ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΣΥΣΤΗΜΑΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Δ Ι Π Λ Ω Μ Α Τ Ι Κ Η Ε Ρ Γ Α Σ Ι Α

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών της Πολυτεχνικής Σχολής του Πανεπιστημίου Πατρών

Γιαννάτου Σπυρίδωνος

Αριθμός Μητρώου: 1058457

ΘΕΜΑ

Σχεδίαση Υποσυστήματος Κρυπτογραφίας σε Υλικό

ΕΠΙΒΛΕΠΩΝ

Σερπάνος Δημήτριος

Καθηγητής

Αριθμός Διπλωματικής Εργασίας:

…….

ΠΑΤΡΑ – ΙΟΥΛΙΟΣ 2023

Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών.

Σπυρίδων Γιαννάτος

© 2023 – Με την επιφύλαξη παντός δικαιώματος

Το σύνολο της εργασίας αποτελεί πρωτότυπο έργο, παραχθέν από τον Σπυρίδωνα Γιαννάτο, και δεν παραβιάζει δικαιώματα τρίτων καθ' οιονδήποτε τρόπο. Αν η εργασία περιέχει υλικό, το οποίο δεν έχει παραχθεί από τον/την ίδιο/α, αυτό είναι ευδιάκριτο και αναφέρεται ρητώς εντός του κειμένου της εργασίας ως προϊόν εργασίας τρίτου, σημειώνοντας με παρομοίως σαφή τρόπο τα στοιχεία ταυτοποίησής του, ενώ παράλληλα βεβαιώνει πως στην περίπτωση χρήσης αυτούσιων γραφικών αναπαραστάσεων, εικόνων, γραφημάτων κ.λπ., έχει λάβει τη χωρίς περιορισμούς άδεια του κατόχου των πνευματικών δικαιωμάτων για την συμπερίληψη και επακόλουθη δημοσίευση του υλικού αυτού.

ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η Διπλωματική Εργασία με τίτλο

Σχεδίαση Υποσυστήματος Κρυπτογραφίας σε Υλικό

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

ΣΠΥΡΙΔΩΝΟΣ ΓΙΑΝΝΑΤΟΥ ΤΟΥ **ΓΕΩΡΓΙΟΥ**

Αριθμός Μητρώου: 1058457

Παρουσιάστηκε δημόσια στο Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών στις

…03.../…07…/…2023…

και εξετάστηκε από την ακόλουθη εξεταστική επιτροπή:

Σερπάνος Δημήτριος, Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών (επιβλέπων)

Θεοδωρίδης Γεώργιος, Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών (μέλος επιτροπής)

Μπίρμπας Μιχαήλ, Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών (μέλος επιτροπής)

Ο Επιβλέπων
Σερπάνος Δημήτριος
Καθηγητής

Ο Διευθυντής του Τομέα
Καλύβας Γρηγόριος
Καθηγητής

ΠΕΡΙΛΗΨΗ

Σχεδίαση Υποσυστήματος Κρυπτογραφίας σε Υλικό

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΦΟΙΤΗΤΗ:
ΣΠΥΙΔΩΝ ΓΙΑΝΝΑΤΟΣ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΕΠΙΒΛΕΠΟΝΤΟΣ:
ΔΗΜΗΤΡΙΟΣ ΣΕΡΠΑΝΟΣ

Στη σύγχρονη εποχή η ανταλλαγή και η διακίνηση πληροφορίας αποτελούν αναπόσπαστο τμήμα της ανθρώπινης καθημερινότητας. Η διαρκώς αυξανόμενη ανάγκη για ανταλλαγή πληροφορίας, εκτός από την ραγδαία ανάπτυξη των δικτύων των υπολογιστών και του απαιτούμενου υλικού για την επίτευξη αυτής της ανταλλαγής, έχει φέρει στο προσκήνιο και το σοβαρό ζήτημα της προστασίας της πληροφορίας. Για την προστασία τόσο της μεταδιδόμενης όσο και της λαμβανόμενης πληροφορίας η επιστήμη της Κρυπτογραφίας κατορθώνει, μέσω της ανάπτυξης αλγορίθμων κρυπτογράφησης και αποκρυπτογράφησης, να διασφαλίσει την αποτελεσματική διάδοση της πληροφορίας μεταξύ αποστολέα – παραλήπτη χωρίς να μπορεί να αντιγραφεί, να διαγραφεί ή και να αλλοιωθεί από κάποιον τρίτο.

Στόχος της παρούσης διπλωματικής εργασίας είναι η ανάπτυξη ενός τέτοιου αλγορίθμου κρυπτογραφίας, συγκεκριμένα του προτύπου Advanced Encryption Standard (AES), με Γλώσσα Περιγραφής Υλικού (VHDL) αλλά και με Σύνθεση Υψηλού Επιπέδου (Vivado HLS, Xilinx) και εν συνεχεία η βελτιστοποίηση τμημάτων του αλγορίθμου ούτως ώστε να γίνει μια εμπεριστατωμένη μελέτη και εξαγωγή των κατάλληλων συμπερασμάτων μεταξύ των δυο υλοποιήσεων. Η τελική υλοποίηση που θα χαρακτηρίζεται για την βέλτιστη δυνατή απόδοση θα κριθεί από την μείωση της καθυστέρησης κατά την υλοποίηση του αλγορίθμου αλλά και από τη χρήση των διαθέσιμων πόρων που απαιτούνται.

ABSTRACT

Hardware Design of Cryptographic Subsystem

STUDENT NAME, SURNAME:
SPYRIDON GIANNATOS

SUPERVISOR NAME, SURNAME:
DIMITRIOS SERPANOS

In modern times, the exchange and circulation of information is an integral part of everyday human life. The ever-increasing need to exchange information, besides the rapid development of computer networks and the hardware required to achieve this exchange, has brought to the fore the serious issue of security of information. In order to protect both the information transmitted and the information received, the science of cryptography has succeeded, through the development of encryption and decryption algorithms, to ensure the effective dissemination of information between sender and receiver without it being copied, deleted or even In modern times, the exchange and circulation of information is an integral part of everyday human life. The ever-increasing need to exchange information, besides the rapid development of computer networks and the hardware required to achieve this exchange, has brought to the fore the serious issue of security of information. In order to protect both the information transmitted and the information received, the science of cryptography has succeeded, through the development of encryption and decryption algorithms, to ensure the effective dissemination of information between sender and receiver without it being copied, deleted or even altered by a third party.

The purpose of this thesis is to develop such a cryptography algorithm, particularly the Advanced Encryption Standard (AES), using both a Hardware Description Language (VHDL) and High Level Synthesis (Vivado HLS, Xilinx) and subsequently to optimize parts of the algorithm in order to make a comprehensive study and draw appropriate conclusions between the two implementations. The final implementation that will be characterized for the optimal performance will be considered by the reduction of the latency during the implementation of the algorithm as well as by the use of the available resources required.

Ευχαριστίες

Με την ολοκλήρωση της παρούσας Διπλωματικής Εργασίας θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Σερπάνο Δημήτριο, για την επίβλεψη, την καθοδήγηση, την υποστήριξη και την παραγωγική συνεργασία μας καθ' όλη τη διάρκεια εκπόνησης της παρούσας εργασίας.

Θα ήθελα επίσης να ευχαριστήσω τον Αναπλ. Καθηγητη κ. Θεοδωρίδη Γεώργιο για τις παρατηρήσεις και τις συμβουλές που μου παρείχε, οι οποίες συνετέλεσαν στη ολοκλήρωση της Διπλωματικής Εργασίας.

Τέλος, θέλω να ευχαριστήσω τους δικούς μου ανθρώπους για την αμέριστη συμπαράσταση και τη στήριξη που μου παρείχαν σε όλη την διάρκεια της ακαδημαϊκής μου πορείας. Τον πατέρα μου Γιώργο και τη μητέρα μου Αθηνά, η οποία αν και δεν βρίσκεται πλέον κοντά μου είμαι σίγουρος πόσο περήφανη θα ένιωθε για την πρόοδο μου, τον αδερφό μου Νάσο και τη σύντροφο μου Αναστασία.

Στους γονείς μου,

Γιώργο και Αθηνά

Περιεχόμενα

Κεφάλαιο 1: Αντικείμενο και στόχοι της εργασίας 12

Κεφάλαιο 2: Advanced Encryption Standard (AES) 14

2.1 Εισαγωγή 14

2.2 Μαθηματικό Υπόβαθρο 15

2.3 Δομή AES Encryptor 16

2.3.1 SubBytes() Μετασχηματισμός 18

2.3.2 ShiftRows() Μετασχηματισμός 19

2.3.3 MixColumns() Μετασχηματισμός 20

2.3.4 AddRoundKey() Μετασχηματισμός 21

2.3.5 Επέκταση κλειδιού (Key Expansion) 21

2.4 Δομή AES Decryptor 22

2.4.1 InverseShiftRows() Μετασχηματισμός 23

2.4.2 InverseSubBytes() Μετασχηματισμός 23

2.4.3 Inverse AddRoundKey() Μετασχηματισμός 23

2.4.4 InverseMixColumns() Μετασχηματισμός 24

2.5 Παράδειγμα υλοποίησης του AES-128 Encryptor 24

Κεφάλαιο 3: Σύνθεση Υψηλού Επιπέδου (High-Level Synthesis) 28

3.1 Εισαγωγή 28

3.2 Vivado – High Level Synthesis 29

3.3 Δομή λειτουργίας του Vivado - HLS 30

3.3.1 Διάγραμμα Λειτουργίας Vivado - HLS 30

3.3.2 Βασικές Μετρικές 31

3.3.3 Αριθμητική Κινητής Υποδιαστολής 32

3.3.4 Περιορισμοί 33

3.3.5 Directives 33

Κεφάλαιο 4: Υλοποίηση του AES-128 σε Υλικό και Εφαρμογή Βελτιστοποιήσεων 40

4.1 Υλοποίηση του AES-128 σε VHDL 40

4.2 Υλοποίηση του AES-128 σε Vivado-HLS 42

4.3 Εφαρμογή Βελτιστοποιήσεων 45

4.3.1 Βελτιστοποίηση της συνάρτησης SubBytes 45

4.3.2 Βελτιστοποίηση της συνάρτησης AddRoundKey 55

4.3.3 Βελτιστοποίηση των συναρτήσεων MixColumns & ShiftRows 62

4.3.4 Βελτιστοποίηση του AES-128 ECB (electronic codebook) 69

4.4 Εφαρμογή Συνδυασμών Βελτιστοποιήσεων 83

Κεφάλαιο 5: Συμπεράσματα 87

Κεφάλαιο 1

Αντικείμενο και στόχοι της εργασίας

Στη σύγχρονη εποχή, η διαρκής και αυξανομένη ανάγκη για μεταφορά και ανταλλαγή πληροφορίας σε συνδυασμό με την ανάπτυξη των δικτύων των υπολογιστών και των εφαρμογών σε επίπεδο υλικού (Hardware) αυξάνουν την ανάγκη για προστασία της πληροφορίας που ανταλλάσσεται. Ο ρόλος, λοιπόν, της Κρυπτογραφίας είναι καθοριστικής σημασίας για τη διαφύλαξη και την ασφαλή μετάδοση της πληροφοριας. Αυτό επιτυγχανεται μεσα από την αναπτυξη αλγοριθμων Κρυπτογραφησης και Αποκρυπτογράφησης ώστε η επικοινωνια μεταξυ αποστολεα και παραλληπτη να γινεται απροσκοπτα, χωρις τον κινδυνο αντιγραφης, διαγραφης ή αλλοιωσης από καποιον τριτο. Η αποτελεσματικοτητα συστηματων Κρυπτογραφιας ωστοσο δεν εξαρταται μονο από τη δομη των αλγοριθμων αλλα και από τα ενσωματωμενα συστηματα στα οποια αναπτυσσονται. Τα ενσωματωμένα συστήματα αποτελούν κοινή λύση στους κρυπτογραφικούς αλγορίθμους για χρήσεις δικτυακής επικοινωνίας. Μπορούν να παρέχουν υψηλό εύρος ζώνης (bandwidth), μεγιστοποιώντας την απόδοση (throughput) και την αξιοπιστία. Η αυξανόμενη τάση στα ενσωματωμένα συστήματα είναι τα FPGAs. Τα Field Programmable Gate Arrays (FPGAs) είναι συσκευές ημιαγωγών που βασίζονται σε ένα πλέγμα διαμορφώσιμων λογικών μπλοκ (CLBs) που συνδέονται μέσω προγραμματιζόμενων διασυνδέσεων. Τα FPGA είναι επαναδιαμορφώσιμες συσκευές που μπορούν να υλοποιήσουν οποιαδήποτε επιθυμητή εφαρμογή ή λειτουργικές απαιτήσεις μετά την κατασκευή. Αυτό το χαρακτηριστικό διακρίνει τα FPGA από τα ολοκληρωμένα κυκλώματα ειδικών εφαρμογών (ASIC), τα οποία κατασκευάζονται κατά παραγγελία για συγκεκριμένες εργασίες σχεδιασμού. Αν και τα FPGA είναι κατώτερα των ASIC όσον αφορά την επεξεργαστική ισχύ, το χαμηλό κόστος και η ευελιξία σχεδιασμού καθιστούν τα FPGA κατάλληλη λύση για ορισμένες εφαρμογές όπως οι δικτυακές επικοινωνίες. Ο ρολος, λοιπον, των FPGAs είναι ιδιαιτέρας σημασίας για την ανάπτυξη των ολοκληρωμένων και ενσωματωμένων συστημάτων, λόγω της δυνατότητάς τους να υλοποιούν την ανάπτυξη του λογισμικού συστήματος ταυτόχρονα με το υλικό, παρεχοντας ετσι τη δυνατότητα προσομοίωσης της απόδοσης του συστήματος σε ενα πρώιμο στάδιο του σχεδιασμου του και να επιτρέποντας την εφαρμογη διαφορων δοκιμων και τροποποιησεων του υποσυστήματος πριν από τον τελικό σχεδιασμό της αρχιτεκτονικής του συστήματος.

Συνεπως, η ανάγκη για ανάπτυξη σύνθετων εφαρμογών σε επιπεδο FPGAs, συνετελεσε στην ανάπτυξη διάφορων σχεδιαστικων προγραμματων που προσφέρουν στο σχεδιαστή τη δυνατότητα γρήγορης και εύκολης σχεδίασης. Ένα από τα εργαλεία αυτά, στη χρηση του οποιου εστιαζει η παρουσα διπλωματικη εργασια είναι το Vivado HLS, του Xilinx, μεσω του οποίου γίνεται μετατροπη μίας γλώσσας υψηλού επιπέδου σε σχεδιαστικη γλώσσα υλικού. Η χρηση εργαλείων high level synthesis για το σχεδιασμο και την αναπτυξη εφαρμογων σε επιπεδο FPGA, συντελει καθοριστικα στη μειωση του απαιτουμενου χρονου για τη συνθεση του κωδικα και τον σχεδιασμο του συτηματος σε αντιθεση με το σχεδιασμο σε επιπεδο καταχωρητων (RTL) χρησιμοποιωντας γλωσσες περιγραφης υλικου (HDLs), διοτι ο κώδικας λογισμικού χρησιμοποιωντας HLS είναι πιο συνοπτικός, ευκολοτερο να διαβαστεί και να κατανοηθεί για βελτιστοποιησεις - τροποποιησεις σε σύγκριση με τον RTL κώδικα. Η συνοπτικότητα και η αναγνωσιμότητα του λογισμικού μειώνουν τον αριθμο σφάλματων στη σχεδίαση του υποσυστηματος, ενώ η λεπτομερέστερη περιγραφη του κωδικα, εστιαζοντας στη μικροαρχιτεκτονικη του συστηματος, παρεχει μια πιο ολοκληρωμενη και πιο κοντα στα δωθεντα προτυπα σχεδιαση του FPGA.

Η παρούσα διπλωματική εργασία εστιαζει στον σχεδιασμο υποσυστηματος κρυπτογραφιας σε μια διαταξη FPGA, υλοποιωντας εναν κρυπτογραφικό αλγόριθμο συμμετρικού κλειδιού, τον Advanced Encryption Standard (AES), χρησιμοποιώντας VHDL και High Level Synthesis. Στοχος της εργασιας είναι η σύγκριση των δυο υλοποιησεων ως προς την μεγιστη δυνατη αποδοση, δηλαδη στους ελάχιστους κύκλους ρολογιού που απαιτούνται για την ολοκλήρωση της λειτουργίας του συστηματος, και τη μικροτερη δυνατή χρησιμοποιηση των διαθεσιμων πορων του συστηματος και τελος η εξαγωγη συμπερασματων επειτα από την εφαρμογη καταλληλων βελτιστοποιησεων για την επιτευξη της βελτιστης δυνατης σχεδιασης. Έχοντας αυτούς τους στόχους κατά νου θα αναλυθουν οι βασικές λειτουργίες του αλγορίθμου, ενώ στη συνέχεια θα ακολουθήσει ανάλυση της απόδοσης και των περιορισμών του υλικού.

Σε πρώτο στάδιο, ο αλγόριθμος του AES εισήχθη στο εργαλείο του Vivado HLS, επιλέχθηκε πλατφόρμα FPGA της οικογένειας Kintex-UltraScale και αναπτύχθηκε ο καταλληλος κωδικας για την υλοποιηση του αλγοριθμου. Εν συνεχεία, αφου πραγματοποιήθηκε προσομοίωση του κώδικα σε C++, ακολούθησε η σύνθεση του κώδικα. Μέσω των synthesis reports που παράχθηκαν με την ολοκλήρωση της σύνθεσης υψηλου επιπεδου , παρατηρήθηκε αρκετή καθυστέρηση στην υλοποίηση και μεγαλη χρηση των διαθεσιμων πορων του συστηματος ξεκινωντας ετσι μια διαδικασία εφαρμογής βελτιστοποιήσεων ξεχωριστα σε κάθε συναρτηση του αλγοριθμου αλλα και συνδυαστικά για την εξαγωγη καλυτερων αποτελεσματων.

Κεφάλαιο 2

Advanced Encryption Standard (AES)

2.1 Εισαγωγή

Το Advanced Encryption Standard (AES) είναι ένα ευρέως χρησιμοποιούμενο πρότυπο κρυπτογράφησης/αποκρυπτογράφησης που χρησιμοποιεί έναν αλγόριθμο συμμετρικού κλειδιού (που σημαίνει ότι το ίδιο κλειδί χρησιμοποιείται τόσο για κρυπτογράφηση όσο και για αποκρυπτογράφηση), για την ασφαλή κρυπτογράφηση και αποκρυπτογράφηση δεδομένων. Χρησιμοποιείται για την προστασία ευαίσθητων πληροφοριών όπως κωδικούς πρόσβασης, αριθμούς πιστωτικών καρτών και άλλα εμπιστευτικά δεδομένα.

Ο AES είναι ένας αλγόριθμος κρυπτογράφησης μπλοκ (Block Cipher) που κρυπτογραφεί δεδομένα σε μπλοκ σταθερού μεγέθους 128 bit, και μέγεθος κλειδιού 128, 192 ή 256 bit. Χρησιμοποιεί μια διαδικασία επέκτασης κλειδιού για να δημιουργήσει μια σειρά από στρογγυλά κλειδιά (Round Keys) από το αρχικό κλειδί, τα οποία στη συνέχεια χρησιμοποιούνται για την κρυπτογράφηση κάθε μπλοκ δεδομένων σε μια σειρά γύρων. Το μέγεθος του κλειδιού που χρησιμοποιείται για ένα κρυπτογράφημα AES καθορίζει τον αριθμό των γύρων μετασχηματισμού που μετατρέπουν την είσοδο (plaintext), στην τελική έξοδο (ciphertext). Ο αριθμός των γύρων έχει ως εξής:

- 10 γύροι για κλειδιά 128 bits.
- 12 γύροι για κλειδιά 192 bits.
- 14 γύροι για κλειδιά 256-bits.

Κάθε γύρος αποτελείται από διάφορα βήματα επεξεργασίας, συμπεριλαμβανομένου ενός που εξαρτάται από το ίδιο το κλειδί κρυπτογράφησης. Για τη διαδικασία της αποκρυπτογράφησης εφαρμόζεται ένα σύνολο αντίστροφων γύρων για τη μετατροπή του ciphertext πίσω στο αρχικό κείμενο (plaintext) χρησιμοποιώντας το ίδιο κλειδί κρυπτογράφησης. Ο αλγόριθμος AES χρησιμοποιεί έναν συνδυασμό λειτουργιών αντικατάστασης, μετάθεσης και ανάμειξης παρέχοντας έτσι ισχυρή κρυπτογράφηση που είναι ανθεκτική σε επιθέσεις κρυπτανάλυσης, χαρακτηριστικά που τον καθιστούν ένα εξαιρετικά ασφαλές και αποτελεσματικό πρωτόκολλο κρυπτογράφησης.

2.2 Μαθηματικό Υπόβαθρο

Ο AES χρησιμοποιεί ένα πεδίο Galois και συγκεκριμένα το $GF(2^8)$ με μη αναγόμενο πολυώνυμο $x^8 + x^4 + x^3 + x + 1$, για να εκτελέσει πολλές βασικές λειτουργίες.

Το πεδίο Galois $GF(p^n)$, όπου p είναι πρώτος και n είναι θετικός ακέραιος, δηλώνει το πεδίο με p^n στοιχεία. Για παράδειγμα, το πεδίο GF(8) (ή $GF(2^3)$) περιέχει όλους τους ακέραιους αριθμούς από το 0 έως το 7. Μια σημαντική ιδιότητα των πεδίων Galois είναι ότι τα στοιχεία του πεδίου $GF(p^n)$ είναι όλα τα πολυώνυμα βαθμού μικρότερου του n με μη αρνητικούς συντελεστές, που αξιολογούνται στο p. Το p ονομάζεται χαρακτηριστικό του πεδίου.

