα παλμούς ρολογιού. Οπότε, για να υλοποιηθεί η παραπάνω διαδικασία απαιτούνται συνολικά 3972, 2652 και 2040 παλμοί ρολογιού για την κάθε αρχιτεκτονική αντίστοιχα.

Η μέγιστη συχνότητα (F_{max}) που παρουσιάζεται στο πίνακα 3.2 έχει καταγραφεί όπως αυτή προέκυψε από το λογισμικό μετά από τη διαδικασία *place & route* και με χρήση διαφόρων αρχείων χρονικών περιορισμών (*timing constraint files*). Η μέγιστη συχνότητα παίζει καθοριστικό ρόλο στη διαμόρφωση του ρυθμού διαμεταγωγής δεδομένων. Το *throughput* αποτελεί τη παράμετρο εκείνη που δείχνει τη ταχύτητα του κυκλώματος και μπορεί να υπολογιστεί με βάση το παρακάτω τύπο

$$Throughput = \frac{(bits \text{ που επεξεργάστηκαν}) \cdot (Μέγιστη \text{ συχνότητα λειτουργίας})}{Κύκλοι \text{ ρολογιού που χρειάστηκαν}} \quad 5.1$$

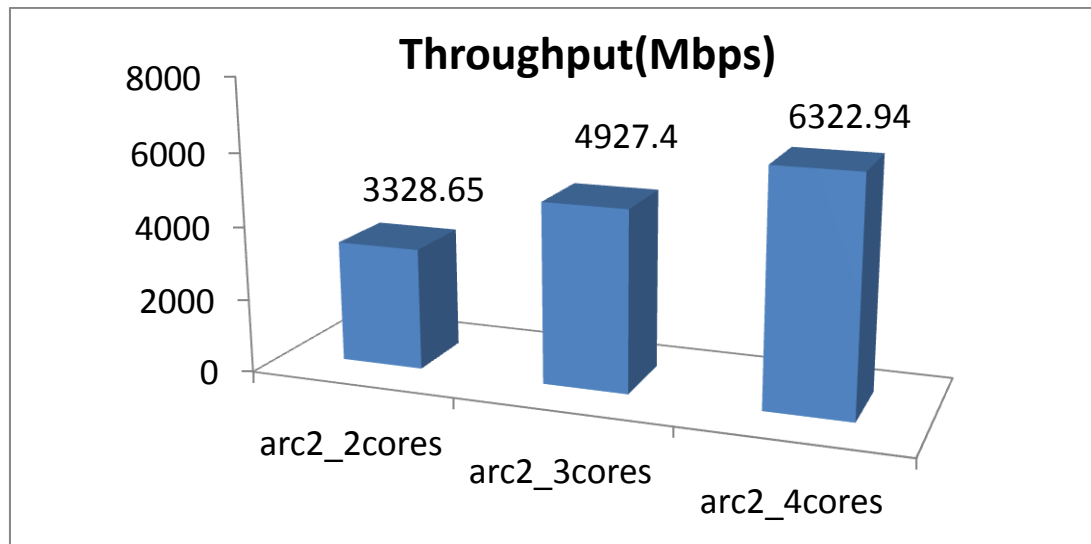
Συμφωνά με το παραπάνω τύπο λοιπόν, πέραν της μέγιστης συχνότητας, καθοριστικό ρόλο παίζουν και οι παλμοί ρολογιού που απαιτούνται για να ολοκληρωθεί η διαδικασία. Όσο λιγότεροι είναι οι παλμοί ρολογιού που απαιτούνται τόσο αυξάνει ο ρυθμός διαμεταγωγής δεδομένων καθώς είναι μεγέθη αντιστρόφως ανάλογα.

Με βάση το παραπάνω τύπο, υπολογίζεται ο ρυθμός διαμεταγωγής δεδομένων και για τις τρεις αρχιτεκτονικές. Ο ρυθμός διαμεταγωγής δεδομένων που μπορεί να πιάσει η αρχιτεκτονική *arc2_2cores* είναι της τάξης των 3328,65 Mbps ή περίπου 3,33 Gbps ενώ η αρχιτεκτονική *arc2_3cores* μπορεί να προσεγγίσει ταχύτητες μέχρι και 4927,4 Mbps ή περίπου 4,93 Gbps. Η αύξηση του ρυθμού διαμεταγωγής οφείλεται, όπως είναι προφανές, στην αύξηση του αριθμού των πυρήνων από δύο σε τρεις τους οποίους ενσωματώνει η αρχιτεκτονική *arc2_3cores*. Στο κεφάλαιο 3, είχε γίνει μια ποιοτική σύγκριση μεταξύ των αρχιτεκτονικών *arc2_3cores* και *arc3_cores*. Η αύξηση που αναμενόταν, όσον αφορά το ρυθμό διαμεταγωγής, ήταν περίπου της τάξης του 50%. Αν λάβουμε τα πειραματικά νούμερα του ρυθμού διαμεταγωγής που παρουσιάζονται στο πίνακα 5.2 και τα ανάγουμε σε ποσοστά, τότε η αύξηση ανέρχεται περίπου στο ποσοστό του 48,05% το οποίο είναι εξαιρετικό διότι συμβαδίζει με τη θεωρητική προσέγγιση.

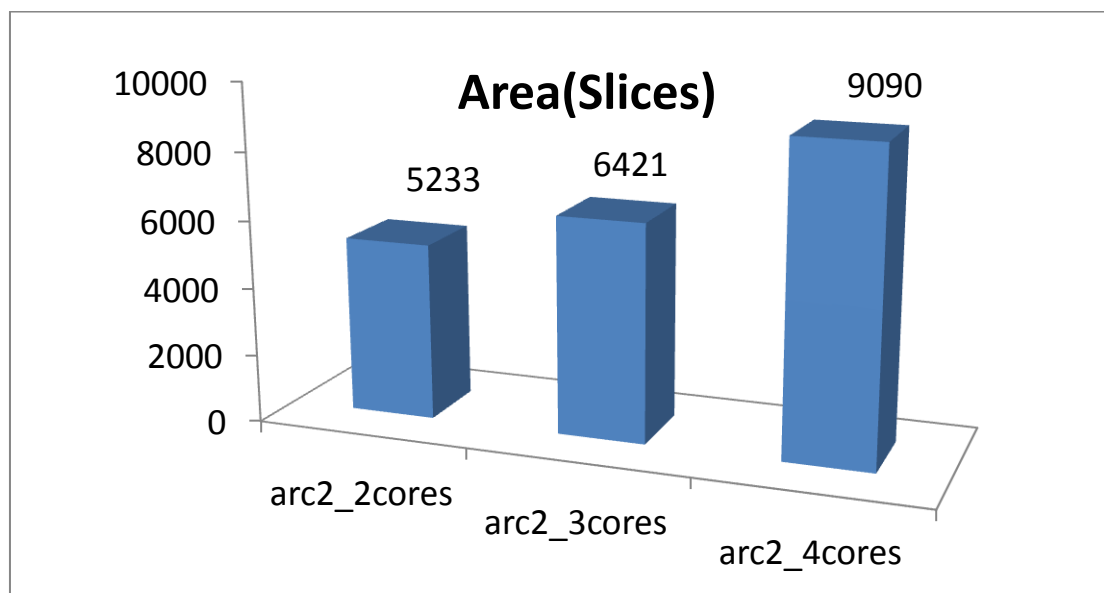
Από την άλλη μεριά, τα επίπεδα του ρυθμού διαμεταγωγής δεδομένων, όσον αφορά την αρχιτεκτονική *arc2_4cores*, μπορούν να φτάσουν έως και 6322,94 Mbps ή αλλιώς 6,32 Gbps. Η πειραματική αύξηση είναι της τάξης του 90,91% σε σύγκριση με την αρχιτεκτονική *arc2_2cores* και αποτελεί ένα άκρως ικανοποιητικό αποτέλεσμα το οποίο δεν απέχει πολύ από το 95% που θεωρητικά αναμέναμε. Επίσης, σημαντικό ρόλο παίζει το γεγονός ότι με την αύξηση του αριθμού των πυρήνων σε κάθε νέα αρχιτεκτονική, αυξάνει και ο βαθμός

