

# **ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ**

## **ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ**

**ΤΟΜΕΑΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ  
ΕΡΓΑΣΤΗΡΙΟ ΣΧΕΔΙΑΣΜΟΥ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ  
ΜΕΓΑΛΗΣ ΚΛΙΜΑΚΑΣ (VLSI DESIGN)**

### **ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

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

### **ΔΑΚΟΥΡΟΥ ΣΤΕΦΑΝΙΑ του ΝΙΚΟΛΑΟΥ**

Αριθμός Μητρώου: 5311

Θέμα:

**«ΥΛΟΠΟΙΗΣΗ ΚΡΥΠΤΟ-ΕΠΕΞΕΡΓΑΣΤΙΚΗΣ ΠΛΑΤΦΟΡΜΑΣ  
ΓΙΑ ΚΡΥΠΤΟΓΡΑΦΗΣΗ ΜΗΝΥΜΑΤΩΝ ΣΤΟ ΠΡΟΤΥΠΟ  
GALOIS / COUNTER MODE (GCM)»**

### **ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΓΚΟΥΤΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ**

Αριθμός Διπλωματικής:

Πάτρα, Ιούλιος 2009



## **ΠΙΣΤΟΠΟΙΗΣΗ**

Πιστοποιείται ότι η διπλωματική εργασία με θέμα :

### **«ΥΛΟΠΟΙΗΣΗ ΚΡΥΠΤΟ-ΕΠΕΞΕΡΓΑΣΤΙΚΗΣ ΠΛΑΤΦΟΡΜΑΣ ΓΙΑ ΚΡΥΠΤΟΓΡΑΦΗΣΗ ΜΗΝΥΜΑΤΩΝ ΣΤΟ ΠΡΟΤΥΠΟ GALOIS / COUNTER MODE (GCM)»**

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

### **ΔΑΚΟΥΡΟΥ ΣΤΕΦΑΝΙΑ του ΝΙΚΟΛΑΟΥ**

Αριθμός Μητρώου: 5311

παρουσιάστηκε δημόσια και εξετάστηκε στο τμήμα Ηλεκτρολόγων  
Μηχανικών και Τεχνολογίας Υπολογιστών  
στις  
06 / 07 / 2009

Ο Επιβλέπων

Ο Διευθυντής του τομέα

Γκούτης Κωνσταντίνος  
Καθηγητής

Γκούτης Κωνσταντίνος  
Καθηγητής



## Abstract

Το τέταρτο recommendation για το συμμετρικό block cipher τρόπο λειτουργίας SP800-38D, Galois/Counter Mode of Operation (GCM) αναπτύχθηκε από τους David A McGrew και John Viega. Ο GCM χρησιμοποιεί έναν εγκεκριμένο συμμετρικού κλειδιού block cipher με block size των 128 bits και μια καθολική hashing λειτουργία ορισμένα σ' ένα δυαδικό Galois Field με σκοπό να προσφέρει εμπιστευτικότητα και πιστοποίηση των δεδομένων. Δημιουργήθηκε ειδικά για να υποστηρίζει πολύ υψηλά data rates καθώς μπορεί να αξιοποιήσει τις τεχνικές της διοχέτευσης και του παραλληλισμού των διεργασιών.

Πριν από τον GCM, το SP800-38A προσέφερε μόνο εμπιστευτικότητα ενώ το SP800-38B μόνο πιστοποίηση. Το SP800-38C παρέχει εμπιστευτικότητα χρησιμοποιώντας τον counter mode καθώς και πιστοποίηση. Ωστόσο, ο μηχανισμός της πιστοποίησης στο SP800-38C δεν μπορεί να εκτελείται παράλληλα με την εμπιστευτικότητα και έτσι επιβραδύνει την απόδοση του cipher. Έτσι, κανένα από τα τρία προηγούμενα recommendations δεν ήταν κατάλληλα για high speed networks και computer system εφαρμογές.

Με την είσοδο του GCM, πιστοποιημένη κρυπτογράφηση σε data rates αρκετών Gbps είναι πλέον εφικτή, επιτρέποντας υψηλού επιπέδου πιστοποιημένη κρυπτογράφηση σε συστήματα που προηγουμένως δεν μπορούσαν να προστατευτούν ολοκληρωτικά.

Σε αυτήν την εργασία παρουσιάζεται μια πλήρως pipelined αρχιτεκτονική σε hardware του μηχανισμού εμπιστευτικότητας του AES-GCM που θα περιέχει μια pipelined-AES δομή με pipelined Key-schedule δομή για τα επεκταμένα κλειδιά. Τα στοιχεία αυτά συνδυάζονται με μια iterative-AES δομή και μια GHASH δομή, συνδέονται μεταξύ τους με ένα control unit και συνθέτουν ένα ολοκληρωμένο σχεδιασμό πλήρως pipelined και parallelizable αρχιτεκτονικής του AES-GCM. Τα αποτελέσματα αυτής της εργασίας θα δείξουν ότι τα round transformations της εμπιστευτικότητας και της hash λειτουργίας του μηχανισμού πιστοποίησης μπορούν να συνεργαστούν πολύ αποτελεσματικά και να εκτελούνται παράλληλα εντός αυτής της pipelined αρχιτεκτονικής, όπου ο GCM εκτελείται υπό τον συμμετρικό block cipher AES πάνω σ' ένα FPGA.

Αντικείμενο αυτής της διπλωματικής εργασίας είναι να δείξει και να αναλύσει την πρακτική εφαρμογή του GCM καθώς και τις απαιτήσεις σε επιφάνια (area) για υλοποίηση σε πραγματική hardware platform. Η υψηλής ταχύτητας FPGA-based αρχιτεκτονική που παρουσιάζεται είναι κατάλληλη για high speed embedded applications. Στο υπόλοιπο της διπλωματικής το AES-GCM θα αναφέρεται στο GCM μαζί με τον AES ως το θεμελιώδη συμμετρικό block cipher.

## Περίληψη

Η παρούσα διπλωματική προτείνει μια hardware υλοποίηση για κρυπτογράφηση μηνυμάτων βασισμένη στο πρότυπο ασφαλείας Galois/Counter Mode (GCM). Ο αλγόριθμος κρυπτογράφησης Galois/Counter Mode (GCM) εκδόθηκε από τον οργανισμό National Institute of Standards and Technology (NIST) τον Νοέμβριο του 2007. Σε συνεργασία με τον μηχανισμό πιστοποίησης μηνυμάτων, υλοποιείται το συνολικό πρότυπο GCM για online λειτουργία.

Στο *Κεφάλαιο 1*, αρχικά γίνεται μια σύντομη ιστορική αναδρομή στον τομέα της κρυπτογραφίας. Στην συνέχεια παρουσιάζονται οι λόγοι που οδήγησαν στην δημιουργία αυτού του προτύπου ασφαλείας. Ακολουθεί μια σύντομη εισαγωγή στον GCM και στους τρόπους με τους οποίους αυτός δουλεύει. Στο τέλος του κεφαλαίου αναφέρονται οι πρακτικές εφαρμογές που έχει η χρήση του G.

Στο *Κεφάλαιο 2*, αρχικά αναφέρονται οι συμβολισμοί και οι βασικές έννοιες που χρησιμοποιούνται στην διπλωματική. Στην συνέχεια, δίνονται συνοπτικές πληροφορίες για το μαθηματικό υπόβαθρο που είναι απαραίτητο για την κατανόηση των εσωτερικών μηχανισμών του GCM. Ακολουθεί η ανάλυση των μαθηματικών συνιστώσων του GCM. Επίσης, παρουσιάζεται το υπόβαθρο των τρόπων λειτουργίας (mode of operation) που υιοθετούνται στον GCM. Τέλος, δίνονται συνοπτικά βασικές πληροφορίες της FPGA πλατφόρμας,

Στο *Κεφάλαιο 3*, γίνεται μια ολοκληρωμένη ανάλυση και μελέτη του Advanced Encryption Standard (AES) και πιο συγκεκριμένα της forward cipher function με υποστηριζόμενο κλειδί 128 bit, καθώς αυτή η διεργασία αποτελεί τον βασικό συμμετρικού κλειδιού block cipher που χρησιμοποιείται στον GCM.

Στο *Κεφάλαιο 4*, πραγματοποιείται η ανάλυση του προτύπου GCM βασισμένη στο τέταρτο recommendation του NIST. Παρουσιάζονται όλοι οι επιμέρους αλγόριθμοι που χρησιμοποιούνται καθώς και ο τρόπος με τον οποίον αυτοί αλληλεπιδρούν και συνεργάζονται για την δημιουργία του GCM αλγορίθμου. Στο τέλος του κεφαλαίου, αναφέρονται οι απαιτήσεις για key και IV που πρέπει να ικανοποιούνται για να θεωρείται ο αλγόριθμος κρυπτογραφικά ισχυρός.

Στο *Κεφάλαιο 5*, αρχικά γίνεται μια top down ανάλυση του GCM. Την ανάλυση αυτή συμπληρώνει η δεύτερη ενότητα στην οποία παρουσιάζεται και αναλύεται η προτεινόμενη αρχιτεκτονική για την hardware υλοποίηση. Στην συνέχεια, δίνεται ο σχεδιασμός των βασικών δομικών στοιχείων του GCM. Στην τελευταία ενότητα παρουσιάζεται η συνολική αρχιτεκτονική του GCM με σκοπό την υλοποίηση για high speed απαιτήσεις.

Στο *Κεφάλαιο 6*, δίνεται μια συνοπτική περιγραφή στα γενικότερα θέματα μια γλώσσας περιγραφής υλικού, και των εργαλείων εξομοίωσης και σύνθεσης.

Στο *Κεφάλαιο 7*, δίνονται τα αποτέλεσμα που προέκυψαν από την εξομοίωση του VHDL κώδικα. Ακολουθούν τα αποτελέσματα της σύνθεσης από δύο διαφορετικές FPGA τεχνολογίες. Στη συνέχεια γίνεται εισαγωγή στον τρόπο με τον οποίο ο μηχανισμός πιστοποίησης και κρυπτογράφησης μηνυμάτων συνεργάζονται για την υλοποίηση του GCM και δίνεται ο τρόπος λειτουργίας του GCM μηχανισμού που υλοποιήθηκε. Ακολουθούν τα αποτελέσματα της εξομοίωσης και σύνθεσης σε δύο διαφορετικές τεχνολογίες καθώς και μια σύγκριση μεταξύ των δύο τεχνολογιών που χρησιμοποιήθηκαν για την σύνθεση.

Στο Παράρτημα A παρουσιάζεται ο τρόπος χρήσης του Xilinx ISE που χρησιμοποιήθηκε για την σύνθεση του VHDL κώδικα για την τεχνολογία Virtex-V.

## ΠΕΡΙΕΧΟΜΕΝΑ

<b>ΚΕΦΑΛΑΙΟ 1 .....</b>	<b>11</b>
<b>ΕΙΣΑΓΩΓΗ .....</b>	<b>11</b>
<b>1.1 ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΚΡΥΠΤΟΓΡΑΦΙΑ.....</b>	<b>11</b>
<b>1.2 Η ΑΝΑΓΚΗ ΥΠΑΡΞΗΣ ΤΟΥ GCM .....</b>	<b>16</b>
<b>1.3 ΕΙΣΑΓΩΓΗ ΣΤΟΝ GCM .....</b>	<b>18</b>
<b>1.4 ΧΡΗΣΗ ΤΟΥ GCM.....</b>	<b>19</b>
<b>ΚΕΦΑΛΑΙΟ 2 .....</b>	<b>21</b>
<b>BACKGROUND .....</b>	<b>21</b>
<b>2.1. ΟΡΙΣΜΟΙ, ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ ΚΑΙ ΣΥΜΒΟΛΑ .....</b>	<b>21</b>
<b>2.1.1 Ορισμοί και Συντομογραφίες.....</b>	<b>21</b>
<b>2.1.2 Σύμβολα και Μεταβλητές .....</b>	<b>23</b>
<b>2.2 ΜΑΘΗΜΑΤΙΚΟ ΥΠΟΒΑΘΡΟ .....</b>	<b>23</b>
<b>2.2.1 Εισαγωγή στο πεπερασμένο πεδίο .....</b>	<b>23</b>
<b>2.2.2 Ανάγκη χρήσης πεδίων <math>GF(2^n)</math> στους Αλγόριθμους Κρυπτογράφησης .....</b>	<b>25</b>
<b>2.2.3 Το πεδίο <math>GF(2^n)</math>.....</b>	<b>25</b>
<b>2.2.4. Υπολογιστικοί Παράγοντες .....</b>	<b>26</b>
<b>2.2.5 Πρόσθεση στο <math>GF(2^n)</math> πεδίο .....</b>	<b>26</b>
<b>2.2.6 Πολλαπλασιασμός στο <math>GF(2^n)</math> πεδίο .....</b>	<b>26</b>
<b>2.3 ΜΑΘΗΜΑΤΙΚΕΣ ΣΥΝΙΣΤΩΣΕΣ ΤΟΥ GCM .....</b>	<b>27</b>
<b>2.3.1 Παραδείγματα από τις βασικές πράξεις και λειτουργίες.....</b>	<b>27</b>
<b>2.3.2 Μαθηματικές Συνιστώσες του AES-CTR .....</b>	<b>31</b>
<b>2.4.1 Electronic Codebook Mode (ECB) .....</b>	<b>32</b>
<b>2.4.2 Counter Mode (CTR) .....</b>	<b>34</b>
<b>2.5 FIELD PROGRAMMABLE GATE ARRAYS (FPGA).....</b>	<b>38</b>
<b>2.5.1 Πλεονεκτήματα των FPGAs στις Κρυπτογραφικές Εφαρμογές .....</b>	<b>39</b>
<b>2.5.2 ASIC Προσέγγιση .....</b>	<b>40</b>
<b>2.5.3 FPGA VirtexII xc2v1000 .....</b>	<b>41</b>
<b>2.5.4 Configurable Logic Blocks (CLBs).....</b>	<b>41</b>
<b>2.5.5 Distributed SelectRAM .....</b>	<b>43</b>
<b>2.5.6 Block SelectRAM .....</b>	<b>45</b>
<b>ΚΕΦΑΛΑΙΟ 3 .....</b>	<b>47</b>
<b>SECURITY STANDARD.....</b>	<b>47</b>
<b>3.1 ADVANCED ENCRYPTION STANDARD (AES) .....</b>	<b>47</b>
<b>3.1.1 AES Cipher.....</b>	<b>48</b>
<b>3.2 SUBBYTES ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ .....</b>	<b>52</b>
<b>3.2.1 Υλοποιήσεις του συστήματος SubBytes.....</b>	<b>54</b>
<b>3.3 SHIFTROWS ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ .....</b>	<b>57</b>
<b>3.4 MIXCOLUMNS ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ.....</b>	<b>58</b>
<b>3.4.1 Προτεινόμενη αρχιτεκτονική του συστήματος MixColumns .....</b>	<b>60</b>
<b>3.5 ADDROUNDKEY ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ .....</b>	<b>60</b>
<b>3.5.1 AES Key Expansion.....</b>	<b>62</b>

<b>ΚΕΦΑΛΑΙΟ 4 .....</b>	<b>67</b>
<b>GALOIS / COUNTER MODE (GCM).....</b>	<b>67</b>
<b>4.1 ΤΑ ΣΤΟΙΧΕΙΩΔΗ ΜΕΡΗ ΤΟΥ GCM.....</b>	<b>68</b>
<b>4.1.1 Block Cipher.....</b>	<b>68</b>
<b>4.1.2 Οι δύο GCM λειτουργίες.....</b>	<b>69</b>
<b>4.1.3 Εμπιστευτικότητα και Πιστοποίηση.....</b>	<b>71</b>
<b>4.1.4 Τύποι Εφαρμογών του GCM .....</b>	<b>72</b>
<b>4.2 ΤΑ ΒΑΣΙΚΑ ΜΑΘΗΜΑΤΙΚΑ ΣΤΟΙΧΕΙΑ ΤΟΥ GCM.....</b>	<b>73</b>
<b>4.2.1 GHASH Function.....</b>	<b>73</b>
<b>4.2.2 GCTR Function.....</b>	<b>75</b>
<b>4.3 GCM SPECIFICATION.....</b>	<b>77</b>
<b>4.3.1 Authenticated Encryption .....</b>	<b>77</b>
<b>4.3.2 Authenticated Decryption .....</b>	<b>79</b>
<b>4.4 ΑΠΑΙΤΗΣΕΙΣ ΜΟΝΑΔΙΚΟΤΗΤΑΣ ΤΩΝ IV ΚΑΙ KEYS .....</b>	<b>81</b>
<b>4.4.1 Επιλογή του κλειδιού .....</b>	<b>81</b>
<b>4.4.2 Κατασκευές IV .....</b>	<b>82</b>
<b>4.4.3 Εμπόδια στον αριθμό των επικλήσεων .....</b>	<b>84</b>
<b>4.5 ΥΨΗΛΗΣ ΤΑΧΥΤΗΤΑΣ GCM ΥΛΟΠΟΙΗΣΕΙΣ.....</b>	<b>85</b>
<b>ΚΕΦΑΛΑΙΟ 5 .....</b>	<b>87</b>
<b>ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗ.....</b>	<b>87</b>
<b>5.1 ΔΙΕΡΓΑΣΙΕΣ ΚΑΙ ΜΗΧΑΝΙΣΜΟΙ ΤΟΥ GCM .....</b>	<b>87</b>
<b>5.1.1 Διεργασία Κρυπτογράφησης (Encryption Function) .....</b>	<b>87</b>
<b>5.1.2 Διεργασία Αποκρυπτογράφησης (Decryption Function) .....</b>	<b>94</b>
<b>5.2 ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΣΤΟΝ GCM.....</b>	<b>95</b>
<b>5.2.1 Μια hardware υλοποίηση του GCM.....</b>	<b>95</b>
<b>5.2.2 Διάγραμμα Ροής Μηχανισμών .....</b>	<b>98</b>
<b>5.3 ΣΧΕΔΙΑΣΜΟΣ ΤΩΝ ΒΑΣΙΚΩΝ ΔΟΜΩΝ ΤΟΥ GCM.....</b>	<b>102</b>
<b>5.3.1 AES Δομή.....</b>	<b>102</b>
<b>5.3.2 Σχεδίαση του GCTR Μηχανισμού Εμπιστευτικότητας.....</b>	<b>105</b>
<b>5.3.3 GHASH Δομή .....</b>	<b>107</b>
<b>5.3.4 Ροή δεδομένων στον GCM και Χρονισμοί .....</b>	<b>108</b>
<b>5.4 ΥΨΗΛΗΣ ΤΑΧΥΤΗΤΑΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ GCM.....</b>	<b>110</b>
<b>ΚΕΦΑΛΑΙΟ 6 .....</b>	<b>115</b>
<b>VHDL, ΣΧΕΔΙΑΣΗ, ΕΞΟΜΟΙΩΣΗ ΚΑΙ .....</b>	<b>115</b>
<b>ΣΥΝΘΕΣΗ.....</b>	<b>115</b>
<b>6.1. ΔΙΑΔΙΚΑΣΙΑ ΣΧΕΔΙΑΣΜΟΥ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ.....</b>	<b>115</b>
<b>6.2. ΙΕΡΑΡΧΙΚΗ ΣΧΕΔΙΑΣΗ ΣΥΣΤΗΜΑΤΩΝ .....</b>	<b>116</b>
<b>6.3. ΓΛΩΣΣΕΣ ΠΕΡΙΓΡΑΦΗΣ ΥΛΙΚΟΥ (HDL) .....</b>	<b>117</b>
<b>6.4. Η VHDL .....</b>	<b>117</b>
<b>6.5. ΣΧΕΔΙΑΣΗ, ΕΞΟΜΟΙΩΣΗ ΚΑΙ ΣΥΝΘΕΣΗ .....</b>	<b>119</b>
<b>6.5.1. Μεθοδολογία Σχεδίασης.....</b>	<b>119</b>
<b>6.5.2. Το εργαλείο Modelsim .....</b>	<b>120</b>
<b>6.5.3. Το εργαλείο Leonardo Spectrum.....</b>	<b>121</b>
<b>6.5.4. Επιλογή Τεχνολογίας .....</b>	<b>122</b>

<b>ΚΕΦΑΛΑΙΟ 7 .....</b>	<b>125</b>
<b>SIMULATION AND SYNTHESIS RESULTS.....</b>	<b>125</b>
<b>7.1 ΕΞΟΜΟΙΩΣΗ ΤΗΣ GCTR ΔΟΜΗΣ ΤΟΥ AES-GCM .....</b>	<b>125</b>
<b>7.1.1 Κρυπτογράφηση – Εξομοίωση και Χρονισμοί .....</b>	<b>127</b>
<b>7.1.2 Αποκρυπτογράφηση – Εξομοίωση και Χρονισμοί.....</b>	<b>133</b>
<b>7.2 ΣΥΝΘΕΣΗ ΤΗΣ GCTR ΔΟΜΗΣ ΤΟΥ AES-GCM .....</b>	<b>137</b>
<b>7.2.1 Αποτελέσματα σύνθεσης ως προς την επιφάνεια και τον χρόνο .....</b>	<b>137</b>
<b>7.3 ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΚΡΥΠΤΟΓΡΑΦΙΚΗ ΠΛΑΤΦΟΡΜΑ AES-GCM .....</b>	<b>141</b>
<b>7.3.1 Τρόπος χρήσης - MANUAL .....</b>	<b>142</b>
<b>7.4 ΕΞΟΜΟΙΩΣΗ ΤΟΥ AES-GCM ΠΡΟΤΥΠΟΥ ΑΣΦΑΛΕΙΑΣ.....</b>	<b>148</b>
<b>7.5 ΣΥΝΘΕΣΗ ΤΟΥ AES-GCM ΠΡΟΤΥΠΟΥ ΑΣΦΑΛΕΙΑΣ.....</b>	<b>152</b>
<b>ΠΑΡΑΡΤΗΜΑ Α .....</b>	<b>157</b>
<b>ΒΙΒΛΙΟΓΡΑΦΙΑ .....</b>	<b>183</b>



# Κεφάλαιο 1

## Εισαγωγή

### 1.1 Εισαγωγή στην κρυπτογραφία

Η κρυπτογραφία είναι ένας επιστημονικός κλάδος που ασχολείται με την μελέτη, την ανάπτυξη και την χρήση τεχνικών κρυπτογράφησης και αποκρυπτογράφησης με σκοπό την απόκρυψη του περιεχομένου των μηνυμάτων. Ιστορικά η κρυπτογραφία χρησιμοποιήθηκε για την κρυπτογράφηση μηνυμάτων δηλαδή μετατροπή της πληροφορίας από μια κανονική κατανοητή μορφή σε έναν γρίφο, που χωρίς την γνώση του κρυφού μετασχηματισμού θα παρέμενε ακατανόητος. Κύριο χαρακτηριστικό των παλαιότερων μορφών κρυπτογράφησης ήταν ότι η επεξεργασία γινόταν πάνω στην γλωσσική δομή. Στις νεότερες μορφές η κρυπτογραφία κάνει χρήση του αριθμητικού ισοδύναμου, η έμφαση έχει μεταφερθεί σε διάφορα πεδία των μαθηματικών, όπως διακριτά μαθηματικά, θεωρία αριθμών, θεωρία πληροφορίας, υπολογιστική πολυπλοκότητα, στατιστική και συνδυαστική ανάλυση.

Η κρυπτογραφία παρέχει 4 βασικές λειτουργίες (αντικειμενικοί σκοποί):

**Εμπιστευτικότητα:** Η πληροφορία προς μετάδοση είναι προσβάσιμη μόνο στα εξουσιοδοτημένα μέλη. Η πληροφορία είναι ακατανόητη σε κάποιον τρίτο.

**Ακεραιότητα:** Η πληροφορία μπορεί να αλλοιωθεί μόνο από τα εξουσιοδοτημένα μέλη και δεν μπορεί να αλλοιώνεται χωρίς την ανίχνευση της αλλοίωσης.

**Μη απάρνηση:** Ο αποστολέας ή ο παραλήπτης της πληροφορίας δεν μπορεί να αρνηθεί την αυθεντικότητα της μετάδοσης ή της δημιουργίας της.

**Πιστοποίηση:** Οι αποστολέας και παραλήπτης μπορούν να εξακριβώνουν τις ταυτότητές τους καθώς και την πηγή και τον προορισμό της πληροφορίας με διαβεβαίωση ότι οι ταυτότητές τους δεν είναι πλαστές.

## Ορολογία

**Κρυπτογράφηση (encryption)** ονομάζεται η διαδικασία μετασχηματισμού ενός μηνύματος σε μία ακατανόητη μορφή με την χρήση κάποιου κρυπτογραφικού αλγορίθμου ούτως ώστε να μην μπορεί να διαβαστεί από κανέναν εκτός του νόμιμου παραλήπτη.

Η αντίστροφη διαδικασία όπου από το κρυπτογραφημένο κείμενο παράγεται το αρχικό μήνυμα ονομάζεται **αποκρυπτογράφηση (decryption)**.

**Κρυπτογραφικός αλγόριθμος (cipher)** είναι η μέθοδος μετασχηματισμού δεδομένων σε μία μορφή που να μην επιτρέπει την αποκάλυψη των περιεχομένων τους από μη εξουσιοδοτημένα μέρη. Κατά κανόνα ο κρυπτογραφικός αλγόριθμος είναι μία πολύπλοκη μαθηματική συνάρτηση.

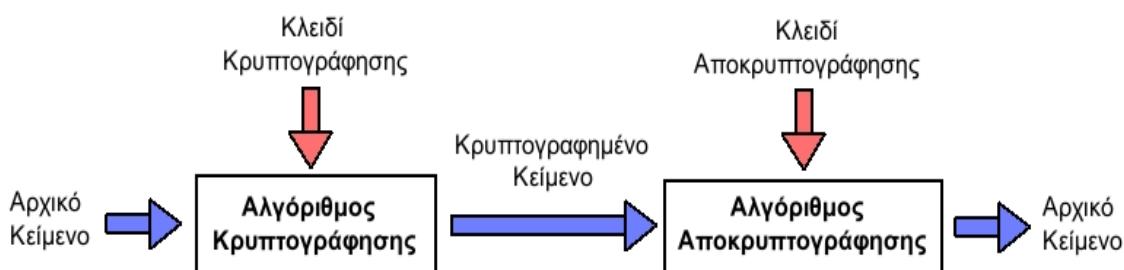
**Αρχικό κείμενο (plaintext)** είναι το μήνυμα το οποίο αποτελεί την είσοδο σε μία διεργασία κρυπτογράφησης.

**Κλειδί (key)** είναι ένας αριθμός αρκετών bit που χρησιμοποιείται ως είσοδος στην συνάρτηση κρυπτογράφησης.

**Κρυπτογραφημένο κείμενο (Ciphertext)** είναι το αποτέλεσμα της εφαρμογής ενός κρυπτογραφικού αλγόριθμου πάνω στο αρχικό κείμενο.

**Κρυπτανάλυση (cryptanalysis)** είναι μία επιστήμη που ασχολείται με το "σπάσιμο" κάποιας κρυπτογραφικής τεχνικής ούτως ώστε χωρίς να είναι γνωστό το κλειδί της κρυπτογράφησης, το αρχικό κείμενο να μπορεί να αποκωδικοποιηθεί.

Η διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης φαίνεται στο παρακάτω σχήμα:



Η κρυπτογράφηση και αποκρυπτογράφηση ενός μηνύματος γίνεται με τη βοήθεια ενός αλγόριθμου κρυπτογράφησης (cipher) και ενός κλειδιού κρυπτογράφησης (key). Συνήθως ο αλγόριθμος κρυπτογράφησης είναι γνωστός, οπότε η εμπιστευτικότητα

του κρυπτογραφημένου μηνύματος που μεταδίδεται βασίζεται ως επί το πλείστον στην μυστικότητα του κλειδιού κρυπτογράφησης.

Το μέγεθος του κλειδιού κρυπτογράφησης μετριέται σε αριθμό bits. Γενικά ισχύει ο εξής κανόνας: όσο μεγαλύτερο είναι το κλειδί κρυπτογράφησης, τόσο δυσκολότερα μπορεί να αποκρυπτογραφηθεί το κρυπτογραφημένο μήνυμα από επίδοξους εισβολείς. Διαφορετικοί αλγόριθμοι κρυπτογράφησης απαιτούν διαφορετικά μήκη κλειδιών για να πετύχουν το ίδιο επίπεδο ανθεκτικότητας κρυπτογράφησης.

## Ιστορική Αναδρομή

### *Πρώτη Περίοδος Κρυπτογραφίας (1900 π.Χ. – 1900 μ.Χ.)*

Κατά την διάρκεια αυτής της περιόδου αναπτύχθηκε μεγάλο πλήθος μεθόδων και αλγορίθμων κρυπτογράφησης, που βασίζονταν κυρίως σε απλές αντικαταστάσεις γραμμάτων. Όλες αυτές δεν απαιτούσαν εξειδικευμένες γνώσεις και πολύπλοκες συσκευές, αλλά στηρίζονταν στην ευφυΐα και την ευρηματικότητα των δημιουργών τους. Όλα αυτά τα συστήματα έχουν στις μέρες μας κρυπταναλυθεί και έχει αποδειχθεί ότι, εάν είναι γνωστό ένα μεγάλο κομμάτι του κρυπτογραφημένου μηνύματος, τότε το αρχικό κείμενο μπορεί σχετικά εύκολα να επανακτηθεί.

### *Δεύτερη Περίοδος Κρυπτογραφίας (1900 μ.Χ. – 1950 μ.Χ.)*

Η δεύτερη περίοδος της κρυπτογραφίας όπως προαναφέρθηκε τοποθετείται στις αρχές του 20ου αιώνα και φτάνει μέχρι το 1950. Καλύπτει, επομένως, τους δύο παγκόσμιους πολέμους, εξαιτίας των οποίων (λόγω της εξαιρετικά μεγάλης ανάγκης που υπήρξε για ασφάλεια κατά την μετάδοση ζωτικών πληροφοριών μεταξύ των στρατευμάτων των χωρών) αναπτύχθηκε η κρυπτογραφία τόσο όσο δεν είχε αναπτυχθεί τα προηγούμενα 3000 χρόνια. Τα κρυπτοσυστήματα αυτής της περιόδου αρχίζουν να γίνονται πολύπλοκα, και να αποτελούνται από μηχανικές και ηλεκτρομηχανικές κατασκευές, οι οποίες ονομάζονται «κρυπτομηχανές». Η κρυπτανάλυση τους, απαιτεί μεγάλο αριθμό προσωπικού, το οποίο εργαζόταν επί μεγάλο χρονικό διάστημα ενώ ταυτόχρονα γίνεται εξαιρετικά αισθητή η ανάγκη για μεγάλη υπολογιστική ισχύ. Παρά την πολυπλοκότητα που αποκτούν τα συστήματα κρυπτογράφησης κατά την διάρκεια αυτής της περιόδου η κρυπτανάλυση τους είναι συνήθως επιτυχημένη. Οι Γερμανοί έκαναν εκτενή χρήση (σε διάφορες παραλλαγές) ενός συστήματος γνωστού ως Enigma.

Ο Marian Rejewski, στην Πολωνία, προσπάθησε και, τελικά, παραβίασε την πρώτη μορφή του γερμανικού στρατιωτικού συστήματος Enigma (που χρησιμοποιούσε μια ηλεκτρομηχανική κρυπτογραφική συσκευή) χρησιμοποιώντας θεωρητικά μαθηματικά το 1932.

### Τρίτη Περίοδος Κρυπτογραφίας (1950 μ.Χ. - Σήμερα)

Αυτή η περίοδος χαρακτηρίζεται από την έξαρση της ανάπτυξης στους επιστημονικούς κλάδους των μαθηματικών, της μικροηλεκτρονικής και των υπολογιστικών συστημάτων. Η εποχή της σύγχρονης κρυπτογραφίας αρχίζει ουσιαστικά με τον Claude Shannon, αναμφισβήτητα ο πατέρας των μαθηματικών συστημάτων κρυπτογραφίας. Το 1949 δημοσίευσε το έγγραφο «Θεωρία επικοινωνίας των συστημάτων μυστικότητας» (Communication Theory of Secrecy Systems) στο τεχνικό περιοδικό Bell System και λίγο αργότερα στο βιβλίο του, «Μαθηματική Θεωρία της Επικοινωνίας» (Mathematical Theory of Communication), μαζί με τον Warren Weaver. Αυτά, εκτός από τις άλλες εργασίες του επάνω στην θεωρία δεδομένων και επικοινωνίας καθιέρωσε μια στερεά θεωρητική βάση για την κρυπτογραφία και την κρυπτανάλυση. Εκείνη την εποχή η κρυπτογραφία εξαφανίζεται και φυλάσσεται από τις μυστικές υπηρεσίες κυβερνητικών επικοινωνιών όπως η NSA. Πολύ λίγες εξελίξεις δημοσιοποιήθηκαν ξανά μέχρι τα μέσα της δεκαετίας του '70, όταν όλα άλλαξαν.

Στα μέσα της δεκαετίας του '70 έγιναν δύο σημαντικές δημόσιες (δηλ. μη-μυστικές) πρόοδοι. Πρώτα ήταν η δημοσίευση του σχεδίου προτύπου κρυπτογράφησης DES (Data Encryption Standard) στον ομοσπονδιακό κατάλογο της Αμερικής στις 17 Μαρτίου 1975. Το προτεινόμενο DES υποβλήθηκε από την IBM, στην πρόσκληση του Εθνικού Γραφείου των Προτύπων (τώρα γνωστό ως NIST), σε μια προσπάθεια να αναπτυχθούν ασφαλείς ηλεκτρονικές εγκαταστάσεις επικοινωνίας για επιχειρήσεις όπως τράπεζες και άλλες μεγάλες οικονομικές οργανώσεις. Μετά από τις συμβουλές και την τροποποίηση από την NSA, αυτό το πρότυπο υιοθετήθηκε και δημοσιεύθηκε ως ένα ομοσπονδιακό τυποποιημένο πρότυπο επεξεργασίας πληροφοριών το 1977 (αυτήν την περίοδο αναφέρεται σαν FIPS 46-3). Ο DES ήταν ο πρώτος δημόσια προσιτός αλγόριθμος κρυπτογράφησης που εγκρίνεται από μια εθνική αντιπροσωπεία όπως η NSA. Η απελευθέρωση της προδιαγραφής του από την NSA υποκίνησε μια έκρηξη δημόσιου και ακαδημαϊκού ενδιαφέροντος για τα συστήματα κρυπτογραφίας.

Ο DES αντικαταστάθηκε επίσημα από το AES το 2001 όταν ανήγγειλε ο NIST το FIPS 197. Μετά από έναν ανοικτό διαγωνισμό, ο NIST επέλεξε τον αλγόριθμο Rijndael, που υποβλήθηκε από δύο Φλαμανδούς κρυπτογράφους, για να είναι το AES. Ο DES και οι ασφαλέστερες παραλλαγές του όπως ο 3DES ή TDES χρησιμοποιούνται ακόμα και σήμερα, ενσωματωμένοι σε πολλά εθνικά και οργανωτικά πρότυπα. Εντούτοις, το βασικό μέγεθος των 56-bit έχει αποδειχθεί ότι είναι ανεπαρκές για να αντισταθεί στις επιθέσεις brute force (επίθεση κατά την οποία ουσιαστικά δοκιμάζει όλα τα δυνατά κλειδιά μέχρι να βρει το κατάλληλο), (μια τέτοια επίθεση πέτυχε να σπάσει τον DES σε 56 ώρες ενώ το άρθρο που αναφέρεται ως το σπάσιμο του DES δημοσιεύθηκε από τον O'Reilly and Associates). Κατά συνέπεια, η χρήση απλής κρυπτογράφησης με τον DES είναι τώρα χωρίς αμφιβολία επισφαλής για χρήση στα νέα σχέδια των κρυπτογραφικών συστημάτων και μηνύματα που προστατεύονται από τα παλαιότερα κρυπτογραφικά συστήματα που χρησιμοποιούν DES, και όλα τα μηνύματα που έχουν αποσταλεί από το 1976 με την χρήση DES, διατρέχουν επίσης σοβαρό κίνδυνο αποκρυπτογράφησης. Ανεξάρτητα από την έμφυτη ποιότητά του, το βασικό μέγεθος του DES (56-bit) ήταν πιθανόν πάρα πολύ μικρό ακόμη και το 1976, πράγμα που είχε επισημάνει ο Whitfield Diffie. Υπήρξε επίσης η υποψία ότι κυβερνητικές οργανώσεις είχαν ακόμα και τότε ικανοποιητική υπολογιστική δύναμη ώστε να σπάσουν μηνύματα που είχαν

κρυπτογραφηθεί με τον DES. Η ανάγκη επομένως για αντικατάσταση του DES ήταν επιβλητική. Τα δεδομένα από τότε έχουν αλλάξει αρκετά. Αρκετές εταιρίες και επιχειρήσεις έχουν δημιουργήσει δικούς τους αλγόριθμους κρυπτογράφησης προκειμένου να διασφαλίσουν τα δεδομένα τους.

## Κατηγορίες Κρυπτοσυστημάτων

### *Συμμετρικά Κρυπτοσυστήματα*

Συμμετρικό κρυπτοσύστημα είναι το σύστημα εκείνο το οποίο χρησιμοποιεί κατά την διαδικασία της κρυπτογράφησης και αποκρυπτογράφησης ένα κοινό κλειδί. Η ασφάλεια αυτών των αλγορίθμων βασίζεται στην μυστικότητα του κλειδιού. Τα συμμετρικά κρυπτοσυστήματα προϋποθέτουν την ανταλλαγή του κλειδιού μέσα από ένα ασφαλές κανάλι επικοινωνίας ή μέσα από την φυσική παρουσία των προσώπων. Αυτό το χαρακτηριστικό καθιστά δύσκολη την επικοινωνία μεταξύ απομακρυσμένων ατόμων.

### *Ασύμμετρα κρυπτοσυστήματα*

Το ασύμμετρο κρυπτοσύστημα ή κρυπτοσύστημα δημοσίου κλειδιού δημιουργήθηκε για να καλύψει την αδυναμία μεταφοράς κλειδιών που παρουσίαζαν τα συμμετρικά συστήματα. Χαρακτηριστικό του είναι ότι έχει δυο είδη κλειδιών ένα ιδιωτικό και ένα δημόσιο. Το δημόσιο είναι διαθέσιμο σε όλους ενώ το ιδιωτικό είναι μυστικό. Η βασική σχέση μεταξύ τους είναι: ότι κρυπτογραφεί το ένα, μπορεί να το αποκρυπτογραφήσει μόνο το άλλο.

Στα σύγχρονα συστήματα συνήθως υιοθετείται μια μέθοδος ασύμμετρου – συμμετρικού όπου χρησιμοποιείται ασύμμετρο σύστημα για την μεταφορά του κλειδιού και μετά συμμετρικό σύστημα για την μεταφορά και κρυπτογράφηση – αποκρυπτογράφηση των δεδομένων. Με αυτό τον τρόπο εκμεταλλεύονται τα προτερήματα και των δύο συστημάτων.

## Εφαρμογές κρυπτογραφίας

Η εξέλιξη της χρησιμοποίησης της κρυπτογραφίας ολοένα αυξάνεται καθιστώντας πλέον αξιόπιστη την μεταφορά της πληροφορίας για διάφορους λειτουργικούς σκοπούς. Μερικοί από αυτούς είναι:

1. Ασφάλεια συναλλαγών σε τράπεζες δίκτυα – ATM
2. Κινητή τηλεφωνία (ΤΕΤΡΑ-ΤΕΤΡΑΠΟΛ-GSM)
3. Σταθερή τηλεφωνία (cryptophones)
4. Διασφάλιση Εταιρικών πληροφοριών
5. Στρατιωτικά δίκτυα (Τακτικά συστήματα επικοινωνιών μάχης)

6. Διπλωματικά δίκτυα (Τηλεγραφήματα)
7. Ηλεκτρονικές επιχειρήσεις (πιστωτικές κάρτες, πληρωμές)
8. Ηλεκτρονική ψηφοφορία
9. Ηλεκτρονική δημοπρασία
10. Ηλεκτρονικό γραμματοκιβώτιο
11. Συστήματα συναγερμών
12. Συστήματα βιομετρικής αναγνώρισης
13. Έξυπνες κάρτες
14. Ιδιωτικά δίκτυα (VPN)
15. Word Wide Web
16. Δορυφορικές εφαρμογές (δορυφορική τηλεόραση)
17. Ασύρματα δίκτυα (Hipperlan, Bluetooth, 802.11x)
18. Συστήματα ιατρικών δεδομένων και άλλων βάσεων δεδομένων
19. Τηλεσυνδιάσκεψη - Τηλεφωνία μέσω διαδικτύου (VOIP)

Τέλος, για να κλείσουμε αυτή την ενότητα αναφέρουμε ότι η κρυπτογραφία και κρυπτανάλυση βρίσκονται σε ένα συνεχή αγώνα δρόμου. Δεν υπάρχει αλγόριθμος κρυπτογράφησης που να μην σπάει (τουλάχιστον προς το παρόν και από αυτά που γνωρίζει το ευρύ κοινό). Οι αλγόριθμοί που χρησιμοποιούνται είναι τόσο ισχυροί που να χρειάζεται πολύς χρόνος και υπολογιστική ισχύ μέχρι να αποκρυπτογραφηθεί το μήνυμα. Επίσης, γίνονται έρευνες για την χρήση διπλής κρυπτογράφησης έτσι ώστε ακόμα και με την χρήση brute force επίθεσης να μην μπορεί κάποιος μη εξουσιοδοτημένος να αποκρυπτογραφήσει το κρυπτογραφημένο μήνυμα.

## 1.2 Η ανάγκη ύπαρξης του GCM

Από το 2001, το National Institute of Standards and Technology (NIST) έχει εκδώσει τέσσερα recommendation για το Block Cipher Mode of Operation, και πιο συγκεκριμένα τα SP800-38A[1], SP800-38B[21], SP800-38C[22] και SP800-38D[2]. Το block cipher mode of operation είναι ένας αλγόριθμος που χρησιμοποιεί ένα συμμετρικό κλειδί για να παρέχει εμπιστευτικότητα, γνησιότητα-πιστοποίηση ή και τα δύο μαζί για την ασφάλεια της πληροφορίας.

Στο πρώτο άρθρο SP800-38A, το NIST εισάγει τον AES ως αντικαταστάτη του DES και προτείνει πέντε τρόπους/μεθόδους διασφάλισης της εμπιστευτικότητας που χρησιμοποιούνται με έναν προαπαιτούμενο συμμετρικό αλγόριθμο κλειδιού. Αυτοί είναι:

- Electronic Codebook (ECB) mode,
- Cipher Block Chaining (CBC) mode,
- Cipher Feedback (CFB) mode,
- Output Feedback (OFB) mode, and
- Counter (CTR) mode

Αυτοί οι πέντε μέθοδοι μπορούν να χωριστούν σε δύο ομάδες : σε ένα non-feedback mode group, που περιλαμβάνει τον ECB και τον CTR, και σε ένα feedback mode group, που περιλαμβάνει τους CBC, CFB και OFB. Στο feedback mode, το τρέχον εκτελέσιμο βήμα εξαρτάται από το αποτέλεσμα του προηγούμενου βήματος. Επομένως, για να εφαρμόσουμε αυτές τις μεθόδους σε hardware, τυπικά χρησιμοποιείται μια επαναληπτική (iterative) αρχιτεκτονική, για απαιτήσεις χαμηλής απόδοσης, και δεν χρησιμοποιείται μια pipelined αρχιτεκτονική. Κατά συνέπεια, η χρήση των ECB και CTR mode (ή των non-feedback modes) που υποστηρίζουν pipelined ή και parallelized αρχιτεκτονικές σχεδίασης (τεχνικές διοχέτευσης και παραλληλισμού) χρησιμοποιούνται για high-speed data flows.

Στο δεύτερο άρθρο SP800-38B, το NIST προτείνει ένα message authentication code (MAC) αλγόριθμο που βασίζεται σε ένα συμμετρικού κλειδιού block cipher. Αυτός ο cipher-based MAC αναφέρεται ως CMAC, είτε ως Cipher Block Chaining MAC αλγόριθμος (CBC-MAC).

Στο τρίτο άρθρο SP800-38C, το NIST προτείνει ένα mode of operation, το οποίο καλείται CCM, βασισμένο σε ένα συμμετρικού κλειδιού block cipher αλγόριθμο του οποίου το block size είναι 128 bits. Ο CCM παρέχει την εμπιστευτικότητα και την γνησιότητα-πιστοποίηση (authenticity) των δεδομένων με το να ενοποιεί τις τεχνικές του CTR mode και του CBC-MAC.

Η απόδοση (throughput) της CTR mode υλοποίησης, που μπορεί να υλοποιηθεί με τεχνικές διοχέτευσης, είναι πολύ μεγαλύτερη από την CBC-MAC υλοποίηση, που δεν μπορεί να υλοποιηθεί με τεχνικές παραλληλισμού. Πιο συγκεκριμένα κανένα από τα τρία προτεινόμενα άρθρα SP800-38A, B, και C δεν μπορούν να υιοθετηθούν για μια high speed network και computer systems εφαρμογή. Επομένως, υπάρχει μια επιτακτική ανάγκη για μια μέθοδο διεργασιών (mode of operation) που να παρέχει αποτελεσματικά εμπιστευτικότητα αλλά και πιστοποίηση των δεδομένων σε υψηλές ταχύτητες.

Το τέταρτο και τελευταίο άρθρο (προς το παρόν) της οικογένειας των security standard of Block Cipher Mode of Operation, SP800-38D, *Galois/Counter Mode of Operation (GCM)*, συμπληρώνει την ανάγκη που παρουσιάστηκε παραπάνω. Στον GCM παρουσιάζεται η χρήση ενός εγκεκριμένου συμμετρικού κλειδιού block cipher μήκους 128 bits και μιας καθολικής hash function, ορισμένης σε ένα δυαδικό Galois field. Το εγκεκριμένου συμμετρικού κλειδιού block cipher μήκους 128 bits που χρησιμοποιείται στο GCM είναι ο Advanced Encryption Standard (AES) αλγόριθμος όπως διευκρινίζεται στο Federal Information Processing Standard (FIPS). Η συγκεκριμένη καθολική hash function ορίζεται σε ένα δυαδικό Galois field και είναι ένας 128-bit πολυωνυμικός πολλαπλασιαστής στο  $GF(2^{128})$  και καλείται GHASH. Η GHASH μπορεί να προσφέρει έναν ασφαλή, παραλληλοποιήσιμο και αποτελεσματικό μηχανισμό πιστοποίησης. Όσο αφορά στον μηχανισμό της εμπιστευτικότητας του GCM, έχει υιοθετηθεί ένα CTR mode που εμπεριέχει ένα ECB mode του AES, το οποίο καλείται GCTR function και χρησιμοποιεί ένα θεμελιώδη block cipher.

### 1.3 Εισαγωγή στον GCM

Σκοπός της διπλωματικής είναι να αναλύσει και να προτείνει μια υλοποίηση του GCM βασισμένη στο τέταρτο και τελευταίο άρθρο της οικογένειας των security standard of Block Cipher Mode of Operation, SP800-38D, *Galois/Counter Mode of Operation (GCM)*. Προσδιορίζοντας τον αλγόριθμο Galois/Counter Mode (GCM) για πιστοποιημένη κρυπτογράφηση δεδομένων, ο GCM αποτελείται από ένα εγκεκριμένου συμμετρικού κλειδιού block cipher μήκους 128 bits, όπως είναι ο Advanced Encryption Standard (AES). Έτσι ο GCM μπορεί να χαρακτηριστεί ως μια τροποποιημένη – βελτιωμένη έκδοση του AES αλγορίθμου.

Ο GCM προσφέρει διασφάλιση της πιστοποίησης των δεδομένων (*μέχρι και 64 gigabyte για κάθε επίκληση*) χρησιμοποιώντας μία universal hash function που ορίζεται στο δυαδικό Galois field. Ο GCM μπορεί επίσης να προσφέρει διασφάλιση της πιστοποίησης για επιπρόσθετα δεδομένα (πρακτικά απείρου μήκους για κάθε επίκληση) τα οποία δεν κρυπτογραφούνται.

Αν η είσοδος του GCM περιοριστεί σε δεδομένα που δεν είναι για κρυπτογράφηση, τότε το αποτέλεσμα που προκύπτει από την εξειδίκευση του GCM καλείται *GMAC* και είναι απλά ένα authentication mode των δεδομένων εισόδου. Στο εξής όλες οι δηλώσεις για το GCM θα εφαρμόζονται και στο GMAC.

Ο GCM παρέχει μια πιο δυνατή διασφάλιση της πιστοποίησης σε σχέση με έναν (non – cryptographic) checksum ή error detection κώδικα και πιο συγκεκριμένα μπορεί να ανιχνεύσει και μια τυχαία τροποποίηση των δεδομένων αλλά και μια εσκεμμένη μη εξουσιοδοτημένη τροποποίηση.

Δύο είναι οι λειτουργίες του GCM, η πιστοποιημένη κρυπτογράφηση και η πιστοποιημένη αποκρυπτογράφηση. Κάθε μία από τις λειτουργίες αυτές είναι σχετικά αποτελεσματική και παραλληλοποιήσιμη, και επομένως υψηλής απόδοσης εφαρμογές είναι δυνατές τόσο σε υλικό όσο και σε λογισμικό. *O GCM έχει πολλά άλλα χρήσιμα χαρακτηριστικά, συμπεριλαμβανομένων των εξής:*

- Οι λειτουργίες του GCM είναι “online” με την έννοια ότι το μήκη των εμπιστευτικών και μη εμπιστευτικών δεδομένων δεν είναι προαπαιτούμενα αλλά μπορούν να υπολογιστούν καθώς τα δεδομένα λαμβάνονται και επεξεργάζονται.
- Οι λειτουργίες του GCM απαιτούν μόνο την forward κατεύθυνση του προαπαιτούμενου block cipher (η αντίστροφη λειτουργία δεν απαιτείται). Η forward κατεύθυνση δηλαδή χρησιμοποιείται τόσο στην κρυπτογράφηση όσο και στην αποκρυπτογράφηση.

- Η πιστοποίηση των προστατευμένων δεδομένων μπορεί να επαληθευθεί ανεξάρτητα από την ανάκτηση των εμπιστευμένων δεδομένων από την κρυπτογραφημένη τους μορφή (authentication tag).
- Αν το μοναδικό μπλοκ αρχικοποίησης (initialization string) είναι προβλέψιμο και το μήκος των εμπιστευτικών δεδομένων είναι γνωστό, τότε οι block cipher επικλήσεις (invocations) μέσα στον GCM μηχανισμό κρυπτογράφησης μπορούν να προϋπολογιστούν.
- Αν κάποια ή όλα τα επιπρόσθετα δεδομένα, μη-πιστοποιημένα δεδομένα, είναι σταθερά (fixed), τότε τα αντίστοιχα στοιχεία του GCM μηχανισμού πιστοποίησης μπορούν να προϋπολογιστούν .

## 1.4 Χρήση του GCM

Ο GCM αλγόριθμος χρησιμοποιείται ήδη στο πρότυπο IEEE 802.1AE (MACsec) Ethernet security το οποίο καθορίζει την connectionless εμπιστευτικότητα και την ακεραιότητα των δεδομένων για την media πρόσβαση ανεξάρτητων πρωτόκολλων. Είναι τυποποιημένο από το IEEE στην 802.1 ομάδα εργασίας. Επίσης χρησιμοποιείται στο πρωτόκολλο ANSI (INCITS) Fibre Channel Security Protocols (FC-SP) το οποίο με την σειρά του είναι ένα gigabit – speed network technology που κατά κύριο λόγο χρησιμοποιείται για storage networking (storage area network) (SAN). Επιπλέον, ο GCM αλγόριθμος χρησιμοποιείται στο πρότυπο IEEE P1619.1 tape storage και στο IETF IPsec standards.

Είναι εμφανές ότι ο αλγόριθμος αυτός **χρησιμοποιείται σε εφαρμογές με high throughput απαιτήσεις**. Αποτελεί έναν από τους κατάλληλους αλγόριθμους όταν το κριτήριο για την απόδοση δεν είναι η επιφάνεια και η κατανάλωση, αλλά η ταχύτητα. Σημαντικό είναι επίσης το γεγονός ότι εκτός από κρυπτογράφηση των δεδομένων παρέχει και πιστοποίηση πράγμα απαραίτητο όταν πρόκειται για storage networks ή tapes. Είναι ένας αλγόριθμος ευέλικτος αλλά και ταυτόχρονα ισχυρός. Έχουν επίσης παρουσιαστεί υλοποιήσεις αυτού του αλγορίθμου χαμηλές σε area και σε κατανάλωση κατάλληλες για wireless και mobile εφαρμογές.

Σκοπός της παρούσας διπλωματικής είναι η υλοποίηση του GCTR αλγορίθμου για high speed εφαρμογές.



## Κεφάλαιο 2

### Background

Το κεφάλαιο αυτό παρέχει τις βασικές έννοιες που είναι αναγκαίες για την κατανόηση του AES-GCM. Στην ενότητα 2.1 παρουσιάζονται οι συμβολισμοί – ακρώνυμα καθώς και μια σύντομη περιγραφή του καθενός. Η ενότητα 2.2 εισάγει τις έννοιες των πεπερασμένων πεδίων. Ακολούθως, στην ενότητα 2.3 αναφέρονται οι βασικές πράξεις, με παραδείγματα, που γίνονται μεταξύ των αλγορίθμων του AES-GCM, καθώς επίσης αναλύονται και τα στοιχεία του στα οποία πραγματοποιούνται μαθηματικές πράξεις. Στην ενότητα 2.4 παρουσιάζεται το υπόβαθρο των καταστάσεων λειτουργίας (modes of operation) που χρησιμοποιούνται στον AES-GCM, ενώ τέλος στην ενότητα 2.5 γίνεται μια εισαγωγή των FPGAs.

#### 2.1. Ορισμοί, Συντομογραφίες και σύμβολα

Στην συνέχεια παρατίθεται πίνακας με τις βασικές έννοιες καθώς και μια συνοπτική περιγραφή αυτών.

##### 2.1.1 Ορισμοί και Συντομογραφίες

AAD	Additional Authenticated Data  Τα δεδομένα εισόδου στην λειτουργία κρυπτογράφησης τα οποία πιστοποιούνται αλλά δεν κρυπτογραφούνται. Αυτά τα δεδομένα συνήθως περιλαμβάνουν πληροφορίες για το πρωτόκολλο δικτύου που χρησιμοποιείται, τις διευθύνσεις, τις θύρες, ακολουθίες αριθμών, την έκδοση πρωτοκόλλου και άλλα πεδία για το πώς το plaintext πρέπει να χειρίστει.
Authenticated Encryption	Η λειτουργία του GCM κατά την οποία το plaintext κρυπτογραφείται σε ciphertext και ένα authentication tag δημιουργείται από τα AAD και το ciphertext. Εξασφαλίζεται έτσι η εμπιστευτικότητα και η πιστοποίηση των δεδομένων αντίστοιχα.
Authenticated Decryption	Η λειτουργία του GCM κατά την οποία το Ciphertext αποκρυπτογραφείται σε plaintext και η γνησιότητα του ciphertext και των AAD επαληθεύεται. Είναι η αντίστροφη της παραπάνω διαδικασίας.

Authentication Tag (Tag)	Είναι ένα κρυπτογραφικό checksum των δεδομένων το οποίο έχει σχεδιαστεί για να αποκαλύπτει και τα τυχαία λάθη αλλά και την σκόπιμη τροποποίηση των δεδομένων. Δημιουργείται με σκοπό να εξασφαλίζει την ακριβή διάδοση δεδομένων.
Block Cipher	Μία παραμετροποιημένη οικογένεια αναστρέψιμων διαδικασιών πάνω σε bit strings καθορισμένου μήκους. Η παράμετρος που καθορίζει τις λειτουργίες που θα γίνουν είναι ένα bit string που ονομάζεται key.
Ciphertext	Η κρυπτογραφημένη μορφή του plaintext.
FIPS	Federal Information Processing Standard.
Forward Cipher Function	Μια διαδικασία διαμόρφωσης των blocks η οποία καθορίζεται από την επιλογή του κλειδιού για το δοσμένο block cipher.
Fresh	Για ένα νέο δημιουργούμενο κλειδί, η ιδιότητα της μοναδικότητας σε σχέση με τα προηγούμενα χρησιμοποιηθέντα κλειδιά.
GCM	Galois/Counter Mode
AES-GCM	Αναφέρεται στον GCM ο οποίος χρησιμοποιεί τον AES ως το προαπαιτούμενο συμμετρικό block cipher αλγόριθμο.
ICB	Initial Counter Block
IV	Initialization Vector
Initialization Vector	Μια αρχική τιμή μοναδική για κάθε κλειδί και διεργασία κατά την διαδικασία κρυπτογράφησης συγκεκριμένου plaintext και AAD.
Inverse Cipher Function	Η αντίστροφη διαδικασία της forward cipher function για ένα δοσμένο κλειδί.
Key	Η παράμετρος των block cipher που καθορίζει την επιλογή της forward cipher function από μία οικογένεια αναστρέψιμων διαδικασιών
Mode of Operation (Mode)	Ένα αλγόριθμός για την κρυπτογράφηση των δεδομένων που βασίζεται σε ένα block cipher.
NIST	National Institute of Standards and Technology.
Permutation	Μια αναστρέψιμη διαδικασία
Plaintext	Τα δεδομένα εισόδου στην διαδικασία πιστοποιημένης κρυπτογράφησης τα οποία κρυπτογραφούνται και πιστοποιούνται.
XOR	Exclusive-OR.

## 2.1.2 Σύμβολα και Μεταβλητές

<i>A</i>	The additional authenticated data Αναφέρεται και ως AAD
<i>C</i>	The Ciphertext
<i>H</i>	The hash subkey.
<i>ICB</i>	The initial counter block
<i>IV</i>	The initialization vector.
<i>K</i>	The block cipher key.
<i>P</i>	The plaintext.
<i>R</i>	The constant within the algorithm for the block multiplication operation.
<i>T</i>	The authentication tag.
<i>t</i>	The bit length of the authentication tag.

## 2.2 Μαθηματικό υπόβαθρο

Οι θεμελιώδης αρχές του AES και της GHASH (όπως αυτά αναλύονται σε παρακάτω κεφάλαια) βασίζονται σε πράξεις – διεργασίες σε πεπερασμένο πεδίο. Στην ενότητα αυτή, αναλύεται η έννοια του πεπερασμένου πεδίου και οι βασικές πράξεις σε αυτό.

### 2.2.1 Εισαγωγή στο πεπερασμένο πεδίο

Ένα πεδίο μπορεί να θεωρηθεί ως ένα σύνολο στοιχείων τα οποία αποτελούν ένα σύνολο  $G$  που αποτελείται από δύο πράξεις: τον πολλαπλασιασμό που αναγράφεται με το σύμβολο "\*" και την πρόσθεση που επισημαίνονται με το σύμβολο "+". Οι πράξεις αυτές τηρούν τις βασικές αλγεβρικές ιδιότητες. Οι σχετικές έννοιες του πεπερασμένου πεδίου καταγράφονται ως εξής:

**Αξίωμα 1.** Το  $(F, +, *)$  είναι ένα πεδίο, εάν ισχύουν οι εξής ιδιότητες:

- Τα στοιχεία του  $F$  αποτελούν ένα σύνολο κατά την πράξη της πρόσθεσης.
  - Τα μη μηδενικά στοιχεία του  $F$  αποτελούν ένα σύνολο κατά την πράξη του πολλαπλασιασμού.
  - Οι πράξεις της πρόσθεσης και του πολλαπλασιασμού είναι μεταθετικές σε όλο το σύνολο του  $F$ , δηλαδή :
- $x + y = y + x$  και  $x * y = y * x$ , για κάθε  $x, y$  που ανήκει στο  $F$ .
- Η πράξη του πολλαπλασιασμού μπορεί να διαχωριστεί από την πράξη της πρόσθεσης μέσω της επιμεριστικής ιδιότητας :
- $x * (y + z) = x * y + x * z$ , για κάθε  $x, y, z$  που ανήκει στο  $F$ .

**Αξίωμα 2.** Ένα πεδίο με πεπερασμένο αριθμό στοιχείων είναι ένα πεπερασμένο πεδίο.

**Αξίωμα 3.** Ένα μη μηδενικό στοιχείο ενός πεπερασμένου πεδίου  $F$  λέγεται ότι είναι το πρωταρχικό στοιχείο του πεδίου (primitive) ή αλλιώς ο παραγωγός του  $F$ , εάν οι δυνάμεις του καλύπτουν όλα τα μη μηδενικά στοιχεία πεδίου.

**Αξίωμα 4.** Ένα μοναδικό πεπερασμένο πεδίο υπάρχει για κάθε πρώτο αριθμό. Τα πεδία αυτά, δείχνονται ως  $GF(p^m)$ , όπου  $p$  είναι πρώτος αριθμός και  $m$  είναι ένας θετικός ακέραιος. Ένα είδος πεδίου που χρησιμοποιείται συνήθως στην κρυπτογραφία είναι το δυαδικό πεπερασμένο πεδίο  $GF(2^m)$ , όπου  $m$  είναι ένας μεγάλος ακέραιος.

**Αξίωμα 5.** Μια βάση για το  $GF(2^m)$  πάνω στο  $GF(2)$  είναι ένα σύνολο από  $m$  γραμμικώς ανεξάρτητων στοιχείων του  $GF(2^m)$ . Κάθε στοιχείο του  $GF(2^m)$ , μπορεί να περιγραφεί ως το αλγεβρικό άθροισμα των στοιχείων της βάσης.

Το δυαδικό πεδίο  $GF(2^m)$  περιέχει  $2^m$  στοιχεία. Κάθε στοιχείο εκπροσωπείται από την επιλεγμένη βάση. Η πιο κοινή εκπροσώπηση βασίζεται σε πολυωνυμική βάση. Με βάση το πολυώνυμο  $\alpha = (1, \alpha, \alpha^2, \dots, \alpha^{m-1})$ , τα στοιχεία του  $GF(2^m)$ , μπορούν να περιγραφούν ως πολυώνυμο βαθμού  $m-1$  ως εξής:

$$GF(2^m) = \{A \mid A = a_0 + a_1 * \alpha + \dots + a_{m-1} * \alpha^{m-1}, \text{ όπου } a_j \text{ ανήκει } GF(2), 0 \leq j \leq m-1\}$$

όπου  $\alpha$  είναι η βάση κάθε αμείωτου (irreducible) πολυώνυμου  $F(x)$  βαθμού  $m$  πάνω στο  $GF(2)$ .

Ας ορίσουμε :

$$F(x) = 1 + f_1 * x + f_2 * x^2 + \dots + f_{m-1} * x^{m-1} + x^m$$

όπου  $f_i$  ανήκει στο  $GF(2)$ ,  $0 \leq i \leq m-1$ . Το αμείωτο (irreducible) πολυώνυμο  $F(x)$  αναφέρεται συχνά ως το πολυώνυμο του πεδίου. Η αριθμητική στον AES - GCM βασίζεται στην πολυωνυμική βάση και χρησιμοποιεί το πολυώνυμο :

$$F(x) = 1 + x + x^2 + x^7 + x^{128}, \text{ ως πολυώνυμο πεδίου.}$$

## Συμπεράσματα

Ουσιαστικά, ένα πεδίο ορισμού είναι ένα σύνολο στο οποίο μπορεί να γίνει πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση μεταξύ των στοιχείων του πεδίου και το αποτέλεσμα αυτών να ανήκει στο πεδίο ορισμού. Η διαίρεση ορίζεται από τον εξής κανόνα  $a/b = a^*(b^{-1})$ , όπως και η αφαίρεση από τον κανόνα  $a-b = a+(-b)$ .

Γνώριμα παραδείγματα των πεδίων είναι οι δυαδικοί αριθμοί, οι πραγματικοί αριθμοί και οι μιγαδικοί αριθμοί. Σημειώνεται ότι οι ακέραιοι αριθμοί δεν αποτελούν είδος πεδίου ορισμού αφού δεν έχουν όλα τα στοιχεία του συνόλου των αντίστροφό του. Στην πραγματικότητα μόνο το 1 και το -1 έχουν αντίστροφο που είναι και ο εαυτός τους.

Στην κρυπτογραφία όμως, τέτοιου είδους πεδία δεν παρουσιάζουν ιδιαίτερο ενδιαφέρον, για την ακρίβεια δεν παρουσιάζουν κανένα ενδιαφέρον. Αντιθέτως, πεπερασμένα πεδία (πεπερασμένων στοιχείων και πεπερασμένου μήκους), παίζουν πολύ σημαντικό ρόλο στους αλγόριθμους κρυπτογραφίας και αυτό γιατί έχουμε να κάνουμε με λογικά κυκλώματα, λογικές πράξεις και πεπερασμένου χώρου μνήμες. Ωστόσο, οι αρχές των πεπερασμένων πεδίων που χρησιμοποιούνται στην κρυπτογραφία, βασίζονται σε αυτές των πεδίων απείρου μήκους.

### 2.2.2 Ανάγκη χρήσης πεδίων $GF(2^n)$ στους Αλγόριθμους Κρυπτογράφησης

Σχεδόν όλοι οι αλγόριθμοι κρυπτογράφησης, τόσο συμμετρικού όσο και κοινού κλειδιού, εμπεριέχουν αριθμητικές πράξεις μεταξύ ακεραίων. Εάν μία από τις πράξεις που χρησιμοποιείται στους αλγορίθμους είναι η διαίρεση, τότε είναι ανάγκη να δουλέψουμε πάνω σε πεπερασμένη αριθμητική σε ένα πεδίο. Για ευκολία και για αποτελεσματικότητα υλοποίησης, θα ήταν βολικό να δουλέψουμε με ακέραιους που “χωράνε” ακριβώς σε ένα δοσμένο αριθμό bits, χωρίς πλεονασμό σε bit patterns. Βασισμένοι σε αυτό δουλεύουμε με ακεραίους από 0 μέχρι  $2^n - 1$ , που “χωράνε” ακριβώς σε μία n-bit word .

Ας υποθέσουμε ότι θέλουμε να ορίσουμε έναν αλγόριθμο κρυπτογράφησης που ενεργεί πάνω σε data 8 bit κάθε φορά και θέλουμε να πραγματοποιήσουμε μία διαίρεση. Με 8 bits, περιμένουμε να έχουμε αριθμού μεταξύ 0 και 255. Ωστόσο, το 256 δεν είναι πρώτος αριθμός, επομένως η αριθμητική δεν μπορεί να εκτελείται στο  $Z_{256}$  (αριθμητικό modulo 256), οπότε αυτό δεν μπορεί να αποτελεί ένα πεπερασμένο πεδίο. Ο κοντινότερος πρώτος αριθμός μικρότερος στο 256 είναι το 251. Έτσι, το σύνολο  $Z_{251}$  που χρησιμοποιεί αριθμητική modulo 251 είναι ένα πεπερασμένο πεδίο. Ωστόσο, στην περίπτωση που χρησιμοποιείται 8 bit pattern οι αριθμοί 251 μέχρι 255 δεν θα χρησιμοποιούνται και κατά συνέπεια δεν χρησιμοποιείται αποτελεσματικά ο χώρος.

Όπως παρουσιάστηκε στο προηγούμενο παράδειγμα, αν όλες οι αριθμητικές πράξεις θα χρησιμοποιηθούν, επιθυμούμε να χρησιμοποιούμε όλο το εύρος των ακεραίων.

### 2.2.3 Το πεδίο $GF(2^n)$

Το ενδιαφέρον μας επικεντρώνεται στη περίπτωση που το πεδίο έχει μορφή  $GF(2^n)$ . Τέτοιας μορφής πεπερασμένο πεδίο έχουμε στον AES ο οποίος ορίζεται στο  $GF(2^8)$  καθώς και στην GHASH function η οποία ορίζεται στο  $GF(2^{128})$ . Πιο συγκεκριμένα, έχουμε πολλαπλασιασμό πινάκων στον AES στο πεδίο  $GF(2^8)$  που ουσιαστικά πρόκειται για πολλαπλασιασμό 8-bit αριθμών με το 2 και το 3 και πρόσθεση αυτών που ισοδυναμεί με λογική πράξη XOR.

#### 2.2.4. Υπολογιστικοί Παράγοντες

Ένα πολυώνυμο  $f(x)$  στο  $GF(2^n)$  :

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0 = \sum_{i=0}^{n-1} a_i x^i$$

Μπορεί να αναπαρασταθεί μοναδικά με τους συντελεστές  $(a_{n-1}a_{n-2}\dots a_0)$ . Έτσι, κάθε πολυώνυμο στο  $GF(2^n)$  μπορεί να αναπαρασταθεί με έναν  $n$ -bit αριθμό.

#### 2.2.5 Πρόσθεση στο $GF(2^n)$ πεδίο

Είδαμε ότι η προσθήκη πολυωνύμων πραγματοποιείται με την προσθήκη αντίστοιχων συντελεστών, και στην περίπτωση των πολυωνύμων στο  $Z_2$  η πρόσθεση είναι μόνο μια λειτουργία XOR. Έτσι, η πρόσθεση δύο πολυωνύμων στο  $GF(2^n)$  αντιστοιχεί σε μια λειτουργία *bitwise XOR*.

#### 2.2.6 Πολλαπλασιασμός στο $GF(2^n)$ πεδίο

Δεν υπάρχει απλή λογική πράξη αντίστοιχη της XOR για τον πολλαπλασιασμό στο  $GF(2^n)$ . Ωστόσο, μία σχετικά απλή, εύκολα υλοποιήσιμη τεχνική είναι διαθέσιμη. Θα συζητήσουμε την τεχνική όσον αναφορά στο  $GF(2^8)$  χρησιμοποιώντας ως πολυώνυμο διαιρέτη, το αδιαίρετο πολυώνυμο  $m(x) = x^8 + x^4 + x^3 + x + 1$ , που αποτελεί το πεπερασμένο πεδίο που χρησιμοποιείται στον AES. Η τεχνική αυτή εύκολα γενικεύεται για  $GF(2^n)$ .

Η τεχνική βασίζεται στην παρατήρηση ότι:

$$x^8 \bmod m(x) = [m(x) - x^8] = (x^4 + x^3 + x + 1) \quad (2.2.6 \alpha)$$

Σε γενικές γραμμές, στα  $GF(2^n)$  με ένα  $n$ -οστού βαθμού πολυώνυμο  $p(x)$ , έχουμε :  
 $x^n \bmod p(x) = [p(x) - x^n]$ .

Τώρα, ας θεωρήσουμε ένα πολυώνυμο στο  $GF(2^8)$  το οποίο έχει την μορφή:  
 $f(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$ , αν το πολλαπλασιάσουμε με το  $x$  έχουμε :

$$\begin{aligned} x \times f(x) = & (b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + \\ & b_2x^3 + b_1x^2 + b_0x) \bmod m(x) \end{aligned}$$

Αν  $b_7 = 0$ , τότε το αποτέλεσμα του πολλαπλασιασμού είναι βαθμού μικρότερου από 8, που είναι ήδη μειωμένη μορφή και δεν χρειάζεται περεταίρω υπολογισμός. Αν  $b_7 = 1$ , τότε η μείωση modulo  $m(x)$  επιτυγχάνεται χρησιμοποιώντας την 2.2.6.α :

$$x \times f(x) = (b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) + (x^4 + x^3 + x + 1)$$

Επομένως, ο πολλαπλασιασμός με  $x$  (δηλαδή, 00000010) μπορεί να εκφραστεί ως μια 1-bit αριστερή ολίσθηση που ακολουθείται από μια υπό όρους bitwise XOR με (00011011), που αντιπροσωπεύει το  $(x^4 + x^3 + x + 1)$ . Συνοψίζοντας:

$$x \times f(x) = \begin{cases} (b_6 b_5 b_4 b_3 b_2 b_1 b_0 0) & \text{if } b_7 = 0 \\ (b_6 b_5 b_4 b_3 b_2 b_1 b_0 0) \oplus (00011011) & \text{if } b_7 = 1 \end{cases} \quad (2.2.6.\beta)$$

Ο πολλαπλασιασμός με υψηλότερη δύναμη μπορεί να επιτευχθεί με συνεχής επαναλήψεις της 2.2.6.a. Με την προσθήκη ενδιάμεσων αποτελεσμάτων, ο πολλαπλασιασμός με οποιοδήποτε σταθερά στο GF(2<sup>8</sup>) μπορεί να επιτευχθεί.