Πρόσθεση/Αφαίρεση

Για την πρόσθεση δυο στοιχείων a και b του $GF(p^n)$, πρώτα τα μετατρέπουμε στις πολυωνυμικές τους μορφές, δηλαδή τα γράφουμε ως αθροίσματα δυνάμεων του p. Στη συνέχεια προσθέτουμε τα πολυώνυμα μεταξύ τους, αλλά με την εξής ιδιαιτερότητα: για κάθε συντελεστή a_k στο a(p), b_k στο b(p), ο συντελεστής c_k που προκύπτει είναι ίσος με $a_k + b_k \mod p$ (δηλαδή $c_k = a_k + b_k \mod p$). Για την αφαίρεση ισχύει αντίστοιχα ο τύπος $c_k = a_k – b_k \mod p$. Το πεδίο $GF(2^8)$, όπως και όλα τα πεδία με χαρακτηριστικό 2, μοιράζονται την ίδια ιδιότητα που τα καθιστά κατάλληλα για χρήση σε υπολογιστές: η πρόσθεση και η αφαίρεση είναι ισοδύναμες με την πράξη XOR/⊕. Αυτό λειτουργεί επειδή κάθε όρος στο πολυώνυμο αντιπροσωπεύει ένα bit.

Πολλαπλασιασμός

Για να πολλαπλασιάσουμε δύο πολυώνυμα a(p) και b(p) στο $GF(p^n)$, πρέπει να επιλέξουμε ένα τρίτο πολυώνυμο m(p) που δεν μπορεί να παραγοντοποιηθεί και έχει βαθμό τουλάχιστον n. Ονομάζουμε το m(p) μη αναγόμενο πολυώνυμο. Για να πολλαπλασιάσουμε τα a(p) και b(p) στο $GF(p^n)$ εκτελούμε τα ακόλουθα βήματα:

- Πολλαπλασιάζουμε τα a(p) και b(p)
- Ανάγουμε τους συντελεστές του πολυωνύμου που προκύπτει modulo p.
- Ανάγουμε ολόκληρο το πολυώνυμο mod m(p). Αυτό γίνεται ώστε το αποτέλεσμα που θα προκύψει να είναι μικρότερο από p^n.
- Το τελικό βήμα μπορεί να πραγματοποιηθεί με τη χρήση πολυωνυμικής διαίρεσης. Ωστόσο, κάθε αριθμητική πράξη που εκτελούμε κατά τη διάρκεια αυτής της διαδικασίας πρέπει να είναι η εκδοχή του πεπερασμένου πεδίου. Στην περίπτωση του $GF(2^8)$, κάθε φορά που εκτελούμε "αφαίρεση", στην πραγματικότητα εκτελούμε την πράξη XOR. Η αναπαράσταση των πολυωνύμων ως δυαδικές συμβολοσειρές θα είναι χρήσιμη για αυτό το βήμα.

Διαίρεση

Ουσιαστικά στην περίπτωση της διαίρεσης γίνεται ένας πολλαπλασιασμός του αντίστροφου πολυωνύμου. Το πολλαπλασιαστικό αντίστροφο ενός πολυωνύμου a(p) είναι το πολυώνυμο b(p) τέτοιο ώστε $a(p) * b(p) \mod m(p) = 1$. Η διαίρεση είναι απλώς θέμα πολλαπλασιασμού του πρώτου τελεστή με το πολλαπλασιαστικό αντίστροφο του δεύτερου.

2.3 Δομή AES Encryptor

Ο AES βασίζεται σε μια αρχή σχεδιασμού γνωστή ως δίκτυο αντικατάστασης-παρεμβολής και είναι αποδοτικός τόσο σε λογισμικό (software) όσο και σε υλικό (hardware). Ο AES είναι μια παραλλαγή του Rijndael, με σταθερό μέγεθος μπλοκ 128 bits και μέγεθος κλειδιού 128, 192 ή 256 bits. Αντίθετα, ο Rijndael καθαυτός καθορίζεται με μεγέθη μπλοκ και κλειδιών που μπορούν να είναι οποιοδήποτε πολλαπλάσιο των 32 bits, με ελάχιστο τα 128 και μέγιστο τα 256 bits.

Ο AES λειτουργεί σε έναν πίνακα 4 × 4 στήλης μείζονος τάξης 16 bytes (128 bits) b0, b1, ..., b15 που ονομάζεται κατάσταση:

[b0 | b4 | b8 | b12 |

| b1 | b5 | b9 | b13 |

| b2 | b6 | b10| b14 |

| b3 | b7 | b11| b15]

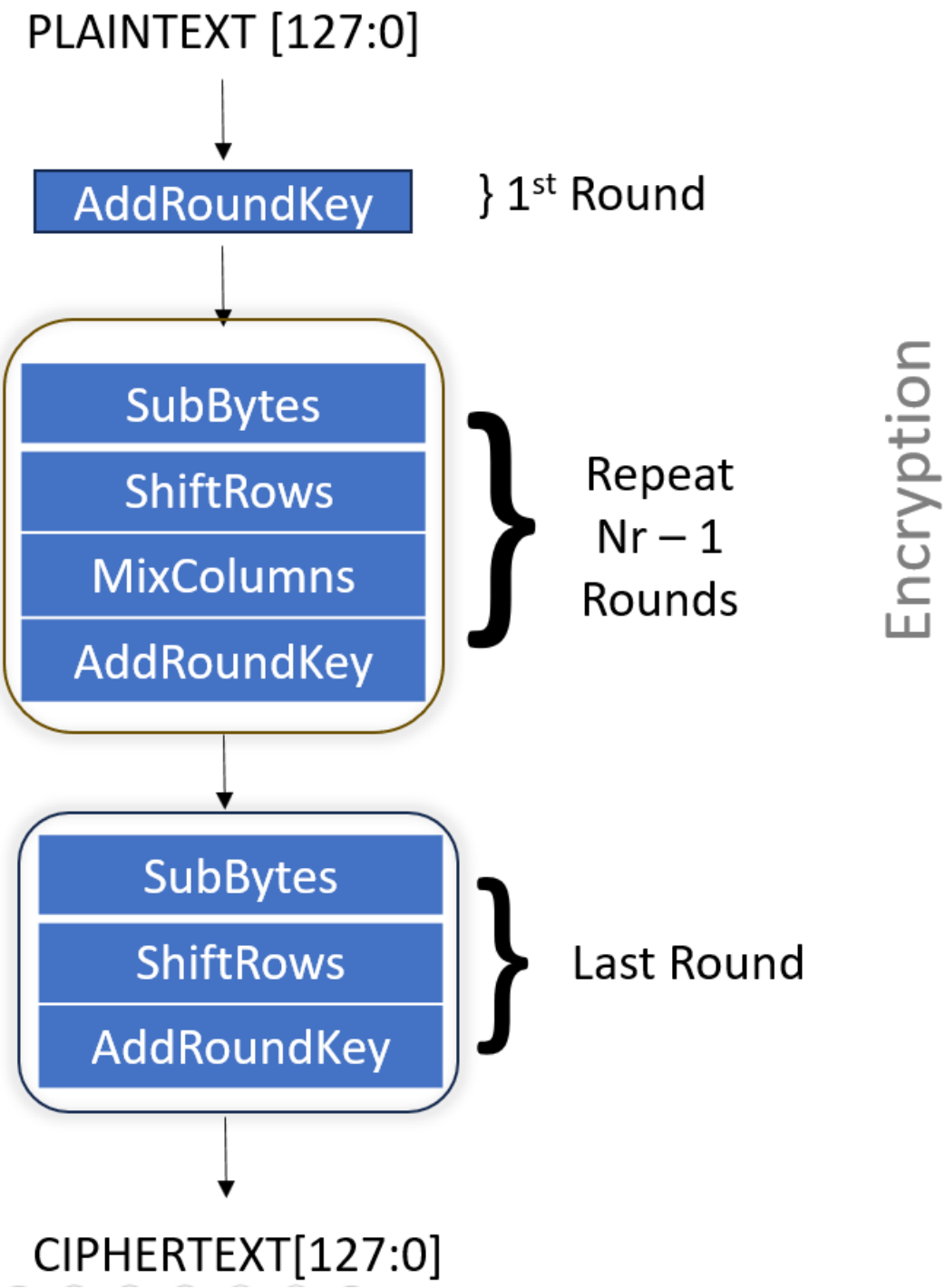
Σε κάθε γύρο εφαρμόζονται αντικαταστάσεις και μεταθέσεις στον πίνακα κατάστασης μέχρι τον τελικό γύρο. Οι υπολογισμοί αυτοί χρησιμοποιούν τους ακόλουθους μετασχηματισμούς:

- SubBytes: Ένας μη γραμμικός μετασχηματισμός όπου κάθε byte αντικαθίσταται με έναν πίνακα υποκατάστασης (S-box).

- ShiftRows: Μια γραμμή μεταφέρεται με κυκλικό μοντέλο για έναν αριθμό βημάτων σε μια νέα θέση.

- MixColumns: Ένας γραμμικός μετασχηματισμός εφαρμόζεται σε κάθε στήλη με βάση τα 4 byte της στήλης.

- AddRoundKey: Κάθε byte του πίνακα κατάστασης συνδυάζεται με ένα byte του κλειδιού γύρου, όπως αυτό έχει διαμορφωθεί από τον αλγόριθμο επέκτασης κλειδιού.



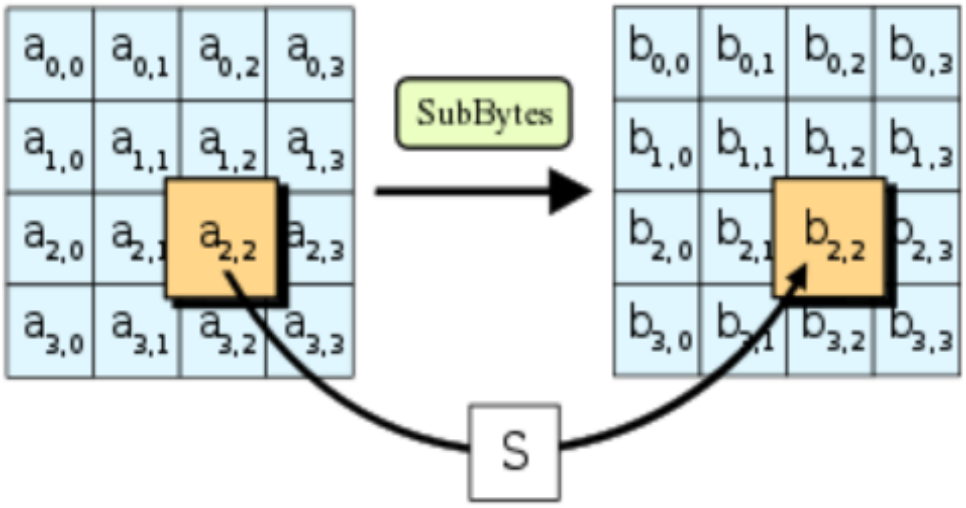
Εικόνα 2.1: AES Encryption Algorithm

2.3.1 SubBytes() Μετασχηματισμός

Σε αυτό το βήμα εφαρμόζεται ένας μη γραμμικός μετασχηματισμός αντικαθιστώντας κάθε byte του πίνακα κατάστασης χρησιμοποιώντας το αντίστοιχο byte από το S-box. Το S-box είναι ένας look-up-table (LUT) που δημιουργείται από έναν αναστρέψιμο πίνακα 16x16 με τα 4 πιο αριστερά bits από κάθε byte να χρησιμοποιούνται ως δείκτης γραμμής και τα 4 πιο δεξιά bits ως δείκτης στήλης.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

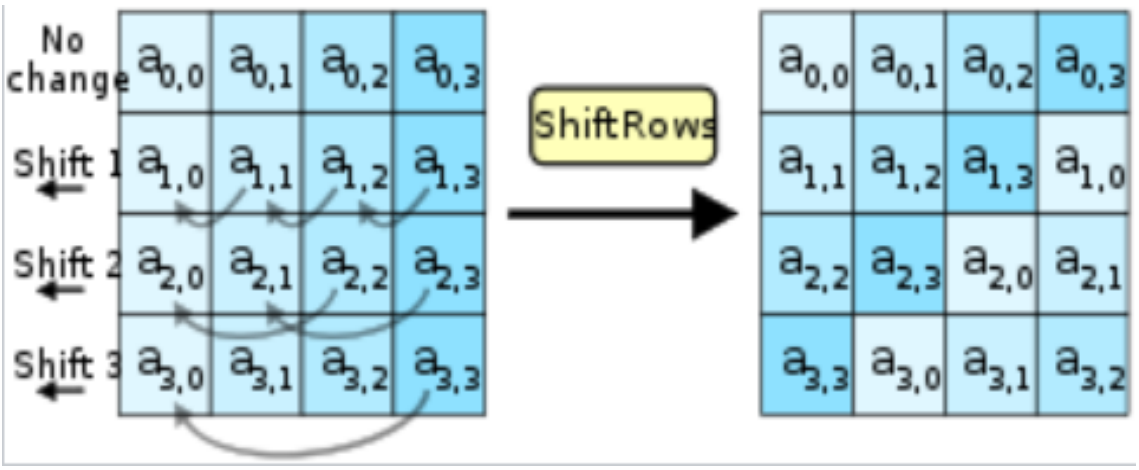
AES Substitution Box (S-Box)



Εικόνα 2.2: SubBytes Transformation

2.3.2 ShiftRows() Μετασχηματισμός

Ο μετασχηματισμός ShiftRows επιδρά στις γραμμές του πίνακα κατάστασης, ολισθαίνοντας αριστερά κυκλικά τα bytes κάθε γραμμής κατά μια ορισμένη μετατόπιση. Για τον AES, η πρώτη σειρά παραμένει αμετάβλητη. Κάθε byte της δεύτερης σειράς μετατοπίζεται κατά μία θέση προς τα αριστερά. Ομοίως, η τρίτη και η τέταρτη σειρά μετατοπίζονται κατά δύο και τρεις θέσεις αντίστοιχα. Με αυτόν τον τρόπο, κάθε στήλη της κατάστασης εξόδου κατά την εφαρμογή του βήματος ShiftRows αποτελείται από bytes από κάθε στήλη της κατάστασης εισόδου. Η σημασία αυτού του βήματος έγκειται στην αποφυγή της ανεξάρτητης κρυπτογράφησης των στηλών, κατά την οποία ο AES θα εκφυλιστεί σε τέσσερα ανεξάρτητα block ciphers.

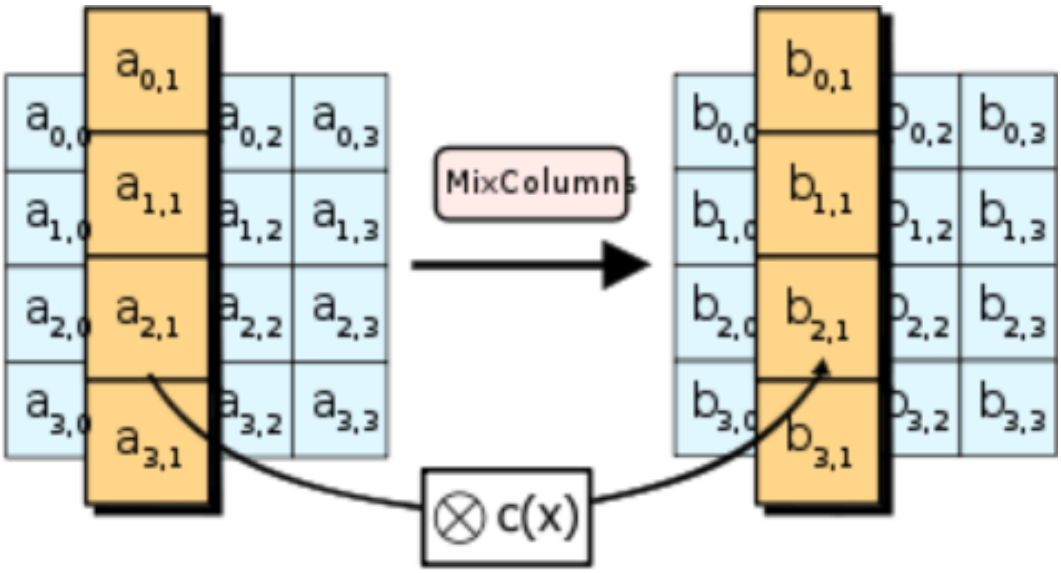


Εικόνα 2.3: ShiftRows Transformation

2.3.3 MixColumns() Μετασχηματισμός

Στο βήμα MixColumns, τα τέσσερα bytes κάθε στήλης της κατάστασης συνδυάζονται χρησιμοποιώντας έναν αντιστρέψιμο γραμμικό μετασχηματισμό. Η συνάρτηση MixColumns λαμβάνει τέσσερα bytes ως είσοδο και εξάγει τέσσερα bytes, όπου κάθε byte εισόδου επηρεάζει και τα τέσσερα bytes εξόδου. Μαζί με την ShiftRows, η MixColumns παρέχει διάχυση στην κρυπτογράφηση. Σε αυτό το βήμα κάθε στήλη αντιμετωπίζεται ως πολυώνυμο στο GF(2^8) και στη συνέχεια πολλαπλασιάζεται mod (x^4+1) με ένα σταθερό πολυώνυμο c(x)=3x^3+x^2+x+2. Έτσι προκύπτει ένας σταθερός πίνακας με τον οποίο πολλαπλασιάζεται κάθε στήλη για να προκύψει η νέα στήλη

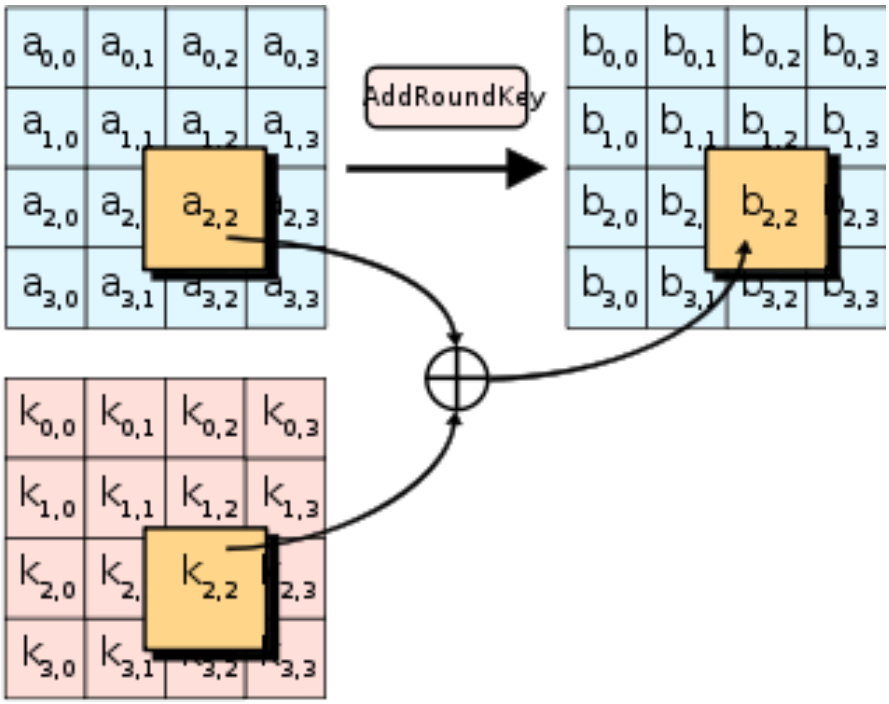
$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$



Εικόνα 2.4: Mix Columns Transformation

2.3.4 AddRoundKey() Μετασχηματισμός

Στο μετασχηματισμό AddRoundKey, το υποκλειδί συνδυάζεται με τον πίνακα κατάστασης. Για κάθε γύρο, ένα υποκλειδί προκύπτει από το κύριο κλειδί και έχει το ίδιο μέγεθος με τον πίνακα κατάστασης. Το υποκλειδί προστίθεται συνδυάζοντας τον πίνακα κατάστασης με το αντίστοιχο byte του υποκλειδιού χρησιμοποιώντας πράξη XOR.