πολυπλοκότητας υλοποίησης της. Συμπερασματικά και όσον αφορά τα αποτελέσματα της οικογένειας Virtex6 μπορούμε να πούμε ότι υπάρχει μια σχεδόν γραμμική αύξηση του ρυθμού διαμεταγωγής δεδομένων καθώς αυξάνουν οι πυρήνες που ενσωματώνει η κάθε λογική.

Τέλος, μπορούμε να δούμε γραφικά στο σχήμα 5.1 τους ρυθμούς διαμεταγωγής δεδομένων συναρτήσει του αριθμού των πυρήνων της κάθε αρχιτεκτονικής.



Σχήμα 5.1 Throughput(Mbps), Virtex6

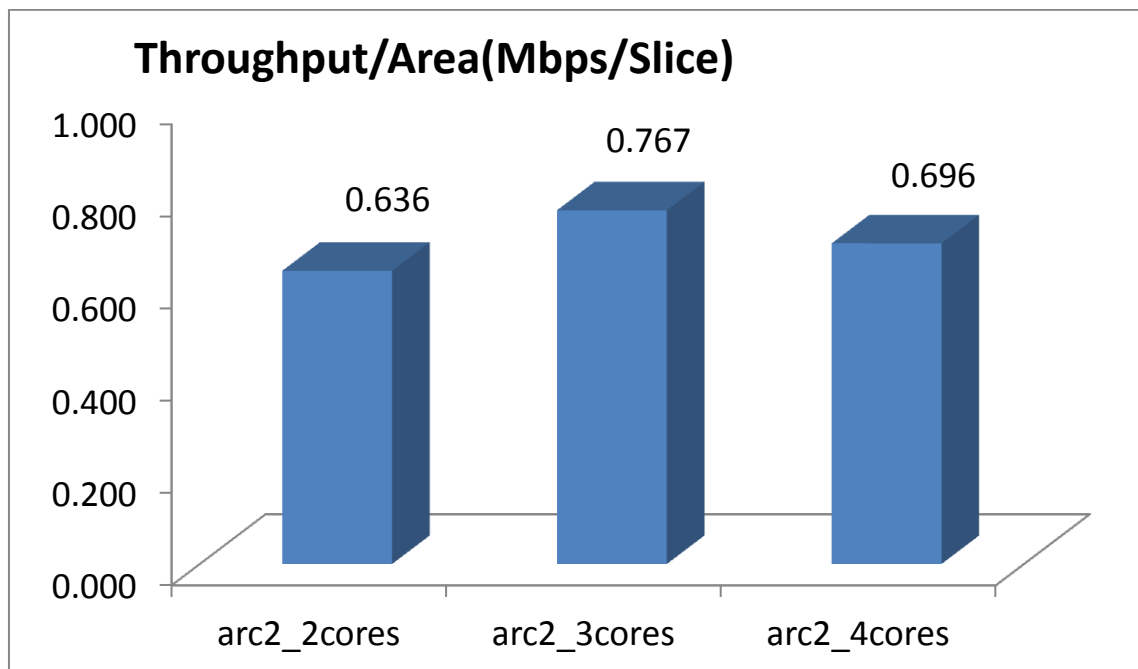


Σχήμα 5.2 Area(Slices), Virtex6

Στο σχήμα 5.2 μπορούμε να δούμε γραφικά την επιφάνεια που καταλαμβάνει η κάθε μια από τις προτεινόμενες αρχιτεκτονικές. Παρατηρούμε ότι η arc2_2cores καταλαμβάνει 5233 slices ενώ η arc2_3cores καταλαμβάνει 6421 slices. Η αύξηση αυτή είναι της τάξης του

22,7% και δείχνει ότι η θεωρητική προσέγγιση δεν συμβαδίζει με το πειραματικό αποτέλεσμα καθώς αναμενόταν αύξηση της τάξης του 50% όπως και στο ρυθμό διαμεταγωγής δεδομένων. Πραγματοποιώντας την ίδια σύγκριση μεταξύ των αρχιτεκτονικών arc2_2cores και arc2_4cores διαπιστώνουμε ότι η αρχιτεκτονική τεσσάρων πυρήνων εμφανίζει αύξηση της επιφάνειας της κατά 73,71%. Ακόμη και στη περίπτωση αυτή, τα αποτελέσματα διαφέρει από το αναμενόμενο το οποίο θα το περιμέναμε στα επίπεδα του 90-95%.

Τα αποτελέσματα της επιφάνειας, που διαφέρουν από τα θεωρητικά, μπορούν να δικαιολογηθούν έως ένα βαθμό από το γεγονός ότι τα slices που αναφέρουμε ως μονάδα μέτρησης δεν είναι πλήρως εκμεταλλεύσιμα. Αφήνει δηλαδή αρκετούς ανεκμετάλλευστους πόρους οι οποίοι όμως φαίνονται δεσμευμένοι από το λογισμικό. Βεβαίως, το φαινόμενο αυτό συμβαίνει και στην arc2_2cores που έχουμε ως σημείο αναφοράς και ίσως στη συγκεκριμένη περίπτωση να είναι και εντονότερο καθώς αποτελεί τη βάση που πατάνε και οι υπόλοιπες αρχιτεκτονικές. Σε γενικές γραμμές όμως η καταγραφή των slices ή των occupied slices αποτελεί την πιο αξιόπιστη μέτρηση που μπορεί να ληφθεί από ένα FPGA.



Σχήμα 5.3 Throughput/Area(Mbps/Slice)

Ένας ακόμη παράγοντας που παίζει σημαντικό ρόλο στα ψηφιακά κυκλώματα είναι ο παράγοντας *throughput/area*. Ουσιαστικά, πρόκειται για έναν ποιοτικό δείκτη που δείχνει το μέσο ρυθμό διαμεταγωγής δεδομένων ανά μονάδα επιφάνειας (slice). Όπως βλέπουμε στο σχήμα 5.3, η αρχιτεκτονική arc2_3cores μπορεί να φτάσει τα 0,767 Mbps/slice και επιβεβαιώνει το γεγονός ότι πρόκειται για την πιο ισορροπημένη αρχιτεκτονική μεταξύ ταχύτητας και επιφάνειας. Ακόμη, η αρχιτεκτονική arc2_4cores μπορεί να παράγει ένα πολύ ικανοποιητικό αποτέλεσμα της τάξης των 0,696 Mbps/slice όπως και η arc2_2cores που να φτάσει στα επίπεδα του 0,636 Mbps/slice.