## 2.3 Μαθηματικές Συνιστώσες του GCM

Έχοντας πλέον παρουσιάσει το απαραίτητο μαθηματικό υπόβαθρο, έτσι όπως αναλύθηκε στην παραπάνω ενότητα, έγινε κατανοητός ο λόγος χρήσης του πεδίου GF. Στην ενότητα αυτή αρχικά, παρουσιάζονται οι απλές πράξεις καθώς και οι συμβολισμοί που θα χρησιμοποιούνται στο εξής ενώ στην συνέχεια παρουσιάζεται η συνάρτηση increment έτσι όπως ορίζεται από το NIST 800-38D και αποτελεί ένα από τα απαραίτητα στοιχεία του GCM. Επιπλέον, παρουσιάζονται οι μαθηματικές πράξεις του AES.

### 2.3.1 Παραδείγματα από τις βασικές πράξεις και λειτουργίες

Στην παρούσα ενότητα αναφέρονται οι βασικές πράξεις που χρησιμοποιούνται στην ανάλυση του AES-GCM καθώς και παραδείγματα αυτών. Επίσης, εδώ παρουσιάζονται και οι συμβολισμοί που χρησιμοποιούνται.

#### 2.3.1.1. Απλές πράξεις- Συμβολισμοί

- Δοσμένου ενός πραγματικού αριθμού  $x$  η *ceiling function* που συμβολίζεται με  $[x]$ , επιστρέφει σαν αποτέλεσμα το μικρότερο ακέραιο αριθμό που δεν είναι μικρότερος από το  $x$ .

Παράδειγμα:

$$\begin{aligned} [2, 1] &= 3 \\ [4] &= 4 \end{aligned}$$

- Δοσμένου ενός θετικού ακεραίου αριθμού  $s$  η πράξη  $0^s$  συμβολίζει μία ακολουθία  $s$  μήκους από ‘0’ bits .Συμβολίζει δηλαδή τον αριθμό των μηδενικών σε ένα bit string .

Παράδειγμα:

$$0^8 = \text{“}00000000\text{”}$$

- Η πράξη *concatenation* (αλληλουχίας) που γίνεται μεταξύ bit strings συμβολίζεται ως  $\|$ . Στην VHDL η πράξη αυτή συμβολίζεται ως “&”. Ουσιαστικά δεν πρόκειται για λογική πράξη μεταξύ bit strings, αλλά για μια απλή συνένωση ακολουθιών bits.

Παράδειγμα:

$$001 \| 10111 = 00110111$$

- Δοσμένων δύο bit string ίσου μήκους η πράξη *exclusive-OR* (XOR) συμβολίζεται  $\square$  . Και είναι η λογική πράξη αποκλειστικό-Η.

Ο πίνακας αληθείας της XOR είναι :

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Είναι μια από της πιο χρήσιμες λογικές πράξεις διότι εκτός από την αυτό καθεαυτό πράξη χρησιμοποιείται και σε άλλες λειτουργίες όπως σύγκριση bit strings ή αντιστροφή της τιμής ενός bit string .

Παράδειγμα:

$$\begin{array}{r} 10011 \{ \text{xor} \} \\ 10101 \\ \hline 00110 \end{array}$$

- Η πράξη  $\text{len}(X)$  επιστρέφει το μήκος του bit string X σε ακέραιο αριθμό. Μας δείχνει δηλαδή πόσα bit έχει ένα bit sting. Είναι μία από τις εντολές που χρησιμοποιείται στους αλγόριθμους που θα παρουσιαστούν παρακάτω.

Παράδειγμα:

$$\begin{aligned} \text{len} (00010) &= 5 \\ \text{len} (11100001 \| 0^{120}) &= 128 \end{aligned}$$

- Δοσμένου ενός bit string X και ενός μη αρνητικού ακεραίου s τέτοιου ώστε  $\text{len}(X) \geq s$  ορίζονται οι πράξεις  $LSB_s(X)$  και  $MSB_s(X)$  οι οποίες επιστρέφουν τα s least significant και s most significant bits αντίστοιχα. Χωρίζει δηλαδή το bit string σε 2 τμήματα με s και  $\text{len}(X)-s$  bits και μας επιστρέφει τα s MSB ή LSB.

Παράδειγμα :

$$\begin{aligned} LSB_3(111011010) &= 010 \\ MSB_4(111011010) &= 1110 \end{aligned}$$

- Δοσμένου ενός bit string X, η (single) right-shift function συμβολίζεται ως  $X>>I$  ή ως  $\text{rightshift}(X)$ .

Παράδειγμα:

$$\begin{aligned} 0110111 >> 1 &= 0011011 \\ \text{rightshift}(0110111) &= 0011011 \end{aligned}$$

Υπάρχει επίσης και right και left shift function πολλών bit.

Παράδειγμα:

$$\begin{aligned} 00111101 >> 4 &= 00000011 \\ 01001110 << 3 &= 01110000 \end{aligned}$$

Οι θέσεις bit που έχουν γίνει shift γεμίζονται με μηδενικά και στις δύο περιπτώσεις. Κάτι τέτοιο όμως δεν θα ίσχυε αν είχαμε να κάνουμε με προσημασμένους ή αρνητικούς αριθμούς. Κάτι τέτοιο εδώ δεν μας ενδιαφέρει γιατί όλα τα bit string είναι ακέραιοι θετικοί, μη προσημασμένοι. Ο ίδιος συμβολισμός πράξης χρησιμοποιείται και στην C γλώσσα προγραμματισμού.

- Δοσμένου ενός θετικού ακεραίου s και ενός μη αρνητικού ακεραίου X που είναι μικρότερος από  $2^s$ , ορίζεται η συνάρτηση *integer-to-string* function και συμβολίζεται με  $[X]_s$ . Θα πρέπει όμως να ισχύει  $X < 2^s$  γιατί αλλιώς θα έχουμε έλλειψη – αλλοίωση πληροφορίας.

Παράδειγμα:

$$\begin{aligned} [39]_8 &= 00100111 \\ [39]_6 &= 100111 \end{aligned}$$

- Παρόμοια ορίζεται και η συνάρτηση *string-to-integer* για μη αρνητικούς ακέραιους.

Παράδειγμα:

$$\text{int}(00011010) = 26$$

Ο συμβολισμός που χρησιμοποιείται για να αναφερόμαστε σε hexadecimal αριθμούς στην C γλώσσα είναι 0x45 ή 0xff.

### 2.3.1.2. Incrementing Function

Για ένα θετικό ακέραιο  $s$  και ένα bit string  $X$  τέτοιο ώστε  $\text{len}(X) \geq s$  ορίζεται η  $s$ -bit increment function και συμβολίζεται με  $\text{inc}_s(X)$ . Η μαθηματική έκφραση της συνάρτησης είναι:

$$\text{inc}_s(X) = \text{MSB}_{\text{len}(X)-s}(X) \parallel [\text{int}(\text{LSB}_s(X)) + 1 \bmod 2^s]_s$$

Παράδειγμα :

$$\begin{aligned} \text{inc}_5(01110110110) &= 011101 \parallel [\text{int}(10110) + 1 \bmod 2^5]_5 = \\ &= 011101 \parallel [22+1 \bmod 32]_5 = \\ &= 011101 \parallel [23 \bmod 32]_5 = \\ &= 011101 \parallel [23]_5 = \\ &= 011101 \parallel 10111 = 1110110111 \end{aligned}$$

Με άλλα λόγια η  $\text{inc}_s(X)$  αυξάνει κατά μία μονάδα τα  $\text{LSB}_s(X)$  αφήνοντας τα υπόλοιπα  $\text{len}(X) - s$  MSB ανεπηρέαστα. Το τελικό αποτέλεσμα είναι το concatenation των δύο bit sting. Σε περίπτωση που έχουμε overflow τότε το αποτέλεσμα της πρόσθεσης mod  $2^s$  είναι η μονάδα, άρα το τελικό αποτέλεσμα της  $\text{inc}_s(X)$  είναι :

$$0011\dots1 \parallel 0000\dots01$$

Παράδειγμα:

$$\begin{aligned} \text{inc}_5(1110111111) &= 11101 \parallel [\text{int}(11111) + 1 \bmod 2^5]_5 = \\ &= 11101 \parallel [32+1 \bmod 32]_5 = \\ &= 11101 \parallel [33 \bmod 32]_5 = \\ &= 11101 \parallel [1]_5 = \\ &= 11101 \parallel 00001 = 1110100001 \end{aligned}$$

Πρόκειται δηλαδή για μοναδιαία κυκλική αύξηση. Όταν φτάσει στο τέλος αρχίζει πάλι από την αρχή. Είναι μια χρήσιμη έκφραση αφού μας εγγυάται ότι δεν θα έχουμε overflow και ότι το κρατούμενο διαδίδεται σε συγκεκριμένο αριθμό bit. Μας επιτρέπει έτσι να εκμεταλλευτούμε διάφορες τεχνικές υλοποίησης για να πετύχουμε μικρούς χρόνους καθυστέρησης. Ουσιαστικά το carry out τροφοδοτεί το carry in αν η πράξη υλοποιείται με adder. Σε FPGA η πράξη αυτή δεν υλοποιείται με adder αλλά με counter αφού η πρόσθεση που γίνεται είναι πάντα γνωστή κι έτσι μπορεί να προϋπολογίζεται η τιμή και επομένως η καθυστέρηση που δίνει στον συνολικό κύκλωμα να μην είναι σημαντική.

## 2.3.2 Μαθηματικές Συνιστώσες του AES-CTR

Τα μαθηματικά στοιχεία του AES σε CTR mode είναι δύο. Πρόκειται για το S-box και για έναν πολλαπλασιασμό στο  $GF(2^8)$  με το 2 και το 3.

### 2.3.2.1. S-box

Το S-box είναι ένα LUT (look up table). Για την κατασκευή του όμως πρέπει να γίνουν πολλαπλασιασμοί στο  $GF(2^8)$  και να βρεθούν οι αντίστροφοι όλων των αριθμών από  $00_H$  μέχρι  $FF_H$ . Στην συνέχεια γίνεται ένας πολλαπλασιασμός πινάκων απ' όπου προκύπτει τελικά το s-box. Πρόκειται για μια διαδικασία χρονοβόρα και απαιτείται μεγάλος αριθμός πράξεων για να κατασκευαστεί. Κάτι τέτοιο όμως δεν απαιτείται σε όλες τις περιπτώσεις αφού πρόκειται για ένα LUT που οι τιμές του είναι γνωστές. Περισσότερες πληροφορίες για το s-box αναφέρονται σε επόμενο κεφάλαιο που αναλύεται ο AES.

### 2.4.2.2. Πολλαπλασιασμός στο $GF(2^8)$

Στον AES σε κάποιο στάδιο της διαδικασίας κρυπτογράφησης και αποκρυπτογράφησης απαιτείται να γίνει ένας πολλαπλασιασμός των 8 bit string τιμών με το 2 και με το 3. Ο πολλαπλασιασμός γίνεται στο  $GF(2^8)$ . Το αδιαίρετο πολυώνυμο που χρησιμοποιείται για τον AES είναι  $m(x) = x^8 + x^4 + x^3 + x + 1$  το οποίο σε δυαδική μορφή των 8 bits εκφράζεται ως  $00011011$ . Ο πολλαπλασιασμός με το 2 και το 3 ενός αριθμού γίνεται με βάση την σχέση:

$$x \times f(x) = \begin{cases} (b_6 b_5 b_4 b_3 b_2 b_1 b_0 0) & \text{if } b_7 = 0 \\ (b_6 b_5 b_4 b_3 b_2 b_1 b_0 0) \oplus (00011011) & \text{if } b_7 = 1 \end{cases}$$

Έχουμε δηλαδή:

Το  $b$  έχει μήκος 8 bit.

Αν  $b(7)$  (το 8<sup>ο</sup> MSB) είναι 0 τότε :

$$b^*2 = b_6 \parallel b_5 \parallel b_4 \parallel b_3 \parallel b_2 \parallel b_1 \parallel 0$$

Αλλιώς :

$$b^*2 = (b_6 \parallel b_5 \parallel b_4 \parallel b_3 \parallel b_2 \parallel b_1 \parallel 0) \text{ XOR } (00011011)$$

Ο πολλαπλασιασμός με το 3 γίνεται ως εξής :

$$b^*3 = b^*(2+1) = b^*2 + b \rightarrow b^*2 \text{ XOR } b$$

Αφού, όπως έχει αναφερθεί, η πρόσθεση στο  $GF(2^8)$  αντιστοιχεί με την λογική πράξη XOR.

## 2.4 Εμπιστευτικότητα και Τρόποι Λειτουργίας

Γενικότερα, ο αλγόριθμος AES μπορεί να χρησιμοποιηθεί για την κρυπτογράφηση και αποκρυπτογράφηση ενός μπλοκ δεδομένων σταθερού μεγέθους. Παρόλα αυτά, στην πράξη τα πραγματικά μηνύματα σπάνια έχουν τη μορφή μπλοκ σταθερού μεγέθους. Για παράδειγμα, τα δεδομένα σε ένα ασύρματο δίκτυο μεταδίδονται σε πλαίσια με διάφορα μήκη, που ποικίλουν από 512 ως 12000 bits σε κάθε πλαίσιο. Επομένως, για να χρησιμοποιήσουμε ένα μπλοκ αλγόριθμο κρυπτογράφησης, όπως ο AES, πρέπει να καθορίσουμε μία μέθοδο ώστε τα μηνύματα με τυχαίο μήκος να μετατρέπονται σε μία ακολουθία από μπλοκ σταθερού μήκους πριν την κρυπτογράφηση. Επιπλέον, η μέθοδος αυτή θα πρέπει να επιτρέπει την επανένωση των μπλοκ, ώστε να προκύψει το αρχικό μήνυμα, κατά τη διάρκεια της αποκρυπτογράφησης. Η μέθοδος που χρησιμοποιείται για τη μετατροπή από τα μηνύματα στα μπλοκ και το αντίστροφο ονομάζεται κατάσταση λειτουργίας (*mode of operation*) των μπλοκ αλγόριθμου.

Υπάρχουν αρκετές καταστάσεις λειτουργίας που μπορούν να χρησιμοποιηθούν σε συνδυασμό με τον AES. Το NIST έχει καθορίσει μία λίστα με δεκαέξι διαφορετικές προσεγγίσεις και είναι ανοικτό σε νέες προτάσεις. Η επιλογή της κατάστασης λειτουργίας είναι πολύ σημαντικό θέμα, καθώς έχει επιπτώσεις τόσο στην πολυπλοκότητα της υλοποίησης, όσο και στην ασφάλεια του τελικού πρωτοκόλλου ασφαλείας. Κακές καταστάσεις λειτουργίας μπορούν να δημιουργήσουν προβλήματα στην ασφάλεια του δικτύου, παρόλο που ο αλγόριθμος AES είναι πολύ ισχυρός.

Δύο καταστάσεις λειτουργίας για συμμετρικού κλειδιού Block Ciphers, ο ECB και ο CTR, επιλέγονται για να σχηματίσουν τον μηχανισμό της εμπιστευτικότητας στον AES-GCM, καθώς αυτοί μπορούν να εισάγουν υλοποιήσεις με τεχνικές διοχέτευσης και παραλληλισμού ενώ επίσης έχουν ελάχιστο υπολογιστικό κόστος για high data rates. Αντές οι καταστάσεις παρουσιάζονται παρακάτω και για περισσότερες λεπτομέρειες παραπέμπουμε στο [1].

Γενικά, η κατάσταση λειτουργίας τοποθετείται ως ένδειξη δίπλα από τη λέξη AES. Επομένως, ένα σύστημα που χρησιμοποιεί το ECB, χαρακτηρίζεται ως AES/ECB.

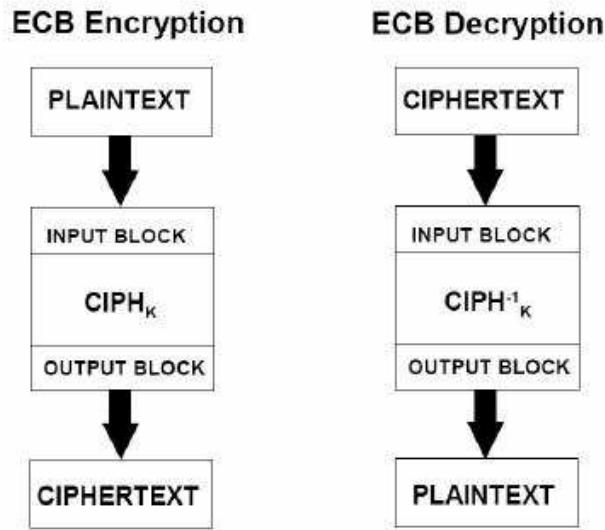
### 2.4.1 Electronic Codebook Mode (ECB)

To ECB mode ορίζεται ως ακολούθως και φαίνεται στην **Εικόνα 1**:

ECB Encryption:  $C_j = \text{CIPH}_K(P_j)$  for  $j = 1 \dots n$ ,

ECB Decryption:  $P_j = \text{CIPH}^{-1}_K(C_j)$  for  $j = 1 \dots n$ ,

Όπου το  $\text{CIPH}_K(P_j)$  είναι η forward cipher function του block cipher αλγόριθμου, δηλαδή του AES, υπό το κλειδί K εφαρμοσμένο στο plaintext  $P_j$ . Το  $\text{CIPH}^{-1}_K(C_j)$  είναι η inverse cipher function του block cipher αλγόριθμου υπό το κλειδί K εφαρμοσμένο στο ciphertext  $C_j$ .

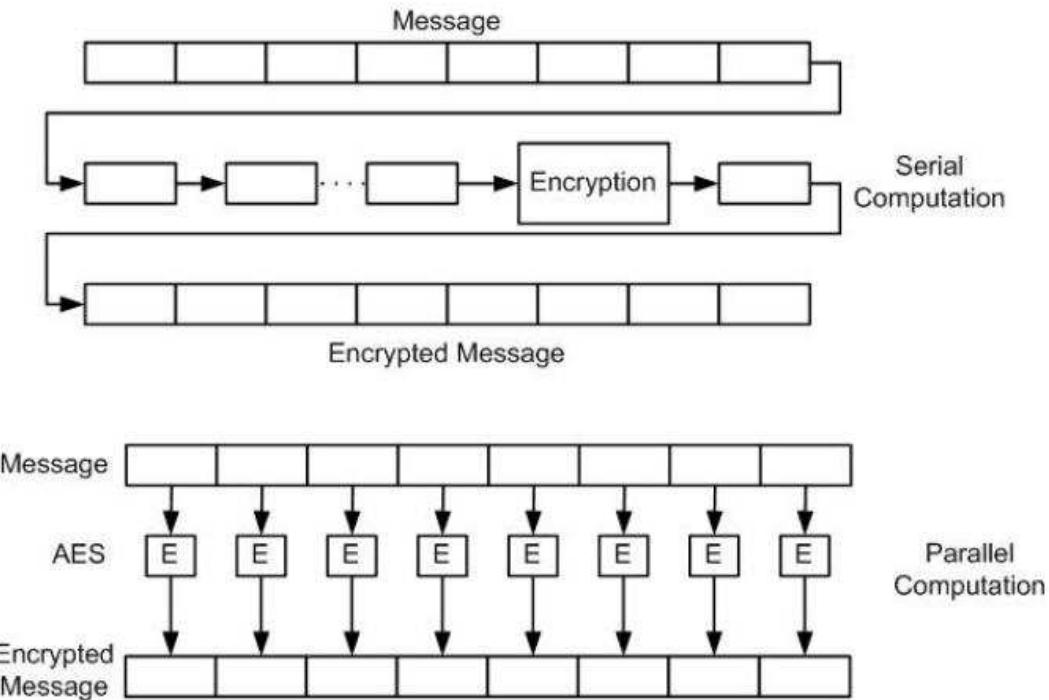


**Εικόνα 1. ECB Encryption and ECB Decryption [1]**

Στην ECB κρυπτογράφηση και ECB αποκρυπτογράφηση, πολλαπλές forward cipher functions και inverse cipher functions μπορούν να εκτελούνται σε parallel ή pipelined αρχιτεκτονική. Στο GCTR module του AES-GCM, το ECB κρυπτογραφικό block εμπεριέχεται στο CTR block.

Η κατάσταση λειτουργίας ECB απλά παίρνει ένα τμήμα του μηνύματος, ένα μπλοκ κάθε φορά και κρυπτογραφεί κάθε μπλοκ ακολουθιακά χρησιμοποιώντας το ίδιο κλειδί μέχρι το τελευταίο τμήμα του μηνύματος. Η διαδικασία αυτή φαίνεται στην Εικόνα 2, που δείχνει τον υπολογισμό τόσο για τη σειριακή, δηλαδή ένα μπλοκ κάθε φορά, όσο και για την παράλληλη κρυπτογράφηση.

Η προσέγγιση αυτή φαίνεται απλή, αλλά παρουσιάζει ορισμένα προβλήματα. Το πιο εμφανές πρόβλημα είναι ότι το μήνυμα μπορεί να μην έχει μέγεθος ακριβώς πολλαπλάσιο του μεγέθους του μπλοκ και πρέπει να συμπληρώσουμε με μηδενικά (padding) το τελευταίο μπλοκ και να θυμόμαστε το πραγματικό μήκος. Υπάρχει όμως και ένα πρόβλημα στην ασφάλεια. Αν δύο μπλοκ περιέχουν τα ίδια δεδομένα, το αποτέλεσμα της κρυπτογράφησης θα περιέχει επίσης δύο ίδια μπλοκ, αποκαλύπτοντας πληροφορίες σε κάποιο εξωτερικό παρατηρητή του δικτύου.



**Εικόνα 2. Η κατάσταση λειτουργίας ECB.**

Θεωρείστε ένα μήνυμα που αποτελείται από μία γραμματοσειρά, όπου το ίδιο γράμμα επαναλαμβάνεται 64 φορές, για παράδειγμα “ΑΑΑΑΑΑ...”. Αν το μήκος μπλοκ του AES είναι 128 bits, δηλαδή 16 bytes, τότε χρησιμοποιώντας το ECB το μήνυμα θα σπάσει σε τέσσερα μπλοκ καθένα από τα οποία περιέχει 16 γράμματα A. Μετά την κρυπτογράφηση τα τέσσερα μπλοκ θα μετατραπούν σε τέσσερα πανομοιότυπα κρυπτογραφημένα μπλοκ. Με τον τρόπο αυτό, ο εξωτερικός παρατηρητής πληροφορείται ότι αυτό το μήνυμα περιέχει στοιχεία που επαναλαμβάνονται.

Εξαιτίας αυτής της αδυναμίας, καθώς και άλλων, τα συστήματα ασφαλείας στην πράξη δε χρησιμοποιούν την κατάσταση λειτουργίας ECB. Δε βρίσκεται καν στη λίστα με τις προτεινόμενες καταστάσεις λειτουργίας του NIST. Ακόμα και ο πιο ισχυρός μπλοκ αλγόριθμος κρυπτογράφησης δεν μπορεί να παρέχει την επιθυμητή ασφάλεια, αν υπάρχουν αδύνατα σημεία στην κατάσταση λειτουργίας.

#### 2.4.2 Counter Mode (CTR)

Το Counter mode είναι ένα mode που προσφέρει εμπιστευτικότητα και περιγράφει την εφαρμογή του block cipher αλγόριθμου σε ένα set group από δεδομένα εισόδου, που ονομάζονται counters, με σκοπό την παραγωγή ενός set group από key streams τα οποία γίνονται XORed με το plaintext ώστε να παραχθεί το ciphertext και αντίστροφα. Το CTR mode περιγράφεται ως ακολούθως και φαίνεται στην **Εικόνα 3**:

### CTR Encryption:

$O_j = \text{CIPH}_K(T_j)$  for  $j = 1, 2 \dots n$ ;

$C_j = P_j \text{XOR } O_j$  for  $j = 1, 2 \dots n-1$ ;

$C_n^* = P_n^* \text{XOR } \text{MSB}_u(O_n)$ .

### CTR Decryption:

$O_j = \text{CIPH}_K(T_j)$  for  $j = 1, 2 \dots n$ ;

$P_j = C_j \text{XOR } O_j$  for  $j = 1, 2 \dots n-1$ ;

$P_n^* = C_n^* \text{XOR } \text{MSB}_u(O_n)$

Τα σύμβολα που χρησιμοποιούνται στις δύο παραπάνω διεργασίες είναι:

$T_j \rightarrow$  οι counter για το jth group δεδομένων εισόδου,

$O_j \rightarrow$  το key stream για το jth group δεδομένων εισόδου,

$P_j \rightarrow$  το jth plaintext group,

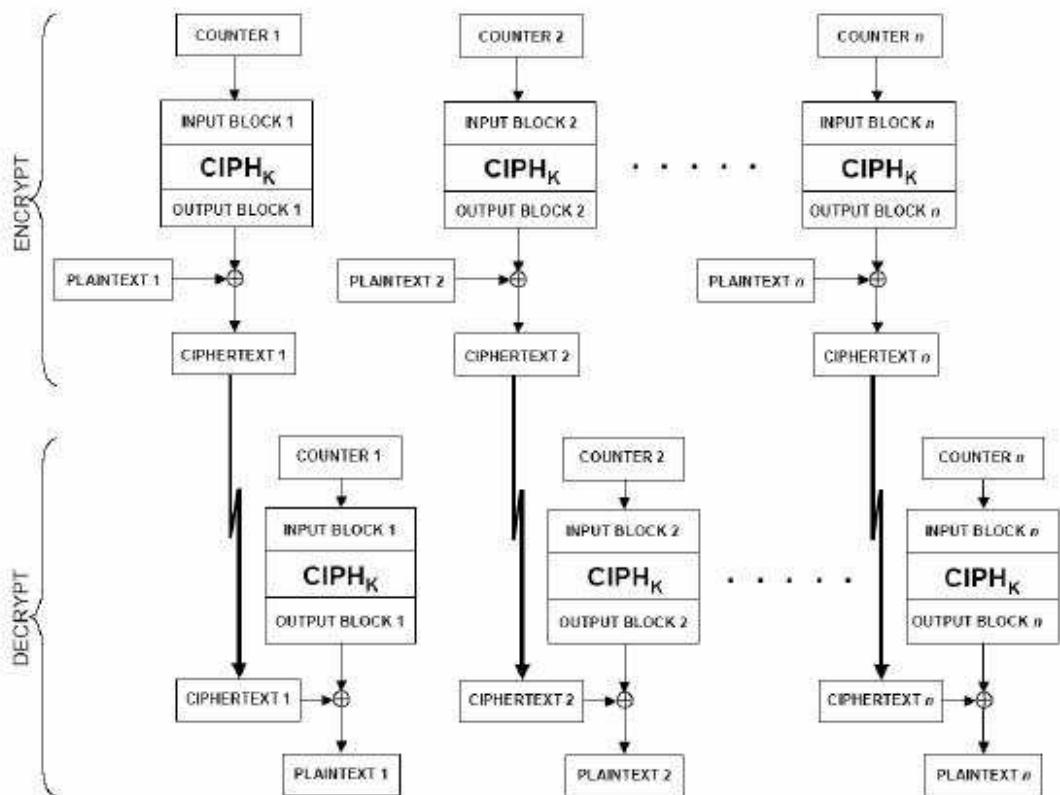
$C_j \rightarrow$  το jth ciphertext group,

$C_n^* \rightarrow$  το τελευταίο group του ciphertext το οποίο μπορεί να είναι ένα partial group,

$P_n^* \rightarrow$  το τελευταίο group του plaintext το οποίο μπορεί να είναι ένα partial group,

$\text{MSB}_u(O_n) \rightarrow$  bit string που αποτελείται από τα u πιο σημαντικά bits του bit string  $O_n$ .

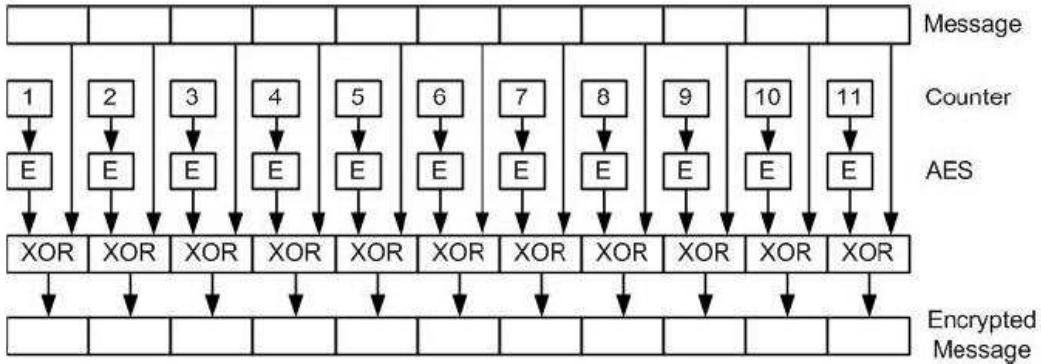
Στην CTR κρυπτογράφηση και αποκρυπτογράφηση, μόνο η *forward cipher function* ενεργοποιείται σε κάθε counter group και όχι η *inverse cipher function*. Αυτό συμβαίνει επειδή τα *termatiká key streams* γίνονται *XORed* με τα *antistoiχa plaintext* ή *ciphertext blocks* για να παράγουν τα ciphertext ή plaintext blocks, και όπως γνωρίζουμε η λογική πράξη *XOR* είναι από μόνη της μια *antistrophi* διεργασία. Για το τελευταίο group το οποίο δύναται να είναι partial group των u bits, χρησιμοποιούνται τα u πιο σημαντικά bits του τελευταίου group εξόδου για την *XOR* λειτουργία. Τα εναπομείναντα bits του τελευταίου group εξόδου παραλείπονται. Η *forward cipher function* μπορεί να εκτελεστεί τόσο σε parallel όσο και σε pipelined αρχιτεκτονική.



**Εικόνα 3. CTR Encryption and CTR Decryption [1]**

Τόσο η CTR κρυπτογράφηση και η CTR αποκρυπτογράφηση εκτελούνται στην AES-GCM κρυπτογράφηση και AES-GCM αποκρυπτογράφηση, αντίστοιχα.

Η κατάσταση λειτουργίας μετρητή (counter mode) είναι πιο πολύπλοκη από την ECB και λειτουργεί με αρκετά διαφορετικό τρόπο. Η διαφορά έγκειται στο ότι δεν χρησιμοποιεί τον αλγόριθμο AES απευθείας για την κρυπτογράφηση των δεδομένων. Αντίθετα, κρυπτογραφεί την τυχαία τιμή ενός μετρητή και στη συνέχεια εφαρμόζει την πράξη XOR ανάμεσα στο αποτέλεσμα της κρυπτογράφησης και τα δεδομένα. Έτσι, με τον τρόπο αυτό προκύπτει το κρυπτογραφημένο κείμενο, όπως φαίνεται στην **Εικόνα 4**. Ο μετρητής γενικά αυξάνεται κατά 1 με κάθε διαδοχικό μπλοκ που κρυπτογραφείται.



**Εικόνα 4. Η κατάσταση λειτουργίας CTR.**

Το μήνυμα χωρίζεται σε μπλοκ και το κρυπτογραφημένο κείμενο είναι το αποτέλεσμα της πράξης XOR ανάμεσα στο μπλοκ δεδομένων και της κρυπτογραφημένης τιμής του μετρητή από τον AES. Στην **Εικόνα 4** ο μετρητής ξεκινάει από την τιμή 1 και αυξάνεται μέχρι την τιμή 11. Στην πράξη ο μετρητής μπορεί να ξεκινάει από μία τυχαία τιμή και να αυξάνεται κατά μία άλλη τιμή. Το σημαντικό είναι ο παραλήπτης του μηνύματος, που επιθυμεί να αποκρυπτογραφήσει το κείμενο, να γνωρίζει την αρχική τιμή του μετρητή καθώς και το βήμα της αύξησης.

Επειδή ο μετρητής αλλάζει τιμή για κάθε μπλοκ, αποφεύγεται το πρόβλημα που εμφανίζεται στην κατάσταση λειτουργίας ECB με τα επαναλαμβανόμενα μπλοκ. Ακόμα και αν δύο μπλοκ δεδομένων είναι πανομοιότυπα, θα συνδυαστούν με διαφορετική τιμή του μετρητή και θα προκύψει διαφορετικό αποτέλεσμα για κάθε μπλοκ. Παρόλα αυτά, με τον τρόπο που παρουσιάζεται, η μέθοδος αυτή έχει ως αποτέλεσμα δύο ίδια, αλλά ξεχωριστά μηνύματα, να παρουσιάζουν το ίδιο αποτέλεσμα μετά την κρυπτογράφηση. Για το λόγο αυτό, στην πράξη, ο μετρητής δεν ξεκινάει από την τιμή 1. Τυπικά, αρχικοποιείται με μία τιμή η οποία αλλάζει για κάθε διαδοχικό μήνυμα (nonce).

**Η κατάσταση λειτουργίας μετρητή παρουσιάζει μερικές ενδιαφέρουσες ιδιότητες :**

- Η αποκρυπτογράφηση είναι ακριβώς η ίδια διαδικασία με την κρυπτογράφηση, καθώς η εφαρμογή της πράξης XOR δύο φορές στα ίδια δεδομένα επιστρέφει τα αρχικά δεδομένα. Αυτό σημαίνει ότι οι διάφορες υλοποιήσεις πρέπει να υλοποιήσουν μόνο τον αλγόριθμο κρυπτογράφησης και όχι τον αλγόριθμο αποκρυπτογράφησης.
- Ένα άλλο πολύ χρήσιμο χαρακτηριστικό, για ορισμένες εφαρμογές, είναι ότι η κρυπτογράφηση μπορεί να πραγματοποιηθεί παράλληλα. Εφόσον όλες οι τιμές του μετρητή είναι γνωστές εκ των προτέρων, μπορούμε να έχουμε μία τράπεζα από μονάδες κρυπτογράφησης AES ώστε να κρυπτογραφήσουμε ολόκληρο το μήνυμα σε ένα μόνο πέρασμα. Αυτό δεν ισχύει για πολλές από τις υπόλοιπες καταστάσεις λειτουργίας.

- Η τελευταία χρήσιμη ιδιότητα είναι ότι δεν υπάρχει πρόβλημα αν το μήνυμα δεν αποτελείται από ακριβή αριθμό μπλοκ. Απλά παίρνουμε το τελευταίο (μη συμπληρωμένο) μπλοκ και εφαρμόζουμε την πράξη XOR με την κρυπτογραφημένη τιμή του μετρητή, χρησιμοποιώντας μόνο τον αριθμό από bits που είναι αναγκαίος. Επομένως, το μήκος του κρυπτογραφημένου μηνύματος μπορεί να είναι το ίδιο ακριβώς με το αρχικό μήνυμα. Επειδή κάθε πράξη σε μπλοκ εξαρτάται από την κατάσταση του μετρητή από το προηγούμενο μπλοκ, η κατάσταση λειτουργίας μετρητή αποτελεί ουσιαστικά, κρυπτογράφηση ροής (stream cipher).

Η κατάσταση λειτουργίας μετρητή έχει χρησιμοποιηθεί για περισσότερο από είκοσι χρόνια, είναι ευρύτατα γνωστή και χαίρει της εμπιστοσύνης της κοινότητας των κρυπτογράφων. Η απλότητά της και η ωριμότητά της, καθιστούν αυτή την κατάσταση λειτουργίας ως μια ελκυστική επιλογή. Παρόλα αυτά, η κατάσταση λειτουργίας μετρητή δεν προστατεύει καθόλου την ακεραιότητα των μηνυμάτων.

## 2.5 Field Programmable Gate Arrays (FPGA)

Σε αυτήν την διπλωματική παρουσιάζεται η αρχιτεκτονική μιας FPGA υλοποίησης του AES-GCM. Οι πιο γνωστές εφαρμογές υλοποίησης αντιστοιχούν σε τρεις διαφορετικές τεχνολογίες. Αυτές είναι:

- Application Specific Integrated Circuits (ASICs)
- Software-Programmed General Purpose CPU (SPGPC)
- Field Programmable Gate Arrays (FPGAs)

Τα ASICs είναι ειδικά σχεδιασμένα για μια καθορισμένη λύση και επομένως είναι πολύ λειτουργικά. Ωστόσο, το κύκλωμα δεν μπορεί να αλλάξει μετά την κατασκευή. Αυτό απαιτεί έναν επανασχεδιασμό του chip εάν πρέπει να γίνουν τυχόν τροποποιήσεις.

Τα SPGPC είναι μια ευέλικτη λύση. Η CPU εκτελεί ένα σετ εντολών για να εκτελέσει έναν αλγόριθμο. Άλλαζοντας τον κώδικα σε software, τροποποιείται και η λειτουργία του συστήματος χωρίς να επεμβαίνουμε στο hardware. Όμως η αποτελεσματικότητα και η απόδοση του SPGPC είναι πολύ χαμηλότερη από αυτή του ASIC.

Τα FPGAs προσφέρουν μια ενδιάμεση λύση πετυχαίνοντας υψηλότερη απόδοση από αυτήν του software ενώ ταυτόχρονα διατηρούν ένα πιο υψηλό επίπεδο ευελιξίας σε σχέση με το hardware.

## **2.5.1 Πλεονεκτήματα των FPGAs στις Κρυπτογραφικές Εφαρμογές**

Τα ακόλουθα χαρακτηριστικά της FPGA τεχνολογίας προσφέρουν ιδιαίτερα πλεονεκτήματα στις κρυπτογραφικές εφαρμογές:

### **Algorithm Agility (Αλγορίθμική Ευκινησία)**

Όλο και περισσότερες εφαρμογές ασφάλειας τείνουν στο να γίνουν ανεξάρτητες αλγόριθμου και επιτρέπουν την αλλαγή των αλγόριθμων κρυπτογράφησης οποιαδήποτε στιγμή. Ο αλγόριθμος κρυπτογράφησης μπορεί να επιλεγεί ύστερα από συνεννόηση μεταξύ των δύο εμπλεκόμενων φορέων της επικοινωνίας.

### **Algorithm Upload ('Ανέβασμα' Αλγορίθμου)**

Από κρυπτογραφική άποψη, το algorithm upload μπορεί να είναι χρήσιμο επειδή ο τρέχον αλγόριθμος μπορεί να είναι ξεπερασμένος ή 'σπασμένος'. Έτσι ένας καινούριος αναβαθμισμένος αλγόριθμος δημιουργείται. Ο σχεδιαστής μπορεί να 'ανεβάσει' τα καινούρια bit streams του πρωτοκόλλου ασφαλείας και να επανασχεδιάσει την FPGA συσκευή μέσω του δικτύου.

### **Throughput (Απόδοση)**

Παρόλο που οι FPGA υλοποιήσεις είναι πιο αργές από τις ASIC υλοποιήσεις, οι πρώτες είναι σαφώς πιο γρήγορες από τις software υλοποιήσεις. Σε ένα κρυπτογραφικό σύστημα, εάν μια software λύση έχει επιλεγεί, τότε μια FPGA υλοποίηση θα πρέπει να εφαρμοστεί για τους servers.

### **Cost efficiency (Αποδοτικότητα Κόστους)**

Το κόστος παραγωγής ενός ASIC είναι συνήθως πολύ υψηλό για έναν μικρό αριθμό servers στα συστήματα ασφαλείας. Έτσι, η χρήση FPGAs είναι μια κοινή εναλλακτική λύση. Εξάλλου, αυτός είναι ένας από τους λόγους για τους οποίους το FPGA επιλέγεται για την έρευνα στον τομέα της ασφάλειας σε ιδρύματα και πανεπιστήμια.

Επομένως, συχνά είναι καλύτερα να επιλέγεται ένα FPGA για την υλοποίηση ενός cipher, όπως το AES-GCM πρότυπο που εξετάζουμε στην διπλωματική αυτή.

Μια μικρή κατηγορία FPGAs είναι διαθέσιμη για εφαρμογές χαμηλής κατανάλωσης ενέργειας. Υπάρχουν FPGAs όπου χρησιμοποιούν ασύγχρονες μεθόδους σχεδιασμού και gated clock τεχνικές. Γενικότερα οι σχεδιαστές των FPGAs προσπαθούν να ενσωματώσουν τεχνικές κατανάλωσης ενέργειας που εφαρμόζονται στα ASICs, όπως αυτά θα παρουσιαστούν παρακάτω .

## **2.5.2 ASIC Προσέγγιση**

Ιδιαίτερη έμφαση έχει δοθεί στο σχεδιασμό ASIC με χαρακτηριστικά χαμηλής κατανάλωσης ενέργειας. Ορισμένες τυπικές μέθοδοι ελάττωσης της κατανάλωσης ισχύος είναι οι εξής:

### **Απομόνωση ρολογιού (Clock Gating)**

Επιτυγχάνεται σημαντική ελάττωση στην κατανάλωση ενέργειας. Κομμάτια λογικής, τα οποία δε λειτουργούν για έναν οι περισσότερους κύκλους ρολογιού, αποκόπτονται από το σήμα ρολογιού και αδρανοποιούνται.

### **Ασύγχρονος σχεδιασμός**

Ο σχεδιασμός ασύγχρονων συστημάτων (με σήματα χειραψίας - handshaking) περιορίζει την κατανάλωση ενέργειας και αυξάνει την απόδοση του συστήματος. Το πρόβλημα στην περίπτωση αυτή είναι η έλλειψη εργαλείων για την υποστήριξη του σχεδιασμού αυτού.

### **Γεννήτρια μεταβλητής τάσης**

Σε συστήματα που έχουν περιθώρια για μείωση της απόδοσης υπάρχει η δυνατότητα να μειώνουμε την εξωτερική τάση. Με μείωση της εξωτερικής τάσης η κατανάλωση ενέργειας υποτετραπλασιάζεται.

### **Μείωση των ανεπιθύμητων φορτίσεων-εκφορτίσεων (glitches)**

Η μείωση των glitches, δηλαδή των μη επιθυμητών φορτίσεων και εκφορτίσεων των τρανζίστορ, μπορεί να γίνει με σωστό σχεδιασμό. Συνήθως γίνονται αναδιατάξεις σε επίπεδο πυλών και επιλέγονται κατάλληλες κωδικοποιήσεις δεδομένων.

### **Παραλληλισμός και Διοχέτευση**

Και οι δύο μέθοδοι αυξάνουν την απόδοση του συστήματος. Πολλές φορές χρησιμοποιούνται ώστε να αντισταθμίσουν την ελάττωση της απόδοσης εξαιτίας της εφαρμογής της μεθόδου της μείωσης της εξωτερικής τάσης.

### **Βελτιστοποιήσεις Συναρτήσεων**

Επανασχεδιασμός συναρτήσεων με περισσότερη απόδοση στο θέμα της κατανάλωσης ισχύος.

### 2.5.3 FPGA VirtexII xc2v1000

Ένα παραδοσιακό FPGA είναι συνήθως ένα ολοκληρωμένο κύκλωμα που αποτελείται από :

- Configurable Logic Blocks (CLBs),
- Input / Output Blocks (IOBs) και
- Programmable routing resources.

Πιο συγκεκριμένα, ο **Table 1** δείχνει όλους τους κύριους πόρους της VirtexII xc2v1000.

*Table 1: Virtex-II Field-Programmable Gate Array Family Members*

Device	System Gates	CLB (1 CLB = 4 slices = Max 128 bits)			Multiplier Blocks	SelectRAM Blocks		DCMs	Max I/O Pads <sup>(1)</sup>
		Array Row x Col.	Slices	Maximum Distributed RAM Kbits		18 Kbit Blocks	Max RAM (Kbits)		
XC2V40	40K	8 x 8	256	8	4	4	72	4	88
XC2V80	80K	16 x 8	512	16	8	8	144	4	120
XC2V250	250K	24 x 16	1,536	48	24	24	432	8	200
XC2V500	500K	32 x 24	3,072	96	32	32	576	8	264
XC2V1000	1M	40 x 32	5,120	160	40	40	720	8	432

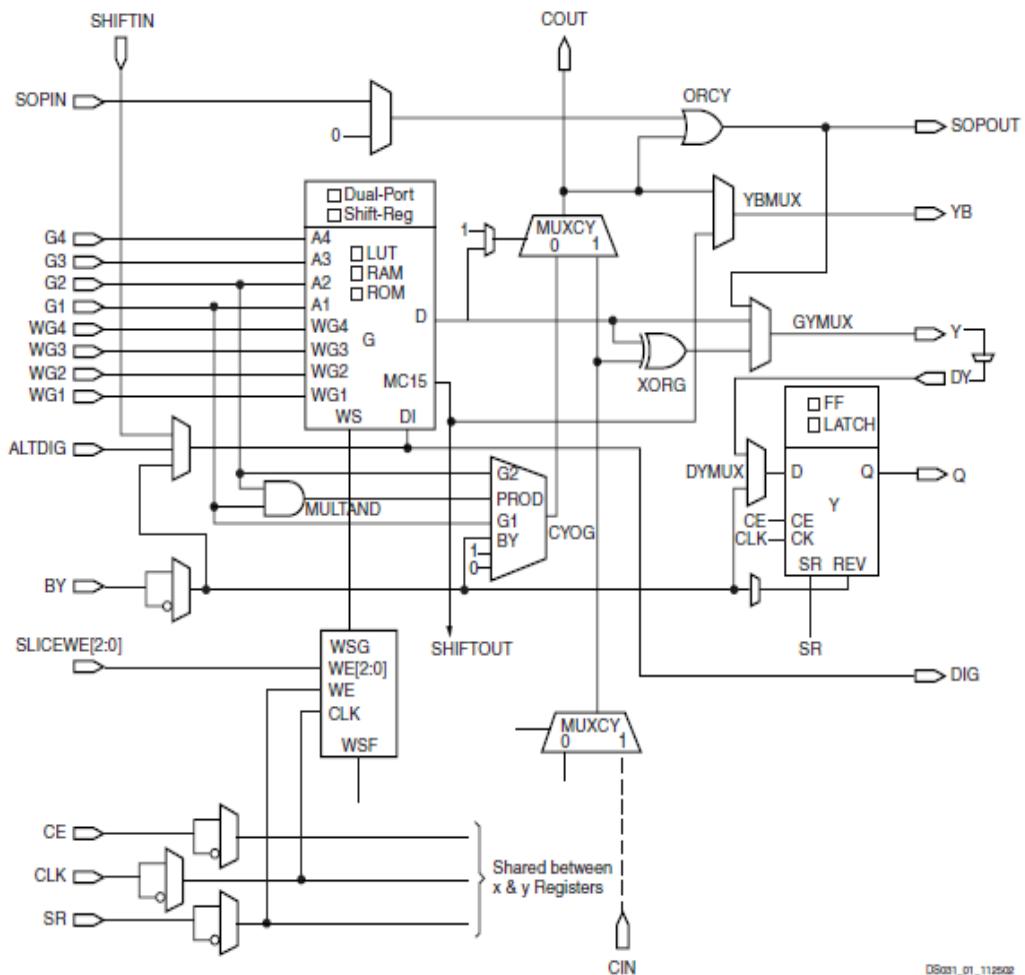
### 2.5.4 Configurable Logic Blocks (CLBs)

Τα CLBs στην VirtexII αποτελούνται από συνδυαστική (combinational) και ακολουθιακή λογική. Η combinational λογική μπορεί να ρυθμιστεί ώστε να καταστεί δυνατή Boolean λειτουργία. Τα Flip-flops είναι η προϋπόθεση για την υποστήριξη της ακολουθιακής λογικής και μπορούν να χρησιμοποιηθούν ή να παρακαμφθούν ανάλογα με τη ρύθμιση. Ένα CLB έχει τέσσερις φέτες. Κάθε κομμάτι είναι ίδιο και περιλαμβάνει:

- Δύο function generators F και G
- Δύο στοιχεία αποθήκευσης
- Αριθμητικές λογικές πύλες
- Μεγάλους πολυπλέκτες
- Ευρεία ικανότητα λειτουργιών

- Γρήγορη carry look-ahead chain
  - Οριζόντια cascade chain (OR gate)

Η γενική διάρθρωση της φέτας VirtexII φαίνεται στην **Εικόνα 5**. Οι function generators F και G μπορούν να ρυθμιστούν ως 4-εισόδων look-up πίνακες (LUTs), ως 16-bit shift registers, ή ως 16-bit distributed SelectRAM memory. Τα δύο στοιχεία αποθήκευσης μπορούν να ρυθμιστούν είτε ως edge-triggered D-type flip-flops είτε ως level-sensitive latches. Κάθε CLB έχει εσωτερική γρήγορη διασύνδεση και συνδέεται σε έναν διακόπτη μήτρας (switch matrix) για την πρόσβαση στη γενική δρομολόγηση των πόρων.



**Εικόνα 5. General Slice in Virtex-II [15]**

### 2.5.5 Distributed SelectRAM

Όπως περιγράφεται παραπάνω, η Virtex-II οικογένεια συσκευών περιέχει Distributed SelectRAM modules που υλοποιούνται με τη χρήση των function generators των CLB πόρων. Η Κατανεμημένη SelectRAM μνήμη γράφει σύγχρονα και διαβάζει ασύγχρονα. Ωστόσο, μια μνήμη που διαβάζει σύγχρονα μπορεί να υλοποιηθεί χρησιμοποιώντας τους registers που είναι διαθέσιμοι στο ίδιο κομμάτι. Ένας LUT μπορεί να ρυθμιστεί ως μια 16x1-bit RAM που είναι σειριακή, εάν επιθυμείται μια βαθύτερη και ευρύτερη εφαρμογή μνήμης.

Ένα primitive είναι το συστατικό στοιχείο στην βιβλιοθήκη λογισμικού Xilinx, το οποίο είναι διαθέσιμο για instantiation χωρίς την ανάγκη για έναν ορισμό του από έναν HDL κώδικα. Ένα CLB που περιλαμβάνει 8 LUTs μπορεί να χρησιμοποιηθεί για να διαμορφώσει μέχρι και 128x1-bit RAM module primitives (βλέπε **Table 2**). Αυτά τα primitives είναι στην πραγματικότητα modules της Κατανεμημένης SelectRAM. Δύο 16x1 RAM πόροι μπορούν να συνδυάζονται για να αποτελέσουν μια διπλή θύρα 16x1 RAM με μια θύρα ανάγνωσης / εγγραφής και μια δεύτερη θύρα μόνο για ανάγνωση. Η πρώτη θύρα γράφει ταυτόχρονα και στις δύο 16x1 RAMs ενώ η δεύτερη θύρα διαβάζει ανεξάρτητα.

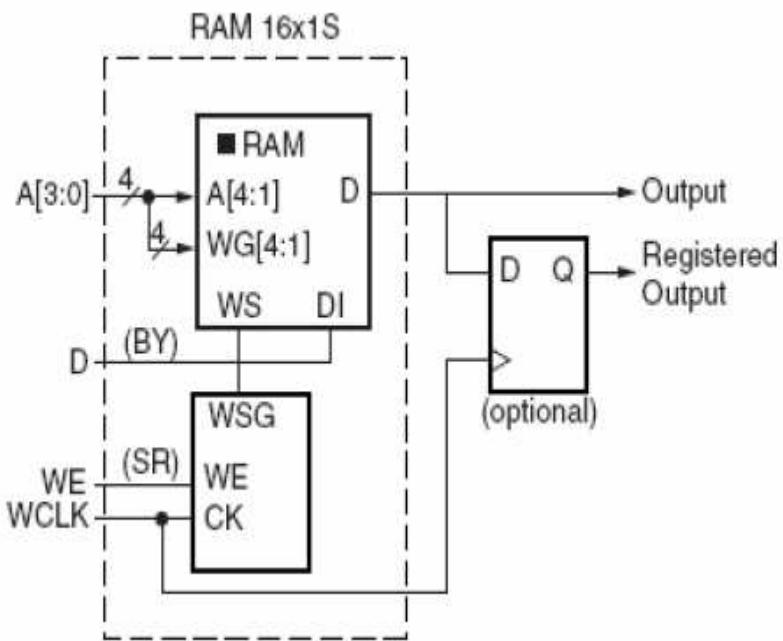
Ο **Table 2** δείχνει τον αριθμό των LUTs που χρησιμοποιούνται σε κάθε κατανεμημένο SelectRAM primitive.

Primitive name	RAM Size	Type	# of LUTs
16x1S	16x1 bits	single port	1
16x1D	16x1 bits	dual port	2
32x1S	32x1 bits	single port	2
32x1D	32x1 bits	dual port	4
64x1S	64x1 bits	single port	4
64x1D	64x1 bits	dual port	8
128x1S	128x1 bits	single port	8

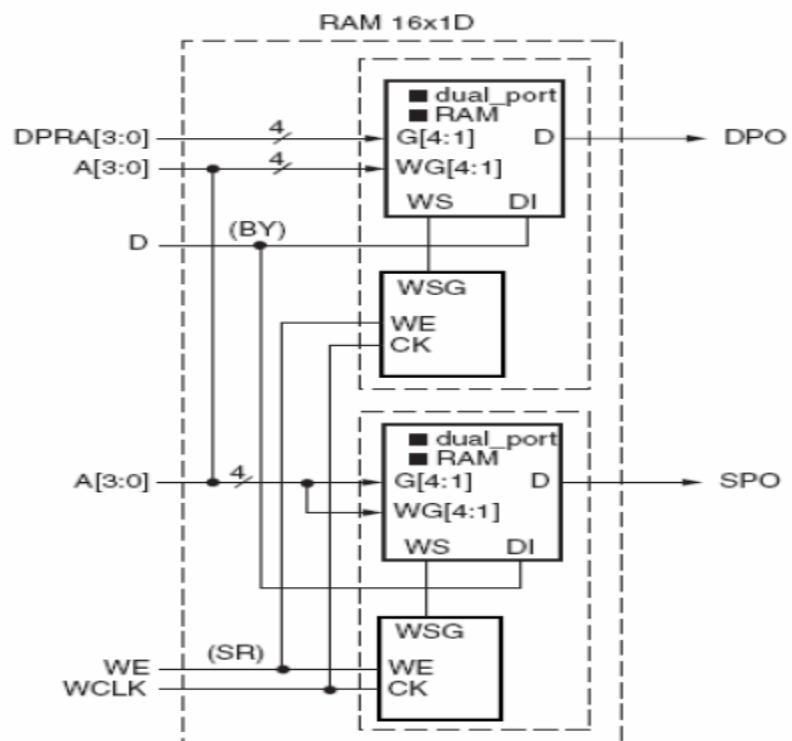
**Table 2. Resources Used by Distributed Memory [15]**

Η single-port Κατανεμημένη SelectRAM, γράφει σύγχρονα στις γραμμές διευθύνσεων WG [4:1], και διαβάζει ασύγχρονα στις γραμμές A [4:1], ενώ χρησιμοποιεί τις ίδιες κοινές γραμμές διευθύνσεων A [3:0] από το εξωτερικό.

Η dual-port Κατανεμημένη SelectRAM, εκτός από έναν LUT που χρησιμοποιείται στην single-port λειτουργία, ένα δεύτερο LUT χρησιμοποιεί τις ίδιες γραμμές διευθύνσεων A [3:0] για να γράψει σύγχρονα και ένα άλλο σύνολο γραμμών διευθύνσεων DPRA [3:0] χρησιμοποιούνται για το δεύτερο ασύγχρονο διάβασμα.



**Εικόνα 6. Single-Port Distributed SelectRAM [15]**



**Εικόνα 7. Dual-Port Distributed SelectRAM [15]**

## 2.5.6 Block SelectRAM

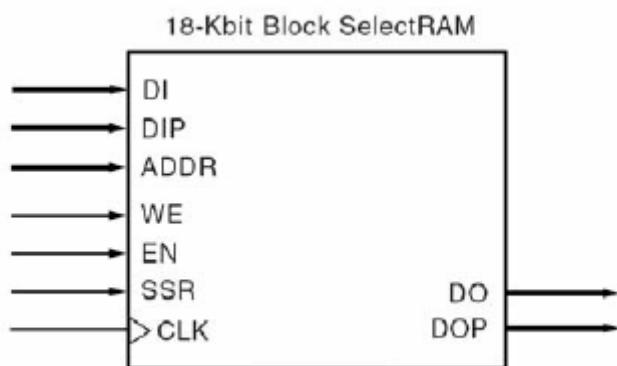
Εκτός από την κατανεμημένη μνήμη (Distributed SelectRAM), η Virtex-II οικογένεια συσκευών περιλαμβάνει ένα μεγάλο ποσό από πόρους των 18 Kb block SelectRAM (Bram), προγραμματιζόμενες από 16Kx1 bit σε 512x36 bit, σε διάφορα βάθη και πλάτη συνθέσεων. Αυτά τα SelectRAM blocks είναι dual-port RAM blocks τα οποία μπορούν εύκολα να ρυθμιστούν ως FIFO. Ο **Table 3** παρουσιάζει τις υποστηριζόμενες διαμορφώσεις μνήμης για single-port και dual-port τρόπους.

16Kx1 bit	2Kx9 bits
8Kx2 bits	1Kx18 bits
4Kx4 bits	512x36 bits

**Table 3. Block SelectRAM Configuration Modes [15]**

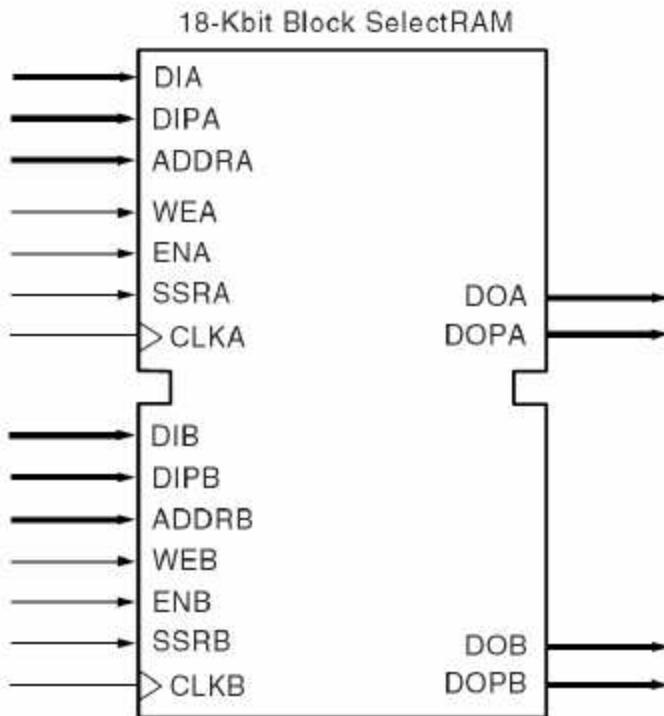
Κάθε port του SelectRAM + block έχει τις εξής εισόδους: Clock, Clock Enable, Write Enable, Set/Reset, Address, και ξεχωριστά Data και Parity buses για τις εισόδους και τις εξόδους. Και οι δύο write και read πράξεις είναι σύγχρονες.

Ως single-port RAM, το μπλοκ SelectRAM έχει πρόσβαση στις θέσεις μνήμης 18 Kb σε οποιαδήποτε από τις συνθέσεις του **Table 3**. Κάθε μπλοκ SelectRAM κύτταρο είναι μια απόλυτα σύγχρονη μνήμη όπως φαίνεται στην **Εικόνα 8**. Το data bus εισόδου και το data bus εξόδου είναι πανομοιότυπα.



**Εικόνα 8. Block SelectRAM in Single-Port Mode [15]**

Οι dual-port RAM, κάθε port του μπλοκ SelectRAM έχει πρόσβαση σε ένα κοινό πόρο 18 Kb μνήμης. Αυτά είναι πλήρως σύγχρονα ports με ανεξάρτητα τα σήματα ελέγχου για κάθε port, όπως φαίνεται στην **Εικόνα 9**. Τα data widths των δύο ports μπορούν να ρυθμιστούν ανεξάρτητα.



**Εικόνα 9. Block SelectRAM in Dual-Port Mode[15]**

Εφόσον τα SelectRAM μπλοκ έχουν μια τακτική array δομή, τα place-and-route εργαλεία μπορούν να επωφεληθούν από αυτήν τη δυνατότητα για να επιτευχθεί η βέλτιστη απόδοση του συστήματος και γρήγοροι χρόνοι κατά το compile. [15].

*Η Κατανεμημένη SelectedRAM και η Block SelectedRAM χρησιμοποιούνται για την αποθήκευση εξαρτημάτων στην AES-GCM υλοποίηση. Για παράδειγμα, ο single port τρόπος λειτουργίας της SelectedRAM ή Block SelectedRAM χρησιμοποιείται για την υλοποίηση των S-box Look Up πίνακα. Επίσης, ο dual port τρόπος λειτουργίας της SelectedRAM ή Block SelectedRAM χρησιμοποιείται για την υλοποίηση μιας 11x128 bit FIFO.*

## Κεφάλαιο 3

# Security Standard

Σε αυτό το κεφάλαιο, θα παρουσιαστούν τα θεμελιώδη στοιχεία του Advanced Encryption Standard (AES) αλγορίθμου.

### 3.1 Advanced Encryption Standard (AES)

Το Advanced Encryption Standard (AES) εκδόθηκε από το NIST το 2001. Ο AES αλγόριθμος ανήκει στην οικογένεια των συμμετρικών αλγόριθμων κρυπτογραφίας, όπου και ο αποστολέας και ο παραλήπτης συμφωνούν σε ένα κοινό μυστικό κλειδί για την κρυπτογράφηση και την αποκρυπτογράφηση. *Ο AES είναι δηλαδή ένα συμμετρικό block cipher που λειτουργεί πάνω σε 128-bit block δεδομένα εισόδου και εξόδου.* Ο αλγόριθμος μπορεί να κρυπτογραφήσει και να αποκρυπτογραφήσει blocks κάνοντας χρήση του κρυφού κλειδιού το οποίο μπορεί να έχει μέγεθος 256-bit, 192-bit, ή 128-bit.

Επιπρόσθετα, ο AES είναι ένας επαναληπτικός αλγόριθμος και ένα από τα κυριότερα χαρακτηριστικά του είναι η απλότητα, που επιτυγχάνεται συνδυάζοντας με επαναλαμβανόμενο τρόπο αντικαταστάσεις (substitutions) και αναδιατάξεις (permutations) σε διαφορετικούς γύρους. Αυτό σημαίνει ότι ο AES κρυπτογραφεί/αποκρυπτογραφεί ένα 128-bit plaintext/ciphertext εφαρμόζοντας επαναλαμβανόμενα τον ίδιο γύρο μετασχηματισμού (round transformation) αρκετές φορές σύμφωνα με το μέγεθος του κλειδιού.

To **Table 1** παρακάτω δείχνει αυτήν την εξάρτηση.

	Key Length (32-bit word)	Block Size (32-bit word)	Number of Rounds
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

**Table 1. Key-Block-Round Combinations [3]**

To block δεδομένων 128-bit χωρίζεται σε 16 bytes. Αυτά τα bytes αντιστοιχίζονται σε έναν  $4 \times 4$  πίνακα ο οποίος καλείται *State array*, και όλες οι εσωτερικές πράξεις του AES αλγορίθμου εκτελούνται στον State. Κάθε byte του State συμβολίζεται ως  $(0,4)$  όπου  $S_{i,j}$  ( $0 \leq i,j \leq 3$ ), και αποτελεί στοιχείο του πεπερασμένου πεδίου  $GF(2^8)$ . Παρόλο που διαφορετικά ανάγωγα (irreducible) πολυώνυμα χρησιμοποιούνται για να δημιουργηθεί το  $GF(2^8)$  πεπερασμένο πεδίο, το ανάγωγο πολυώνυμο που χρησιμοποιείται στον AES αλγόριθμο είναι το  $p(x) = x^8 + x^4 + x^3 + x + 1$ .

Το πραγματικό μέγεθος του κλειδιού εξαρτάται από το επιθυμητό επίπεδο ασφαλείας (security level). Ο αλγόριθμος AES-128 είναι ο επικρατέστερος και υποστηρίζεται από τις περισσότερες hardware υλοποιήσεις. Στην ενότητα αυτή, συζητείται κυρίως η AES forward cipher λειτουργία (δηλαδή η AES κρυπτογράφηση) με κλειδί μεγέθους 128-bit, καθώς η διαδικασία αυτή καλείται στο GCTR module του AES-GCM standard για να διασφαλίσει την εμπιστευτικότητα (confidentiality) των δεδομένων.

### 3.1.1 AES Cipher

Για μέγεθος κλειδιού 128-bit, υπάρχουν 10 rounds από substitutions και permutations που πρέπει να εκτελεστούν στον AES cipher (βλέπε **Table 1**). Το 128-bit plaintext εισόδου παρουσιάζεται στον πίνακα  $4 \times 4$  από bytes. Έτσι, υπάρχουν 32 bits σε κάθε γραμμή και κάθε στήλη του πίνακα. Αυτός ο πίνακας, που όπως αναφέραμε προηγούμενα λέγεται State array, φαίνεται στην **Εικόνα 1**.

State array			
$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

**Εικόνα 1. Illustration of State Array [3]**

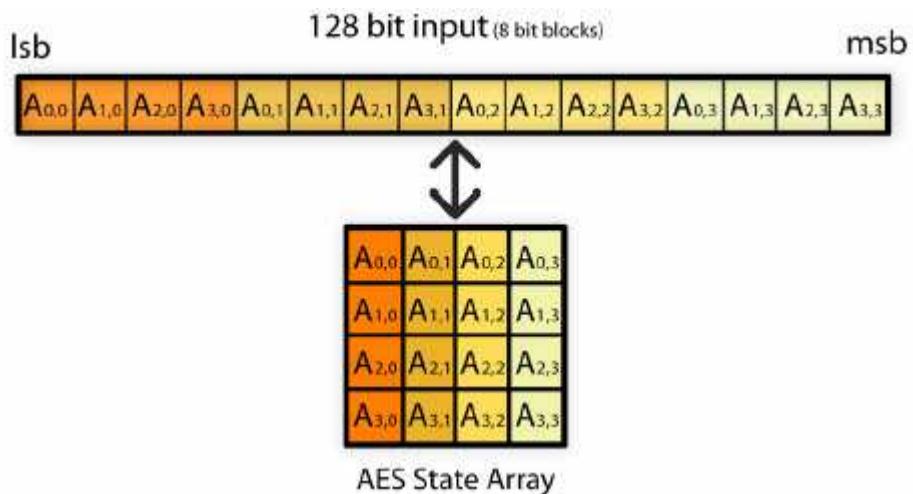
Στην **Εικόνα 1**, το  $S_{i,j}$  φανερώνει ένα byte, όπου  $0 \leq i, j \leq 3$ . Ο State array υπόκειται σε τροποποιήσεις σε κάθε γύρο. Το κλειδί εισόδου επεκτείνεται σε έναν πίνακα των 44 32-bit words, και κάθε φορά 4 words (128 bits) του επεκταμένου κλειδιού θα χρησιμοποιηθούν σε κάθε γύρο. Η επέκταση του κλειδιού πρέπει να γίνει πριν την cipher λειτουργία.

Κατά τη διαδικασία της κρυπτογράφησης στον AES αλγόριθμο, κάθε κύκλος, εκτός από τον τελικό, αποτελείται από τέσσερεις μετασχηματισμούς: SubBytes, ShiftRows, MixColumns, και AddRoundKey, ενώ στον τελικό κύκλο δεν υπάρχει ο MixColumns μετασχηματισμός. Παρακάτω παρουσιάζονται οι βασικές κρυπτογραφικές πράξεις και οι ιδιότητές τους, εν συντομίᾳ.

- **SubBytes** – αποτελεί έναν μη γραμμικό μετασχηματισμό όπου κάθε byte αντικαθίσταται σύμφωνα με έναν πίνακα αντικατάστασης.
- **ShiftRows** – αποτελεί βήμα αλληλομετάθεσης όπου κάθε γραμμή από το State ολισθαίνει κυκλικά για ένα συγκεκριμένο αριθμό θέσεων.
- **MixColumns** – η πράξη αυτή εφαρμόζεται σε κάθε στήλη του State και αναπαριστά ένα γραμμικό μετασχηματισμό βασισμένο στα 4 bytes της στήλης.
- **AddRoundKey** – κάθε byte του state συνδυάζεται με ένα byte από το κλειδί του τρέχοντος κύκλου. Κάθε κλειδί του κύκλου προέρχεται από το κλειδί κρυπτογράφησης χρησιμοποιώντας το σύστημα παραγωγής κλειδιών του AES (key schedule).

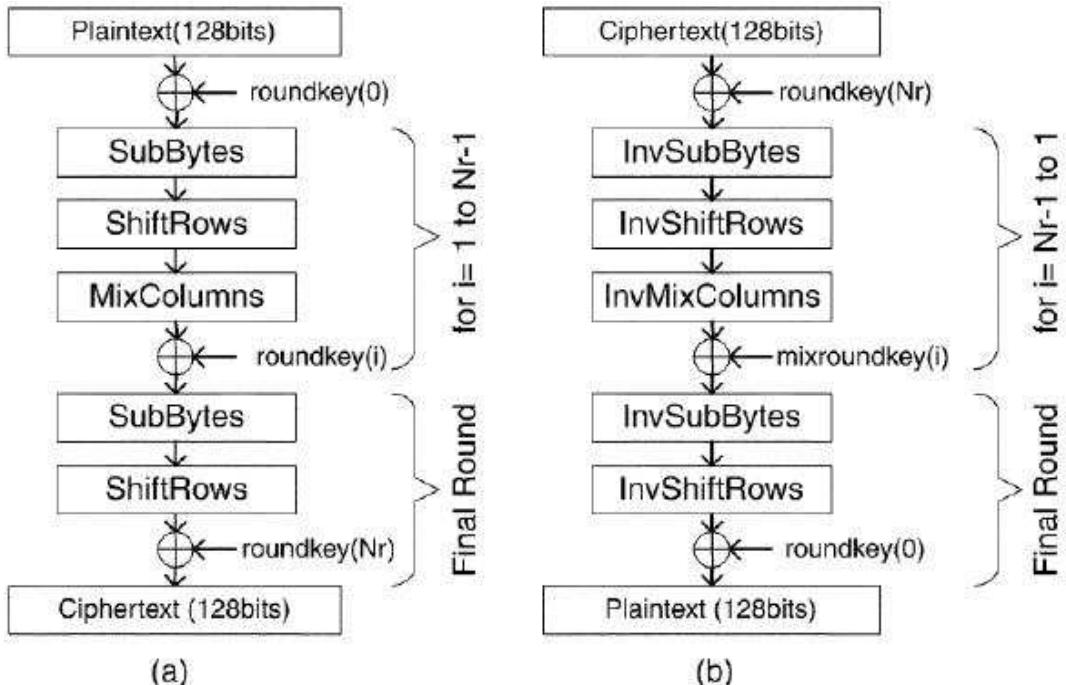
Οι προηγούμενοι μετασχηματισμοί εφαρμόζονται και στην αποκρυπτογράφηση αλλά με παρόμοια λογική. Οι μετασχηματισμοί που προκύπτουν και χρησιμοποιούνται στην αποκρυπτογράφηση είναι οι InvShiftRows, InvSubBytes, InvMixColumns, και AddRoundKey.

Στις ακόλουθες ενότητες το State array μπλοκ θα χρησιμοποιείται για να περιγράψει τα διαφορά round operations και γι 'αυτό είναι σημαντικό να κατανοήσουμε τον τρόπο με τον οποίο η είσοδος μετατρέπεται στον State array. Η **Εικόνα 2** παρακάτω δείχνει αυτή την μετατροπή, γεμίζοντας bytes δεδομένων στον State ανά στήλη. Μετά το τέλος της AES κρυπτογράφησης, ο τελευταίος State που δίνεται στην έξοδο μετατρέπεται πάλι πίσω σε ένα 128 bit stream.