Εικόνα 2.5: AddRoundKey Transformation

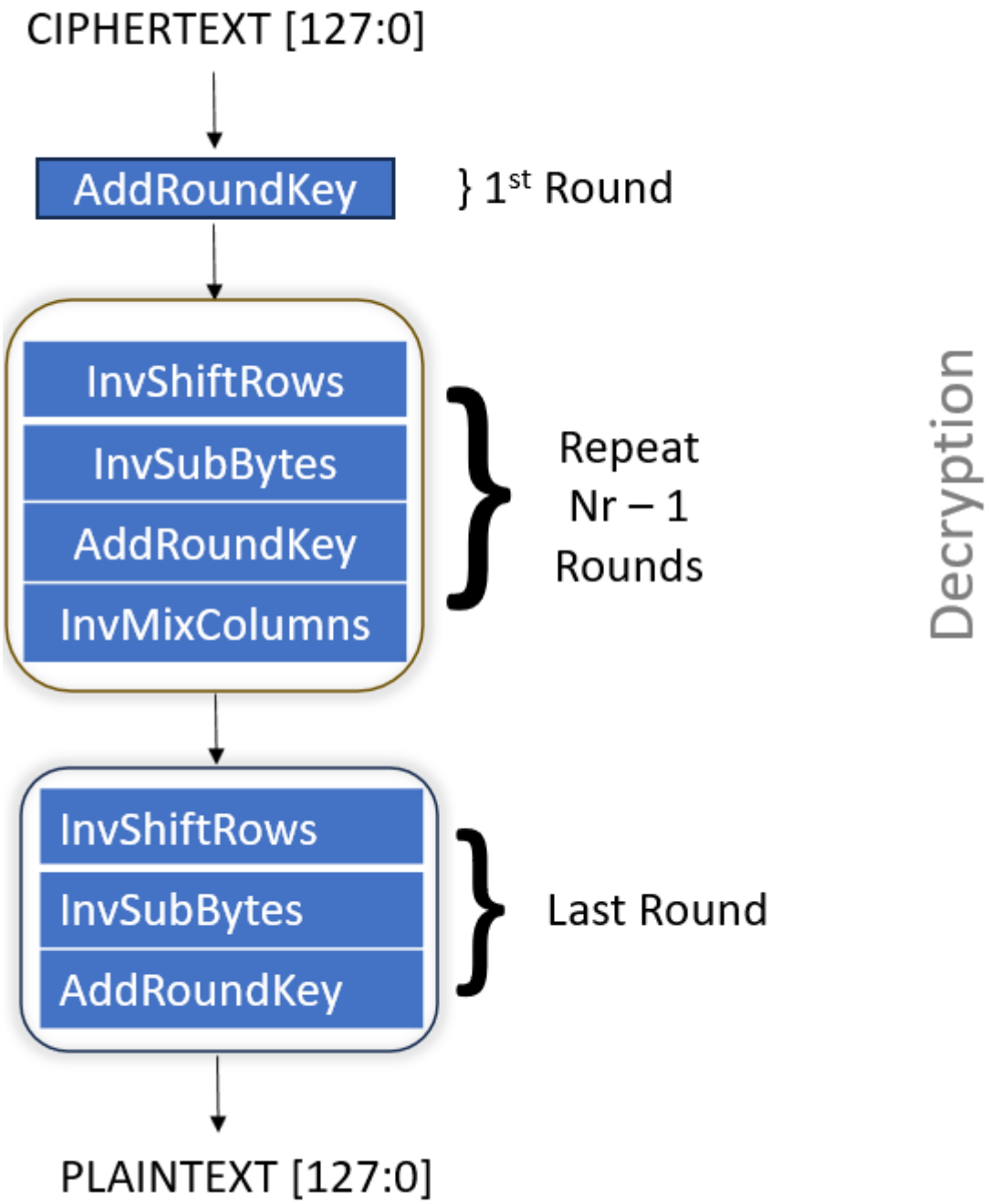
2.3.5 Επέκταση κλειδιού (Key Expansion)

Στον AES το αρχικό κλειδί επεκτείνεται έτσι ώστε να χρησιμοποιείται διαφορετικό κλειδί σε κάθε γύρο. Ο συνολικός αριθμός των κλειδιών που χρειάζονται για την πλήρη υλοποίηση του αλγορίθμου είναι n*κλειδιά γύρου +1 και κάθε κλειδί χωρίζεται σε λέξεις των 32 bit (4, 6 ή 8 για κλειδί 128, 192 ή 256 bit αντίστοιχα). Για τον πρώτο γύρο το κλειδί είναι το αρχικό κλειδί, δηλαδή οι 4 πρώτες λέξεις w[0], w[1],w[2],w[3]. Για τους επόμενους γύρους κάθε λέξη w[i] κατασκευάζεται από την w[i-1] και την w[i-4] με ένα XOR μεταξύ τους. Αλλά για κάθε w[i] με i mod 4 = 0 υπάρχει ένας επιπλέον υπολογισμός: w[i] = ((w[i-4]) S-box (RotWord(w[i-1]))) XOR (Rcon). Όπου:

- RotWord: Περιστρέφει μια λέξη κάνοντας κυκλική ολίσθηση κατά μία θέση αριστερά (RotWord(w[0],w[1],w[2],w[3]) = w[1],w[2],w[3],w[0])
- SubWord: Αντικαθιστά κάθε byte από w[i-4] σε κάθε byte από RotWord(w[i-1])
- Rcon(Round Constant): Είναι μια σταθερά με τα 3 δεξιότερα bit πάντα ίσα με 0.

2.4 Δομή AES Decryptor

Για την αποκρυπτογράφηση του ciphertext, ακολουθείται η αντίστροφη διαδικασία, δηλαδή ο αλγόριθμος που χρησιμοποιείται προκύπτει από την αναστροφή του αλγορίθμου κρυπτογράφησης και την αντικατάσταση των συναρτήσεών από τις αντίστροφές τους (InvShiftRows(), InvSubBytes(), InvMixColumns() και AddRoundKey()). Η αρίθμηση των γύρων στην αποκρυπτογράφηση γίνεται αντίστροφα (ακολουθώντας φθίνουσα πορεία), ώστε να επιτυγχάνεται η αντιστοιχία των κλειδιών κάθε γύρου.



Εικόνα 2.6: AES Decryption Algorithm

2.4.1 InverseShiftRows() Μετασχηματισμός

Ο μετασχηματισμός Inverse ShiftRows εκτελεί την αντίστροφη κυκλική ολίσθηση του ShiftRows, ολισθαίνοντας τα byte δεξιά αντί για αριστερά. Η πρώτη σειρά δεν μετατοπίζεται. Η δεύτερη σειρά μετατοπίζεται δεξιά κατά 1 byte. Η τρίτη σειρά μετατοπίζεται δεξιά κατά 2 byte. Η τέταρτη σειρά μετατοπίζεται δεξιά κατά 3 byte.

2.4.2 InverseSubBytes() Μετασχηματισμός

Κατά τον μετασχηματισμό Inverse SubBytes εφαρμόζεται η αντίστροφη διαδικασία από αυτή του SubBytes μετασχηματισμού, κάνοντας χρήση του Inverse S-box για την αντικατάσταση των byte.

Inverse S-box

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

Εικόνα 2.7: Inverse S-Box

2.4.3 Inverse AddRoundKey() Μετασχηματισμός

Ο αντίστροφος μετασχηματισμός InvAddRoundKey ταυτίζεται με τον μετασχηματισμό AddRoundKey που εκτελείται κατά την κρυπτογράφηση, διότι η μοναδική πράξη που εκτελείται κατ' αυτή τη διαδικασία είναι η XOR η οποία είναι από μονή της αντιστρεπτή.

2.4.4 InverseMixColumns() Μετασχηματισμός

Στο βήμα InverseMixColumns, εφαρμόζεται ο αντίστροφος μετασχηματισμός του MixColumns. Σε αυτό το βήμα κάθε στήλη του πίνακα κατάστασης αντιμετωπίζεται ως πολυώνυμο στο GF(2^8) και στη συνέχεια πολλαπλασιάζεται mod (x^4+1) με ένα σταθερό πολυώνυμο d(x)=0Bx^3+0Dx^2+09x+0E. Το d(x) είναι το αντίστροφο πολυώνυμο του c(x) (d(x) * c(x) = 1 mod(x^4+1)). s'(x) = d(x) ⊗ s(x):

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

2.5 Παράδειγμα υλοποίησης του AES-128 Encryptor

Παρακάτω παρατίθεται ένα παράδειγμα κρυπτογράφησης κειμένου κάνοντας χρήση του προτύπου AES με είσοδο 128 bit και κλειδί κρυπτογράφησης 128 bit.

Χρησιμοποιώντας ως είσοδο το κείμενο (plaintext): Two One Nine Two

και ως κλειδί (key): Thats my Kung Fu

Με μετατροπή σε δεκαεξαδικό σύστημα προκύπτει αντίστοιχα:

Plaintext (128 bit HEX): 54 77 6F 20 4F 6E 65 20 4E 69 6E 65 20 54 77 6F

Key (128 bit HEX): 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75

Το κλειδί του πρώτου γύρου προκύπτει ως εξής: Χωρίζοντας το αρχικό κλειδί σχηματίζονται τέσσερεις λέξεις w[0] = (54, 68, 61, 74),w[1] = (73, 20, 6D, 79),w[2] = (20, 4B, 75, 6E),w[3] = (67, 20, 46, 75)

Η λέξη w[3] = (67, 20, 46, 75) μετασχηματίζεται:

i. Αριστερή κυκλική ολίσθηση κατά μια θέση: w[3] = (20, 46, 75, 67)
ii. SubBytes Μετασχηματισμός (S-Box): (B7, 5A, 9D, 85)
iii. Προσθήκη σταθεράς του γύρου (01, 00, 00, 00): g(w[3]) = (B6, 5A, 9D, 85)

Ακολούθως τα w[4],w[5],w[6] και w[7] είναι: w[4] = w[0] ⊕ g(w[3]) = (E2, 32, FC, F1), w[5] = w[4] ⊕ w[1] = (91, 12, 91, 88), w[6] = w[5] ⊕ w[2] = (B1, 59, E4, E6), w[7] = w[6] ⊕ w[3] = (D6, 79, A2, 93)

Συνεπώς το κλειδί του πρώτου γύρου είναι: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93

Ακολουθώντας σε κάθε γύρο την ίδια διαδικασία προκύπτουν τα παρακάτω κλειδιά:

Round 0: 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75

Round 1: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93

Round 2: 56 08 20 07 C7 1A B1 8F 76 43 55 69 A0 3A F7 FA

Round 3: D2 60 0D E7 15 7A BC 68 63 39 E9 01 C3 03 1E FB

Round 4: A1 12 02 C9 B4 68 BE A1 D7 51 57 A0 14 52 49 5B

Round 5: B1 29 3B 33 05 41 85 92 D2 10 D2 32 C6 42 9B 69

Round 6: BD 3D C2 B7 B8 7C 47 15 6A 6C 95 27 AC 2E 0E 4E

Round 7: CC 96 ED 16 74 EA AA 03 1E 86 3F 24 B2 A8 31 6A

Round 8: 8E 51 EF 21 FA BB 45 22 E4 3D 7A 06 56 95 4B 6C

Round 9: BF E2 BF 90 45 59 FA B2 A1 64 80 B4 F7 F1 CB D8

Round 10: 28 FD DE F8 6D A4 24 4A CC C0 A4 FE 3B 31 6F 26

ROUND 0

PLAINTEXT

$$\begin{pmatrix} 54 & 4F & 4E & 20 \\ 77 & 6E & 69 & 54 \\ 6F & 65 & 6E & 77 \\ 20 & 20 & 65 & 6F \end{pmatrix}$$

KEY

$$\begin{pmatrix} 54 & 73 & 20 & 67 \\ 68 & 20 & 4B & 20 \\ 61 & 6D & 75 & 46 \\ 74 & 79 & 6E & 75 \end{pmatrix}$$

$\oplus$

AddRoundKey

$$\begin{pmatrix} 00 & 3C & 6E & 47 \\ 1F & 4E & 22 & 74 \\ 0E & 08 & 1B & 31 \\ 54 & 59 & 0B & 1A \end{pmatrix}$$

ROUND 1

$$\begin{pmatrix} 00 & 3C & 6E & 47 \\ 1F & 4E & 22 & 74 \\ 0E & 08 & 1B & 31 \\ 54 & 59 & 0B & 1A \end{pmatrix}$$

SubBytes

$$\begin{pmatrix} 63 & EB & 9F & A0 \\ C0 & 2F & 93 & 92 \\ AB & 30 & AF & C7 \\ 20 & CB & 2B & A2 \end{pmatrix}$$

ShiftRows

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} 63 & EB & 9F & A0 \\ 2F & 93 & 92 & C0 \\ AF & C7 & AB & 30 \\ A2 & 20 & CB & 2B \end{pmatrix}$$

MixColumns

$$\begin{pmatrix} BA & 84 & E8 & 1B \\ 75 & A4 & 8D & 40 \\ F4 & 8D & 06 & 7D \\ 7A & 32 & 0E & 5D \end{pmatrix}$$

Round1Key

$$\begin{pmatrix} E2 & 91 & B1 & D6 \\ 32 & 12 & 59 & 79 \\ FC & 91 & E4 & A2 \\ F1 & 88 & E6 & 93 \end{pmatrix}$$

$$\oplus$$

AddRoundKey

$$\begin{pmatrix} 58 & 15 & 59 & CD \\ 47 & B6 & D4 & 39 \\ 08 & 1C & E2 & DF \\ 8B & BA & E8 & CE \end{pmatrix}$$

AES Round 1 output: 58 47 08 8B 15 B6 1C BA 59 D4 E2 E8 CD 39 DF CE

Ακολουθώντας την ίδια διαδικασία σε κάθε γύρο υλοποίησης του αλγορίθμου λαμβάνουμε τις έξεις εξόδους ως αποτελέσματα:

Round 2: 43 0E 09 3D C6 57 08 F8 A9 C0EB 7F 62 C8FE 37

Round 3: 78 70 99 4B 76 76 3C 39 30 7D 37 34 54 23 5B F1

Round 4: B1 08 04 E7 CA FC B1 B2 51 54 C9 6C ED E1 D3 20

Round 5: 9B 23 5D 2F 51 5F 1C 38 20 22 BD 91 68 F0 32 56

Round 6: 14 8F C0 5E 93 A4 60 0F 25 2B 24 92 77 E8 40 75

Round 7: 53 43 4F 85 39 06 0A 52 8E 93 3B 57 5D F8 95 BD

Round 8: 66 70 AF A3 25 CE D3 73 3C 5A 0F 13 74 A8 0A 54

Round 9: 09 A2 F0 7B 66 D1FC 3B 8B 9A E6 30 78 65 C4 89

Round 10: 29 57 40 1A C3 14 22 02 50 20 99 D7 5F F6B3 3A

Άρα το τελικό κρυπτογραφημένο κείμενο (ciphertext) θα είναι:

29 57 40 1A C3 14 22 02 50 20 99 D7 5F F6 B3 3A

Το οποίο όταν μετατραπεί ξανά σε κώδικα ASCII παίρνει τη μορφή:)W@Ã"P □×_ö³:

Κεφάλαιο 3

Σύνθεση Υψηλού Επιπέδου

(High-Level Synthesis)

3.1 Εισαγωγή

Η Σύνθεση Υψηλού Επιπέδου (High Level Synthesis - HLS), είναι μια αυτοματοποιημένη διαδικασία σχεδίασης η οποία μετασχηματίζει μια περιγραφή κυκλωματικής λειτουργίας σε μια δομική περιγραφή που πραγματοποιεί τη επιθυμητή λειτουργία. Τα επίπεδα που χρησιμοποιούνται συνήθως για το σχεδιασμό του συστήματος είναι το επίπεδο πυλών, το επίπεδο μεταφοράς καταχωρητών (Register Transfer Level - RTL) και το αλγοριθμικό επίπεδο.

Οι γλώσσες περιγραφής υλικού χρησιμοποιούνται κυρίως για τη σχεδίαση αρχιτεκτονικών και συστημάτων υλικού, όπου η συμπεριφορά σχεδιασμού μοντελοποιείται σε πρωτότυπα RTL. Από την άλλη πλευρά, οι γλώσσες σύνθεσης υψηλού επιπέδου μπορούν να μοντελοποιήσουν αποτελεσματικά συστήματα και εφαρμογές με λιγότερη προσπάθεια και μεγαλύτερη ευκολία διαμόρφωσης. Επομένως, προκειμένου να μειωθεί ο χρόνος σχεδίασης των συστημάτων που βασίζονται σε υλικό, χρησιμοποιούνται εργαλεία σύνθεσης υψηλού επιπέδου για τον μετασχηματισμό των συστημάτων που μοντελοποιούνται σε μια γλώσσα υψηλού επιπέδου σε RTL, υλοποίηση που μπορεί να συντεθεί σε FPGA.

Το High Level Synthesis όχι μόνο μειώνει την προσπάθεια σχεδιασμού και αποσφαλμάτωσης σε Hardware Description Language, αλλά παρέχει επίσης ευελιξία στην τελική υλοποίηση υλικού ώστε να ικανοποιούνται οι ανάγκες σχεδιασμού. Το HLS προσφέρει επίσης τη δυνατότητα συγκριτικής αξιολόγησης υλικού σε πρώιμα στάδια σχεδιασμού και την εις βάθος ανάλυση της χρήσης των διαθέσιμων πόρων κατά τον σχεδιασμό του συστήματος.

3.2 Vivado – High Level Synthesis

Το εργαλείο του Xilinx, Vivado High Level Synthesis είναι ένα HLS εργαλείο για FPGAs το οποίο επιταχύνει την διαδικασία της σχεδίασης υλικού. Λαμβάνει έναν κώδικα γραμμένο

σε C/C++ ή SystemC και τον μετατρέπει αμέσως σε RTL επίπεδο ώστε να επιτευχθεί η σύνθεσή του σε ένα FPGA, όπως φαίνεται στο Σχήμα 4.1. Το FPGA παρέχει πολύ μεγάλη παράλληλη αρχιτεκτονική με οφέλη στην απόδοση, το κόστος και την ισχύ έναντι παραδοσιακών επεξεργαστών [2].

Το εργαλείο Xilinx - Vivado HLS είναι ένα εργαλείο σύνθεσης υψηλού επιπέδου για το σχεδιασμό συστημάτων υλικού (FPGA), το οποίο μετατρέπει μια προδιαγραφή σχεδιασμού C, όπως η C, η C++ και η System-C, σε σχεδιασμό RTL. Το εργαλείο περνάει από διάφορες φάσεις για να επιτύχει μια βελτιστοποιημένη υλοποίηση. Το Vivado - HLS προγραμματίζει τις λογικές πράξεις για κάθε κύκλο ρολογιού και αναθέτει πόρους υλικού για κάθε προγραμματισμένη λειτουργία.

Από τη ροή σχεδιασμού, εξάγεται η λογική ελέγχου, δημιουργώντας μια μηχανή πεπερασμένων καταστάσεων που διαδέχεται τις λειτουργίες στη σχεδίαση RTL. Το εργαλείο Vivado HLS συνθέτει τις λειτουργίες C σε μπλοκ εντός της ιεραρχίας RTL. Τα ορίσματα της συνάρτησης ανώτατου επιπέδου(top-level function) συντίθενται, ανάλογα με την περίπτωση, σε θύρες εισόδου/εξόδου συνοδευόμενες από κατάλληλα σήματα χειραψίας (handshaking signals).

Το εργαλείο HLS επιτρέπει στον προγραμματιστή να αναλύσει και να βελτιστοποιήσει τη σχεδίαση. Βασιζόμενο στην προκαθορισμένη συμπεριφορά, στους περιορισμούς και τυχόν directives βελτιστοποίησης το εργαλείο δημιουργεί μια βέλτιστη υλοποίηση RTL της σχεδίασης υψηλού επιπέδου. Στη συνέχεια, το εργαλείο παράγει ένα σύνολο αναφορών συνθέσεων που χρησιμοποιούνται για την ανάλυση της υλοποίησης και μπορούν να εφαρμοστούν διάφορες βελτιστοποιήσεις για την ικανοποίηση των προδιαγραφών και των περιορισμών της σχεδίασης

Στην παρούσα εργασία υλοποιήθηκε ο αλγόριθμος κρυπτογράφησης AES-128 σε γλώσσα C++. Ο αλγόριθμος ήταν συντέθηκε και προσομοιώθηκε από το εργαλείο Vivado HLS για τον έλεγχο των της λειτουργικότητας της σχεδίασης RTL. Στη συνέχεια, η σχεδίαση αναλύθηκε και βελτιστοποιήθηκε για να επιτευχθεί υψηλότερη δυνατή απόδοση με την ελάχιστη καθυστέρηση και την μικρότερη χρήση των διαθέσιμων πόρων του συστήματος.