Στη συνέχεια παρουσιάζουμε τους πίνακες με τα αποτελέσματα έτσι όπως αυτά προέκυψαν από τις οικογένειες Virtex5(Πίνακας 5.3) και Spartan6(Πίνακας 5.4).

	Cycles	F _{max} (MHz)	Throughput (Mbps)	Area (Slices)	Throughput/Area (Mbps/Slices)
Arc2_2cores	3972	139,880	2884,94	4750	0.607
Arc2_3cores	2652	134,264	4147,4	7302	0.568
Arc2_4cores	2040	128,617	5164,86	8737	0.591

Πίνακας 5.3 Αποτελέσματα Virtex5

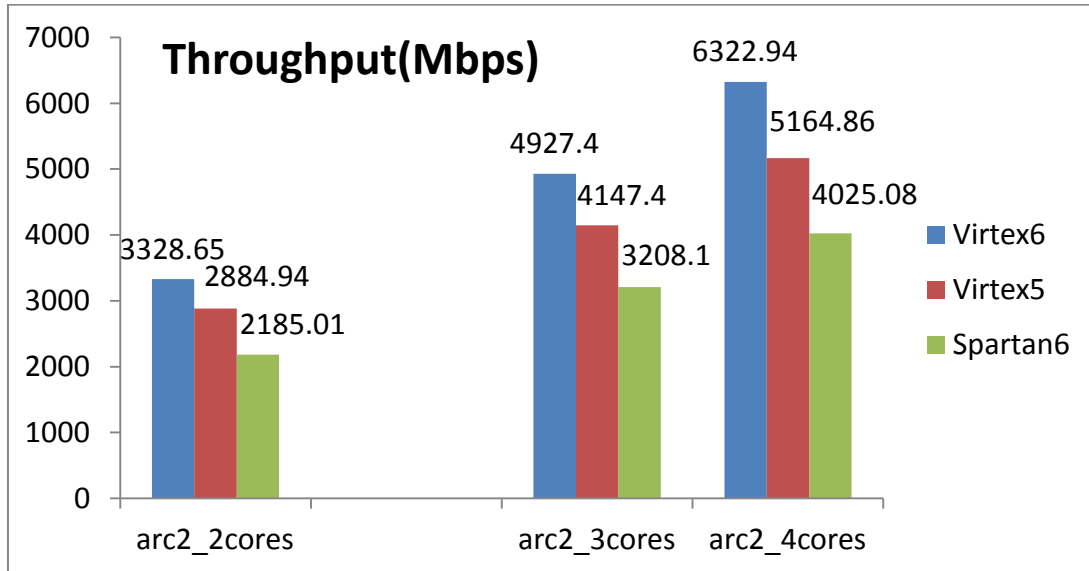
	Cycles	F _{max} (MHz)	Throughput (Mbps)	Area (Slices)	Throughput/Area (Mbps/Slices)
Arc2_2cores	3972	105,943	2185.01	5008	0.436
Arc2_3cores	2652	103,856	3208.1	6685	0.480
Arc2_4cores	2040	100,234	4025.08	9036	0.445

Πίνακας 5.4 Αποτελέσματα Spartan6

Σχολιάζοντας τα αποτελέσματα του Πίνακα 5.3, το πρώτο πράγμα που μπορούμε να παρατηρήσουμε είναι ότι υπάρχει μια πτώση στις τιμές των μεγίστων συχνοτήτων(F_{max}) που έχει άμεσο αντίκτυπο στις τιμές των ρυθμών διαμεταγωγής δεδομένων(*throughput*). Η πτώση αυτή ανέρχεται στα 5~6 MHz για κάθε επιπλέον πυρήνα που ενσωματώνεται στην αρχιτεκτονική. Αυτό συμβαίνει κυρίως διότι το device που επιλέχθηκε προσπαθεί να πάρει τη μέγιστη συχνότητα κάνοντας παράλληλα και βελτιστοποίηση στην επιφάνεια του κυκλώματος. Επίσης, η οικογένεια Virtex5 είναι υλοποιημένη σε διαφορετική τεχνολογία ολοκλήρωσης, technology 65nm, από ότι η οικογένεια Virtex6, technology 40nm. Τέλος, σημαντικό ρόλο παίζει το γεγονός ότι κάθε slice της οικογένειας Virtex6 περιέχει οχτώ flip-flops ενώ αντίστοιχα στην οικογένεια Virtex5 υπάρχουν μόνο τέσσερα αυξάνοντας με αυτόν τον τρόπο τη πολυπλοκότητα υλοποίησης του κυκλώματος.

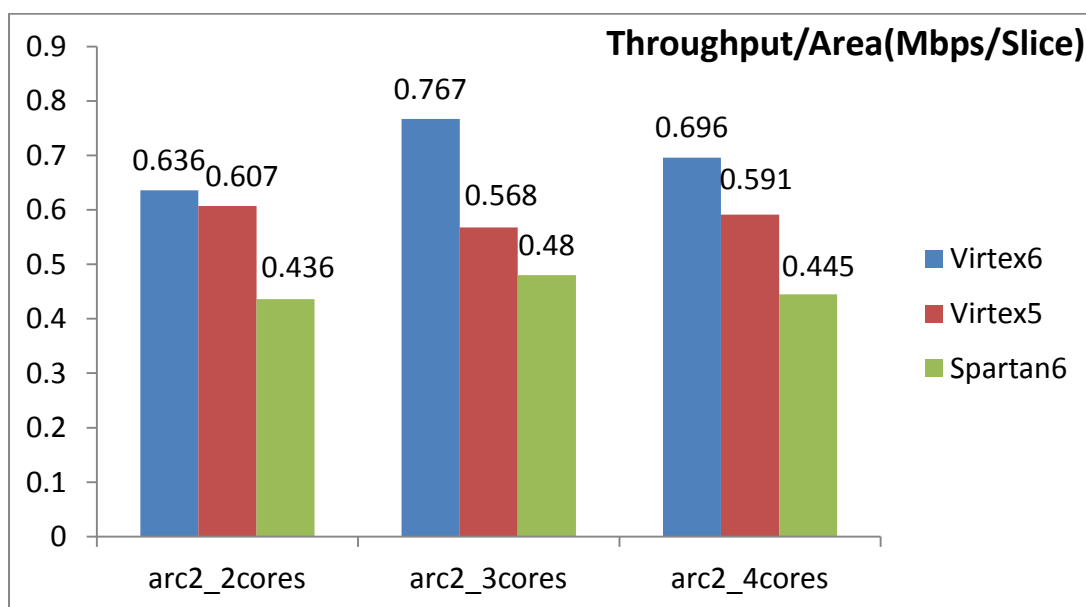
Όσον αφορά τα αποτελέσματα της οικογένειας Spartan6, παρατηρούμε ότι, ενώ οι μέγιστες συχνότητες ήταν περίπου όλες της τάξης των 160MHz, έχουν πέσει τουλάχιστον κατά 50 MHz φτάνοντας να έχουν μέγιστη συχνότητα περίπου 100MHz. Αυτό συμβαίνει διότι οι οικογένειες Virtex γενικά είναι υλοποιημένες σε καλύτερες τεχνολογίες, διαθέτουν βελτιωμένες αρχιτεκτονικές, μεγαλύτερες χωρητικότητες κ.α. Γενικά, η οικογένεια Virtex της Xilinx έχει σχεδιαστεί ούτως ώστε να πετυχαίνει καλύτερους χρόνους και προτείνεται για εφαρμογές που απαιτούν υψηλούς ρυθμούς διαμεταγωγής δεδομένων. Αυτός είναι και ο λόγος που τα Virtex FPGAs είναι ακριβότερα από τα αντίστοιχα Spartan.