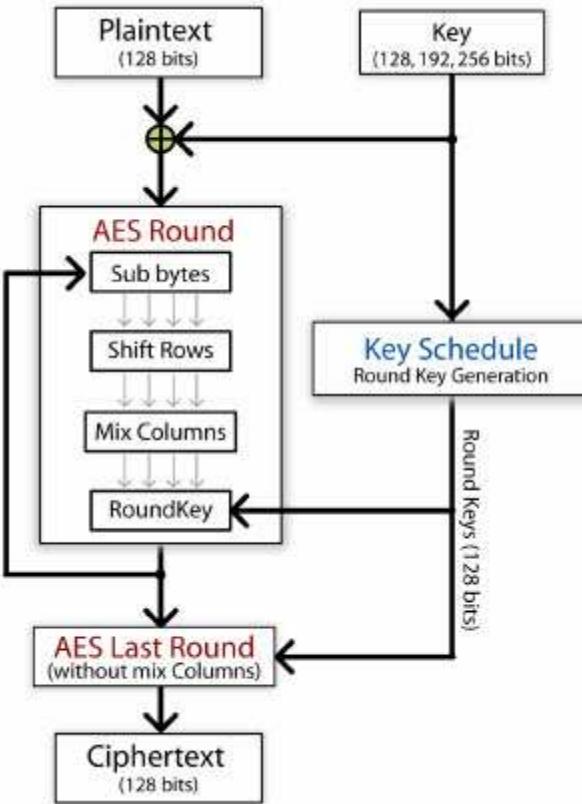
**Εικόνα 2. Illustration of State Array's Inputs**

Η Εικόνα 3 παρακάτω, παρουσιάζει το διάγραμμα δομικών στοιχείων του AES αλγορίθμου για την κρυπτογράφηση και την αποκρυπτογράφηση, την σειρά που είναι συνδεδεμένα και τον αριθμό των κύκλων που απαιτούνται ανάλογα με το μέγεθος του κλειδιού, ενώ στη συνέχεια παρουσιάζονται πιο λεπτομερώς οι τέσσερις μετασχηματισμοί που συμβαίνουν κατά την διαδικασία της κρυπτογράφησης στον AES.



**Εικόνα 3. Illustration of AES functions**

Το διάγραμμα που ακολουθεί στην **Εικόνα 4** απεικονίζει τη γενική δομή του γύρου του AES που επαναλαμβάνεται με βάση το κλειδί εισόδου.



**Εικόνα 4. Illustration of AES round structure**

Διάφορα datapaths υλικού μπορούν να δημιουργηθούν από αυτή τη modular (αποτελούμενη από υπομονάδες) δομή γύρου.

Ένας επαναληπτικός σχεδιασμός μπορεί να χρησιμοποιήσει το ίδιο σχέδιο που αναφέραμε, αλλά απλώς να προσθέτει έναν 128 bit data register στο τέλος της δομής. Μετά από το πολύ 14 κύκλους της κρυπτογράφησης AES το αποτέλεσμα μπορεί να επιτευχθεί. Αυτός ο επαναληπτικός σχεδιασμός μπορεί να δημιουργήσει μια εφαρμογή που χρησιμοποιεί τεχνικές διοχέτευσης τοποθετώντας registers μεταξύ των round μπλοκ.

Αυτή είναι μια εξωτερική τεχνική διοχέτευσης στον σχεδιασμό του AES και μια 128 bit έξοδο μπορεί να παραχθεί σε κάθε κύκλο ρολογιού. Ωστόσο, υπάρχει αρκετή ευελιξία στην επιλογή των σημείων της διοχέτευσης. Επιπλέον στάδια διοχέτευσης μπορούν να προστεθούν σε καθένα από τα components του AES γύρου. Αυτό χαρακτηρίζεται ως μια εσωτερική τεχνική διοχέτευσης στον σχεδιασμό του AES, και παρόλο που παρουσιάζεται μια μεγαλύτερη καθυστέρηση και επιφάνεια, είναι δυνατόν να επιτευχθεί υψηλότερη απόδοση.

## 3.2 SubBytes Μετασχηματισμός

Ο SubBytes μετασχηματισμός είναι ο μόνος μη γραμμικός μετασχηματισμός που εφαρμόζεται ανεξάρτητα σε κάθε byte του State χρησιμοποιώντας έναν πίνακα αντικατάστασης. Αντικαθιστά όλα τα bytes του State array χρησιμοποιώντας ένα substitution table, το οποίο είναι ένας αντιστρέψιμος πίνακας 16x16 από bytes, και συχνά καλείται S-box. Το S-box, αποτελεί ένα look-up-table (LUT), και χρησιμοποιείται στον SubBytes μετασχηματισμό καθώς περιέχει τα αποτελέσματα των substitution and permutation όλων των πιθανών 8-bit τιμών. Το περιεχόμενο του S-box υπολογίζεται μέσω των δύο παρακάτω βημάτων [3]:

1. Εφαρμογή της αντιστροφής στο πεπερασμένο πεδίο  $GF(2^8)$ , το στοιχείο {00} αντιστοιχεί στον εαυτό του.
2. Εφαρμογή του ακόλουθου μετασχηματισμού στο  $GF(2)$ :

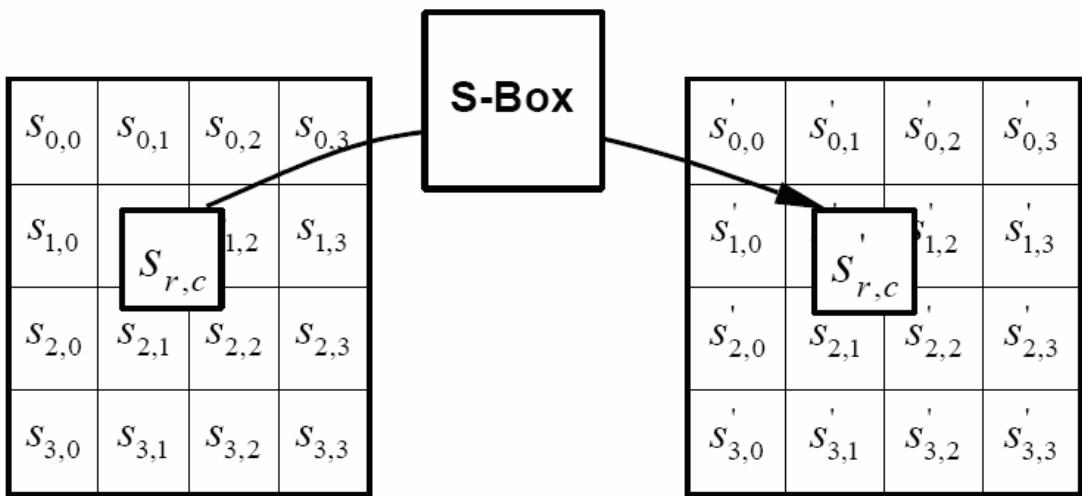
$$b'_i = b_i \{ \text{xor} \} b_{(i+4)} \{ \text{xor} \} b_{(i+5)\text{mod}8} \{ \text{xor} \} b_{(i+6)\text{mod}8} \{ \text{xor} \} b_{(i+7)\text{mod}8} \{ \text{xor} \} c_i$$

ο μετασχηματισμός αυτός καλείται affine μετασχηματισμός, όπου  $b_i$  είναι το i bit του byte b, και  $c_i$  το i bit του byte c με τιμή {63} ή {01100011}. Σε μορφή μητρώου ο δεύτερος μετασχηματισμός του S-box μπορεί να παρουσιαστεί ως εξής:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

### Εικόνα 5. Ο affine μετασχηματισμός σε μορφή μητρώου

Δηλαδή, βρίσκεται το πολλαπλασιαστικό αντίστροφο σε ένα finite-field και ακολούθως εφαρμόζεται έναν affine μετασχηματισμό στο  $GF(2)$  (μετασχηματισμός που αποτελείται από πολλαπλασιασμό με έναν πίνακα, ακολουθούμενο από την προσθήκη ενός vector). Καθένα από τα byte του state αντιστοιχίζεται σε ένα byte του S-box. Τα 4 leftmost bits χρησιμοποιούνται σαν δείκτες γραμμών (row index) ενώ τα 4 rightmost bits χρησιμοποιούνται σαν δείκτες στηλών (column index). Η Εικόνα 6 εικονίζει την επίδραση του SubBytes μετασχηματισμού στον State array. Το S-box είναι σχεδιασμένο ώστε να αντέχει στις γνωστές κρυπταναλυτικές επιθέσεις [5]. Ο SubBytes μετασχηματισμός έχει την ιδιότητα η έξοδος του να μην μπορεί να περιγραφεί σαν μια απλή μαθηματική συνάρτηση της εισόδου του.



**Εικόνα 6. Illustration of SubBytes Operation [3]**

Το S-box χρησιμοποιεί το SubBytes μετασχηματισμό και εκφράζεται μέσω του πίνακα οποίος απαρτίζεται από στοιχεία σε δεκαεξαδική μορφή όπως παρουσιάζεται στην **Εικόνα 7**.

Για παράδειγμα, αν  $S_{i,j} = \{53\}$ , τότε η τιμή αντικατάστασης θα καθοριστεί από την τομή της γραμμής '5' και της στήλης '3' της **Εικόνα 7**. Ως αποτέλεσμα το '53' θα αντικατασταθεί από την τιμή {ed}.

		y																
		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

**Εικόνα 7. Ο πίνακας μετασχηματισμών για την κρυπτογράφηση  
(σε δεκαεξαδική μορφή).**

Το ακόλουθο σχήμα αποτελεί ένα παράδειγμα του SubBytes μετασχηματισμού σύμφωνα με την περιγραφή που δόθηκε προηγούμενα:

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

### 3.2.1 Υλοποιήσεις του συστήματος SubBytes

Ο πιο απλός τρόπος για να ολοκληρωθεί ο S-box πίνακας σε υλικό σύμφωνα με την **Εικόνα 7** είναι να ακολουθηθούν πιστά οι μετασχηματισμοί όπως ορίζονται από τον πίνακα.

Η subBytes λειτουργία χρησιμοποιεί πολλαπλά substitution box components (Sbox) κάθε ένα από τα οποία εκτελεί μια 8 bit υποκατάσταση (substitution). Κάθε 8-bit λέξη δεδομένων στον state array, αντικαθίσταται με τη χρήση του Sbox. Αυτό έχει ως αποτέλεσμα την παραγωγή 16 Sbox components που χρησιμοποιούνται για κάθε round block, και από άποψη υλικού αυτό αποτελεί το κομμάτι που καταναλώνει το μεγαλύτερο μέρος επιφάνειας στον AES γύρο (75%). Ο υπολογισμός του Sbox είναι ουσιαστικά ένα πολλαπλασιαστικό αντίστροφο στο GF(2<sup>8</sup>), ακολουθούμενο από έναν affine μετασχηματισμό ο οποίος είναι μια γραμμική χαρτογράφηση από ένα διανυσματικό χώρο σε έναν άλλο. Ένας πίνακας αναζήτησης, με 2<sup>8</sup> τιμές μπορεί να χρησιμοποιηθεί για την υλοποίηση του Sbox, αλλά μπορεί επίσης να υπολογιστεί μαθηματικά με τη χρήση λογικών πυλών (μέσω πίνακα Karnaugh και απλοποιήσεων).

Σε αυτή την πρώτη προσέγγιση, δεν γίνεται χρήση του μαθηματικού υπόβαθρου που διέπει τον S-box πίνακα. Η διαδικασία βελτιστοποίησης βασίζεται αποκλειστικά στα χαρακτηριστικά του εργαλείου σύνθεσης και στις βιβλιοθήκες που σχετίζονται σε αυτό. Επομένως, η κατανάλωση ισχύος εξαρτάται από τις βιβλιοθήκες του εργαλείου σύνθεσης και τις οικογένειες των πυλών. Αν το εργαλείο σύνθεσης τρέχει μια διαδικασία βελτιστοποίησης που συνδυάζει χαμηλούς πόρους σε επιφάνεια και χαμηλή κατανάλωση ισχύος, το προκύπτον κύκλωμα είναι μικρής επιφάνειας και με χαμηλή κατανάλωση ισχύος. Ένα σημαντικό μειονέκτημα αυτής της προσέγγισης είναι η μεγάλη επιφάνεια εφόσον δύο κυκλώματα υπάρχουν, ένα για την κρυπτογράφηση και ένα για την αποκρυπτογράφηση. Όμως στην προκειμένη περίπτωση, αυτό το μειονέκτημα εξαλείφεται καθώς η χρησιμότητα του AES αλγόριθμου έγκειται στην AES forward cipher λειτουργία (δηλαδή την AES κρυπτογράφηση), καθώς μονάχα αυτή η διαδικασία καλείται στο GCTR module του AES-GCM standard.

Μια παρόμοια προσέγγιση είναι αυτή της αντικατάστασης του Sbox με ROM. Σε αυτή την περίπτωση, η καθυστέρηση του κυκλώματος είναι ίση με την πρώτη προσέγγιση αλλά η κατανάλωση μπορεί να είναι τάξεις μεγέθους μεγαλύτερη.

Υπάρχουν πιο προχωρημένες προσεγγίσεις στο σχεδιασμό SubBytes μετασχηματισμών. Οι προσεγγίσεις αυτές χρησιμοποιούν το μαθηματικό υπόβαθρο του πεπερασμένου πεδίου  $GF(2^8)$  (αριθμητικές προσεγγίσεις Sbox). Το βασικό θέμα σε αυτές τις προσεγγίσεις είναι η αποδοτική πραγματοποίηση της αντιστροφής  $GF(2^8)$  η οποία μπορεί να στηριχτεί στην ανάλυση του  $GF(2^8)$  σε πεπερασμένα πεδία  $GF(2^4)$  και  $GF(2^2)$ .

Ο Rijmen, ένας από τους δημιουργούς του AES, έδειξε μια μέθοδο υπολογισμού του Sbox ‘σπάζοντας’ τις πράξεις του  $GF(2^8)$  σε ένα σύνθετο πεδίο  $GF((2^4)^2)$ , με αποτέλεσματα σημαντικά στην εξοικονόμηση επιφάνειας υλικού που διαφορετικά δεν θα ήταν δυνατή με τη χρήση εφαρμογών look-up πίνακα.

Ο τύπος για την αντίστροφη Sbox δίνεται στην μειωμένη του απόδοση, στην παρακάτω εξίσωση, όπου το  $\lambda$  είναι  $(1100)_2$ .

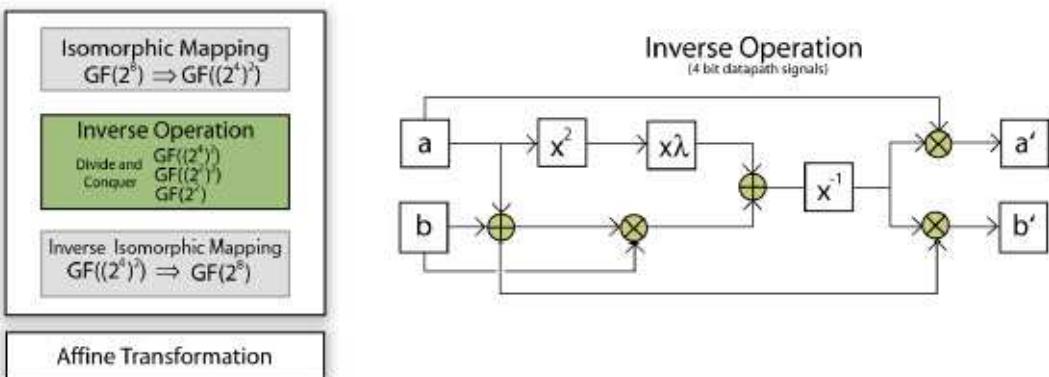
$$a'x + b' = (ax + b)^{-1} = a(a^2\lambda + b(a+b))^{-1}x + (b+a)(a^2\lambda + b(a+b))^{-1}$$

Η πρόσθεση, ο πολλαπλασιασμός, και οι αντίστροφες διεργασίες υπολογίζονται στο  $GF((2^4)^2)$ , και μπορούν να κατανεμηθούν περαιτέρω σε μικρότερα σύνθετα πεδία,  $GF((2^2)^2)$  και  $GF(2^2)$ , χρησιμοποιώντας τη μέθοδο του ‘διαιρεί και βασίλευε’.

Συνεπώς, η αντίστροφή ενός πεπερασμένου πεδίου μπορεί να πραγματοποιηθεί με διάφορους τρόπους σύμφωνα με αυτήν την προσέγγιση. Το βασικό στοιχείο αυτών των προσεγγίσεων είναι η μικρή επιφάνεια εξαιτίας του γεγονότος ότι η κωδικοποίηση και η αποκωδικοποίηση βασίζονται στο ίδιο κύκλωμα. Η μείωση όμως στην επιφάνεια δεν οδηγεί σε αντίστοιχη μείωση στην κατανάλωση ισχύος. Επιπλέον στις προσεγγίσεις αυτές η κατανάλωση ισχύος δεν λαμβάνεται υπόψη. Εξαιτίας των ανεπιθύμητων φορτίσεων-εκφορτίσεων εσωτερικά του κυκλώματος και γενικότερα στην τοπολογία του κυκλώματος (topology network), η κατανάλωση ισχύος μπορεί να είναι 10-30 φορές υψηλότερη από την προσέγγιση των πινάκων που αναφέραμε παραπάνω.

Στην **Εικόνα 8** φαίνεται σχηματικά αυτή η τελευταία προσέγγιση του σύνθετου Sbox. Η ισόμορφη χαρτογράφηση στο σύνθετο πεδίο,  $(GF(2^8) \rightarrow GF((2^4)^2))$  μπορεί να υλοποιηθεί χρησιμοποιώντας έναν matrix vector πολλαπλασιασμό. Ο affine μετασχηματισμός αποτελείται από έναν γραμμικό μετασχηματισμό που ακολουθείται από μια μετάφραση που μπορεί να επιτευχθεί με έναν πολλαπλασιασμό μήτρας διανυσμάτων και μια διανυσματική πρόσθεση, αντίστοιχα. Η ισόμορφη χαρτογράφηση και ο affine μετασχηματισμός χρησιμοποιούνται και τα δύο καθορισμένες (fixed) μήτρες που είναι αραιές ώστε το υπολογιστικό κόστος των πράξεων αυτών να είναι ελάχιστο.

## Composite SBox Stages



Εικόνα 8. Αριθμητική προσέγγιση του Sbox (AES Composite Sbox design)

Η επιφάνεια που καταναλώνεται από τα σύνθετα Sbox κυκλώματα είναι πολύ χαμηλή σε σύγκριση με την προσέγγιση του πίνακα αναζήτησης (LUT). Στο παραπάνω σχεδιασμό, ο οποίος προτάθηκε από τον Satoh, η Sbox συνιστώσα καταναλώνει 250 πύλες ενώ η LUT υλοποίηση είναι λίγο περισσότερο από 4 φορές μεγαλύτερη σε επιφάνεια. Το υπολογιστικό κόστος όμως αυξάνει την καθυστέρηση του κυκλώματος, και έτσι για νψηλής ταχύτητας σχέδια, η LUT υλοποίηση και τα Δυαδικά Διαγράμματα Απόφασης (Binary Decision Diagram - BDD) προτιμώνται.

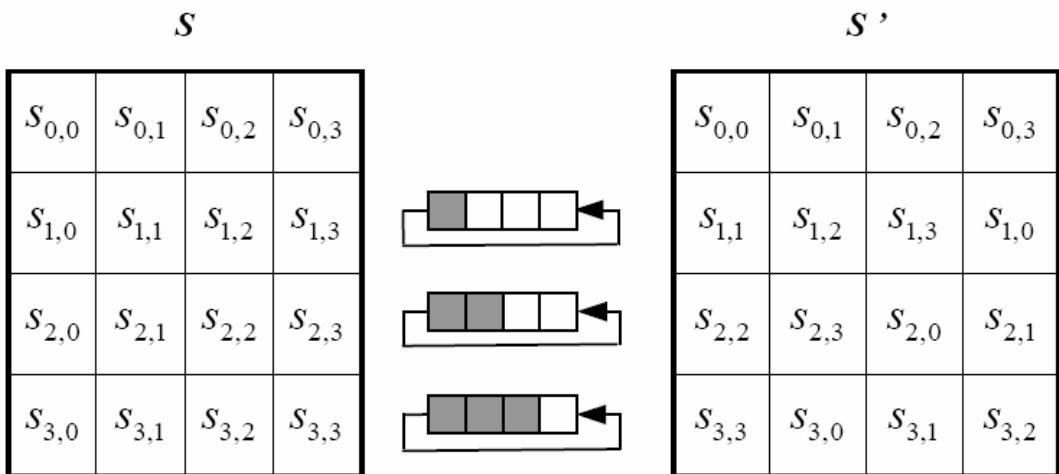
Η BDD υλοποίηση του Sbox παρέχει μια ελαφρώς ταχύτερη εναλλακτική λύση από αυτήν του LUT, καταναλώνοντας λιγότερους πόρους επιφάνειας. Κάθε bit της 8 bit εξόδου, συνδέεται με ένα δυαδικό δέντρο και βασίζεται στα bits εισόδου, κάθε δέντρο βοηθά να αποφασιστεί ποιά πρέπει να είναι τα bits εξόδου. Η 8 bit είσοδος χρησιμοποιείται ως τιμή επιλογής για αρκετά στρώματα πολυπλεκτών, ώστε να υλοποιηθούν τα δυαδικά δένδρα στο hardware. Αυτού του τύπου η κατασκευή αντιμετωπίζει μεγάλο πρόβλημα στο fan out για τα αρχικά στρώματα των πολυπλεκτών.

Μια βελτιωμένη εναλλακτική λύση που βελτιώνει τα θέματα αυτά είναι η Twisted-BDD, και είναι η ταχύτερη υλοποίηση του Sbox που έχει αναφερθεί στη βιβλιογραφία μέχρι σήμερα. Ωστόσο, οι απαιτήσεις επιφάνειας αυτού του σχεδιασμού είναι σχεδόν διπλάσιες από αυτές του LUT Sbox σχεδιασμού.

### 3.3 ShiftRows Μετασχηματισμός

Στον ShiftRows μετασχηματισμό, τα bytes στις τελευταίες τρείς γραμμές του State ολισθαίνουν κυκλικά για διαφορετικές τιμές bytes (βλέπε **Εικόνα 9**). Η πρώτη γραμμή δεν ολισθαίνει. Η δεύτερη γραμμή ολισθαίνει κυκλικά και αριστερόστροφα (left-shifted) κατά ένα byte. Στην τρίτη γραμμή πραγματοποιείται ένα κυκλικό left shift κατά δύο bytes αντή τη φορά. Τέλος, η τέταρτη γραμμή γίνεται κυκλικά left-shifted κατά τρία byte.

Εφόσον οι μετασχηματισμοί MixColumns και AddRoundKey γίνονται στήλη παρά στήλη, ο ShiftRows διασφαλίζει ότι 4 bytes από μία στήλη διανέμονται σε τέσσερις διαφορετικές στήλες. Η **Εικόνα 9** εικονίζει την επίδραση του ShiftRows μετασχηματισμού στον State array.



**Εικόνα 9. Illustration of ShiftRows Transformation [3]**

Το ακόλουθο σχήμα αποτελεί ένα παράδειγμα του ShiftRows μετασχηματισμού σύμφωνα με την περιγραφή που δόθηκε προηγούμενα:

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Ο ShiftRows μετασχηματισμός είναι πιο ουσιαστικός από ότι μπορεί να φαίνεται αρχικά. Αυτό οφείλεται στο γεγονός ότι ο State, καθώς και οι είσοδοι/έξοδοι του cipher, θεωρούνται ως ένας πίνακας από τέσσερις στήλες των 4 byte. Έτσι, για την κρυπτογράφηση, τα πρώτα 4 bytes του plaintext αντιγράφονται στην πρώτη στήλη

του State, και ούτω καθεξής. Περαιτέρω, όπως θα δούμε, το round key εφαρμόζεται στον State στήλη παρά στήλη. Έτσι, μια shift row μετακινεί ένα ξεχωριστό byte από μια στήλη σε μια άλλη στήλη, η οποία είναι μια γραμμική απόσταση ενός πολλαπλάσιου των 4 byte. Με τον τρόπο αυτό η μετατροπή εξασφαλίζει ότι τα 4 bytes μιας στήλης, κατανέμονται σε τέσσερις διαφορετικές στήλες.

*Σε υλικό δεν απαιτείται κάποια λογική για αυτό το βήμα και απλές συνδέσεις σύρματος χρησιμοποιούνται για να δρομολογήσουν την είσοδο στην έξοδο.*

### 3.4 MixColumns Μετασχηματισμός

Ο μετασχηματισμός MixColumns λειτουργεί χωριστά σε κάθε στήλη του State του αλγορίθμου AES και αντιμετωπίζει κάθε στήλη σαν ένα πολυώνυμο τεσσάρων όρων. Μπορούμε να θεωρήσουμε τις στήλες σαν πολυώνυμα  $GF(2^8)$  τα οποία πολλαπλασιάζονται modulo το ανάγωγο πολυώνυμο για τον AES, με ανάγωγο πολυώνυμο το :

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

Καθένα από τα byte στήλης του state αντιστοιχίζεται σε μια νέα τιμή που είναι συνάρτηση και των τεσσάρων bytes αυτής της στήλης όπως δείχνεται παρακάτω. Για την περίπτωση της κρυπτογράφησης έχουμε την παρακάτω σχέση για κάθε στήλη του State :

$S'(x) = a(x) * S(x)$ , που σε μορφή μητρώου είναι :

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

Τη διαφορετικά σε πιο συνεπτυγμένη μορφή :

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}, c \in (0 \leq c < 4)$$

Ως αποτέλεσμα της παραπάνω σχέσης παίρνουμε τις παρακάτω εξισώσεις:

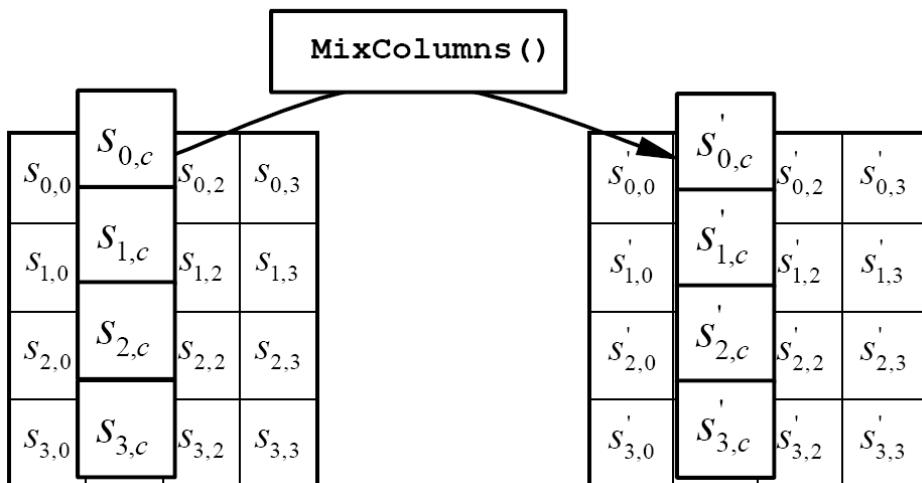
$$S'0,c = (\{02\} * S0,c) + (\{03\} * S1,c) + S2,c + S3,c$$

$$S'1,c = (\{02\} * S1,c) + (\{03\} * S2,c) + S0,c + S3,c$$

$$S'2,c = (\{02\} * S2,c) + (\{03\} * S3,c) + S0,c + S1,c$$

$$S'3,c = (\{02\} * S3,c) + (\{03\} * S0,c) + S1,c + S2,c$$

Στην **Εικόνα 10**, παρουσιάζεται η γραφική απεικόνιση της MixColumns πράξης :



**Εικόνα 10. Illustration of MixColumns Transformation [3]**

Το ακόλουθο σχήμα αποτελεί ένα παράδειγμα του MixColumns μετασχηματισμού σύμφωνα με την περιγραφή που δόθηκε προηγούμενα:

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

Οι συντελεστές της μήτρας του μετασχηματισμού MixColumns, βασίζονται σε ένα γραμμικό κώδικα με μέγιστη απόσταση μεταξύ των κωδικών λέξεων, που εξασφαλίζει ένα καλό ‘ανακάτεμα’ μεταξύ των bytes κάθε στήλης. Οι ShiftRows και MixColumns μετασχηματισμοί μαζί εξασφαλίζουν ότι μετά την εκτέλεση όλων των γύρων, όλα τα bits εξόδου εξαρτώνται από όλα τα bits εισόδου.

Επιπλέον, η επιλογή των συντελεστών στον MixColumns, που είναι όλα (01), (02), ή (03), είχε επηρεαστεί από τις απαιτήσεις της υλοποίησης. Ο πολλαπλασιασμός με αυτούς τους συντελεστές περιλαμβάνει το πολύ μια μετατόπιση και μια XOR. Οι συντελεστές του InvMixColumns είναι πιο δύσκολο να υλοποιηθούν. Ωστόσο, η κρυπτογράφηση θεωρείται πιο σημαντική από την αποκρυπτογράφηση για δύο λόγους:

1. Για τα CFB και OFB cipher modes, χρησιμοποιείται μόνο η κρυπτογράφηση.
2. Όπως και κάθε block cipher, έτσι και ο AES, μπορεί να χρησιμοποιηθεί για την κατασκευή ενός μηνύματος γνησιότητας (authentication message) και για τον σκοπό αυτό χρησιμοποιείται μόνο η κρυπτογράφηση.

### 3.4.1 Προτεινόμενη αρχιτεκτονική του συστήματος MixColumns

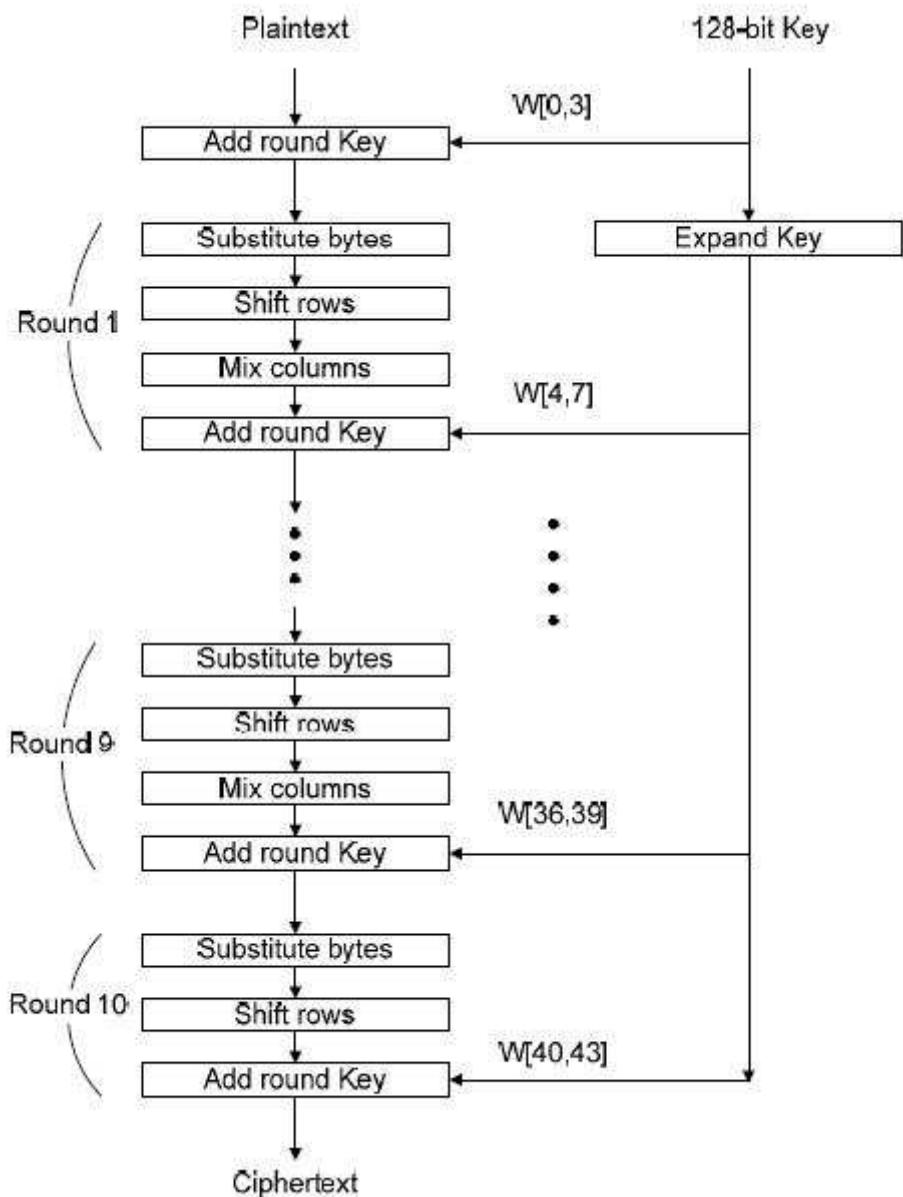
Το προτεινόμενο σύστημα έχει είσοδο-έξοδο 32-bit. Κάθε είσοδος του συστήματος είναι η στήλη του AES State. Το ενός bit σήμα encrypt/decrypt καθορίζει αν το σύστημα βρίσκεται σε λειτουργία κρυπτογράφησης ή αποκρυπτογράφησης. Το σύστημα εμπεριέχει τέσσερα δομικά στοιχεία Multiplier. Τα τέσσερα Multiplier δομικά στοιχεία εκτελούν πολλαπλασιασμούς σε  $GF(2^8)$  πεπερασμένα πεδία, μεταξύ της 32-bit εισόδου και των σταθερών {01}, {01}, {02} και {03} για την κρυπτογράφηση και {09}, {0B}, {0D} και {0E} για την αποκρυπτογράφηση. Η έξοδος του συστήματος είναι 32-bit. Με τη κατάλληλη συσχέτιση των εξόδων τα 8-bit γινόμενα των πολλαπλασιαστών ακολουθούν το κατάλληλο XOR δέντρο όπως ορίστηκε από τους πολλαπλασιασμούς μητρώων παραπάνω. Στο τέλος της διεργασίας, η 32-bit έξοδος είναι η i στήλη του AES State.

## 3.5 AddRoundKey Μετασχηματισμός

Ο AddRoundKey μετασχηματισμός είναι σχεδιασμένος σαν ένας stream cipher. Όλα τα 128 bits του State γίνονται XORED με τέσσερις 32-bit words του επεκταμένου κλειδιού. Η AddRoundKey είναι η μόνη λειτουργία που εμπλέκει τη χρήση του κλειδιού για να διασφαλίσει την προστασία των δεδομένων. Η πράξη θεωρείται σαν μία columnwise λειτουργία μεταξύ των 4 byte μιας στήλης του State και μιας λέξης του round key. Μπορεί επίσης να θεωρηθεί ως μία byte-level λειτουργία.

Η AES 128-bit key size forward cipher λειτουργία, που χρησιμοποιείται στην κρυπτογράφηση, φαίνεται στην **Εικόνα 11** όπου το  $w[i, i+3]$  δείχνει μία ομαδοποίηση ανά 4 words του επεκταμένου κλειδιού, με  $0 \leq i \leq 40$ .

Ο AddRoundKey μετασχηματισμός είναι όσο το δυνατόν πιο απλός και επηρεάζει κάθε bit του State. Η πολυπλοκότητα του round key expansion, καθώς και η πολυπλοκότητα των άλλων σταδίων του AES, διασφαλίζουν την ασφάλεια της εφαρμογής.



**Εικόνα 11. AES Forward Cipher Operation (Pipelining Data Path)**

Το ακόλουθο σχήμα αποτελεί ένα παράδειγμα του AddRoundKey μετασχηματισμού σύμφωνα με την περιγραφή που δόθηκε προηγούμενα:

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC
+			
AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A
=			
EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D2

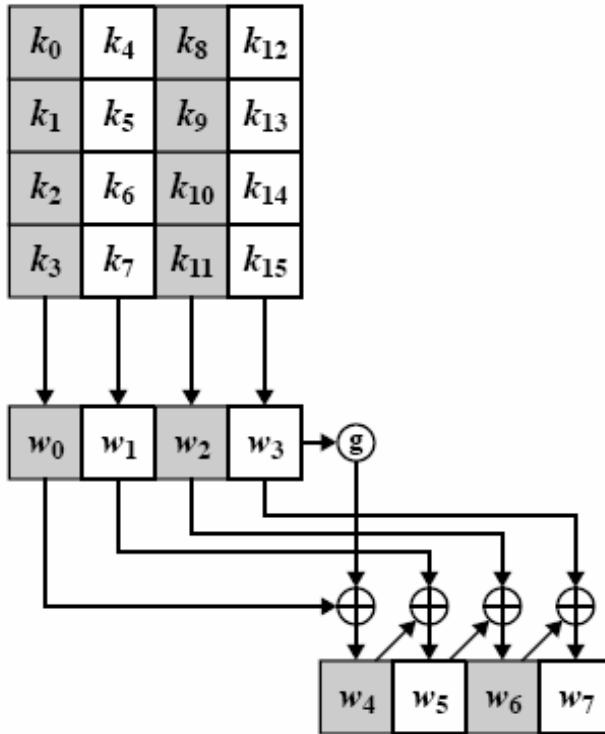
Αναφορικά, η αποκρυπτογράφηση είναι η αντίστροφη διαδικασία της κρυπτογράφησης. Αντιστρέφει τους γύρους μετασχηματισμού για να υπολογίσει το αρχικό plaintext μέσω του κρυπτογραφημένου ciphertext. Ο γύρος μετασχηματισμού της αποκρυπτογράφησης χρησιμοποιεί τις μετατροπές: AddRoundKey, InvMixColumns, InvShiftRows, και InvSubBytes αντίστοιχα. Η AddRoundKey είναι από μόνη της μια αντίστροφη συνάρτηση γιατί η XOR κάνει από μόνη της αντίστροφή. Τα round keys πρέπει να επιλεγούν με αντίστροφη σειρά. Η InvMixColumns απαιτεί διαφορετικό σταθερό πολυώνυμο από την MixColumns. Η InvShiftRows περιστρέφει τα bytes δεξιόστροφα αντί για αριστερόστροφα. Η InvSubBytes αντιστρέφει το S-box look-up table με έναν αντίστροφο affine μετασχηματισμό που ακολουθείται από την ίδια αντίστροφή στο GF(2<sup>8</sup>) με αυτήν που χρησιμοποιήθηκε στην κρυπτογράφηση. Για περισσότερες πληροφορίες σχετικά με την AES αποκρυπτογράφηση, παραπέμπουμε στα [3] και [5].

### 3.5.1 AES Key Expansion

Η διαδικασία επέκτασης του κλειδιού παίρνει το 128-bit κλειδί ως είσοδο και σε κάθε συνεδρία δίνει στην έξοδο ένα 44 32-bits word επεκταμένο κλειδί. Σε κάθε γύρο, ο AES cipher χρησιμοποιεί 4 από τις 44-word του επεκταμένου κλειδιού, στον AddRoundKey μετασχηματισμό, όπως φαίνεται στην **Εικόνα 11**.

Στην **Εικόνα 11** δείχνεται το πώς γίνεται η επέκταση του κλειδιού. Οι 4 πρώτες words του πίνακα εξόδου, ο οποίος δείχνεται ως w, δεν είναι τίποτα άλλο από τα 16-byte (128 bits) εισόδου του κρυφού κλειδιού. Δηλαδή, το κλειδί αντιγράφεται στις τέσσερις πρώτες words του επεκταμένου κλειδιού.

Το υπόλοιπο επεκταμένο κλειδί γεμίζεται ανά τέσσερις λέξεις κάθε φορά. Η κάθε λέξη που προστίθεται w [i] εξαρτάται από την αμέσως προηγούμενη λέξη, w [i - 1], και τη λέξη τέσσερις θέσεις πίσω, w [i - 4]. Στις τρεις από τις τέσσερις περιπτώσεις, μια απλή XOR χρησιμοποιείται. Για μια λέξη της οποίας η θέση στο W array είναι πολλαπλάσιο του 4, μια πιο πολύπλοκη λειτουργία χρησιμοποιείται.



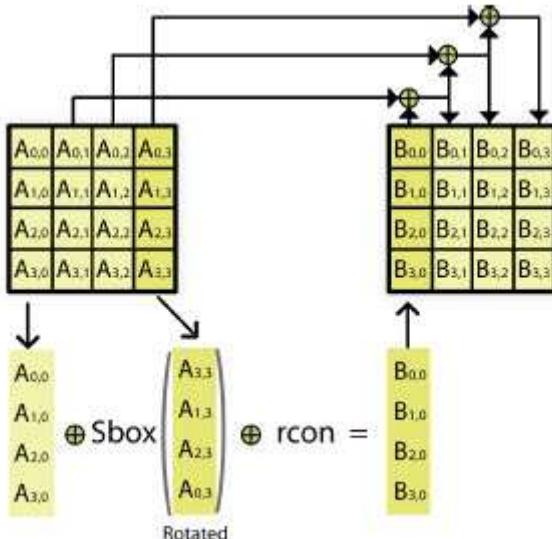
Εικόνα 11. Key Expansion [5]

Στην Εικόνα 11 απεικονίζεται η δημιουργία των πρώτων οκτώ λέξεων του επεκταμένου κλειδιού, χρησιμοποιώντας το σύμβολο  $g$  που εκπροσωπεί την πολύπλοκη λειτουργία. Η λειτουργία  $g$  αποτελείται από τις ακόλουθες υπολειτουργίες:

- **RotWord** - εκτελεί μία κυκλική αριστερόστροφη ολίσθηση κατά ένα byte σε μια λέξη. Αυτό σημαίνει ότι μια λέξη εισόδου [B0, B1, B2, B3] μετατρέπεται σε [B1, B2, B3, B0].
- **SubWord** - εκτελεί μια αντικατάσταση (substitution) ενός byte, σε κάθε byte του μετατοπισμένου αποτελέσματος, κάνοντας χρήση των S-box. Μια λειτουργία παρόμοια με αυτήν που συναντήσαμε στον μετασχηματισμό SubBytes του AES cipher.
- Στο τελευταίο βήμα, το αποτέλεσμα των δύο παραπάνω ενεργειών, δηλαδή η αλλαγμένη word, γίνεται XORed με την σταθερά γύρου (round constant)  $Rcon[i]$ .

Στην **Εικόνα 12** παρακάτω φαίνεται ακριβώς αυτή η διαδικασία ενός single round key computation. Η διαδικασία αυτή επαναλαμβάνεται χρησιμοποιώντας το round key ως cipher για να παράγει τα επόμενα 128 bits του επεκταμένου κλειδιού.

Η τιμή του  $Rcon[i]$  ξεκινά από την τιμή 1 και προσανξάνεται για κάθε κλειδί γύρου.



**Εικόνα 12. AES 128 bit Key Schedule Round**

Η σταθερά γύρου (round constant) είναι μια λέξη τις οποίας τα τρία δεξιότερα bytes είναι πάντα 0. Έτσι, το αποτέλεσμα με XOR μιας λέξης με την Rcon είναι η πραγματοποίηση μίας λειτουργίας XOR για το αριστερό byte της λέξης.

Η round constant είναι διαφορετική για κάθε γύρο και ορίζεται ως  $Rcon[i] = (RC[1], 0, 0, 0)$ , με  $RC[1] = 1$ ,  $RC[i] = 2 \times RC[j - 1]$ , με τον πολλαπλασιασμό να ορίζεται στο πεδίο  $GF(2^8)$ . Οι τιμές της  $RC[i]$  δίνονται για κάθε γύρο στον **Table 2** σε δεκαεξαδική μορφή. Ο λόγος που χρησιμοποιούνται σταθερές γύρου (round constants) είναι για να εξαλειφθούν οι συμμετρίες και ομοιότητες κατά την υλοποίηση του 4-word επεκταμένου κλειδιού κάθε γύρου.

I (round number)	1	2	3	4	5	6	7	8	9	10
RC(i)	01	02	04	08	10	20	40	80	1B	36

**Table 2. Round Constant Bytes, RC in Hexadecimal [5]**

Για παράδειγμα, ας υποθέσουμε ότι το round key για τον όγδοο γύρο είναι :

EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

Τότε τα πρώτα 4 byte (πρώτη στήλη) του round key για τον ένατο γύρο υπολογίζονται ως εξής:

i (decimal)	temp	After RotWord	After SubWord	Rcon (9)	After XOR with Rcon	w[i 4]	w[i] = temp $\oplus$ w[i 4]
36	7F8D292F	8D292F7F	5DA515D2	1B000000	46A515D2	EAD27321	AC7766F3

Οι Rijndael προγραμματιστές σχεδίασαν τον αλγόριθμο επέκτασης κλειδιού ώστε να είναι ανθεκτικός στις γνωστές κρυπτανάλυτικές επιθέσεις. Η συμπερίληψη ενός γύρου που εξαρτάται από την σταθερά γύρου καταργεί τη συμμετρία, ή τις ομοιότητες, μεταξύ των τρόπων με τους οποίους τα round keys δημιουργούνται στους διάφορους γύρους. Τα συγκεκριμένα κριτήρια που χρησιμοποιήθηκαν είναι τα εξής :

- Η γνώση ενός μέρους των cipher key ή round key δεν επιτρέπει τον υπολογισμό πολλών άλλων round key bits.
- Αντιστρέψιμος μετασχηματισμός, δηλαδή, η γνώση οποιονδήποτε  $N_K$  συνεχόμενων λέξεων του Expanded Key επιτρέπει αναγέννηση ολόκληρου του επεκταμένου κλειδιού (όπου  $N_K$  = key size in words)
- Η ταχύτητα σε ένα ευρύ φάσμα επεξεργαστών.
- Χρήση των σταθερών γύρου για την εξάλειψη συμμετριών.
- Διάχυση των cipher key διαφορών μέσα στα διαφορά round keys, όπου αυτό σημαίνει ότι κάθε key bit επηρεάζει πολλά round key bits.
- Αρκετή μη-γραμμικότητα για να αποφευχθεί ο πλήρης καθορισμός των round key διαφορών μόνο από τις διαφορές του cipher key.
- Η απλότητα της περιγραφής.

Το πρώτο σημείο της προηγούμενης λίστας δεν ποσοτικοποιείται, αλλά η ιδέα είναι ότι εάν γνωρίζουμε λιγότερο από  $N_K$  συνεχόμενες λέξεις είτε του cipher κλειδιού ή ένα από τα κλειδιά γύρου, τότε είναι δύσκολο να ανακατασκευαστούν τα υπόλοιπα άγνωστα bit. Όσο λιγότερα bits γνωρίζει κάποιος, τόσο πιο δύσκολο είναι να κάνει την ανοικοδόμηση ή να καθορίσει και άλλα bits του επεκταμένου κλειδιού.



## Κεφάλαιο 4

# Galois / Counter Mode (GCM)

Ο Galois / Counter Mode (GCM) για λειτουργία σε μπλοκ ciphers αλγόριθμους σχεδιάστηκε για να ικανοποιήσει την ανάγκη για πιστοποιημένο τρόπο κρυπτογράφησης που να μπορεί να συμβάλει αποτελεσματικά στην επίτευξη ταχυτήτων των 10 gigabits ανά δευτερόλεπτο, και υψηλότερο, σε υλικό, να μπορεί να έχει καλές επιδόσεις και στον τομέα του λογισμικού και να είναι απαλλαγμένο από πνευματικούς περιορισμούς ιδιοκτησίας.

Η κατάσταση λειτουργίας μετρητή (CTR) έχει γίνει ο τρόπος λειτουργίας που επιλέγεται για εφαρμογές υψηλής ταχύτητας διότι μπορεί να χρησιμοποιήσει αποτελεσματικά τις τεχνικές διοχέτευσης σε hardware υλοποιήσεις. Ωστόσο, δεν παρέχει μήνυμα γνησιότητας (message authentication). Ο GCM ενσωματώνει τον CTR και στηρίζεται πάνω του με την προσθήκη ενός κώδικα γνησιότητας μηνύματος (MAC) με βάση μια καθολική hashing. Χρησιμοποιεί πολυωνυμικό hashing στο πεπερασμένο πεδίο  $GF(2^W)$ , η βασική λειτουργία του οποίου είναι πολλαπλασιασμός με ένα σταθερό στοιχείο πεδίου (fixed field element). Το δυαδικό πεδίο πολλαπλασιασμού μπορεί να υλοποιηθεί εύκολα σε hardware, και μπορεί να γίνει σε εκπληκτικά αποτελεσματικό λογισμικό μέσω μεθόδων οδηγούμενων από πίνακα (table-driven methods). Επιπλέον, ο GCM μπορεί να χρησιμοποιηθεί ως αυτόνομο (stand-alone) MAC, και μπορεί να χρησιμοποιηθεί και ως πρόσθετο MAC.

Μια απλή προσέγγιση για το σχεδιασμό υλικού υψηλών ταχυτήτων μιας GCM αρχιτεκτονικής είναι να χρησιμοποιήσουμε γρήγορες εφαρμογές των πυρήνων του AES και του GF πολλαπλασιαστή. Αποτελεσματικές hardware υλοποιήσεις του AES και του πολλαπλασιαστή GF έχουν μελετηθεί εκτενώς. Για παράδειγμα, διαφορές εφαρμογές, όπως η αναζήτηση πίνακα, τα σύνθετα πεδία και τα δυαδικά διαγράμματα αποφάσεων, έχουν προταθεί για να βελτιστοποιήσουμε το S-box κύκλωμα που κυριαρχεί στη κρίσιμη διαδρομή του κυκλώματος του AES. Ο πολλαπλασιαστής Galois πεδίου μπορεί να βελτιστοποιηθεί για ορισμένους συγκεκριμένους τύπους modulus πολυωνύμων ή χρησιμοποιώντας διαφορετικές βάσεις για την εκπροσώπηση.

Όλα τα παραπάνω όμως θα φανούν καλύτερα αφού πρώτα πραγματοποιηθεί μια ολοκληρωμένη ανάλυση του Galois / Counter Mode σύμφωνα με όσα περιγράφονται στο τέταρτο recommendation που αφορά στις καταστάσεις λειτουργίας των συμμετρικών block ciphers κλειδιού, όπως αντό αναπτύχθηκε από το National Institute of Standards and Technology (NIST) κατ 'επέκταση των αρμοδιοτήτων βάσει του Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

## 4.1 Τα Στοιχειώδη Μέρη του GCM

Τα στοιχειώδη μέρη του GCM και τα συσχετιζόμενα σχόλια και απαιτήσεις παρουσιάζονται παρακάτω στις ακόλουθες ενότητες. Το block cipher και το κλειδί συζητούνται στην ενότητα 4.1.1. Τα data elements των συναρτήσεων που παρέχουν την πιστοποιημένη κρυπτογράφηση (authenticated encryption) και πιστοποιημένη αποκρυπτογράφηση (authenticated decryption) του GCM συζητούνται στην ενότητα 4.1.2. Τα κρυπτογραφικά βασικά στοιχεία (primitives) που παρέχουν την εμπιστευτικότητα και την πιστοποίηση εσωτερικά στις δύο παραπάνω λειτουργίες παρουσιάζονται στην ενότητα 4.1.3. Οι τύποι των εφαρμογών του GCM που είναι διαθέσιμοι, συνοψίζονται στην ενότητα 4.1.4.

### 4.1.1 Block Cipher

Στο AES-GCM standard, οι λειτουργίες που περιγράφονται, εξαρτώνται από την επιλογή ενός βασικού συμμετρικού κλειδιού block cipher AES και επομένως μπορούν να θεωρηθούν ως κατάσταση λειτουργίας (mode of operation) του block cipher. Το AES-GCM κλειδί είναι το block cipher κλειδί.

Για κάθε δοσμένο κλειδί, το αντίστοιχο block cipher του mode, συνίσταται από δύο λειτουργίες που είναι αντίστροφες η μία της άλλη. Η επιλογή του block cipher περιλαμβάνει την ονομασία μίας από τις δύο λειτουργίες του block cipher, ως η forward cipher function, όπως και στις προδιαγραφές του αλγορίθμου AES. Ο AES-GCM υποδεικνύει την λειτουργία κρυπτογράφησης (encryption function) του AES block cipher καθώς η forward cipher function, είναι στην πραγματικότητα ο AES σε ECB mode. Ο GCM δεν χρησιμοποιεί την inverse cipher function.

Η forward cipher function είναι μια αναδιάταξη (permutation) πάνω σε bit strings σταθερού μήκους, τα strings ονομάζονται μπλοκ. Το μήκος ενός μπλοκ ονομάζεται το μέγεθος μπλοκ (block size). Το κλειδί παριστάνεται ως  $K$ , και η προκύπτουσα forward cipher λειτουργία του κρυπτογραφικού αλγόριθμου χαρακτηρίζεται ως  $CIPH_K$ .

Το βασικό block cipher θα πρέπει να είναι εγκεκριμένο, το μέγεθος μπλοκ να είναι 128 bits, και το μέγεθος κλειδιού θα πρέπει να είναι τουλάχιστον 128 bits. Το κλειδί πρέπει να παράγεται τυχαία με ομοιόμορφο τρόπο ή τυχαία με σχεδόν ομοιόμορφο τρόπο, έτσι ώστε κάθε δυνατό κλειδί να είναι (σχεδόν) εξίσου πιθανό να δημιουργηθεί. Συνεπώς, το κλειδί θα είναι φρέσκο (fresh), δηλαδή, άνισο με κάθε προηγούμενο κλειδί, με μεγάλη πιθανότητα. Το κλειδί πρέπει να εγκαθίσταται μυστικά μεταξύ των εμπλεκόμενων φορέων που συμμετέχουν στην επικοινωνία και να χρησιμοποιείται αποκλειστικά για τον GCM με το επιλεγμένο block cipher. Πρόσθετες απαιτήσεις σχετικά με τη δημιουργία και τη διαχείριση των κλειδιών συζητούνται παρακάτω στην ενότητα 4.7.

## 4.1.2 Οι δύο GCM λειτουργίες

Ο GCM αποτελείται από δύο λειτουργίες που ονομάζονται πιστοποιημένη κρυπτογράφηση (authenticated encryption) και πιστοποιημένη αποκρυπτογράφηση (authenticated decryption).

Η πιστοποιημένη κρυπτογράφηση κρυπτογραφεί τα confidential data και υπολογίζει ένα authentication tag (T) κοινό για confidential data και για τα AAD .

Η πιστοποιημένη αποκρυπτογράφηση αποκρυπτογραφεί τα confidential data και μέσω του authentication tag επαληθεύει την γνησιότητά τους.

Μια υλοποίηση μπορεί να περιορίζει τα δεδομένα εισόδου σε non-confidential data χωρίς καθόλου confidential data. Η προκύπτουσα εκδοχή του GCM καλείται GMAC. Συνεπώς, στην περίπτωση που έχουμε μόνο AAD το αποτέλεσμα της GMAC λειτουργίας είναι αντίστοιχα ο υπολογισμός και η επιβεβαίωση του Tag για τα non confidential data.

### 4.1.2.1 Authenticated Encryption

#### Input Data

Με δεδομένα τον εγκεκριμένο block cipher και το κλειδί, υπάρχουν τρία strings εισόδου στην λειτουργία της πιστοποιημένης κρυπτογράφησης :

- Ένα Plaintext, που δείχνεται ως P και μπορεί να έχει μήκος από 0 bits μέχρι  $2^{39}$  – 256 bits.
- Τα Additional authentication data (AAD), που δείχνονται ως A και μπορούν να έχουν μήκος από 0 μέχρι  $2^{64}$  bits.
- Ένα Initialization Vector, που δείχνεται ως IV και μπορεί να έχει οποιοδήποτε μήκος μεταξύ 1 και  $2^{64}$ -1 bits.

Ο IV αποτελεί ουσιαστικά μία nonce, δηλαδή, μια τιμή που είναι μοναδική εντός του καθορισμένου πλαισίου της εφαρμογής, και η οποία καθορίζει μία επίκληση της Authenticated Encryption λειτουργίας για τα δεδομένα εισόδου που πρέπει να προστατευτούν. Η απαίτηση σχετικά με την μοναδικότητα των IVs (και κλειδιού) αναφέρεται με ακρίβεια στην ενότητα 4.4 και δύο frameworks για την κατασκευή IVs δίνονται στην ενότητα 4.4.2.

*Εδώ χρησιμοποιείται ένας 96 bits IV για να επιτυγχάνεται απλότητα στον σχεδιασμό, αποτελεσματικότητα και διαλειτουργικότητα, όπως προτείνεται στη βιβλιογραφία.*

Το plaintext και τα AAD, είναι οι δύο κατηγορίες δεδομένων που προστατεύει ο GCM. Ο GCM επαληθεύει την γνησιότητα (authenticity) του P και του AAD και ταυτόχρονα προστατεύει την εμπιστευτικότητα (confidentiality) του P ενώ το AAD μεταδίδεται καθαρό χωρίς την ανάγκη για κρυπτογράφηση. Για παράδειγμα, σε ένα πρωτόκολλο δικτύου, τα AAD θα μπορούσαν να περιλαμβάνουν τις διευθύνσεις, τις θύρες, τα sequences numbers, τους αριθμούς έκδοσης πρωτόκολλου, καθώς και άλλους τομείς που δείχνουν το πώς πρέπει να αντιμετωπίζεται το plaintext.

Παρόλο που ο GCM ορίζεται για bit strings, τα μήκη των plaintext, AAD, και του IV, πρέπει να είναι όλα τα πολλαπλάσια του 8, έτσι ώστε οι τιμές αυτές να είναι byte strings.

Μια εφαρμογή μπορεί να περιορίσει τα μήκη από αυτά τα δεδομένα, σύμφωνα με τις παραπάνω απαιτήσεις. Για παράδειγμα, μια εφαρμογή μπορεί να συστήνει μικρότερες μέγιστες τιμές των εισόδων. Τα μήκη που επιτρέπει μια εφαρμογή ονομάζονται υποστηριζόμενα μήκη (supported bit lengths). Ένα ενιαίο σύνολο για τα μήκη για κάθε μία από τις τρεις εισόδους, θα πρέπει να καθορισθεί για ολόκληρη την εφαρμογή, ανεξάρτητα από το κλειδί.

## Output Data

Τα ακόλουθα δύο bit strings αποτελούν τα δεδομένα εξόδου στην λειτουργία της πιστοποιημένης κρυπτογράφησης :

→ To Ciphertext, που δείχνεται ως C και το μήκος του είναι ακριβώς το ίδιο με αυτό του plaintext.

→ Ένα Authentication tag, που δείχνεται ως T με μήκος μια από τις ακόλουθες τιμές 128, 120, 112, 104, 96, (64, 32). Το μήκος του T δείχνεται ως t.

Η δύναμη της γνησιότητας των P, A, IV καθορίζεται από το μήκος του T,  $t = \text{len}(T)$ .

Το μήκος του tag, είναι μια παράμετρος ασφάλειας, όπως αναφέρεται στο παράρτημα B του NIST. Σε γενικές γραμμές, το t μπορεί να είναι μία από τις ακόλουθες πέντε τιμές: 128, 120, 112, 104, ή 96. Για ορισμένες εφαρμογές, το t μπορεί να είναι 64 ή ακόμα και 32. Καθοδήγηση για τη χρήση αυτών των δύο μηκών του tag, συμπεριλαμβανομένων των απαιτήσεων σχετικά με το μήκος των δεδομένων και τη διάρκεια ζωής του κλειδιού σε αυτές τις περιπτώσεις, δίνεται στο παράτημα Γ του NIST.

Μια εφαρμογή δεν μπορεί να υποστηρίζει τιμές για το t διαφορετικές από τις επτά επιλογές που αναφέρθηκαν. Μια εφαρμογή μπορεί να περιορίσει το υποστηριζόμενο μήκος σε μέχρι μία από τις τιμές αυτές. Μια ενιαία, σταθερή τιμή για το t, ανάμεσα από τις τιμές που υποστηρίζονται από τις παραπάνω επιλογές, πρέπει να συνδέεται με κάθε κλειδί.

#### **4.1.2.2 Authenticated Decryption**

Με δεδομένα τον εγκεκριμένο block cipher, το κλειδί και το αντίστοιχο μήκος του tag, οι είσοδοι στην λειτουργία της πιστοποιημένης αποκρυπτογράφησης είναι οι τιμές των :

- Initialization Vector (IV)
- Ciphertext (C)
- Additional authentication data (AAD) (A)
- Authentication tag (T)

Όπως αυτές περιγράφηκαν παραπάνω.

Η έξοδος είναι μία από τα ακόλουθα:

- To Plaintext P που αντιστοιχεί στο Ciphertext C
- ή ένας ειδικός κωδικός σφάλματος, δηλαδή ένα indication Fail, που υποδεικνύει ότι τα δεδομένα εισόδου δεν ήταν γνήσια.

Επίσης Fail παίρνουμε και στην περίπτωση που τα δεδομένα εισόδου δεν έχουν κρυπτογραφηθεί με το συγκεκριμένο κλειδί.

Η GCM πιστοποιημένη αποκρυπτογράφηση υπολογίζει το authentication tag T' βασισμένη στα δεδομένα εισόδου και το συγκρίνει με το authentication tag T που έχει λάβει. Αν τα δύο tag, T και T', είναι ίδια τότε το P θα είναι η έξοδος της authenticated decryption function. Άλλιώς το FAIL θα αποτελεί την έξοδο.

Οι τιμές για τα len (C), len (A), και len (IV) που υποστηρίζει η εφαρμογή για την πιστοποιημένη αποκρυπτογράφηση θα πρέπει να είναι οι ίδιες με τις τιμές των len (P), len (A), και len (IV) που υποστηρίζει η εφαρμογή για τη λειτουργία της πιστοποιημένης κρυπτογράφησης.

#### **4.1.3 Εμπιστευτικότητα και Πιστοποίηση**

Ο μηχανισμός για την προστασία του απορρήτου του plaintext εντός του GCM είναι μια παραλλαγή του Counter mode, με μια συγκεκριμένη προσαυξητική συνάρτηση, για να παράγει την απαραίτητη ακολουθία των counter μπλοκ. Το πρώτο counter μπλοκ για την κρυπτογράφηση του plaintext παράγεται από ένα μπλοκ που αυξάνεται και το οποίο δημιουργείται από το IV.

Ο μηχανισμός πιστοποίησης εντός του GCM βασίζεται σε μια hash συνάρτηση, που ονομάζεται GHASH, και περιγράφει τα χαρακτηριστικά του πολλαπλασιασμού με μια σταθερή παράμετρο, που ονομάζεται hash δευτερεύον κλειδί (hash subkey), μέσα σε ένα δυαδικό Galois πεδίο.

Το δευτερεύον κλειδί hash, που παριστάνεται με το  $H$ , παράγεται με την εφαρμογή του block cipher στο "μηδενικό" μπλοκ. Το αποτέλεσμα που προκύπτει από αυτή την hash συνάρτηση, το οποίο παριστάνεται ως  $\text{GHASH}_H$ , χρησιμοποιείται για τη συμπίεση μιας κωδικοποίησης των AAD και ciphertext σε ένα ενιαίο μπλοκ, το οποίο στη συνέχεια κρυπτογραφείται για την παραγωγή της ετικέτας αυθεντικότητας (authentication tag).

Η GHASH είναι μια keyed hash λειτουργία, αλλά όχι από μόνη της μια κρυπτογραφική hash λειτουργία. Η παρούσα σύσταση εγκρίνει μόνο GHASH για χρήση στο πλαίσιο του GCM.

Οι ενδιάμεσες τιμές κατά την εκτέλεση του GCM είναι απόρρητες. Ειδικότερα, η απαίτηση αυτή αποκλείει ένα σύστημα στο οποίο ο GCM έχει υλοποιηθεί με τη χρήση του hash subkey δημοσίως για κάποιο άλλο σκοπό, όπως για παράδειγμα, σαν μια απρόβλεπτη τιμή ή ως έλεγχος της ακεραιότητας της τιμής του κλειδιού.

#### 4.1.4 Τύποι Εφαρμογών του GCM

Υπάρχουν τέσσερις τύποι εφαρμογών του GCM που προτείνονται στο SP800-380D. Αυτοί είναι:

- 1) GCM με αυθαίρετου μεγέθους IV.
- 2) GCM με καθορισμένου μεγέθους (default) IV, συγκεκριμένα, στην ανάπτυξη αυτή, το μέγεθος του IV περιορίζεται ακριβώς στα 96 bits.
- 3) GMAC, δηλαδή ο αλγόριθμος που παράγει ένα stand-alone authentication tag  $T$  για τα AAD με IV αυθαίρετου μεγέθους. Το plaintext  $P$  είναι το κενό string.
- 4) GMAC με καθορισμένο IV.

Σε αυτήν την ανάπτυξη επιλέγεται η εφαρμογή GCM με τον default IV και θα συζητηθεί παρακάτω λεπτομερώς.

## 4.2 Τα Βασικά μαθηματικά στοιχεία του GCM

Η ενότητα αυτή παρουσιάζει τα μαθηματικά στοιχεία που περιλαμβάνονται στη προδιαγραφή των υποχρεώσεων (specification) στις λειτουργίες της πιστοποιημένης κρυπτογράφησης και αποκρυπτογράφησης στην ενότητα 4.3 κατωτέρω. Ο Algorithm 1, για την GHASH λειτουργία που είναι κατασκευασμένη από τον πολλαπλασιασμό σε πεπερασμένο δυαδικό Galois πεδίο, ορίζεται στην ενότητα 4.2.1. Ο Algorithm 2 για την GCTR λειτουργίας καθορίζεται στην ενότητα 4.2.2.

Οι προδιαγραφές των Αλγόριθμων 1, 2 περιλαμβάνουν τις εισόδους, τις εξόδους, τα βήματα του αλγορίθμου, διαγράμματα, καθώς και περιλήψεις. Ισοδύναμα σύνολα ενεργειών που παράγουν την σωστή έξοδο επιτρέπονται. Οι είσοδοι που είναι κατά κανόνα καθορισμένες σε πολλές επικλήσεις (invocations) της λειτουργίας καλούνται ως προϋποθέσεις (Prerequisites), αν και μπορεί επίσης να θεωρηθούν ως (κυματινόμενες) είσοδοι.

### 4.2.1 GHASH Function

Ο μηχανισμός της πιστοποίησης (authentication mechanism) στον GCM βασίζεται σε μία Hash function που καλείται GHASH και κάνει πολλαπλασιασμό με μία fixed παράμετρο που λέγεται hash subkey ( $H$ ) μέσα στο binary Galois Field  $GF(2^{128})$ . Το hash subkey παράγεται εφαρμόζοντας το block cipher σε ένα zero block και το αποτέλεσμα αυτής της πράξης δείχνεται ως  $GHASH_H$ . Αυτό χρησιμοποιείται για να συμπιέσει μία κωδικοποίηση των AAD και ciphertext σε ένα μόνο block το οποίο μετά κρυπτογραφείται για να παράγει το authentication tag. Η GHASH είναι μία hash function κλειδιού αλλά από μόνη της δεν είναι μια κρυπτογραφική hash function. Ουσιαστικά αυτό που προσφέρει η GHASH είναι να υπογράφει μοναδικά το AAD και το ciphertext.

Ο **Algorithm 1** παρακάτω δείχνει την λειτουργία που θα κληθεί μέσα στους μηχανισμούς πιστοποιημένης κρυπτογράφησης και αποκρυπτογράφησης του AES-GCM.

#### Algorithm 1 below specifies the GHASH function:

Algorithm 1:  $GHASH_H(X)$

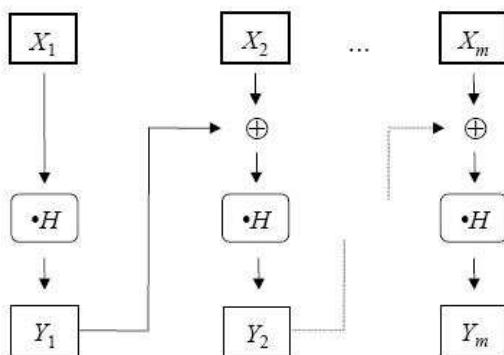
Prerequisites - προϋποθέσεις:  
block  $H$ , the hash subkey.

Input:  
bit string  $X$  such that  $\text{len}(X) = 128m$  for some positive integer  $m$ .

Output:  
block  $GHASH_H(X)$ .

Steps:

1. Let  $X_1, X_2, \dots, X_{m-1}, X_m$  denote the unique sequence of blocks such that  $X = X_1 \| X_2 \| \dots \| X_{m-1} \| X_m$ .
2. Let  $Y_0$  be the “zero block,”  $0^{128}$ .
3. For  $i = 1, \dots, m$ , let  $Y_i = (Y_{i-1} \{ \text{xor} \} X_i) \bullet H$ .
4. Return  $Y_m$ .



**Εικόνα 1:  $\text{GHASH}_H(X_1 \| X_2 \| \dots \| X_m) = Y_m$ .**

Τελικά, η GHASH function υπολογίζει το :

$$X_1 \bullet H^m \{ \text{xor} \} X_2 \bullet H^{m-1} \{ \text{xor} \} \dots \{ \text{xor} \} X_{m-1} \bullet H^2 \{ \text{xor} \} X_m \bullet H.$$

Η GHASH function φαίνεται στην **Εικόνα 1** παραπάνω, χωρίς το zero block  $Y_0$  αφού αυτό όταν γίνει exclusive-OR με το  $X_1$  δεν αλλάζει το  $X_1$ .

Για περισσότερη ευκολία στην κατανόηση του Algorithm 1, δίνεται παρακάτω μια περιγραφή σε ψευδοκώδικα :

### Algorithm 1

```

Y ← 0
for i = 1 to m do
    Y ← (Y { xor } X_i) • H
end for

return Y
  
```

Η GHASH δηλαδή είναι μια λειτουργία  $m$  βαθμίδων, όπου το  $m$  αντιστοιχεί στον αριθμό των πακέτων των 128 bits εισόδου, ξεκινά από την βαθμίδα 1 πολλαπλασιάζοντας το πρώτο πακέτο εισόδου των 128 bits με το hash subkey μέσα στο  $GF(2^{128})$  και στη συνέχεια, για τον υπολογισμό των επόμενων βαθμίδων χρησιμοποιεί το αποτέλεσμα της προηγούμενης βαθμίδας για να το κάνει XOR με το νέο πακέτο των 128 bits εισόδου. Αυτό που προκύπτει από την XOR πράξη πολλαπλασιάζεται με το  $H$  μέσα στο  $GF(2^{128})$ . Ουσιαστικά, η  $\text{GHASH}_H$  είναι μια σειριακή λειτουργία καθώς κάθε βαθμίδα πρέπει να περιμένει την έξοδο της

προηγούμενης βαθμίδας για να ολοκληρωθεί ο υπολογισμός. Για το λόγο αυτό, δεν μπορεί να χρησιμοποιηθεί τεχνική παραλληλισμού στην υλοποίηση της GHASH (δύο διαδοχικά πακέτα ταυτόχρονα). Επιπλέον, λόγο της σειριακής εκτέλεσης, η καθυστέρηση της GHASH εξαρτάται από την καθυστέρηση του πολλαπλασιαστή, εφόσον η ολική καθυστέρηση της GHASH είναι μια φορές επί την καθυστέρηση του πολλαπλασιαστή. Είναι συνεπώς καθοριστικός ο παράγοντας της σωστής επιλογής ενός γρήγορου πολλαπλασιαστή. Άρα σύμφωνα με όσα αναφέρθηκαν παραπάνω, η επιλογή ενός παράλληλου πολλαπλασιαστή αποτελεί μονόδρομο εφόσον ενδιαφερόμαστε για θέματα ταχύτητας και απόδοσης του AES-GCM.

#### 4.2.2 GCTR Function

Ο μηχανισμός της εμπιστευτικότητας (confidentiality) του GCM είναι μια παραλλαγή του CTR mode, που ονομάζεται GCTR, χρησιμοποιώντας μια συγκεκριμένη incrementing function, που δείχνεται ως inc, για να παράγει την απαραίτητη ακολουθία των counter blocks. Το πρώτο counter block για την κρυπτογράφηση του plaintext παράγεται αυξάνοντας ένα block το οποίο προέρχεται από τον IV.

Ο **Algorithm 2** παρακάτω περιγράφει την GCTR Function που θα κληθεί μέσα στους αλγόριθμους κατά τις λειτουργίες της GCM πιστοποιημένης κρυπτογράφησης και αποκρυπτογράφησης.

**Algorithm 2 below specifies the  $\text{GCTR}_K(\text{ICB}, X)$  function:**

Input:

1. Bit string X, of arbitrary length
2. Initial counter block ICB, i.e. IV or some value generated from IV
3. Approved block cipher CIPH (such as AES) with a 128-bit block size
4. Key K

Output:

Bit string Y of bit length  $\text{len}(X)$ .

Steps:

0. If  $X$  is the empty string, then return the empty string as  $Y$ .
1. Let  $n = [\text{Len}(X)/128]$ .
2. Let  $X_1, X_2, \dots, X_{n-1}, X_n^*$  denote the unique sequence of bit strings such that  

$$X = X_1 \parallel X_2 \parallel \dots \parallel X_{n-1} \parallel X_n^*$$
.
  3. Let  $\text{CB}_1 = \text{ICB}$ .
  4. For  $i=2$  to  $n$ , let  $\text{CB}_i = \text{inc}(\text{CB}_{i-1})$ .