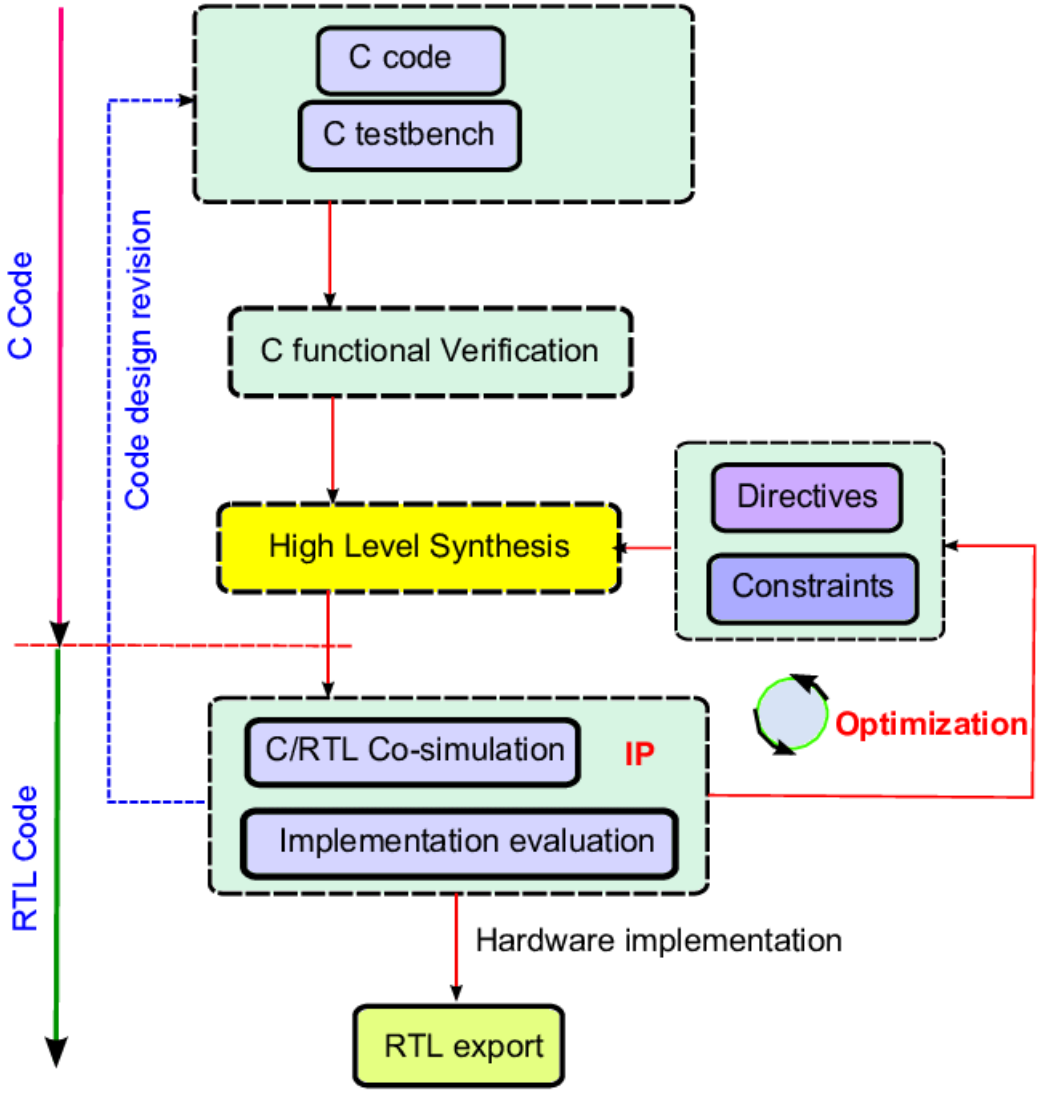
3.3 Δομή λειτουργίας του Vivado - HLS

3.3.1 Διάγραμμα Λειτουργίας Vivado - HLS

Αρχικά το πρόγραμμα λαμβάνει ως εισόδους αρχεία κώδικα σε γλώσσες C/C++ ή SystemC, μέσω των οποίων γίνεται μια περιγραφή της επιθυμητής υλοποίησης. Η περιγραφή αυτή συνδέεται με ένα αρχείο testbench που δημιουργείται από τον χρήστη, μέσω του οποίου μπορεί να επαληθευτεί η ορθή λειτουργία του κυρίως προγράμματος. Στη συνέχεια τα αρχεία C μαζί με το testbench περνούν από τη φάση της μεταγλώττισης (compilation) ώστε να επαληθευτεί η ορθή λειτουργία τους για την παραγωγή των δεδομένων στην τελική έξοδο.

Έπειτα ακολουθεί η διαδικασία της Σύνθεσης Υψηλού Επιπέδου, όπου εισάγονται τα απαραίτητα directives και constraints για την περιγραφή του RTL κυκλώματος. Πριν την έναρξη της διαδικασίας Σύνθεσης Υψηλού Επιπέδου, το πρόγραμμα παρέχει στον χρήστη την δυνατότητα να επιλέξει της επιθυμητή γλώσσα Hardware Description Language (VHDL/Verilog), καθώς και τον προσομοιωτή(simulator). Με το πέρας της διαδικασίας της σύνθεσης, παράγεται ένα κύκλωμα της μορφής IP(Intellectual Property) με ένα σύνολο αρχείων εξόδου συμπεριλαμβανομένων αρχείων RTL.

Μόλις δημιουργηθεί το RTL IP, ακολουθεί η επαλήθευση του σχεδιασμού μέσω C/RTL Co-simulation. Οι είσοδοι που παρέχονται από το testbench τροφοδοτούνται απευθείας στον RTL κώδικα που έχει παραχθεί και ο simulator ελέγχει το αποτέλεσμα εξόδου. Μετά εφαρμόζονται οι περιορισμοί και τα directives και το σύστημα που σχεδιάστηκε υλοποιείται σε μια συσκευή FPGA. Το Vivado HLS παρέχει επίσης την δυνατότητα βελτιστοποίησης του συστήματος που υλοποιείται ως προς την απόδοση του (σε χρόνο που απαιτείται για την υλοποίηση αλλά και σε πόρους που χρησιμοποιούνται). Αν η επιδόσεις της σχεδίασης δεν είναι οι επιθυμητές, υπάρχει η δυνατότητα εισαγωγής κάποιων directives ή και επιπλέον περιορισμών ώστε να αναδιαμορφωθεί η αρχική περιγραφή του κώδικα. Τέλος, μόλις επιτευχθεί η βέλτιστη δυνατή σχεδίαση, παράγεται ένα τελικό IP κύκλωμα.



Εικόνα 3.1: Vivado-HLS Design Flow

3.3.2 Βασικές Μετρικές

Οι μετρικές που παρέχονται από το Vivado HLS είναι εκτιμήσεις της συμπεριφοράς της υλοποίησης, που περιγράφονται από στοιχεία όπως:

- Χρονισμός(Timing): Η ταχύτερη συχνότητα ρολογιού που μπορεί να επιτευχθεί από τη μονάδα. Στις ρυθμίσεις του project μπορεί να οριστεί μια κατώτατη τιμή χρόνου που τίθεται ως στόχος, μαζί με ένα αποδεκτό περιθώριο αβεβαιότητας. Στους σχεδιασμούς που υλοποιήθηκαν, οι τιμές αυτές ορίστηκαν στα 10 ns και 0 ns αντίστοιχα. Εάν ο εκτιμώμενος χρόνος υπερβαίνει τον στόχο, η μονάδα δεν θα είναι σε θέση να λειτουργήσει στη συχνότητα που έχει οριστεί και θα πρέπει να πραγματοποιηθούν περαιτέρω βελτιστοποιήσεις.
- Καθυστέρηση (Latency): Ο αριθμός των κύκλων ρολογιού που περνούν από τη στιγμή που δίνεται μια είσοδος στο σύστημα μέχρι την παραγωγή της εξόδου. Η ελάχιστη και η μέγιστη καθυστέρηση έχουν σημασία εάν ο κώδικας περιέχει βρόχους υπό όρους που περιέχουν λογική η οποία μπορεί να επηρεάσει

τους συνολικούς κύκλους παραγωγής της εξόδου. Η μετρική διάστημα αρχικοποίησης (initialization interval (II)) αντιπροσωπεύει τον ρυθμό σε κύκλους ρολογιού, της ικανότητας του συστήματος να επεξεργάζεται νέες εισόδους (throughput) και συνδέεται με το μέγεθός τους. Η καθυστέρηση μπορεί να εξεταστεί με περισσότερες λεπτομέρειες: Για κάθε υπομονάδα, προκειμένου να προσδιοριστεί η επιβάρυνση της κάθε μίας. Για τους βρόχους, όπου η τιμή της "καθυστέρησης επανάληψης" αφορά τον αριθμό των κύκλων που χρειάζεται μία επανάληψη για να ολοκληρωθεί. Το "Tripcount" απεικονίζει τον αριθμό των επαναλήψεων του βρόχου.

- Loop Latency: Ο αριθμός των κύκλων που απαιτούνται μέχρι να ολοκληρωθούν όλες οι επαναλήψεις ενός βρόχου.
- Χρήση πόρων (Resource Utilization): Απεικονίζει τον αριθμό των στοιχείων, όπως LUTs, Flip-Flops, BRAMs κλπ., που χρησιμοποιεί κάθε μονάδα για μνήμη, πολυπλεξία, υλοποιήσεις FIFO, καταχωρητές και άλλα. Και πάλι, το Resource Utilization μπορεί να προβληθεί με περισσότερες λεπτομέρειες για κάθε υπομονάδα. Εάν η χρήση υπερβαίνει το 100%, δημιουργείται μια προειδοποίηση από το σύστημα.
- Initiation Interval: Ο αριθμός των κύκλων ρολογιού που απαιτούνται πριν το κύκλωμα να μπορεί να δεχτεί νέες εισόδους.
- Loop Initiation Interval: Ο αριθμός των κύκλων ρολογιού που απαιτούνται πριν η επόμενη επανάληψη βρόχου αρχίσει να επεξεργάζεται δεδομένα.

Οι εν λόγω μετρήσεις προκύπτουν μετά τη διαδικασία σύνθεσης. Πιο ακριβείς τιμές μπορούν να εξαχθούν σε μεταγενέστερα στάδια της ανάπτυξης, όπως, για παράδειγμα, εάν επιλεγεί η επιλογή αξιολόγησης "Vivado Synthesis, place and route" όταν πρόκειται να εξαχθεί το RTL. Ωστόσο, πρόκειται για μια αργή διαδικασία που αντιμετωπίζει την αρχή της έγκαιρης γνώσης. Τέλος, το Vivado HLS επιστρέφει αποτελέσματα μόνο για την υπό εξέταση μονάδα. Στις περιπτώσεις όπου χρησιμοποιούνται δύο διαφορετικές μονάδες, γίνονται ορισμένες υποθέσεις σχετικά με τα αποτελέσματα της σύνθεσης.

3.3.3 Αριθμητική Κινητής Υποδιαστολής

Σημαντικό χαρακτηριστικό του Vivado HLS είναι η δυνατότητα υλοποίησης υλικού αριθμητικής κινητής υποδιαστολής που παρέχει.

Αν και η σταθερού σημείου αριθμητική είναι πιο γρήγορη και πιο αποτελεσματική, μερικές φορές είναι επιθυμητό να υλοποιηθεί κάποιος υπολογισμός χρησιμοποιώντας αριθμητική κινητής υποδιαστολής. Ενώ οι πράξεις αριθμητικής σταθερού σημείου μπορούν να επιτύχουν μεγαλύτερη ακρίβεια στον υπολογισμό των αποτελεσμάτων, έχουν περιορισμένο εύρος και για αυτό απαιτείται εστιασμένη ανάλυση προκειμένου να προσδιοριστούν τα μοτίβα με τα οποία αναπτύσσονται τα bit σε ένα πολύπλοκο σχέδιο και πολλές φορές χρειάζεται να εισαχθούν ενδιάμεσοι τύποι δεδομένων για να επιτευχθεί το βέλτιστο δυνατό αποτέλεσμα.

Οι μορφές κινητής υποδιαστολής αντιπροσωπεύουν πραγματικούς αριθμούς σε μεγαλύτερο εύρος, γεγονός που επιτρέπει τη χρήση ενός τύπου δεδομένων σε μεγάλες ακολουθίες. Ως προς το σχεδιαστικό τμήμα, ωστόσο, καταλαμβάνονται περισσότεροι πόροι του συστήματος και αυξάνεται η καθυστέρηση, εφόσον η λογική που απαιτείται για την εφαρμογή μιας δεδομένης αριθμητικής λειτουργίας είναι πιο περίπλοκη. Το Vivado HLS υποστηρίζει τύπους δεδομένων κινητής υποδιαστολής και διπλού μήκους, οι οποίοι είναι είτε μονής είτε διπλής ακριβείας, καθώς και αριθμητικούς και σχεσιακούς τελεστές και μετατροπές μορφής (integer σε float και float σε double).

3.3.4 Περιορισμοί

Μολονότι το εργαλείο Vivado HLS υποστηρίζει ένα ευρύ φάσμα των γλωσσών C/C++ και SystemC, ορισμένες δομές και λειτουργίες δεν ανταποκρίνονται στις απαιτήσεις της σύνθεσης και είναι πιθανό να οδηγήσουν σε σφάλματα κατά την σχεδίαση.

Συγκεκριμένα για να μπορεί ο κώδικας να υλοποιείται στο Synthesis πρέπει οι συναρτήσεις να περιέχουν τη λειτουργικότητα του σχεδιασμού, οι δομές στον κώδικα να έχουν πεπερασμένο μέγεθος και η υλοποίηση τους να είναι σαφής και οι λειτουργίες δεν μπορεί να εκτελούνται μέσω κλήσεων συστήματος στο λειτουργικό σύστημα.

3.3.5 Directives

Τα Directives είναι οδηγίες για το πώς η μεταγλώττιση του RTL θα χειριστεί τη σύνθεση. Η χρήση τους απαιτεί καλή κατανόηση του κώδικα και της σχέσης μεταξύ των εμπλεκόμενων στοιχείων, καθώς αφορούν αρχιτεκτονικές αποφάσεις. Η χρήση τους όμως, μπορεί να επηρεάσει σημαντικά την απόδοση του κώδικα. Η χρήση των Directives δεν εγγυάται ότι η ορθότητα της λειτουργικότητας διατηρείται μετά τις αλλαγές που επιβάλλουν. Η καλύτερη πρακτική είναι να ελέγχεται η επίδρασή τους μέσω C/RTL Co-simulation.

Τα Directives εφαρμόζονται μέσα στον κώδικα με τη μορφή ορισμών (pragmas) ή, όπως επιτρέπει το Vivado HLS, μέσω ενός εξειδικευμένου παραθύρου UI, με κάθε ενέργεια να καταγράφεται σε ένα ξεχωριστό αρχείο .tcl που ορίζεται για κάθε λύση. Τα Directives εφαρμόζονται είτε σε δομές δεδομένων είτε σε μπλοκ κώδικα, όπως βρόχους ή συναρτήσεις.

Στο Vivado HLS, μπορούν να εφαρμοστούν 21 Directives:

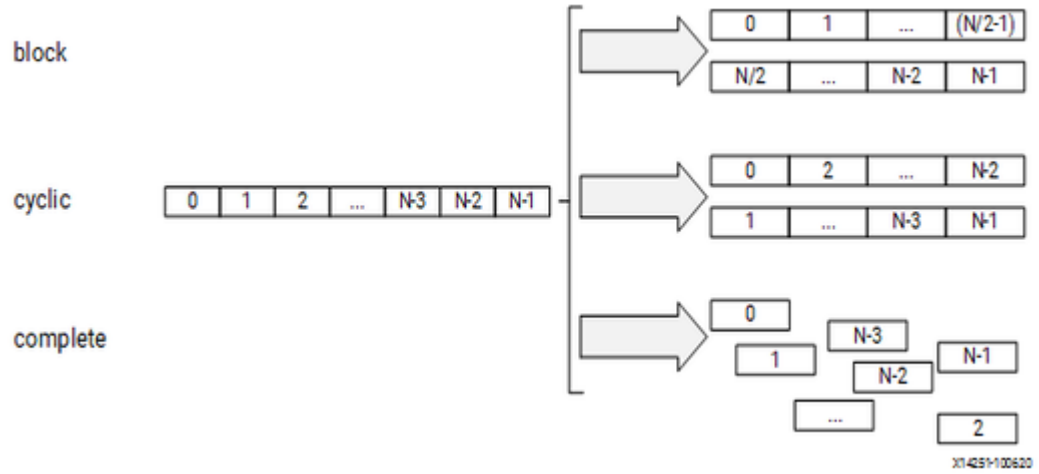
- ALLOCATION: Επιτρέπει τον περιορισμό του αριθμού των πόρων υλικού που εκχωρούνται για την υλοποίηση μιας λειτουργικότητας. Ως αποτέλεσμα, η Σύνθεση επιβάλλει τον διαμοιρασμό των περιορισμένων πόρων σε όλη τη λειτουργικότητα, κάτι που μπορεί να ωφελήσει τη χρήση της περιοχής εις βάρος της καθυστέρησης.

- ARRAY_MAP: Κάθε καθορισμένος πίνακας συντίθεται σε ένα ξεχωριστό στοιχείο μνήμης (από προεπιλογή, ROM για πίνακες μόνο για ανάγνωση, RAM για ανάγνωση-εγγραφή). Αυτό μπορεί προφανώς να αντισταθμιστεί στην περίπτωση μικρών πινάκων. Οι πίνακες που ανήκουν στην ίδια εμβέλεια μπορούν να συναθροιστούν σε έναν μεγαλύτερο πίνακα, ο οποίος μπορεί στη συνέχεια να τοποθετηθεί σε ένα ενιαίο στοιχείο μνήμης αντί για ξεχωριστά, και να μειώσει σημαντικά τη χρήση της μνήμης. Υπάρχουν δύο τρόποι αντιστοίχισης:

-Οριζόντια αντιστοίχιση: Όλοι οι δεδομένοι πίνακες συνενώνονται σε έναν, με μέγεθος στοιχείου ίσο με το μέγιστο μέγεθος που υπάρχει μεταξύ των συνενωμένων πινάκων. Όταν ορίζεται η μετατόπιση (offset), υποδεικνύει τον τρόπο τοποθέτησης των στοιχείων στο νέο πίνακα.

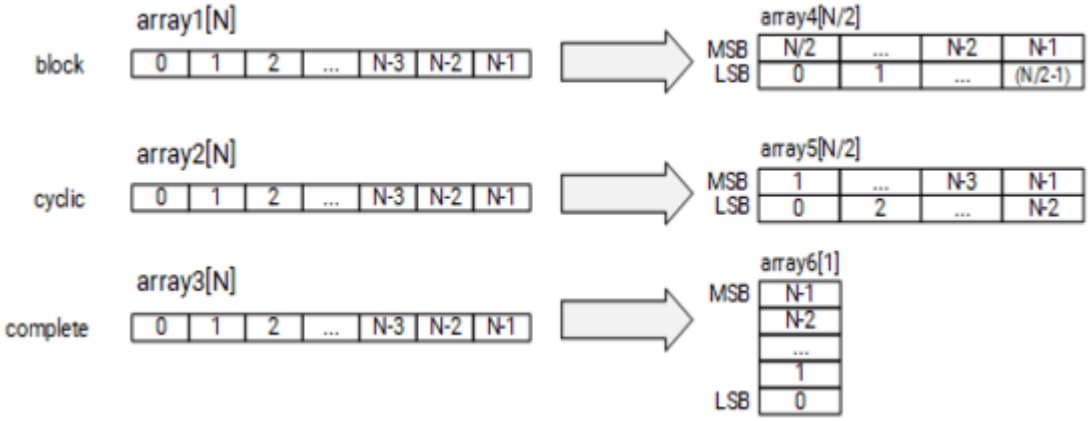
-Κατακόρυφη αντιστοίχιση: Τα δεδομένα των συγκεκριμένων πινάκων συνενώνονται, δημιουργώντας στοιχεία μεγαλύτερου μεγέθους.

- ARRAY_PARTITION : Στην περίπτωση αυτή, ένας πίνακας διαχωρίζεται σε περισσότερους από έναν πίνακες και, επομένως, απαιτούνται περισσότερες θέσεις μνήμης. Αν και αυτό φαίνεται αντιπαραγωγικό από την άποψη της έκτασης, όταν χρησιμοποιούνται περισσότεροι διαχωρισμένοι πίνακες, δημιουργούνται περισσότερες θύρες πρόσβασης σε αυτούς και έτσι αυξάνεται το εύρος ζώνης δεδομένων. Ο αριθμός των πινάκων που προκύπτουν ορίζεται από τον παράγοντα N. Οι πολυδιάστατοι πίνακες μπορούν επίσης να αντιμετωπιστούν ανάλογα, δεδομένης της απαρίθμησης των διαστάσεων πάνω στις οποίες πρόκειται να γίνει ο διαχωρισμός. Υπάρχουν τρεις περιπτώσεις:
-Block partitioning: Ο δεδομένος πίνακας χωρίζεται σε N μικρότερους πίνακες ίσου μεγέθους. Η χρήση του παρουσιάζεται στην Εικόνα 18.
-Κυκλική διαμέριση: Τα στοιχεία του δεδομένου πίνακα θα τοποθετούνται εναλλάξ στους διαμερισμένους πίνακες.
-Πλήρης διαμέριση: Κάθε στοιχείο του δεδομένου πίνακα τοποθετείται σε έναν ειδικό πίνακα. Όταν πρόκειται για πολυδιάστατους πίνακες, ο καταχωρητής θα περιέχει τη διεύθυνση του πίνακα που θα δημιουργήσει η διαμέριση.



Εικόνα 3.2: Array_Partition

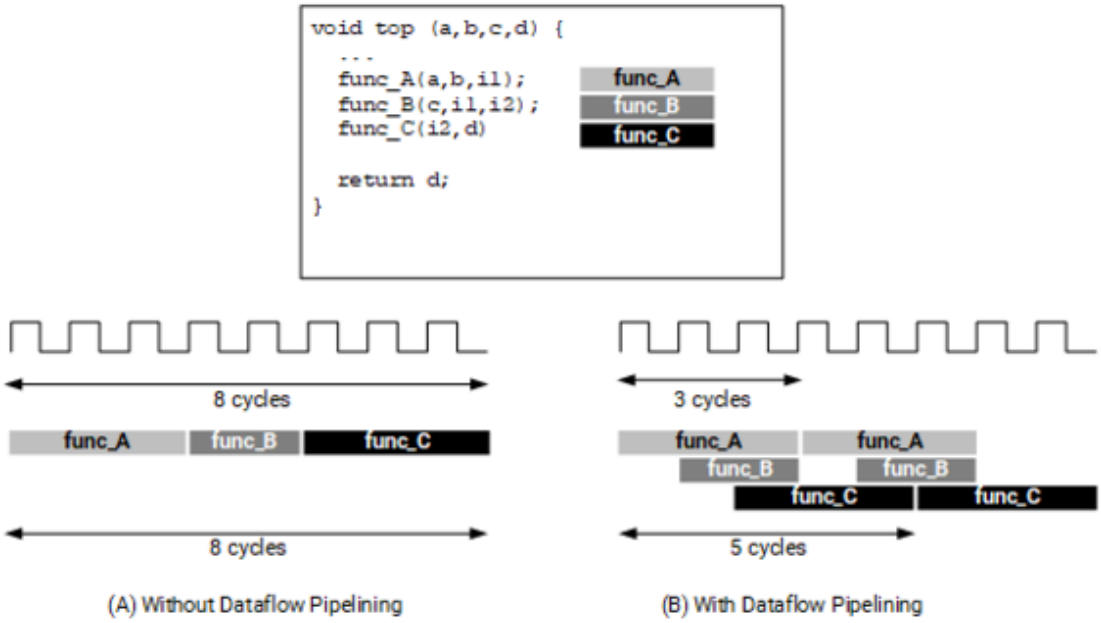
- ARRAY_RESHAPE : Ο καθορισμένος πίνακας διαιρείται πρώτα σύμφωνα με την οδηγία ARRAY_PARTITION και στη συνέχεια συγχωνεύεται κάθετα, δημιουργώντας έναν πίνακα "ευρύτερων" στοιχείων, όπως στην ARRAY_MAP, όταν επιλέγεται η κάθετη αντιστοίχιση. Επιτρέπει την καλύτερη αξιοποίηση των στοιχείων μνήμης της σχεδίασης.