Στη συνέχεια παρουσιάζουμε τη συνολική συμπεριφορά των ρυθμών διαμεταγωγής δεδομένων όλων των οικογενειών (Virtex6, Virtex5, Spartan6) και όλων των αρχιτεκτονικών. Στο σχήμα 5.4 μπορούμε να δούμε αυτή τη συμπεριφορά



Σχήμα 5.4 Throughput(Mbps)

Καταρχήν, να εξηγήσουμε ότι το γράφημα είναι χωρισμένο σε τρία τμήματα. Το κάθε τμήμα ουσιαστικά αποτελεί και μια ξεχωριστή αρχιτεκτονική. Τα χρώματα καθορίζουν την οικογένεια του FPGA από το οποίο προέρχεται η μέτρηση. Όσο αφορά την ουσία του γραφήματος, αυτό που μπορούμε να παρατηρήσουμε είναι ότι σε όλες τις αρχιτεκτονικές ο ρυθμός διαμεταγωγής δεδομένων ελαττώνεται καθώς πηγαίνουμε σε FPGAs χαμηλότερου επιπέδου. Αυτό που συμβαίνει με το *throughput* είναι άμεση συνέπεια της χαμηλότερης τιμής που έχουν πλέον οι μέγιστες συχνότητες καθώς *throughput* και μέγιστη συχνότητα χαρακτηρίζονται ως μεγέθη ανάλογα.



Σχήμα 5.5 Throughput/Area(Mbps/Slice)

Ένα ακόμη σημαντικό γράφημα είναι αυτό που φαίνεται στο σχήμα 5.5. Το συγκεκριμένο γράφημα εστιάζει στον μέσο ρυθμό διαμεταγωγής δεδομένων ανά μονάδα επιφάνειας (slice) και αφορά μετρήσεις από όλα τα FPGAs και για όλες τις αρχιτεκτονικές. Η δομή του σχήματος 5.5 είναι ίδια με εκείνη του σχήματος 3.4 που μόλις περιγράφηκε και δεν χρειάζεται περαιτέρω εξήγηση.

Κοιτάζοντας το σχήμα 5.5 ανά αρχιτεκτονική, αυτό που παρατηρούμε, όπως είναι φυσιολογικό κιόλας, είναι ότι καθώς κατεβαίνουμε κατηγορίες στα FPGAs τόσο ελαττώνεται και ο μέσος ρυθμός διαμεταγωγής δεδομένων ανά μονάδα επιφάνειας (slice). Από την άλλη μεριά, το γράφημα αυτό φανερώνει την ποιότητα της οικογένειας Virtex6 καθώς χρησιμοποιώντας καλύτερα το διαθέσιμο υλικό του μπορεί να παράγει εξαιρετικά επίπεδα *throughput* χωρίς να χρειάζεται να χρησιμοποιεί επιπλέον επιφάνεια.

Στη συνέχεια παρουσιάζουμε τους πίνακες με τις αρχιτεκτονικές που αφορούν το τυπικό προγραμματισμό. Στο πίνακα 5.5 μπορούμε να παρατηρήσουμε τη μέγιστη συχνότητα (F_{max}), το ρυθμό διαμεταγωγής δεδομένων (*throughput*), την επιφάνεια (*area*) καθώς και το μέσο ρυθμό διαμεταγωγής δεδομένων ανά μονάδα επιφάνειας (*throughput/area*) τόσο της αρχιτεκτονικής δύο πυρήνων (arc1_2cores) όσο και εκείνης των τεσσάρων πυρήνων (arc1_4cores). Τα αποτελέσματα του πίνακα 5.5 αφορούν την οικογένεια Virtex6 και φαίνονται παρακάτω

	Cycles	F_{max} (MHz)	Throughput (Mbps)	Area (Slices)	Throughput/Area (Mbps/Slices)
Arc1_2cores	4080	157,928	3170.95	5423	0.585
Arc1_4cores	2160	156,348	5929.64	8284	0.716

Πίνακας 5.5 Αποτελέσματα Virtex6

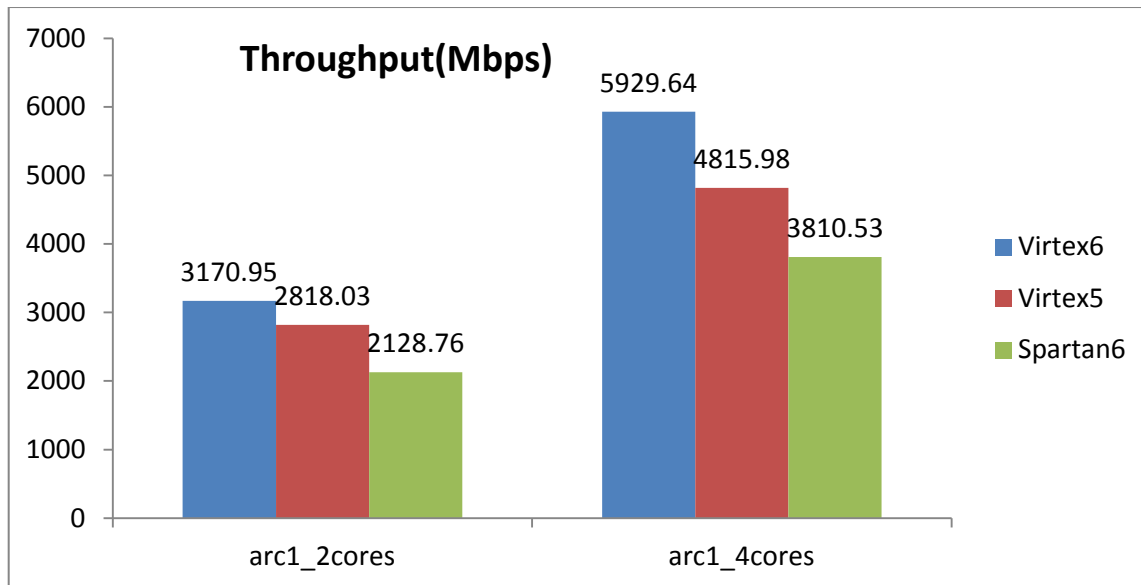
Ακόμη, παραθέτουμε τα αποτελέσματα από τα FPGAs Virtex5 και Spartan6. Τα αποτελέσματα αυτά φαίνονται στους πίνακες 5.6 και 5.7 αντίστοιχα. Να υπενθυμίσουμε εδώ ότι και τα συγκεκριμένα νούμερα, όσον αφορά το *throughput*, προέκυψαν μετά τη προσέγγιση ότι πρόκειται να κρυπτογραφήσουμε δεδομένα της τάξης των 10.24 KB. Όπως δηλαδή είχε γίνει και στα αποτελέσματα από την προτεινόμενη αρχιτεκτονική