5. For  $i = 1$  to  $n-1$ , let  $Y_i = X_i \{ \text{xor} \} \text{CIPH}_K(CB_i)$ .

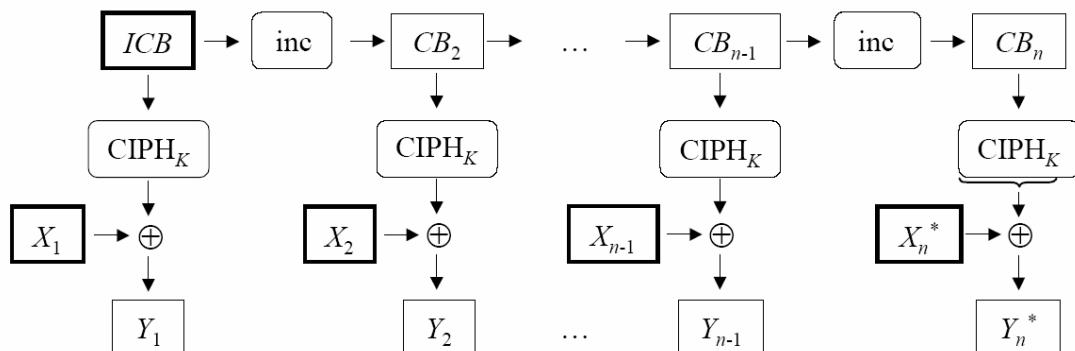
6. Let  $Y_n^* = X_n^* \{ \text{xor} \} \text{MSB}_{\text{len}(X^*n)}(\text{CIPH}_K(CB_n))$ .

7. Let  $Y = Y_1 \parallel Y_2 \parallel \dots \parallel Y_{n-1} \parallel Y_n^*$ .

8. Return  $Y$ .

Σύντομη περιγραφή του GCTR Algorithm :

Στο step1 το string εισόδου αυθαίρετου μεγέθους χωρίζεται σε μια ακολουθία από blocks των 128 bits έτσι ώστε μόνο το rightmost string της σειράς να μπορεί να είναι είτε ένα complete block είτε ένα nonempty partial block. Στα steps 2, 3, 4 η incrementing function  $\text{inc}_{32}$  επαναλαμβάνεται συνεχώς στο ICB ώστε να παράγεται μια σειρά από CB. Στα steps 5 και 6 το block cipher εφαρμόζεται στα CB και το αποτέλεσμα που προκύπτει γίνεται XORed με τα αντίστοιχα blocks του string εισόδου. Στο step 7 η ακολουθία των αποτελεσμάτων που προέκυψαν συνενώνεται για να παραχθεί η έξοδος.



**Εικόνα 2.  $\text{GCTR}_K(\text{ICB}, \text{X1} \parallel \text{X2} \parallel \dots \parallel \text{X}^*n) = \text{Y1} \parallel \text{Y2} \parallel \dots \parallel \text{Y}^*n$ . [2]**

## 4.3 GCM Specification

Οι Algorithms 3 και 4 για την πιστοποιημένη κρυπτογράφηση και πιστοποιημένη αποκρυπτογράφηση του GCM ορίζονται στις δύο παρακάτω ενότητες. Αυτές οι προδιαγραφές περιλαμβάνουν τις εισόδους, εξόδους, τα βήματα τα διαγράμματα και μια περιγραφή των αλγορίθμων. Οι προτεινόμενοι αλγόριθμοι δεν προσδιορίζουν την επιλογή του προαπαιτούμενου block cipher, γνωρίζουμε βέβαια ότι στην ανάπτυξη της διπλωματικής αυτής περιγράφεται ο AES-GCM. Οι είσοδοι, που είναι κατά κανόνα καθορισμένες σε πολλές επικλήσεις (invocations) της λειτουργίας, καλούνται ως προϋποθέσεις (prerequisites). Ωστόσο, ορισμένες από τις απαραίτητες προϋποθέσεις μπορούν επίσης να θεωρηθούν ως (κυμαινόμενες) είσοδοι. Οι προϋποθέσεις και τις λοιπές είσοδοι πρέπει να πληρούν τις απαιτήσεις της ενότητας 4.1.2.1 και 4.1.2.2 αντίστοιχα.

### 4.3.1 Authenticated Encryption

O Algorithm 3 παρακάτω εκτελεί την authenticated encryption function.

#### Algorithm 3: AES-GCM-AE<sub>K</sub> (IV, P, A)

Input:

1. Block cipher CIPH (i.e. AES) with a 128-bit block size
2. Key K
3. Tag length t.
4. Initialization vector IV
5. Plaintext P
6. Additional authenticated data A.

Output:

1. Cipher text C
2. Authentication tag T.

Steps:

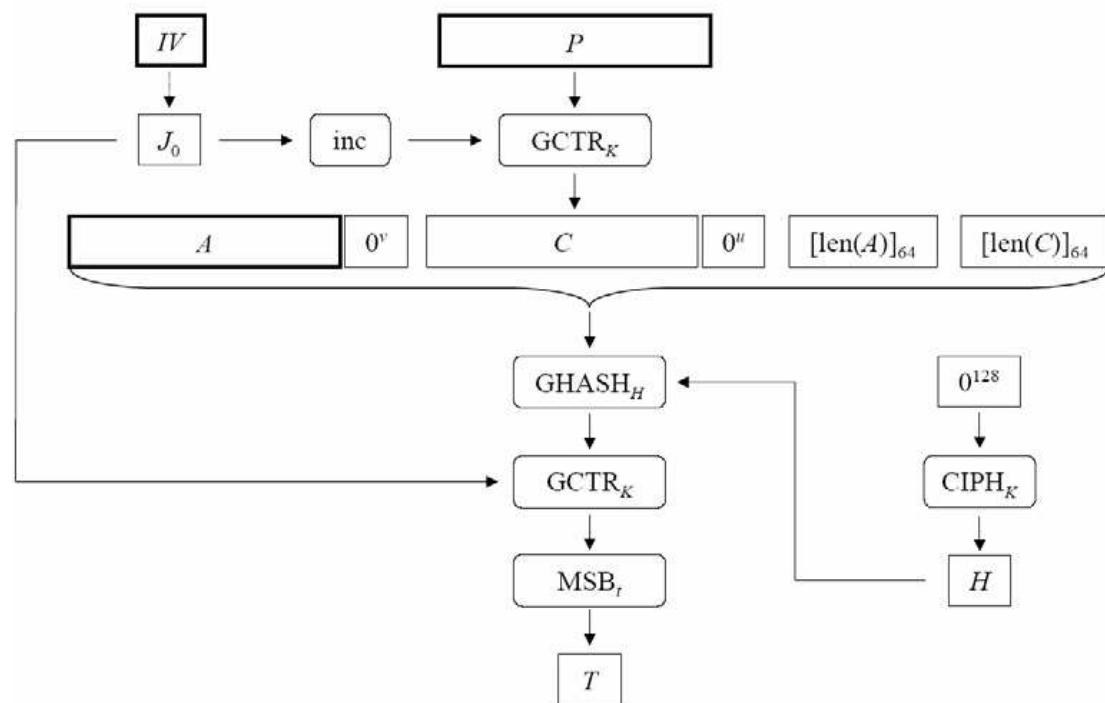
1. Let  $H = \text{CIPH}_K(0^{128})$
2. Define a block,  $J_0$ , as follows:  
 $J_0 = IV \| 0^{31}1$ , i.e.  $J_0$  is a 128-bit string consisted of 96-bit IV, 31 ‘0’ bits, and 1 ‘1’ bit.
3. Let  $C = \text{GCTR}_K(\text{inc}(J_0), P)$ .
4. Let  $u = 128 * [\text{len}(C)/128]-\text{len}(C)$ , and let  $v = 128 * [\text{len}(A)/128]-\text{len}(A)$
5. Define a block, S, as follows:  $S = \text{GHASH}_H(A \| 0^v \| C \| 0^u \| [\text{len}(A)]_{64} \| [\text{len}(C)]_{64})$

6. Let  $T = \text{MSB}_t(\text{GCTR}_K(J_0, S))$ .

7. Return  $(C, T)$ .

Στο step 1 το hash subkey  $H$  για την GHASH Function παράγεται, εφαρμόζοντας το block cipher στο zero block. Στο step 2, το pre-counter block ( $J$ ) παράγεται από τον IV. Πιο συγκεκριμένα εφόσον το μήκος του IV περιορίζεται αυστηρά στα 96 bits, τότε το padding string  $0^{31}||1$ , προσάπτεται στον IV για να σχηματίσει το pre-counter block των 128 bits. Στο step 3, η 32-bit incrementing function εφαρμόζεται στο pre-counter block για να δημιουργηθεί το initial counter block (ICB) για μια κλίση της GCTR function για το plaintext. Το αποτέλεσμα της κλίσης της GCTR function είναι το ciphertext. Στα steps 4 και 5, καθένα από τα AAD και ciphertext προσάπτονται με τον ελάχιστο αριθμό μηδενικών bits, πιθανώς και κανένα, έτσι ώστε το bit length των τερματικών strings να είναι να είναι πολλαπλάσιο του block size. Η συνένωση αυτών των strings προσάπτεται με την 64-bit αναπαράσταση του μήκους των AAD και ciphertext, και η GHASH function εφαρμόζεται στο αποτέλεσμα για να παράγει ένα single block εξόδου. Στο step 6 αντό το block εξόδου κρυπτογραφείται χρησιμοποιώντας την GCTR function και το pre-counter block που είχε παραχθεί στο step 2 και το αποτέλεσμα περικόπτεται στο specified tag length για να σχηματίσει το authentication tag.

H authenticated encryption function εικονίζεται στην **Εικόνα 3** παρακάτω.



**Εικόνα 3. AES-GCM-AE<sub>K</sub> (IV, P, A) = (C, T).[2]**

### 4.3.2 Authenticated Decryption

Algorithm 4 παρακάτω εκτελεί την authenticated decryption function.

#### Algorithm 4: AES-GCM-AD<sub>K</sub> (IV, C, A, T)

Input:

1. Block cipher CIPH (i.e. AES) with a 128-bit block size
2. Key K
3. Tag length t.
4. Initialization vector IV
5. Ciphertext C
6. Additional authenticated data A.
7. Authentication tag T.

Output:

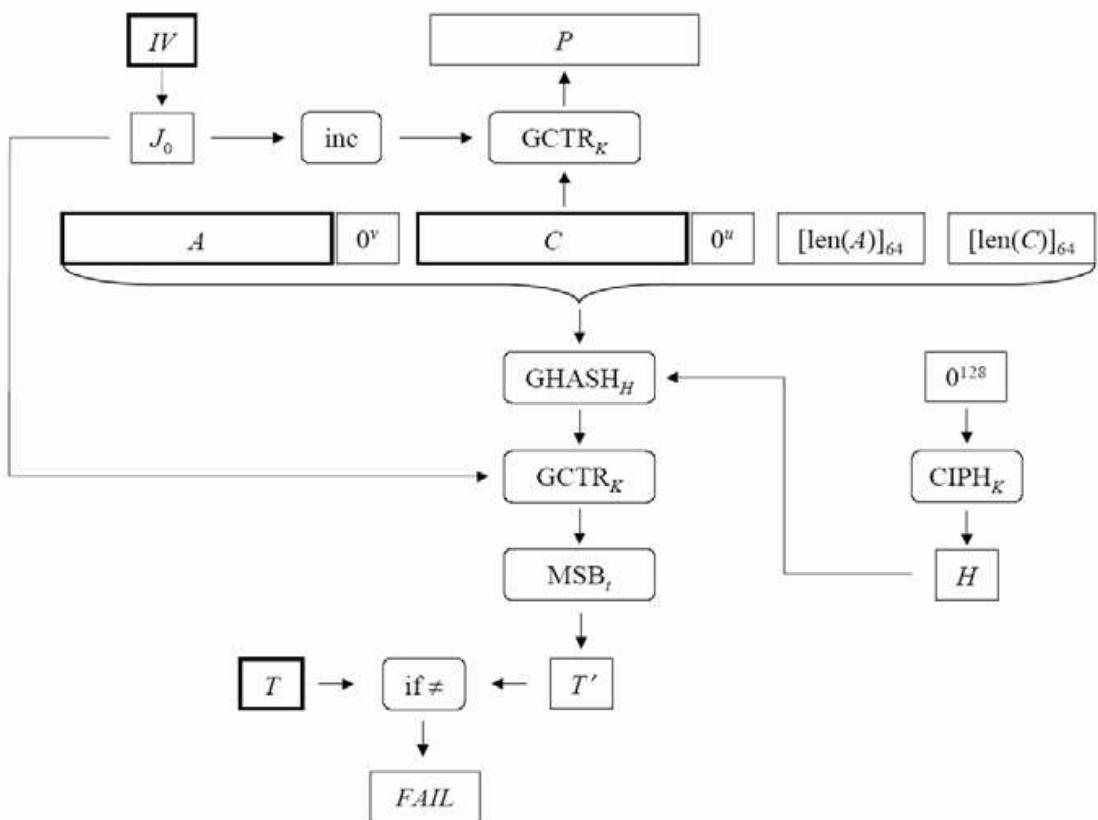
Plaintext P or indication of inauthenticity FAIL.

Steps:

1. Let  $H = \text{CIPH}_K(0^{128})$
2. Define a block,  $J_0$ , as follows:  $J_0 = IV || 0^{31} 1$ . i.e.  $J_0$  is a 128-bit string consisted of 96-bit IV, 31 ‘0’ bits, and 1 ‘1’ bit.
3. Let  $P = \text{GCTR}_K(\text{inc}(J_0), C)$ .
4. Let  $u = 128 * [\text{len}(C)/128] - \text{len}(C)$ , and let  $v = 128 * [\text{len}(A)/128] - \text{len}(A)$
5. Define a block, S, as follows:  $S = \text{GHASH}_H(A || 0^v || C || 0^u || [\text{len}(A)]_{64} || [\text{len}(C)]_{64})$
6. Let  $T' = \text{MSB}_t(GCTR_K(J_0, S))$ .
7. If  $T = T'$ , then return P; else return FAIL.

Στο step 1 το hash subkey H για την GHASH function παράγεται, εφαρμόζοντας το block cipher στο zero block. Στο step 2, το pre-counter block (J) παράγεται όπως ακριβώς είδαμε στην authenticated encryption function (step 2). Στο step 3, η 32-bit incrementing function εφαρμόζεται στο pre-counter block για να δημιουργηθεί το initial counter block (ICB) για μια κλίση της GCTR function για το ciphertext αυτή τη φορά. Το αποτέλεσμα της κλίσης της GCTR function είναι το plaintext που αντιστοιχεί στο ciphertext για τον δοσμένο IV. Στα steps 4 και 5, καθένα από τα AAD και ciphertext προσάπτονται με τον ελάχιστο αριθμό μηδενικών bits, πιθανώς και κανένα, έτσι ώστε το bit length των τερματικών strings να είναι να είναι πολλαπλάσιο του block size. Η συνένωση αυτών των strings προσάπτεται με την 64-bit αναπαράσταση του μήκους των AAD και ciphertext, και η GHASH function εφαρμόζεται στο αποτέλεσμα για να παράγει ένα single block εξόδου. Στο step 6 αυτό το block εξόδου κρυπτογραφείται χρησιμοποιώντας την GCTR function με το pre-counter block που είχε παραχθεί στο step 2 και το αποτέλεσμα περικόπτεται στο specified tag length για να σχηματίσει το authentication tag. Στο step 7 το αποτέλεσμα του step 6 συγκρίνεται με το authentication tag που έχει ληφθεί από την είσοδο και αν είναι πανομοιότυπα τότε επιστρέφεται το plaintext, αλλιώς το indication FAIL.

H authenticated decryption function εικονίζεται στην **Εικόνα 4** παρακάτω.



**Εικόνα 4. AES-GCM-AD<sub>K</sub>(IV, C, A, T) = P or FAIL.[2]**

## 4.4 Απαιτήσεις μοναδικότητας των IV και Keys

Τα IVs στον GCM πρέπει να εκπληρώνουν την ακόλουθη απαίτηση μοναδικότητας :

**Η πιθανότητα να κληθεί η authenticated encryption function με το ίδιο IV και το ίδιο key για δύο ή περισσότερα διαφορετικά set δεδομένων εισόδου, δεν πρέπει να είναι μεγαλύτερη από  $2^{-32}$ .**

Συμμόρφωση με αυτήν την απαίτηση είναι κρίσιμης σημασίας για την ασφάλεια του GCM. Σε όλες τις εμφανίσεις της πιστοποιημένης κρυπτογράφησης που λειτουργεί με ένα συγκεκριμένο κλειδί, ακόμη και αν ένας IV έχει επαναληφθεί, τότε η υλοποίηση μπορεί να είναι ευάλωτη σε επιθέσεις πλαστογραφίας που περιγράφονται στο Ref [5], και συνοψίζονται στο παράρτημα A του NIST. Στην πράξη, αυτή η απαίτηση είναι σχεδόν το ίδιο σημαντική με τη μυστικότητα του κλειδιού.

Ο ρόλος της επιλογής του κλειδιού για την υποστήριξη αυτής της απαίτησης συζητείται στην ενότητα 4.4.1. Οι δύο επιτρεπόμενες IV κατασκευές για την ικανοποίηση αυτής της απαίτησης δίνονται στην ενότητα 4.4.2. Περιορισμοί σχετικά με τον αριθμό των κλίσεων της authenticated encryption λειτουργίας δίνονται στην ενότητα 4.4.3.

### 4.4.1 Επιλογή του κλειδιού

Η ακόλουθη απαίτηση, η οποία είναι ο κανόνας για το μυστικό κλειδί κρυπτογραφικών αλγορίθμων σε γενικές γραμμές, κατέχει ρητή σημασία στον GCM για να υποστηρίξει την απαίτηση της μοναδικότητας.

**Κάθε GCM κλειδί που είναι εγκατεστημένο μεταξύ των προβλεπόμενων χρηστών θα πρέπει, με μεγάλη πιθανότητα, να είναι φρέσκο.**

Στην πράξη, οι απαιτήσεις αυτές θα πρέπει να διασφαλίσουν ότι ένα κλειδί είναι φρέσκο κατά την παραγωγή του, εάν ο μηχανισμός αναπαραγωγής του κλειδιού είναι ανθεκτικός σε παραβιάσεις. Η επίτευξη της ανθεκτικότητας αυτής συνήθως επιβάλλει απαιτήσεις σχετικά με τη διαχείριση του μηχανισμού αναπαραγωγής του κλειδιού (key generation mechanism).

Ειδικότερα, αν ο βασικός μηχανισμός αναπαραγωγής του κλειδιού είναι ντετερμινιστικός, τότε η διαχείριση του μηχανισμού παρέχει ισχυρή διαβεβαίωση ότι καμία εξωγενής οντότητα δεν μπορεί να προκαλέσει την επανάληψη προηγούμενων set εισόδων στον μηχανισμό, είτε με οποιονδήποτε άλλο τρόπο να προκαλέσει την επανάληψη μιας προηγούμενης εξόδου. Για παράδειγμα, τα GCM κλειδιά μπορούν να καθορίζονται με βάση την παραγωγή βασικών λειτουργιών από τα ακόλουθα πρωτόκολλα, όπως περιγράφεται στο [9] της βιβλιογραφίας του NIST: Transport Layer Security, Internet Key Exchange v1 και v2, και Secure Shell.

Ομοίως, αν ένα νέο κλειδί θα πρέπει να μεταφερθεί στους αποδέκτες, ο τρόπος μεταφοράς / διανομής θα πρέπει να παρέχει ισχυρή διαβεβαίωση έναντι της επανάληψης, έτσι ώστε κανένα μέρος να μην μπορεί να προκαλέσει την

αντικατάσταση προηγούμενου κλειδιού με το κλειδί που προορίζεται για την συγκεκριμένη εφαρμογή.

Τα GCM κλειδιά θα πρέπει να εγκατασταθούν στο πλαίσιο ενός εγκεκριμένου κλειδιού διοικητικής δομής για να εξασφαλίσουν τη φρεσκάδα τους, καθώς και την εμπιστευτικότητα και την αυθεντικότητα τους. Οι λεπτομέρειες των εν λόγω δομών είναι εκτός του πεδίου αυτής της παρουσίασης.

#### 4.4.2 Κατασκευές IV

Η παρούσα σύσταση παρέχει δύο πλαίσια για την κατασκευή IVs. Η πρώτη κατασκευή, που περιγράφεται στην ενότητα 4.4.2.1, βασίζεται σε ντετερμινιστικά στοιχεία για την επίτευξη της απαίτησης της μοναδικότητας. Η δεύτερη κατασκευή, που περιγράφεται στην ενότητα 4.4.2.2, βασίζεται σε μια αρκετά μεγάλη συμβολοσειρά εξόδου από εγκεκριμένη RBG (Random Bit Generator) με επαρκή αντοχή ασφαλείας.

Για οποιαδήποτε μήκος IV που υποστηρίζεται από την εφαρμογή και είναι αυστηρώς τουλάχιστον μέχρι 96 bits, η κατασκευή της επόμενης ενότητας (ντετερμινιστική κατασκευή), θα πρέπει να χρησιμοποιείται, σε όλες τις εμφανίσεις της πιστοποιημένης κρυπτογράφησης με το δεδομένο κλειδί.

Για οποιαδήποτε IV που υποστηρίζεται από την εφαρμογή και είναι μήκους 96 bit και άνω, ακριβώς μια από τις κατασκευές, αλλά όχι και οι δύο, θα πρέπει να χρησιμοποιείται, σε όλες τις εμφανίσεις της πιστοποιημένης κρυπτογράφησης με το δεδομένο κλειδί.

Για παράδειγμα, ας υποθέσουμε ότι μια εφαρμογή υποστηρίζει IV μήκους 64-bits, 96 bits, 128 bits, και 160 bit. Για 64-bit IVs η μόνη επιλογή είναι η ντετερμινιστική κατασκευή. Για τα άλλα τρία μήκη IV, ένας δυνατός συνδυασμός των επιλογών είναι η ντετερμινιστική κατασκευή για 96-bit IVs και η RBG-based κατασκευή για τους 128-bit και 160-bit IVs.

##### 4.4.2.1 ντετερμινιστική κατασκευή

Στην ντετερμινιστική κατασκευή, το IV είναι η αλληλουχία δύο πεδίων, που αναφέρονται ως σταθερό πεδίο (fixed field) και πεδίο επίκλησης (invocation field). Το fixed field προσδιορίζει τη συσκευή, ή γενικότερα, το πλαίσιο για την εμφάνιση της πιστοποιημένης κρυπτογράφησης. Το invocation field προσδιορίζει τα set των εισόδων για την λειτουργία της πιστοποιημένης κρυπτογράφησης στη συγκεκριμένη συσκευή.

Για κάθε κλειδί, δύο διαφορετικές συσκευές δεν πρέπει να μοιράζονται το ίδιο fixed field, και δύο διαφορετικά set εισόδων, για οποιαδήποτε μεμονωμένη συσκευή, δεν πρέπει να έχουν το ίδιο invocation field. Η συμμόρφωση με αυτές τις δύο απαιτήσεις συνεπάγεται τη συμμόρφωση με την απαίτηση για μοναδικότητα των IVs .

Εάν είναι επιθυμητό, το fixed field μπορεί να είναι κατασκευασμένο από δύο ή περισσότερα μικρότερα πεδία. Επιπλέον, ένα από τα μικρά αυτά πεδία θα μπορούσε να αποτελείται από κομμάτια που είναι αυθαίρετα (δηλαδή, δεν είναι κατ' ανάγκη ντετερμινιστικά ούτε μοναδικά για τη συσκευή), εφόσον τα υπόλοιπα bits εξασφαλίζουν ότι το σταθερό πεδίο δεν επαναλαμβάνεται στο σύνολό του για κάποια άλλη συσκευή με το ίδιο κλειδί.

Ομοίως, το fixed field μπορεί να συνίσταται από αυθαίρετα bits, όταν υπάρχει μόνο ένα πλαίσιο για αναγνώριση, όπως όταν ένα φρέσκο κλειδί περιορίζεται σε μία μόνο συνεδρία ενός πρωτόκολλου επικοινωνιών. Στην περίπτωση αυτή, εάν οι συμμετέχοντες μοιράζονται ένα κοινό σταθερό πεδίο, τότε το πρωτόκολλο εξασφαλίζει ότι τα πεδία επίκλησης είναι διαφορετικά για διαφορετικά δεδομένα εισόδου.

Τα πεδία επίκλησης συνήθως είναι είτε 1) ένας ακέραιος μετρητής ή 2) ένας γραμμικός shift register που οδηγείται από ένα πρωταρχικό (primitive) πολυώνυμο για να εξασφαλιστεί η μέγιστη διάρκεια του κύκλου. Και στις δύο περιπτώσεις, τα πεδία επίκλησης αυξάνονται μετά από κάθε επίκληση της λειτουργίας της πιστοποιημένης κρυπτογράφησης.

Τα μήκη και οι θέσεις του σταθερού πεδίου και του πεδίου επίκλησης πρέπει να καθορίζεται για κάθε υποστηριζόμενο μήκος IV. Για την προώθηση της διαλειτουργικότητας για το default IV μήκους 96 bits, προτείνεται, αλλά δεν απαιτείται, ότι τα πρώτα (δηλαδή, αριστερότερα) 32 bit των IV να έχουν το σταθερό πεδίο, και τα τελικά (δηλαδή, δεξιότερα) 64 bit να έχουν το πεδίο επίκλησης.

#### 4.4.2.2 RBG-based κατασκευή

Στην RBG-based κατασκευή, ο IV είναι η αλληλουχία από δύο πεδία, που αναφέρονται ως το τυχαίο πεδίο (random field) και το ελεύθερο πεδίο (free field). Για κάθε μήκος IV που υποστηρίζεται από την εφαρμογή και χρησιμοποιείται στην RBG-based κατασκευή, το μήκος των εν λόγω πεδίων πρέπει να καθορίζεται για τη ζωή του κλειδιού. Επιπλέον, το μήκος του τυχαίου πεδίου πρέπει να είναι τουλάχιστον 96 bits, και το ελεύθερο πεδίο μπορεί να είναι άδειο.

Εάν i είναι ένα υποστηριζόμενο IV μήκος που σχετίζεται με την κατασκευή RBG-based, τότε ας χαρακτηριστεί  $r(i)$  το μήκος του τυχαίου πεδίου. Το τυχαίο πεδίο είτε αποτελείται από 1) ένα string εξόδου των  $r(i)$  bits από εγκεκριμένη RBG με επαρκή δύναμη ασφάλειας, ή 2) το αποτέλεσμα της εφαρμογής της  $r(i)$ -bit προσαυξητικής λειτουργίας στο τυχαίο πεδίο του προηγούμενου IV για το δοσμένο κλειδί. Το  $r(i)$ -bit string εξόδου από την RBG ονομάζεται άμεση τυχαία σειρά (direct random string), και τα τυχαία πεδία που προκύπτουν από την εφαρμογή της  $r(i)$ -bit προσαυξητικής λειτουργίας καλούνται διάδοχοι (successors).

Δεν υπάρχουν απαιτήσεις σχετικά με τα bits στο ελεύθερο πεδίο. Για παράδειγμα, μπορούν να εντοπίζουν τη συσκευή, παρόμοια με το σταθερό πεδίο στην ντετερμινιστική κατασκευή, με εξαίρεση την RBG-based κατασκευή, αυτοί οι αναγνωριστικές ποσότητες δεν απαιτείται να είναι διαφορετικές για κάθε συσκευή.

Για κάθε μήκος IV που σχετίζεται με την RBG-κατασκευή, το ελεύθερο πεδίο συνιστάται να είναι κενό, έτσι ώστε το τυχαίο πεδίο να είναι το σύνολο του IV.

Τα στιγμιότυπα των RBGs σε δύο διαφορετικές συσκευές πρέπει να είναι ανεξάρτητα, έτσι ώστε η κατανομή των direct random string σε όλα τα RBG στιγμιότυπα να αναμένεται να είναι ομοιόμορφη. Για παράδειγμα, αν η αρχικοποίηση των RBG στιγμιότυπων εξαρτάται μόνο για ένα μυστικό "σπόρο", τότε κάθε στιγμιότυπο πρέπει να ξεκινά με ένα ξεχωριστό "σπόρο".

#### 4.4.3 Εμπόδια στον αριθμό των επικλήσεων

Η ακόλουθη απαίτηση εφαρμόζεται σε όλες τις υλοποιήσεις που χρησιμοποιούν είτε 1) την ντετερμινιστική κατασκευή των IVs το μήκος των οποίων δεν είναι 96, ή 2) την RBG-based κατασκευή, για οποιοδήποτε μήκος των IVs. Με άλλα λόγια, αν μια εφαρμογή χρησιμοποιεί μόνο 96-bit IVs που παράγονται από την ντετερμινιστική κατασκευή τότε :

**Ο συνολικός αριθμός των επικλήσεων (invocations) της λειτουργίας της πιστοποιημένης κρυπτογράφησης δεν πρέπει να υπερβαίνει τα  $2^{32}$ , συμπεριλαμβανομένων όλων των μήκων του IV και όλες τις εμφανίσεις της λειτουργίας της πιστοποιημένης κρυπτογράφησης με το δεδομένο κλειδί.**

Αυτή είναι μια "καθολική" απαίτηση που μπορεί να επιτευχθεί με τα κατάλληλα "τοπικά" όρια για κάθε εμφάνιση της λειτουργίας της πιστοποιημένης κρυπτογράφησης με δεδομένο κλειδί. Για παράδειγμα, ας υποθέσουμε ότι μια εφαρμογή αποτελείται από  $2^{10}$  συσκευές που υποστηρίζουν μόνο 64-bit, 96-bit και 128-bit IVs. Ένας τρόπος για να πληρείται η παραπάνω απαίτηση θα είναι να περιοριστεί κάθε συσκευή σε  $2^{20}$  invocations με 64-bit IVs,  $2^{21}$  invocations με 96-bit IVs, και  $2^{20}$  invocations με 128-bit IVs.

Για την RBG-based κατασκευή των IVs, η ανωτέρω απαίτηση, σε συνδυασμό με την απαίτηση ότι  $r(i) \geq 96$ , είναι επαρκής για να εξασφαλίσει την απαίτηση της μοναδικότητας.

Για την ντετερμινιστική κατασκευή, τα μήκη των δύο πεδίων εισάγουν δύο πρόσθετους λειτουργικούς περιορισμούς. Οι εν λόγω περιορισμοί ισχύουν για κάθε υποστηριζόμενο μήκος IV, συμπεριλαμβανομένου των 96 bits:

- Το μήκος bit του πεδίου επίκλησης περιορίζει τον αριθμό των επικλήσεων της λειτουργίας της πιστοποιημένης κρυπτογράφησης με δεδομένο κλειδί και δεδομένο σταθερό πεδίο. Ειδικότερα, εάν s υποδηλώνει τον αριθμό των bits του πεδίου επίκλησης, τότε η λειτουργία της πιστοποιημένης κρυπτογράφησης δεν μπορεί να κληθεί για περισσότερο από  $2^s$  ξεχωριστές σειρές εισόδων, χωρίς να παραβιάζεται η απαίτηση της μοναδικότητας.
- Ομοίως, ένα s-bit σταθερό πεδίο συνεπάγεται ένα όριο των  $2^s$  για τον αριθμό των διακριτών συσκευών / πλαισίων που μπορούν να εφαρμόσουν τη λειτουργίας της πιστοποιημένης κρυπτογράφησης με δεδομένο κλειδί και με δεδομένο μήκος IVs.

## 4.5 Υψηλής Ταχύτητας GCM Υλοποιήσεις

Οι **ASIC υλοποιήσεις του GCM** που έχουν αναφερθεί στη βιβλιογραφία έχουν αποδόσεις (throughputs) μέχρι και 42 Gbps με τη χρήση εξωτερικής διοχέτευσης γύρων AES και έναν παράλληλο Mastrovito πολλαπλασιαστή για την Galois λειτουργία [14]. Σύνθετες Sbox υλοποιήσεις είχαν χρησιμοποιηθεί για να παραχθούν σχέδια με μεγαλύτερη λειτουργικότητα επιφάνειας, ενώ για υψηλότερη απόδοση τα BDD Sboxes χρησιμοποιήθηκαν. Το πλάτος του DataPath του σχεδιασμού είναι 128 bit, ώστε σε κάθε κύκλο ρολογιού ένα μπλοκ να δίνεται στην έξοδο.

Εμπορικά σχέδια έχουν αναφέρει παρόμοια throughputs για standalone GCM υλοποιήσεις. Μία παράλληλη αρχιτεκτονική η οποία υπολογίζει τέσσερις λειτουργίες GCM ταυτόχρονα προτάθηκε από τον Satoh και είναι η υψηλότερη απόδοση που έχει αναφερθεί μέχρι σήμερα, ικανή για έως και 160 Gbps [12]. Αυτός ο σχεδιασμός, ωστόσο, έχει έναν αυξημένο αριθμό εισόδων που απαιτούνται για κάθε παράλληλη GCM λειτουργία.

Μια αρχιτεκτονική διοχέτευσης πολλαπλασιασμού προτάθηκε στο [11] με επίτευξη 54,9 Gbps. Σε αυτό το σχεδιασμό γίνεται χρήση μιας εσωτερικής και μιας εξωτερικής διοχέτευσης του AES μπλοκ με έναν pipelined πολλαπλασιαστή μικρής καθυστέρησης ο οποίος κατασκευάστηκε από την παράλληλη αρχιτεκτονική του GCM. Με τη χρήση αυτού του τύπου κατασκευής, η ανάδραση, που είναι προϋπόθεση για τον πολλαπλασιαστή, διατηρήθηκε, και μια υψηλή απόδοση επιτεύχθηκε λόγω των αυξημένων σταδίων της διοχέτευσης. Η υψηλότερη καθυστέρηση αλλαγής του κλειδιού που κυμαίνεται από 40 έως 56 κύκλους, μειώνει τη συνολική αποδοτικότητα αν υπάρχουν περισσότερες απότομες ενάρξεις που προκύπτουν ως αποτέλεσμα της ανανέωσης του κλειδιού.

Διάφορα **FPGA σχέδια** έχουν επίσης προταθεί για το GCM στο [13] με τη χρήση ενός παράλληλου πολλαπλασιαστή μετατοπισμένης πολυωνυμικής βάσης, στα οποία έχει επιτευχθεί απόδοση μέχρι και 15,3 Gbps. Το πλάτος του DataPath προτείνεται να κυμαίνεται από 8 έως 128 bits, όπου προκύπτουν σχέδια με διάφορα tradeoffs στις καθυστερήσεις και την επιφάνεια. Τρεις Sbox λύσεις, ο LUT, τα σύνθετα Sbox και μια Block Ram υλοποίηση χρησιμοποιήθηκαν στην εφαρμογή του GCM.

Ένας άλλος FPGA σχεδιασμός με σημαντικά αποτελέσματα παρουσιάστηκε στο [14], το οποίο παρείχε μία Karatsuba Algorithm υλοποίηση του πολλαπλασιαστή (KA). Αυτός ο σχεδιασμός κάνει χρήση των σύνθετων Sbox με τον KA πολλαπλασιαστή για την επίτευξη ενός ποσοστού απόδοσης των 15,23 Gbps με τη χρήση ενός γύρου εξωτερικής διοχέτευσης. Χρησιμοποιώντας γύρο εσωτερικής διοχέτευσης και έναν brute force πολλαπλασιαστή, μια 20,61 Gbps απόδοση επιτεύχθηκε, αλλά είχε τη διπλάσια καθυστέρηση. Η εφαρμογή αυτή, επίσης, υποστηρίζει μόνο 128 bit AES κλειδιών σε αντίθεση με την εφαρμογή που προβλέπεται στο [13], η οποία υποστηρίζει όλα τα είδη κλειδιών.



# Κεφάλαιο 5

## Υλοποίηση και Αρχιτεκτονική

Το κεφάλαιο αυτό περιγράφει την AES-GCM υλοποίηση και τις τεχνικές της αρχιτεκτονικής. Η ενότητα 5.1 επεξηγεί, μέσω σχηματικών αναπαραστάσεων, την προτεινόμενη υλοποίηση αναλύοντας με ακρίβεια τις εσωτερικές διεργασίες και μηχανισμούς που συνθέτουν το συγκεκριμένο πρότυπο ασφαλείας. Στα επόμενα μέρη αυτού του κεφαλαίου, ενότητα 5.2 και έπειτα, εμβαθύνουμε σε θέματα αρχιτεκτονικής από την οποία προκύπτει η προτεινόμενη HDL υλοποίηση του AES-GCM που προορίζεται για την επιλεγμένη FPGA πλατφόρμα. Η ενότητα 5.3 πραγματεύεται τις αρχιτεκτονικές των βασικών δομών AES και GHASH. Η ενότητα 5.4 ασχολείται με την αρχιτεκτονική ολόκληρου του AES-GCM συστήματος.

### 5.1 Διεργασίες και Μηχανισμοί του GCM

Στα προηγούμενα κεφάλαια έγινε μια πλήρης μαθηματική και θεωρητική ανάλυση της λειτουργίας του AES-GCM αλγορίθμου, η οποία στηρίχθηκε στα αντίστοιχα recommendation του NIST. Στο κεφάλαιο αυτό θα γίνει μια πλήρης και λεπτομερής ανάλυση του AES-GCM η οποία βασίζεται στην τμηματοποίηση, τον διαχωρισμό και την ομαδοποίηση των βασικών δομών (modules) του αλγορίθμου ώστε να γίνει σαφές το πώς αυτές οι δομές αλληλεπιδρούν και συνεργάζονται μεταξύ τους για την επίτευξη του επιθυμητού αποτελέσματος. Τέλος, η ανάλυση που γίνεται σε αυτό το πρώτο μέρος του κεφαλαίου δεν είναι αυστηρά hardware προσέγγιση αλλά μπορεί να χρησιμοποιηθεί και για υλοποίηση σε software, αν θεωρήσουμε τα διάφορα στοιχεία ως συναρτήσεις ή υποσυναρτήσεις.

#### 5.1.1 Διεργασία Κρυπτογράφησης (Encryption Function)

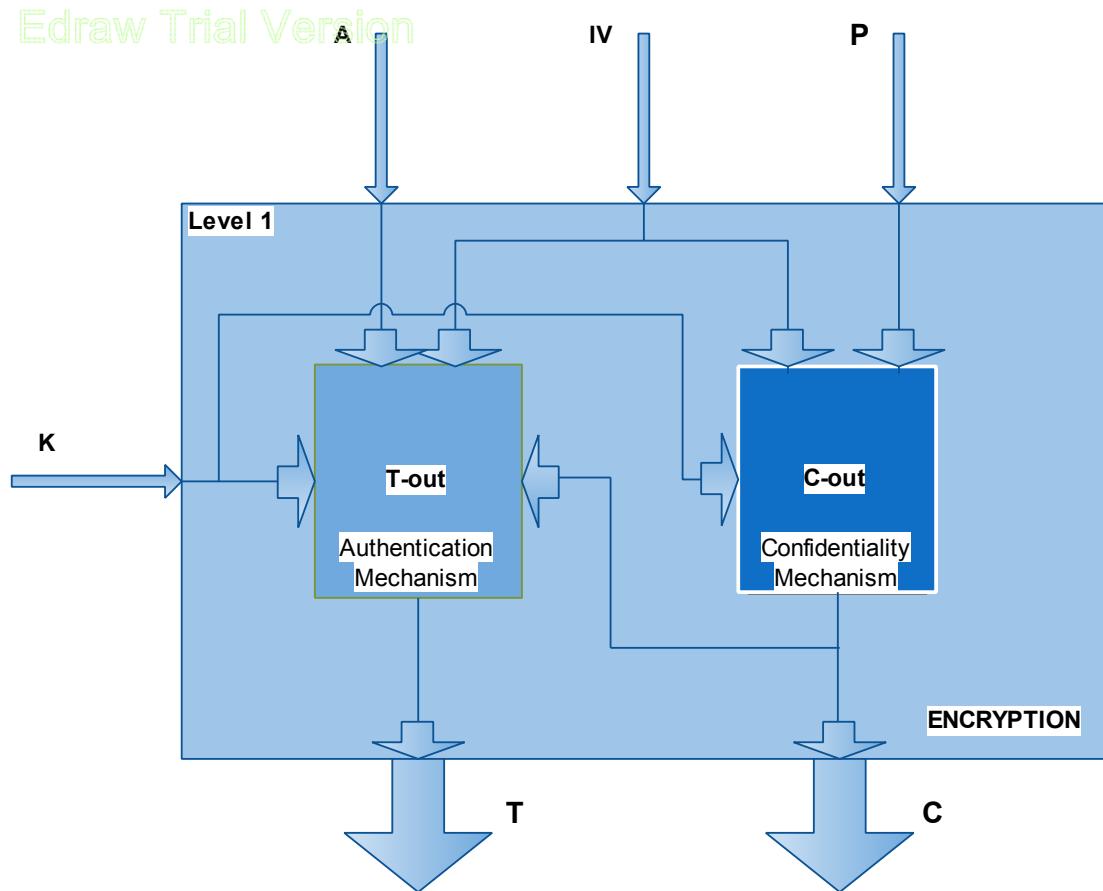
Παρακάτω φαίνονται σχηματικά τα επίπεδα και οι δομές που συνθέτουν τον αλγόριθμο κρυπτογράφησης έτσι όπως αυτός παρουσιάστηκε στο κεφάλαιο 4. Αυτό το βήμα είναι απαραίτητο πριν προχωρήσουμε σε περαιτέρω ανάλυση αρχιτεκτονικής, ώστε να γίνει κατανοητή η εσωτερική κατανομή των στοιχειώδων δομών των δύο βασικών μηχανισμών που συνθέτουν τον AES-GCM. Η προσέγγιση που ακολουθείται είναι της λογικής top-down. Ουσιαστικά, μια top-down προσέγγιση είναι ο κατακερματισμός ενός συστήματος για να αποκτηθούν γνώσεις σχετικά με την σύνθεση των επιμέρους συστημάτων. Σε μία top-down προσέγγιση διατυπώνεται μια πρώτη και γενική εικόνα του συστήματος αλλά δεν διευκρινίζεται λεπτομερώς κάθε πρωτοβάθμιο επίπεδο υποσυστημάτων. Κάθε υποσύστημα μπορεί ακόμα να τελειοποιηθεί με μεγαλύτερη λεπτομέρεια, και μερικές φορές σε πολλά επιπρόσθετα επίπεδα, μέχρι το σύνολο της προδιαγραφής να μειωθεί στα βασικά του στοιχεία.

Ένα top-down μοντέλο εκφράζεται με την βοήθεια των “black boxes” που διευκολύνει τον χειρισμό του συστήματος. Με τον όρο “black boxes” αναφερόμαστε στα στοιχεία του κυκλωματικού για τα οποία μας είναι γνωστό ποιά διεργασία υλοποιούν αλλά δεν μας είναι γνωστή η εσωτερική αρχιτεκτονική τους.

Σε αυτή την ενότητα θα γίνει μια top-down ανάλυση σε τέτοιο βαθμό που να είναι εφικτό μεταγενέστερα να εφαρμόσουμε οποιαδήποτε τεχνική υλοποίησης και χρονισμού για την βελτίωση του συστήματος.

Εδώ, ως εξωτερικό “black box” ορίζεται ο αλγόριθμος AES-GCM στο σύνολό του με εισόδους  $K$ ,  $A$ ,  $IV$  και  $P$  δηλαδή Key, Additional Authenticated Data, Initialization Vector και Plaintext αντίστοιχα, και εξόδους  $T$ ,  $C$  δηλαδή TAG και Ciphertext αντίστοιχα.

Ως εσωτερικά “black boxes” ορίζονται οι δύο μηχανισμοί που επιτελούνται κατά την εκτέλεση του αλγορίθμου όπως δείχνονται παρακάτω:



Σε μια σύντομη περιγραφή έχουμε ότι :

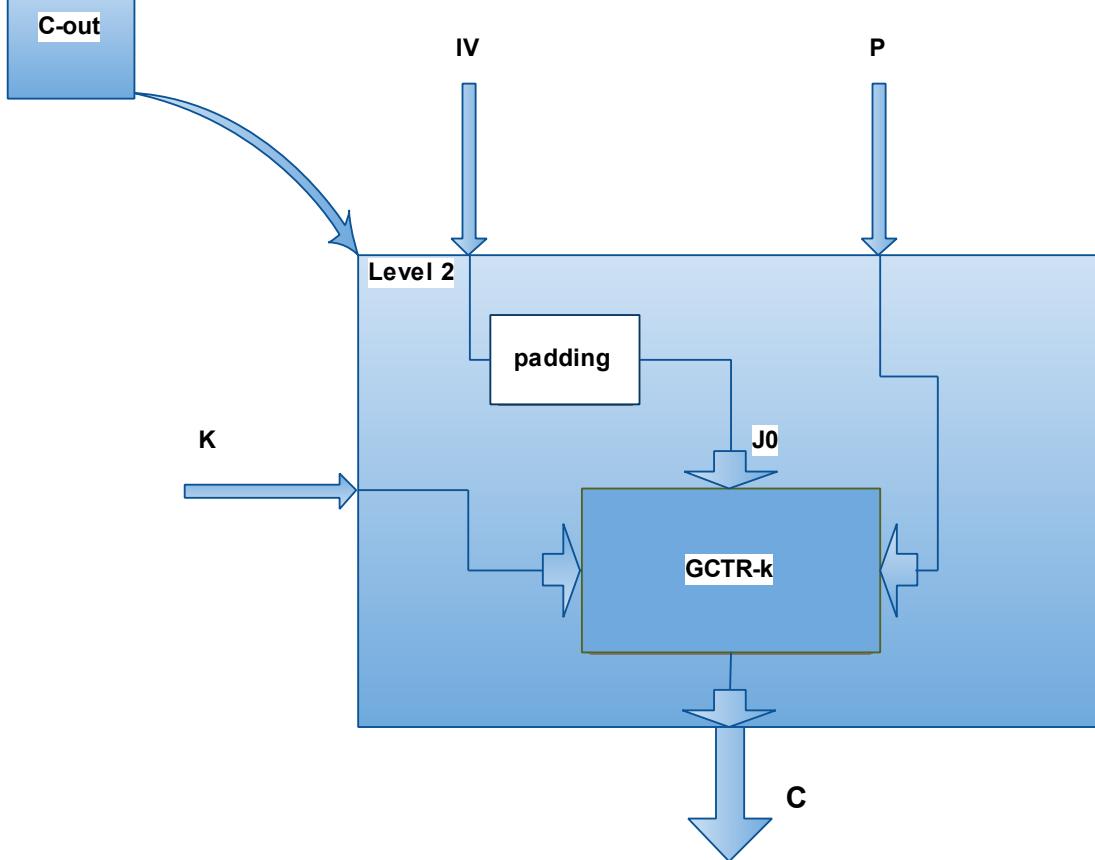
- Ο μηχανισμός κρυπτογράφησης τροφοδοτείται από τις βασικές εξωτερικές εισόδους K, IV και Plaintext και δίνει στην έξοδο τα κρυπτογραφημένα πλέον δεδομένα, το Ciphertext. Τα A αποτελούν τα non-confidential δεδομένα και επομένως δεν θα περάσουν από τον κρυπτογραφικό μηχανισμό.
- Ο μηχανισμός παραγωγής του μηνύματος αυθεντικότητας τροφοδοτείται από τα K, IV, A όμως δεν έχει ως είσοδο το Plaintext αλλά τα κρυπτογραφημένα δεδομένα Ciphertext που προέρχονται από τον C-out μηχανισμό.

Η δεδομένη σχηματική ανάλυση αντιπροσωπεύει την software υλοποίηση κατά κύριο λόγο. Στην HDL αρχιτεκτονική, όπως θα δούμε παρακάτω, ο T-out μηχανισμός δεν χρειάζεται να έχει στην είσοδό του τον IV καθώς το block Ciphk\_j0 που απαιτείται για την λειτουργία του και προκύπτει από την κρυπτογράφηση του initial counter block ICB, προέρχεται εσωτερικά από το GCTR module και δίνεται έτοιμο στον T-out μηχανισμό.

Στην συνέχεια ακολουθεί η ανάλυση των δύο βασικών, εσωτερικών μηχανισμών της διαδικασίας της κρυπτογράφησης.

#### **5.1.1.1 Μηχανισμός Εμπιστευτικότητας (Confidentiality Mechanism)**

Η ανάλυση σε κατώτερα επίπεδα ξεκινάει από τον μηχανισμό διασφάλισης της εμπιστευτικότητας. Η ανάλυση επιβάλλεται να ξεκινήσει από αυτόν τον μηχανισμό, μιας και όπως φάνηκε από το παραπάνω σχηματικό, η έξοδος του τροφοδοτείται στην είσοδο του μηχανισμού πιστοποίησης. Επιπλέον, κάποια στοιχεία που ανήκουν στα κατώτερα επίπεδα αυτού χρησιμοποιούνται και στον μηχανισμό πιστοποίησης.

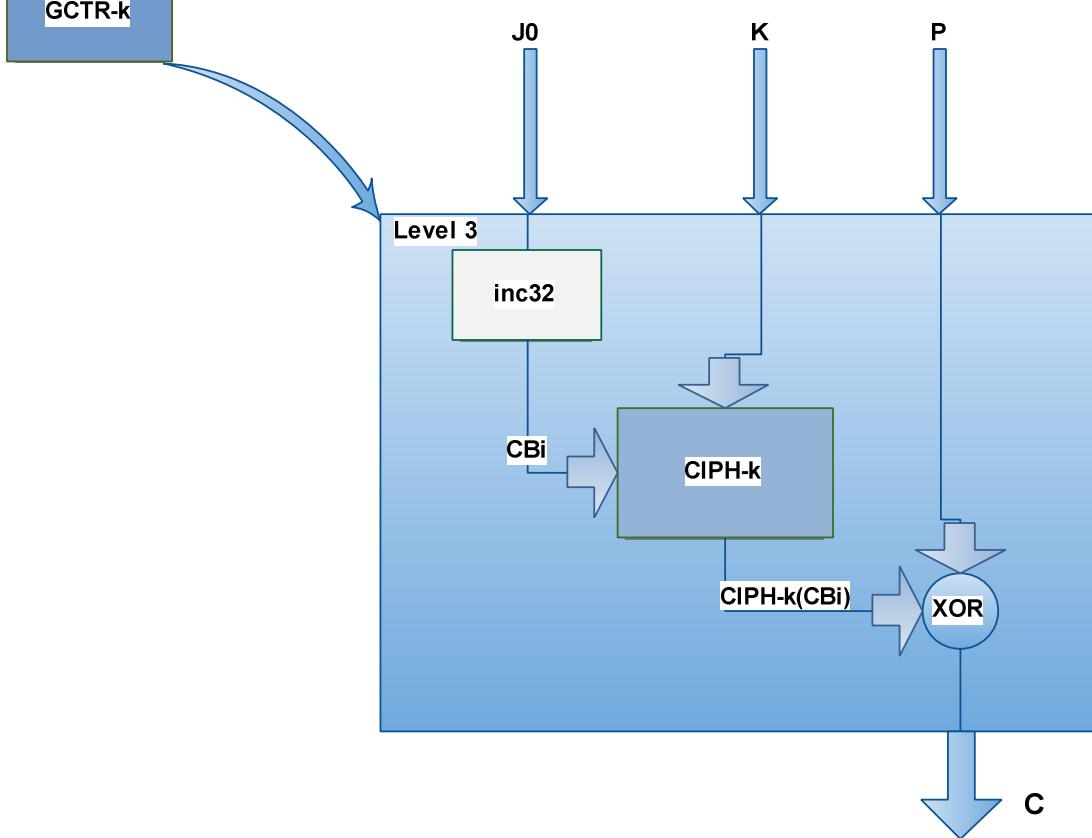


Η δεδομένη σχηματική ανάλυση αντιπροσωπεύει την software υλοποίηση κατά κύριο λόγο. Στην HDL αρχιτεκτονική, όπως θα δούμε παρακάτω, το padding είναι ενσωματωμένο στην διαδικασία παραγωγής των counter blocks, εσωτερικά του GCTR module.

Σε μια σύντομη περιγραφή έχουμε ότι :

- Ο IV προκαθορισμένου μήκους 96 bits συμπληρώνεται με μηδενικά (padding) ώστε όλες οι είσοδοι του GCTR module να είναι 128 bits.

Προχωράμε ένα επίπεδο παρακάτω και αναλύουμε την  $GCTR_k$  αφού το padding είναι απλώς μια συνένωση των 96 bits του IV με ένα μηδενικό string των 32 bits. Το GCTR module έχει αναλυθεί λεπτομερώς στο προηγούμενο κεφάλαιο (βλέπε ενότητα 4.2.2).

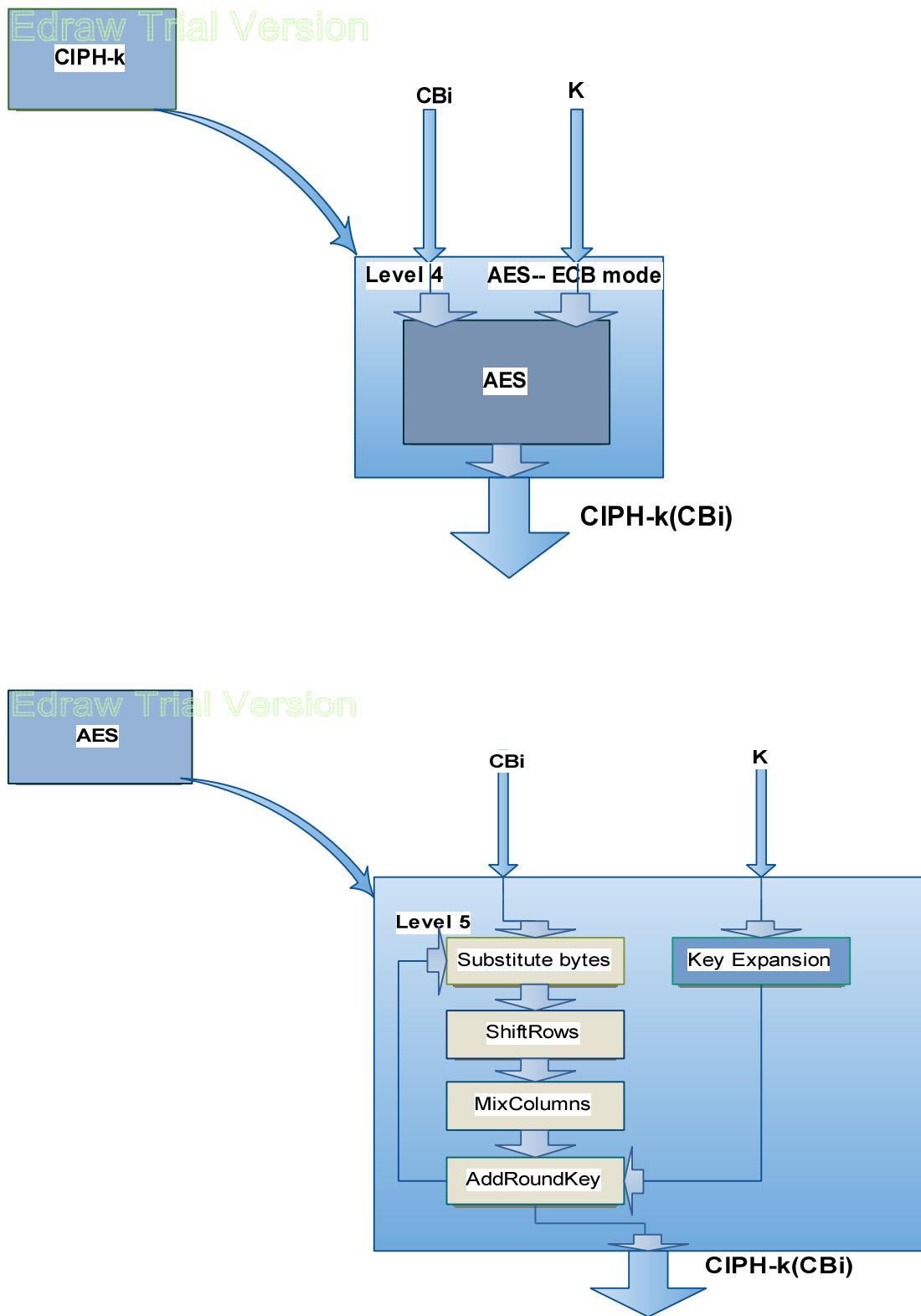


Η δεδομένη σχηματική ανάλυση αντιπροσωπεύει την software αλλά και την HDL αρχιτεκτονική, με την μόνη διαφορά ότι στην HDL αρχιτεκτονική η ncr32 που παράγει τα counter blocks τροφοδοτείται απευθείας με τον IV, με το padding ενσωματωμένο στο εσωτερικό της, ενώ μια 11\*128 bits FIFO θα αποθηκεύει τα plaintexts μέχρι να γεμίσει το pipeline του AES.

Σε μια σύντομη περιγραφή έχουμε ότι:

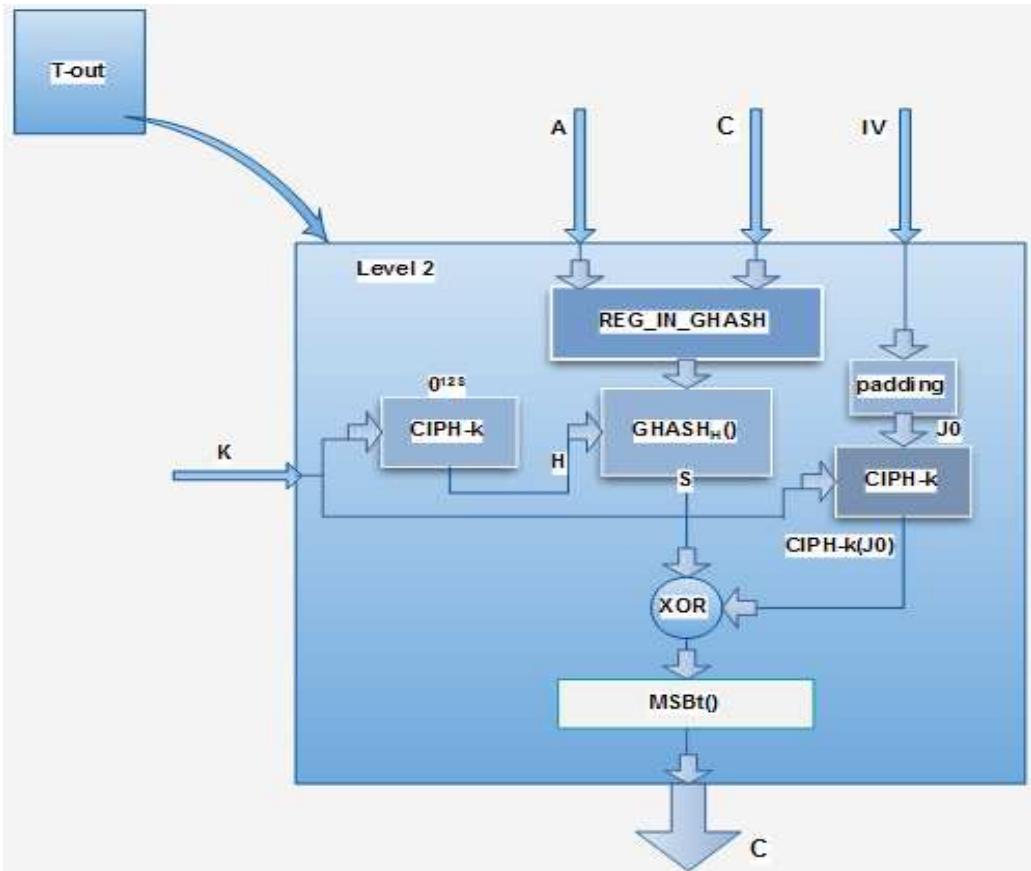
- Το GCTR module αντιπροσωπεύει ουσιαστικά τον CTR τρόπο λειτουργίας του οποίου ο πυρήνας είναι ο ECB τρόπος λειτουργίας καθώς δεν είναι τα Plaintext blocks αυτά που κρυπτογραφούνται αλλά τα counter blocks που προέρχονται από την αύξηση του IV. Τα Ciphertext blocks προκύπτουν μέσω XOR πράξης των κρυπτογραφημένων counter blocks με τα Plaintext blocks. Αυτός είναι άλλωστε και ο λόγος που το ίδιο σύστημα χρησιμοποιείται για κρυπτογράφηση και αποκρυπτογράφηση μιας και η XOR είναι από μόνη της μια αντίστροφη διαδικασία.

Τα Level 4, Level 5 έχουν μελετηθεί πλήρως στο κεφάλαιο 3. Ουσιαστικά το Level 4 μπορεί να ενσωματωθεί με το Level 5 αφού επιτελούν την ίδια λειτουργία. Απλώς για λόγους συνέπειας δείχνεται σχηματικά ότι ο πυρήνας του GCTR module, δηλαδή ο AES σε ECB mode. Τέλος, το Level 5 εκτελείται επαναληπτικά μέχρι να προκύψει το αποτέλεσμα, είναι δηλαδή το βασικό component για το pipeline του AES.



### 5.1.1.2 Μηχανισμός Πιστοποίησης (Authentication Mechanism)

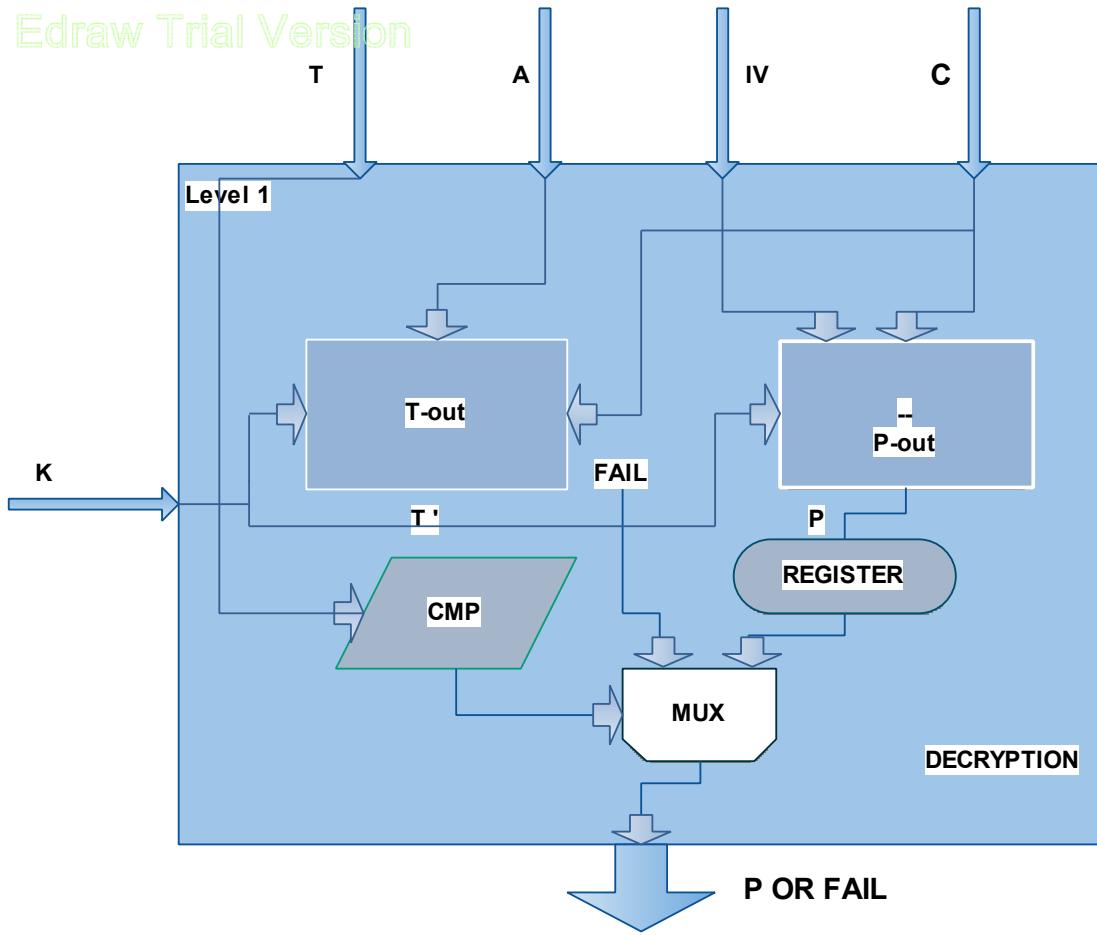
Στο παρακάτω σχηματικό φαίνεται μια πρώτη υλοποίηση του μηχανισμού πιστοποίησης από τον οποίο παράγεται το μήνυμα αυθεντικότητας που υπογράφει μοναδικά το plaintext και τα AAD.



Η ανάλυση αυτού του μηχανισμού δεν αποτελεί μέρος αυτής της διπλωματικής εργασίας. Όμως, για την τελική υλοποίηση του AES-GCM προτύπου ασφαλείας τα δύο βασικά modules θα συνεργαστούν, οπού και μαζί με ένα control unit, θα υλοποιούν τον AES-GCM στο σύνολό του.

Όπως αναφέραμε και νωρίτερα, στην τελική HDL αρχιτεκτονική ο T-out μηχανισμός δεν χρειάζεται να έχει στην είσοδό του τον IV καθώς το Ciphk\_j0 που απαιτείται για την λειτουργία του, και δείχνεται ως J<sub>0</sub>, προέρχεται εσωτερικά από το GCTR module και δίνεται έτοιμο στον T-out μηχανισμό.

### 5.1.2 Διεργασία Αποκρυπτογράφησης (Decryption Function)



Σε μια σύντομη περιγραφή έχουμε ότι:

- Στην διεργασία αποκρυπτογράφησης δίνεται μόνο το πρώτο επίπεδο αφού όλα τα επόμενα επίπεδα είναι ίδια με αυτά της διεργασίας κρυπτογράφησης. Εδώ, το  $P\text{-out}$  είναι ακριβώς το ίδιο με το  $C\text{-out}$  της κρυπτογράφησης με την μόνη διαφορά στις εισόδους – εξόδους. Εδώ, έχει είσοδο το  $C$  και έξοδο  $P$  αλλά εσωτερικά ο μηχανισμός είναι ακριβώς ο ίδιος και υλοποιείται ακριβώς με τον ίδιο τρόπο.
- Μια επιπλέον είσοδος υπάρχει στην διεργασία της αποκρυπτογράφησης, αυτή του TAG. Μετά το τέλος των δύο μηχανισμών η μονάδα CMP συγκρίνει το TAG της εισόδου με αυτό που προέκυψε από το σύστημα. Αν τα δύο TAGs είναι ίδια τότε σαν έξοδο παίρνουμε το Plaintext διαφορετικά ένα μήνυμα FAIL εμφανίζεται στην έξοδο. Όλα αυτά έχουν αναλυθεί στο προηγούμενο κεφάλαιο.

## 5.2 Θέματα Αρχιτεκτονικής στον GCM

Η υλοποίηση του GCM είναι μια straightforward διαδικασία τόσο σε hardware όσο και σε software, δοσμένης μιας υλοποίησης του βασικού block cipher αλγορίθμου και της πράξης του πολλαπλασιασμού στο  $GF(2^{128})$ . Σε αυτήν την ενότητα, παρουσιάζεται συνοπτικά μια λειτουργική hardware υλοποίηση.

Ο αριθμός των block cipher κλίσεων που χρειάζονται να γίνουν ώστε να κρυπτογραφηθεί ένα plaintext p-bits με τον AES-GCM είναι ίσος με  $[p/128] + 1$ . Ο ίδιος αριθμός πολλαπλασιασμών στο  $GF(2^{128})$  απαιτούνται επίσης. Η μία επιπλέον block cipher κλίση απαιτείται για να υπολογιστεί το hash subkey  $H$  εάν δεν είναι αποθηκευμένο. Εάν υπάρχουν επιπρόσθετα q bits AAD δεδομένων που προορίζονται για πιστοποίηση (authentication), τότε χρειάζονται ακόμα άλλες  $[q/128]$  πράξεις πολλαπλασιασμού. Η αποκρυπτογράφηση είναι μια διαδικασία παρόμοια με την κρυπτογράφηση και συμμερίζεται τα χαρακτηριστικά απόδοσής της.

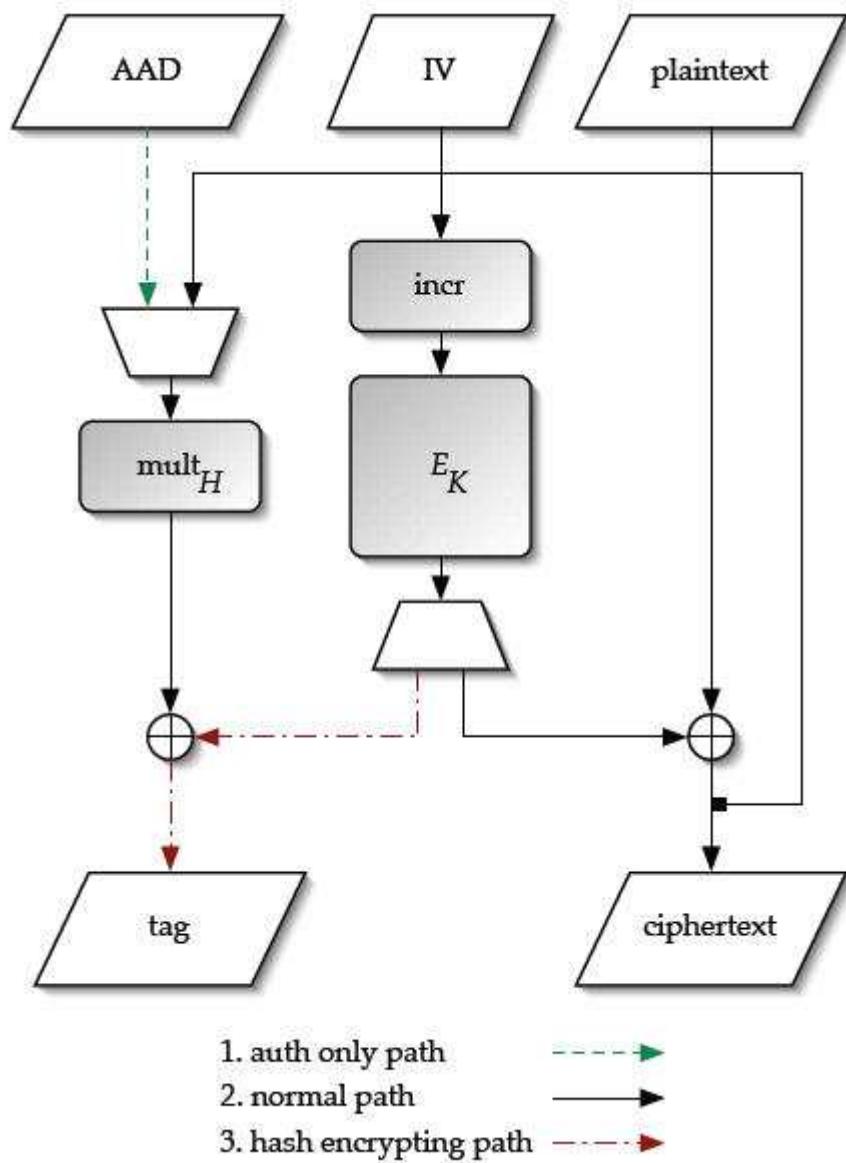
Με βάση όσα έχουν αναφερθεί παραπάνω διακρίνονται δύο διεργασίες. Η διεργασία της κρυπτογράφησης (encryption function) και η διεργασία της αποκρυπτογράφησης (decryption function). Σε υλικό, οι δύο αυτές διεργασίες πρέπει να υπάρχουν μαζί και είτε να ενεργοποιείται το σύστημα για κρυπτογράφηση είτε για αποκρυπτογράφηση. Η διαδικασία αποκρυπτογράφησης έχει όλα τα στοιχεία της διαδικασίας κρυπτογράφησης και κάποια επιπλέον που μπορεί να χαρακτηριστούν ως στοιχεία ελέγχου. Επομένως, ο σωστός σχεδιασμός της διεργασίας κρυπτογράφησης είναι η αναγκαία συνθήκη για τον σωστό σχεδιασμό της διεργασίας αποκρυπτογράφησης. Είναι λογικό, λοιπόν, κατά την σχεδίαση να δίνεται επιπλέον βάρος στην διαδικασία υλοποίησης της διεργασίας κρυπτογράφησης χωρίς αυτό να σημαίνει ότι ο σχεδιασμός της διεργασίας αποκρυπτογράφησης δεν είναι σημαντική υπόθεση.