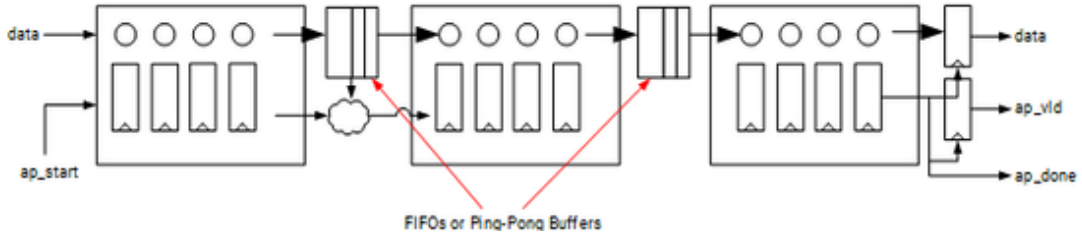
Εικόνα 3.3: Array_Reshape

- CLOCK (υποστηρίζεται μόνο στο SystemC) : Όταν εφαρμόζεται σε μια συνάρτηση, αυτή λειτουργεί σε διαφορετική συχνότητα ρολογιού από αυτή που εφαρμόζεται στη μονάδα, παράγοντας ή καταναλώνοντας έτσι δεδομένα με διαφορετικό ρυθμό από τον υπόλοιπο σχεδιασμό.

- DATAFLOW : Αυτή η Directive εφαρμόζει μια μορφή παραλληλισμού μεταξύ συναρτήσεων ή βρόχων που εξαρτώνται από δεδομένα. Σε πολλές περιπτώσεις, για να ξεκινήσει η εκτέλεση μιας συνάρτησης, πρέπει να έχουν ολοκληρωθεί όλες οι πράξεις πάνω στα απαιτούμενα δεδομένα, ακόμα και αν το αποτέλεσμα των πράξεων μπορεί να είναι διαθέσιμο νωρίτερα. Η DATAFLOW επιτρέπει τη χρήση αυτών των δεδομένων σε προηγούμενο στάδιο, εξοικονομώντας στο σχεδιασμό μερικούς κύκλους ρολογιού.



Εικόνα 3.4: Dataflow



Εικόνα 3.5: Δομή που δημιουργείται κατά την εφαρμογή της Dataflow Directive

- DEPENDENCE : Χρησιμοποιείται για την παροχή πληροφοριών για τον καλύτερο χειρισμό των εξαρτήσεων. Το Vivado HLS υστερεί όσον αφορά την ικανότητα να εντοπίζει και να επιλύει σωστά τα προγραμματιστικά εμπόδια, ειδικά όταν υπάρχουν διαφορετικές κατευθύνσεις τους στον κώδικα, όπως:

-Αληθινές εξαρτήσεις : Εμφανίζονται όταν μια θέση μνήμης προσπελαύνεται για μια λειτουργία ανάγνωσης, πριν οριστικοποιηθεί μια λειτουργία εγγραφής (Read after Write - RAW).

-Αντι-εξαρτήσεις : Εμφανίζονται όταν μια θέση μνήμης προσπελαύνεται για μια λειτουργία εγγραφής, πριν οριστικοποιηθεί μια λειτουργία ανάγνωσης (Write after Read - WAR).

-Εξαρτήσεις εξόδου : Εμφανίζονται όταν προσπελαύνεται μια θέση μνήμης για μια λειτουργία εγγραφής, πριν οριστικοποιηθεί μια άλλη λειτουργία εγγραφής (Write after Write- WAW).

Ειδικά για τους βρόχους, εμφανίζονται δύο πρόσθετες εξαρτήσεις:

-Εξάρτηση ανεξάρτητη από το βρόχο: Μια από τις παραπάνω κατευθύνσεις εμφανίζεται για μια θέση μνήμης εντός της ίδιας επανάληψης του βρόχου. Ορίζεται ως "intra" στις επιλογές της Directive "type".

-Εξάρτηση από τη μεταφορά βρόχου: Μία από τις παραπάνω κατευθύνσεις εμφανίζεται για μια θέση μνήμης σε διαφορετικές επαναλήψεις βρόχου. Ορίζεται ως "inter" στις επιλογές της Directive "type".

Στην περίπτωση που εφαρμόζεται pipelining των επαναλήψεων, η εξάρτηση loop-carry θα επηρεάσει την εφαρμογή της. Αυτό όμως δεν ισχύει για τις εξαρτήσεις που δεν εξαρτώνται από το βρόχο. Ωστόσο, το Vivado HLS αντιμετωπίζει όλες τις παραπάνω ως μη επιλύσιμες εξαρτήσεις. Ως εκ τούτου, η χρήση της Directive DEPENDENCE

βοηθά τον μεταγλωττιστή να προσδιορίσει τη φύση αυτών των εξαρτήσεων και να αγνοήσει τις "ψευδείς", επιτρέποντας στις άλλες Directives να έχουν πλήρη ισχύ.

- EXPRESSION_BALANCE : Από προεπιλογή, το Vivado HLS "εξισορροπεί" το δέντρο που δημιουργεί η εκτέλεση, προκειμένου να ελαχιστοποιηθεί η καθυστέρηση. Η Directive επιτρέπει την απενεργοποίηση αυτής της συμπεριφοράς για την ελαχιστοποίηση της χρήσης των πόρων.

- FUNCTION_INSTANTIATE : Αυτή η Directive μπορεί να χρησιμοποιηθεί όταν υπάρχουν πολλαπλές εμφανίσεις της ίδιας υπο-συνάρτησης σε μια συνάρτηση. Η σύνθεσή τους σε διαφορετικές περιπτώσεις επιτρέπει τη βελτιστοποίησή τους για τις διαφορετικές μεταβλητές. Ίδια αντίγραφα συναρτήσεων στο ίδιο επίπεδο ιεραρχίας συντίθενται σε ένα ενιαίο μπλοκ. Αυτή η Directive, βοηθά τον παραλληλισμό, συντελώντας στη μείωση της καθυστέρησης και στην αύξηση της απόδοσης εις βάρος της περιοχής. Η επιβάρυνση της περιοχής θα μπορούσε να είναι πολύ μεγάλη, αλλά σε συνδυασμό με άλλες Directives περιορισμού της περιοχής, μπορεί να επιτευχθεί ισορροπία των επιδόσεων.

- INLINE : Χρησιμοποιείται για την αφαίρεση μιας συνάρτησης από το μπλοκ ιεραρχίας. Η λειτουργικότητά της περνάει στη συνάρτηση που την καλεί, όπου μπορεί να συνεργαστεί καλύτερα με την παρούσα λειτουργικότητα. Η Directive INLINE μπορεί επίσης να εφαρμοστεί αναδρομικά, εισάγοντας μια σειρά από υπο-συναρτήσεις στην επόμενη στην ιεραρχία τους.

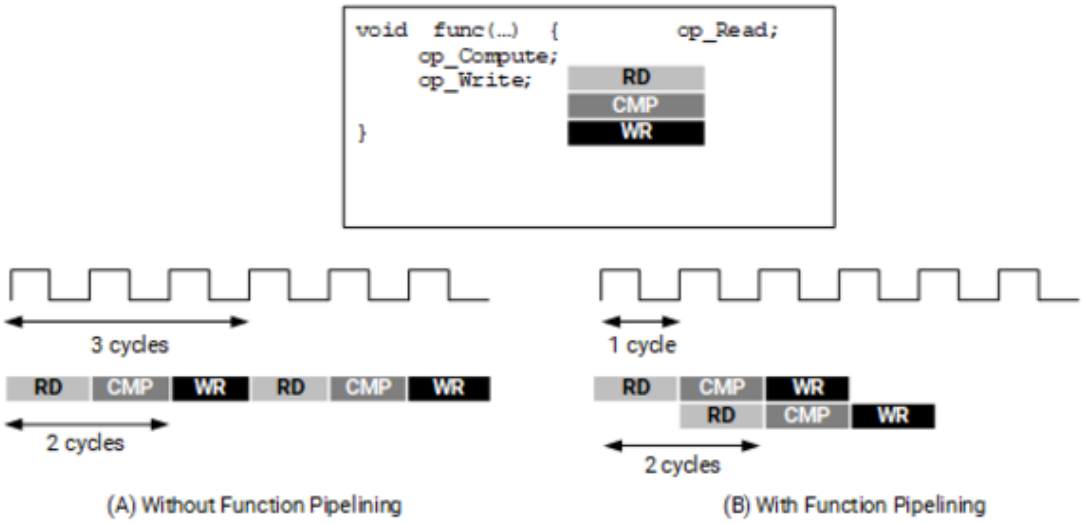
- INTERFACE : Επιτρέπει τον προσδιορισμό του τρόπου με τον οποίο το μπλοκ θα επικοινωνεί με άλλα στοιχεία της σχεδίασης, μέσω των θυρών διασύνδεσης. Τα πρόσθετα σήματα θα καθορίσουν τη συμπεριφορά του συντιθέμενου μπλοκ, όπως για παράδειγμα ο ρυθμός με τον οποίο μπορεί να δέχεται νέες εισόδους.

- LATENCY : Ορίζει τους περιορισμούς μέγιστης και ελάχιστης καθυστέρησης. Εάν η καθυστέρηση της σύνθεσης πέσει κάτω από το καθορισμένο ελάχιστο όριο, η καθυστέρηση θα αυξηθεί πρακτικά μέσω της χρήσης καταχωρητών. Εάν η προκύπτουσα καθυστέρηση υπερβαίνει το καθορισμένο μέγιστο όριο, το εργαλείο θα προσπαθήσει να πλησιάσει προς αυτό, εφαρμόζοντας βελτιστοποιήσεις μείωσης της καθυστέρησης που δεν αντιτίθενται στα αποτελέσματα των υπολοίπων Directives.

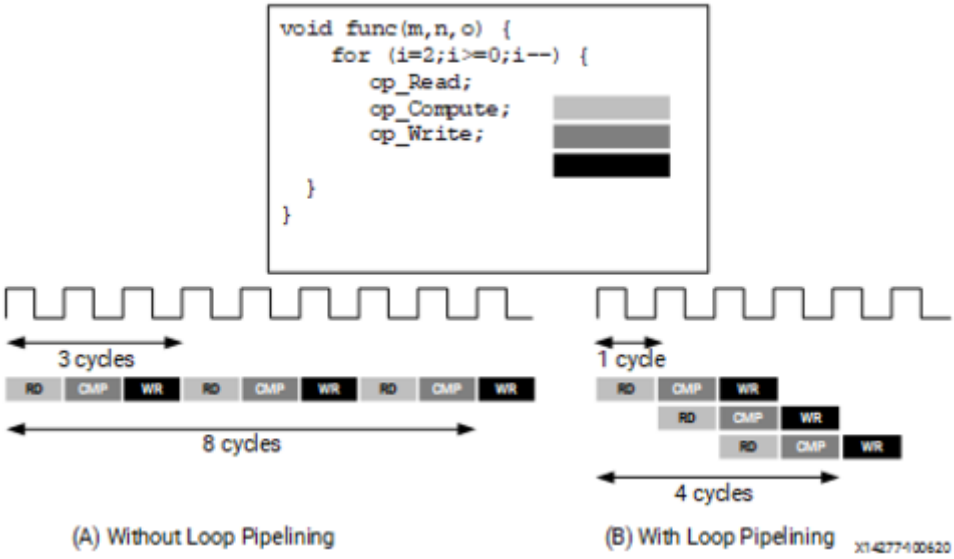
- LOOP_MERGE : Επιτρέπει το συνδυασμό της λογικής εντός των βρόχων επανάληψης, για τους οποίους έχει οριστεί η Directive, σε έναν ενιαίο βρόχο. Σαφώς, ο αριθμός των επαναλήψεων όλων των βρόχων που πρόκειται να συγχωνευθούν πρέπει να είναι ο ίδιος.

- OCCURRENCE: Χρησιμοποιείται όταν μέρος ενός κώδικα εντός μιας pipelined λογικής βρόχου εκτελείται υπό όρους. Ένα τέτοιο OCCURRENCE θα λειτουργεί με χαμηλότερο ρυθμό απόδοσης (initialization interval) από την υπόλοιπη συνάρτηση/βρόχο (ίσο με 1 όταν είναι pipelined). Το Vivado HLS ορίζει ως διάστημα αρχικοποίησης ενός code block το μέγιστο διάστημα αρχικοποίησης που παρουσιάζεται στη λειτουργία, πράγμα που σημαίνει ότι τέτοιες περιπτώσεις μπορεί να οδηγήσουν σε υποβάθμιση της απόδοσης του σχεδιασμού, ειδικά αν είναι σπάνιες. Η Directive OCCURRENCE επιτρέπει τη διάκριση αυτών των υπό συνθήκη εκτελέσεων και τον ορισμό ξεχωριστού initialization interval για αυτές.

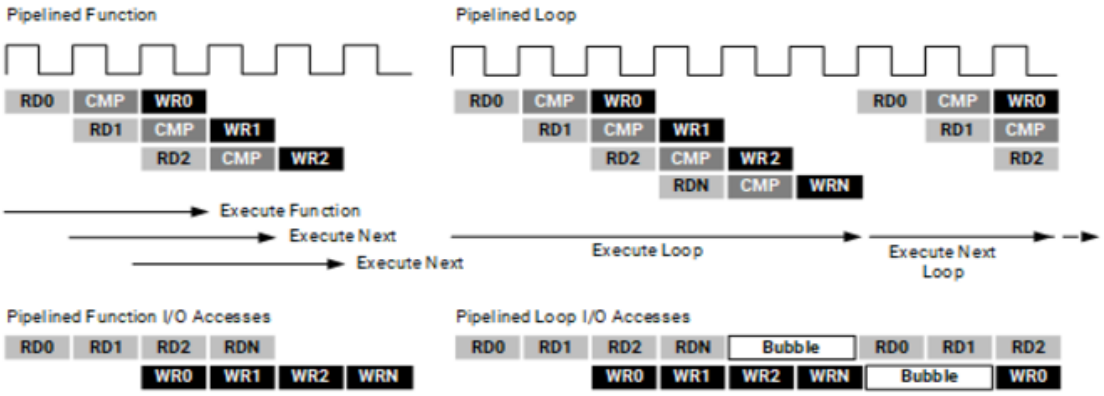
- PIPELINE : Το PIPELINE είναι μια κοινή τεχνική παραλληλισμού που μπορεί να αυξήσει σημαντικά την εκτέλεση μιας συνάρτησης/βρόχου. Κανονικά, στην περίπτωση υλοποιήσεων υλικού, ένα μπλοκ κώδικα, όπως μια συνάρτηση ή το κύριο σώμα ενός βρόχου, δεσμεύει τα εμπλεκόμενα λογικά μπλοκ για όλη τη διάρκεια της εκτέλεσης. Αυτό είναι αντιπαραγωγικό, δεδομένου ότι μπορούν να χρησιμοποιηθούν για άλλους υπολογισμούς, χωρίς να επηρεάζεται η λειτουργικότητα του εκτελούμενου κώδικα. Η χρήση του μπορεί να επηρεαστεί από εξαρτήσεις δεδομένων, περιστατικά και εγγενώς αργά μπλοκ, τα οποία μπορούν να βελτιστοποιηθούν με τη χρήση άλλων Directives.



Εικόνα 3.6: Function Pipeline



Εικόνα 3.7: Loop Pipeline



Εικόνα 3.8: Function & Loop Pipeline

- PROTOCOL : Επιτρέπει τη δημιουργία ενός χειροκίνητου πρωτοκόλλου για την καθορισμένη περιοχή κώδικα, καθορίζοντας τον τρόπο με τον οποίο τα σήματα RTL που προκύπτουν θα πρέπει να συμπεριφέρονται εντός της λειτουργικότητάς της. Για παράδειγμα, οι εντολές μπορεί να μην φέρουν καμία εξάρτηση μεταξύ τους, ωστόσο η σειρά εκτέλεσης πρέπει να ακολουθείται ρητά. Η περιοχή κώδικα μπορεί να εκτελεστεί παράλληλα με την υπόλοιπη λειτουργικότητα ή ξεχωριστά, ανάλογα με τον επιλεγμένο τρόπο λειτουργίας (floating, fixed respectively).

- RESOURCE : Η Directive RESOURCE χρησιμοποιείται για τον ορισμό των πόρων (πυρήνα) που θα χρησιμοποιηθούν για τη σύνθεση μιας μεταβλητής ή μιας αριθμητικής πράξης, ειδικά όταν υπάρχουν πολλαπλοί ορισμοί πυρήνων στον κώδικα. Παραδείγματα χρήσης είναι τα εξής:

-Για τους τελεστές : Η επιλογή του πυρήνα πρόσθεσης και αφαίρεσης εξαρτάται από τον σχετικό τύπο δεδομένων (ακέραιος, μισής ακρίβειας, απλής ακρίβειας, διπλής ακρίβειας), αν υλοποιείται με τη χρήση της τεχνικής pipeline, καθώς και από τη χρήση ή όχι ενός τύπου DSP (DSP48, full DSP, medium DSP, no-DSP). Δέκα διαφορετικοί πυρήνες υπάρχουν μόνο για την πρόσθεση και την αφαίρεση. Άλλες επιλογές υπάρχουν για πυρήνες πολλαπλασιασμού και διαίρεσης, λογαριθμικούς και εκθετικούς υπολογισμούς, τετραγωνική ρίζα, αντιστροφή και πολυπλεξία.

-Αποθήκευση : Διατίθενται τρεις τύποι μνήμης FIFO, RAM και ROM. Ο αριθμός των θυρών μπορεί επίσης να καθοριστεί, όταν πρόκειται για RAM και ROM. Ο τύπος του πόρου μπορεί επίσης να οριστεί, είτε πρόκειται για BRAM, Distributed RAM (LUTRAM), Ultra RAM (URAM) ειδικά για RAM και Shift Registers ειδικά για FIFO.

- STABLE : Μπορεί να οριστεί για μεταβλητές οποιουδήποτε τύπου που αποτελούν είσοδο ή έξοδο μιας περιοχής του κώδικα και καθορίζει ότι δεν χρειάζονται σήματα ελέγχου συγχρονισμού. Εφαρμόζεται βέλτιστα σε μεταβλητές read-only, όπου τα δεδομένα δεν μπορούν να μεταβληθούν. Όταν εφαρμόζεται σε δεδομένα που μπορούν να εγγραφούν σε οποιοδήποτε σημείο της εκτέλεσης, η συμπεριφορά των δεδομένων πρέπει να εξετάζεται διεξοδικά. Σε κάθε περίπτωση, γλιτώνει την εφαρμογή από μερικούς κύκλους ρολογιού που διαφορετικά θα απαιτούνταν για την επικύρωση του σημάτων εισόδου/εξόδου.

- STREAM : Υλοποιεί μια συνεχή τροφοδοσία δεδομένων, χρήσιμη σε ορισμένα σενάρια σχεδιασμού. Όταν χρησιμοποιείται σε έναν σχεδιασμό, μπορεί να έχει μία από τις ακόλουθες χρήσεις:

-Για βελτιστοποίηση ροής δεδομένων : Επιτρέπει την αλλαγή του τύπου πίνακα από την προεπιλεγμένη RAM σε FIFO.