	Cycles	F_{max} (MHz)	Throughput (Mbps)	Area (Slices)	Throughput/Area (Mbps/Slices)
Arc1_2cores	4080	140,351	2818.03	4612	0.611
Arc1_4cores	2160	126,984	4815.98	9199	0.524

Πίνακας 5.6 Αποτελέσματα Virtex5

	Cycles	F_{max} (MHz)	Throughput (Mbps)	Area (Slices)	Throughput/Area (Mbps/Slices)
Arc1_2cores	4080	106,022	2128.76	4444	0.479
Arc1_4cores	2160	100,473	3810.53	8404	0.453

Πίνακας 5.7 Αποτελέσματα Spartan6

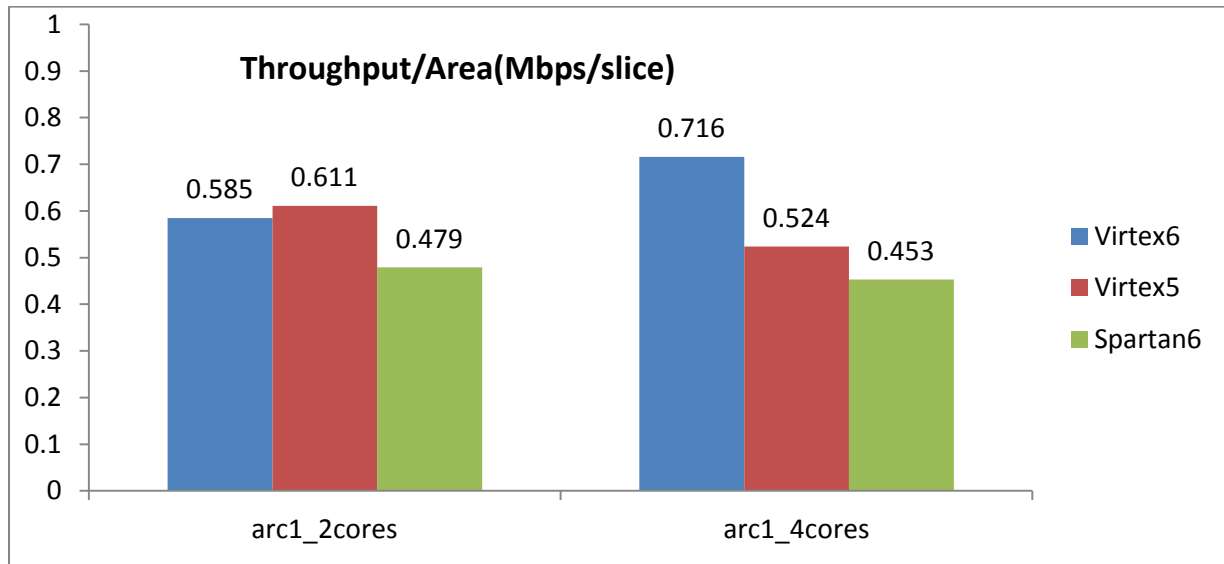


Σχήμα 5.6 Throughput(Mbps)

Στο σχήμα 5.6, παρουσιάζονται τα αποτελέσματα των ρυθμών διαμεταγωγής δεδομένων όπως προέκυψαν από τα FPGAs για τις τυπικές αρχιτεκτονικές των δύο και των τεσσάρων πυρήνων αντίστοιχα. Στην αρχιτεκτονική arc1_2cores το *throughput* μεταβάλλεται από 2,13Gbps έως και 3,17Gbps ανάλογα το FPGA που χρησιμοποιούμε. Στην αρχιτεκτονική arc1_4cores αντίστοιχα αυτό το εύρος τιμών κυμαίνεται από 2,81Gbps έως και 5,93Gbps. Αν δούμε τις περιπτώσεις ξεχωριστά ανά FPGA τότε μεταβαίνοντας από την αρχιτεκτονική δύο πυρήνων σε εκείνη των τεσσάρων πετυχαίνεται αύξηση της τάξης του 78,87% όσον αφορά το FPGA Spartan6. Η αύξηση αυτή διαμορφώνεται στα επίπεδα του 70,92% όσον αφορά το Virtex5 και στα επίπεδα του 87,07% για το Virtex6. Τα αποτελέσματα από το Virtex6, για ακόμη μια φορά δείχνουν ότι προσεγγίζουν καλύτερα τις θεωρητικές τιμές (θεωρητικά αναμενόταν αύξηση της τάξης του 90~95%). Τα αποτελέσματα σαφώς και είναι ικανοποιητικά αλλά το ζητούμενο δεν είναι αυτό. Αυτό που μας ενδιαφέρει είναι ποιος υλοποιημένος χρονοπρογραμματισμός αποδίδει καλύτερα. Στη συνέχεια θα γίνει αναλυτική σύγκριση των δύο χρονοπρογραμματισμών με βάση τα πειραματικά αποτελέσματα.

Ένα ακόμη ενδιαφέρον γράφημα είναι και εκείνο του σχήματος 5.7 όπου απεικονίζονται οι μέσοι ρυθμοί διαμεταγωγής δεδομένων ανά μονάδα επιφάνειας και περιλαμβάνει συνολικά όλα τα αποτελέσματα των FPGAs που χρησιμοποιήθηκαν τόσο για την arc1_2cores όσο και την arc1_4cores. Αυτό που κάνει τη μεγαλύτερη εντύπωση είναι ότι στην αρχιτεκτονική arc1_2cores, ο μέσος ρυθμός διαμεταγωγής δεδομένων ανά επιφάνεια, όσο αφορά την οικογένεια Virtex5 είναι της τάξης του 0,611Mbps ενώ εκείνος του Virtex6, για την ίδια αρχιτεκτονική, φτάνει μόλις τα 0,585Mbps. Όπως είχαμε αναφέρει και νωρίτερα, το device που επιλέχθηκε προσπαθεί πρώτα να κάνει και βελτιστοποίηση στην επιφάνεια και στη προκείμενη περίπτωση φαίνεται ότι το πέτυχε. Αντίθετα, στη arc1_4cores

κάτι τέτοιο δεν έχει συμβεί καθώς το Virtex6 έχει πετύχει ένα άκρως ικανοποιητικό *throughput/area* της τάξεως του 0,716Mbps.



Σχήμα 5.7 Throughput/Area(Mbps/Slice)

Στη συνέχεια αυτού του κεφαλαίου, προβαίνουμε σε μια αναλυτική σύγκριση των δύο χρονοπρογραμματισμών, με βάση τα πειραματικά αποτελέσματα, που έχουν υλοποιηθεί σε διάφορες αρχιτεκτονικές και έχουν μετρηθεί στα FPGAs Virtex5, Virtex6 και Spartan6 αντίστοιχα.