### 5.2.1 Μια hardware υλοποίηση του GCM

Σε αυτήν την ενότητα σκιαγραφείται ένας pipelined hardware σχεδιασμός, ο οποίος φαίνεται στην **Εικόνα 1**. Οι δύο ρόμβοι αποτελούν πολυπλέκτες και δείχνουν τα σημεία όπου αλλάζουν τα data path. Υπάρχουν τρείς είσοδοι: δεδομένα AAD που προορίζονται για πιστοποίηση μόνο (authenticated-only), το IV και το plaintext. Το IV τροφοδοτείται στην increment function, για να παραχθούν στη συνέχεια επιτυχείς τιμές από counters οι οποίοι τροφοδοτούνται στο block cipher pipeline, που δείχνεται ως EK στην **Εικόνα 1**. Το πρώτο κρυπτογραφημένο counter block (Ciph-k(Jo)), το οποίο χαρακτηρίζεται ως initial counter block ICB, στέλνεται για να κρυπτογραφήσει την έξοδο της GHASH (path 3), και έπειτα η έξοδος του πολυπλέκτη αλλάζεται έτσι ώστε τα επόμενα κρυπτογραφημένα counter blocks, τα οποία δείχνονται ως CBIs, να γίνονται XORed με το plaintext για να σχηματίσουν το ciphertext (path 2) (AES ECB mode).

Τα authenticated-only δεδομένα τροφοδοτούνται στην GHASH function (path 1), και στη συνέχεια η είσοδος αυτής της function αλλάζεται μέσω του πολυπλέκτη ώστε να δέχεται το ciphertext (path 2). Όταν όλα τα δεδομένα εισόδου της GHASH έχουν επεξεργαστεί, η έξοδος της γίνεται XORed με τον πρώτο κρυπτογραφημένο counter block και έτσι παράγεται το authentication tag.

Στο συγκεκριμένο σχεδιασμό, το tag-generating pipeline και τα ciphertext-generating pipelines είναι ανεξάρτητα, εκτός από το βήμα που έχουμε το tag-encryption. Αυτά τα δύο pipelines μπορούν να γίνουν τελείως ανεξάρτητα προσθέτοντας έναν ακόμα AES μηχανισμό αφιερωμένο στην κρυπτογράφηση της εξόδου της GHASH.



**Εικόνα 1 : A hardware implementation of GCM, showing the different data paths through the circuit.**

Όπως έχουμε αναλύσει, η κρυπτογράφηση αποτελείται από δύο μηχανισμούς, τον μηχανισμό πιστοποίησης (path1-path3) και τον μηχανισμό εμπιστευτικότητας (path2). Αυτοί οι δύο μηχανισμοί μπορούν να εκτελούνται υπό ορισμένες συνθήκες ανεξάρτητα και παράλληλα. Τα δεδομένα όμως του ενός μηχανισμού (Cout / path2) χρησιμοποιούνται στον άλλον μηχανισμό (Tout) μέσω του πολυπλέκτη, το αντίθετο όμως δεν συμβαίνει. Ωστόσο, για να θεωρείται η κρυπτογράφηση του AES-GCM προτύπου ασφαλείας ολοκληρωμένη πρέπει να παράγεται στην έξοδο τόσο το Ciphertext όσο και το Tag καθώς, η δύναμη του GCM έγκειται στο να παρέχει αποτελεσματικά εμπιστευτικότητα αλλά και πιστοποίηση των δεδομένων σε υψηλές ταχύτητες.

Διακρίνουμε ότι κάποιες δομές του Cout μηχανισμού χρησιμοποιούνται στο Tout μηχανισμό. Για το λόγο αυτό υπάρχει και ο διαχωρισμός μεταξύ των data paths και ουσιαστικά η χρήση των δομών αυτών διακρίνεται καθαρά από το path3. Η κοινή δομή στους δύο μηχανισμούς είναι ο block Cipher ( $E_k$ ). Πιο συγκεκριμένα, ο block Cipher ( $E_k$ ) χρησιμοποιείται δύο φορές για να παράγει δύο διαφορετικές τιμές που χρειάζονται για την λειτουργία παραγωγής του Tag. Την πρώτη φορά για να παράγει το hash subkey H, και τη δεύτερη φορά για να παράγει το κρυπτογραφημένο initial counter block. Εφόσον ο block Cipher ( $E_k$ ) χρησιμοποιείται μόνο δύο φορές στον Tout μηχανισμό, θα ήταν άσκοπο σε υλικό να υπάρχει ξεχωριστό κύκλωμα που να εκτελεί αυτές τις διεργασίες. Γενικά, κατά την σχεδίαση υλικού δεν πρέπει να υπάρχουν κυκλώματα που καταλαμβάνουν επιφάνεια αν αυτά δεν χρησιμοποιούνται λειτουργικά.

Άρα, η τεχνική που θα ακολουθήσουμε είναι να παράγουμε τις τιμές που απαιτούνται χρησιμοποιώντας τα ήδη υπάρχοντα κυκλώματα και επιλέγοντας να τα χρησιμοποιήσουμε όταν το σύστημα δεν θα είναι επιβαρυμένο με κάποια άλλη διεργασία. Θα έχουμε λοιπόν δύο registers που θα αποθηκεύουν τις τιμές που χρειαζόμαστε από τον block Cipher και θα τις διαθέτουν στο σύστημα Tout όταν αυτές ζητηθούν.

Το hash subkey H είναι προαπαιτούμενο για την έναρξη της λειτουργίας του Tout. Έτσι, η τιμή αυτή πρέπει να παράγεται αρχικά, πριν ξεκινήσει η διαδικασία παραγωγής του Ciphertext ώστε οι δύο μηχανισμοί να δουλεύουν παράλληλα. Βέβαια, μια διαφορετική προσέγγιση είναι το hash subkey H να παράγεται από εξωτερικό κύκλωμα block Cipher που θα συνίσταται από μία επαναληπτική AES δομή όπου η μόνη της λειτουργία θα είναι η παραγωγή του H. Η τελευταία αυτή πρόταση είναι και αυτή που εφαρμόστηκε στην εργασία αυτή.

Το κρυπτογραφημένο initial counter block, Ciph-k(J0), απαιτείται μόνο στο τελικό στάδιο του Tout οπότε μπορεί να παράγεται αφού έχει τελειώσει η διαδικασία παραγωγής του Ciphertext, ή διαφορετικά πριν την έναρξη της διαδικασίας παραγωγής του Ciphertext καθώς η GHASH θα επεξεργάζεται ακόμα τα AAD δεδομένα. Η τελευταία αυτή πρόταση είναι και αυτή που εφαρμόστηκε στην εργασία αυτή.

### 5.2.2 Διάγραμμα Ροής Μηχανισμών

Σύμφωνα με την hardware υλοποίηση που αναλύθηκε προηγούμενα, το διάγραμμα ροής που παρουσιάζεται στο παρακάτω σχηματικό αναπαριστά τον τρόπο με τον οποίο γίνονται οι διεργασίες στο πεδίο του χρόνου. Παρουσιάζονται δύο διαδρομές. Η αριστερή διαδρομή είναι υπεύθυνη για την παραγωγή του Tag, ενώ η δεξιά διαδρομή είναι υπεύθυνη για την παραγωγή του ciphertext.

Αν υποθέσουμε ότι έχουμε όλο το πακέτο με τα δεδομένα (plaintext και AAD) τότε ακολουθείται η εξής διαδικασία :

→ Αρχικά χρησιμοποιείται ο εξωτερικός block Cipher για την παραγωγή του hash subkey (H) ο οποίος αποθηκεύεται σε register και δίνεται στον Tout μηχανισμό κατά την έναρξη της λειτουργίας του.

→ Στην συνέχεια οι δρόμοι χωρίζονται και το μεν Tout (αριστερή διαδρομή) επεξεργάζεται τα AAD δεδομένα ενώ το Cout (δεξιά διαδρομή) χρησιμοποιείται για την παραγωγή του Ciph-k(J0) και του ciphertext.

→ Σε κάθε κύκλο διεργασιών επεξεργάζονται και από τους δύο μηχανισμούς 128 bits.

→ Όταν το Cout έχει ολοκληρώσει την παραγωγή του Ciph-k(J0) τότε αυτό αποθηκεύεται σε register και δίνεται στον Tout μηχανισμό όταν του ζητηθεί.

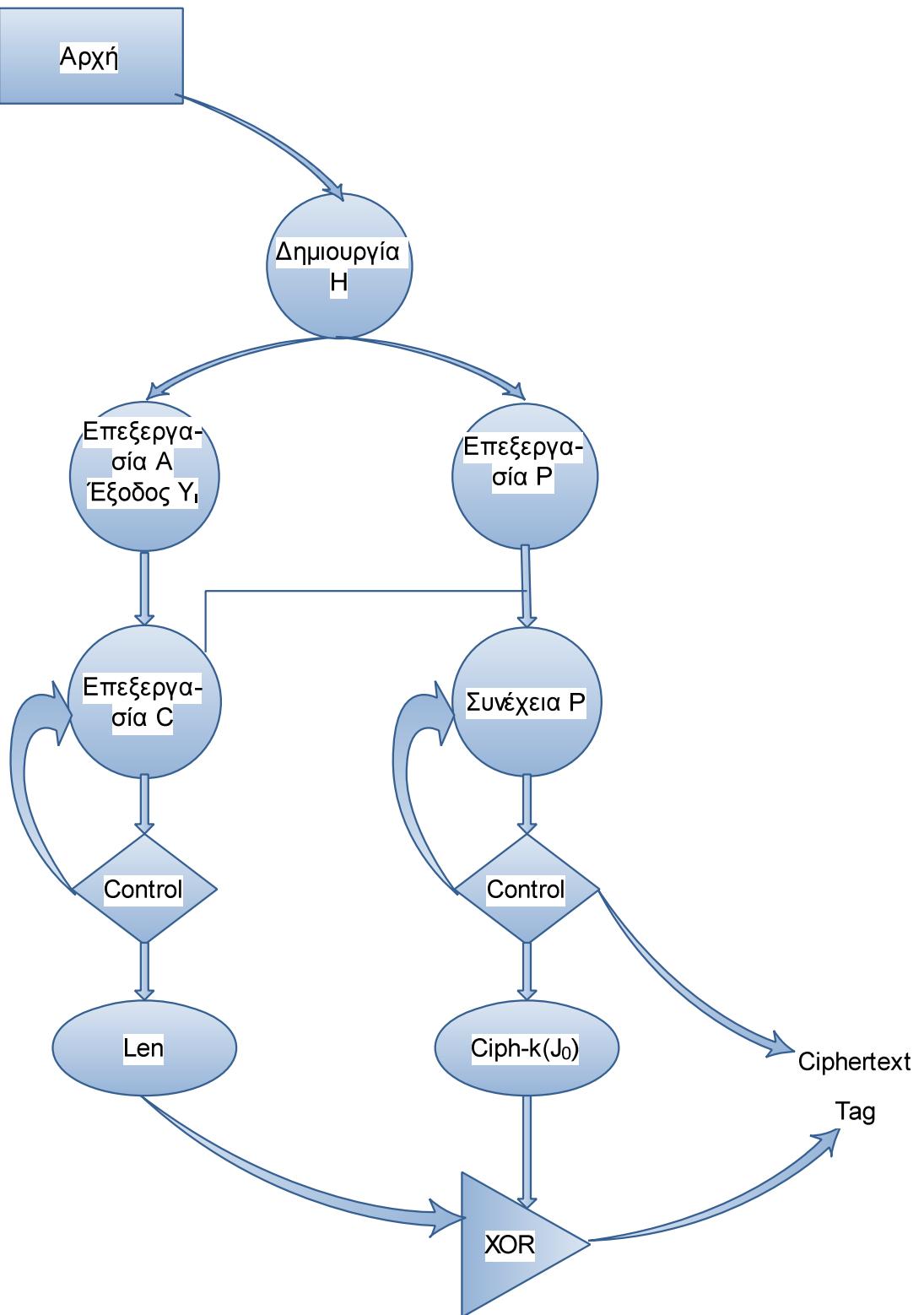
→ Όταν το Tout έχει επεξεργαστεί τα AAD δεδομένα τότε τροφοδοτείται με τα Ciphertext πακέτα των 128 bits.

→ Ακολουθεί η παράλληλη επεξεργασία για την παραγωγή του Ciphertext από το Plaintext και στον μηχανισμό αυθεντικότητας η επεξεργασία του προηγούμενου πακέτου ciphertext.

→ Όταν έχει ολοκληρωθεί η διαδικασία παραγωγής του Ciphertext τότε στην GHASH απομένουν άλλα δύο πακέτα να επεξεργαστούν. Αυτά είναι το τελευταίο πακέτο Ciphertext και το τελικό πακέτο εισόδου της GHASH που αποτελείται από τις πληροφορίες σχετικά με τον αριθμό των πακέτων των AAD και plaintext.

Όσα εξηγήθηκαν παραπάνω, αναπαρίσταται και σχηματικά στο διάγραμμα ροής που ακολουθεί. Το παρακάτω διάγραμμα ροής στηρίζεται στην hardware υλοποίηση που δόθηκε στην **Εικόνα 1** και αφορά στον τρόπο που οι δύο μηχανισμοί συνεργάζονται μεταξύ τους κατά την διεργασία της κρυπτογράφησης ενώ δεν μελετώνται θέματα χρονισμού. Η μελέτη αυτή δίνεται αναλυτικά σε επόμενη ενότητα.

## Διάγραμμα ροής Μηχανισμών στην Κρυπτογράφηση



Το πρόβλημα που υπάρχει εδώ είναι το ακόλουθο. Πρέπει οι δύο μηχανισμοί να είναι συγχρονισμένοι. Πρέπει δηλαδή τα δεδομένα να επεξεργάζονται με τον ίδιο ρυθμό ούτος ώστε να μην περιμένει πολύ ή να προτρέχει σε επεξεργασία κάποιος μηχανισμός. Με άλλα λόγια, θα πρέπει οι δύο μηχανισμοί να έχουν καθυστέρηση της ίδιας τάξης μεγέθους. Αν κάτι τέτοιο δεν επιτευχθεί τότε το συνολικό κύκλωμα δεν θα παρουσιάζει ομοιόμορφη συμπεριφορά και η συνολική καθυστέρηση του κυκλώματος θα εξαρτάται από τον αριθμό των δεδομένων. Κάτι τέτοιο όμως θα αποτελούσε κακή σχεδίαση.

Στο μηχανισμό εμπιστευτικότητας μπορούν να χρησιμοποιηθούν τεχνικές διοχέτευσης αφού υπάρχει σαφής διαχωρισμός μεταξύ των πακέτων αλλά και μεταξύ των εσωτερικών μηχανισμών. Η χρήση των τεχνικών αυτών σταματά όταν έχουμε πετύχει χρόνους καθυστέρησης ίδιας τάξης μεγέθους με αυτούς του μηχανισμού πιστοποίησης καθώς περεταίρω βελτίωση του Cout δεν θα επιφέρει συνολική βελτίωση του συστήματος αφού το ο μηχανισμός Tout θα τροχοπεδεί το σύστημα.

Ο πολλαπλασιαστής μπορεί να υλοποιηθεί με τρείς κατηγορίες πολλαπλασιαστών. Οι τρείς αυτές κατηγορίες είναι ο σειριακός, ο digital serial και ο παράλληλος πολλαπλασιαστής. Χρησιμοποιούμε τον όρο κατηγορίες γιατί σε κάθε οικογένεια έχουν προταθεί διάφορες μέθοδοι υλοποίησης. Με βάση το γεγονός ότι το σύστημα που υλοποιούμε σχεδιάζεται για high throughput απαιτήσεις, η επιλογή του ταχύτερου πολλαπλασιαστή που ικανοποιεί τις ανάγκες του συστήματος αποτελεί μονόδρομο.

Ο πολλαπλασιαστής που έχει επιλεγεί να χρησιμοποιηθεί είναι ο παράλληλος πολλαπλασιαστής Mastrovito. Η επιφάνεια όμως που απαιτείται για την υλοποίηση του πολλαπλασιαστή αυτού είναι ανάλογη του τετραγώνου του αριθμού των bits που πολλαπλασιάζει (στην περίπτωσή μας 128 bits). Ο τρόπος υλοποίησης του παράλληλου Mastrovito πολλαπλασιαστή ξεφεύγει από τα όρια της παρούσας διπλωματικής.

Εσωτερικά, όσο αφορά στον συγχρονισμό των δύο μηχανισμών, η τακτική που υιοθετήσαμε είναι η εξής:

Εφόσον οι δύο μηχανισμοί Cout και Tout έχουν καθυστέρηση ίδιας τάξης μεγέθους έχουμε:

- Παραγωγή του hash subkey (H) από εξωτερικό κύκλωμα.
- Αφίξη των δεδομένων που προορίζονται για πιστοποίηση και κρυπτογράφηση.
- Επεξεργασία των δεδομένων AAD από τον μηχανισμό Tout και παράλληλη επεξεργασία των δεδομένων Plaintext του μηχανισμού Cout, μέχρι να επεξεργαστούν όλα τα πακέτα των δύο διαφορετικών δεδομένων εισόδου. Σε περίπτωση που το σύστημα λειτουργεί σε GMAC mode, δηλαδή δεν έχουμε plaintext δεδομένα προς κρυπτογράφηση, τότε τίθεται σε απενεργοποίηση ο μηχανισμός του Cout. Ο block Cipher του Cout μηχανισμού ενεργοποιείται μόνο για να παράγει το Ciph-k (J0).

→ Όταν το πρώτο πακέτο ciphertext έχει παραχθεί τότε αυτό εισέρχεται στον μηχανισμό Tout για επεξεργασία.

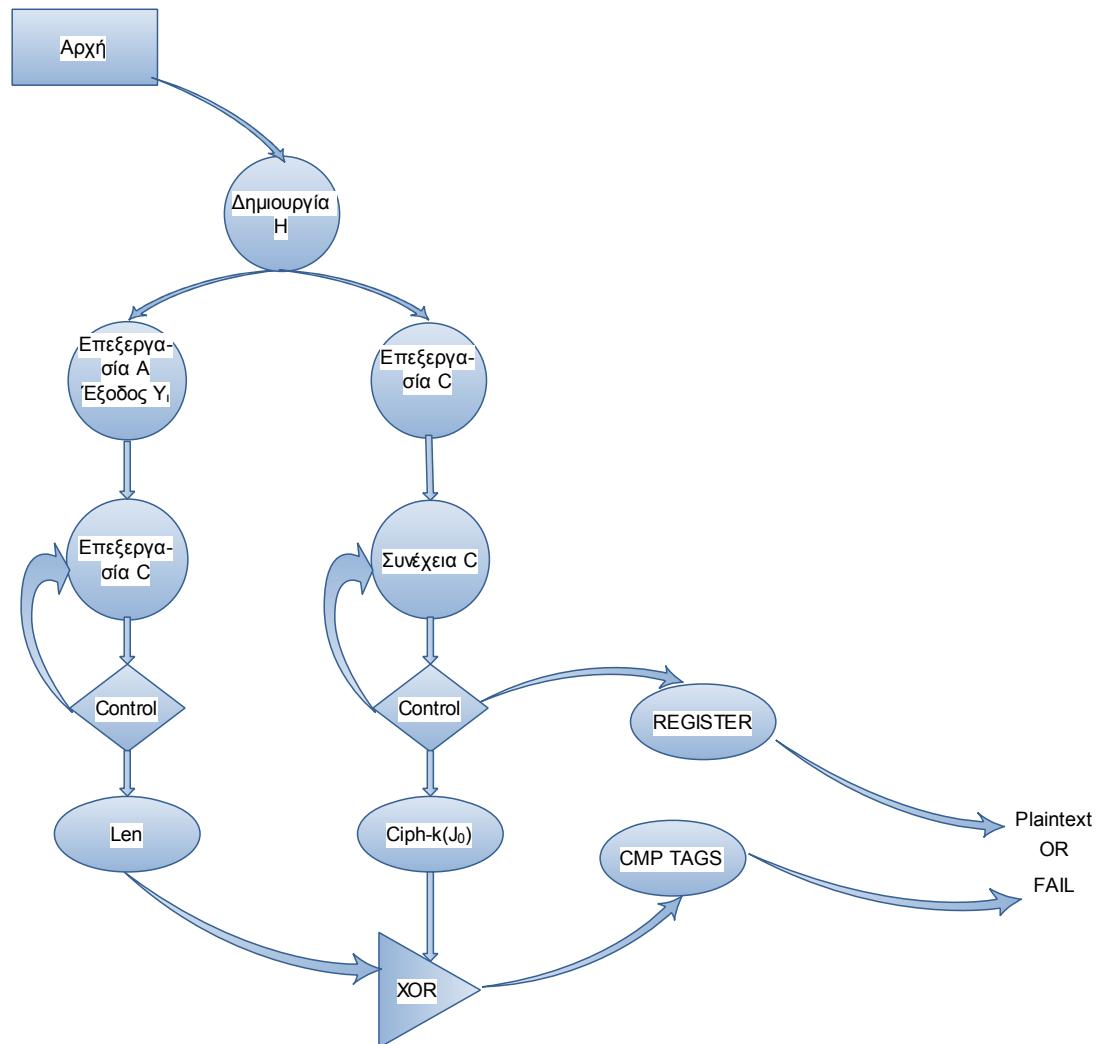
→ Το παραπάνω βήμα επαναλαμβάνεται για όλα τα πακέτα των 128 bit.

→ Όταν πλέον έχει παραχθεί και το τελευταίο πακέτο του ciphertext, τότε ο Cout μηχανισμός απενεργοποιείται ενώ στον Tout απομένουν άλλα δύο πακέτα για επεξεργασία. Αυτά είναι το τελευταίο πακέτο του Ciphertext και το πακέτο των 128 bits που περιέχει τις πληροφορίες σχετικά με τα μεγέθη του AAD και Plaintext\ciphertext.

→ Τέλος το αποτέλεσμα της εξόδου του πολλαπλασιαστή γίνεται XOR με την τιμή Ciph-k(J0) και προκύπτει το Tag.

Στην συνέχεια ακολουθεί το αντίστοιχο διάγραμμα ροής των μηχανισμών για την διαδικασία της αποκρυπτογράφησης:

### Διάγραμμα ροής Μηχανισμών στην Αποκρυπτογράφηση

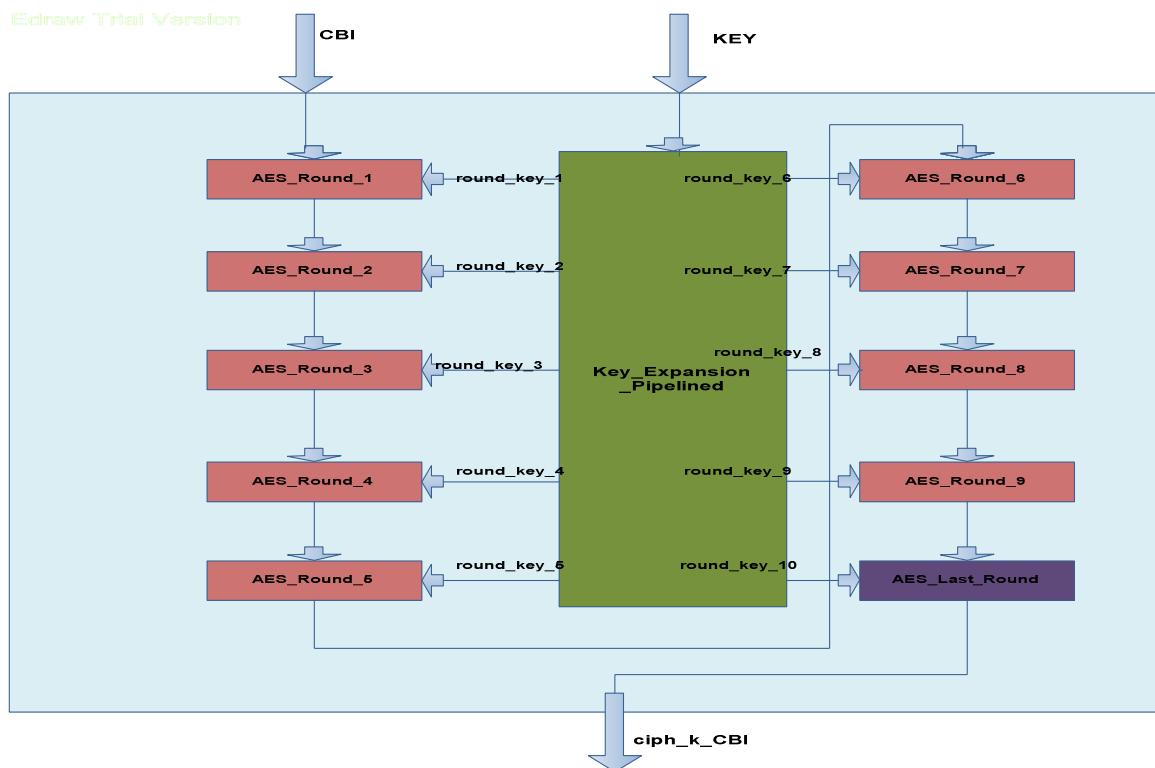


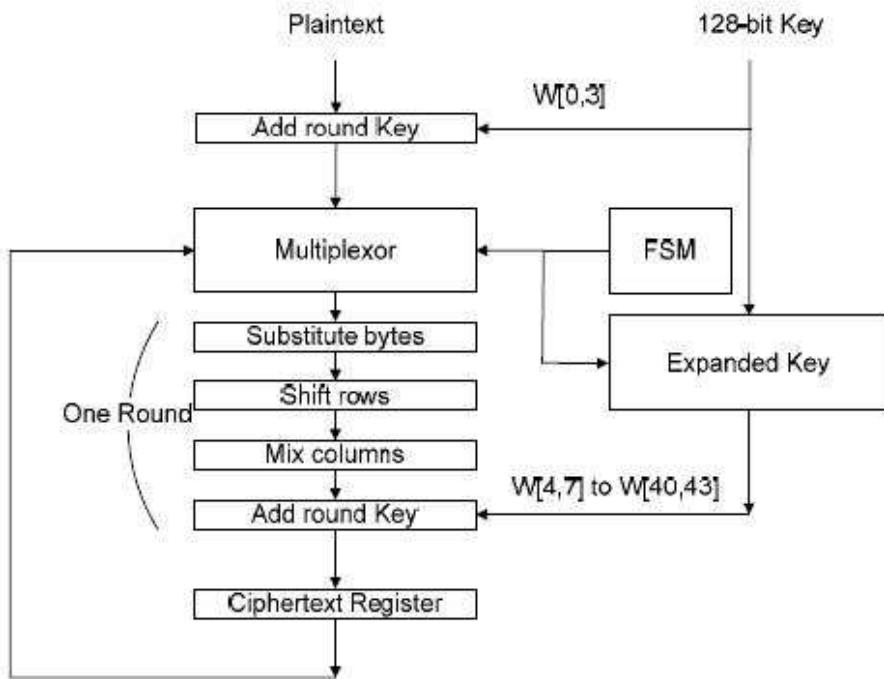
## 5.3 Σχεδιασμός των βασικών δομών του GCM

Στην AES-GCM κρυπτογράφηση και AES-GCM αποκρυπτογράφηση, τα AES και GHASH είναι τα βασικά modules που είναι υπεύθυνα για την εμπιστευτικότητα και αυθεντικότητα, αντίστοιχα. Το AES module είναι το βασικό component του GCTR module το οποίο υλοποιεί τον μηχανισμό της εμπιστευτικότητας (confidentiality mechanism) και είναι το ίδιο για τις δύο διεργασίες της πιστοποιημένης κρυπτογράφησης και πιστοποιημένης αποκρυπτογράφησης του AES-GCM προτύπου ασφαλείας. Στις ενότητες που ακολουθούν παρουσιάζονται όσα προαναφέρθηκαν καθώς και η ροή δεδομένων του AES-GCM.

### 5.3.1 AES Δομή

Για 128-bit κλειδί, ο AES αλγόριθμος απαιτεί για τον υπολογισμό 10 γύρους από μετασχηματισμούς, και κάθε γύρος περιλαμβάνει τέσσερις φάσεις: SubBytes, ShiftRows, MixColumns και AddRoundKey (βλέπε ενότητα 3.1). Αυτό επιτρέπει την υλοποίηση του AES αλγόριθμου είτε με επαναληπτική μέθοδο ή με μέθοδο διοχέτευσης. Σε έναν επαναληπτικό AES σχεδιασμό, ο γύρος μετασχηματισμού υλοποιείται μόνο μία φορά (βλέπε **Εικόνα 2**). Αυτός ο γύρος μετασχηματισμού χρησιμοποιείται 10 φορές σε 10 υπολογιστικούς κύκλους ρολογιού (computational round clocks) ενώ η προσωρινή τιμή αποθηκεύεται σε έναν register\_Ciphertext και χρησιμοποιείται ως είσοδο για τον επόμενο κύκλο. Ένας Pipelined AES σχεδιασμός μπορεί να υπολογίσει και τους 10 γύρους μετασχηματισμών σε ένα μόνο κύκλο ρολογιού αντιγράφοντας έναν γύρο 10 φορές (βλέπε **Εικόνα 11** κεφαλαίου 3). Μία Pipelined AES αρχιτεκτονική μπορεί να επιτευχθεί τοποθετώντας 128-bit registers μεταξύ κάθε γύρου. Σε μεγάλα FPGAs, οι registers είναι σχεδόν χωρίς καθυστερήσεις. Μια Pipelined δομή μπορεί να επωφεληθεί από αυτό το χαρακτηριστικό των FPGAs.





**Εικόνα 2. AES Iterative Data Path**

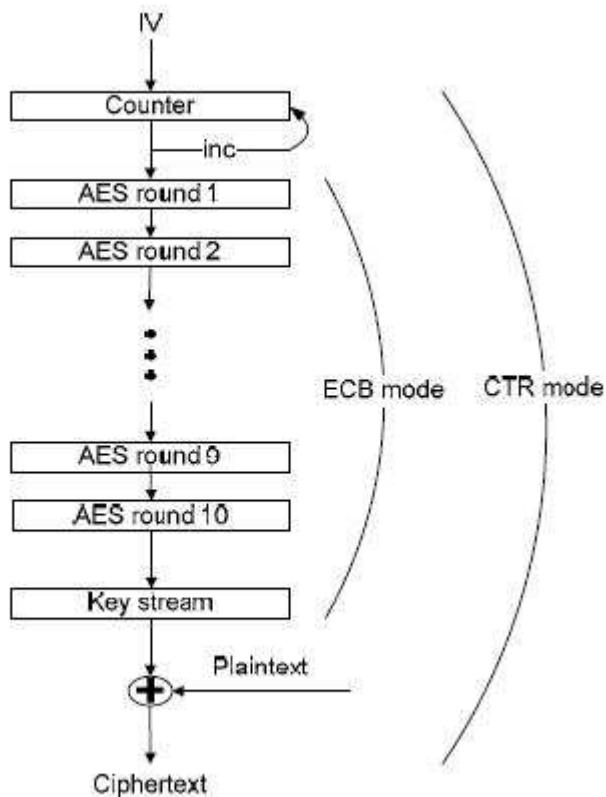
Η λογική ελέγχου και των δύο AES αρχιτεκτονικών υλοποιείται με χρήση μιας finite state machine (FSM) ή διαφορετικά ο έλεγχος γίνεται μέσω counters. Η τελευταία πρόταση είναι και αυτή που εφαρμόστηκε στην παρούσα εργασία. Ο **Table 1** δείχνει μια πρόχειρη σύγκριση μεταξύ των δύο αυτών προσεγγίσεων για την απόδοση και το κόστος.

Architectures (expanded keys stored in CLBs)	Num of unrolled rounds in hardware	Num of ciphertext per 10 clock cycles
Iterative AES	1	1
Pipelined AES	10	10

**Table 1. Comparison between Iterative and Pipelined AES**

Η αρχιτεκτονική του AES-GCM αλγόριθμου που προτείνεται σε αυτή την εργασία, υποδεικνύει το εσωτερικό AES block να υλοποιείται με αρχιτεκτονική διοχέτευσης. Αυτό το AES block αποτελεί τον πυρήνα ενός υβριδικού mode που προκύπτει από την ECB και CTR mode. Αυτό το υβρίδιο είναι στην πραγματικότητα το GCTR (βλέπε ενότητα 4.2.2). Η **Εικόνα 3** δείχνει το πώς η pipelined δομή του AES είναι ενσωματωμένη στο ECB module και το πώς το ECB module είναι ενσωματωμένο στο

CTR module. Στο ECB module, το AES block κρυπτογραφεί μια είσοδο που στην πραγματικότητα είναι μια συνεχώς αυξανόμενη counter τιμή του CTR module, και παράγει την έξοδο ως ένα keystream. Στο CTR module, τα παραγόμενα keystreams γίνονται XORed με τα plaintext για να παραχθούν τα αποτελέσματα, δηλαδή τα ciphertext. Μετά τους πρώτους 10 κύκλους ρολογιού, το pipeline του AES έχει συμπληρωθεί πλήρως, έχει γεμίσει, έτσι ώστε το AES module να μπορεί να παράγει ένα νέο 128-bit keystream σε κάθε κύκλο ρολογιού.



**Εικόνα 3. AES CTR over ECB Mode Cipher Structure**

Ο επαναληπτικός AES μπορεί επίσης και να εφαρμοστεί στον AES-GCM αλγόριθμο, ειδικά για τη δημιουργία του hash subkey H. Ο υπολογισμός αυτός μπορεί να γίνει εκ των προτέρων από ένα εξωτερικό κύκλωμα, αν το 128-bit κλειδί είναι γνωστό, καθώς το H δεν είναι τίποτε άλλο από την έξοδο του AES iterative module με μηδενική είσοδο. Ως εκ τούτου, χρειάζεται 10 κύκλους ρολογιού για να δημιουργήσει το H.

### 5.3.2 Σχεδίαση του GCTR Μηχανισμού Εμπιστευτικότητας

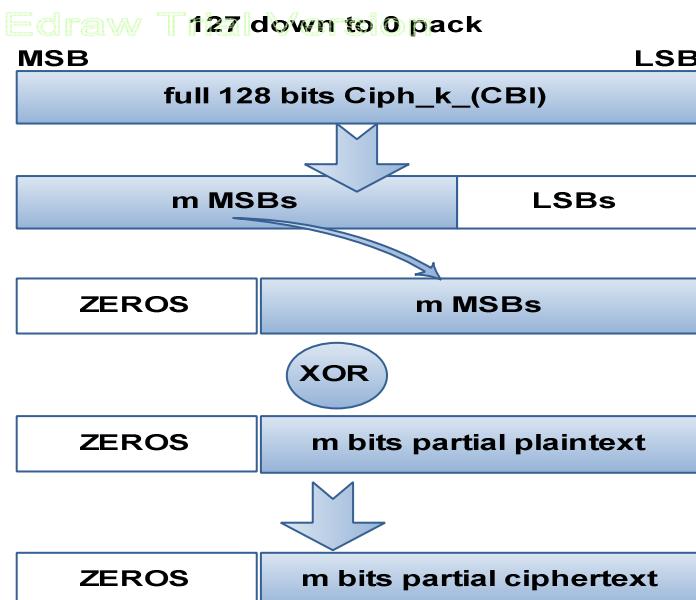
Το GCTR εκτός από το AES module και την incrementing function περιλαμβάνει και τα ακόλουθα components:

#### FIFO

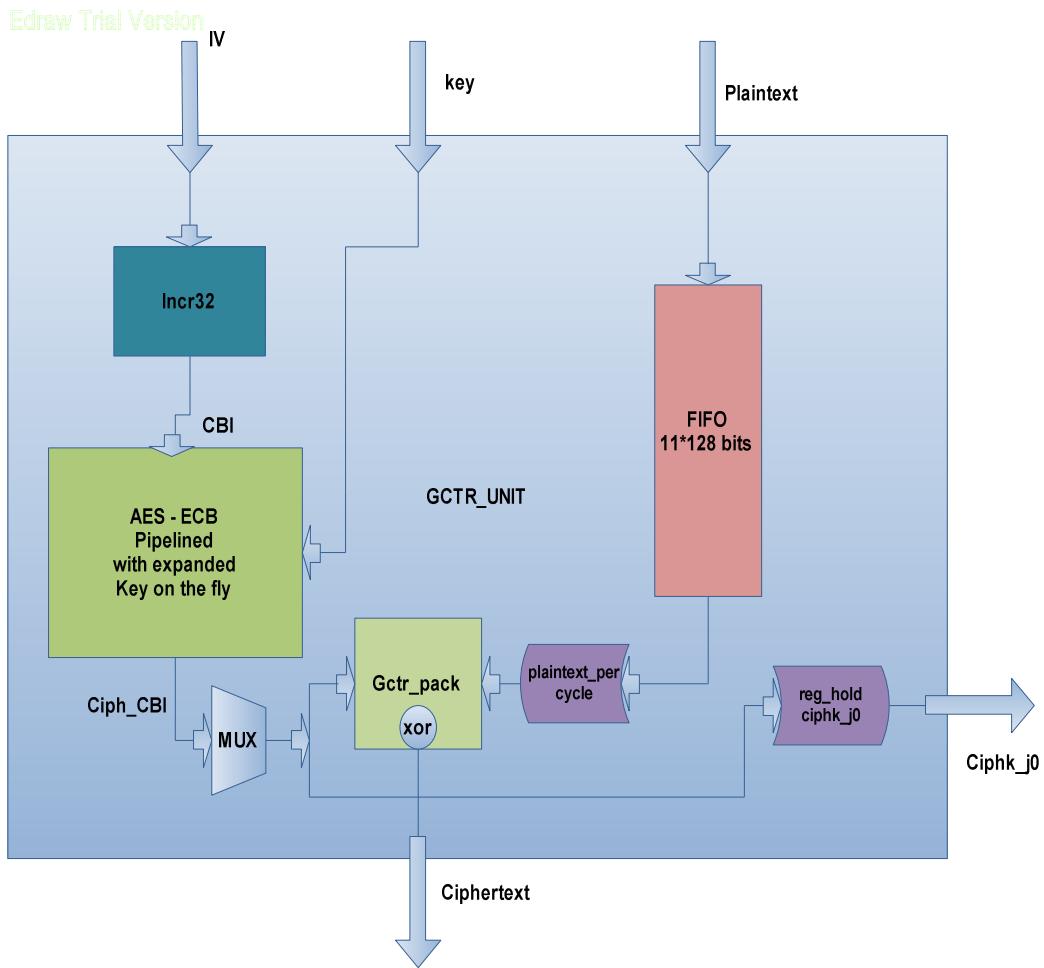
Δεδομένου ότι η καθυστέρηση είναι 11 κύκλοι ρολογιού (10 κύκλους ρολογιού για να συμπληρωθεί το pipeline του AES + 1 κύκλο ρολογιού για να εκτελεστεί η stream κρυπτογράφηση), μία 11\*128-bit FIFO πρέπει να χρησιμοποιηθεί για την αποθήκευση των εισερχόμενων πακέτων text.

#### Gctr\_pack

Δεδομένου ότι τα μήκη των AAD και text δεν είναι προαπαιτούμενα και το σύστημα δεν γνωρίζει εάν υπάρχει μερικό πακέτο (partial group) στην είσοδο, μια δομή από πολυπλέκτες χρησιμοποιείται για την ανίχνευση αυτών των μερικών πακέτων. Αυτή η δομή δείχνεται ως Gctr\_pack και ανιχνεύει μέσω των text αν υπάρχει κάποιο μπλοκ δεδομένων μικρότερο των 128 bits, και αν ναι, βρίσκει από πόσα bits αποτελείται αυτό το μερικό πακέτο. Αυτό το βήμα είναι απαραίτητο γιατί το κρυπτογραφημένο counter block που έρχεται από τον AES-ECB είναι πάντα full pack των 128 bits. Έτσι, στην περίπτωση των μερικών πακέτων θα πρέπει να κρατήσουμε μόνο τα m MSB. Αυτά συμπληρώνουν τα ‘m down to 0’ bits του 128-bit πακέτου ενώ τα υπόλοιπα ‘127 down to m’ bits του 128-bit πακέτου συμπληρώνονται με μηδενικά. Έτσι μόνο η XOR πράξη του text με το κρυπτογραφημένο counter block θα δώσει το σωστό αποτέλεσμα στην έξοδο. Διαφορετικά η κρυπτογράφηση / αποκρυπτογράφηση θα είναι ανεπιτυχείς για τα partial group. Παρακάτω δείχνεται σχηματικά η διαδικασία που μόλις περιγράφηκε.



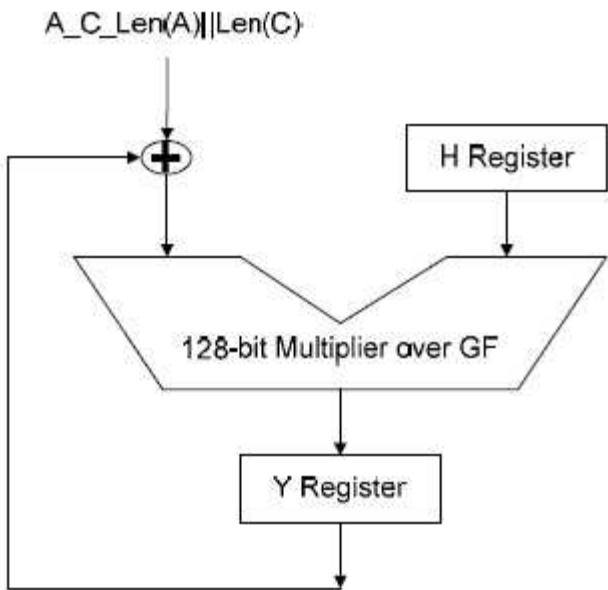
Η αρχιτεκτονική του GCTR module δίνεται παρακάτω.



Παρατηρούμε ότι τελικά στην έξοδο της GCTR παίρνουμε δύο τιμές. Τα πακέτα των 128-bit text και το πακέτο των 128-bit Ciphk\_j0 το οποίο αποθηκεύεται για να δοθεί στην GHASH. Ο MUX ελέγχει το πότε το Ciphk\_j0 είναι έτοιμο. Το σήμα ελέγχου του MUX δεν φαίνεται στο σχήμα αλλά προέρχεται από την FIFO η οποία ειδοποιεί τον MUX έναν κύκλο πριν αρχίσει να δίνει τα plaintext, όταν δηλαδή το Ciphk\_j0 έχει παραχθεί από τον AES-ECB.

### 5.3.3 GHASH Δομή

Ένας 128-bit πολλαπλασιαστής πάνω στο πεδίο  $GF(2^{128})$  αποτελεί τον πυρήνα της αρχιτεκτονικής της GHASH. Στον AES-GCM, ο  $GF(2^{128})$  πολλαπλασιαστής πεδίου, πολλαπλασιάζει δύο τελεστές των 128-bit ‘πράξη modulo’ το πολυώνυμο πεδίου, το οποίο είναι το :  $F(x) = 1 + x + x^2 + x^7 + x^{128}$ , και δημιουργεί τελικά μια 128-bits έξοδο. Η GHASH αρχιτεκτονική φαίνεται στην **Εικόνα 4**. Ένας τελεστής του GF πολλαπλασιαστή είναι το hash subkey H το οποίο μπορεί να αντιμετωπιστεί ως μια fixed 128-bit σταθερά και για το λόγο αυτό δεν θα αλλάξει αν το 128-bit κλειδί και ο IV δεν αλλάξουν. Ο register Y, του οποίου η αρχική τιμή είναι μηδέν, κατέχει την ενδιάμεση τιμή hash για το επόμενο βήμα του υπολογισμού της πιστοποίησης (authentication).



**Εικόνα 4. GHASH Hardware Architecture**

Η αρχιτεκτονική του φαίνεται στην **Εικόνα 4**, και βασίζεται σε μια λειτουργία επανάληψης. Ας υποθέσουμε ότι όλα τα δεδομένα εισόδου και εξόδου ικανοποιούν τους ορισμούς που έχουν αναφερθεί σε προηγούμενο κεφάλαιο. Κατά τους πρώτους m κύκλους ρολογιού, η ακολουθία του 128-bit μπλοκ δεδομένων AAD A1, A2, ..., Am γίνεται hashed μέσα στην GHASH με μία από τις δύο εισόδους των XOR πυλών, όπως περιγράφεται από τον αλγόριθμο 3. Στους επόμενους n κύκλους ρολογιού, η ακολουθία του 128-bit ciphertext μπλοκ C1, C2, ..., Cn-1, Cn γίνεται hashed με τις ίδιες εισόδους των XOR πυλών όπως και στην περίπτωση της AAD ακολουθίας. Στον τελευταίο κύκλο ρολογιού, η 128-bit λέξη length(A) || length(C) γίνεται hashed. Εν τω μεταξύ, η ενδιάμεση τιμή hash Yi έχει ανατροφοδοτηθεί στην άλλη είσοδο της XOR πυλης για τη δημιουργία του άλλου τελεστή του GF πολλαπλασιαστή.

Απαιτούνται τελικά  $m + n + 1$  κύκλοι ρολογιού για να υπολογιστεί η τιμή της hash για τον παράλληλο πολλαπλασιαστή, και  $128 * (m + n + 1)$  κύκλοι ρολογιού για τον Bit Serial πολλαπλασιαστή. Υπάρχει μια πρόχειρη σύγκριση στον **Table 2** μεταξύ των GHASH αρχιτεκτονικών που χρησιμοποιούν αυτά τα δύο είδη των πολλαπλασιαστών.

Ο τρόπος υλοποίησης του παράλληλου Mastrovito πολλαπλασιαστή ξεφεύγει από τα όρια της παρούσας διπλωματικής.

GHASH architecture	Latency (clock cycle)	Hardware complexity ( $k=128$ )	Num of Slices
Using Bit Serial Multiplier	$128 * (m + n + 1)$	$O(k)$	282
Using Mastrovito Bit Parallel Multiplier	$m + n + 1$	$O(k^2)$	8297

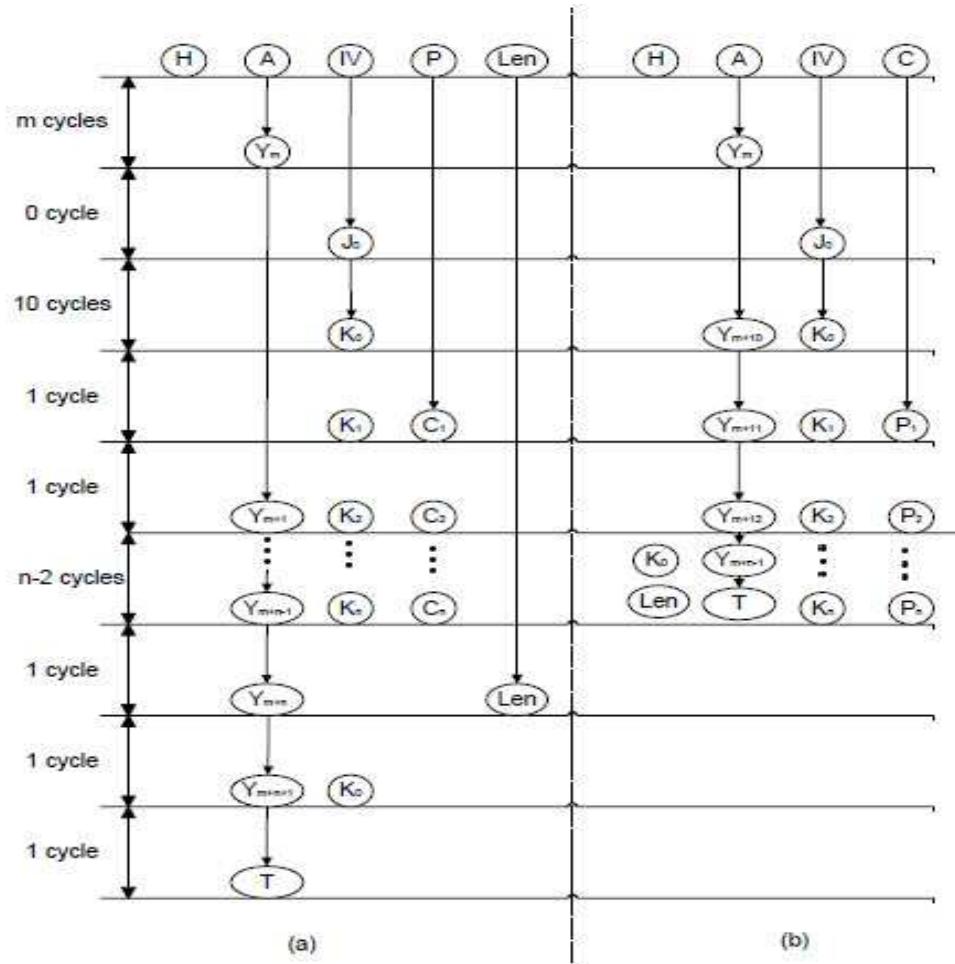
**Table 2. Comparison between Different GHASH Architectures**

### 5.3.4 Ροή δεδομένων στον GCM και Χρονισμοί

Εάν η AES δομή υλοποιείται με μια pipelined αρχιτεκτονική, και η GHASH δομή υλοποιείται με την επιλογή ενός παράλληλου πολλαπλασιαστή ως πυρήνα της, και το hash subkey  $H$  μπορεί να υπολογιστεί εξωτερικά από μια επαναληπτική AES δομή, όπου κάθε υπολογιστικό μέρος βασίζεται σε ένα γνωστό κλειδί, τότε η ροή δεδομένων στην GCM κρυπτογράφηση φαίνεται στην **Εικόνα 4 (α)** ενώ η ροή δεδομένων στην GCM αποκρυπτογράφηση φαίνεται στην **Εικόνα 4 (β)**.

Αρχικά, στην GCM κρυπτογράφηση, ο AES-GCM ξεκινά με τον υπολογισμό των ενδιάμεσων hash τιμών  $Y_i$  όταν λαμβάνει τα AAD δεδομένα. Απαιτούνται  $m$  κύκλοι ρολογιού για να δημιουργηθεί το  $Y_m$ , όσα δηλαδή και τα πακέτα των 128 bits δεδομένων AAD. Στη συνέχεια, αφού έχει ολοκληρωθεί η επεξεργασία των AAD πακέτων, η GHASH πρέπει να είναι σε αδράνεια για 11 κύκλους ρολογιού, μέχρι το πρώτο ciphertext μπλοκ,  $C_1$ , να προέλθει από την GCTR η οποία έχει υλοποιηθεί με τη χρήση ενός pipelined AES module. Για GCM με προεπιλεγμένο IV, ο IV είναι πάντα μήκους 96 bits, και το  $J_0$  μπορεί να δημιουργηθεί απευθείας από την συνένωση δύο αλληλουχιών bit strings, τον IV και ένα zero 32-bit string.

Τα key streams για την GCM κρυπτογράφηση δημιουργούνται μετά τον  $10^0$  κύκλο ρολογιού από την στιγμή που ο  $J_0$  τοποθετείται μέσα στο pipeline του GCTR. Συγκεκριμένα, στον  $10^0$  κύκλο ρολογιού, όπου το pipeline του AES γεμίζει πλήρως, το  $K_0$  (δηλαδή το CIPH-K ( $J_0$ )) είναι έτοιμο και αποθηκεύεται. Στον  $11^0$  κύκλο ρολογιού, το πρώτο cipher block  $C_1$  δημιουργείται και τοποθετείται στην GHASH και η GHASH αρχίζει να κάνει hashing τα δεδομένα ξανά. Το ίδιο συμβαίνει σε κάθε κύκλο μέχρι να κρυπτογραφηθούν όλα τα  $n$  plaintext πακέτα. Στον  $11 + m + n + 1$  κύκλο ρολογιού, το  $Y_{m+n+1}$  δημιουργείται και γίνεται XORed με το  $K_0$  για να δημιουργήσει την ετικέτα γνησιότητας (Tag).



**Εικόνα 4. (a) The Data Flow of GCM Encryption (b) The Data Flow of GCM Decryption**

Για την GCM αποκρυπτογράφηση, η GHASH μπορεί να υπολογίζει απευθείας την ετικέτα γνησιότητας  $T'$  βασισμένη στα AAD και ciphertext  $C$  που δέχεται από την είσοδο της GCM αποκρυπτογράφησης. Ως εκ τούτου, 11 κύκλοι ρολογιού εξοικονομούνται σε σύγκριση με την ροή δεδομένων στην GCM κρυπτογράφηση.

Στο σημείο αυτό πρέπει να σημειώσουμε ότι η ανάλυση που δόθηκε παραπάνω αφορά στην περίπτωση που τα μήκη των AAD και text είναι προαπαιτούμενα, δηλαδή το σύστημα γνωρίζει τον ακριβή αριθμό τους και τα παίρνει από την είσοδο. Επιπλέον, στην ανάλυση αυτή τα επεκταμένα κλειδιά, το hash subkey και τα CBIs είναι προϋπολογισμένα και αποθηκευμένα στη μνήμη.

Στην παρούσα εργασία έχει αναπτυχθεί ο "online" τρόπος του AES-GCM προτύπου ασφαλείας με την έννοια ότι τα μήκη των AAD και text δεν είναι προαπαιτούμενα και τα επεκταμένα κλειδιά, το hash subkey καθώς και τα CBIs, δεν είναι προϋπολογισμένα και αποθηκευμένα στη μνήμη αλλά όλα τα επιμέρους στοιχεία υπολογίζονται on the fly. Αυτός ο τρόπος επιφέρει μια καθυστέρηση 3 κύκλων ρολογιού στην έξοδο του πρώτου text πακέτου, το οποίο είναι διαθέσιμο στο 14<sup>ο</sup> κύκλο και μια καθυστέρηση 4 κύκλων ρολογιού στην έξοδο της ετικέτας γνησιότητας (Tag).

Τελικά, το  $Y_{m+n+1}$  δημιουργείται στον  $14 + m + n + 4$  κύκλο ρολογιού, και γίνεται XORed με το  $K_0$  για να δημιουργήσει την ετικέτα γνησιότητας (Tag).

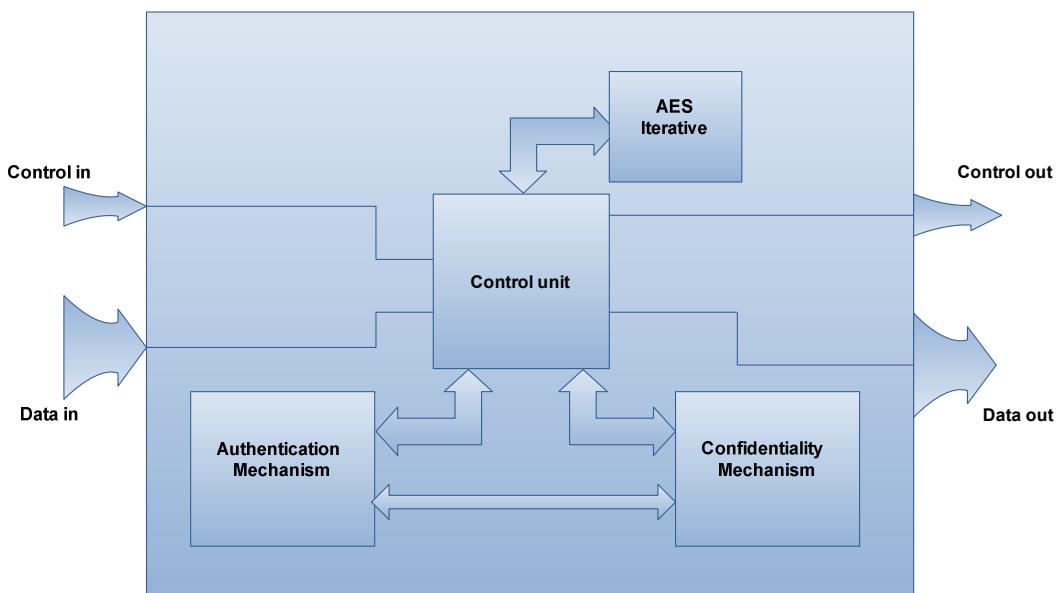
Παρόλο που υπάρχει αυτή η διαφορά στον "online" τρόπο λειτουργίας, θεωρούμε ότι ο τρόπος αυτός είναι πιο δυνατός σε θέματα ασφαλείας. Ο λόγος είναι ότι οι μνήμες αποτελούν ένα ευάλωτο σημείο στη λειτουργία ενός κρυπτογραφικού προτύπου ασφαλείας, επειδή κάποιος εξωτερικός παρατηρητής έχει εύκολη πρόσβαση σε αυτές και είναι δυνατόν να παραβιαστούν για την συλλογή πληροφοριών που μπορεί να προκαλέσουν το "σπάσιμο" του συστήματος.

## 5.4 Υψηλής Ταχύτητας Αρχιτεκτονική του GCM

Όπως αναφέρθηκε παραπάνω, ο μηχανισμός αυθεντικότητας πιστοποιεί τόσο τα AAD δεδομένα όσο και τα ciphertext δεδομένα κατά την διεργασία της κρυπτογράφησης. Κατά την διεργασία της αποκρυπτογράφησης, πιστοποιεί την ορθή λήψη των δεδομένων με βάση το tag που δέχεται στην είσοδο. Οι δύο μηχανισμοί μπορούν να τρέχουν παράλληλα. Το πρόβλημα που υπάρχει όμως είναι το ακόλουθο. Κατά την διεργασία της κρυπτογράφησης η GCTR μπορεί να παράγει το ciphertext και η GHASH να επεξεργάζεται τα AAD δεδομένα. Όμως, τα παραχθέν ciphertext πακέτα πρέπει να περιμένουν μέχρι να τελειώσει η επεξεργασία των AAD δεδομένων από την GHASH. Αν τα AAD και ciphertext δεδομένα είναι αυθαιρέτου μεγέθους τότε θα πρέπει να υπάρχει και register αυθαίρετου μεγέθους ή μια αρκετά μεγάλη FIFO που θα διασφαλίζει ότι δεν θα χαθούν τα δεδομένα. Κάτι τέτοιο σε hardware θα επιβαρύνει σε μεγάλο βαθμό το σύστημα εφόσον δεν γνωρίζουμε το μέγεθος των δεδομένων εισόδου.

Η λύση που προτείνεται σε αυτό το πρόβλημα δίνεται παρακάτω. Να υπάρχει μία μοναδική είσοδος στο σύστημα και τα δεδομένα να εισέρχονται σε πακέτα των 128 bits. Πρώτα τα AAD πακέτα και μετά τα plaintext ή ciphertext πακέτα ή αλλιώς τα text πακέτα. Ο παραλληλισμός των διεργασιών δεν χάνεται αφού όταν η GHASH επεξεργαστεί τα AAD δεδομένα, τότε και οι δύο μηχανισμοί θα είναι σε στάδιο επεξεργασίας των text δεδομένων με μια καθυστέρηση στην είσοδο της GHASH το πολύ 11 με 14 κύκλους μέχρι να γεμίσει το pipeline του AES και να προκύψει το πρώτο πακέτο text που θα κατευθυνθεί στον μηχανισμό πιστοποίησης. Μετά την καθυστέρηση άφιξης του πρώτου πακέτου στην GHASH, σε κάθε κύκλο και οι δύο μηχανισμοί, οι δύο δομές, θα επεξεργάζονται ένα καινούριο πακέτο των 128 bits κάθε φορά. Με αυτό τον τρόπο δεν υπάρχει περιορισμός στο μέγεθος των δεδομένων που τροφοδοτούνται στην είσοδο ενώ ταυτόχρονα ο παράλληλος χαρακτήρας του GCM αλγορίθμου δεν χάνεται. Η λογική που χρησιμοποιήθηκε κατά την υλοποίηση σε hardware παρουσιάζεται σε στο παρακάτω σχήμα.

## AES-GCM MECHANISM



Το control in είναι ένα σήμα εισόδου μεγέθους 8 bits. Περιέχει πληροφορίες σχετικά με την κατάσταση λειτουργίας στην οποία βρίσκεται το AES-GCM σύστημα και σχετικά με το πακέτο εισόδου. Το control out είναι ένα σήμα εξόδου μεγέθους 8 bit και περιέχει πληροφορίες σχετικά με την εσωτερική κατάσταση των μηχανισμών. Το data in είναι σήμα εισόδου των 128 bits και μέσω αυτού εισέρχονται όλα τα δεδομένα. Το data out είναι σήμα εξόδου των 128 bits από όπου δίνονται τα ciphertext, plaintext, tag, error και ok σήματα. Όλες οι είσοδοι και έξοδοι των επιμέρους συστημάτων είναι συνδεδεμένες με το control unit το οποίο ουσιαστικά αποτελεί και τον εγκέφαλο του συστήματος και είναι υπεύθυνο για τον συντονισμό των επιμέρους κυκλωμάτων. Η μόνη απευθείας σύνδεση που υπάρχει μεταξύ των δύο μηχανισμών είναι η μεταφορά του πακέτου CIPH<sub>K</sub>(J<sub>0</sub>). Όλα τα υπόλοιπα σήματα περνάνε και ελέγχονται από το control unit.

Επιπλέον, οι επιμέρους μηχανισμοί δεν έχουν καμία απευθείας σύνδεση με τις εισόδους εξόδους του ολικού συστήματος παρά μόνο με το clock. Να σημειωθεί τέλος, ότι το control unit είναι υπεύθυνο για την εξοικονόμηση ενέργειες του συστήματος κλειδώνοντας τα επιμέρους κυκλώματα σε κατάσταση reset όταν αυτά δεν χρησιμοποιούνται ή όταν αυτά έχουν τελειώσει τη λειτουργία τους.

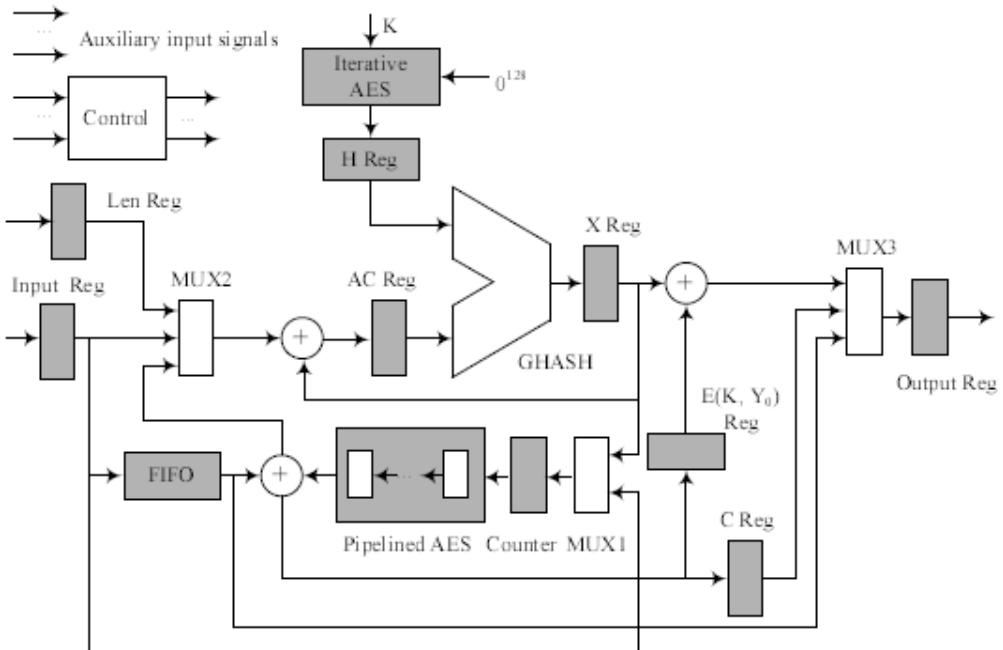
Τα control σήματα είναι σήματα handshaking. Τα σήματα ελέγχου εισόδου θα μπορούσαν να χαρακτηριστούν ως ένα set εντολών που δέχεται το σύστημα. Με αυτό τον τρόπο καθιστάτε εύκολη η επικοινωνία μεταξύ του χρήστη ή των εξωτερικών περιφερειακών συστημάτων και του εσωτερικού συστήματος. Το σήμα ελέγχου εξόδου ενημερώνει τον χρήστη σχετικά με την εσωτερική λειτουργία και με το είδος του σήματος που υπάρχει στη γραμμή δεδομένων.

Μελλοντική επέκταση του παρόντος είναι εφικτή με σκοπό να αυξηθεί το set εντολών και να εκτελούνται περισσότερες διεργασίες ή να ενημερώνεται ο χρήστης με πιο λεπτομερή τρόπο σχετικά με το στάδιο στο οποίο βρίσκεται το σύστημα εσωτερικά. Τέλος, να σημειώσουνε ότι το σύστημα θα μπορούσε να χαρακτηριστεί ως ένας κρυπτογραφικός επεξεργαστής που δουλεύει με συγκεκριμένο set εντολών. Επίσης, αφού επιλέχθηκε για υλοποίηση η χρήση FPGAs η αναβάθμιση του παρόντας δεν θα είναι δύσκολη διαδικασία.

Το set εντολών καθώς και τρόπος χειρισμού του συστήματος δίνονται σε επόμενο κεφάλαιο, ως manual για την λειτουργία του κρυπτογραφικού προτύπου ασφαλείας AES-GCM.

Το παρακάτω σχήμα δείχνει την εσωτερική δομή και λειτουργία του GCM :

### Εσωτερική δομή GCM



Μια επαναληπτική AES δομή χρησιμοποιείται για τον υπολογισμό του H από το μυστικό κλειδί K που δίνεται από το χρήστη. Ο υπολογισμός του H για μελλοντικά πακέτα μπορεί να επικαλυφθεί από τις τρέχουσες εργασίες της GCM και επομένως δεν είναι στην κρίσιμη διαδρομή του σχεδιασμού. Επιπλέον αυτή η δομή είναι η πρώτη που ενεργοποιείται πριν ακόμα ο χρήστης επιλέξει αν θα έχει τρόπο λειτουργίας κρυπτογράφησης ή αποκρυπτογράφησης.

Στην υλοποίηση που πραγματοποιήσαμε, ως μοναδικό αποδεκτό IV μέγεθος είναι τα 96 bit που αποτελεί το προεπιλεγμένο μέγεθος του IV. Έτσι ο MUX1 δεν χρειάζεται αφού χρησιμοποιείται σε περίπτωση που ο IV είναι διαφορετικός από 96 bits.

Ο MUX 2 αποτελεί εσωτερικό στοιχείο του μηχανισμού πιστοποίησης και κυκλωματικά βρίσκεται εντός του FSM.

Τα Input reg Len reg είναι τα κυκλώματα reg\_len\_pack που είναι υπεύθυνα για την μετατροπή των LSB σε MSB στα partial πακέτα και για την συνολική καταμέτρηση των πακέτων.

Ο μηχανισμός GCTR είναι μια πλήρως pipelined υλοποίηση του AES με ενσωματωμένο online μηχανισμό key expansion. Η FIFO, απαραίτητη για την σωστή λειτουργία του μηχανισμού, είναι εντός αυτού.

Ο MUX3 είναι ουσιαστικά ενσωματωμένος στο control unit του GCM και ελέγχει ποιά τιμή εξέρχεται από το σύστημα. Επιπλέον το control unit ενημερώνει τον χρήστη σχετικά με το τι είδους είναι το εξερχόμενο πακέτο μέσω της γραμμής control out.

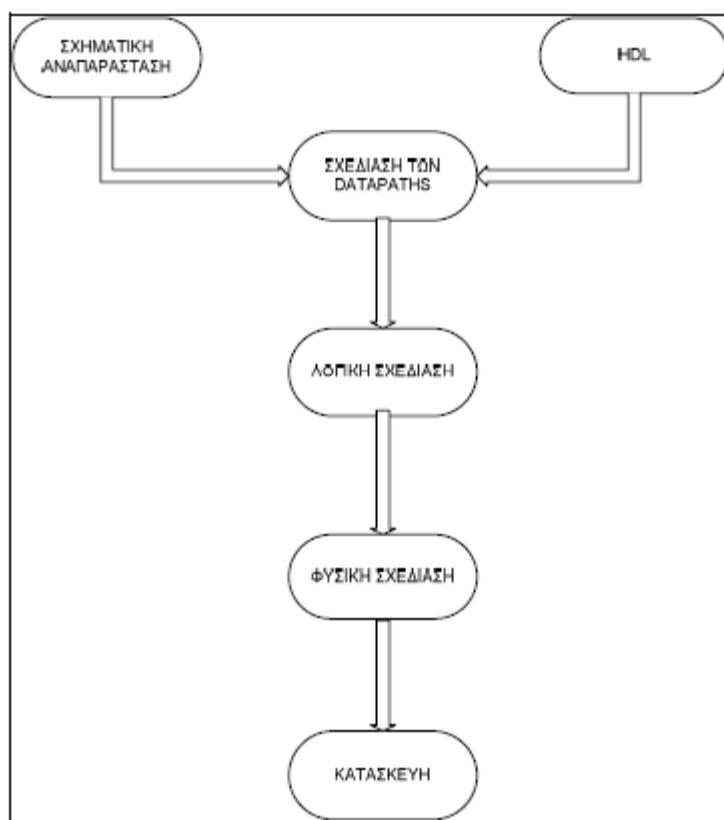


# Κεφάλαιο 6

## VHDL, ΣΧΕΔΙΑΣΗ, ΕΞΟΜΟΙΩΣΗ ΚΑΙ ΣΥΝΘΕΣΗ

### 6.1. Διαδικασία Σχεδίασμού Ψηφιακών Συστημάτων

Μία τυπική μεθοδολογία σχεδίασης ψηφιακών συστημάτων φαίνεται στην παρακάτω εικόνα.



Εικόνα 1: Μεθοδολογία σχεδίασης ψηφιακών συστημάτων

Η διαδικασία σχεδίασης ενός ψηφιακού συστήματος ξεκινάει από τον καθορισμό της συμπεριφοράς του συστήματος (behavioral description). Σε αυτό το στάδιο σχεδιασμού, γίνεται περιγραφή του συστήματος και ανάλυση αυτού σε διασυνδεδεμένα υποσυστήματα είτε με σχηματική αναπαράσταση (schematic capture) είτε με την περιγραφή του συστήματος και των επιμέρους υποσυστημάτων με την χρήση μιας γλώσσας περιγραφής υλικού (Hardware Description Language).

Το επόμενο στάδιο σχεδίασης περιλαμβάνει τον σχεδιασμό των data paths του υπό κατασκευή συστήματος. Η συμπεριφορά του συστήματος καθορίζει τις διαδικασίες μεταφοράς των δεδομένων μεταξύ των καταχωρητών και των λογικών μονάδων. Στο επόμενο στάδιο, που είναι γνωστό ως λογικός σχεδιασμός, χρησιμοποιούνται λογικές

πύλες και flip-flops για την υλοποίηση των καταχωρητών δεδομένων, των διαύλων και των μονάδων ελέγχου. Στο επόμενο επίπεδο, που είναι ο φυσικός σχεδιασμός, πραγματοποιείται η σχεδίαση των στοιχειωδών λογικών πυλών και των flip-flops σαν στοιχεία της βιβλιοθήκης. Στο τελευταίο βήμα της σχεδίασης χρησιμοποιείται η φυσική σχεδίαση του προηγούμενου επιπέδου για να κατασκευαστεί το ολοκληρωμένο κύκλωμα.

Οι παραπάνω διαδικασίες έχουν αυτοματοποιηθεί σε πολύ μεγάλο βαθμό και υποστηρίζονται από εργαλεία σύνθεσης. Με την βοήθεια αυτών των εργαλείων και χρησιμοποιώντας μία συγκεκριμένη βιβλιοθήκη λογικών και ηλεκτρονικών στοιχείων, προκύπτει από την περιγραφή του υλικού ο σχεδιασμός σε φυσικό επίπεδο, που είναι το τελευταίο στάδιο της διαδικασίας σχεδίασης που περιγράφηκε.