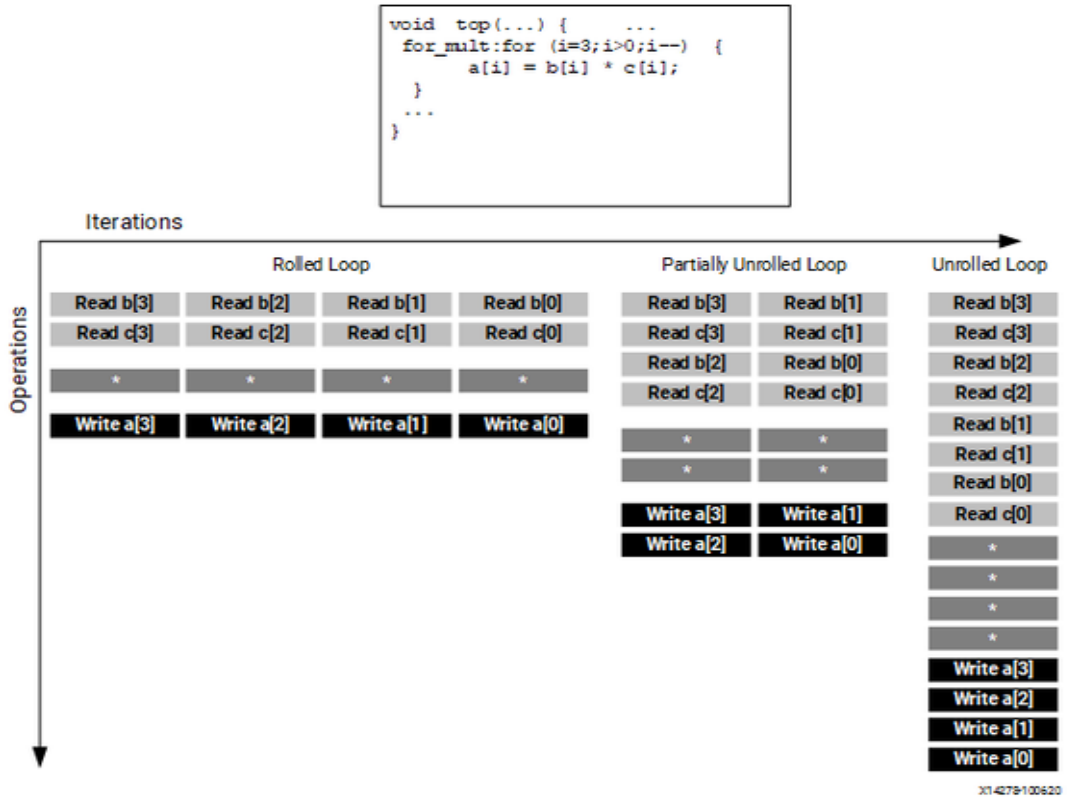
-Για διαμόρφωση FIFO : Επιτρέπει τον καθορισμό του μεγέθους της δομής FIFO.

- TOP : Χρησιμοποιείται για τον προσδιορισμό της ανώτατης μονάδας της σχεδίασης μέσω της χρήσης μιας Directive. Εναλλακτικά, μπορεί να καθοριστεί μέσω των ρυθμίσεων του solution.

- UNROLL : Το ξετύλιγμα βρόχων είναι μια άλλη κοινή τεχνική προγραμματισμού που έχει γίνει συνήθης πρακτική λόγω του παραλληλισμού. Επιτρέπει την υλοποίηση του σώματος ενός βρόχου σε διαφορετικές περιπτώσεις, οι οποίες μπορούν να εκτελούνται ταυτόχρονα, μειώνοντας έτσι σημαντικά την καθυστέρηση της σχεδίασης. Με βάση το επίπεδο του ξετυλίγματος ορίζονται:

-Πλήρες ξετύλιγμα : Εφαρμόζεται όταν η Directive δίνει εντολή στο Synthesis να υλοποιεί κάθε επανάληψη σε μια ειδική υλοποίηση. Η πρωταρχική προϋπόθεση είναι ότι ο αριθμός των επαναλήψεων είναι σταθερός. Για παράδειγμα, οι βρόχοι while και do-while που εξετάζουν συνθήκες δεν μπορούν να ξετυλιχθούν με τη χρήση αυτής της Directive. Εφαρμόζεται βέλτιστα όταν οι επαναλήψεις δεν περιέχουν εξαρτήσεις δεδομένων μεταξύ τους, αλλά μπορεί να είναι πολύ δαπανηρή σε επίπεδο περιοχής.

-Μερικό ξετύλιγμα : Για να εξισορροπηθεί το κόστος περιοχής και η καθυστέρηση, μπορεί να δημιουργηθεί ένας συγκεκριμένος αριθμός αντιγράφων, ξετυλίγοντας τον βρόχο κατά έναν παράγοντα N. Το N πρέπει να είναι ακέραιο, πολλαπλάσιο του συνολικού αριθμού των απαιτούμενων επαναλήψεων, διαφορετικά, θα πρέπει να ελέγχεται η συνθήκη εξόδου. Αυτό επιτρέπει το ξετύλιγμα βρόχων με άγνωστο αριθμό επαναλήψεων.



Εικόνα 3.9: Loop_Unroll

- RESET (μόνο για δεδομένα) : Επιτρέπει τη διαμόρφωση της επαναφοράς μέσω στατικών ή global μεταβλητών. Μπορεί να είναι ενεργοποιημένη (προεπιλογή) ή απενεργοποιημένη.

Τα Directives που χρησιμοποιήθηκαν για τη βελτιστοποίηση του αλγόριθμου AES-128 στη παρούσα διπλωματική εργασία είναι οι εξής:

- Function Inline
- Function Pipeline
- Loop Pipeline
- Loop Unroll
- Array_Partition
- Dataflow
- Loop_Flatten

Κεφάλαιο 4

Υλοποίηση του AES-128 σε Υλικό και Εφαρμογή Βελτιστοποιήσεων

4.1 Υλοποίηση του AES-128 σε VHDL

Η πρώτη υλοποίηση του AES-128 σε υλικό πραγματοποιήθηκε με χρήση της γλώσσας περιγραφής υλικού VHDL με σκοπό να γίνει σύγκριση ανάμεσα σε μια κλασσική υλοποίηση του αλγορίθμου σε επίπεδο RTL και στην μετέπειτα υλοποίηση του σε Σύνθεση Υψηλού Επιπέδου, ως προς την απόδοση του συστήματος στον εκάστοτε σχεδιασμό. Για τον σχεδιασμό του συστήματος χρησιμοποιήθηκε το σχεδιαστικό εργαλείο Modelsim (Quartus Prime 18.1).

Από την υλοποίηση προέκυψε η εξής εξομοίωση:



Θέτοντας συχνότητα 10000ps (ή 10ns) λαμβάνουμε ως καθυστέρηση για την υλοποίηση ενός πλήρους κύκλου του αλγορίθμου latency=110,050ns και η απόδοση (Throughput) υπολογιζεται:

$$\text{Throughput} = \frac{Data(in\ bits)}{Nc * Tclk} \longrightarrow \text{Throughput} = \frac{128\ bits}{110{,}050ns}$$

Nc = 10 rounds

Throughput = 1,16 Gbps

Η χρήση των διαθέσιμων πόρων του υλικού για τον σχεδιασμό και την υλοποίηση του αλγορίθμου είναι:

Resources Utilization

	Available	Used	Resource Utilization(%)
LUT	145440	7971	5,48%
FF	290880	2269	0,78%

Συγκεντρωτικά: Resource Utilization (%) = 2,4%

4.2 Υλοποίηση του AES-128 σε Vivado-HLS

Η υλοποίηση του AES-128 σε High Level Synthesis, έγινε με χρήση του εργαλείου Vivado-HLS 2019.2. Για τη δημιουργία των solutions ορίστηκε περίοδος κύκλου ρολογιού 10ns χωρίς uncertainty. Η συσκευή που χρησιμοποιήθηκε για τον σχεδιασμό είναι η xcku025-ffva-1-c της οικογένειας Kintex-Ultrascale Παρακάτω παρατίθενται οι μετρήσεις που παρείχε η σύνθεση κατά την υλοποίηση του αλγορίθμου:

Synthesis Report for 'aes_ctr128_axis'

General Information

Date:	Mon Jun 12 13:33:19 2023
Version:	2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project:	hls_prj
Solution:	solution0
Product family:	kintexu
Target device:	xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.190 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
488	488	4.880 us	4.880 us	488	488	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_152	aes_ecb128	485	485	4.850 us	4.850 us	485	485	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	1082	6628	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	969	-	-
Total	0	0	2051	6885	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	~0	4	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_152	aes_ecb128	0	0	1082	6628	0
Total	1	0	0	1082	6628	0

DSP48E

N/A

Memory

N/A

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_216_p2	+	0	0	39	32	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	2	0	0	167	160	129

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	27	5	1	5
ap_phi_mux_v2_V_phi_fu_145_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_131	9	2	1	2
v2_V_reg_142	9	2	32	64
v2_i_i_i_i_0184_i_reg_122	9	2	32	64
Total	90	19	132	267

Register

Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	4	0	4	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_310	1	0	1	0
grp_aes_ecb128_fu_152_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Result_s_14_reg_330	128	0	128	0
p_Result_s_reg_325	128	0	128	0
p_Val2_s_reg_298	384	0	384	0
stateIn_last_V_tmp_reg_304	1	0	1	0
storemerge_i_reg_131	1	0	1	0
trunc_ln647_reg_335	128	0	128	0
v2_V_reg_142	32	0	32	0
v2_i_i_i_i_0184_i_reg_122	32	0	32	0
Total	969	0	969	0

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

$$Throughput = \frac{128\ bits}{488 * 6{,}19ns} \longrightarrow Throughput = 42{,}4\ Mbps$$

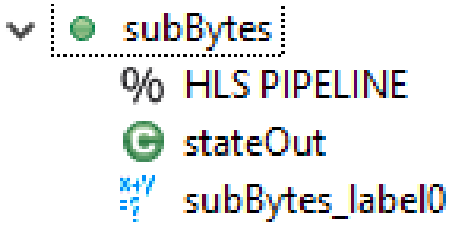
Fmax = 161.54 MHz

4.3 Εφαρμογή Βελτιστοποιήσεων

4.3.1 Βελτιστοποίηση της συνάρτησης SubBytes

Οι Βελτιστοποιήσεις που επιτευχθήκαν στην υλοποίηση του αλγορίθμου έπειτα από εφαρμογή των τεχνικών Βελτιστοποίησης: 1) Function Pipeline, 2)Loop Pipeline και 3)Loop Unroll στην συνάρτηση SubBytes() είναι οι εξής:

1)Function Pipeline



Synthesis Report for 'aes_ctr128_axis'

General Information

Date:	Mon Jun 12 14:33:27 2023
Version:	2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project:	hls_prj
Solution:	SubBytes
Product family:	kintexu
Target device:	xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.008 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
136	136	1.360 us	1.360 us	136	136	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_152	aes_ecb128	133	133	1.330 us	1.330 us	133	133	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	807	3191	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	969	-	-
Total	0	0	1776	3448	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	~0	2	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_152	aes_ecb128	0	0	807	3191	0
Total	1	0	0	807	3191	0

DSP48E

N/A

Memory

N/A

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_216_p2	+	0	0	39	32	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	2	0	0	167	160	129

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	27	5	1	5
ap_phi_mux_v2_V_phi_fu_145_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_131	9	2	1	2
v2_V_reg_142	9	2	32	64
v2_i_i_i_i_0184_i_reg_122	9	2	32	64
Total	90	19	132	267

Register

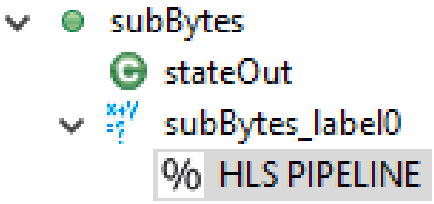
Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	4	0	4	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_310	1	0	1	0
grp_aes_ecb128_fu_152_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Result_s_13_reg_330	128	0	128	0
p_Result_s_reg_325	128	0	128	0
p_Val2_s_reg_298	384	0	384	0
stateIn_last_V_tmp_reg_304	1	0	1	0
storemerge_i_reg_131	1	0	1	0
trunc_ln647_reg_335	128	0	128	0
v2_V_reg_142	32	0	32	0
v2_i_i_i_i_0184_i_reg_122	32	0	32	0
Total	969	0	969	0

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

2)Loop Pipeline



Synthesis Report for 'aes_ctr128_axis'

General Information

Date: Mon Jun 12 14:04:35 2023
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: hls_prj
Solution: SubBytes
Product family: kintexu
Target device: xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.190 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
323	323	3.230 us	3.230 us	323	323	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_152	aes_ecb128	320	320	3.200 us	3.200 us	320	320	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	1080	6540	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	969	-	-
Total	0	0	2049	6797	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	~0	4	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_152	aes_ecb128	0	0	1080	6540	0
Total	1	0	0	1080	6540	0

DSP48E

N/A

Memory

N/A

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_216_p2	+	0	0	39	32	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	2	0	0	167	160	129

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	27	5	1	5
ap_phi_mux_v2_V_phi_fu_145_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_131	9	2	1	2
v2_V_reg_142	9	2	32	64
v2_i_i_i_i_0184_i_reg_122	9	2	32	64
Total	90	19	132	267

Register

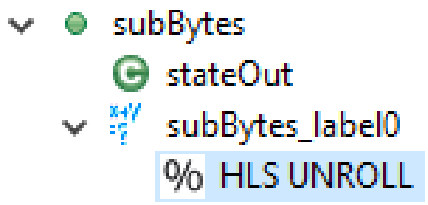
Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	4	0	4	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_310	1	0	1	0
grp_aes_ecb128_fu_152_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Result_s_15_reg_330	128	0	128	0
p_Result_s_reg_325	128	0	128	0
p_Val2_s_reg_298	384	0	384	0
stateIn_last_V_tmp_reg_304	1	0	1	0
storemerge_i_reg_131	1	0	1	0
trunc_ln647_reg_335	128	0	128	0
v2_V_reg_142	32	0	32	0
v2_i_i_i_i_0184_i_reg_122	32	0	32	0
Total	969	0	969	0

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

3)Loop Unroll



Synthesis Report for 'aes_ctr128_axis'

General Information

Date:	Mon Jun 12 14:22:39 2023
Version:	2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project:	hls_prj
Solution:	SubBytes
Product family:	kintexu
Target device:	xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.008 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
125	125	1.250 us	1.250 us	125	125	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_152	aes_ecb128	122	122	1.220 us	1.220 us	122	122	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	803	3182	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	969	-	-
Total	0	0	1772	3439	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	~0	2	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_152	aes_ecb128	0	0	803	3182	0
Total	1	0	0	803	3182	0

DSP48E

N/A

Memory

N/A

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_216_p2	+	0	0	39	32	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	2	0	0	167	160	129

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

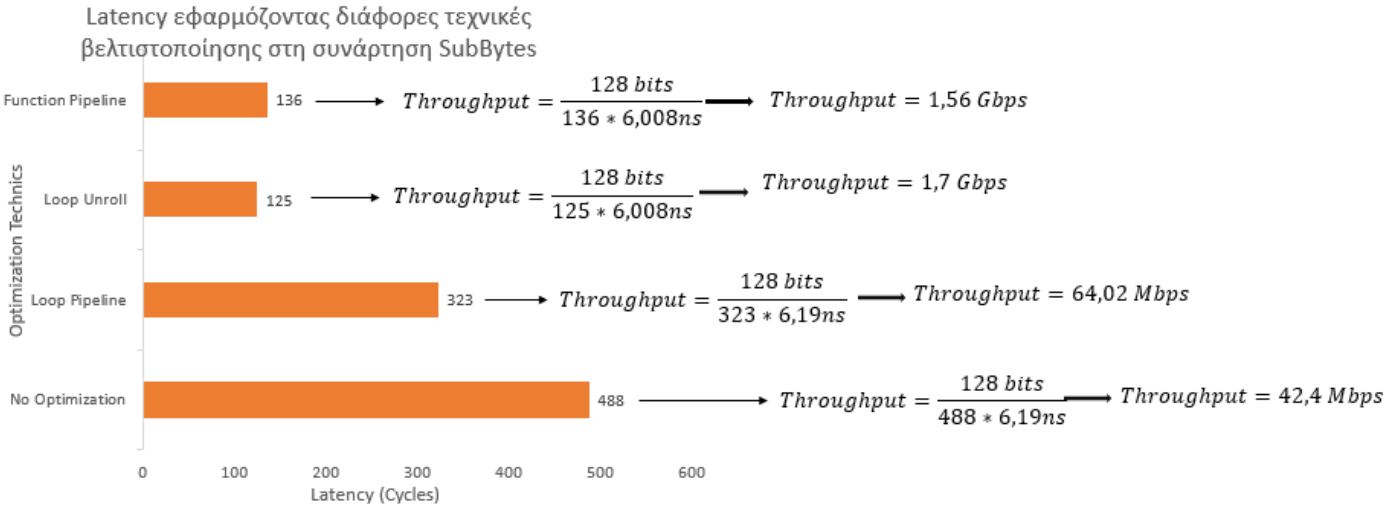
Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	27	5	1	5
ap_phi_mux_v2_V_phi_fu_145_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_131	9	2	1	2
v2_V_reg_142	9	2	32	64
v2_i_i_i_i_0184_i_reg_122	9	2	32	64
Total	90	19	132	267

Register

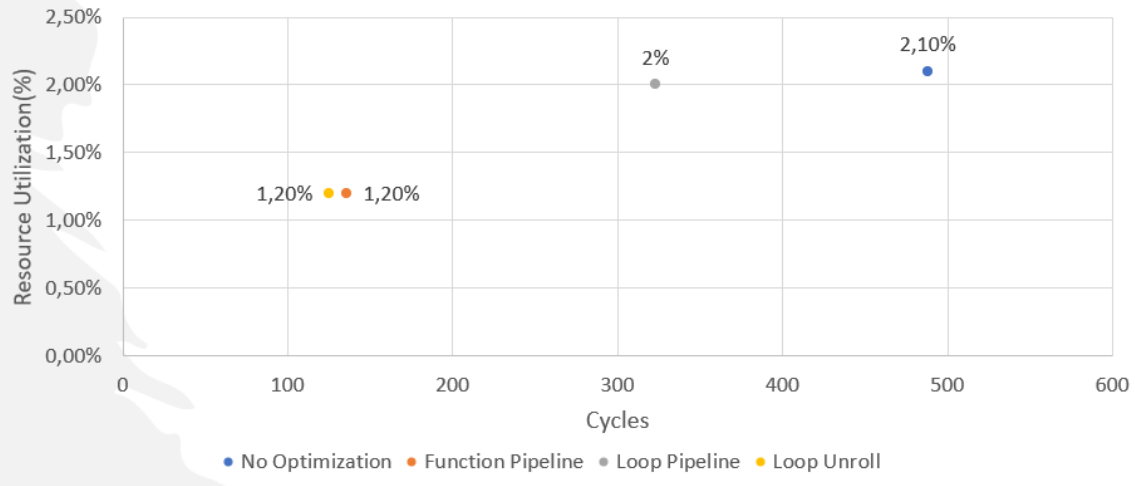
Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	4	0	4	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_310	1	0	1	0
grp_aes_ecb128_fu_152_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Result_s_13_reg_330	128	0	128	0
p_Result_s_reg_325	128	0	128	0
p_Val2_s_reg_298	384	0	384	0
stateIn_last_V_tmp_reg_304	1	0	1	0
storemerge_i_reg_131	1	0	1	0
trunc_ln647_reg_335	128	0	128	0
v2_V_reg_142	32	0	32	0
v2_i_i_i_i_0184_i_reg_122	32	0	32	0
Total	969	0	969	0

Συγκρίνοντας τις συνθέσεις που προέκυψαν από την εφαρμογή βελτιστοποιήσεων με την αρχική υλοποίηση του AES-128 εξάγουμε τα ακόλουθα συμπεράσματα ως προς την απόδοση και την χρήση διαθέσιμων πόρων του συστήματος:



Resources Utilization

	Available	No Optimization	Function Pipeline	Loop Pipeline	Loop Unroll
BRAM_18K	720	0	0	0	0
DSP48E	1152	0	0	0	0
FF	290880	2051	1776	2049	1772
LUT (Look-Up Tables)	145440	6885	3448	6797	3439
URAM	0	0	0	0	0



Παρατηρούμε συγκεντρωτικά πως η εφαρμογή τεχνικών βελτιστοποίησης στη συνάρτηση SubBytes συντελεί σε μείωση των απαιτούμενων κύκλων για την υλοποίηση του συστήματος ενώ παράλληλα υπάρχει και σημαντικός περιορισμός των πόρων του υλικού που αξιοποιούνται.