Στο σχήμα 5.8 μπορούμε να παρατηρήσουμε συνολικά τους ρυθμούς διαμεταγωγής δεδομένων και των δύο τύπων αρχιτεκτονικών. Με πράσινο χρώμα συμβολίζονται οι αρχιτεκτονικές που ακολουθούν το υπάρχον χρονοπρογραμματισμό ενώ με κόκκινο χρώμα συμβολίζονται εκείνες που ακολουθούν τον προτεινόμενο. Η οικογένεια που έχει επιλεγεί να γίνει η σύγκριση είναι η Virtex6 διότι, με βάση και όλους τους πίνακες που έχουμε αναφέρει, αποτελεί την πιο αξιόπιστη πλατφόρμα από αυτές που χρησιμοποιήθηκαν.

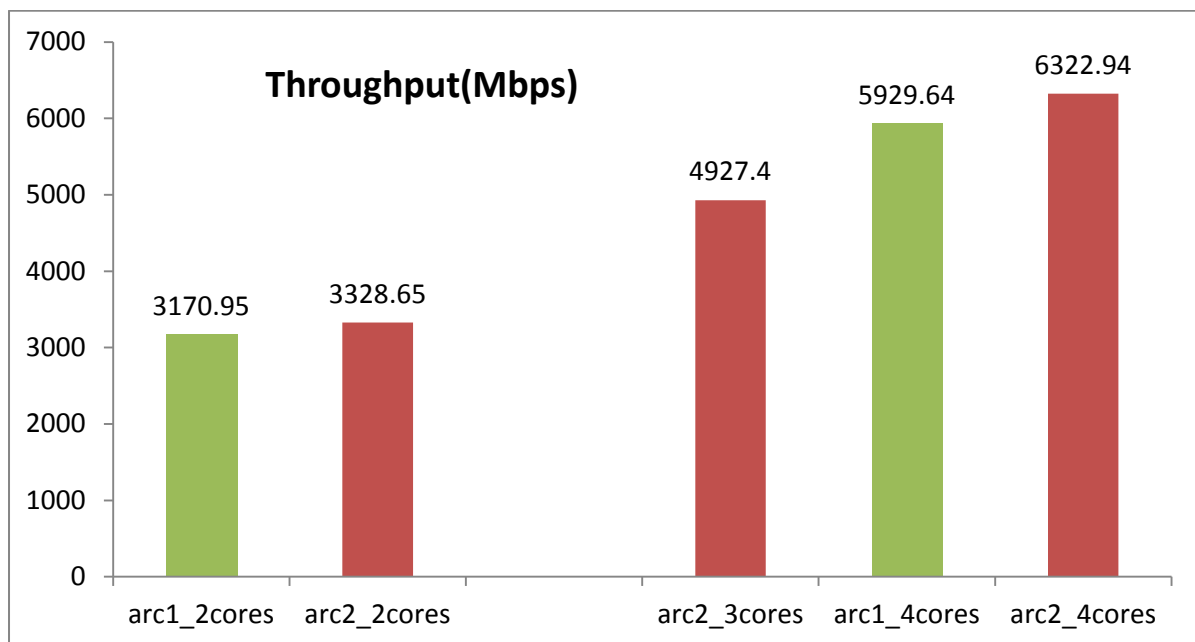
Παρατηρώντας το σχήμα 5.8, το πρώτο πράγμα που παρατηρούμε είναι ότι με βάση τον τυπικό προγραμματισμό δεν έχει υλοποιηθεί αρχιτεκτονική τριών πυρήνων. Το αποτέλεσμα αυτό μπορεί να εξηγηθεί από το γεγονός της προσέγγισης που έχει υιοθετήσει ο τυπικός χρονοπρογραμματισμός. Η προσέγγιση αυτή, ουσιαστικά περιορίζει τους αριθμούς των πυρήνων που μπορούν να ενσωματώσουν οι αρχιτεκτονικές οι οποίες βασίζονται στο συγκεκριμένο χρονοπρογραμματισμό. Ο αριθμός των πυρήνων που μπορούν να διαθέτουν οι αρχιτεκτονικές αυτές περιορίζεται σε 2^n , όπου το n κυμαίνεται από το ένα έως το πέντε.

Αυτό, σημαίνει ότι αν κάποιος ενδιαφέρεται για τη συγκεκριμένη εφαρμογή και επιθυμεί τα όρια του *throughput* να κυμαίνονται γύρω στα 5Gbps, τότε αναγκαστικά θα επιλέξει η δική μας προσέγγιση καθώς η επιλογή μιας αρχιτεκτονικής τεσσάρων πυρήνων θα του προσθέσει επιπλέον επιφάνεια και κόστος.

Όσον αφορά τις δύο αρχιτεκτονικές δύο πυρήνων που εμφανίζονται στο σχήμα 5.8, αυτό που φαίνεται ξεκάθαρα είναι ότι η δική μας προσέγγιση πιάνει ρυθμό διαμεταγωγής

δεδομένων της τάξης του 3,33Gbps ενώ εκείνη που ακολουθεί το τυπικό χρονοπρογραμματισμό φτάνει στο επίπεδο του 3,17Gbps. Η δικιά μας προσέγγιση λοιπόν κερδίζει, έστω και ελαφρά, την υπάρχουσα. Προφανώς η αύξηση αυτή οφείλεται κυρίως, καθώς η μέγιστη συχνότητα τους δεν διαφέρει σημαντικά μεταξύ τους, στο γεγονός ότι η δική μας προσέγγιση ολοκληρώνει την απαιτούμενη διαδικασία σε λιγότερους κύκλους ρολογιού. Θυμίζουμε ότι ο ρυθμός διαμεταγωγής δεδομένων και οι απαιτούμενοι κύκλοι ολοκλήρωσης είναι έννοιες αντιστρόφως ανάλογες. Τη παραπάνω αύξηση που πετύχαμε στο *throughput* μπορούμε να την εκφράσουμε σε ποσοστά για να διαπιστώσουμε το πραγματικό της μέγεθος. Η αύξηση αυτή λοιπόν είναι της τάξης του 4,8% και αποτελεί ένα ικανοποιητικό αποτέλεσμα που μας δείχνει ως πρώτο σημάδι ότι η σκέψη και η ιδέα μας να προτείνουμε ένα καινούργιο χρονοπρογραμματισμό για τη βελτίωση του *throughput* αποδεικνύεται σωστή.

Κοιτάζοντας τις αντίστοιχες αρχιτεκτονικές των τεσσάρων πυρήνων, παρατηρούμε ότι και σε αυτή την περίπτωση η δική μας προσέγγιση πετυχαίνει καλύτερο αποτέλεσμα από την υπάρχουσα. Το σημαντικό είναι όμως ότι διαφέρουν σημαντικά καθώς ο ρυθμός διαμεταγωγής δεδομένων της δική μας προσέγγισης είναι της τάξης του 6,32Gbps ενώ αντίθετα ο ρυθμός αυτός της τυπικής προσέγγισης ανέρχεται στα 5,93Gbps. Η αύξηση αυτή μπορεί να γίνει πιο εύκολα αντιληπτή αν αναχθεί σε ποσοστό επί της εκατό. Η αύξηση που πετυχαίνεται σε αυτή τη περίπτωση είναι της τάξης του 6,6% και αποτελεί ένα άκρως ικανοποιητικό αποτέλεσμα. Τα αποτελέσματα που ελήφθησαν, τόσο εκείνα των αρχιτεκτονικών δύο πυρήνων όσο και εκείνα των τεσσάρων, μας δείχνουν ότι η προσέγγιση μας έχει πραγματικό αντίκτυπο στους ρυθμούς διαμεταγωγής δεδομένων καθώς κρίνονται ως άκρως ικανοποιητικοί.