## 6.2. Ιεραρχική Σχεδίαση Συστημάτων

Μία βασική τεχνική σχεδίασης για την ανάπτυξη ολοκληρωμένων συστημάτων είναι η ιεραρχική σχεδίαση. Η συγκεκριμένη τεχνική στηρίζεται στην λογική του “Διαίρει και Βασίλευε” και χρησιμοποιείται για την επιτυχή αντιμετώπιση των αυξανόμενων απαιτήσεων του σχεδιασμού πολύπλοκων συστημάτων (πολλές φορές της τάξεως των εκατομμυρίων πυλών). Σύμφωνα με την τεχνική αυτή, το κύκλωμα τεμαχίζεται σε κομμάτια (μπλοκ), επαναληπτικά, μέχρι που καταλήγει σε πρωταρχικά/βασικά (primitive/common) στοιχεία.

Αναλυτικότερα, το ολοκληρωμένο τοποθετείται στο πρώτο επίπεδο της ιεραρχίας. Όμως το ολοκληρωμένο αποτελείται από διάφορα υποσυστήματα, τα οποία συνθέτουν το αμέσως επόμενο επίπεδο της ιεραρχίας. Το επίπεδο που ακολουθεί καλείται λογικό επίπεδο και σχετίζεται με τις πύλες, τα flip-flops και τα άλλα δομικά στοιχεία από τα οποία αποτελείται η βιβλιοθήκη τεχνολογίας. Το τελευταίο επίπεδο της ιεραρχίας είναι το φυσικό επίπεδο, όπου πραγματοποιείται ο φυσικός σχεδιασμός των πυλών και όλων των στοιχείων της βιβλιοθήκης.

Σημειώνεται ότι στην ιεραρχική σχεδίαση πρέπει να επαληθεύεται η ορθή λειτουργία του κυκλώματος σε κάθε επίπεδο της ιεραρχίας.

Η ιεραρχική σχεδίαση είναι ιδιαίτερα δημοφιλής καθώς επιτρέπει την επαναχρησιμοποίηση (reuse) ήδη σχεδιασμένων μπλοκ. Έτσι, πανομοιότυπα κομμάτια μπορούν να χρησιμοποιηθούν σε διάφορα σημεία του σχεδιασμού ή και ακόμη σε διαφορετικούς σχεδιασμούς. Επίσης, όπου είναι δυνατόν, προσπαθούμε να αποσυνθέσουμε (decompose) έναν πολύπλοκο σχεδιασμό σε βασικά επαναχρησιμοποιήσιμα μπλοκ (reusable blocks), τα οποία έχουν ήδη επαληθευτεί και τεκμηριωθεί και συνήθως αποθηκεύονται σε βιβλιοθήκες για γενική χρήση.

Συνεπώς, η ιεραρχική σχεδίαση μειώνει την πολυπλοκότητα του σχεδιασμού και της αντιπροσώπευσης του συνολικού σχεδίου (schematic) του κυκλώματος.

### **6.3. Γλώσσες Περιγραφής Υλικού (HDL)**

Υπάρχουν δύο βασικές ιεραρχίες σχεδίασης, η top-down και η bottom-up. Πολύ συχνά για την σχεδίαση ενός κυκλώματος ακολουθείται η top-down μεθοδολογία, σύμφωνα με την οποία ο σχεδιασμός ξεκινά από ένα υψηλό επίπεδο περιγραφής όπου ορίζεται όλο το σύστημα. Στο επόμενο επίπεδο, το σύστημα διασπάται σε επιμέρους υποσυστήματα, τα οποία μοιράζονται σε ομάδες σχεδιαστών, όπου η καθεμία αναλαμβάνει να υλοποιήσει το συγκεκριμένο υποσύστημα. Έτσι προέκυψε η ανάγκη εργαλείων σχεδιασμού που να υποστηρίζουν την top-down μεθοδολογία. Τέτοιου είδους εργαλεία πρέπει να υποστηρίζουν:

- Δυνατότητα περιγραφής της συμπεριφοράς και της δομής κάθε κυκλώματος.
- Εξομοίωση που να συνδυάζει διαφορετικούς τύπους περιγραφών. Έτσι θα είναι εφικτή η εξομοίωση της δομής ενός συστήματος, ξέροντας μόνο την συμπεριφορά και όχι τη δομή των υποσυστημάτων του.
- Υποστήριξη περιγραφής σε πολλά επίπεδα, από το πιο γενικό επίπεδο του συστήματος, μέχρι το επίπεδο περιγραφής λογικών πυλών και το επίπεδο τρανζίστορ.
- Δομή και οργάνωση της γλώσσας. Με τον όρο γλώσσα εννοούνται οι κλασικές γλώσσες προγραμματισμού, αλλά και τα σύγχρονα γραφικά περιβάλλοντα που χρησιμοποιούνται για τον προγραμματισμό.

Οι γλώσσες προγραμματισμού που χρησιμοποιούνται για software δεν καλύπτουν όλες τις απαιτήσεις που τέθηκαν παραπάνω και ιδιαίτερα δυσκολεύονται να υποστηρίζουν τα χαμηλότερα επίπεδα περιγραφής. Ακόμα, δυσκολεύονται να περιγράψουν τη δομή, ενώ παρέχουν τη δυνατότητα περιγραφής της συμπεριφοράς του υπό επεξεργασία συστήματος. Λόγω της διαρκούς αύξησης του μεγέθους και της πολυπλοκότητας των σχεδιαζόμενων ψηφιακών ολοκληρωμένων συστημάτων, νέες μέθοδοι αναζητήθηκαν για την σχεδίαση συστημάτων με γρήγορους ρυθμούς και μεγάλο βαθμό αξιοπιστίας. Το αποτέλεσμα ήταν η κατασκευή ειδικών γλωσσών περιγραφής υλικού (Hardware Description Languages, HDL), έτσι ώστε να περιγραφεί το χρησιμοποιούμενο υλικό με απλό και κατανοητό τρόπο, και να εξομοιωθεί η λειτουργία του πριν τον φυσικό σχεδιασμό του. Με αυτόν τον τρόπο επιτυγχάνεται τόσο η μείωση του χρόνου σχεδιασμού και υλοποίησης ενός συστήματος, όσο και η μεθοδολογία που ακολουθείται, αφού ο κώδικας είναι εύκολα τροποποιήσιμος, ενώ θεωρείται επαναχρησιμοποιήσιμος από άλλα συστήματα, τα οποία εμφανίζουν κοινά ή πανομοιότυπα λογικά μέρη (components). Η γνωστότερη από τις γλώσσες περιγραφής υλικού είναι η VHDL.

### **6.4. Η VHDL**

Η VHDL είναι μία γλώσσα περιγραφής υλικού για την ανάπτυξη ολοκληρωμένων ψηφιακών ηλεκτρονικών κυκλωμάτων και συστημάτων. Ως λέξη αποτελεί συντόμευση των λέξεων: VHSIC Hardware Description Language. Τα δε αρχικά

VHSIC είναι με τη σειρά τους συντόμευση για Very High-Speed Integrated Circuit (Ολοκληρωμένα Κυκλώματα Υψηλής Ταχύτητας).

Η VHDL ως γλώσσα προγραμματισμού μπορεί να χρησιμοποιηθεί για την περιγραφή της συμπεριφοράς, της δομής αλλά και της εφαρμογής ψηφιακών συστημάτων. Με βάση αυτά τα χαρακτηριστικά η VHDL χαρακτηρίζεται σαν ένα εργαλείο ECAD (Electronic Computer Aided Design).

Γενικά, σήμερα, η χρήση εργαλείων CAD έχει επεκταθεί καθώς η τεράστια ανάπτυξη της τεχνολογίας ημιαγωγών στην κατασκευή ολοκληρωμένων κυκλωμάτων έχει μετατοπίσει το κέντρο βάρους των μηχανικών από την λεπτομερειακή υλοποίηση κυκλωμάτων στην διαχείριση της αυξανόμενης πολυπλοκότητας. Πιο συγκεκριμένα τη σημερινή εποχή ο μηχανικός-σχεδιαστής, περιορίζεται περισσότερο από την δυνατότητά του να ανταπεξέλθει την πολυπλοκότητα της σχεδίασης του παρά από την ικανότητα της τεχνολογίας να την υποστηρίζει. Αυτό το χάσμα έρχεται να γεφυρώσει η VHDL επιτρέποντας μια υψηλού επιπέδου περιγραφή (Abstract) της σχεδίασης και κατόπιν με την χρήση εργαλείων σύνθεσης (Logic Synthesis Tools) την αυτόματη αποτύπωση αυτής της σχεδίασης σε ολοκληρωμένη μορφή η οποία να είναι εντός των προδιαγραφών που θέτει ο μηχανικός.

Η γλώσσα VHDL είναι η πρότυπη βιομηχανική γλώσσα περιγραφής ψηφιακών κυκλωμάτων. Μια πρώτη έκδοση του προτύπου αυτού έγινε γνωστή το 1987 με το όνομα IEEE 11076, ενώ αργότερα το 1993 εμφανίστηκε μια βελτιωμένη έκδοσή της με το όνομα IEEE 1164. Θα πρέπει να σημειώσουμε επίσης ότι η γλώσσα αυτή έχει κληρονομήσει πολλά στοιχεία από τη γλώσσα προγραμματισμού ADA. Η VHDL διακρίνεται από τα εξής χαρακτηριστικά:

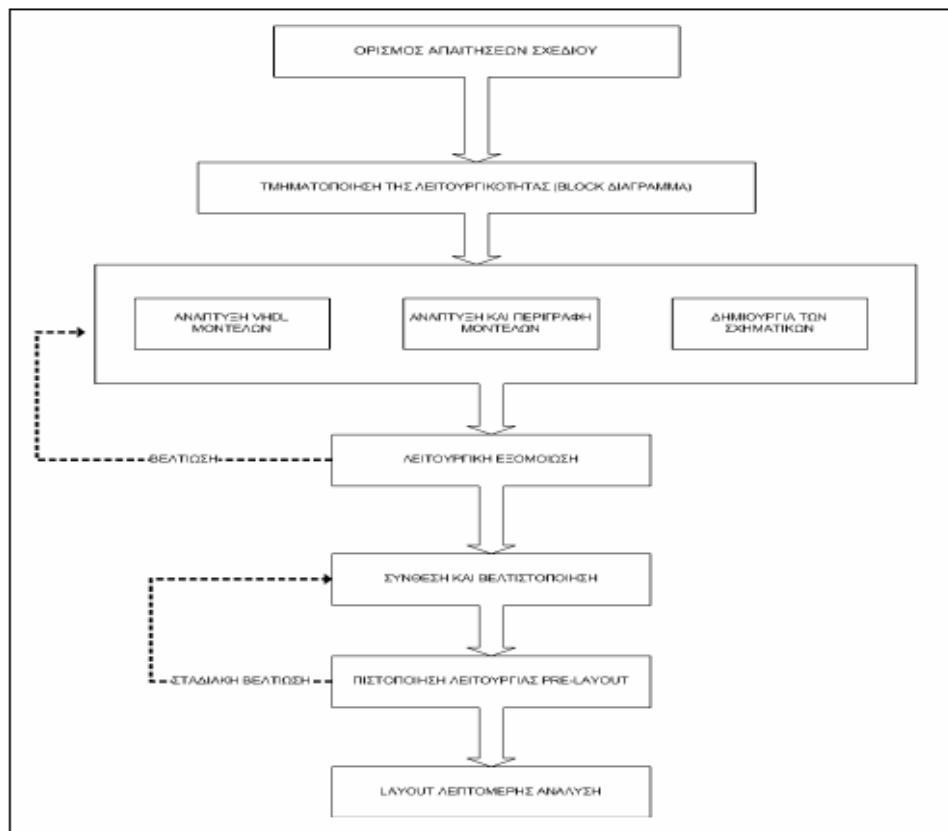
- Ταυτόχρονη δήλωση λειτουργίας των επί μέρους στοιχείων του συστήματος, αφού θα πρέπει να είναι δυνατόν όλα να λειτουργούν ταυτόχρονα, όπως σε οποιοδήποτε ψηφιακό σύστημα.
- Σειριακή δήλωση λειτουργίας, ώστε να είναι δυνατή η διαδοχική εκτέλεση διαδικασιών που ορίζονται από τις απαιτήσεις του συστήματος.
- Υποστήριξη iεραρχικής σχεδίασης.
- Υποστήριξη βιβλιοθηκών, προκειμένου να χρησιμοποιούνται μοντέλα που έχουν οριστεί από τον χρήστη και έχουν μεταφραστεί με την βοήθεια του συμβολομεταφραστή (compiler) που παρέχει η γλώσσα.
- Δήλωση τύπων, ώστε να μην περιορίζεται η VHDL μόνο σε τύπους bit ή Boolean.
- Χρήση υποπρογραμμάτων, προκειμένου να καθορίζονται εύκολα οι λειτουργίες (functions) και οι υπορουτίνες (procedures).
- Έλεγχος του χρονισμού, ώστε να είναι δυνατός ο καθορισμός της καθυστέρησης ενός σήματος ή της αναμονής για ένα σήμα ή γεγονός (event).

## 6.5. Σχεδίαση, Εξομοίωση και Σύνθεση

### 6.5.1. Μεθοδολογία Σχεδίασης

Ο έλεγχος του πηγαίου κώδικα των σχεδιάσεων, η λειτουργική εξομοίωσή τους καθώς και η σύνθεση του κώδικα πραγματοποιήθηκε με την χρήση δύο ιδιαίτερα εύχρηστων εργαλείων: το Modelsim και το Leonardo Spectrum. Το εργαλείο Modelsim αναλαμβάνει την εξομοίωση του συστήματος, ενώ το Leonardo Spectrum αναλαμβάνει την σύνθεση και βελτιστοποίησή του. Συνεπώς, ο συνδυασμός της γλώσσας VHDL με τα οφέλη των δύο εργαλείων απλοποιούν κατά πολύ την δουλειά των σχεδιαστών. Συγκεκριμένα, αρχικά, ορίζεται το σχέδιο στην VHDL. Στη συνέχεια, με την βοήθεια της εξομοίωσης, το σχέδιο εξετάζεται αν ικανοποιεί τις απαιτήσεις λειτουργικότητάς του. Το επόμενο βήμα είναι η δημιουργία του gate-level επιπέδου. Τα εργαλεία Modelsim και Leonardo Spectrum δίνουν την δυνατότητα στον σχεδιαστή να ασχοληθεί με την οργάνωση και την λειτουργικότητα του σχεδίου, αντί να σχεδιάζει σε gatelevel επίπεδο. Αυτοματοποιούν τις διεργασίες και μειώνουν το σχεδιαστικό χρόνο, αλλά πάντα υπό τον έλεγχο και τη θέληση του σχεδιαστή για όσο το δυνατόν επιθυμητή βελτιστοποίηση.

Όπως αναφέρθηκε προηγουμένως, μία από τις πιο δημοφιλείς τεχνικές σχεδίασης είναι η τεχνική top-down. Η λογική που ακολουθεί παρουσιάζεται στην επόμενη εικόνα.



Εικόνα 2: Η top-down τεχνική σχεδίασης

Σύμφωνα με την μεθοδολογία αυτή ο σχεδιασμός ξεκινά από ένα υψηλό επίπεδο περιγραφής και συνεχίζει στο αμέσως επόμενο επίπεδο, όπου είναι πιο λεπτομερής από ότι ο προηγούμενος. Η μετάβαση από το ένα επίπεδο στο άλλο συνεχίζει μέχρι και το επίπεδο που αποτελείται από τα πρωταρχικά ηλεκτρονικά στοιχεία, για την εκάστοτε χρησιμοποιούμενη τεχνολογία.

Το πρώτο στάδιο είναι ο καθορισμός των απαιτήσεων και των προδιαγραφών του σχεδίου του συστήματος (Design Specification). Ο σχεδιαστής μπορεί να χρησιμοποιήσει πολλές διαφορετικές μεθόδους για την περιγραφή του σχεδίου και την επαλήθευση της λειτουργικότητας του συστήματος σε υψηλό επίπεδο.

Το επόμενο βήμα είναι η τμηματοποίηση της λειτουργίας, όπου η λειτουργία του κυκλώματος διασπάται σε απλούστερες λειτουργίες. Στην συνέχεια, η καθεμία από τις απλούστερες λειτουργίες υλοποιείται σε VHDL. Όλες οι μονάδες που υλοποιούνται ελέγχονται ως προς την λειτουργικότητα μέσω της λειτουργικής εξομοίωσης. Η λειτουργική εξομοίωση πραγματοποιείται με το εργαλείο Modelsim, όπου ο κώδικας αρχικά ελέγχεται για λάθη και στην συνέχεια, το μοντέλο που έχει δημιουργήσει ο συμβολομεταφραστής (compiler) ελέγχεται από τον χρήστη ώστε να διαπιστωθεί αν το σύστημα έχει την αναμενόμενη συμπεριφορά. Συγκεκριμένα, ο προγραμματιστής ελέγχει αν οι τιμές των σημάτων είναι οι αναμενόμενες και αν οι αποκρίσεις των εξόδων, στις αντίστοιχες εισόδους, είναι οι προβλεπόμενες αριθμητικές τιμές και στους απαιτούμενους χρόνους.

Έπειτα, ακολουθεί η σύνθεση και η βελτιστοποίηση του σχεδίου. Σύνθεση είναι η διεργασία ανάλυσης ενός VHDL προγράμματος και της δημιουργίας ενός ψηφιακού κυκλώματος, το οποίο υλοποιεί την συμπεριφορά του μοντέλου που περιγράφεται από την VHDL. Το κύκλωμα που προκύπτει από την σύνθεση περνά από την διαδικασία της βελτιστοποίησης, όπου το κύκλωμα μπορεί να βελτιστοποιηθεί ως προς κάποιο συγκεκριμένο χαρακτηριστικό, όπως την ταχύτητα ή την επιφάνεια. Η σύνθεση και βελτιστοποίηση πραγματοποιούνται με το εργαλείο Leonardo.

Το επόμενο βήμα του σχεδιασμού είναι η πιστοποίηση λειτουργίας. Το κύκλωμα περνά από την διαδικασία της προσομοίωσης με την χρήση χρονισμών των ηλεκτρονικών στοιχείων της χρησιμοποιούμενης βιβλιοθήκης, για να εξασφαλιστεί ότι το κύκλωμα ανταποκρίνεται στις πραγματικές απαιτήσεις χρόνου, με την χρήση των απαραίτητων test vectors. Το τελευταίο στάδιο του σχεδιασμού είναι η εξέταση του παραγόμενου προϊόντος για το αν πληροί τις αρχικές προδιαγραφές.

Στις επόμενες τρεις παραγράφους, αναπτύσσονται αναλυτικότερα τα δύο εργαλεία που χρησιμοποιούνται στην σχεδίαση, καθώς και οι δύο τεχνολογίες υλοποίησης ASIC και FPGA.

### 6.5.2. Το εργαλείο Modelsim

Η υλοποίηση των επιμέρους μονάδων του συστήματος έγινε σε κώδικα VHDL και εξομοιώθηκε με το εργαλείο Modelsim 5.4e. Ο σχεδιαστής αφού ολοκληρώσει την περιγραφή του συστήματος στην VHDL, χρησιμοποιεί τον compiler που παρέχει το εργαλείο για να ελέγχει τον κώδικα που έχει γράψει. Ο compiler εντοπίζει τα συντακτικά λάθη του κώδικα.

Στην συνέχεια, το Modelsim δημιουργεί σχηματικά απλής μορφής από τον κώδικα περιγραφής VHDL, επιτρέποντας την λογική εξομοίωση του υπό ανάλυση συστήματος. Διαθέτει δυνατότητα παρακολούθησης των κυματομορφών των σημάτων που χρησιμοποιούνται στο σύστημα καθώς και των τιμών που παίρνουν σε συγκεκριμένα χρονικά διαστήματα. Δίνει επίσης την δυνατότητα χρησιμοποίησης βιβλιοθηκών από τον προγραμματιστή, καθώς επίσης και δημιουργίας νέων στοιχείων τους.

### 6.5.3. Το εργαλείο Leonardo Spectrum

Η σύνθεση και η βελτιστοποίηση του κώδικα έγινε με το εργαλείο Leonardo Spectrum version 20001b. Το συγκεκριμένο λογισμικό παρέχει την δυνατότητα να μετατρέψει τον μεταφρασμένο κώδικα VHDL σε κύκλωμα επιπέδου λογικής πύλης.

Όπως έχει ήδη αναφερθεί, σύνθεση είναι η διεργασία ανάλυσης ενός VHDL προγράμματος και δημιουργίας ενός ψηφιακού κυκλώματος το οποίο υλοποιεί την συμπεριφορά του μοντέλου που περιγράφεται από την VHDL. Το κύκλωμα κατασκευάζεται χρησιμοποιώντας ένα σταθερό σύνολο πρωταρχικών δομών υλικού. Ο ρόλος του μεταγλωττιστή είναι να βρει την ιδανικότερη υλοποίηση που καθορίζεται από την περιγραφή στην VHDL. Τα υλοποιημένα κυκλώματα που θα προκύψουν εξαρτώνται από τον χρησιμοποιούμενο μεταγλωττιστή σύνθεσης.

Η διαδικασία της σύνθεσης είναι πλήρως αυτοματοποιημένη και το μόνο που απαιτείται από τον σχεδιαστή είναι να ορίσει την αρχιτεκτονική του στοιχείου. Επίσης, πρέπει να οριστεί και η βιβλιοθήκη των πρωτογενών ηλεκτρονικών στοιχείων υλοποίησης.

Αν στο σύστημα υπάρχει ιεραρχία, οι κατώτερες ιεραρχικά οντότητες μπορεί να περάσουν από σύνθεση πρώτες και στην συνέχεια, τα αποτελέσματά τους να χρησιμοποιηθούν στην σύνθεση των ανώτερων ιεραρχικά οντοτήτων, χωρίς να επέμβει σε αυτές το εργαλείο σύνθεσης.

Από την σύνθεση προκύπτουν το netlist, δηλαδή η περιγραφή του κυκλώματος σε επίπεδο-πύλης, και το σχηματικό του συστήματος. Στην συνέχεια, το κύκλωμα που προκύπτει από την σύνθεση περνά από την διαδικασία της βελτιστοποίησης, όπου πρέπει να επιλεχθεί η τεχνολογία υλοποίησης. Στην προκειμένη περίπτωση επιλέχθηκε η αρχιτεκτονική FPGA Virtex II της Xilinx και συγκεκριμένα το device 2V1000bf957. Η βελτιστοποίηση μπορεί να γίνει ως προς την ταχύτητα και την επιφάνεια. Οι δύο έννοιες είναι αντικρουόμενες, ωστόσο το εργαλείο καταλήγει στην βέλτιστη δυνατή λύση.

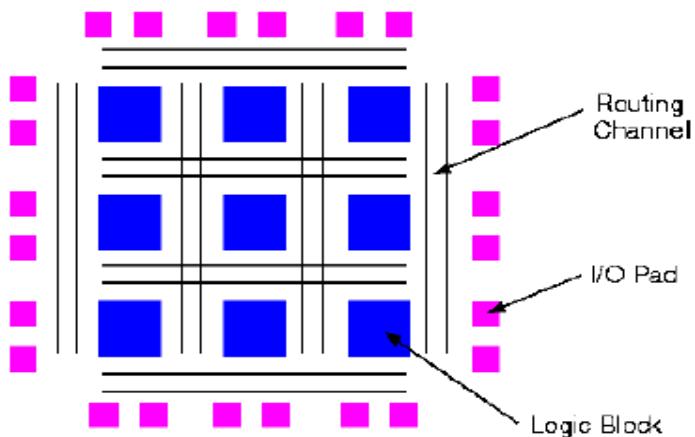
Από τη διαδικασία της σύνθεσης και της βελτιστοποίησης, προκύπτουν διάφορα φυσικά χαρακτηριστικά του υλοποιημένου κυκλώματος, όπως η ταχύτητα και η επιφάνεια.

#### 6.5.4. Επιλογή Τεχνολογίας

Κάθε σύστημα πρέπει τελικά να υλοποιηθεί σε ένα ολοκληρωμένο κύκλωμα (integrated circuit). Η τεχνολογία ολοκληρωμένων κυκλωμάτων αναφέρεται στον τρόπο κατά τον οποίο απεικονίζεται μία ψηφιακή υλοποίηση επιπέδου πύλης σε ένα ολοκληρωμένο κύκλωμα. Ένα ολοκληρωμένο κύκλωμα είναι μία διάταξη ημιαγωγών που αποτελείται από ένα σύνολο στοιχειωδών διατάξεων (π.χ. transistors). Υπάρχει ένας αριθμός από διαφορετικές διεργασίες για την κατασκευή ημιαγωγών, η πιο δημοφιλής από τις οποίες είναι η CMOS (Complementary Metal Oxide Semiconductor).

Η υλοποίηση σε ολοκληρωμένο κύκλωμα, δηλαδή μία hardware υλοποίηση, γράφεται και σχεδιάζεται αρχικά σε μία γλώσσα περιγραφής υλικού, όπως η VHDL και η Verilog. Στην συνέχεια, το πρόγραμμα χρησιμοποιείται για να προγραμματίσει υλικό, και διακρίνονται δύο βασικές τεχνολογίες υλοποίησης, τα Application Specific Integrated Circuits (ASICs) και Field Programmable Gate Arrays (FPGAs).

Τα ASICs σχεδιάζονται εξ' ολοκλήρου από την αρχή, δηλαδή από το στάδιο περιγραφής της συμπεριφοράς μέχρι και τον φυσικό σχεδιασμό, και κατόπιν στέλνονται για μία ακριβή και χρονοβόρα κατασκευή. Τα ASICs σχεδιάζονται συγκεκριμένα για να εκτελούν έναν δεδομένο μετασχηματισμό και έτσι είναι πολύ γρήγορα και αποδοτικά κατά την εκτέλεση του ακριβούς υπολογισμού για τα οποία σχεδιάστηκαν. Όμως, τα κυκλώματα δεν μπορούν να αλλάξουν μετά από την επεξεργασία. Αυτό σημαίνει ότι επανασχεδιασμός και επανακατασκευή του τσιπ ή μέρος αυτού απαιτεί την τροποποίηση του κυκλώματος, και έτσι προκύπτει μία χρονοβόρος διαδικασία. Τα FPGA αποτελούν σύνθετη κατηγορία της τεχνολογίας PLD (Programmable Logic Device). Όπως, όλα τα ολοκληρωμένα κυκλώματα τεχνολογίας PLD, τα FPGA μπορούν να αγοραστούν μόνα τους και να επαναπρογραμματιστούν από τους σχεδιαστές. Ο προγραμματισμός που λαμβάνει χώρα μπορεί να πραγματοποιείται μέσω της δημιουργίας ή της καταστροφής συνδέσεων ανάμεσα σε καλώδια που συνδέουν πύλες, είτε καίγοντας μία ασφάλεια είτε θέτοντας ένα bit σε ένα προγραμματιζόμενο διακόπτη. Μικρές συσκευές, που ονομάζονται προγραμματιστές, συνδεδεμένες σε ένα προσωπικό υπολογιστή μπορούν να επιτελέσουν αυτόν τον προγραμματισμό. Η δομή ενός FPGA φαίνεται στο επόμενο σχήμα.



Εικόνα 3: Δομή FPGA

Τα FPGA προσφέρουν πολύ χαμηλό κόστος σχεδίασης και σχεδόν άμεση διαθεσιμότητα ολοκληρωμένων κυκλωμάτων. Είναι, ωστόσο, μεγαλύτερα συνήθως από τα ASICs, μπορεί να έχουν υψηλότερο κόστος ανά μονάδα, να καταναλώνουν περισσότερη ισχύ και να είναι πιο αργά. Παρέχουν, παρόλα αυτά, πολύ ικανοποιητική απόδοση και ενδείκνυνται για ταχύτατη κατασκευή πρωτότυπου.

Η αναδιατασσόμενη λογική (FPGA) εφευρέθηκε προς τα τέλη της δεκαετίας του 1980 σαν φυσική εξέλιξη των προγραμματιζόμενων λογικών συσκευών (PLD's – Programmable Logic Devices). Παρότι παρουσιάστηκαν διάφορες δομές, ειδικά στα πρώτα χρόνια της εμφάνισης της αναδιατασσόμενης λογικής, η βασική δομή που ίσχυε από τότε μέχρι και τώρα αποτελείται από ένα πλέγμα λογικών πυλών που υλοποιούν συναρτήσεις με την μέθοδο των πινάκων (LUT – Look Up Tables), ενδεικτικού μεγέθους 5 μεταβλητών εισόδου/1 μεταβλητής εξόδου ή 4 μεταβλητών εισόδου/2 μεταβλητών εξόδου (δηλαδή μνήμη 32 Bits), με flip-flop σε κάθε έξοδο το οποίο μπορεί να παρακαμφθεί για συνδυαστική λογική ή να χρησιμοποιηθεί για ακολουθιακή λογική. Οι λογικές αυτές πύλες έχουν διάφορα ονόματα όπως π.χ. CLB (Configurable Logic Blocks) και περιβάλλονται από προγραμματιζόμενα σύρματα για την διασύνδεσή τους. Σαν αποτέλεσμα ο συνολικός χρόνος που χρειάζεται για να γίνει μία λογική συνάρτηση δεν είναι σταθερός όπως στις PLD αλλά εξαρτάται από τον αριθμό των λογικών κυκλωμάτων και των διασυνδέσεων που περιλαμβάνονται στο critical path.

Αντίθετα με τους γενικής χρήσης επεξεργαστές που εκτελούν εντολές σειριακά ή με μικρό παραλληλισμό, σε μία FPGA μπορούν να λειτουργούν παράλληλα πολλές διεργασίες απ' ευθείας στο υλικό. Επίσης, το datapath σε μία FPGA μπορεί να είναι ανάλογο με την εφαρμογή, π.χ. 293 bits. Ο τρόπος σχεδίασης με FPGA διαφέρει πολύ από τον τρόπο σχεδίασης υπολογιστών γενικής χρήσης λόγω διαφορετικών τεχνολογικών περιορισμών και ευκαιριών, αλλά οι συνηθισμένες τεχνικές που εφαρμόζονται στην αρχιτεκτονική υπολογιστών (π.χ. pipelining) έχουν άμεση χρήση και σε σχεδίαση με FPGA.

Ήδη από τις αρχές της δεκαετίας του 1990 οι σχεδιαστές διαπίστωσαν ότι μπορούσαν να απεικονίσουν αλγορίθμους απ' ευθείας σε FPGA και αυτό έδωσε ώθηση στην περιοχή εφαρμογών FCCM (Field-programmable Custom Computing Machines) που εξελίχθηκε ραγδαία. Κάποιες από τις πρώτες εφαρμογές ήταν η σύγκριση συμβολοσειρών για DNA sequencing, η εξεύρεση κανόνων Golomb, υλοποίηση νευρωνικών δικτύων, κλπ., η δε λίστα συνεχίζεται μέχρι και σήμερα με εφαρμογές επεξεργαστών δικτύων (network processors), επεξεργαστές για κρυπτογραφία, εφαρμογές σε ιδεατή πραγματικότητα, κλπ.

Από τα τέλη της δεκαετίας του 1990 και τις αρχές της δεκαετίας του 2000 αρχίζουν δύο σημαντικές αλλαγές στον τρόπο με τον οποίο κατασκευάζονται και χρησιμοποιούνται οι FPGA's:

- Οι FPGA's αρχίζουν και έχουν ενσωματωμένο επεξεργαστή, καταρχήν μικρής υπολογιστικής ισχύος.
- Οι FPGA's άρχισαν να εκτοπίζουν σε εφαρμογές τους επεξεργαστές σήματος DSP για εφαρμογές σταθερής υποδιαστολής (ενώ οι πράξεις κινητής υποδιαστολής παρέμειναν η αχύλειος πτέρνα των FPGA).



# Κεφάλαιο 7

## Simulation and Synthesis Results

### 7.1 Εξομοίωση της GCTR Δομής του AES-GCM

Όσα περιγράφηκαν προηγουμένως εξάγονται με την βοήθεια του εργαλείου ModelSim SE/EE Plus 5.4e. Οι προκύπτουσες εξομοιώσεις επιβεβαιώνουν την ορθή λειτουργία των παραπάνω μονάδων και παρουσιάζονται στις επόμενες εικόνες. Τα αποτέλεσμα που προκύπτουν είναι αναγκαία για να χαρακτηριστεί η λειτουργία του κυκλώματος σωστή χωρίς όμως αυτό να σημάνει ότι ο κώδικας είναι και λειτουργικός. Για να επιβεβαιωθεί ότι ο VHDL κώδικας αναπαριστά την σωστή hardware αρχιτεκτονική πρέπει να ολοκληρωθεί και το στάδιο της σύνδεσης επιτυχώς.

Θα παρουσιάσουμε την εξομοίωση δύο διαφορετικών test vector που δίνονται από το NIST. Τα δύο αυτά test vector θα τα εξομοιώσουμε διαδοχικά. Και στις δύο περιπτώσεις θα χρησιμοποιηθεί το ίδιο κλειδί, ο ίδιος IV και τα ίδια τρία πρώτα πακέτα εισόδου plaintext. Η διαφορά βρίσκεται στο τέταρτο και τελευταίο πακέτο plaintext όπου στην πρώτη περίπτωση αποτελεί ένα partial pack ενώ στην δεύτερη περίπτωση είναι ένα full pack των 128 bits.

Να υπενθυμίσουμε ότι ο AES-GCM αλγόριθμος που παρουσιάζεται στην διπλωματική αυτή αποτελείται από "online" GCM λειτουργίες, καθώς τα μήκη των confidential data και των επιπρόσθετων non-confidential data, δηλαδή τα μήκη του Plaintext και των AAD αντίστοιχα, δεν είναι προαπαιτούμενα αλλά υπολογίζονται κατά την άφιξη των πακέτων και επεξεργάζονται. Για τον λόγο αυτό, είναι πολύ σημαντικό να παρουσιάσουμε τα δύο test vector που αναφέραμε προηγούμενα καθώς μέσα από αυτά γίνεται ξεκάθαρο ότι ο AES-GCM αλγόριθμος είναι όντως "online" και αναγνωρίζει αν το τελικό πακέτο Plaintext που θα λάβει αποτελεί ένα full pack των 128 bits ή ένα partial pack, χωρίς να υπάρχει η ανάγκη να έχει ειδοποιηθεί εκ των πρότερων.

Τα δύο test vector δίνονται παρακάτω στην πλήρη τους μορφή, καθώς απευθύνονται στον AES-GCM στο σύνολό του. Βέβαια, στην παρούσα ενότητα θα παρουσιαστεί η εξομοίωση του GCTR module στο οποίο στηρίζεται η διαδικασία της κρυπτογράφησης και αποκρυπτογράφησης και όχι η διαδικασία της πιστοποίησης (authentication).

Η εξομοίωση ολοκλήρου του AES-GCM αλγόριθμου θα παρουσιαστεί σε επόμενη ενότητα στο παρών κεφάλαιο. Σημαντικό είναι να αναφερθεί ότι παρόλο που τα δύο συγκεκριμένα test vectors διαφέρουν μονάχα στο τελευταίο πακέτο εισόδου, Plaintext, και συγκεκριμένα μόνο κατά κάποια bytes, οι επικέτες γνησιότητας που προκύπτουν είναι τελείως διαφορετικές, και αυτή ακριβώς είναι και η δύναμη του επιπέδου ασφαλείας του AES-GCM, καθώς το authentication tag υπογράφει μοναδικά τα προς κρυπτογράφηση δεδομένα.

## Test vectors – NIST

**Example #5 :** Lengths NOT prerequisites

---

### **Encrypt-Generate**

**K** is FFFE992 8665731C 6D6A8F94 67308308

**IV** is CAFEBABE FACEDBAD DECAF888

**A** is 3AD77BB4 0D7A3660 A89ECAF3 2466EF97  
F5D3D585

**P** is

D9313225 F88406E5 A55909C5 AFF5269A  
86A7A953 1534F7DA 2E4C303D 8A318A72  
1C3C0C95 95680953 2FCF0E24 49A6B525  
B16AEDF5 AA0DE657 BA637B39

### **Result**

**Cipher(K, J0)** is 3247184B 3C4F69A4 4DBCD228 87BBB418

**C** is

42831EC2 21777424 4B7221B7 84D0D49C  
E3AA212F 2C02A4E0 35C17E23 29ACA12E  
21D514B2 5466931C 7D8F6A5A AC84AA05  
1BA30B39 6A0AAC97 3D58E091

**Tag** is

F07C2528 EEA2FCA1 211F905E 1B6A881B

---

**Example #2:** Lengths NOT prerequisites

---

### **Encrypt-Generate**

**K** is FFFE992 8665731C 6D6A8F94 67308308

**IV** is CAFEBABE FACEDBAD DECAF888

**P** is

D9313225 F88406E5 A55909C5 AFF5269A  
86A7A953 1534F7DA 2E4C303D 8A318A72  
1C3C0C95 95680953 2FCF0E24 49A6B525  
B16AEDF5 AA0DE657 BA637B39 1AAFD255

### **Result**

**Cipher (K, J0)** is 3247184B 3C4F69A4 4DBCD228 87BBB418

**C** is

42831EC2 21777424 4B7221B7 84D0D49C  
E3AA212F 2C02A4E0 35C17E23 29ACA12E  
21D514B2 5466931C 7D8F6A5A AC84AA05  
1BA30B39 6A0AAC97 3D58E091 473F5985

**Tag** is 4D5C2AF3 27CD64A6 2CF35ABD 2BA6FAB4

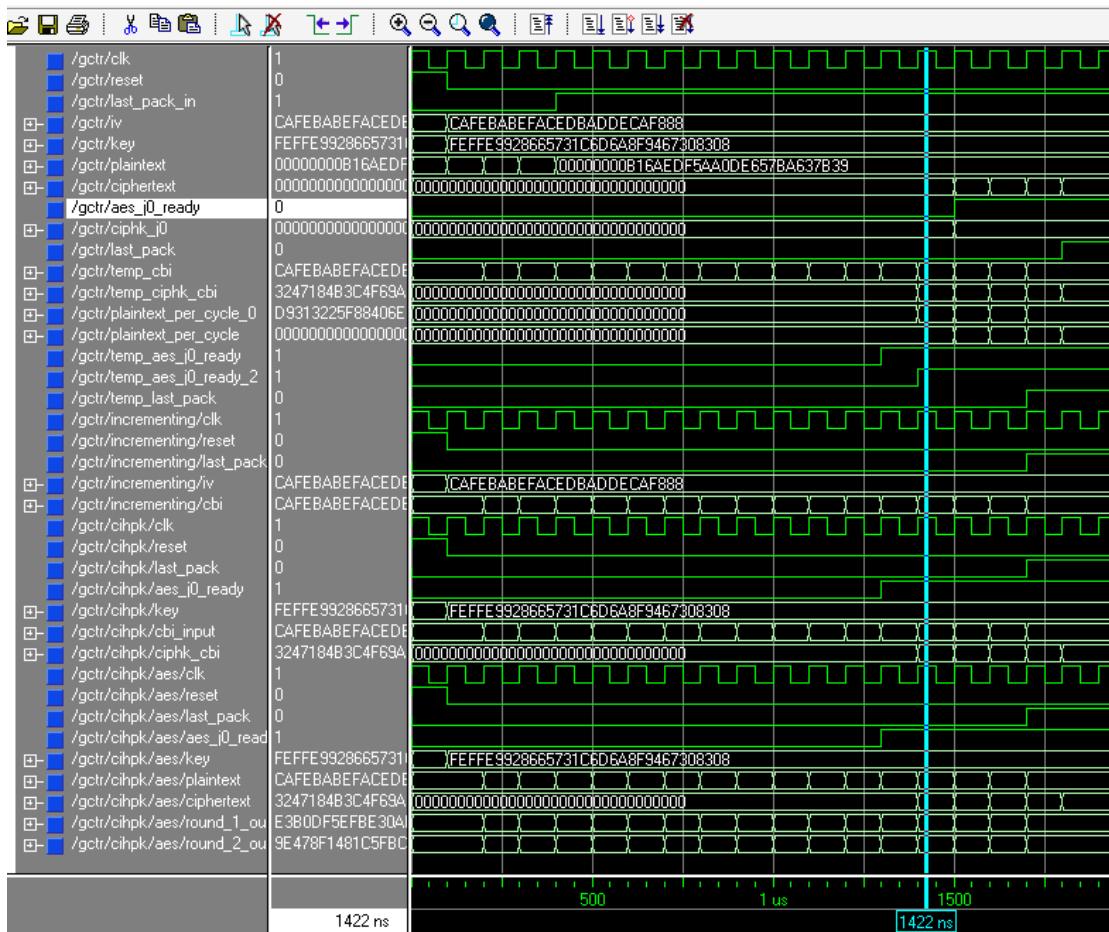
### 7.1.1 Κρυπτογράφηση – Εξομοίωση και Χρονισμοί

Αρχικά, στο reset το σύστημα είναι απενεργοποιημένο και όλα τα σήματα εξόδου καθώς και τα ενδιάμεσα σήματα που αποθηκεύονται τις προσωρινές τιμές, είναι στο μηδέν.

Όταν το σήμα reset απενεργοποιηθεί, οι είσοδοι του συστήματος τοποθετούνται στο κύκλωμα. Ο IV (96 bits) και το Key (128 bits) δίνονται μία φορά ενώ τα προς κρυπτογράφηση δεδομένα, Plaintext, δίνονται σε μπλοκ των 128 bits. Οι δύο βασικές έξοδοι, ciphk\_j0 και ciphertext των 128 bits, παραμένουν στο μηδέν μέχρι να ενεργοποιηθεί το σήμα ελέγχου aes\_j0\_ready το οποίο ουσιαστικά σηματοδοτεί την χρονική στιγμή που έχει γεμίσει πλήρως το pipeline του AES και το κύκλωμα είναι πλέον σε θέση να δίνει στην έξοδο ένα μπλοκ 128 bits κρυπτογραφημένων πλέον δεδομένων σε κάθε κύκλο ρολογιού. Οι ενδιάμεσοι καταχωρητές, που έχουν τις προσωρινές τιμές, αλλάζουν σε κάθε κύκλο, αφού σε κάθε κύκλο γίνεται επεξεργασία ενός νέου 128-bit πακέτου, μέχρις ότου να κλειδωθούν τα βασικά components του κυκλώματος στις τελικές τιμές τους.

Στο παρακάτω σχήμα φαίνεται αναλυτικά η πορεία και η ροή του κυκλώματος όπου μέχρι και τα 1499 ns το σήμα ελέγχου aes\_j0\_ready παραμένει απενεργοποιημένο.

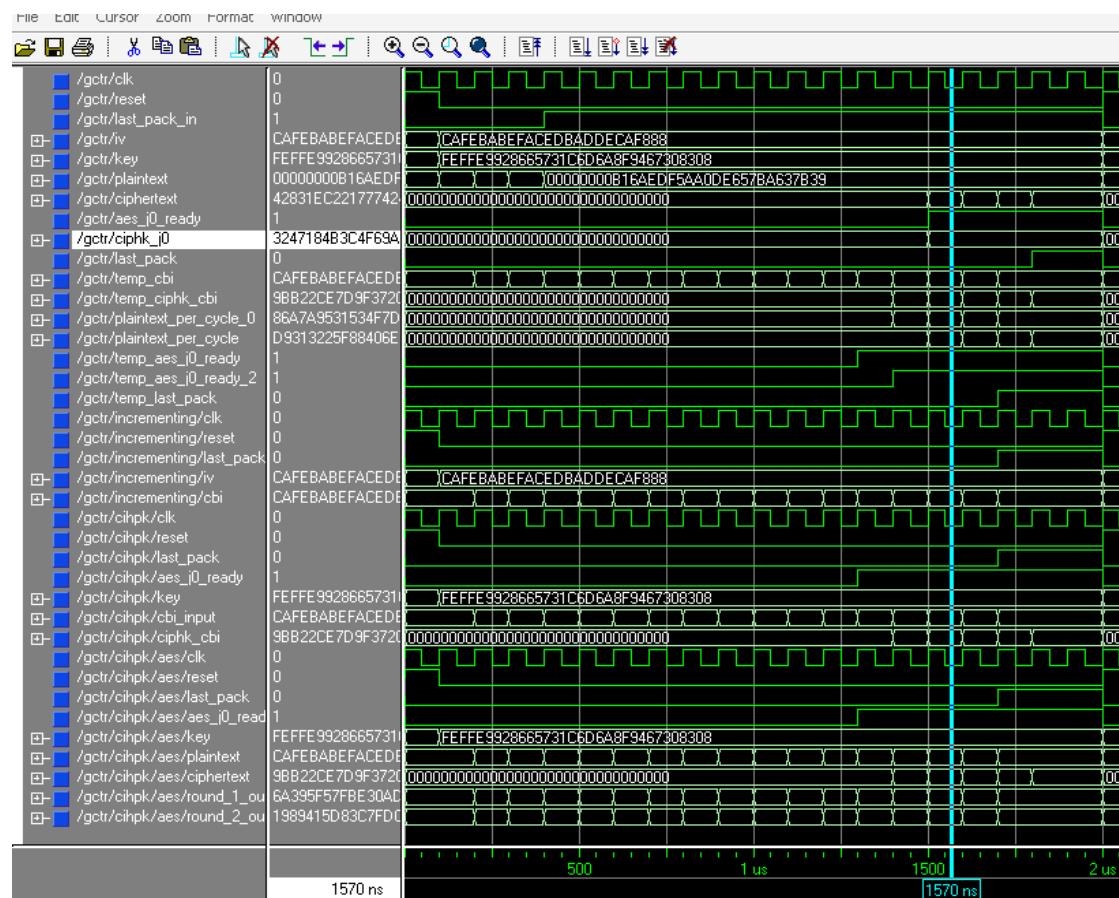
### Αρχή Λειτουργίας



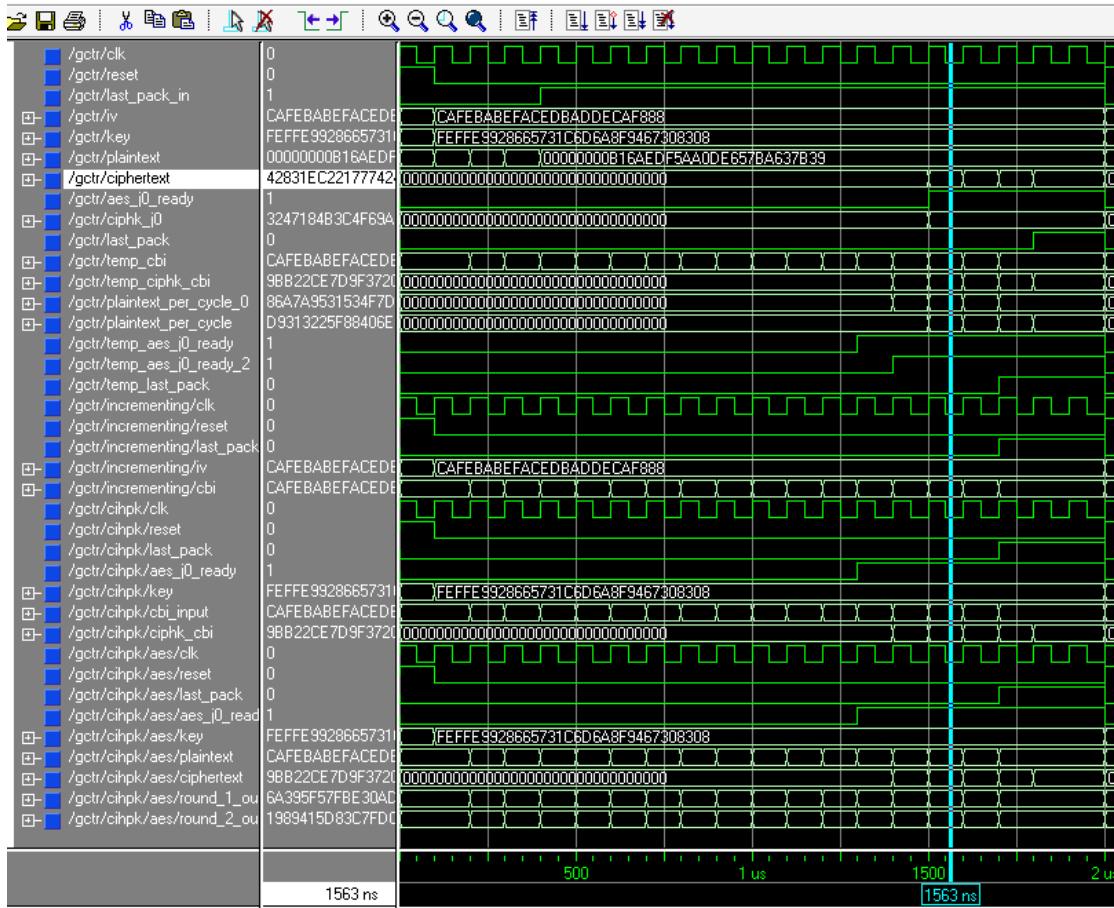
Στα 1500 ns, δηλαδή στον 14<sup>ο</sup> κύκλο λειτουργίας του κυκλώματος μετά το reset, το σήμα ελέγχου aes\_j0\_ready ενεργοποιείται καθώς έχει γεμίσει πλήρως το pipeline του AES και θα παίρνουμε έξοδο σε κάθε κύκλο μέχρις ότου ενεργοποιηθεί το σήμα last\_pack. Το σήμα last\_pack ειδοποιεί ότι και το τελευταίο πακέτο εισόδου έχει κρυπτογραφηθεί, ότι δηλαδή είναι έτοιμο και το τελευταίο πακέτο ciphertext, επομένως κλειδώνονται τα επιμέρους κυκλώματα και διακόπτεται η λειτουργία τους ώστε να μην έχουμε άσκοπη κατανάλωση ενέργειας εφόσον τα επιθυμητά αποτελέσματα έχουν προκύψει.

Να τονιστεί εδώ ότι το σήμα ελέγχου **aes\_j0\_ready ειδοποιεί ότι η έξοδος ciphk\_j0 είναι διαθέσιμη** και μπορεί να τροφοδοτηθεί στο GHASH module για την διαδικασία του authentication. Βέβαια, να σημειώσουμε ότι η έξοδος ciphk\_j0 είναι έτοιμη στον 13<sup>ο</sup> κύκλο καθώς αποτελεί την τελευταία τιμή με την οποία γεμίζει το pipeline του AES ενώ στον 14<sup>ο</sup> κύκλο η τιμή αυτή αποθηκεύεται στον register εξόδου.

### Τελικό Αποτέλεσμα ciphk\_j0 :

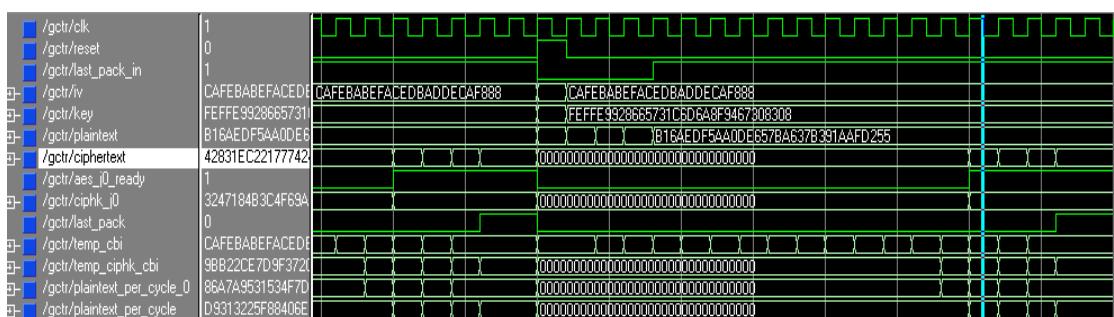


## Ciphertext\_1, example\_5:

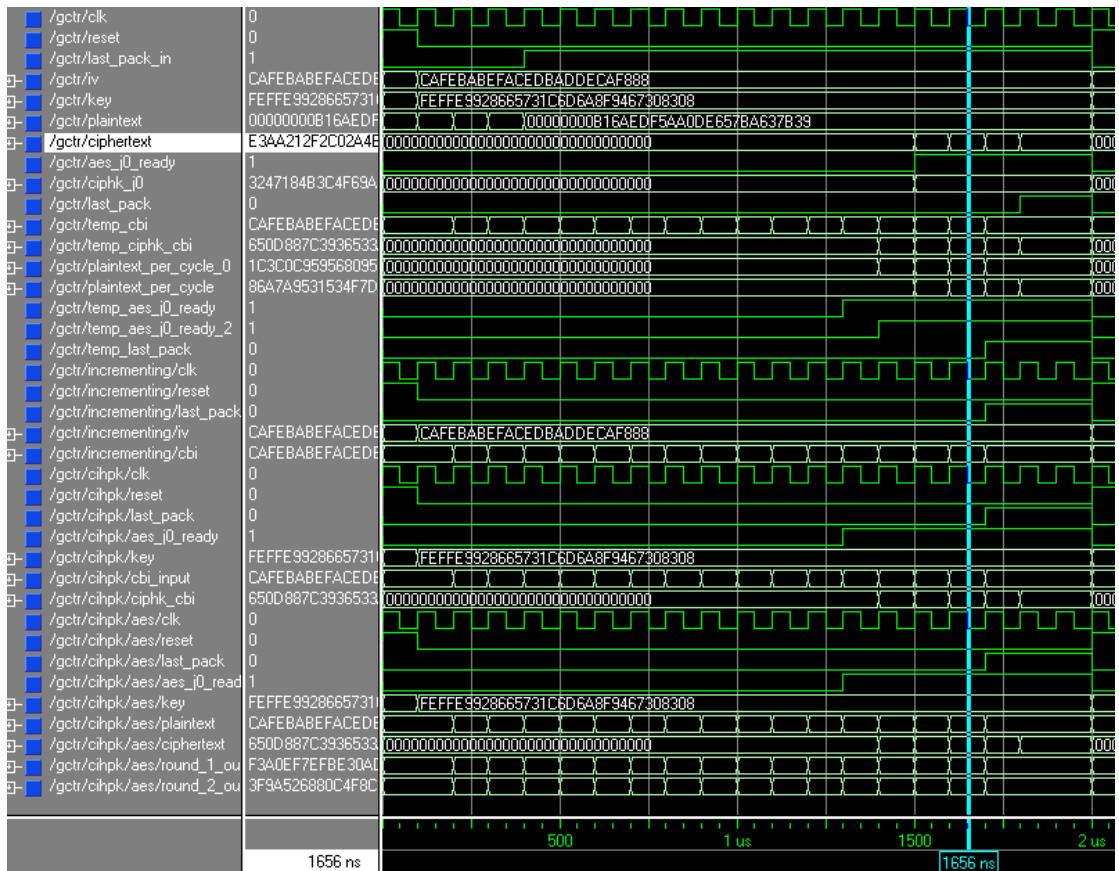


Για τα δύο διαφορετικά test vectors βλέπουμε ότι το πρώτο ciphertext δίνεται στην έξοδο στον 14<sup>ο</sup> κύκλο μετά το reset, δηλαδή στα 1500 ns για το πρώτο test vector στο πάνω σχήμα, και στα 3500 ns για το δεύτερο test vector, στο κάτω σχήμα, εφόσον έχουμε 4 πακέτα εξόδου κάθε φορά και έχουμε εξομοιώσει το κύκλωμά μας να τρέχει διαδοχικά (με 1 κύκλο hold πριν το επόμενο reset το οποίο συμβαίνει στα 2000 ns).

## Ciphertext\_1, example\_2:

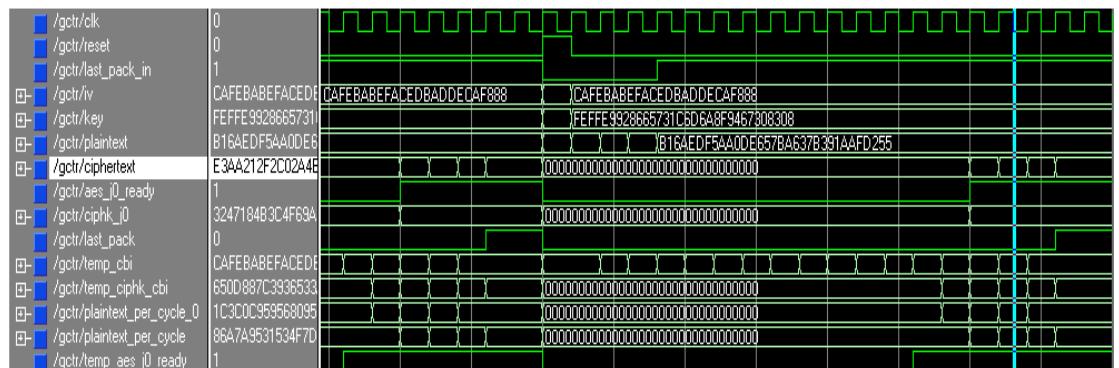


## Ciphertext\_2, example\_5:

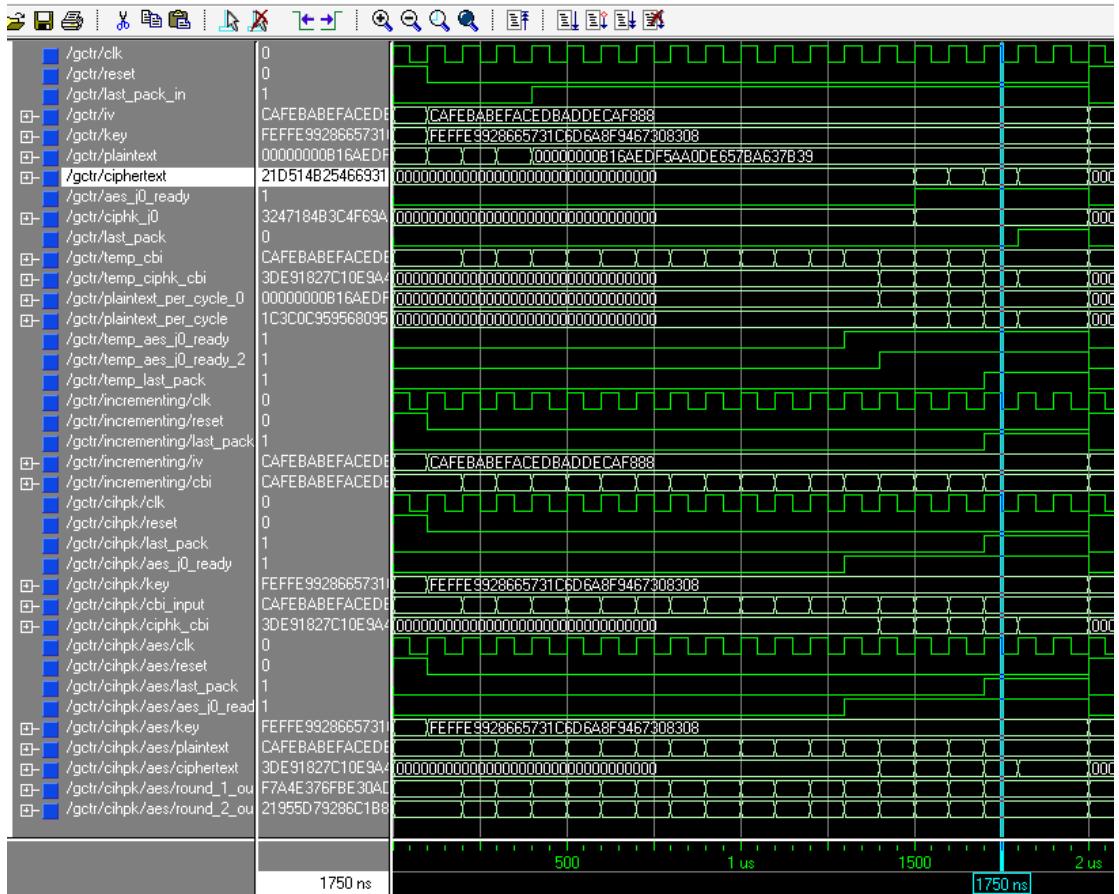


Παρόμοια με τα δύο προηγούμενα σχήματα, εδώ φαίνεται το δεύτερο ciphertext που δίνεται στην έξοδο στον 15<sup>ο</sup> κύκλο λειτουργίας του συστήματος μετά το reset, δηλαδή στα 1600 ns και 3600 ns αντίστοιχα για τα δύο παραδείγματα.

## Ciphertext\_2, example\_2:



### Ciphertext\_3, example\_5:

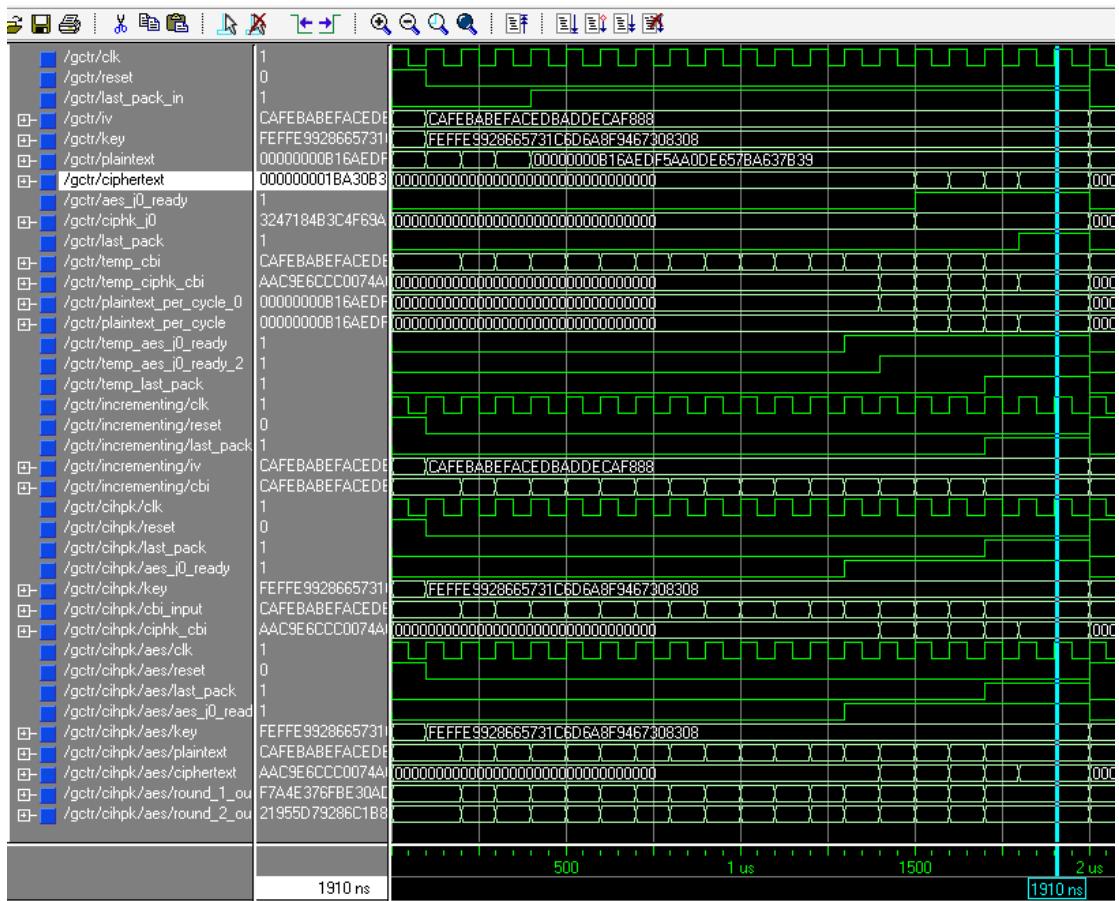


Παρόμοια με τα προηγούμενα σχήματα, εδώ φαίνεται το τρίτο ciphertext που δίνεται στην έξοδο στον 16<sup>ο</sup> κύκλο, δηλαδή στα 1700 ns και 3700 ns αντίστοιχα.

### Ciphertext\_3, example\_2:



### Ciphertext\_4\_partial\_pack, example\_5:



Παρόμοια με τα προηγούμενα σχήματα, στα δύο αυτά, φαίνεται το τέταρτο και τελευταίο ciphertext που δίνεται στην έξοδο στον 17<sup>ο</sup> κύκλο, δηλαδή στα 1800 ns και 3800 ns αντίστοιχα για τα δύο διαφορετικά test vectors.

Εδώ βέβαια να τονίσουμε ότι το σήμα ελέγχου *last\_pack* ενεργοποιείται τη στιγμή που το τελευταίο πακέτο εξόδου δίνεται στην έξοδο και έτσι το κύκλωμα κλειδώνεται χωρίς να συμβαίνει πλέον κάποια άλλη αλλαγή.

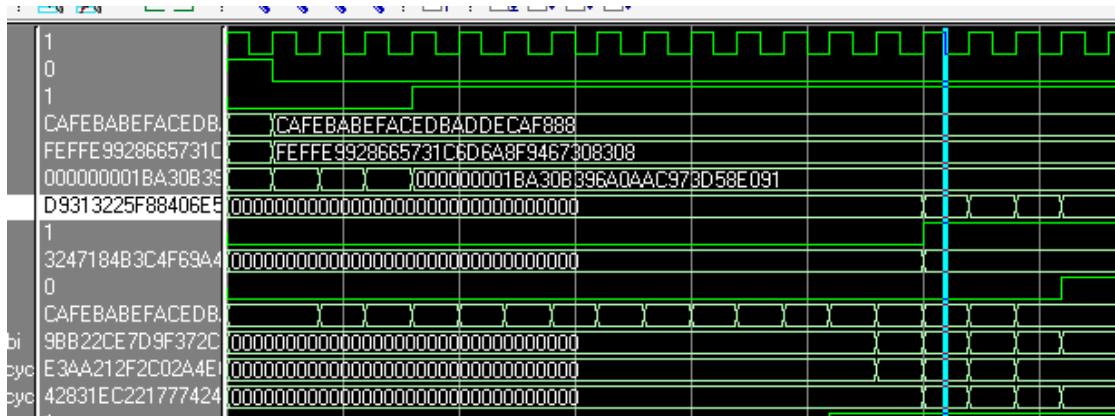
### Ciphertext\_4\_full pack, example\_2:



### 7.1.2 Αποκρυπτογράφηση – Εξομοίωση και Χρονισμοί

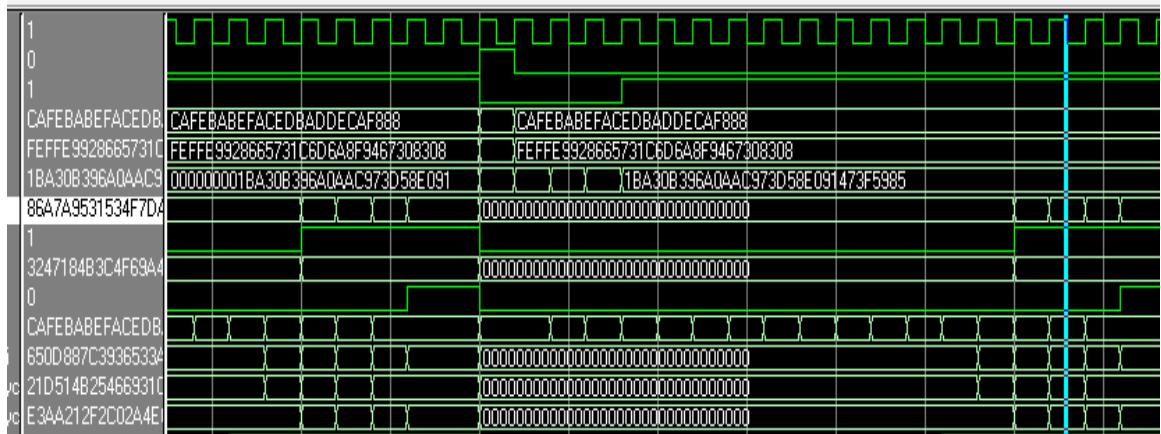
Τώρα μας μένει να επιβεβαιώσουμε το γεγονός ότι το ίδιο κύκλωμα χρησιμοποιείται και για την διαδικασία της αποκρυπτογράφησης χωρίς να υπάρχει ανάγκη κάποιου επιπλέον κυκλώματος, καθώς αρκεί μόνο να τροφοδοτηθούν στην είσοδο τα Ciphertexts που θέλουμε να αποκρυπτογράψουμε και φυσικά το ίδιο μυστικό κλειδί και IV.

#### Plaintext\_1, example\_5:

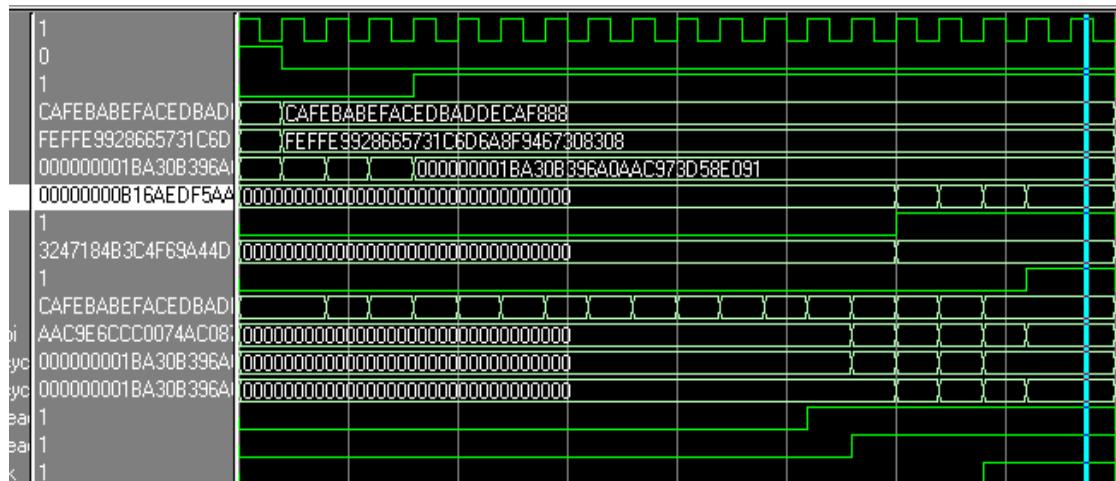


Ισχύουν ακριβώς τα ίδια που αναλύσαμε προηγούμενα σχετικά με τα θέματα χρονισμού. Οπότε το πρώτο *plaintext* δίνεται στην έξοδο στον 14<sup>ο</sup> κύκλο μετά το reset, δηλαδή στα 1500 ns για το πρώτο test vector στο πάνω σχήμα, ενώ το δεύτερο *plaintext* δίνεται στην έξοδο στον 15<sup>ο</sup> κύκλο, στα 3600 ns για το δεύτερο test vector.