4.3.2 Βελτιστοποίηση της συνάρτησης AddRoundKey

Οι Βελτιστοποιήσεις που επιτευχθήκαν στην υλοποίηση του αλγορίθμου έπειτα από εφαρμογή των τεχνικών Βελτιστοποίησης: 1) Function Pipeline και 2)Function Dataflow στην συνάρτηση AddRoundKey() είναι οι εξής:

1)Function Pipeline

Synthesis Report for 'aes_ctr128_axis'

General Information

Date: Mon Jun 12 15:10:35 2023
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: hls_prj
Solution: AddRoundKey
Product family: kintexu
Target device: xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.190 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
400	400	4.000 us	4.000 us	400	400	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_152	aes_ecb128	397	397	3.970 us	3.970 us	397	397	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	1200	5635	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	969	-	-
Total	0	0	2169	5892	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	~0	4	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_152	aes_ecb128	0	0	1200	5635	0
Total	1	0	0	1200	5635	0

DSP48E

N/A

Memory

N/A

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_216_p2	+	0	0	39	32	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	2	0	0	167	160	129

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	27	5	1	5
ap_phi_mux_v2_V_phi_fu_145_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_131	9	2	1	2
v2_V_reg_142	9	2	32	64
v2_i_i_i_i_0184_i_reg_122	9	2	32	64
Total	90	19	132	267

Register

Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	4	0	4	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_310	1	0	1	0
grp_aes_ecb128_fu_152_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Result_s_12_reg_330	128	0	128	0
p_Result_s_reg_325	128	0	128	0
p_Val2_s_reg_298	384	0	384	0
stateIn_last_V_tmp_reg_304	1	0	1	0
storemerge_i_reg_131	1	0	1	0
trunc_ln647_reg_335	128	0	128	0
v2_V_reg_142	32	0	32	0
v2_i_i_i_i_0184_i_reg_122	32	0	32	0
Total	969	0	969	0

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

2)Function Dataflow

Synthesis Report for 'aes_ctr128_axis'

General Information

Date: Mon Jun 12 15:23:39 2023
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: hls_prj
Solution: AddRoundKey
Product family: kintexu
Target device: xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.190 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
499	499	4.990 us	4.990 us	499	499	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_152	aes_ecb128	496	496	4.960 us	4.960 us	496	496	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	1392	7330	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	841	-	-
Total	0	0	2233	7587	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	~0	5	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_152	aes_ecb128	0	0	1392	7330	0
Total	1	0	0	1392	7330	0

DSP48E

N/A

Memory

N/A

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_216_p2	+	0	0	39	32	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	2	0	0	167	160	129

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	27	5	1	5
ap_phi_mux_v2_V_phi_fu_145_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_131	9	2	1	2
v2_V_reg_142	9	2	32	64
v2_i_i_i_i_0184_i_reg_122	9	2	32	64
Total	90	19	132	267

Register

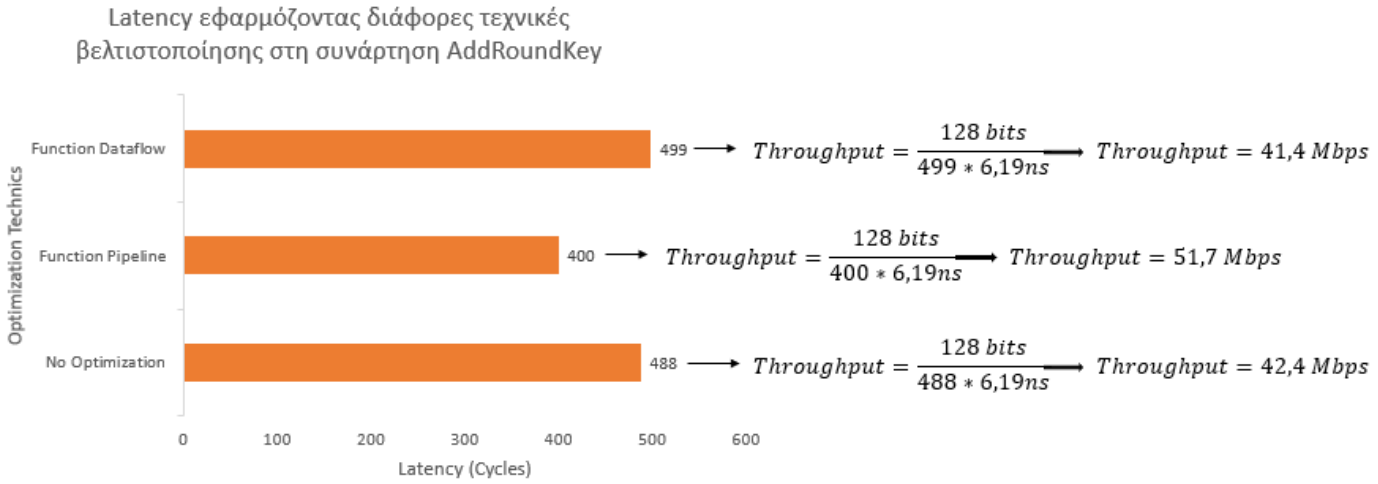
Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	4	0	4	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_315	1	0	1	0
grp_aes_ecb128_fu_152_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Result_s_20_reg_335	128	0	128	0
p_Result_s_reg_330	128	0	128	0
p_Val2_s_reg_304	384	0	384	0
stateOut_last_V_tmp_reg_309	1	0	1	0
storemerge_i_reg_131	1	0	1	0
v2_V_reg_142	32	0	32	0
v2_i_i_i_i_0184_i_reg_122	32	0	32	0
Total	841	0	841	0

Interface

Summary

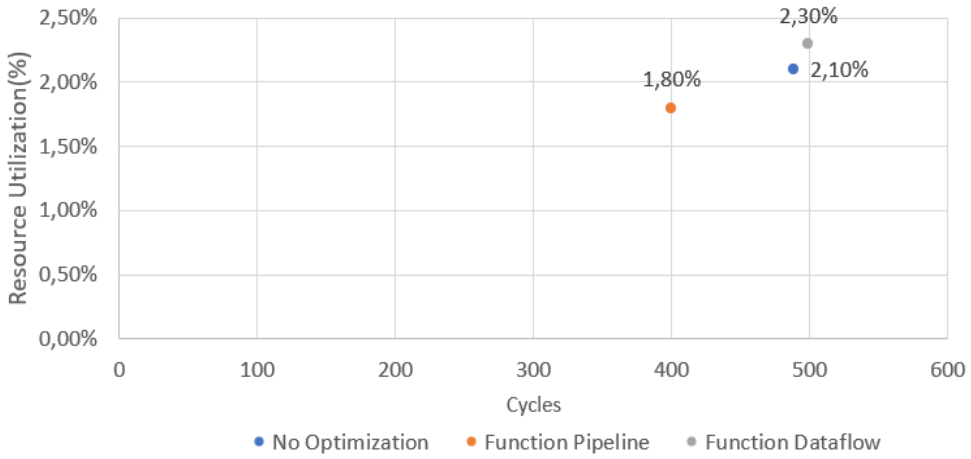
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

Συγκρίνοντας τις συνθέσεις που προέκυψαν από την εφαρμογή των βελτιστοποιήσεων με την αρχική υλοποίηση του AES-128 εξάγουμε τα ακόλουθα συμπεράσματα ως προς την απόδοση και την χρήση διαθέσιμων πόρων του συστήματος:



Resources Utilization

	Available	No Optimization	Function Pipeline	Function Dataflow
BRAM_18K	720	0	0	0
DSP48E	1152	0	0	0
FF	290880	2051	2169	2233
LUT (Look-Up Tables)	145440	6885	5892	7587
URAM	0	0	0	0



Παρατηρείται πως η εφαρμογή της Directive βελτιστοποίησης Pipeline στην συνάρτηση AddRoundKey αν και συνετέλεσε στη μείωση των κύκλων που απαιτούνται για την υλοποίηση του συστήματος δεν βοήθησε στην αύξηση της απόδοσης (Throughput) διατηρώντας σταθερή την μέγιστη συχνότητα λειτουργίας(Fmax = 161.54 MHz) ενώ παράλληλα αύξησε την κατανάλωση πόρων του υλικού.

4.3.3 Βελτιστοποίηση των συναρτήσεων MixColumns & ShiftRows

Εφαρμόζοντας την τεχνική Βελτιστοποίησης Function Pipeline στις συναρτήσεις MixColumns() και ShiftRows() παρατηρείται ότι δεν προέκυψε κάποια βελτιστοποίηση στο σχεδιασμό του συστήματος:

1) Mix Columns() Pipeline

Synthesis Report for 'aes_ctr128_axis'

General Information

Date: Mon Jun 12 14:56:49 2023
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: hls_prj
Solution: MixColumns
Product family: kintexu
Target device: xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.190 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
488	488	4.880 us	4.880 us	488	488	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_152	aes_ecb128	485	485	4.850 us	4.850 us	485	485	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	1082	6628	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	969	-	-
Total	0	0	2051	6885	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	~0	4	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_152	aes_ecb128	0	0	1082	6628	0
Total	1	0	0	1082	6628	0

DSP48E

N/A

Memory

N/A

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_216_p2	+	0	0	39	32	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	2	0	0	167	160	129

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	27	5	1	5
ap_phi_mux_v2_V_phi_fu_145_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_131	9	2	1	2
v2_V_reg_142	9	2	32	64
v2_i_i_i_i_0184_i_reg_122	9	2	32	64
Total	90	19	132	267

Register

Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	4	0	4	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_310	1	0	1	0
grp_aes_ecb128_fu_152_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Result_s_15_reg_330	128	0	128	0
p_Result_s_reg_325	128	0	128	0
p_Val2_s_reg_298	384	0	384	0
stateIn_last_V_tmp_reg_304	1	0	1	0
storemerge_i_reg_131	1	0	1	0
trunc_ln647_reg_335	128	0	128	0
v2_V_reg_142	32	0	32	0
v2_i_i_i_i_0184_i_reg_122	32	0	32	0
Total	969	0	969	0

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

2) ShiftRows() Pipeline

Synthesis Report for 'aes_ctr128_axis'

General Information

Date: Mon Jun 12 14:44:23 2023
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: hls_prj
Solution: ShiftRows
Product family: kintexu
Target device: xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.190 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
488	488	4.880 us	4.880 us	488	488	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_152	aes_ecb128	485	485	4.850 us	4.850 us	485	485	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	1082	6628	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	969	-	-
Total	0	0	2051	6885	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	~0	4	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_152	aes_ecb128	0	0	1082	6628	0
Total	1	0	0	1082	6628	0

DSP48E

N/A

Memory

N/A

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_216_p2	+	0	0	39	32	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	2	0	0	167	160	129

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	27	5	1	5
ap_phi_mux_v2_V_phi_fu_145_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_131	9	2	1	2
v2_V_reg_142	9	2	32	64
v2_i_i_i_i_0184_i_reg_122	9	2	32	64
Total	90	19	132	267

Register

Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	4	0	4	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_310	1	0	1	0
grp_aes_ecb128_fu_152_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Result_s_15_reg_330	128	0	128	0
p_Result_s_reg_325	128	0	128	0
p_Val2_s_reg_298	384	0	384	0
stateIn_last_V_tmp_reg_304	1	0	1	0
storemerge_i_reg_131	1	0	1	0
trunc_ln647_reg_335	128	0	128	0
v2_V_reg_142	32	0	32	0
v2_i_i_i_i_0184_i_reg_122	32	0	32	0
Total	969	0	969	0

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

Κάνοντας την σύγκριση των δυο βελτιστοποιημένων υλοποιήσεων με την αρχική υλοποίηση:

Total Latency

	No Optimization	MixColumns Funct.Pipeline	ShiftRows Funct.Pipeline
Latency (cycles)	488	488	488
Latency (absolute)	4,880 μs	4,880 μs	4,880 μs
Interval (cycles)	488	488	488

Δεν προκύπτει Βελτιστοποίηση με χρήση της Pipeline Directive

$$Throughput = \frac{128\ bits}{488 * 6{,}19ns} \rightarrow Throughput = 42{,}4\ Mbps$$

Resources Utilization

	Available	No Optimization	MixColumns Funct.Pipeline	ShiftRows Funct.Pipeline
BRAM_18K	720	0	0	0
DSP48E	1152	0	0	0
FF	290880	2051	2051	2051
LUT (Look-Up Tables)	145440	6885	6885	6885
URAM	0	0	0	0

Το γεγονός της μη ύπαρξης αλλαγών ως προς την απόδοση και τους χρησιμοποιούμενους πόρους κατά την εφαρμογή τεχνικών βελτιστοποίησης στις συναρτήσεις MixColumns και ShiftRows οφείλεται στη δομή του εκτελέσιμου κώδικα των συναρτήσεων που είναι απλοποιημένη χωρίς εσωτερικούς βρόχους, συνεπώς οι Directives δεν μπορούν να προσφέρουν κάποια βελτιστοποίηση στην υλοποίηση.

4.3.4 Βελτιστοποίηση του AES-128 ECB (electronic codebook)

Για λόγους σύγκρισης της απόδοσης και της αποτελεσματικότητας του σχεδιασμού μεταξύ παραλλαγών του AES υλοποιήθηκε και ο AES-128 ECB Encryptor, στον οποίο και εφαρμόστηκαν οι βελτιστοποιήσεις: 1) Dataflow, 2) Function Pipeline, 3) Loop Pipeline και 4) Loop Unroll. Ο AES-128 ECB Encryptor αποτελεί το απλούστερο (και πλέον ξεπερασμένο) mode κρυπτογράφησης του AES. Το αρχικό μήνυμα χωρίζεται σε μπλοκ και κάθε μπλοκ κρυπτογραφείται ξεχωριστά. Το μειονέκτημα αυτής της μεθόδου είναι η έλλειψη διάχυσης. Επειδή ο ECB κρυπτογραφεί πανομοιότυπα μπλοκ plaintext σε πανομοιότυπα μπλοκ ciphertext, δεν κρύβει αποτελεσματικά τα μοτίβα των δεδομένων. Η λειτουργία ECB μπορεί επίσης να καταστήσει τα πρωτόκολλα χωρίς προστασία ακεραιότητας ακόμη πιο ευάλωτα σε επιθέσεις αναπαραγωγής, καθώς κάθε μπλοκ αποκρυπτογραφείται με τον ίδιο ακριβώς τρόπο. Για τον λόγο αυτό ο ECB δεν συνιστάται για χρήση σε κρυπτογραφικά πρωτόκολλα.

1) Dataflow

Synthesis Report for 'aes_ctr128_axis'

General Information

Date:	Mon Jun 12 16:22:49 2023
Version:	2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project:	hls_prj
Solution:	AES_ECB
Product family:	kintexu
Target device:	xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.190 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
489	489	4.890 us	4.890 us	489	489	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_158	aes_ecb128	485	485	4.850 us	4.850 us	486	486	dataflow

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	171	-
FIFO	-	-	-	-	-
Instance	0	-	1086	6667	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	96	-
Register	-	-	972	-	-
Total	0	0	2058	6934	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	~0	4	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_158	aes_ecb128	0	0	1086	6667	0
Total	1	0	0	1086	6667	0

DSP48E

Memory

FIFO

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_223_p2	+	0	0	39	32	1
ap_sync_grp_aes_ecb128_fu_158_ap_done	or	0	0	2	1	1
ap_sync_grp_aes_ecb128_fu_158_ap_ready	or	0	0	2	1	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	4	0	0	171	162	131

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	33	6	1	6
ap_phi_mux_v2_V_phi_fu_151_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_137	9	2	1	2
v2_V_reg_148	9	2	32	64
v2_i_i_i_i_0184_i_reg_128	9	2	32	64
Total	96	20	132	268

Register

Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	5	0	5	0
ap_sync_reg_grp_aes_ecb128_fu_158_ap_done	1	0	1	0
ap_sync_reg_grp_aes_ecb128_fu_158_ap_ready	1	0	1	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_331	1	0	1	0
grp_aes_ecb128_fu_158_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
op2_V_assign_fu_106	128	0	128	0
p_Result_s_13_reg_351	128	0	128	0
p_Result_s_reg_346	128	0	128	0
p_Val2_s_reg_320	384	0	384	0
stateOut_last_V_tmp_reg_325	1	0	1	0
storemerge_i_reg_137	1	0	1	0
v2_V_reg_148	32	0	32	0
v2_i_i_i_i_0184_i_reg_128	32	0	32	0
Total	972	0	972	0

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

2) Function Pipeline

Synthesis Report for 'aes_ctr128_axis'

General Information

Date: Mon Jun 12 16:05:38 2023
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: hls_prj
Solution: AES_ECB
Product family: kintexu
Target device: xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.706 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
13	13	0.130 us	0.130 us	13	13	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_152	aes_ecb128	11	11	0.110 us	0.110 us	1	1	function

Loop

N/A

Utilization Estimates

Close

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	4515	18066	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	125	-
Register	-	-	723	-	-
Total	0	0	5238	18358	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	1	12	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_152	aes_ecb128	0	0	4515	18066	0
Total	1	0	0	4515	18066	0

DSP48E

N/A

Memory

N/A

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_216_p2	+	0	0	39	32	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	2	0	0	167	160	129

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	62	15	1	15
ap_phi_mux_v2_V_phi_fu_145_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_131	9	2	1	2
v2_V_reg_142	9	2	32	64
v2_i_i_i_i_0184_i_reg_122	9	2	32	64
Total	125	29	132	277

Register

Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	14	0	14	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_310	1	0	1	0
grp_aes_ecb128_fu_152_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Val2_s_reg_298	384	0	384	0
stateIn_last_V_tmp_reg_304	1	0	1	0
storemerge_i_reg_131	1	0	1	0
trunc_ln647_reg_325	128	0	128	0
v2_V_reg_142	32	0	32	0
v2_i_i_i_i_0184_i_reg_122	32	0	32	0
Total	723	0	723	0

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

3) Loop Pipeline

Synthesis Report for 'aes_ctr128_axis'

General Information

Date: Mon Jun 12 15:54:15 2023
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: hls_prj
Solution: AES_ECB
Product family: kintexu
Target device: xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.808 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
16	16	0.160 us	0.160 us	16	16	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_152	aes_ecb128	13	13	0.130 us	0.130 us	13	13	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	660	2248	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	969	-	-
Total	0	0	1629	2505	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	~0	1	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_152	aes_ecb128	0	0	660	2248	0
Total	1	0	0	660	2248	0

DSP48E

N/A

Memory

N/A

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_216_p2	+	0	0	39	32	1
stateOut_TDATA_int	xor	0	0	128	128	128
Total	2	0	0	167	160	129

Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	27	5	1	5
ap_phi_mux_v2_V_phi_fu_145_p4	9	2	32	64
count_V	9	2	32	64
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_131	9	2	1	2
v2_V_reg_142	9	2	32	64
v2_i_i_i_i_0184_i_reg_122	9	2	32	64
Total	90	19	132	267

Register

Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	4	0	4	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_310	1	0	1	0
grp_aes_ecb128_fu_152_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Result_s_11_reg_330	128	0	128	0
p_Result_s_reg_325	128	0	128	0
p_Val2_s_reg_298	384	0	384	0
stateIn_last_V_tmp_reg_304	1	0	1	0
storemerge_i_reg_131	1	0	1	0
trunc_ln647_reg_335	128	0	128	0
v2_V_reg_142	32	0	32	0
v2_i_i_i_i_0184_i_reg_122	32	0	32	0
Total	969	0	969	0

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

4) Loop Unroll

General Information

Date: Mon Jun 12 15:43:03 2023
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: hls_prj
Solution: AES_ECB
Product family: kintexu
Target device: xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.964 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
363	363	3.630 us	3.630 us	363	363	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_subBytes_fu_180	subBytes	34	34	0.340 us	0.340 us	34	34	none
grp_subBytes_fu_217	subBytes	34	34	0.340 us	0.340 us	34	34	none
grp_keyExpEngine_fu_254	keyExpEngine	11	11	0.110 us	0.110 us	11	11	none
grp_encMixCols_fu_282	encMixCols	0	0	0 ns	0 ns	0	0	none
op2_V_assign_5_0_1_i_encMixCols_fu_287	encMixCols	0	0	0 ns	0 ns	0	0	none
grp_shiftRows_fu_292	shiftRows	0	0	0 ns	0 ns	0	0	none
op2_V_assign_4_0_1_i_shiftRows_fu_299	shiftRows	0	0	0 ns	0 ns	0	0	none

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	683	-
FIFO	-	-	-	-	-
Instance	0	-	1870	10600	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	211	-
Register	-	-	1364	-	-
Total	0	0	3234	11494	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	1	7	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_encMixCols_fu_282	encMixCols	0	0	0	704	0
op2_V_assign_5_0_1_i_encMixCols_fu_287	encMixCols	0	0	0	704	0
grp_keyExpEngine_fu_254	keyExpEngine	0	0	540	1320	0
grp_shiftRows_fu_292	shiftRows	0	0	0	0	0
op2_V_assign_4_0_1_i_shiftRows_fu_299	shiftRows	0	0	0	0	0
grp_subBytes_fu_180	subBytes	0	0	665	3936	0
grp_subBytes_fu_217	subBytes	0	0	665	3936	0
Total	7	0	0	1870	10600	0

DSP48E

N/A

Memory

N/A

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_none	aes_ctr128_axis	return value
ap_rst_n	in	1	ap_ctrl_none	aes_ctr128_axis	return value
stateIn_TDATA	in	384	axis	stateIn_data_V	pointer
stateIn_TVALID	in	1	axis	stateIn_last_V	pointer
stateIn_TREADY	out	1	axis	stateIn_last_V	pointer
stateIn_TLAST	in	1	axis	stateIn_last_V	pointer
stateOut_TDATA	out	128	axis	stateOut_data_V	pointer
stateOut_TVALID	out	1	axis	stateOut_last_V	pointer
stateOut_TREADY	in	1	axis	stateOut_last_V	pointer
stateOut_TLAST	out	1	axis	stateOut_last_V	pointer

FIFO

N/A

Expression

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
add_ln700_fu_339_p2	+	0	0	39	32	1
ap_block_state10_on_subcall_done	or	0	0	2	1	1
ap_block_state3_on_subcall_done	or	0	0	2	1	1
grp_fu_306_p2	xor	0	0	128	128	128
stateOut_TDATA_int	xor	0	0	128	128	128
xor_ln1357_52_fu_419_p2	xor	0	0	128	128	128
xor_ln1357_fu_412_p2	xor	0	0	128	128	128
xor_ln209_fu_429_p2	xor	0	0	128	128	128
Total		8	0	0	683	674	643