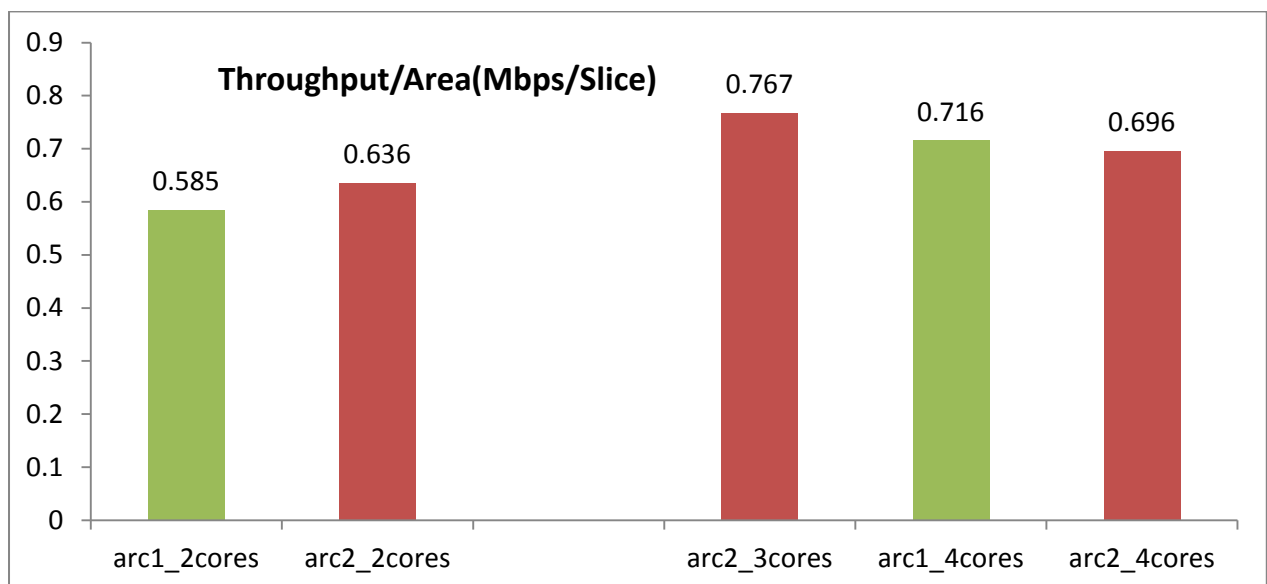


Σχήμα 5.8 Throughput(Mbps), Virtex6

Ένα ακόμη πολύ σημαντικό γράφημα είναι εκείνο του σχήματος 5.9, το οποίο δείχνει τους μέσους ρυθμούς διαμεταγωγής δεδομένων ανά μονάδα επιφάνειας (*throughput/area*, *Mbps/Slice*). Αυτό που προκαλεί τη περισσότερη εντύπωση είναι ότι η αρχιτεκτονική τριών πυρήνων επιτυγχάνει το καλύτερο λόγο *throughput/area*. Αυτό συμβαίνει κυρίως επειδή η συγκεκριμένη αρχιτεκτονική, με βάση το χρονοπρογραμματισμό της, μπορεί να λειτουργεί αποτελεσματικά απαιτώντας κάθε φορά να γνωρίζει μόνο μια *tweak value*.

Παρατηρώντας τις αρχιτεκτονικές δύο πυρήνων παρατηρούμε ότι η δική μας προσέγγιση μπορεί να αποδώσει καλύτερο *throughput/area* συντελεστή. Ο λόγος *throughput/area*, στην δική μας περίπτωση, ανέρχεται στα 0,636Mbps ενώ για την αρχιτεκτονική *arc1_2cores* το αντίστοιχο νούμερο είναι της τάξης του 0,585Mbps. Η αύξηση αυτή που πετύχαμε μπορεί να εκφραστεί καλύτερα αν την εκφράσουμε επί της εκατό. Το ποσοστό της αύξησης ανέρχεται στα επίπεδα του 8,72% και μπορεί να χαρακτηριστεί ως εξαιρετικό. Το περίεργο είναι ότι στη δική μας περίπτωση απαιτείται περισσότερο υλικό διότι χειριζόμαστε ταυτοχρόνως περισσότερες *tweak values*. Προφανώς, στην αρχιτεκτονική *arc2_2cores* το λογισμικό που χρησιμοποιήθηκε κατέφερε να κάνει καλύτερη αξιοποίηση των δεσμευμένων *slices* από ότι έκανε για την αρχιτεκτονική *arc1_2cores*.

Από την άλλη μεριά, στη περίπτωση των τεσσάρων πυρήνων η αρχιτεκτονική *arc1_2cores*, που βασίζεται στον τυπικό χρονοπρογραμματισμό, επιτυγχάνει ένα πολύ καλό *throughput/area* συντελεστή ο οποίος ανέρχεται στα επίπεδα του 0,716Mbps. Η δική μας προσέγγιση υπολείπεται κατά μόλις 0,02Mbps το οποίο συνεπάγεται ότι η δική μας προσέγγιση ανέρχεται στα 0,696Mbps το οποίο επίσης θεωρείται άκρως ικανοποιητικό. Η πολύ μικρή αυτή διαφορά, που είναι της τάξεως του 2,87%, μπορεί να δικαιολογηθεί από το γεγονός ότι η αρχιτεκτονική *arc2_4cores* διαχειρίζεται ταυτοχρόνως έως και πέντε *tweak values*, ούτως ώστε πετύχει καλύτερο *throughput*, το οποίο συνεπάγεται τη χρήση επιπλέον υλικού (δηλαδή επιφάνειας).