#### Plaintext\_2, example\_2:

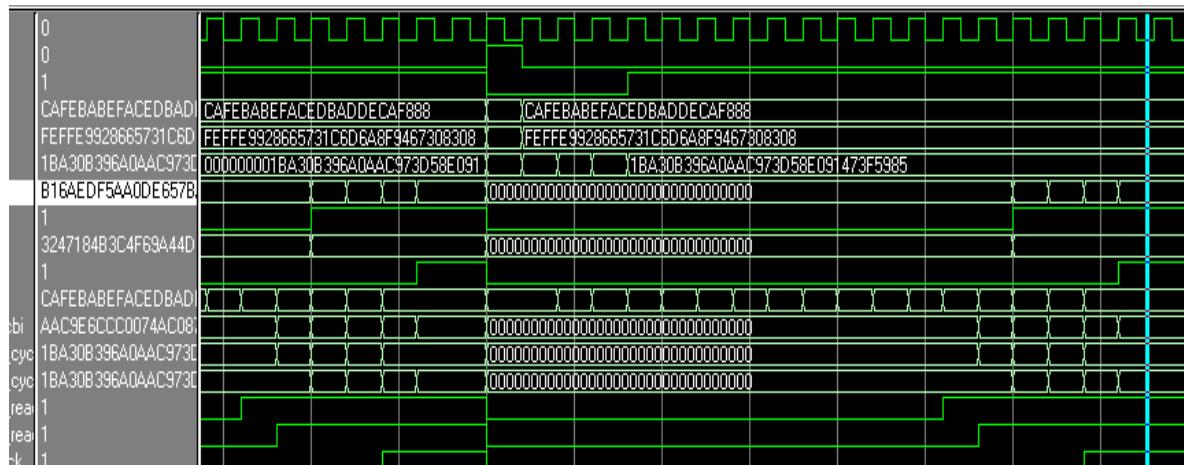


### Plaintext\_4\_partial\_pack, example\_5:



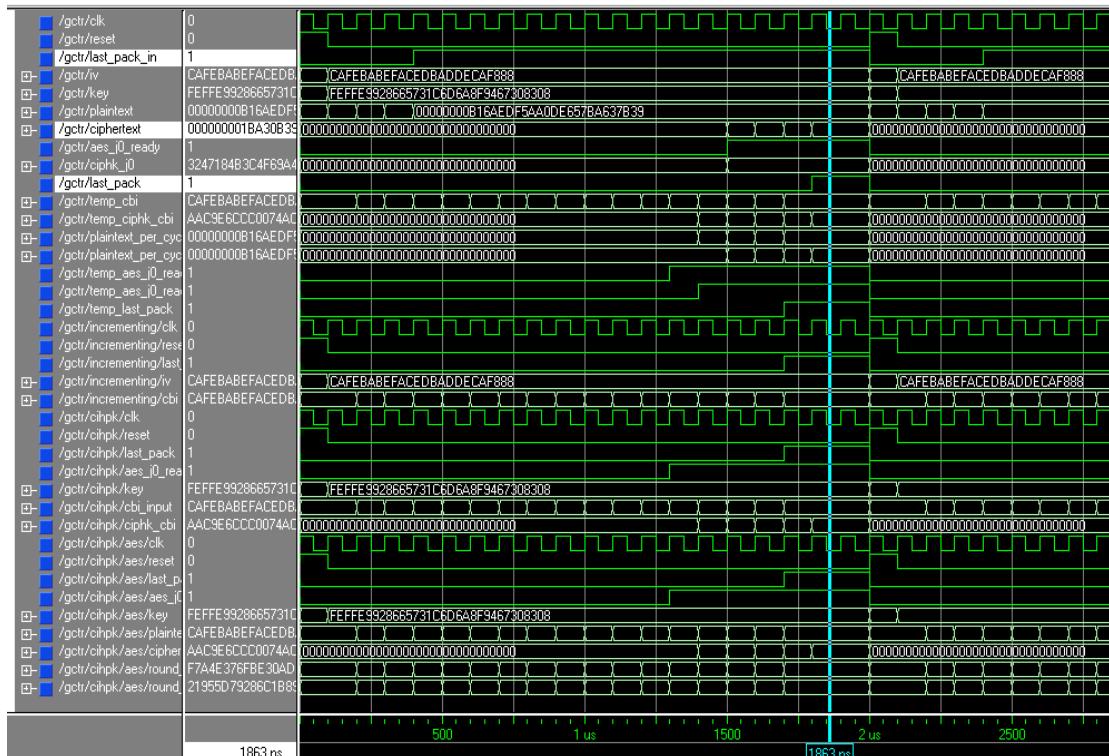
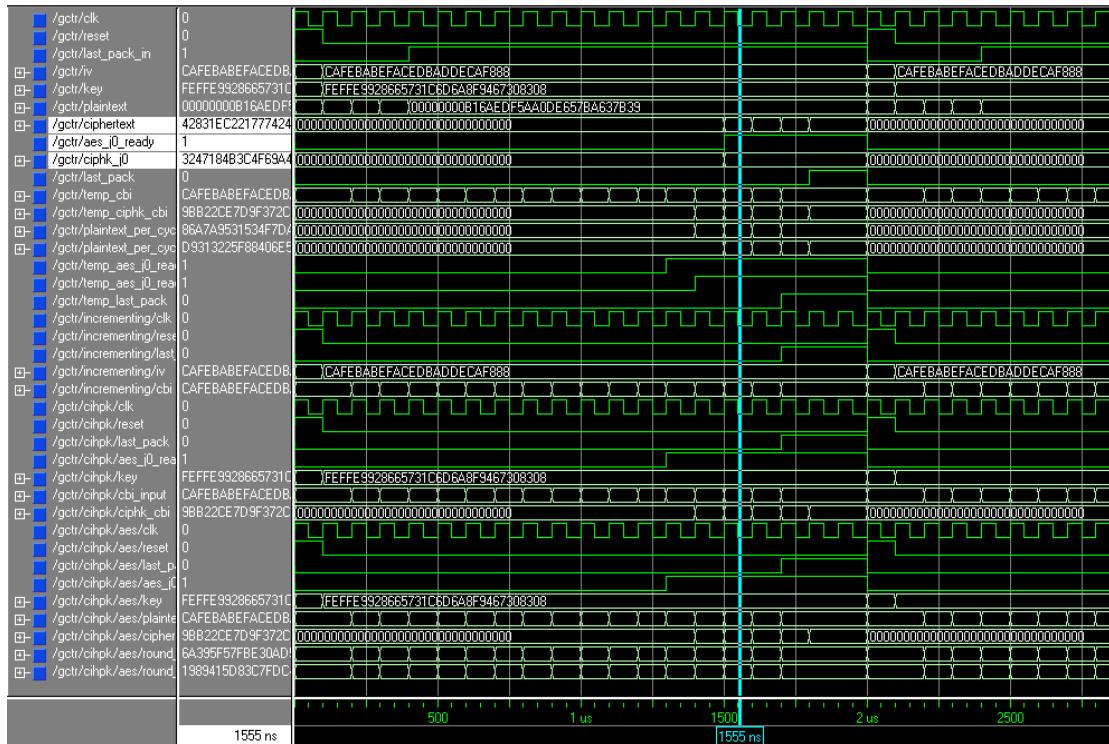
Παρόμοια, το τέταρτο και τελευταίο *plaintext* δίνεται στην έξοδο στον 17<sup>ο</sup> κύκλο, δηλαδή στα 1800 ns και 3800 ns αντίστοιχα για τα δύο διαφορετικά test vectors.

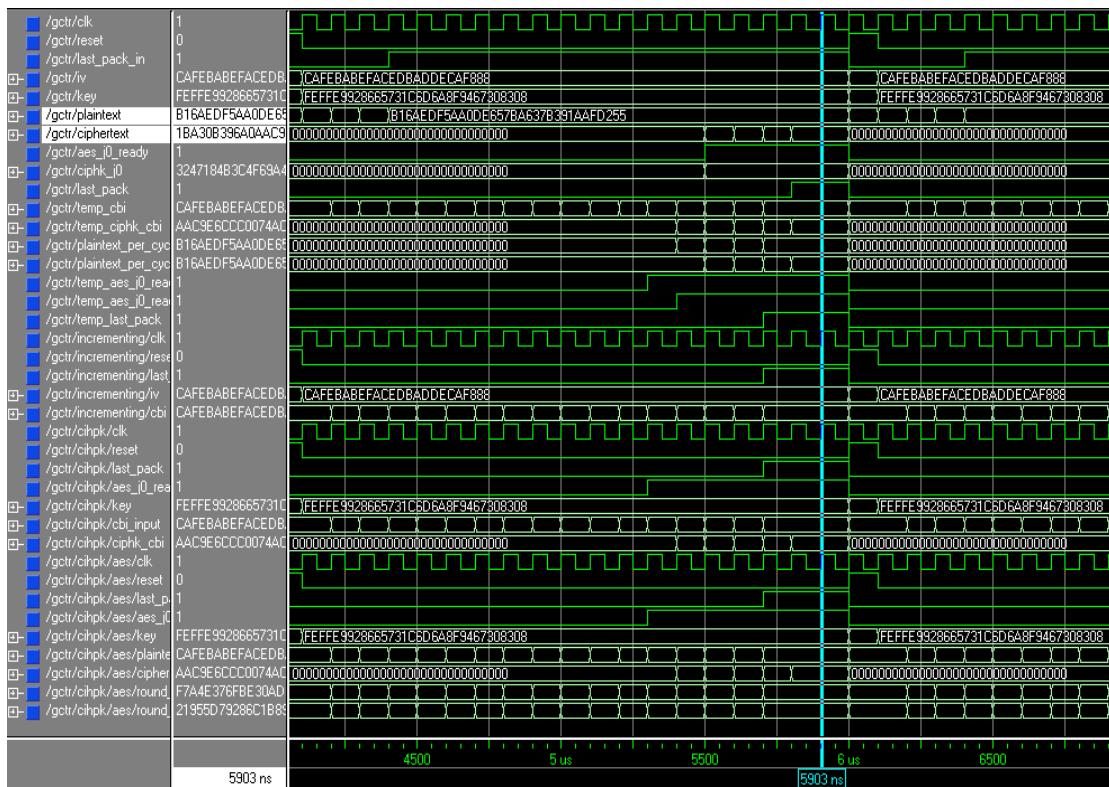
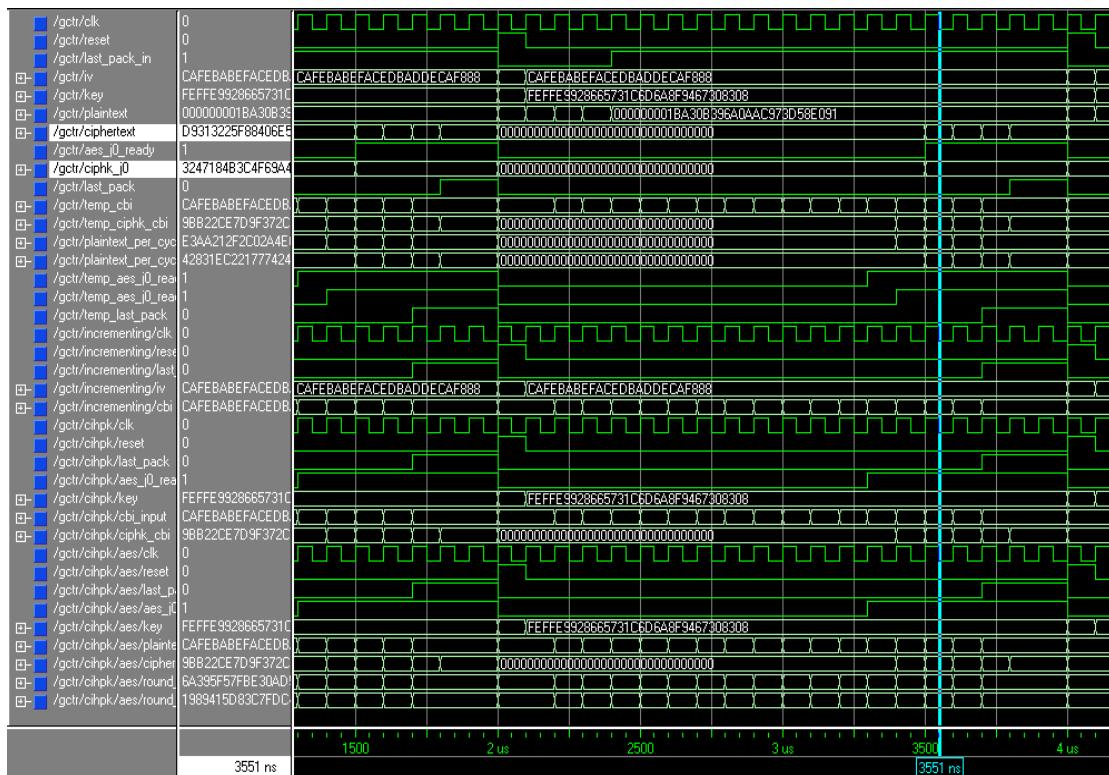
### Plaintext\_4\_full\_pack, example\_2:



Επιπρόσθετα test vectors μπορούν να δημιουργηθούν μέσω του κώδικα C που αναπτύξαμε και επιβεβαιώνουν την σωστή λειτουργία των επιμέρους κομματιών για διαφορετικά κλειδιά και IV.

Στις παρακάτω εικόνες δίνονται τα αποτελέσματα σύνθεσης που προέκυψαν από διαδοχικές επικλήσεις του **GCTR μηχανισμού** εμπιστευτικότητας για κρυπτογράφηση και αποκρυπτογράφηση ώστε να φανεί πιο ολοκληρωμένα η ροή των δεδομένων.





## 7.2 Σύνθεση της GCTR Δομής του AES-GCM

Για την σύνθεση των επιμέρους τμημάτων και ολόκληρου του συστήματος χρησιμοποιήθηκαν δύο διαφορετικά εργαλεία για δύο διαφορετικές τεχνολογίες *FPGAs*. Με τον τρόπο αυτό θα δείξουμε πόσο σημαντικό ρόλο έχει η επιλογή της τεχνολογίας στην τελική υλοποίηση του συστήματος πάνω σε ένα ολοκληρωμένο κύκλωμα (integrated circuit). Χωρίς να γίνει καμία απολύτως αλλαγή στο σχεδιασμό VHDL θα πάρουμε τα αποτελέσματα μας από τα δύο παρακάτω εργαλεία.

- Αρχικά, χρησιμοποιήθηκε το εργαλείο Leonardo Spectrum version 2000.1b. Συγκεκριμένα, χρησιμοποιήθηκε FPGA της εταιρείας Xilinx, από όπου επιλέχθηκε η οικογένεια *VIRTEX-II* και η συσκευή 2V10000bf957.
- Στη συνέχεια χρησιμοποιήθηκε το εργαλείο Xilinx ISE 9.2i, του οποίου το manual δίνεται στο Appendix C. Συγκεκριμένα, χρησιμοποιήθηκε FPGA της εταιρείας Xilinx, από όπου επιλέχθηκε αυτή τη φορά η οικογένεια *VIRTEX-V* και η συσκευή xc5vlx50-3ff1153.

### 7.2.1 Αποτελέσματα σύνθεσης ως προς την επιφάνεια και τον χρόνο

#### VIRTEX-II device 2V10000bf957

Resource	Used	Available	Utilization
IOs	613	648	89,62%
Function Generators	27942	122880	22,74%
CLB Slices	13971	61440	22,74%
Dffs or Latches	5696	124932	4,56%

Από την σύνθεση, προέκυψε, επίσης, η συχνότητα ρολογιού του GCTR module του AES-GCM αλγόριθμου η οποία, σύμφωνα με το Leonardo Spectrum version 2000.1b είναι :

Clock Frequency Report
72.7 MHz

Το κομμάτι που είναι υπεύθυνο για την εισαγωγή της μέγιστης καθυστέρησης (*critical path*) των συνολικού κυκλώματος είναι ο πυρήνας της *CTR mode*, δηλαδή ο *AES ECB mode* και ειδικότερα το κρίσιμο μονοπάτι περνάει μέσα από τους γύρους των *SubBytes* Μετασχηματισμού καθώς εκεί βρίσκεται το *S-box*.

Ενδεικτικά παρακάτω δίνονται τα αποτελέσματα σύνθεσης των επιμέρους components που συνθέτουν τον GCTR μηχανισμό εμπιστευτικότητας.

```
*****
```

Cell: AES\_Pipelined View: str Library: work

```
*****
```

```
*****
```

Device Utilization for 2V10000bf957

```
*****
```

Resource	Used	Avail	Utilization
<hr/>			
IOs	388	684	56.73%
Function Generators	27002	122880	21.97%
CLB Slices	13501	61440	21.97%
Dffs or Latches	3632	124932	2.91%

```
-----  
Using wire table: xcv2-250-5_avg
```

#### Clock Frequency Report

Clock	: Frequency
clk	: 72.7 MHz

```
*****
```

Cell: key\_expansion\_pipelined View: str Library: work

```
*****
```

```
*****
```

Device Utilization for 2V10000bf957

```
*****
```

Resource	Used	Avail	Utilization
<hr/>			
IOs	1410	684	206.14%
Function Generators	870	122880	0.71%
CLB Slices	728	61440	1.18%
Dffs or Latches	1456	124932	1.17%

```
-----  
This design does not fit in the device specified!
```

This design does not fit into any device in this technology!

```
Using wire table: xcv2-250-5_avg
```

#### Clock Frequency Report

Clock	: Frequency
clk	: 76.8 MHz

Παρόλο που στις εισόδους του key\_expansion\_pipelined υπάρχει overmapped αυτό δεν εμπόδισε το ολικό σύστημα του GCM να χωράει στη συγκεκριμένη συσκευή, καθώς τα IOs του key\_expansion\_pipelined είναι εσωτερικά καλώδια πλέον και δεν θα συνδέονται στα εξωτερικά IOs ports του FPGA.

## VIRTEX-V device xc5vlx50-3ff1153

Resource	Used	Available	Utilization
IOs	613	560	109%
SliceL	8,137	28,800	28%
SliceM	5,019	28,800	17%

Συγκρίνοντας τα αποτελέσματα που πήραμε από τις δύο τεχνολογίες, παρατηρούμε ότι τα IOs είναι ακριβώς ίδια και στις δύο περιπτώσεις. Παρατηρούμε όμως ότι στη συγκεκριμένη συσκευή που χρησιμοποιήθηκε υπάρχει overmapped στα IOs καθώς χρησιμοποιούνται περισσότερα από αυτά που είναι διαθέσιμα. Βέβαια να σημειώσουμε ότι αυτό δεν σημαίνει απαραίτητα πως το ολικό σύστημα του AES-GCM δεν θα χωράει στη συγκεκριμένη συσκευή, καθώς τα IOs του GCTR module θα είναι εσωτερικά καλώδια πλέον και δεν θα συνδέονται στα εξωτερικά IOs ports του FPGA. Και όπως θα παρουσιάσουμε παρακάτω, ο AES-GCM τοποθετείται επιτυχώς στην επιλεγμένη συσκευή.

Σύγκριση όσο αφορά στην επιφάνεια δεν είναι εφικτή και δεν προτείνεται με την προηγούμενη γενιά FPGA αφού η όλη αρχιτεκτονική έχει αλλάξει. Τα slices στην VIRTEX-V τεχνολογία είναι διπλάσια σε σχέση με αυτά των Virtex II. Επίσης, υπάρχουν δυο ειδών slices τα sliceM και sliceL. Τα πρώτα περιέχουν και μνήμη ενώ τα δεύτερα περιέχουν LUTs. Αυτό προέρχεται από το γεγονός ότι στην VIRTEX-V τεχνολογία ο ορισμός του CLB είναι δύο φέτες και κάθε φέτα περιέχει τέσσερα 6-εισόδων LUTs, τέσσερα flip-flops, ευρείας λειτουργίας, πολυπλέκτες, και carry logic ενώ ένα CLB στην τεχνολογία Virtex-II αποτελείται από τέσσερις φέτες και κάθε φέτα περιέχει δύο 4-εισόδων LUTs, δύο flip-flops, ευρείας λειτουργία πολυπλέκτες, και carry logic. Η παραδοσιακή τεσσάρων εισόδων LUT έχει ένα πίνακα αλήθειας για 16 διαφορετικούς συνδυασμούς. Το νέο έξι εισόδων LUT αυξάνει τον πίνακα αλήθειας σε 64 διαφορετικούς συνδυασμούς. Για παράδειγμα, αν εξεταστεί η εφαρμογή της απλής σύγκρισης μεταξύ των δύο 16-bit αριθμούς, απαιτούνται 11 LUT 4 εισόδων σε 3 επίπεδα ενώ απαιτούνται μόνο 7 LUT 6 εισόδων σε 2 επίπεδα. Η λιθογραφία που χρησιμοποιείται από 90nm έχει πέσει στα 65nm. Η όλη διάταξη των slices έχει αλλάξει χρησιμοποιώντας διαγώνια συμμετρία με αποτέλεσμα οι συνδέσεις μεταξύ τους να είναι πιο γρήγορες. Γίνεται κατανοητή η όποια διαφορά υπάρχει μεταξύ των δύο συνθέσεων.

Η συχνότητα ρολογιού του GCTR module του AES-GCM αλγόριθμου η οποία, σύμφωνα με το Xilinx ISE 9.2i είναι :

Clock Frequency Report
107.2 MHz

Είναι φυσικά της ίδιας τάξης μεγέθους και περνάει μέσα από το ίδιο κρίσιμο μονοπάτι, δεν αναμενόταν κάτι διαφορετικό εφόσον δεν έγινε καμία αλλαγή στο σχεδιασμό VHDL, αλλά η πιο προηγμένη τεχνολογία προσφέρει μια σημαντική αύξηση των 34,5 MHz και προκαλεί αντίστοιχη αύξηση στην απόδοση όπως αναφέρεται παρακάτω.

## 7.2.2 Μετρήσεις Απόδοσης

Ένα σημαντικό μέτρο της απόδοσης του συστήματος είναι το throughput, δηλαδή ο ρυθμός επεξεργασίας των δεδομένων στην μονάδα του χρόνου. Το throughput ορίζεται σύμφωνα με την σχέση:

$$\text{Throughput} = \frac{N * \text{Frequency}}{P},$$

όπου N είναι το μήκος των δεδομένων που παράγει η μονάδα, Frequency είναι η μέγιστη συχνότητα λειτουργίας της μονάδας και P είναι ο απαιτούμενος αριθμός παλμών για την εξαγωγή του τελικού αποτελέσματος στην έξοδο της μονάδας.

- Παρακάτω, έχει υπολογιστεί αναλυτικά το throughput για **VIRTEX-II** :

To GCTR module του AES-GCM αλγόριθμου για την κρυπτογράφηση και αποκρυπτογράφηση, παράγει το πρώτο Ciphertext / Plaintext μήκους 128 bits μέσα σε 14 κύκλους ρολογιού το οποίο στην ουσία αποτελεί την καθυστέρηση (delay) της πρώτης τιμής και δεν παίζει ρόλο στην απόδοση του κυκλώματος καθώς οι ενδιάμεσοι καταχωρητές, που έχουν τις προσωρινές τιμές, αλλάζουν σε κάθε κύκλο, αφού σε κάθε κύκλο γίνεται επεξεργασία ενός νέου 128-bit πακέτου. Στη συνέχεια, αφού το pipeline του AES είναι γεμάτο, σε κάθε κύκλο ρολογιού παράγεται το επόμενο μπλοκ των 128 bits ανεξαρτήτως των μέγεθος των προς κρυπτογράφηση / αποκρυπτογράφηση δεδομένων. Έτσι έχουμε :

$$\text{Throughput} = \frac{128 \text{ bits} * 72.7 \text{ MHz}}{1 \text{ clock\_cycle}} = 9.31 \text{ Gbits/sec}$$

- Παρακάτω, έχει υπολογιστεί αναλυτικά το throughput για **VIRTEX-V** :

$$\text{Throughput} = \frac{128 \text{ bits} * 107.2 \text{ MHz}}{1 \text{ clock\_cycle}} = 13.72 \text{ Gbits/sec}$$

Ουσιαστική σύγκριση των δύο διαφορετικών τεχνολογιών επιτυγχάνεται μόνο μέσω της απόδοσης. Επομένως, ο ίδιος VHDL σχεδιασμός όταν υλοποιηθεί πάνω σε ολοκληρωμένο κύκλωμα της πιο προηγμένης τεχνολογίας προσφέρει μια σημαντική αύξηση των 4,41 Gbps. Έχουμε δηλαδή μια αύξηση στην απόδοσης της τάξης του 47%. Γίνεται έτσι επιβλητική η ανάγκη για την υιοθέτηση νέων τεχνολογιών όταν κριτήριο της υλοποίησης είναι η απόδοση. Επιπλέον και η κατανάλωση είναι μικρότερη αφού η Virtex II έχει τάση λειτουργίας 1,5 V ενώ η Virtex V έχει 1,0 V τάση λειτουργίας.

### 7.3 Εισαγωγή στην κρυπτογραφική πλατφόρμα AES-GCM

Ο αλγόριθμος AES-GCM αποτελεί την ένωση των μηχανισμών πιστοποίησης και εμπιστευτικότητας. Αν ένα από τα δύο αυτά στοιχεία λείπει τότε σε καμία περίπτωση ο αλγόριθμος δεν μπορεί να θεωρηθεί πλήρης. Το δυνατό σημείο του αλγορίθμου αυτού έγκειται στο γεγονός ότι εγγυάται την εμπιστευτικότητα και την γνησιότητα των δεδομένων μέσα από ένα συνδυασμό των δύο μηχανισμών οι οποίοι συνεργάζονται και δουλεύουν παράλληλα.

Η συνεργασία των δύο μηχανισμών απαιτεί λεπτούς χειρισμούς κατά την σχεδίαση. Πολλά πράγματα έπρεπε να συμφωνηθούν και από τις δύο πλευρές προκειμένου να προκύψει το επιθυμητό αποτέλεσμα.

Ο μηχανισμός εμπιστευτικότητας έπρεπε να σχεδιαστεί με βάση συγκεκριμένα κριτήρια προκειμένου να παρέχει τις επιθυμητές τιμές στον μηχανισμό πιστοποίησης. Πολλές φορές ήταν αναγκαίο να σχεδιαστούν κάποια κομμάτια ξανά μέχρι να φτάσουν σε ένα συγκεκριμένο επίπεδο συμβατότητας με σκοπό να δουλεύουν μαζί και παράλληλα.

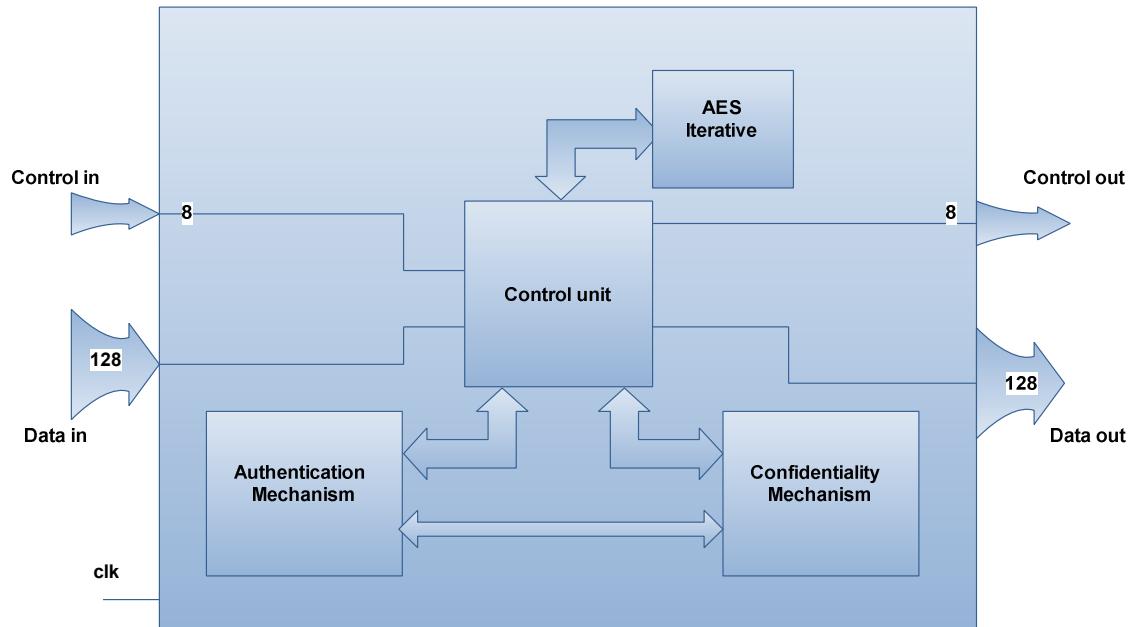
Ο μηχανισμός γνησιότητας από την άλλη, έπρεπε να σχεδιαστεί με βάση το γεγονός ότι προορίζεται για λειτουργία σε συνεργασία με τον μηχανισμό εμπιστευτικότητας. Η GHASH λόγο του αναδρομικού της χαρακτήρα έπρεπε να ελέγχει τις τιμές που δέχεται και να κλειδώνει αν αυτές δεν έχουν αλλάξει. Αυτό ήταν αναγκαίο λαμβάνοντας υπόψη ότι στην σχεδίαση του κρυπτογραφικού μέρους υπάρχουν pipeline αρχιτεκτονικές. Επίσης, πολλές αλλαγές χρειάστηκε να γίνουν σε όλα τα στάδια του σχεδιασμού με σκοπό η συχνότητα λειτουργίας να πλησιάζει αυτή του κρυπτογραφικού μέρους.

Για την σωστή συνεργασία των μηχανισμών καθώς και για τον έλεγχο των τιμών των εισόδων επιλέχθηκε η κατασκευή ενός control unit. Βεβαία, η σύνδεση των μηχανισμών θα μπορούσε να είχε γίνει με πολυπλέκτες και με ειδικά σήματα. Μια τέτοια προσέγγιση υλοποιήθηκε αλλά κρίθηκε ως πιο κατάλληλη η επιλογή του control unit ώστε να είναι εμφανές ο τρόπος που αλληλεπιδρά το σύστημα. Από μόνο του το control unit έχει συχνότητα λειτουργίας 72,8 MHz πράγμα που σημαίνει ότι δεν θα καθυστερεί το συνολικό κύκλωμα. Βέβαια, η συχνότητα λειτουργίας του συνολικού κυκλώματος είναι ελαφρά μικρότερη από την συχνότητα λειτουργίας του πιο αργού κυκλώματος. Κάτι τέτοιο όμως ήταν αναμενόμενο. Σκοπός της παρούσας σχεδίασης ήταν η υλοποίηση του GCM για high speed απαιτήσεις ώστε να αποτελεί μια ολοκληρωμένη ready to use λύση.

Ο τρόπος με τον οποίο λειτουργεί ο αλγόριθμος από τη μεριά του χρήστη είναι με ένα set εντολών. Αυτή η υλοποίηση στηρίχτηκε στις βασικές αρχές με τις οποίες κατασκευάστηκαν οι πρώτοι επεξεργαστές. Δηλαδή το σύστημα έχει μια γραμμή για την λήψη των εντολών και μία γραμμή για την λήψη των δεδομένων. Διαθέτει μία επιπλέον γραμμή εντολών στην έξοδό του που σκοπό έχει να ενημερώνει τον χρήστη σχετικά με το στάδιο στο οποίο βρίσκεται το σύστημα.

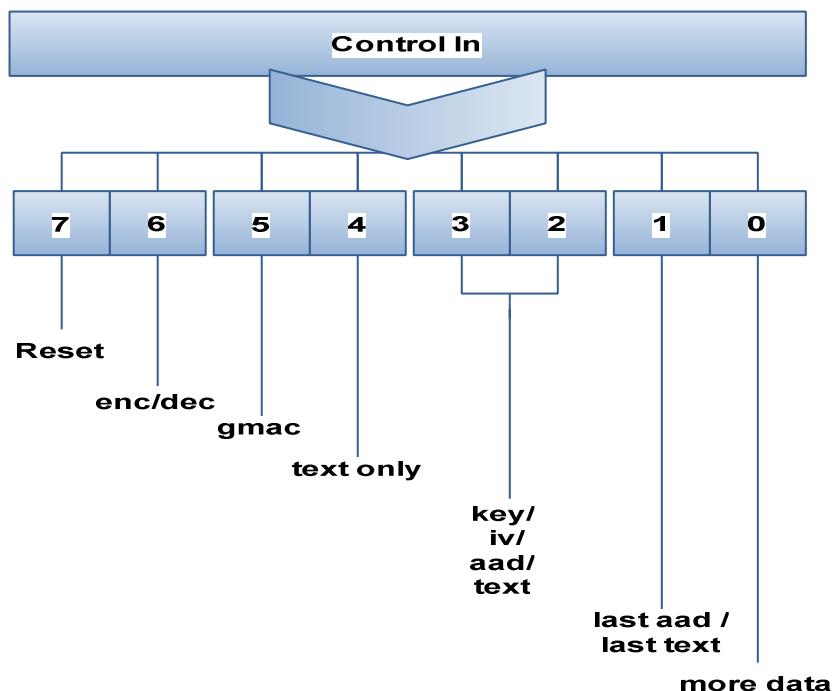
Ως μελλοντική επέκταση, εκτός από ενδεχόμενες βελτιστοποιήσεις που μπορούν να γίνουν στον AES-GCM, θα μπορούσε το ολοκληρωμένο να εμπλουτιστεί με περισσότερους κρυπτογραφικούς αλγόριθμους. Έτσι θα επεκτείνονταν και το set εντολών και με αυτό τον τρόπο θα πλησιάζε περισσότερο την έννοια του κρυπτογραφικού επεξεργαστή.

Πριν προχωρήσουμε σε περεταίρω ανάλυση, δίνεται ο σχεδιασμός που υλοποιήθηκε:



### 7.3.1 Τρόπος χρήσης - MANUAL

Όπως αναφέρθηκε παραπάνω, το συνολικό κύκλωμα λειτουργεί με ένα συγκεκριμένο set εντολών. Οι εντολές αυτές επιλέχθηκαν με βάση τη λογική :



Ο κωδικός που δίνεται αποτελείται από δύο hex χαρακτήρες. Ακολουθεί πίνακας με του κωδικούς που εισέρχονται στο σύστημα :

Λειτουργία	Κωδικός
reset	x80
Key in	x04
IV in	x05
Tag in	x06
encryption aad and text	x41
encryption aad only (GMAC)	x61
encryption text only	x51
decryption aad and text	x01
decryption aad only (GMAC)	x11
decryption text only	x09
AAD data	x49
Last AAD	x4b
Text data	x4d
last Text	x4e
Waiting / Finish	x00

Παρατηρούμε ότι η κρυπτογράφηση και η αποκρυπτογράφηση χωρίζονται σε τρείς κατηγορίες η κάθε μία. Αυτές είναι :

→ **Κανονική λειτουργία.** Σε αυτή την λειτουργία, στην είσοδο έρχονται και AAD δεδομένα και text δεδομένα. Ο μηχανισμός της GCTR μένει απενεργοποιημένος μέχρι να έρθει το πρώτο text. Αυτή η λογική υιοθετήθηκε για να έχουμε χαμηλότερη κατανάλωση και για να μην υπάρχουν συγκρούσεις μεταξύ AAD και text κατά την είσοδό τους στη GHASH. Επιπλέον, με αυτό τον τρόπο δεν χρησιμοποιούμε RAM. Κατά την κρυπτογράφηση, αφού έρθει το πρώτο plaintext, ο Tout μηχανισμός περιμένει την GCTR μέχρι να παράγει το πρώτο ciphertext, και στη συνέχεια δουλεύουν παράλληλα. Όταν η GCTR έχει δώσει την τελευταία τιμή τότε απομένουν ακόμα 5 κύκλοι μέχρι το tag να μεταφερθεί στην έξοδο. Κατά την αποκρυπτογράφηση, όταν έρθει το πρώτο ciphertext, και οι δύο μηχανισμοί δουλεύουν παράλληλα Τώρα, το tag έχει παραχθεί και περιμένει μέχρι να μεταφερθεί όλο το plaintext στην έξοδο και να δώσει την επιβεβαίωση ή το FAIL.

→ **GMAC λειτουργία.** Σε αυτή την λειτουργία αρχικά δουλεύουν και οι δύο μηχανισμοί παράλληλα. Ο μηχανισμός Tout επεξεργάζεται τα AAD δεδομένα ενώ η GCTR λειτουργεί για 14 κύκλους, μέχρι να παράγει την τιμή  $CIPH_K(J_0)$ . Μετά απενεργοποιείται. Το ίδιο ισχύει στην κρυπτογράφηση και στην αποκρυπτογράφηση. Κατά την κρυπτογράφηση, το tag που παράγεται βγαίνει στην έξοδο. Κατά την αποκρυπτογράφηση, επιβεβαιώνεται η ορθή λήψη των δεδομένων ή όχι με βάση το tag εισόδου.

→ **Text only λειτουργία.** Υπάρχει διαχωρισμός μεταξύ κρυπτογραφικής και αποκρυπτογραφικής λειτουργίας. Κατά την κρυπτογράφηση, ο Tout μηχανισμός μένει ανενεργός μέχρι να παραχθεί το πρώτο ciphertext, δηλαδή για 14 κύκλους. Έπειτα οι δύο μηχανισμοί δουλεύουν παράλληλα. Κατά την αποκρυπτογράφηση και στους δύο μηχανισμούς εισέρχεται το ciphertext και δουλεύουν παράλληλα. Το μεν Tout για την παραγωγή του tag για σύγκριση με το tag in, ο δε GCTR μηχανισμός για την παραγωγή του plaintext.

Βλέπουμε λοιπόν, ότι οι υποστηριζόμενες καταστάσεις δεν είναι δυο, κρυπτογράφηση και αποκρυπτογράφηση, αλλά είναι ουσιαστικά 6 καταστάσεις που επιτελούν διαφορετικές λειτουργίες.

Υπάρχουν 4 εντολές για τα εισερχόμενα δεδομένα. Αυτές οι εντολές συνοδεύουν τα δεδομένα και έρχονται ταυτόχρονα. Από την στιγμή που υπάρχει μία γραμμή εισόδου δεδομένων αυτό κρίθηκε αναγκαίο για την σωστή λειτουργία. Οι εντολές αυτές είναι κατά κάποιο τρόπο η ταυτότητα των εισερχόμενων δεδομένων. Έτσι έχουμε :

- x49. Ο κωδικός αυτός συνοδεύει όλα τα AAD δεμένα εκτός του τελευταίου AAD.
- x4b. Ο κωδικός αυτός συνοδεύει μόνο το τελευταίο πακέτο των AAD δεδομένων.
- x4d. Ο κωδικός αυτός συνοδεύει όλα τα text δεμένα εκτός του τελευταίου
- x4e. Ο κωδικός αυτός συνοδεύει μόνο το τελευταίο πακέτο των text δεδομένων.

Ακολουθεί στη συνέχεια πίνακας με τους κωδικούς που εξέρχονται από το σύστημα :

Λειτουργία	Κωδικός
reset ok	x80
Key ok	x04
IV ok	x05
Tag ok	x06
text out	x0e
Tag out	x0f
Decryption ok	xaa
Decryption fail	xff
Processing	x89
Hold	x99
Ready to take command	x88
Ready to take data	x87
Finish	x00

Κάναμε προσπάθεια να δίνονται στον χρήστη όσο τον δυνατό περισσότερες εντολές ώστε το interface να είναι πλούσιο.

Έχουμε :

- x80. Ενημερώνει τον χρήστη ότι έχει γίνει reset και ότι είναι έτοιμο να δεχτεί εντολές.
- x04. Ενημερώνει τον χρήστη ότι έχει γίνει έχει πάρει το key επιτυχώς.
- x05. Ενημερώνει τον χρήστη ότι έχει γίνει έχει πάρει το IV επιτυχώς.
- x06. Ενημερώνει τον χρήστη ότι έχει γίνει έχει πάρει το Tag εισόδου επιτυχώς (χρειάζεται και κατά την κρυπτογράφηση).
- x0e. Ενημερώνει τον χρήστη ότι τα δεδομένα που είναι στην έξοδο είναι text.
- x0f. Ενημερώνει τον χρήστη ότι τα δεδομένα που είναι στην έξοδο είναι το tag.
- xaa. Ενημερώνει τον χρήστη ότι η αποκρυπτογράφηση ολοκληρώθηκε επιτυχώς.
- xff. Ενημερώνει τον χρήστη ότι η αποκρυπτογράφηση απέτυχε. “FAIL”.
- x89. Ενημερώνει τον χρήστη ότι εσωτερικά επεξεργάζεται δεδομένα.
- x99. Ενημερώνει τον χρήστη ότι δεν είναι έτοιμο να πάρει εντολή.
- x87 . Ενημερώνει τον χρήστη ότι το σύστημα είναι έτοιμο να πάρει δεδομένα.
- x00. Ενημερώνει τον χρήστη ότι όλες οι εσωτερικές ενέργειες έχουν ολοκληρωθεί.

Μετά την κατάσταση x00 (Finish) το σύστημα περνάει αυτόματα στην κατάσταση x80 (reset ok). Κάνει δηλαδή reset εσωτερικά και είναι έτοιμο να ξεκινήσει καινούριο κύκλο διεργασιών. Αυτό έγινε με σκοπό να διευκολύνει τον χρήστη και να μην χρειάζεται να κάνει κάθε φορά reset.

### **Σειρά εντολών:**

Η σειρά με την οποία το σύστημα δέχεται εντολές είναι :

Εισερχόμενες εντολές	Εξερχόμενες εντολές
x80	x80
x04	x04
x05	x05
x06	x06
	x99
	x88
x41	
x61	
x51	
x01	
x11	
x09	
x00	x87
x49	
x4b	
x4d	
x4e	
	x0e
	x0f
	xaa
	xff
	x00
	x80

Όπως βλέπουμε, πρώτα παίρνει τα δεδομένα key, IV, tag in, και μετά είναι σε θέση να δεχτεί την εντολή για του είδους την διεργασία που θα εκτελέσει. Σε πρωτόκολλα που στέλνουν τα δεδομένα με μορφή πακέτων αυτό μπορεί να είναι ιδιαίτερα χρήσιμο.

Όταν το σύστημα πάρει στην είσοδο και το tag\_in θα βγάλει μήνυμα αναμονής για 8 κύκλους καθώς τόσοι απομένουν μέχρι την παραγωγή του hash subkey. Σε αυτό το διάστημα δύναται ενδεχομένως σε μελλοντική αναβάθμιση να δέχεται άλλου είδους δεδομένα.

Αφού επιλέξει ο χρήστης το είδος της διεργασίας που θέλει να επιτελέσει το σύστημα, τον ενημερώνει ότι η επιλογή του έγινε δεκτή και είναι έτοιμο να λάβει τα δεδομένα. Αν ο χρήστης δεν δώσει κάποια από τις επιτρεπόμενες εντολές, το σύστημα αναμένει μέχρι να πάρει μία από τις αποδεκτές εντολές. Στην συνέχεια, τα δεδομένα εισέρχονται ως εξής: Πρώτα τα AAD, στην συνέχεια το τελευταίο πακέτο των AAD και μετά τα text δεδομένα και το τελευταίο text δεδομένο.

Το να ορίζεται τελευταίο πακέτο, είτε πρόκειται για AAD είτε για text, είναι σημαντικό διότι ενδέχεται το τελικό πακέτο να είναι ένα partial πακέτο ή ένα full πακέτο.

Κατά την διάρκεια που δέχεται τα δεδομένα και μέχρι να αρχίσει το σύστημα να βγάζει δεδομένα στην έξοδο, δηλαδή μέχρι να γεμίσουν τα pipeline της GCTR είτε μέχρι να ολοκληρωθούν οι εσωτερικές διεργασίες του Tout, δίνει το μήνυμα x87. Όταν αρχίσει να βγάζει δεδομένα θα βγάλει το μήνυμα x0e αν πρόκειται για text δεδομένα. Σε περίπτωση που συνεχίσουν να εισέρχονται δεδομένα, θα βγάζει και ταυτόχρονα θα δέχεται δεδομένα.

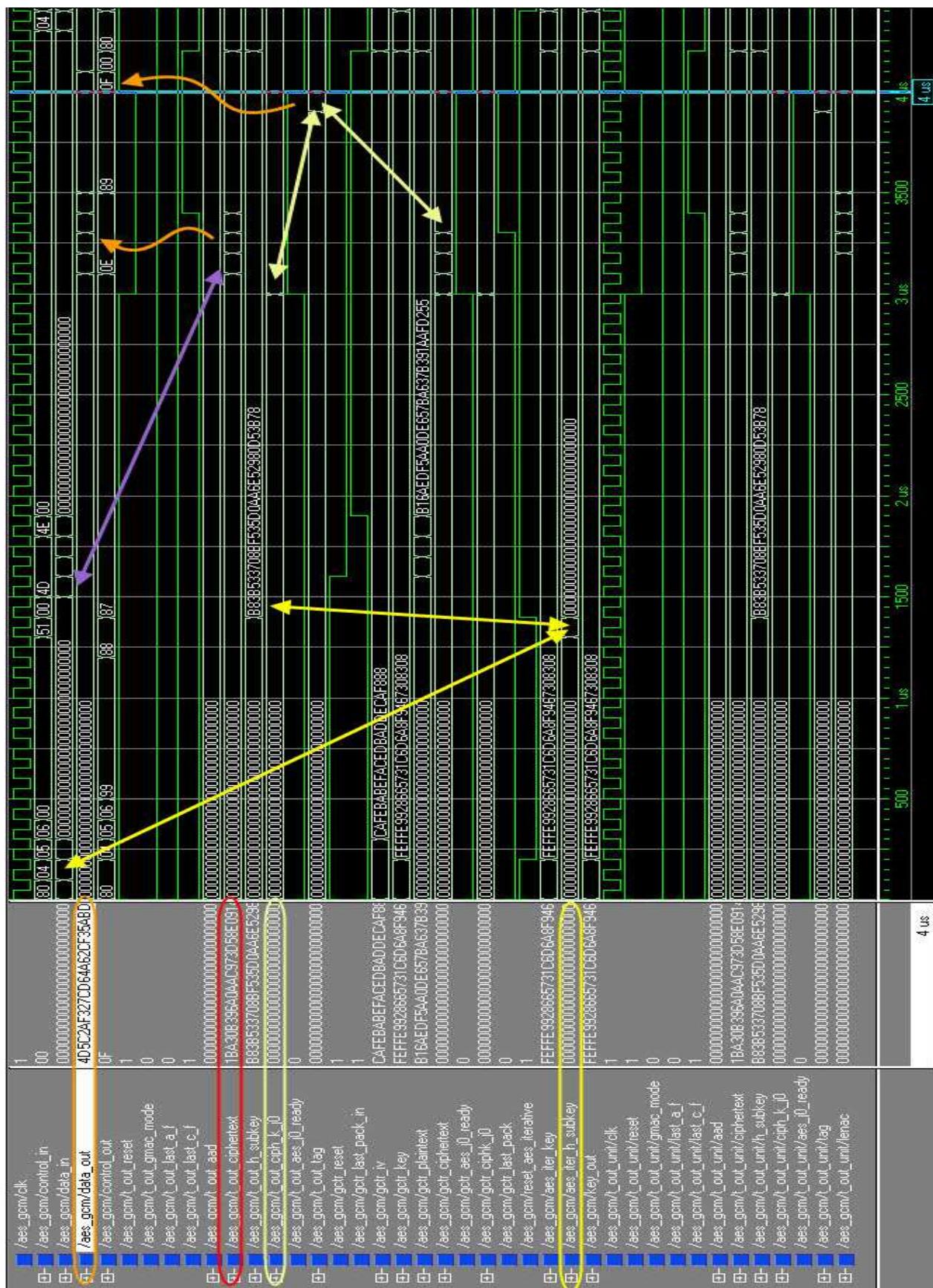
Αν πρόκειται για κρυπτογράφηση μετά τα text θα βγάλει το tag μετά από 5 κύκλους.

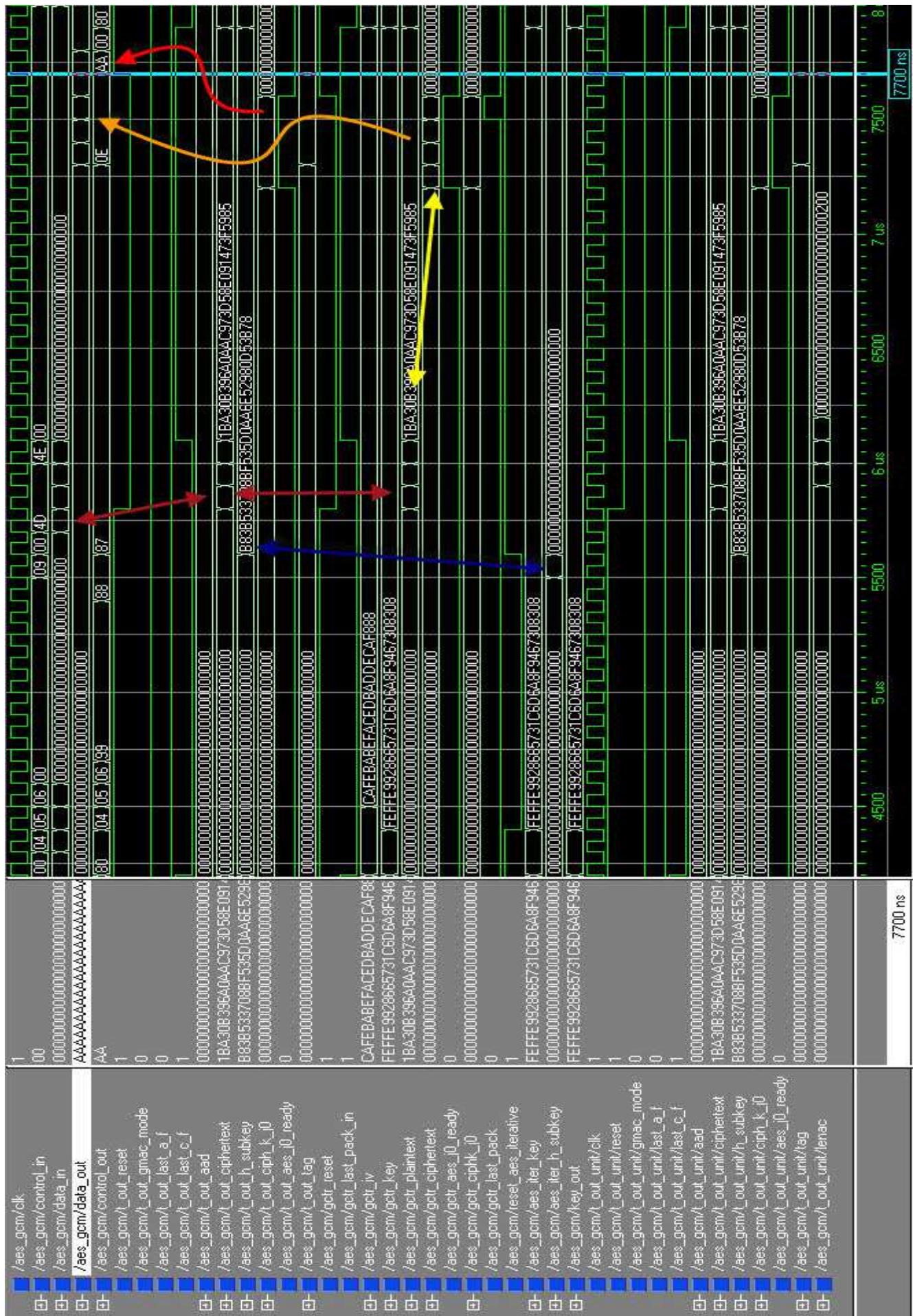
Αν πρόκειται για αποκρυπτογράφηση, αφού βγάλει όλο το plaintext, θα ενημερώσει τον χρήστη αν τα δεδομένα που εξήλθαν από το σύστημα, είναι έγκυρα (xaa) ή αν η κρυπτογράφηση ήταν ανεπιτυχείς (xff). Το σύστημα λειτουργεί καταυτόν τον τρόπο γιατί το μέγεθος των δεδομένων που θα λάβει δεν είναι γνωστό, και έτσι δεν απαιτείται καθόλου μνήμη κατά την υλοποίηση.

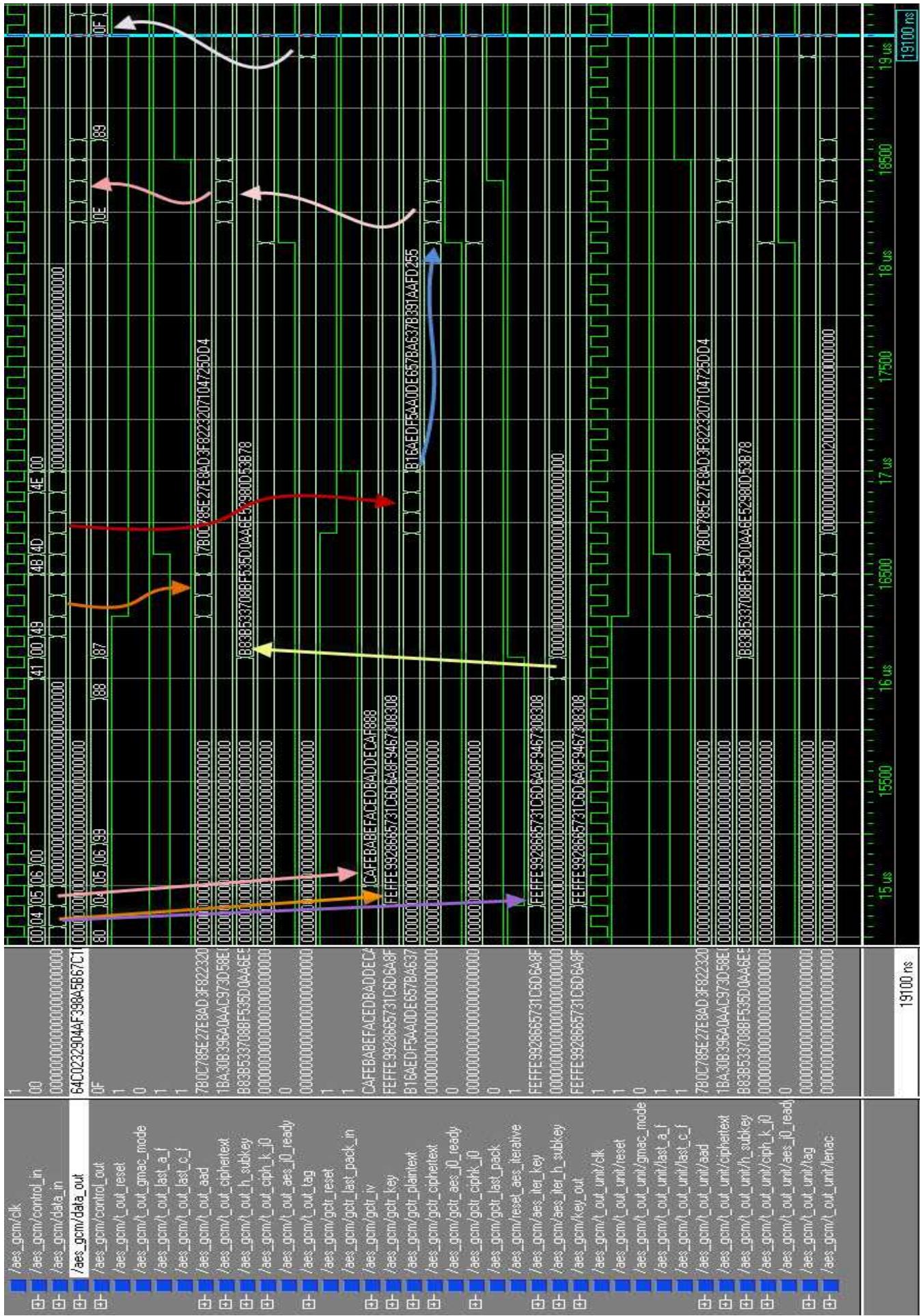
Στην συνέχεια, το σύστημα ενημερώνει ότι τελείωσε με τις εσωτερικές διεργασίες (x00) και κάνει μόνο του reset (x80) προκειμένου να είναι έτοιμο να χρησιμοποιηθεί εκ νέου. Reset από τον χρήστη χρειάζεται μόνο κατά την αρχική ενεργοποίηση του κυκλώματος.

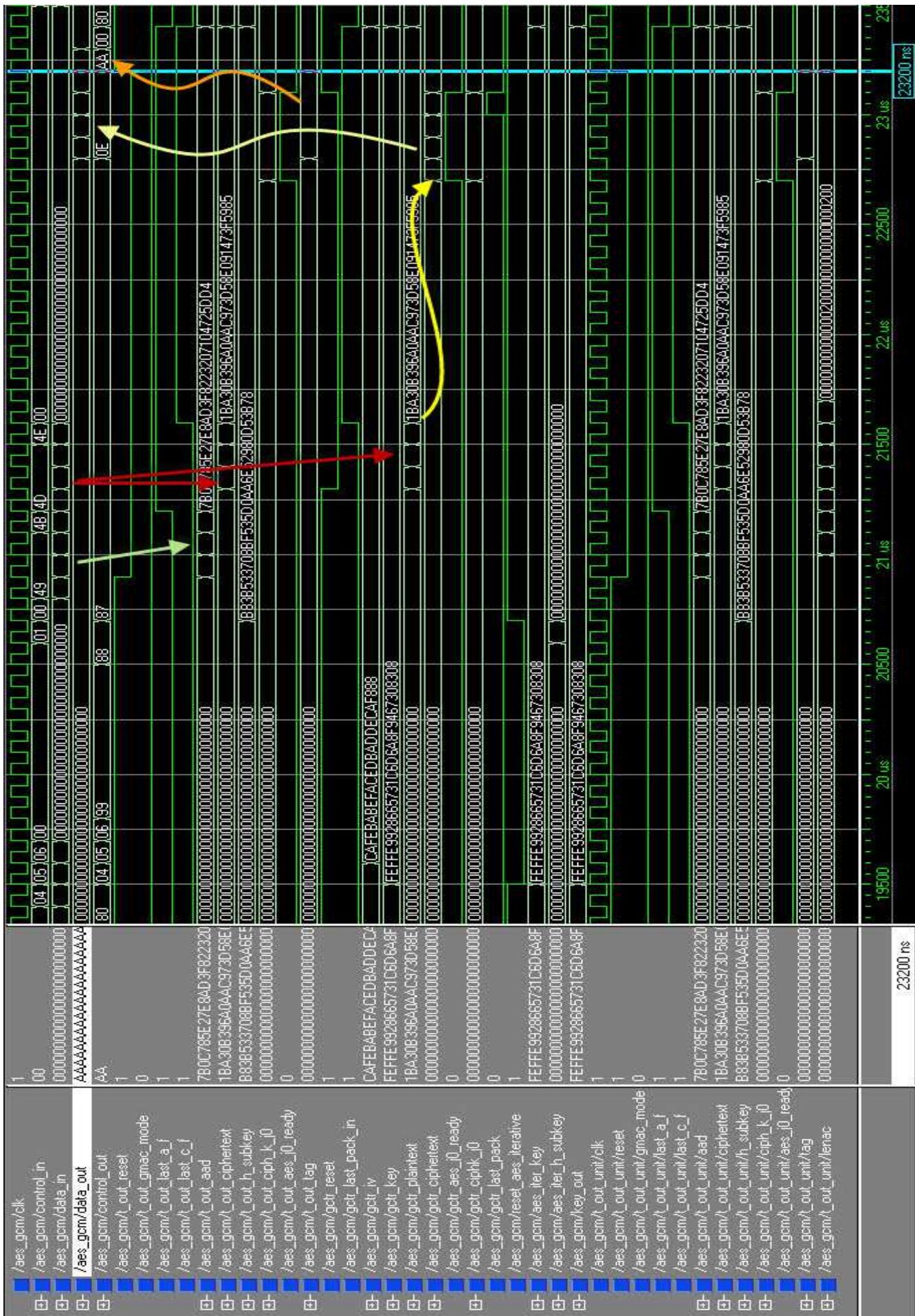
Σημαντικό είναι το γεγονός ότι όλα τα εξωτερικά σήματα, εισερχόμενα και εξερχόμενα, είναι handshaking σήματα και απαιτούν έναν κύκλο ρολογιού. Να σημειωθεί επίσης ότι από την στιγμή που εισέρχονται τα text δεδομένα απαιτείται συνεχή ροή λόγο της ύπαρξης των τεχνικών διοχέτευσης.

#### 7.4 Εξομοίωση του AES-GCM Προτύπου Ασφαλείας









Στις παραπάνω εξομοιώσεις παρουσιάστηκαν με την ακόλουθη σειρά:

- Κρυπτογράφηση μόνο με plaintext δεδομένα.
- Αποκρυπτογράφηση μόνο με ciphertext δεδομένα
- Κρυπτογράφηση με AAD και plaintext δεδομένα
- Αποκρυπτογράφηση με AAD και ciphertext δεδομένα.

Η παρουσίαση αυτών γίνεται με σκοπό την καλύτερη κατανόηση του τρόπου λειτουργίας.

## 7.5 Σύνθεση του AES-GCM Προτύπου Ασφαλείας

- **Σύνθεση με χρήση του Leonardo Spectrum, VIRTEX-II device 2V10000bf957**

Resource	Used	Available	Utilization
IOs	273	648	39.91%
Function Generators	56976	122880	46.37%
CLB Slices	28488	61440	46.37%
Dffs or Latches	8637	124932	6.91%

Από την σύνθεση, προέκυψε η συχνότητα ρολογιού του AES-GCM αλγόριθμου η οποία, σύμφωνα με το Leonardo Spectrum version 2000.1b είναι :

Clock Frequency Report
54.0 MHz

Το κομμάτι που είναι υπεύθυνο για την εισαγωγή της μέγιστης καθυστέρησης (*critical path*) του συνολικού κυκλώματος βρίσκεται εντός του παράλληλου πολλαπλασιαστή και πιο συγκεκριμένα κατά την κατασκευή του πίνακα 128x128.

Η απόδοση σύμφωνα με το εργαλείο αυτό είναι :

Throughput
7.0 Gbps

- Σύνθεση με χρήση του Xilinx ISE 9.2i, VIRTEX-V device xc5vlx50-3ff1153

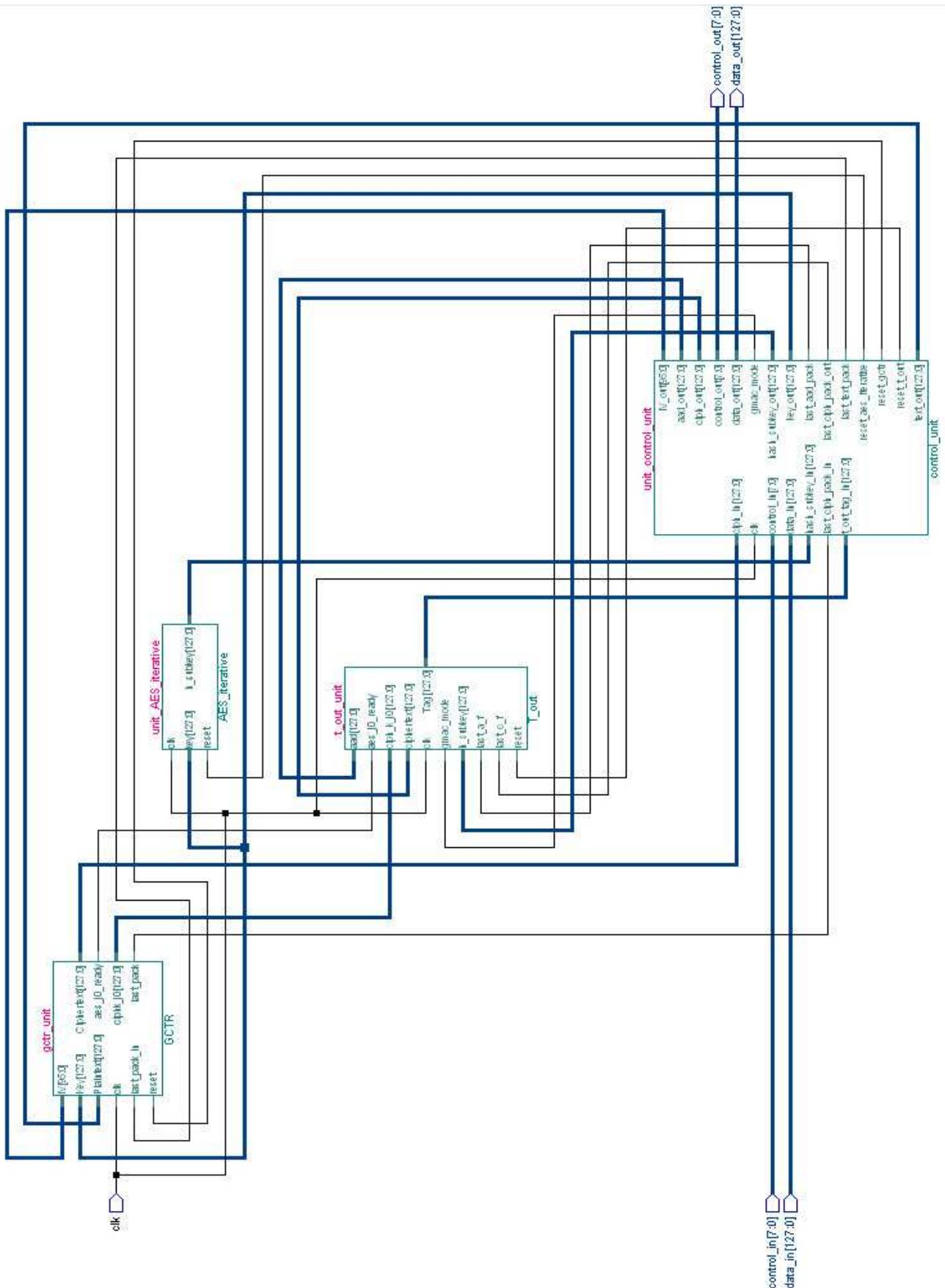
Resource	Used	Available	Utilization
IOs	273	560	48%
SliceL-> CLB	22,705	28,800	78%
SliceM ->Dff	7,579	28,800	26%

Clock Frequency Report
85.6 MHz

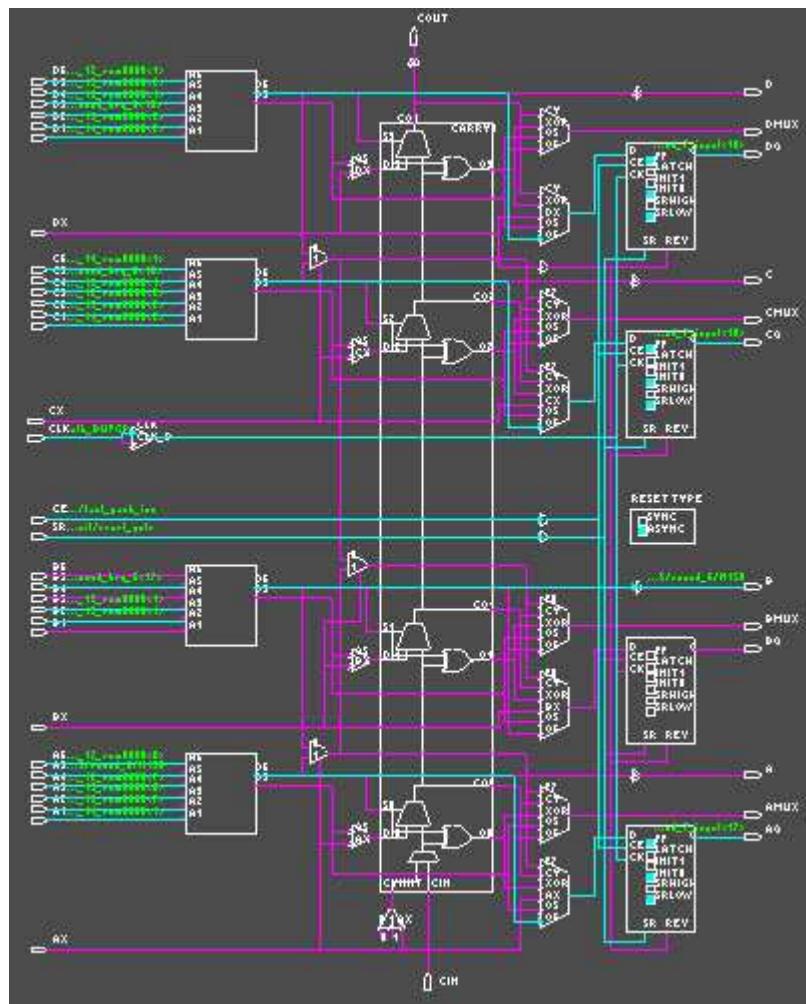
Throughput
11.0 Gbps

Συγκρίνοντας τις δύο τεχνολογίες ως προς την απόδοση βλέπουμε ότι στην τεχνολογία Virtex V παρατηρείται μια αύξηση των 4 Gbps, δηλαδή μια αύξηση της τάξης του 57%.

Ακολουθεί το κύκλωμα έτσι όπως παράχθηκε με την βοήθεια του Leonardo Spectrum v2000.1b:



Ενδεικτικά ένα slice του AES-pipeline round 7 παρουσιάζεται στην συνέχεια:





# **Παράρτημα Α**

**Xilinx ISE WebPACK Design Software**

**Έκδοση 9.2i for windows**

**Εγκατάσταση**

**&**

**Τρόπος χρήσης**

Το πρόγραμμα αυτό αποτελεί ένα πολύ χρήσιμο εργαλείο για hardware υλοποίηση καθώς δίνει στον χρήστη πληθώρα επιλογών. Είναι ένα εργαλείο της οικογένειας XILINX και αποτελεί την ελεύθερη έκδοση του ISE programming tools.

Στο παρόν θα παρουσιαστεί ο τρόπος εγκατάστασης και πώς χρησιμοποιούνται τα βασικά εργαλεία του προγράμματος. Σε κάθε βήμα θα αναφέρονται κάποιες χρήσιμες πληροφορίες.