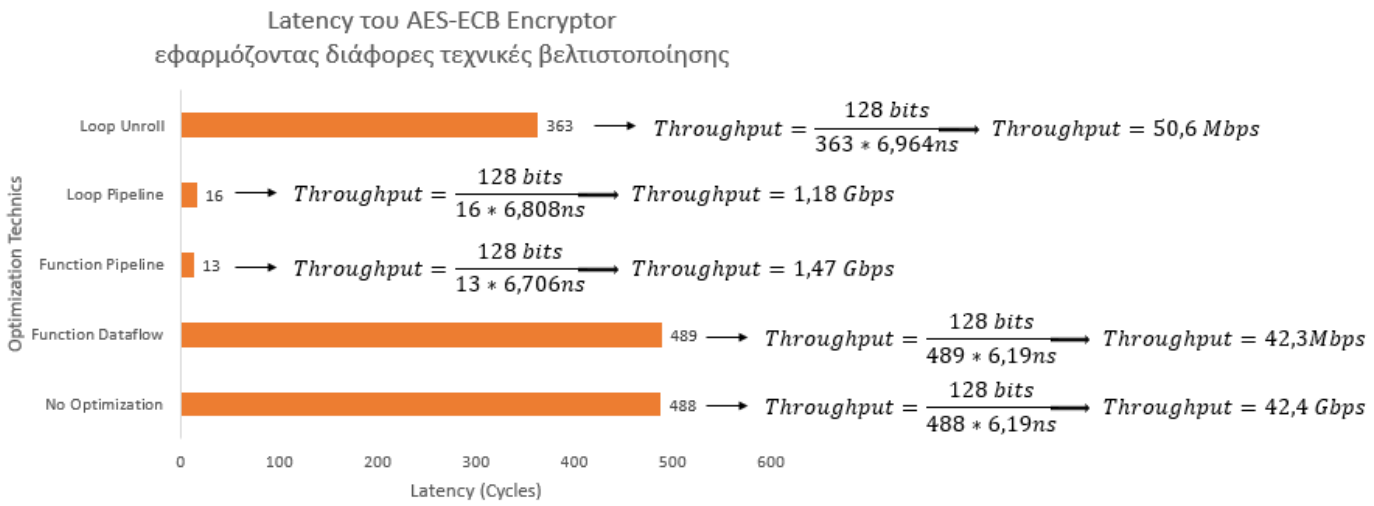
Multiplexer

Name	LUT	Input Size	Bits	Total Bits
ap_NS_fsm	59	14	1	14
ap_phi_mux_v2_V_phi_fu_173_p4	9	2	32	64
count_V	9	2	32	64
grp_keyExpEngine_fu_254_keyState_V	15	3	128	384
grp_keyExpEngine_fu_254_rConAddr_V	53	12	4	48
grp_subBytes_fu_180_stateIn_V	21	4	128	512
stateIn_TDATA_blk_n	9	2	1	2
stateOut_TDATA_blk_n	9	2	1	2
storemerge_i_reg_159	9	2	1	2
v2_V_reg_170	9	2	32	64
v2_i_i_i_i_0184_i_reg_150	9	2	32	64
Total	211	47	392	1220

Register

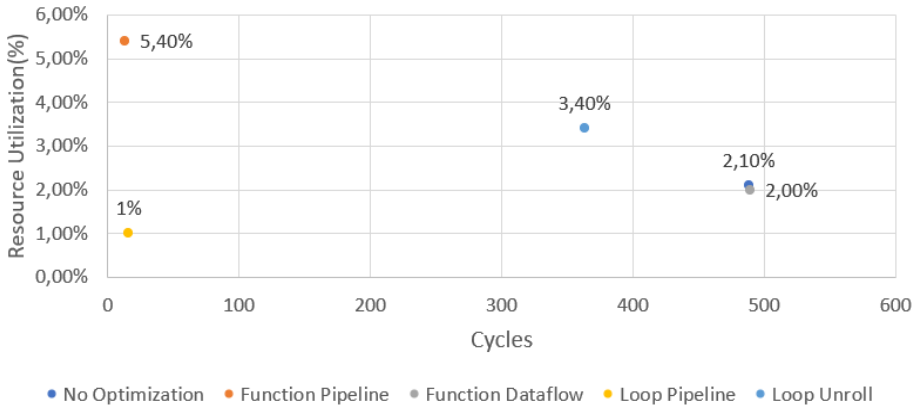
Name	FF	LUT	Bits	Const Bits
ap_CS_fsm	13	0	13	0
count_V	32	0	32	0
firstBlkInt	1	0	1	0
firstBlkInt_load_reg_454	1	0	1	0
grp_keyExpEngine_fu_254_ap_start_reg	1	0	1	0
grp_subBytes_fu_180_ap_start_reg	1	0	1	0
grp_subBytes_fu_217_ap_start_reg	1	0	1	0
nonce_V	96	0	96	0
p_Result_s_14_reg_474	128	0	128	0
p_Result_s_reg_469	128	0	128	0
p_Val2_s_reg_442	384	0	384	0
reg_313	128	0	128	0
reg_318	128	0	128	0
stateIn_last_V_tmp_reg_448	1	0	1	0
storemerge_i_reg_159	1	0	1	0
v2_V_reg_170	32	0	32	0
v2_i_i_i_i_0184_i_reg_150	32	0	32	0
xor_ln1357_52_reg_484	128	0	128	0
xor_ln1357_reg_479	128	0	128	0
Total	1364	0	1364	0

Από την σύγκριση των συνθέσεων που προέκυψαν από την υλοποίηση του AES-128 ECB και την εφαρμογή των βελτιστοποιήσεων με την αρχική υλοποίηση του AES-128 εξάγουμε τα ακόλουθα συμπεράσματα ως προς την απόδοση και την χρήση διαθέσιμων πόρων του συστήματος



Resources Utilization

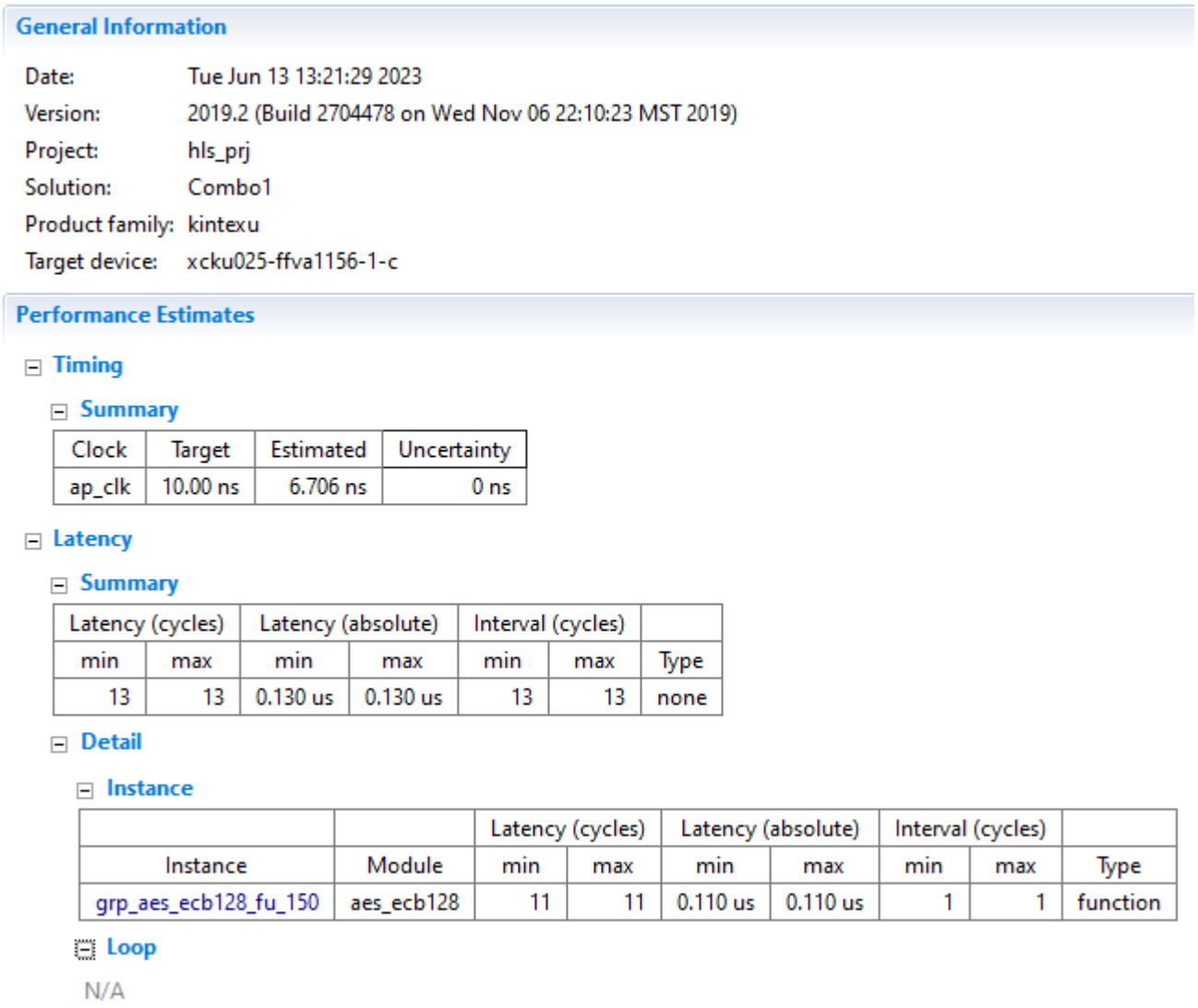
	Available	No Optimization	Function Pipeline	Function Dataflow	Loop Pipeline	Loop Unroll
BRAM_18K	720	0	0	0	0	0
DSP48E	1152	0	0	0	0	0
FF	290880	2051	5238	2058	1629	3234
LUT (Look-Up Tables)	145440	6885	18358	6934	2505	11494
URAM	0	0	0	0	0	0



4.4 Εφαρμογή Συνδυασμών Βελτιστοποιήσεων

Ως τελευταίο βήμα για την εξαγωγή συμπερασμάτων για την βέλτιστη υλοποίηση του αλγορίθμου AES-128 με σκοπό τη μείωση της καθυστέρησης και των χρησιμοποιούμενων πόρων εφαρμόστηκαν δυο συνδυασμοί τεχνικών βελτιστοποιήσεων στις συναρτήσεις του αλγορίθμου.

Τεχνική 1	
Συνάρτηση	**Τεχνική Βελτιστοποίησης**
Mix Columns	Function Pipeline
AddRoundKey	Data_Pack
AddRoundKey	Inline
ShiftRows	Function Pipeline
SubBytes	Loop Unroll
SubBytes	Pipeline
SubWord	Loop Unroll
SubWord	Pipeline



General Information

Date: Tue Jun 13 13:21:29 2023
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: hls_prj
Solution: Combo1
Product family: kintexu
Target device: xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.706 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
13	13	0.130 us	0.130 us	13	13	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_150	aes_ecb128	11	11	0.110 us	0.110 us	1	1	function

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	0	-	3339	17949	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	125	-
Register	-	-	723	-	-
Total	0	0	4062	18241	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	1	12	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_150	aes_ecb128	0	0	3339	17949	0
Total	1	0	0	3339	17949	0

Τεχνική 2	
Συνάρτηση	**Τεχνική Βελτιστοποίησης**
Mix Columns	Function Pipeline
AddRoundKey	Inline
ShiftRows	Pipeline
ShiftRows	Inline
SubBytes	Inline
SubBytes	Loop_Flatten
SubBytes	Loop Pipeline
SubWord	Inline

General Information

Date: Tue Jun 13 14:02:56 2023
Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project: hls_prj
Solution: Combo2
Product family: kintexu
Target device: xcku025-ffva1156-1-c

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.190 ns	0 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
211	211	2.110 us	2.110 us	211	211	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_aes_ecb128_fu_150	aes_ecb128	208	208	2.080 us	2.080 us	208	208	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	167	-
FIFO	-	-	-	-	-
Instance	-	-	5341	43031	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	90	-
Register	-	-	969	-	-
Total	0	0	6310	43288	0
Available	720	1152	290880	145440	0
Utilization (%)	0	0	2	29	0

Detail

Instance

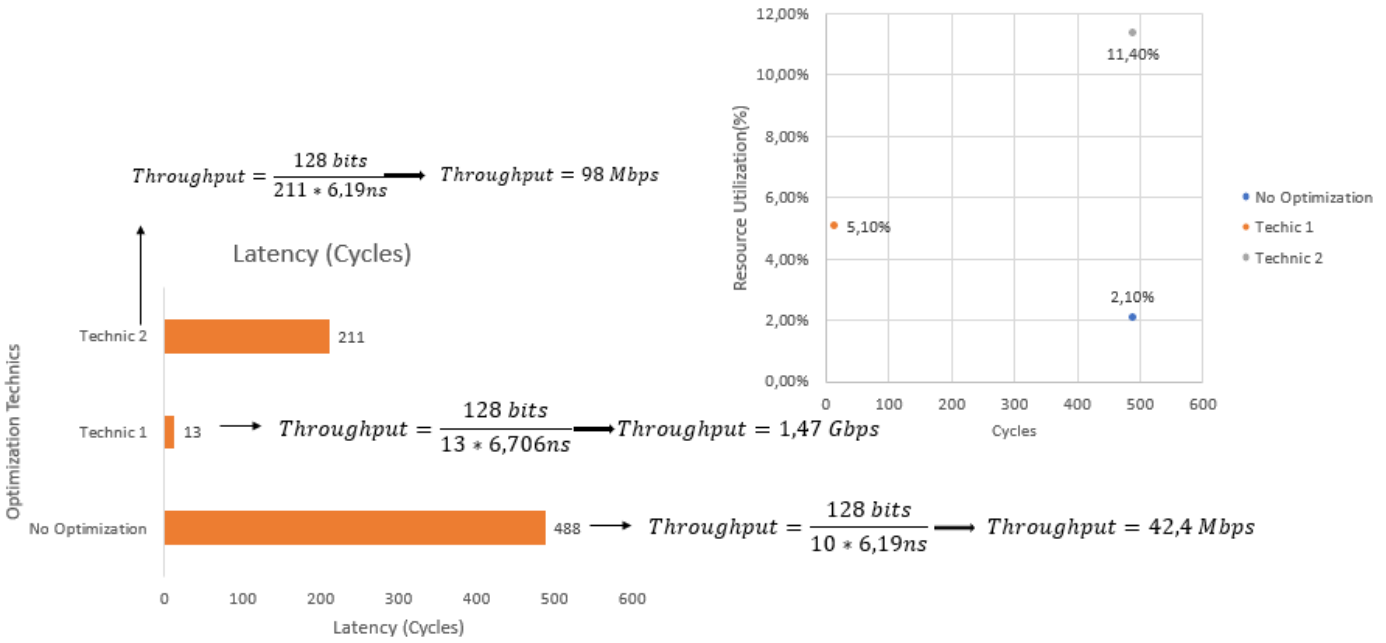
Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_aes_ecb128_fu_150	aes_ecb128	0	0	5341	43031	0
Total	1	0	0	5341	43031	0

Εφαρμόζοντας τις ανωτέρω συνδυαστικές τεχνικές βελτιστοποιήσεων το εργαλείο Vivado HLS μέσω της Σύνθεσης που παρείχε οδήγησε στα ακόλουθα συμπεράσματα για την υλοποίηση του αλγορίθμου:

Τεχνικη1: Pipeline SubBytes
Loop Unroll SubBytes
Pipeline ShiftRows
Inline AddRoundKey
Data_Pack AddRoundKey
Pipeline MixColumns

	Latency (Cycles)	Resource Utilization(%)				Επιτάχυνση	Αύξηση Κατανάλωσης Πόρων
		BRAM	DSP	FF	LUT		
No Optimization	488	0%	0%	0,7%	4,7%	-	-
Τεχνικη1	13	0%	0%	1,4%	12,5%	37,5x	2,6x
Τεχνικη2	211	0%	0%	2,2%	29,8%	2,3x	5,6x

Τεχνικη2: Loop Pipeline SubBytes
Loop_Flatten SubBytes
Inline SubBytes
Pipeline ShiftRows
Inline AddRoundKey
Pipeline MixColumns



Από την εφαρμογή των συνδυαστικών αυτών τεχνικών (και ειδικά στην τεχνική1) είναι εμφανής η μείωση των απαιτούμενων κύκλων υλοποίησης του αλγορίθμου. Ωστόσο, παρόλο που υπάρχει μια μικρή μείωση της απόδοσης λόγω αύξησης της περιόδου κάθε κύκλου ρολογιού και σχεδόν διπλασιασμός των χρησιμοποιούμενων πόρων του υλικού, επιτυγχάνεται η επιτάχυνση της υλοποίησης του συστήματος κατά 37,5 φορές σε σχέση με τη μη βελτιστοποιημένη υλοποίηση.

Κεφάλαιο 5

Συμπεράσματα

Στην παρούσα διπλωματική εργασία, στα πλαίσια της μελέτης και της ανάπτυξης υποσυστημάτων κρυπτογραφίας σε υλικό, παρουσιάστηκε η υλοποίηση του προτύπου κρυπτογράφησης Advanced Encryption Standard (AES) με μέγεθος κλειδιού κρυπτογράφησης/αποκρυπτογράφησης 128 bit σε πλατφόρμα υλικού Xilinx της οικογενείας Kintex-UltraScale. Η διαδικασία που ακολουθήθηκε αρχικά από τη συγγραφή του κώδικα και τον έλεγχο της ορθής λειτουργίας του αλγορίθμου μέχρι το στάδιο της Σύνθεσης Υψηλού Επιπέδου και ανάλυσης των παραγόμενων μετρήσεων, παρέχει τη δυνατότητα πληρέστερης και πιο κοντά στα σχεδιαστικά πρότυπα υλοποίηση του αλγορίθμου καθώς επίσης και την προοπτική βελτιστοποίησης τμημάτων του αλγορίθμου μεμονωμένα ή και συνδυαστικά με σκοπό την βελτίωση της απόδοσης του.

Το σχεδιαστικό εργαλείο Vivado-HLS 2019.2 του Xilinx που χρησιμοποιήθηκε για τους σκοπούς της εργασίας, προσφέρει ποικίλες δυνατότητες στον σχεδιαστή όπως η αυτοματοποίηση της απαιτούμενης διαδικασίας για την σύνθεση σε επίπεδο μεταφοράς καταχωρητών (RTL) και ο έλεγχος ορθότητας της λειτουργίας του υλοποιήσιμου συστήματος μέσω της δημιουργίας ενός αρχείου testbench. Ακόμα, προτέρημα των εργαλείων Σύνθεσης Υψηλού Επιπέδου αποτελεί η δυνατότητα εφαρμογής τεχνικών βελτιστοποίησης στον πηγαίο κώδικα του αλγορίθμου για ταχύτατες υλοποιήσεις, υψηλότερες αποδόσεις και μείωση της χρήσης των διαθέσιμων πόρων του υλικού.

Τέλος, και όπως αποτέλεσε στόχο της παρούσης εργασίας, από τα πειραματικά δεδομένα και τα εξαγόμενα αποτελέσματα κατά την υλοποίηση του προτύπου AES τόσο σε γλώσσα περιγραφής υλικού (VHDL), όσο και σε σύνθεση υψηλού επιπέδου (HLS), διαπιστώθηκαν κάποια προτερήματα της υψηλής σύνθεσης έναντι της λογικής. Πρωτίστως, παρατηρήθηκε επιτάχυνση του απαιτούμενου χρόνου για τη σύνθεση του εκτελέσιμου κώδικα αλλά και για την τελική δημιουργία του RTL σχεδίου, γεγονός που αποτελεί ευχέρεια για τον σχεδιαστή, ενώ παράλληλα, και όσον αφορά την υλοποίηση του αλγορίθμου, παρατηρήθηκε βελτίωση της καθυστέρησης και συνεπώς αύξηση της απόδοσης του συστήματος καθώς και μείωση του χρησιμοποιούμενων πόρων από τους συνολικά διαθέσιμους της πλατφόρμας υλικού. Επίσης, η δυνατότητα των εργαλείων HLS για την εφαρμογή τεχνικών βελτιστοποίησης του συστήματος, που δεν παρέχεται από τις γλώσσες σχεδιασμού υλικού (HDL), επεμβαίνοντας στην μικρο-αρχιτεκτονική του συστήματος, συνέβαλε σε καθοριστική βελτίωση του σχεδιασμού.

Μελλοντικά, όπως διαπιστώθηκε και από την εφαρμογή συνδυαστικών τεχνικών βελτιστοποίησης στο πρότυπο κρυπτογράφησης AES, θα μπορούσαν να εφαρμοστούν περαιτέρω συνδυασμοί βελτιστοποιήσεων στις κύριες συναρτήσεις του αλγορίθμου ή ακόμα και πιο στοχευμένα στο S-Box που συντελεί στην αντικατάσταση των bit του αρχικού κειμένου κατά την έναρξη κάθε κύκλου της κρυπτογράφησης.

Βιβλιογραφία

[1] Vivado Design Suite User Guide - High-Level Synthesis. UG902. Rev. 2020.1. Xilinx.

[2] High-level synthesis — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/High-level_synthesis

[3] UltraFast Vivado HLS Methodology Guide. UG1197. Rev. 2020.1. Xilinx.

[4] Vivado HLS Optimization Methodology Guide. UG1270. Rev. 2017.4. Xilinx.

[5] High-Level Synthesis Optimization of AES-128/192/256 Encryption Algorithms. Daoud, Luka; Hussein, Fady; and Rafla, Nader. (2019).

[6] Advanced Encryption Standard — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[7] Design of AES Algorithm for 128/192/256 Key Length in FPGA. Pravin V. Kinge , S.J. Honale , C.M. Bobade

[8] Implementation of the AES Algorithm for a Reconfigurable, Bit Serial, Fully Pipelined Architecture. Raphael Weber & Achim Rettberg

[9] LOW VS HIGH LEVEL PROGRAMMING FOR FPGA. Jan Marjanovic

[10] A Comparative Study between RTL and HLS for Image Processing Applications with FPGAs.Muhsin Gurel

[11] Design and implementation of power and area optimized AES architecture on FPGA for IoT application. Rajasekar P., Mangalam H.

[12] Optimizing AES Implementation for High-Speed Embedded Application. Atnafu, Dessalegn

[13] Vitis High-Level Synthesis User Guide (UG1399). Xilinx

[14] Optimization of AES Encryption Algorithm with SBox. Dr. J. Abdul Jaleel , Anu Assis and Sherla A.