Σχήμα 5.9 Throughput/Area(Mbps/Slice), Virtex6

Τέλος, παραθέτουμε έναν πίνακα αποτελεσμάτων των οποίων οι τιμές έχουν καταγραφεί από το *synthesis report*, και όχι από το *post place & route report*, καθώς κάποιες υλοποιήσεις παραθέτουν αυτές τις τιμές. Όπως παρατηρούμε, τα αποτελέσματα αυτά είναι πολύ υψηλότερα από τα αντίστοιχα του *place & route report*, και δεν αποτελούν αντιπροσωπευτικό δείγμα των ικανοτήτων του εκάστοτε κυκλώματος. Στο πίνακα 5.8 φαίνονται οι μετρήσεις F_{max} και *throughput* αντίστοιχα

	Arc1_2cores	Arc1_4cores	Arc2_2cores	Arc2_3cores	Arc2_4cores
F_{max} (MHz)	203,503	203,503	202,782	195,404	193,224
Throughput (Mbps)	2160	100,473	3810.53	8404	0.453

Πίνακας 5.8 Αποτελέσματα σύνθεσης, Virtex6

Κεφάλαιο 6

Συμπεράσματα-Προτάσεις

Κοιτώντας συνολικά όλα τα αποτελέσματα που αναφέρθηκαν σε κεφάλαιο 5, καταλήγουμε στο συμπέρασμα ότι ο προτεινόμενος χρονοπρογραμματισμός μπορεί να έχει εφαρμογή επάνω στο συγκεκριμένο πρότυπο. Τα αποτελέσματα που επιτυγχάνονται θεωρούνται άκρως ικανοποιητικά καθώς τα επίπεδα του throughput κυμαίνονται, όσον αφορά τη πλατφόρμα Virtex6, στα επίπεδα του 6.3Gbps, 4.9Gbps, 3.3Gbps για τις αρχιτεκτονικές δύο, τριών και τεσσάρων πυρήνων αντίστοιχα. Όσον αφορά τη βελτίωση που επιτυγχάνει ο προτεινόμενος χρονοπρογραμματισμός, σε σύγκριση με τον υπάρχον, αυτή κυμαίνεται στα επίπεδα του 4,8%, όσον αφορά την αρχιτεκτονική δύο πυρήνων, και 6,6% όσον αφορά την αρχιτεκτονική τεσσάρων πυρήνων για τη πλατφόρμα Virtex6. Για τις υπόλοιπες πλατφόρμες (Virtex5, Spartan6), οι βελτιώσεις που επιτυγχάνονται κυμαίνονται στα ίδια επίπεδα.

Η αρχιτεκτονική τριών πυρήνων, η οποία δεν έχει μέτρο σύγκρισης, αποτελεί μια εξαιρετική επιλογή από μόνη της καθώς επιτυγχάνει να έχει ρυθμούς διαμεταγωγής δεδομένων της τάξης του 4,93Gbps και ο παράγοντας *throughput/area* να αγγίζει το εξαιρετικό 0,716Mbps. Ο συνδυασμός των δύο παραπάνω αποτελεσμάτων καθιστούν την αρχιτεκτονική τριών πυρήνων ως μιας άκρως ελκυστική επιλογή υλοποίησης του προτύπου P1619.

Τέλος, όσον αφορά τις επιπλέον βελτιώσεις που μπορεί να επιφέρει ο σχεδιαστής πάνω στις συγκεκριμένες αρχιτεκτονικές ούτως ώστε να αυξήσει το *throughput* είναι

- Επιλογή διαφορετικού AES (π.χ. *pipeline AES*). Ο AES που επιλέχθηκε δεν επικεντρώνεται στη βελτιστοποίηση του *throughput*, αλλά της επιφάνειας καθώς χρειάζεται αρκετούς γύρους για να ολοκληρώσει τη διαδικασία κρυπτογράφησης/αποκρυπτογράφησης. Με έναν αυτόν τον τρόπο μπορούμε να επιτύχουμε μια πολύ καλή αύξηση του *throughput*
- Με χρήση μια διαφορετικής εισόδου/εξόδου δεδομένων. Θυμίζουμε ότι εμείς επιλέξαμε ένα μόνο 64-bit bus για να φορτώνονται όλα τα δεδομένα(*keys, i values, ciphertext/plaintext*). Η αλλαγή αυτή μπορεί να επιφέρει βελτίωση μόνο του χρόνου αρχικοποίησης(*latency*).
- Επιλογή μιας διαφορετικής προσέγγισης στην αρχιτεκτονική, η οποία εκμεταλλεύεται με ένα καλύτερο τρόπο τους διαθέσιμους πόρους.
- Χρήση επιπλέον πυρήνων AES. Με αυτή τη λογική επιτυγχάνεται σημαντική αύξηση του ρυθμού διαμεταγωγής δεδομένων, αλλά ταυτόχρονα αυξάνεται σημαντικά και η επιφάνεια του κυκλώματος.

Βιβλιογραφία

- [1] “Κρυπτογραφία.” [Online]. Available:
<https://el.wikipedia.org/wiki/%CE%9A%CF%81%CF%85%CF%80%CF%84%CE%BF%CE%B3%CF%81%CE%B1%CF%86%CE%AF%CE%B1>.
- [2] Θ. Κομηνέας, *Σχεδίαση και υλοποίηση σε FPGA , αχιτεκτονικών πολλαπλών λειτουργιών χαμηλής επιφάνειας ολοκλήρωσης, για κρυπτογραφικές συναρτήσεις κατακερματισμού*, Πάτρα, 2012.
- [3] Hughes, J., & Cole, J. (2003). Security in storage [standards]. *Computer*, 36(1), 124–125.
<http://doi.org/10.1109/MC.2003.1160064>
- [4] “Αλγόριθμοι και Κλετδιά.” [Online]. Available:
<http://www.image.ntua.gr/meleti172KTP/?q=node/23>.
- [5] H. Feistel, “Data Encryption Standard (Des),” *Fips Pub 46-3*, vol. 3, 1999.
- [6] I. G. Torrego, “Study of the IEEE Standard 1619 . 1 : Authenticated Encryption with Length Expansion for Storage Devices,” no. June, 2009.
- [7] “Disk Encryption.”[Online]. Available:
https://en.wikipedia.org/wiki/Disk_encryption
- [8] “SISWG-Security in Storage Working Group.” [Online]. Available:
<http://siswg.net/>.
- [9] “Filesystem-level encryption.” [Online]. Available:
https://en.wikipedia.org/wiki/Filesystem-level_encryption.
- [10] “The Benefits of AES Hardware Encryption for Secure USB Flash Drives, Hard Drives and Solid State Drives.” [Online]. Available:
<https://www.kanguru.com/info/secure-hardware-encryption-for-usb-flash-drives.shtml>.
- [11] J. Hughes, “IEEE Standard for Encrypted Storage,” *IEEE Comput.*, vol. 37, pp. 110–112, 2004.

- [12] IEEE, “Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices,” no. May, p. 38, 2007.
- [13] NIST Fips-197,; Announcing the advanced encryption standard (AES),” 2001.
- [14] Ε. Χατζηδημητρίου, « Διερεύνηση του προτύπου P1619 για διαμοιραζόμενα αποθηκευτικά μέσα και πρότυπες προτάσεις υλοποίησης .», Πάτρα, 2013.
- [15] Α. Ρ. Kakarountas, Ε. Hatzidimitriou, and Α. Milidonis, “A survey on throughput-efficient architectures for IEEE P1619 for shared storage media,” in *2011 IEEE Symposium on Computers and Communications (ISCC)*, 2011, pp. 758–763.
- [16] V. A. Pedroni, *Finite State Machines in Hardware*. The MIT press, 2013.
- [17] V. Pedroni, *Circuit Design with VHDL*. 2004.
- [18] “Xilinx.” [Online]. Available: <http://www.xilinx.com/>.
- [19] “Mentor Graphics.” [Online]. Available: <https://www.mentor.com/>.
- [20] “FPGA.” [Online]. Available: https://en.wikipedia.org/wiki/Field-programmable_gate_array.