## Εγκατάσταση

Download:

Το πρόγραμμα μπορεί να το βρει κάποιος στην ιστοσελίδα της XILINX :

<http://www.xilinx.com/>

επιλέγοντας :

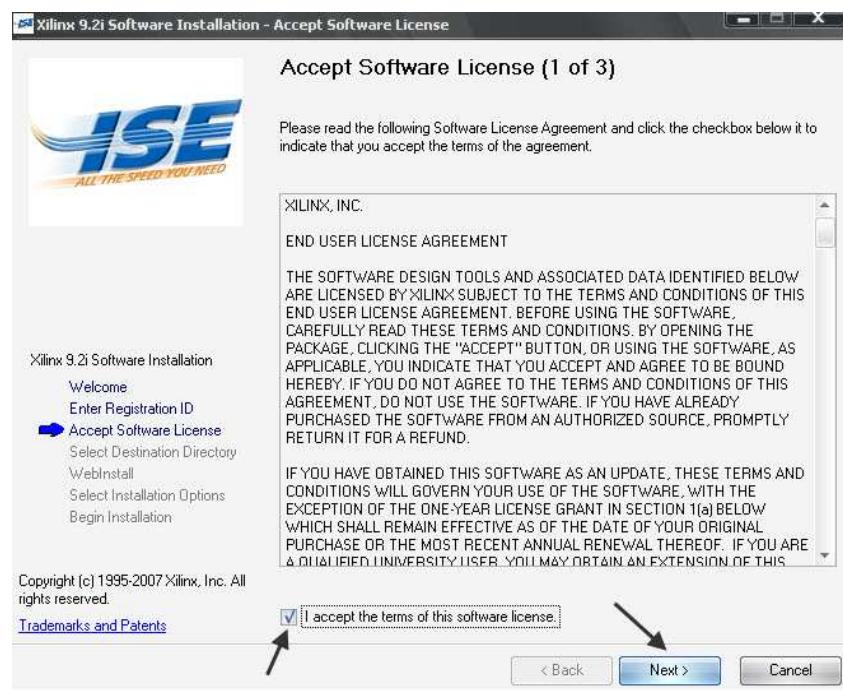
Downloads → Full Products and Tools → ISE WebPACK → 9.2i →  
ISE 9.2i Windows – WebPACK →

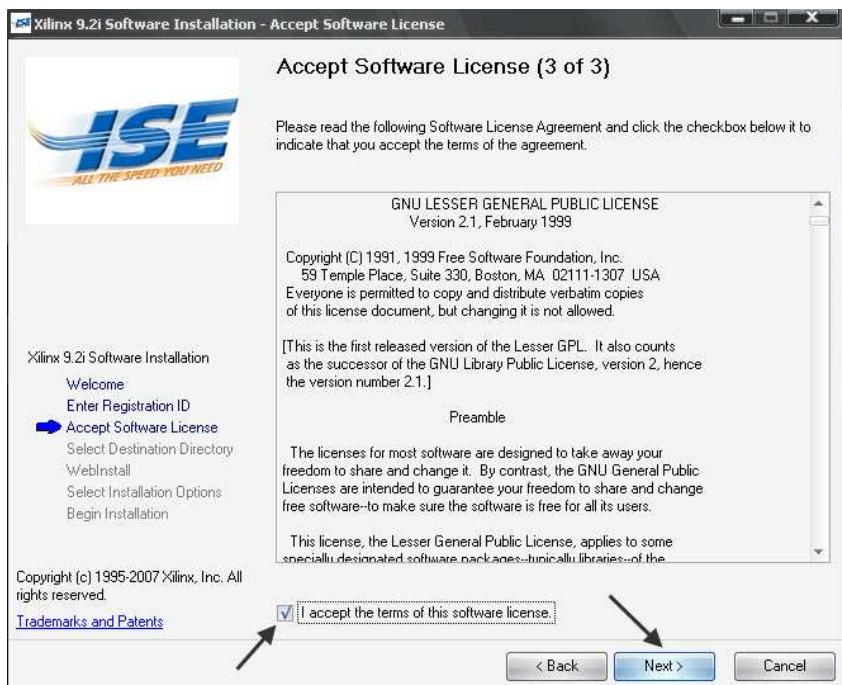
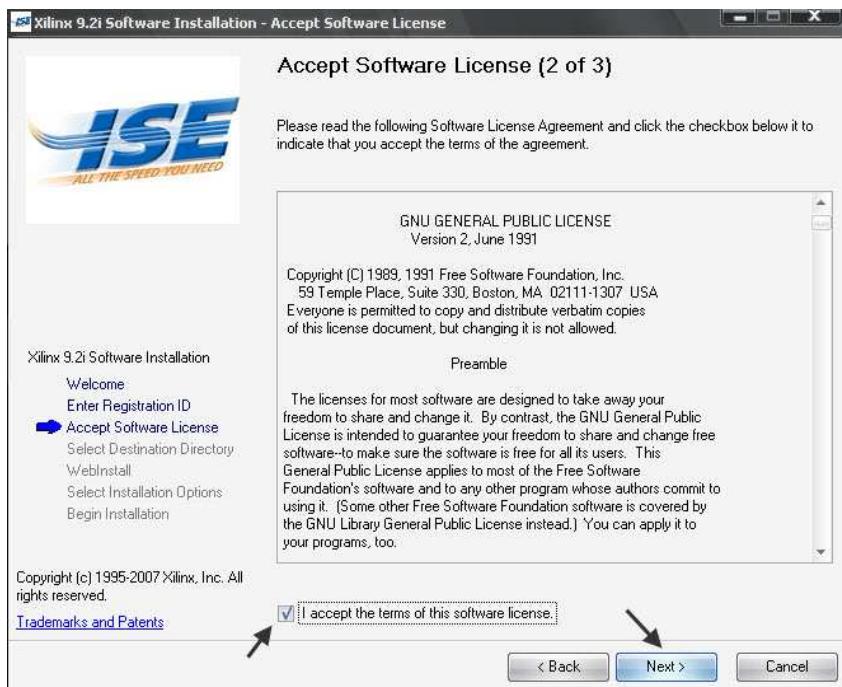


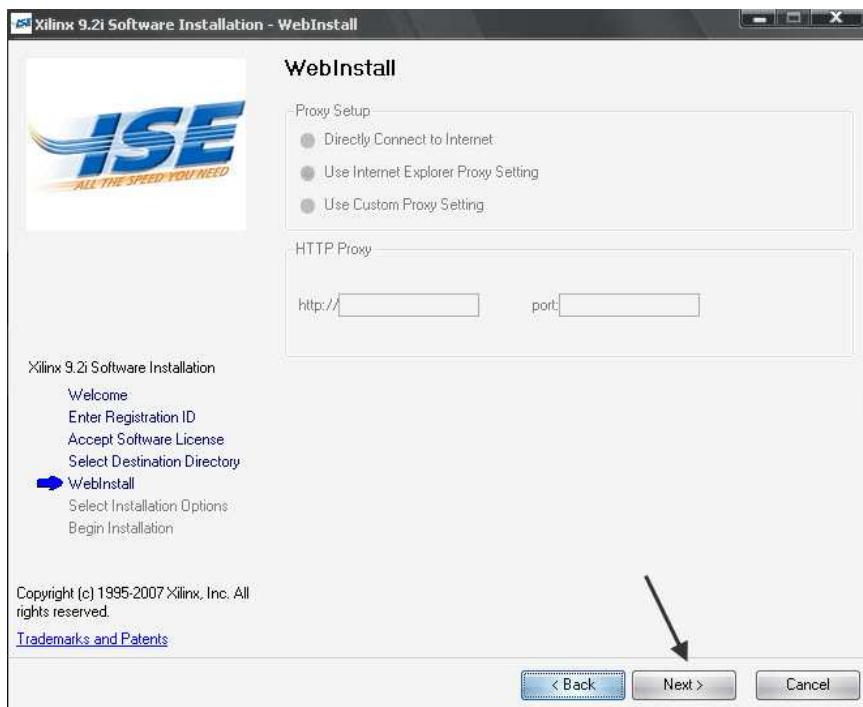
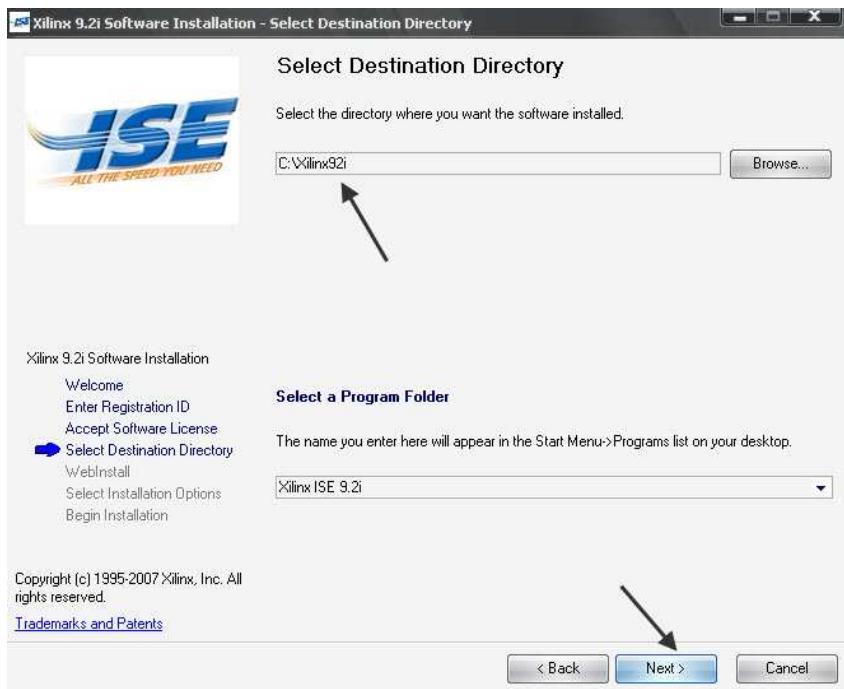
Single File Download (32-bit only)

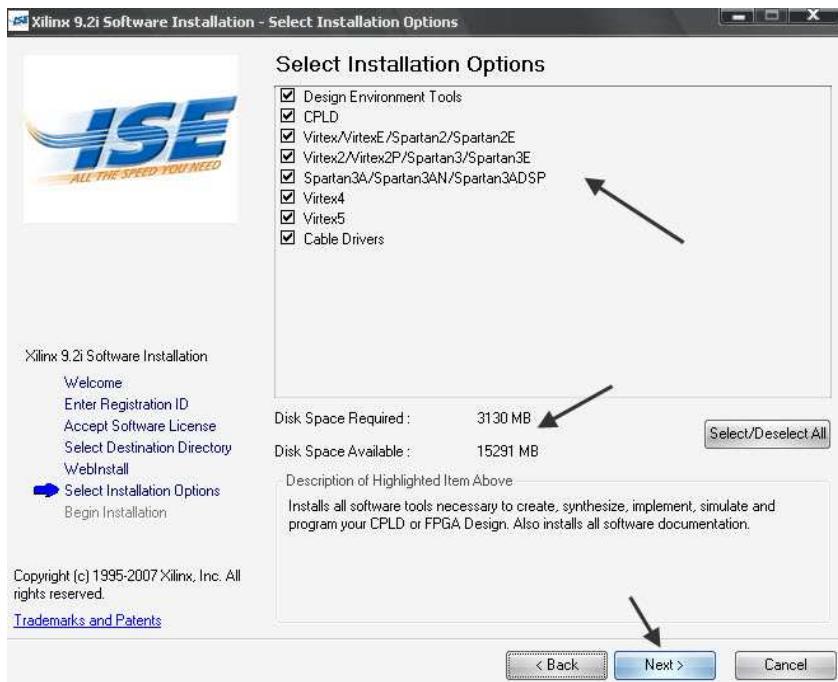
Σημείωση : το αρχείο είναι πολύ μεγάλο ZIP (1.7GB)

Install:

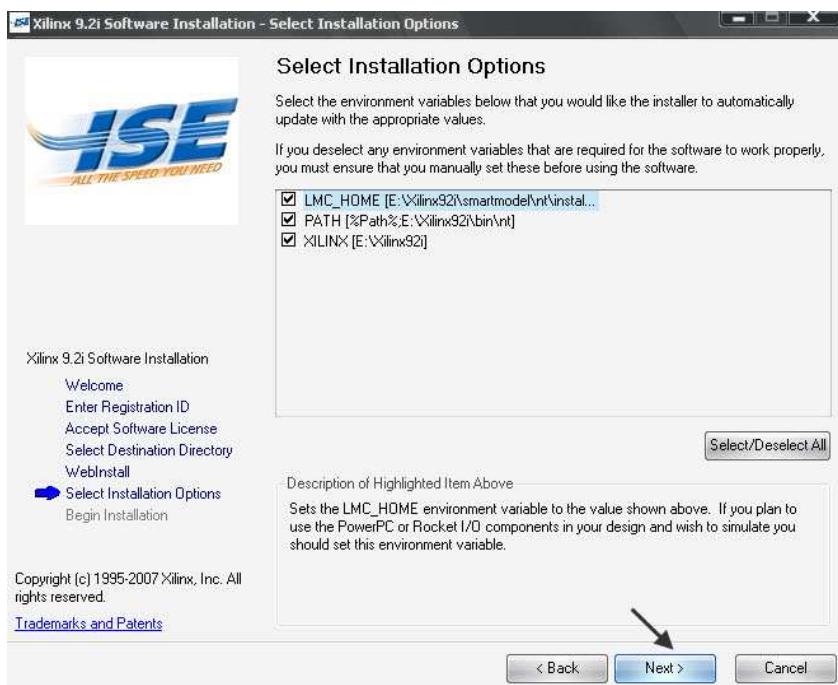


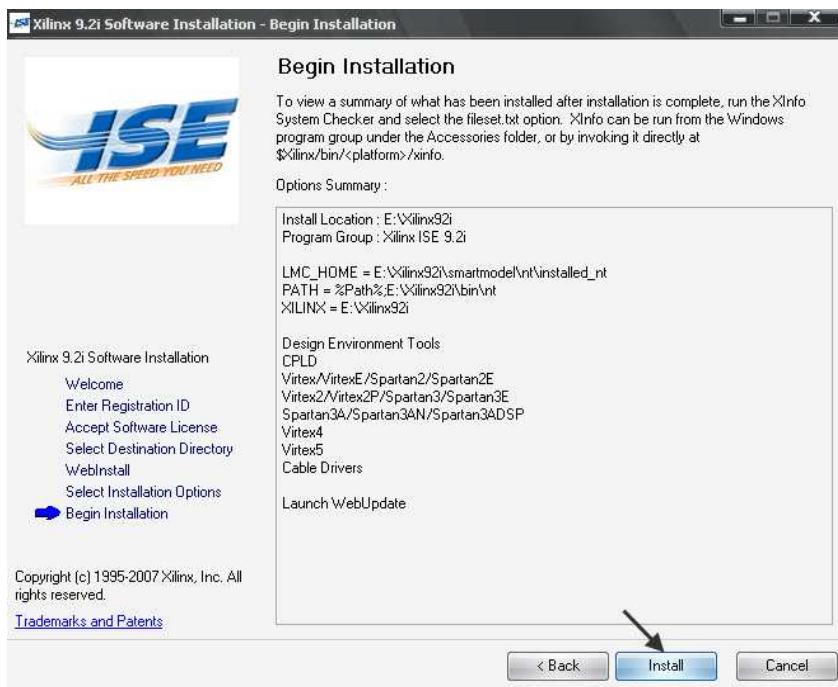
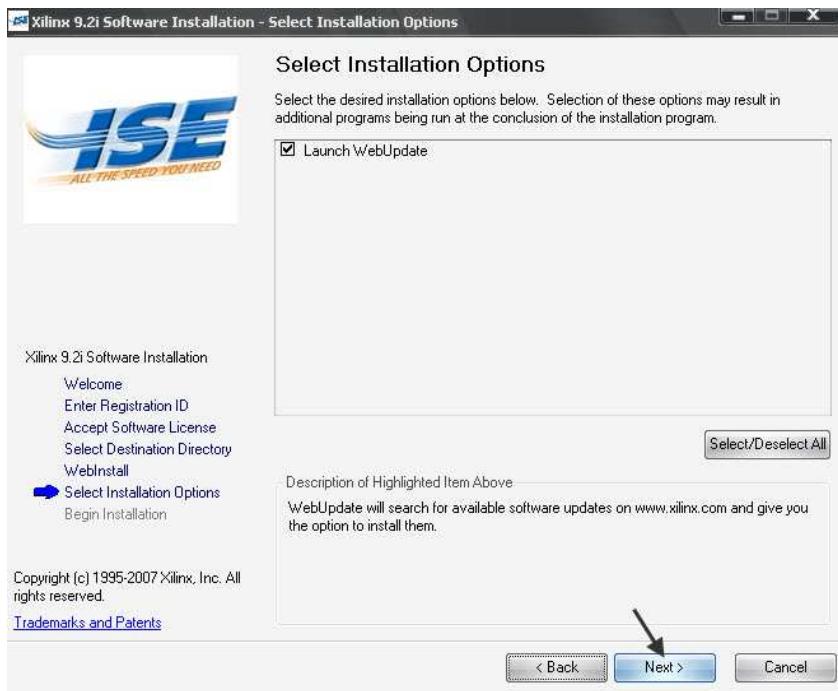


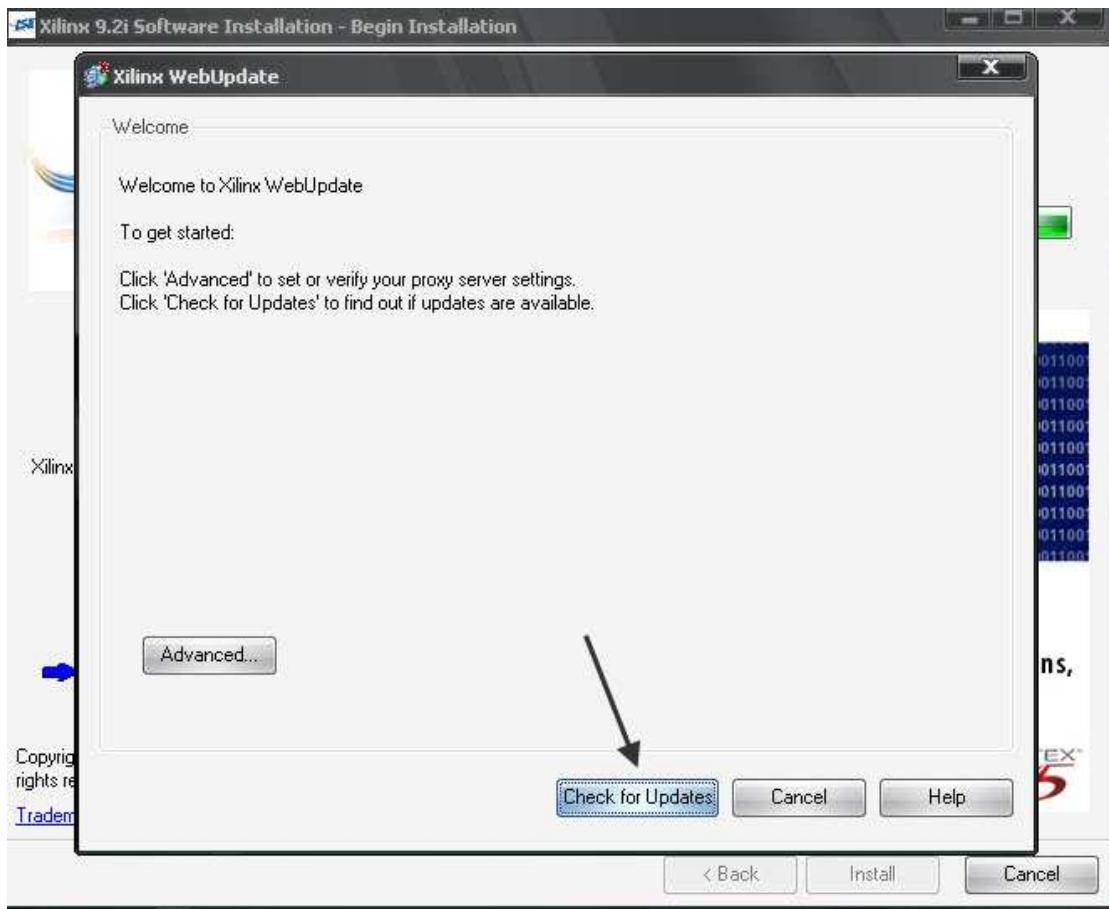




**Σημείωση :** Φροντίστε να υπάρχει ο απαιτούμενος ελεύθερος χώρος στον δίσκο για να πραγματοποιηθεί η εγκατάσταση .







**Σημείωση :** Όταν τελειώσει η εγκατάσταση θα βγάλει ένα παράθυρο για updates . Γενικά είναι χρήσιμο να γίνουν τα updates αλλά μπορεί να είναι και 1GB.

Τέλος εγκατάστασης .

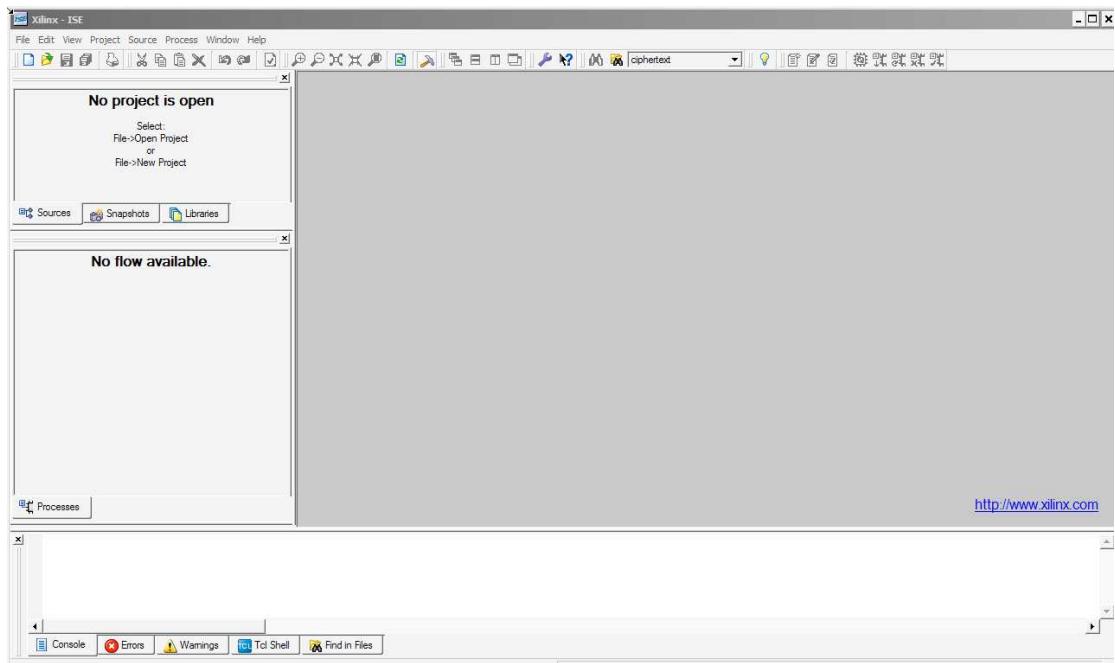
Για να ανοίξουμε το πρόγραμμα επιλέγουμε :



από την επιφάνεια εργασίας .

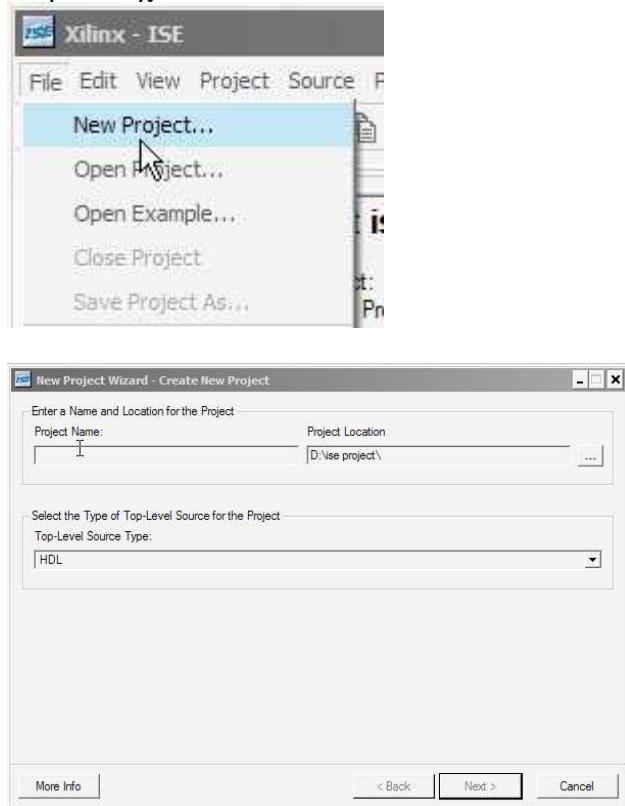
## Τρόπος Χρήσης

Το αρχικό παράθυρο είναι :

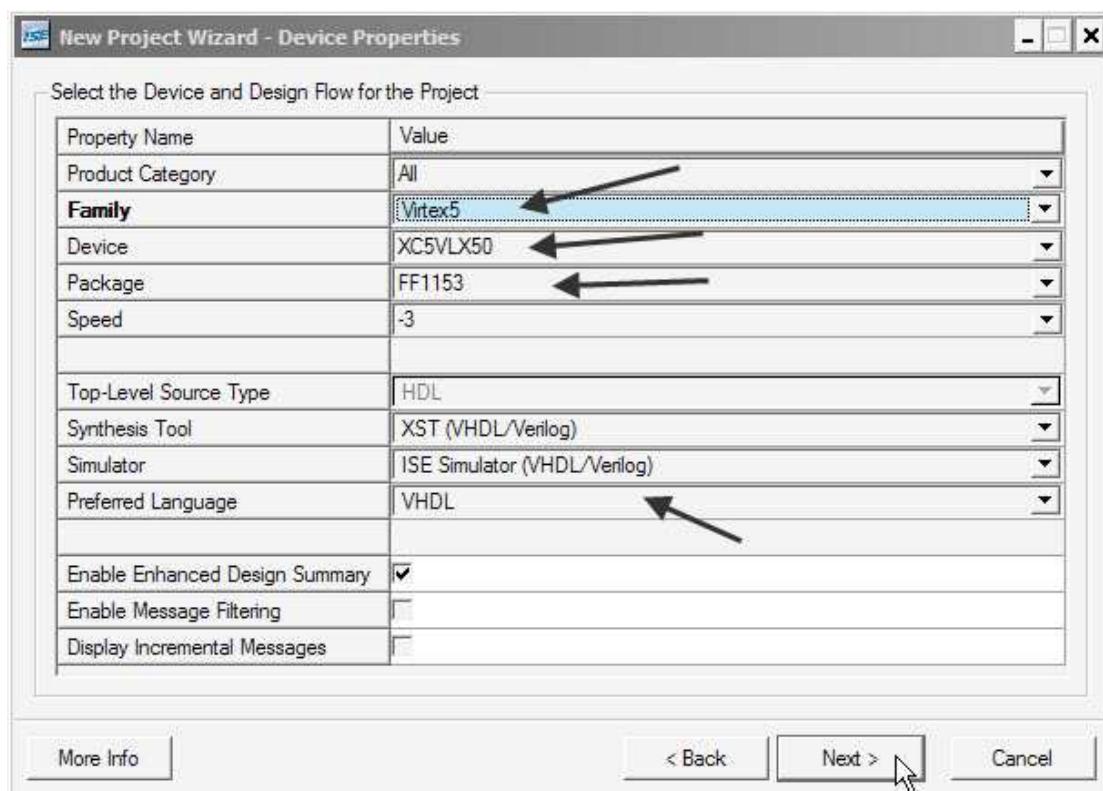
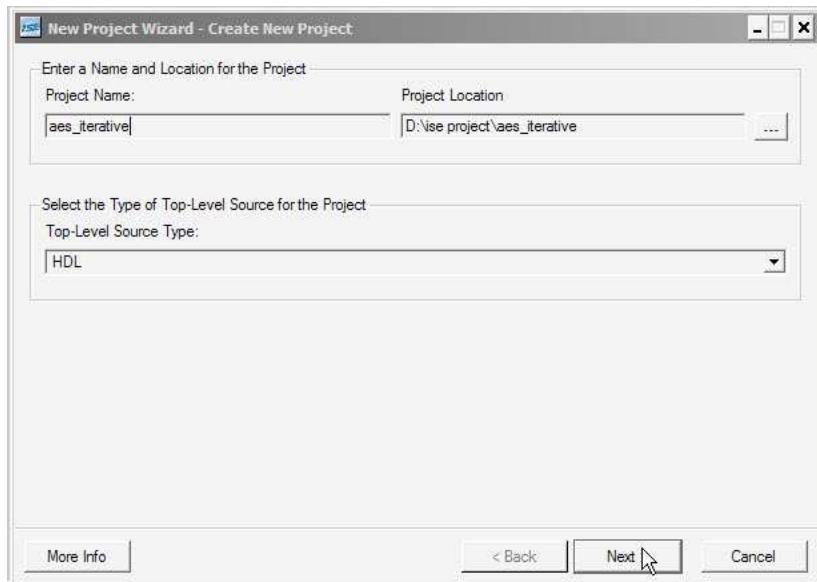


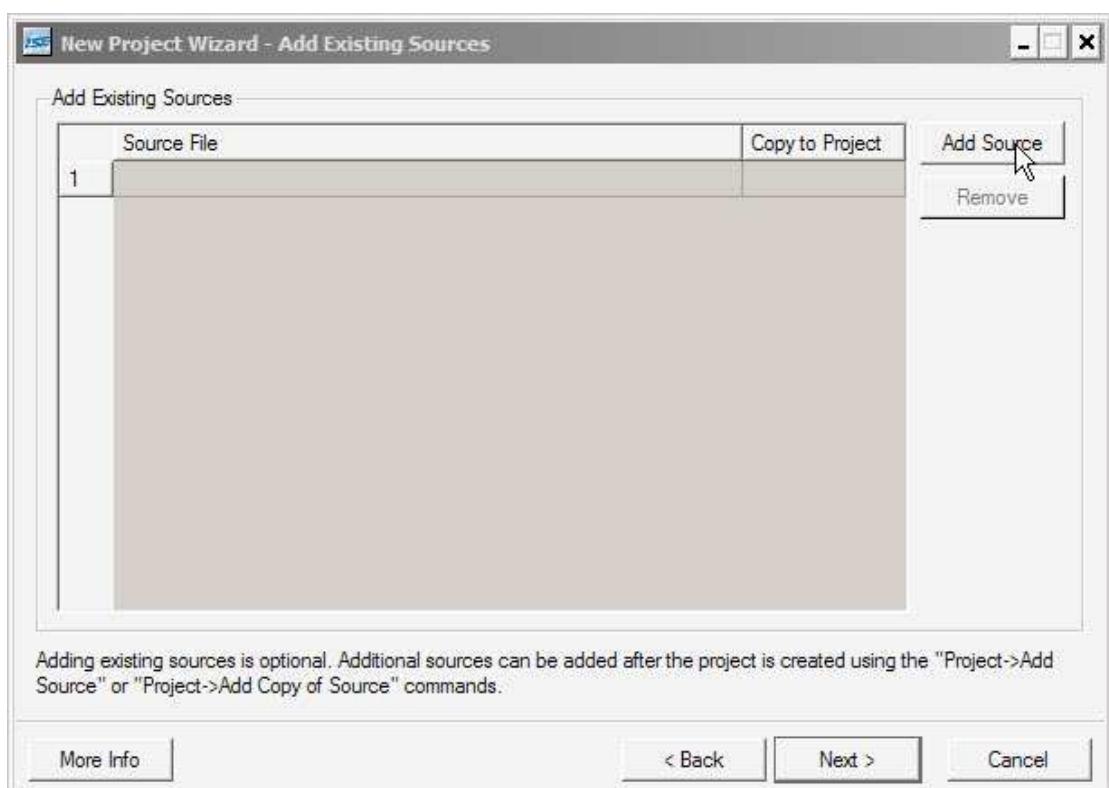
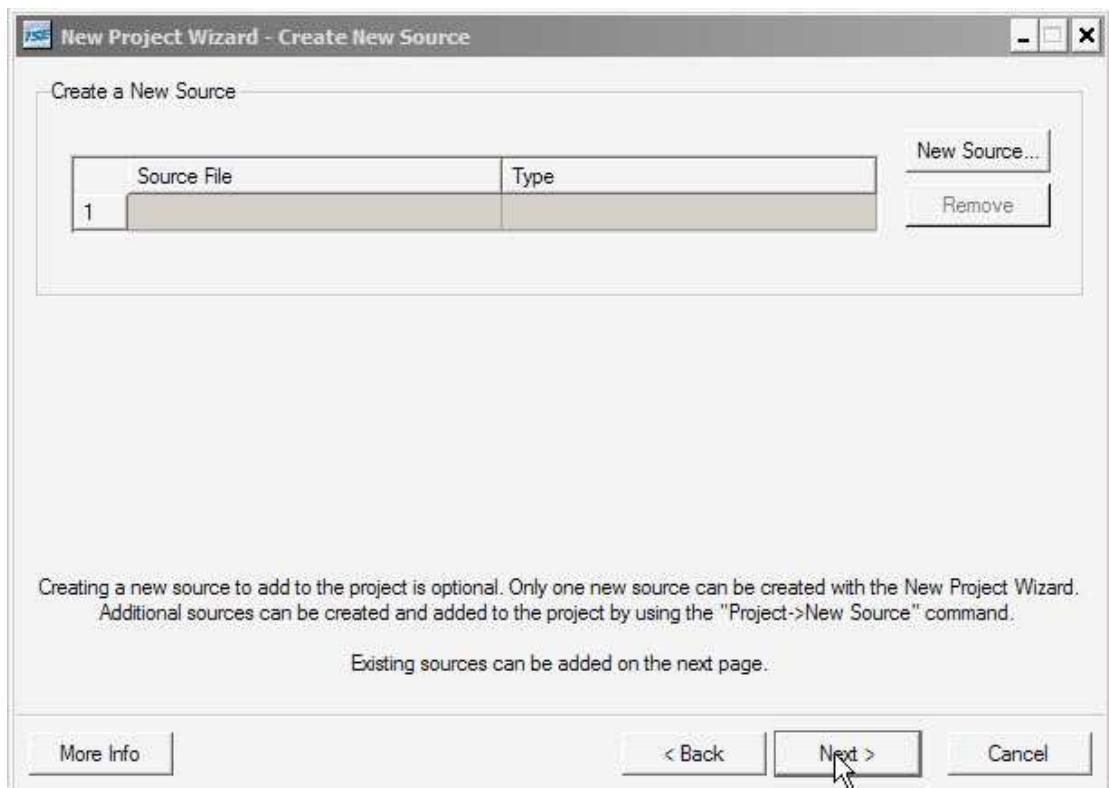
Σημείωση : αν έχουν εγκατασταθεί θέματα στα window , για την σωστή λειτουργία του προγράμματος τα θέματα πρέπει να απενεργοποιηθούν για αυτή την εφαρμογή  
Ιδιότητες →Συμβατότητα →Απενεργοποίηση οπτικών θεμάτων.

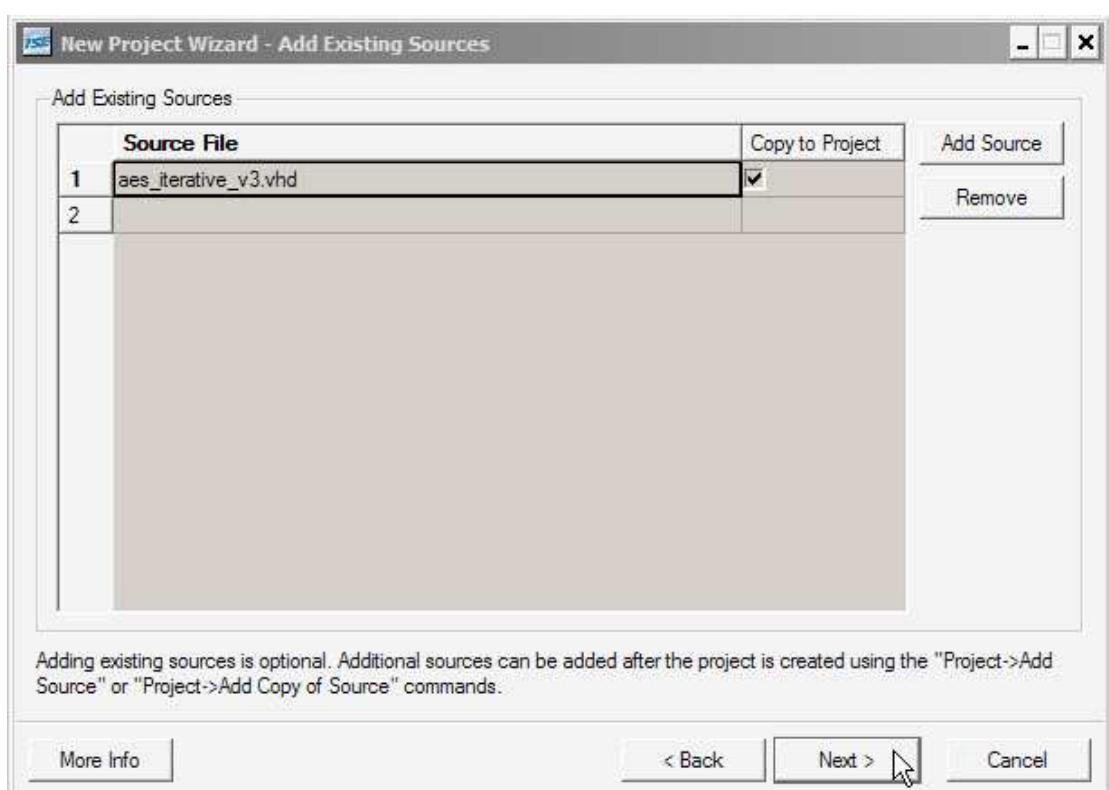
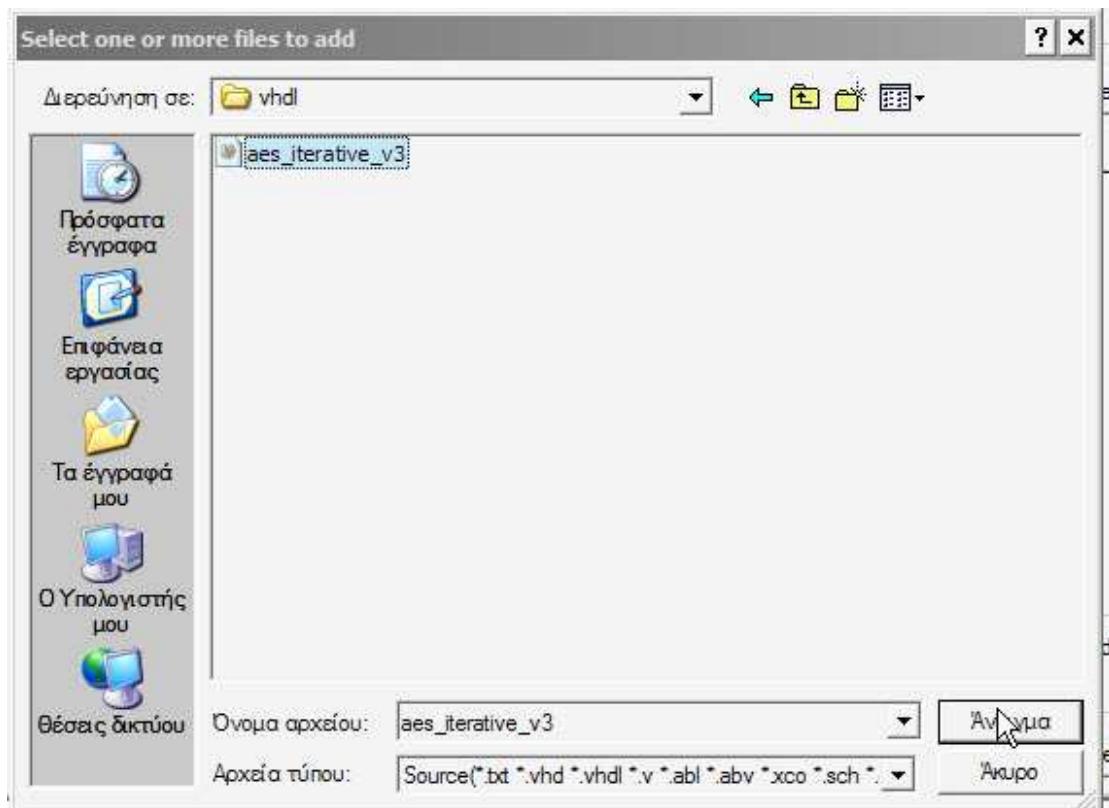
Στην συνέχεια :



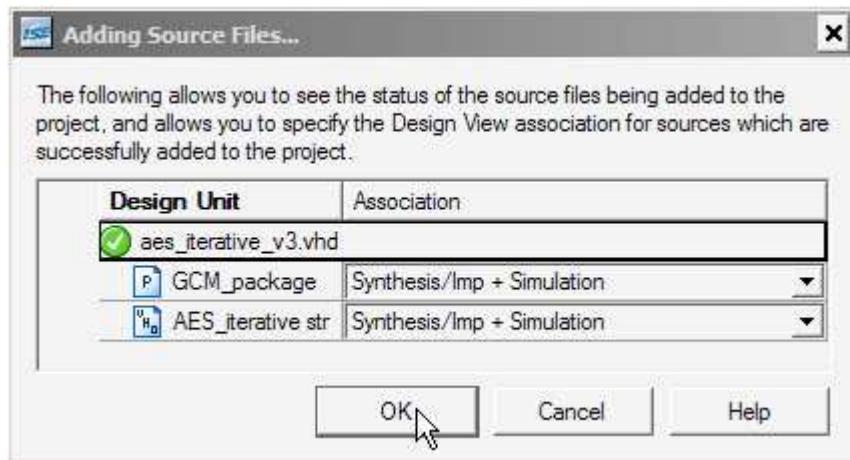
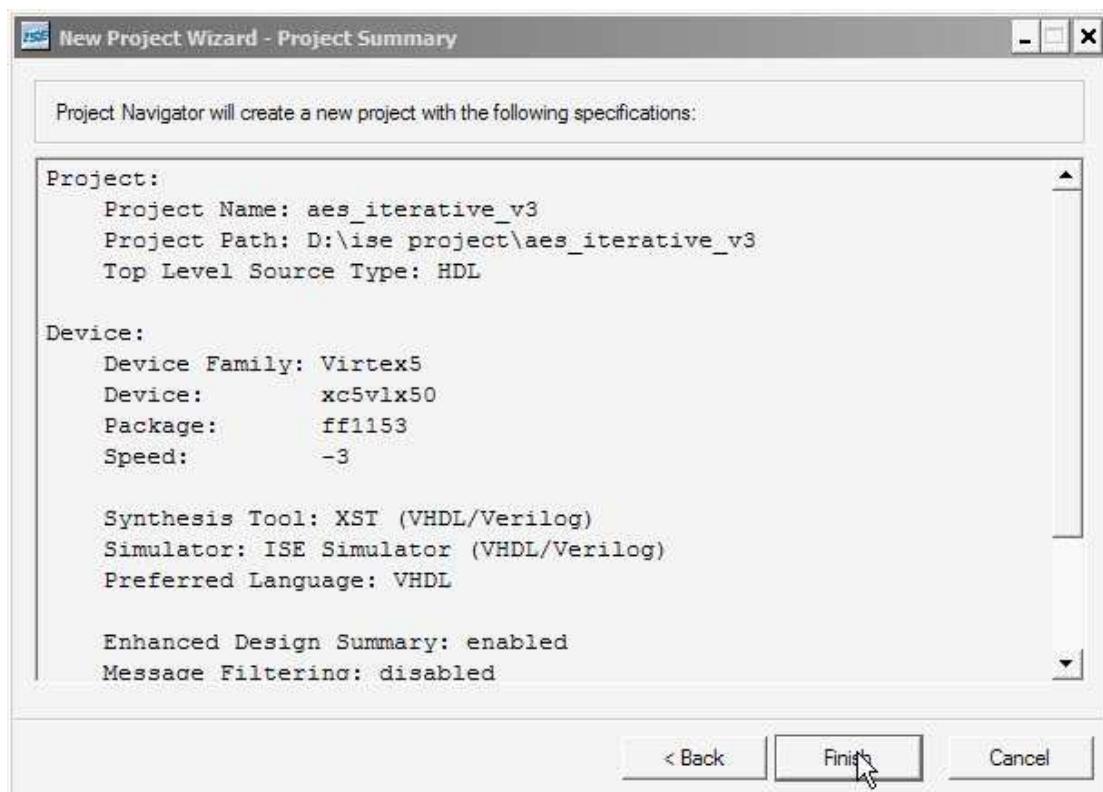
Συμπληρώνουμε το επιθυμητό όνομα του project καθώς επίσης αν θέλουμε αλλάζουμε και το location .



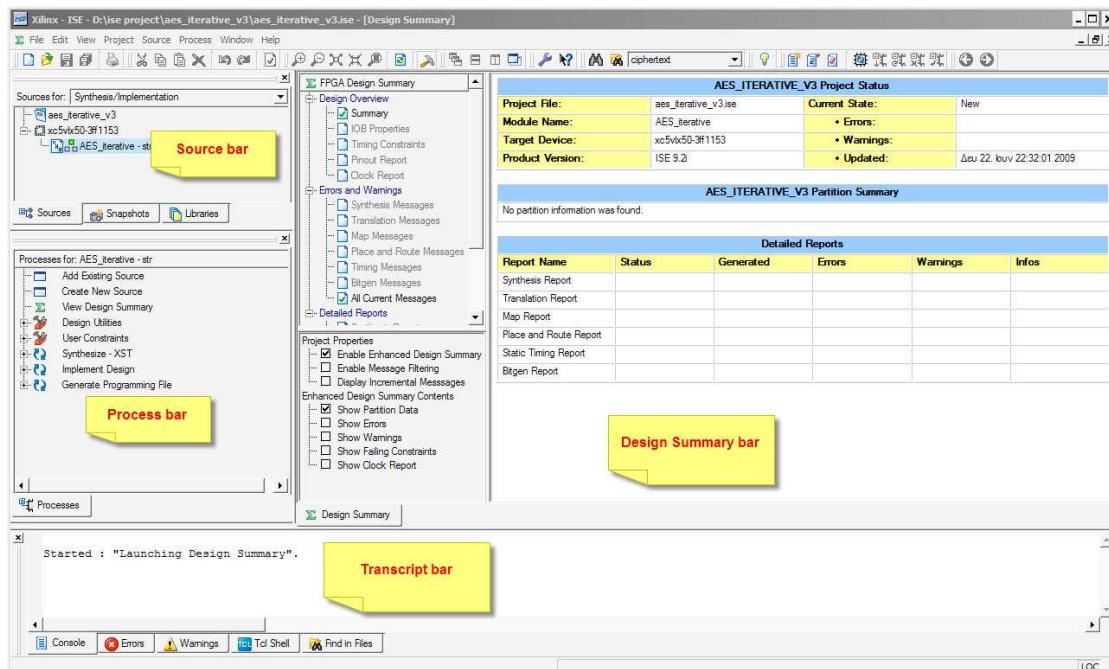




*Σημείωση :Μπορούμε να προσθέσουμε πέρα του ενός αρχείου . Αν πχ χρησιμοποιούνται component ή custom libraries προσθέτουμε τα επιπλέον αρχεία .*



**Σημείωση :** το πρόγραμμα κάνει auto save . Δεν χρειάζεται να κάνουμε save εμείς. Αν δεν βρίσκεται σε στάδιο επεξεργασίας κάποιας process, μπορούμε να εγκαταλείψουμε το πρόγραμμα να πάσα στιγμή και την επόμενη φορά που θα ανοίξουμε το πρόγραμμα , θα φορτωθεί το τελευταίο project στο οποίο δουλεύαμε χωρίς να έχουν χαθεί τα δεδομένα.



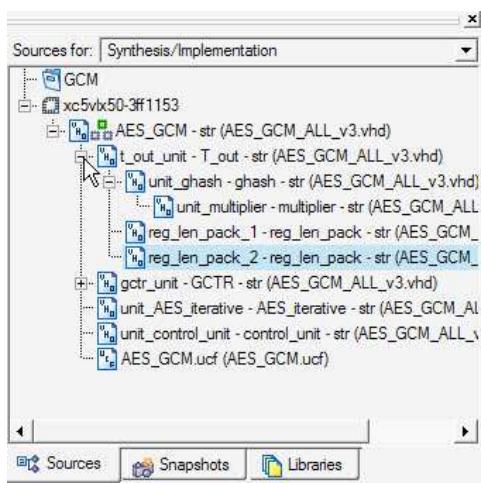
Βλέπουμε ότι έχει προστεθεί το αρχείο που θέλουμε .

Από το process bar επιλέγουμε την διεργασία που θέλουμε να εκτελεστεί.

Στο Design Summary bar βλέπουμε σε ποιο στάδιο βρισκόμαστε και άλλες πληροφορίες .(εκτενής αναφορά γίνεται παρακάτω )

Στο Transcript bar μπορούμε να παρακολουθούμε την διεργασία που εκτελείται σε real time.

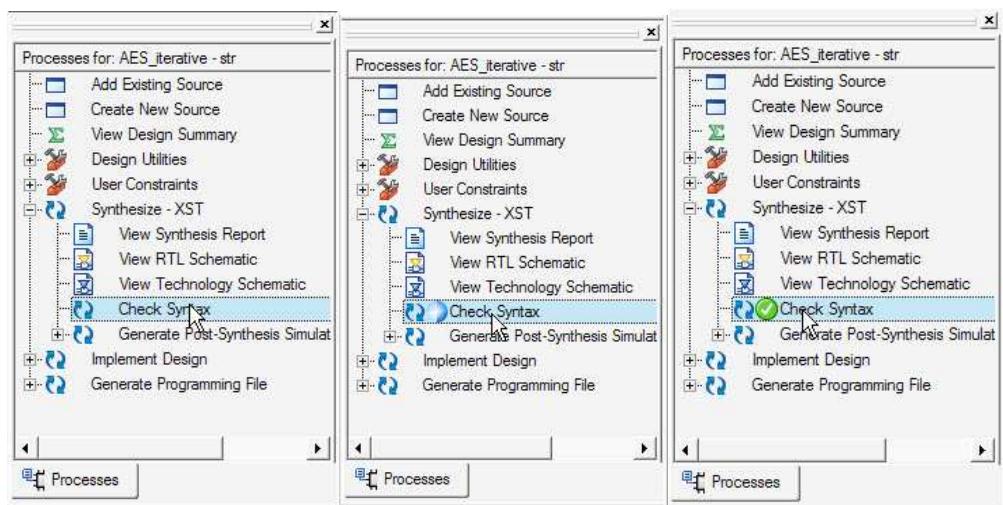
Από το source bar , μπορούμε να επέμβουμε στο κώδικα καθώς επίσης και να αλλάξουμε ενδεχομένως family και device . Στο source bar εμφανίζεται το component που είναι στο υψηλότερο επίπεδο. Για να δούμε τα components χαμηλότερου επιπέδου έχουμε:



Τέλος δίνεται η δυνατότητα να προσθέσουμε άλλο κώδικα ή να φτιάξουμε καινούριο κώδικα .

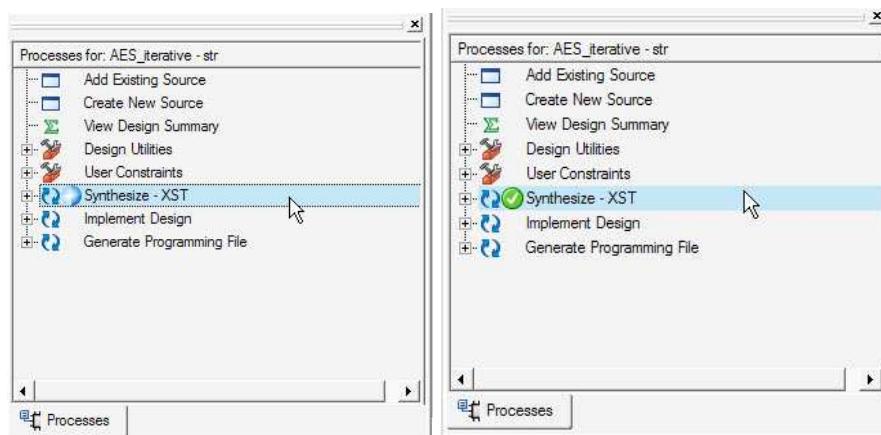
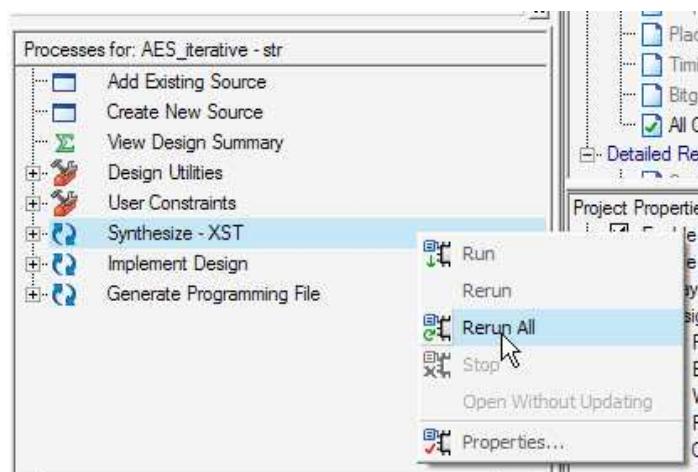
Το πρώτο βήμα μετά την εισαγωγή κώδικα είναι να ελέγξουμε τον κώδικα για τυχών συντακτικά λάθη . Από το process bar εκτίνουμε το Synthesize XST και επιλέγουμε → Check Syntax με διπλό click.

Αν υπάρχουν λάθη, πρέπει να διορθωθούν.



Στην συνέχεια :

## Synthesize step



Μόλις τελειώσει η διεργασία στο Summary bar περνούμε :

The screenshot shows the Xilinx ISE 9.2 interface with the 'FPGA Design Summary' window open. On the left, there's a tree view of reports under 'Design Overview' and 'Errors and Warnings'. A 'Project Properties' section is also visible. The main area displays the 'AES\_ITERATIVE\_V3 Project Status' and 'AES\_ITERATIVE\_V3 Partition Summary' tables. Arrows point from the 'Current State' row in the status table to the 'Synthesized', 'No Errors', and 'No Warnings' columns. Another arrow points from the 'Utilization' column in the partition summary table to the 'Detailed Reports' table below. The 'Detailed Reports' table lists 'Synthesis Report', 'Translation Report', 'Map Report', and 'Place and Route Report', with the 'Synthesis Report' row currently selected.

Αν υπάρχουν warring ή errors τότε μπορούμε να κάναμε click πάνω του και εμφανίζονται. (περισσότερες λεπτομέρειες παρακάτω).

This screenshot shows the same 'FPGA Design Summary' window, but the 'Synthesis Report' item in the 'Detailed Reports' section of the tree view is now highlighted with a blue selection bar and has a cursor pointing at it. The other report items (Translation Report, Map Report, Place and Route Report, Static Timing Report, Bitgen Report) are shown below it.

Επιλέγουμε → για να δούμε αναλυτικά τα αποτελέσματα.

Προς το τέλος της αναφοράς υπάρχει και το εξής :

```
Timing Summary:  
-----  
Speed Grade: -3  
  
Minimum period: 3.184ns (Maximum Frequency: 314.058MHz)  
Minimum input arrival time before clock: 2.703ns  
Maximum output required time after clock: 2.520ns  
Maximum combinational path delay: No path found  
  
Timing Detail:  
-----  
All values displayed in nanoseconds (ns)  
=====
```

Η συχνότητα που μας δίνει **δεν** είναι η συχνότητα λειτουργίας του κυκλώματος αλλά η μέγιστη εσωτερική συχνότητα, δηλαδή εσωτερικά το κύκλωμα που αργεί περισσότερο. Για να βρούμε την συχνότητα λειτουργίας, αρκεί να προσθέσουμε αυτά τα νούμερα. (8,407 ns) → **118,9 MHz** συχνότητα λειτουργίας.

Αν υπάρχουν components ασύγχρονα τότε η γραμμή Maximum combinational path delay εμφανίζει την μέγιστη καθυστέρηση αυτών των κυκλωμάτων .

Αμέσως ποιο κάτω στην αναφορά υπάρχουν και τα κρίσιμα μονοπάτια .

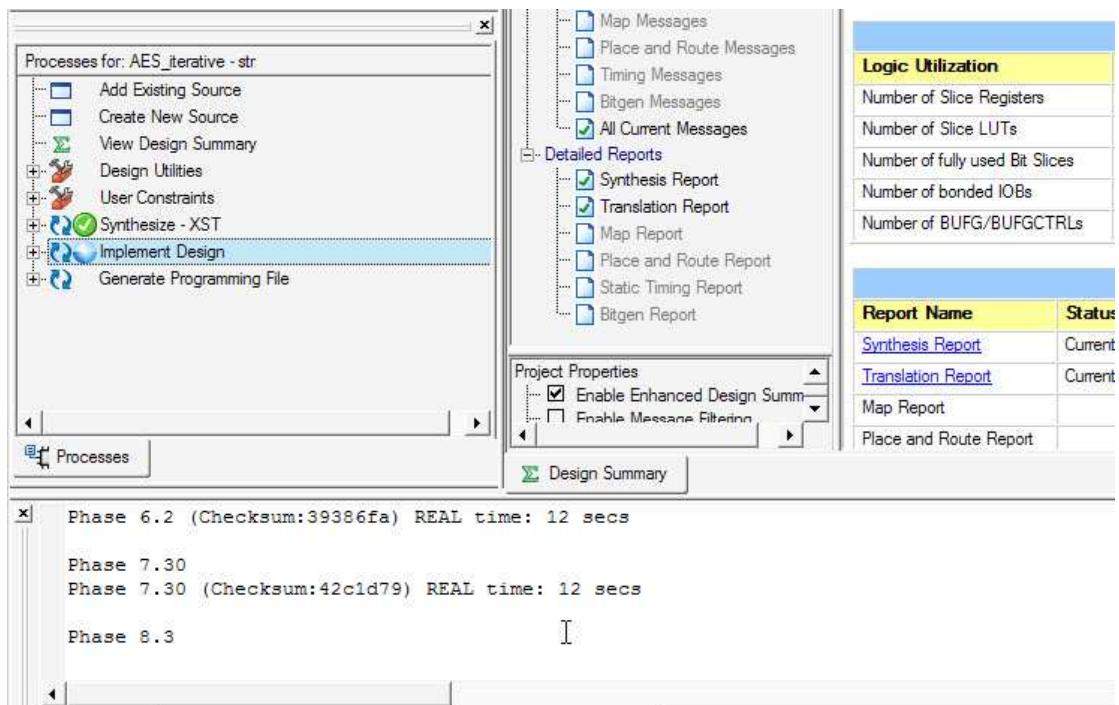
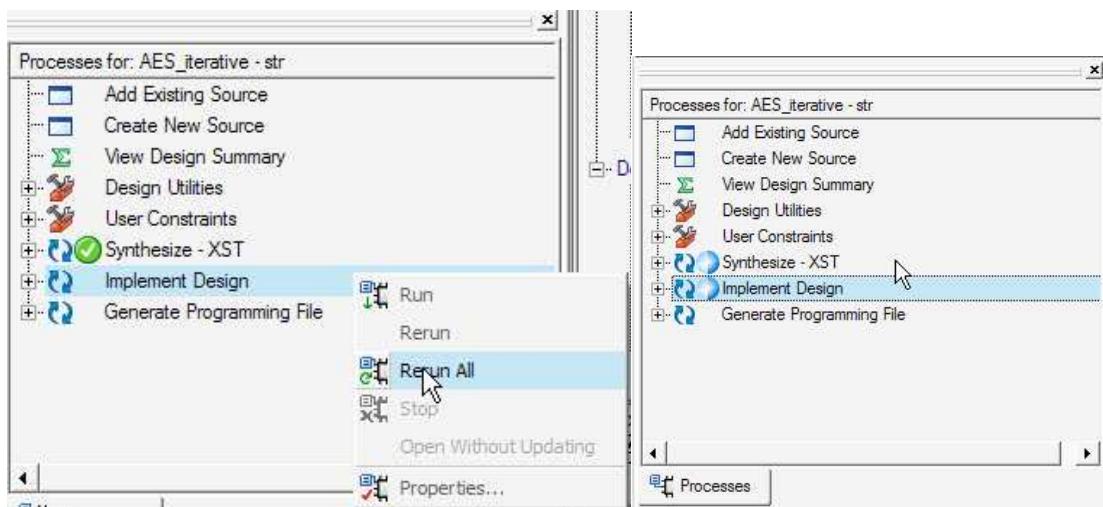
*Σημείωση : Αν υπάρχουν errors πρέπει να διορθωθούν. Αν υπάρχουν warnings μερικά θεωρούνται αποδεκτά ενώ υπάρχουν άλλα που πρέπει να διορθωθούν. (πχ αν υπάρχουν signal ή variables στον κώδικα που δεν χρησιμοποιούνται πρέπει να διορθωθούν διότι αυτό μεταφράζεται σε επιπλέον καθυστέρηση).*

*Σημείωση : Σε περίπτωση που το συγκεκριμένο κύκλωμα δεν χωράει σε αυτή την συσκευή πρέπει να επιλέξουμε μεγαλύτερη συσκευή και ενδεχομένως και άλλη οικογένεια συσκευών. Ειδάλλως δεν μπορούμε να προχωρήσουμε στο επόμενο βήμα .*

*Σημείωση : Ένα από τα αρνητικά του προγράμματος είναι ότι σχεδόν απ' όλες τις οικογένειες συσκευών απονοσιάζουν τα μεγάλα device .*

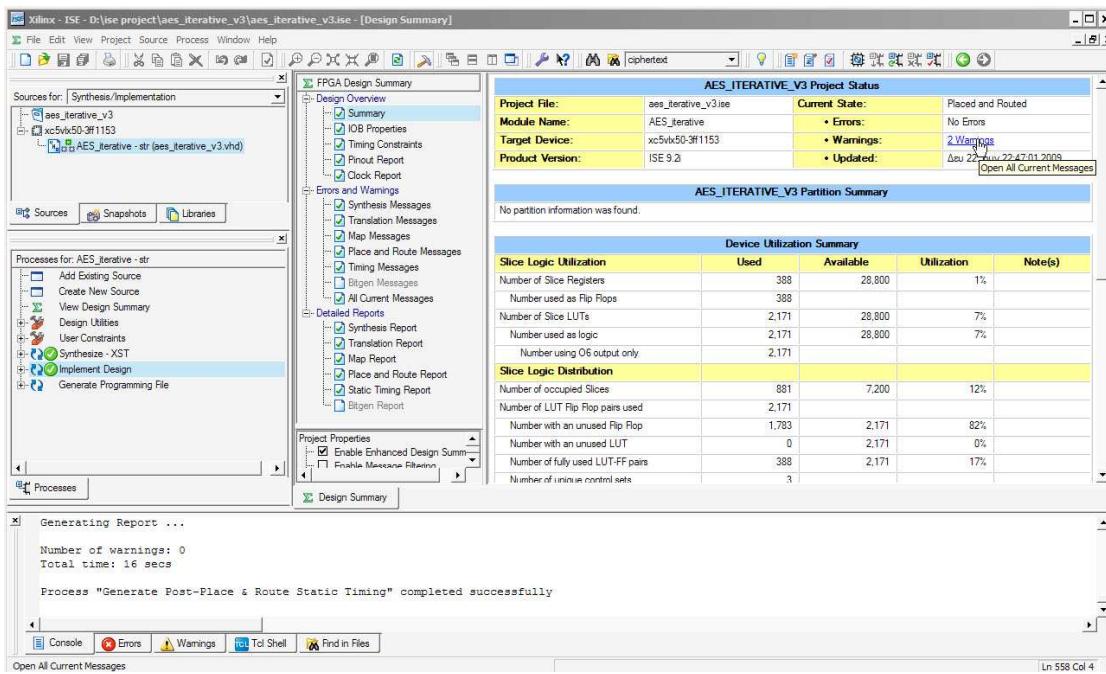
Στην συνέχεια :

## Implementation step



Στο Transcript bar παρατηρούμε ότι τρέχει το κύκλωμα σε πραγματικό χρόνο

Όταν ολοκληρωθεί η process παίρνουμε :



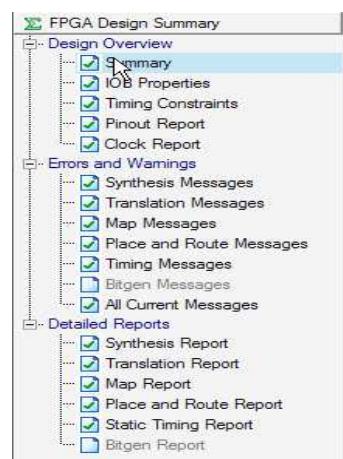
Πατώντας πάνω στα warnings ανοίγει η καρτέλα Errors and Warnings → All Current Messages

```

xst  [i] Xst:2146 - In block <MTP_>, ROM <Mrrom_cells_a_7_rom0000>
xst  [i] Xst:2146 - In block <MTP_>, ROM <Mrrom_cells_b_10_rom0001>
xst  [i] Xst:2146 - In block <MTP_>, ROM <Mrrom_cells_a_2_rom0000>
map  [!] LIT:243 - Logical network N19211 has no load.
map  [!] LIT:395 - The above warning message base_net_load_rule is repeated 1 more times for the following (max. 5 sh...
      N19212
      To see the details of these warning messages, please use the -detail switch.
map  [i] MapLib:562 - No environment variables are currently set.
map  [i] LIT:244 - All of the single ended outputs in this design are using slew rate limited output drivers. The delay on spe...
map  [i] Pack:1716 - Initializing temperature to 85.000 Celsius. (default - Range: 0.000 to 85.000 Celsius)
map  [i] Pack:1720 - Initializing voltage to 0.950 Volts. (default - Range: 0.950 to 1.050 Volts)
map  [i] Pack:1650 - Map created a placed design.
map  [i] Timing:3284 - This timing report was generated using estimated delay information. For accurate numbers, please r...

```

Σε περίπτωση που τα warnings οφείλονται στο χρήστη πρέπει να διορθωθούν (υπάρχουν κάποια που θεωρούνται αποδεκτά) Σε ορισμένα warnings το πρόγραμμα σε παραμπαίνει στην ιστοσελίδα του κατασκευαστή όπου με βάση τον κωδικό σφάλματος μπορεί κάποιος να βρει περισσότερες πληροφορίες.



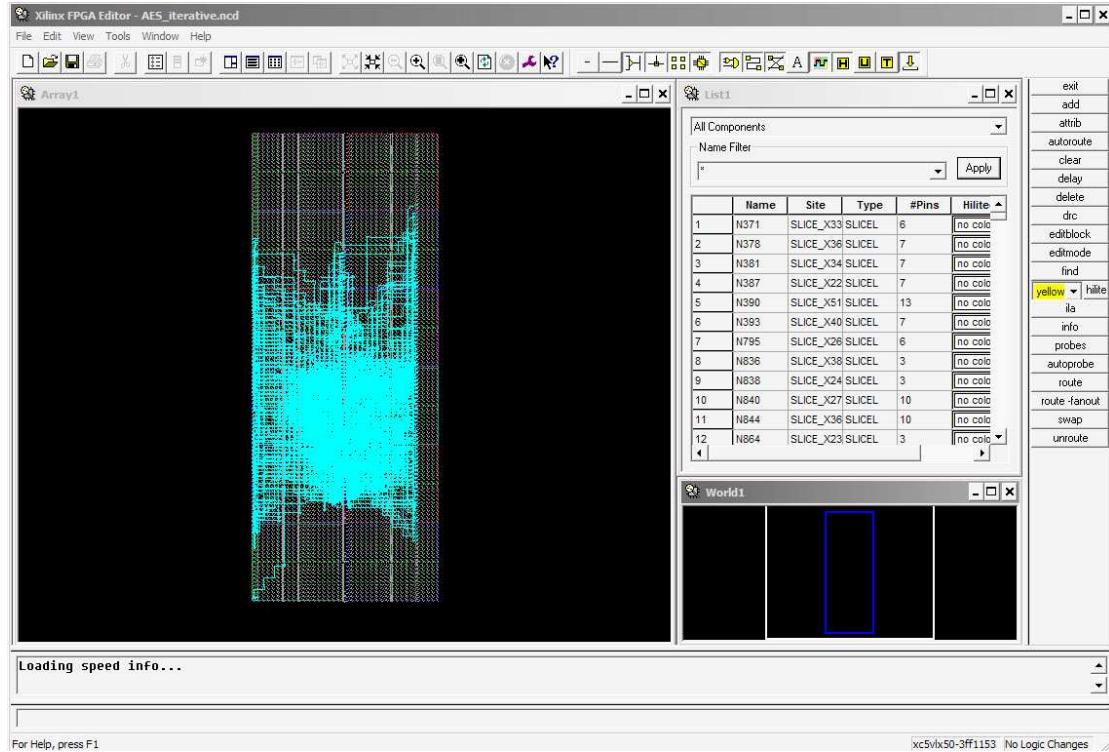
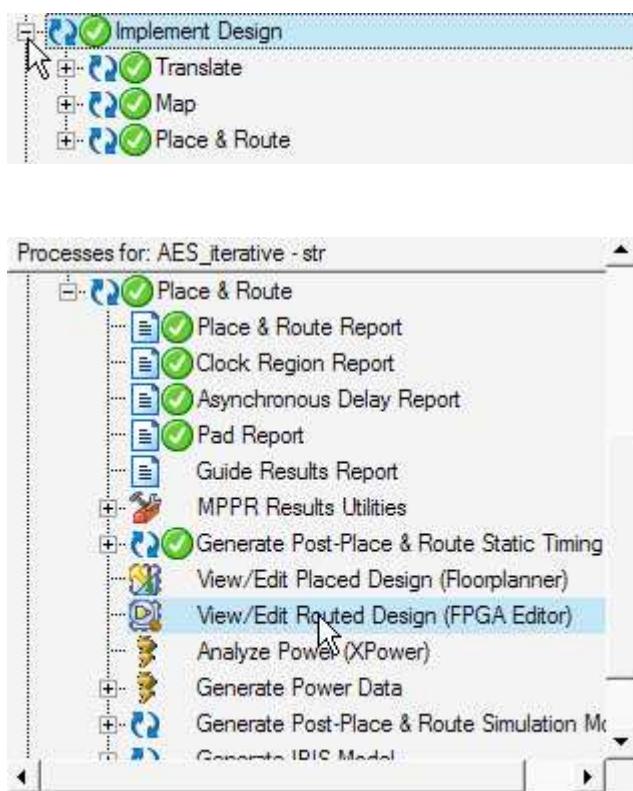
AES_ITERATIVE_V3 Project Status			
<b>Project File:</b>	aes_iterative_v3.ise	<b>Current State:</b>	Placed and Routed 
<b>Module Name:</b>	AES_iterative	<b>• Errors:</b>	No Errors
<b>Target Device:</b>	xc5vlx50-3ff1153	<b>• Warnings:</b>	2 Warnings
<b>Product Version:</b>	ISE 9.2i	<b>• Updated:</b>	Δευ 22. Ιουν 22:47:01 2009

Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	388	28,800	1%	
Number used as Flip Flops	388			
Number of Slice LUTs	2,171	28,800	7%	
Number used as logic	2,171	28,800	7%	
Number using O6 output only	2,171			
Slice Logic Distribution				
Number of occupied Slices	881	7,200	12%	
Number of LUT Flip Flop pairs used	2,171			
Number with an unused Flip Flop	1,783	2,171	82%	
Number with an unused LUT	0	2,171	0%	
Number of fully used LUT-FF pairs	388	2,171	17%	
Number of unique control sets	3			
IO Utilization				
Number of bonded IOBs	386	560	68%	
Specific Feature Utilization				
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Total equivalent gate count for design	18,781			
Additional JTAG gate count for IOBs	18,528			

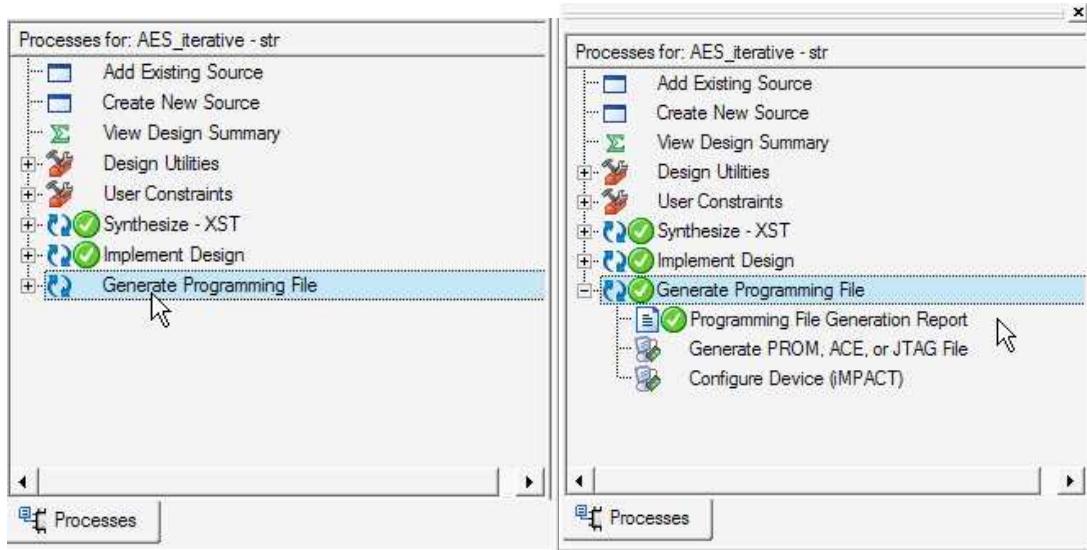
Performance Summary			
<b>Final Timing Score:</b>	0	<b>Pinout Data:</b>	<a href="#">Pinout Report</a>
<b>Routing Results:</b>	All Signals Completely Routed	<b>Clock Data:</b>	<a href="#">Clock Report</a>
<b>Timing Constraints:</b>	All Constraints Met		

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
<a href="#">Synthesis Report</a>	Current	Δευ 22. Ιουν 22:42:16 2009	0	0	<a href="#">20 Infos</a>
<a href="#">Translation Report</a>	Current	Δευ 22. Ιουν 22:42:23 2009	0	0	0
<a href="#">Map Report</a>	Current	Δευ 22. Ιουν 22:45:59 2009	0	<a href="#">2 Warnings</a>	<a href="#">7 Infos</a>
<a href="#">Place and Route Report</a>	Current	Δευ 22. Ιουν 22:46:43 2009	0	0	<a href="#">3 Infos</a>
<a href="#">Static Timing Report</a>	Current	Δευ 22. Ιουν 22:47:01 2009	0	0	<a href="#">3 Infos</a>
Bitgen Report					

Επιπλέον δυνατότητες :



## Generate Programmable File



Αν ανοίξουμε τον φάκελο που βρίσκεται το project βλέπουμε ότι εκτός των άλλων έχει παράγει και ένα αρχείο .bit που είναι και το αρχείο που κατεβαίνει στο FPGA.



Καθώς επίσης και δύο html αρχεία .

AES_ITERATIVE_V3 Project Status						
Project File:	aes_iterative_v3.ise	Current State:	Programming File Generated			
Module Name:	AES_iterative	• Errors:		No Errors		
Target Device:	xc5vh50-5ft1153	• Warnings:		2 Warnings		
Product Version:	ISE 9.2i	• Updated:		???.???.???.???.23:21:30 2009		
AES_ITERATIVE_V3 Partition Summary						
No partition information was found.						
Device Utilization Summary						
Slice Logic Utilization	Used	Available	Utilization	Note(s)		
Number of Slice Registers	388	28,800	1%			
Number used as Flip Flops	388					
Number of Slice LUTs	2,171	28,800	7%			
Number used as logic	2,171	28,800	7%			
Number using O6 output only	2,171					
Slice Logic Distribution	Used	Available	Utilization	Note(s)		
Number of occupied slices	81	7,200	12%			
Number of LUT Flip Flop pairs used	2,171					
Number with an unused Flip Flop	1,783	2,171	82%			
Number with an unused LUT	0	2,171	0%			
Number of fully used LUT-FF pairs	388	2,171	17%			
Number of unique control sets	3					
IO Utilization	Used	Available	Utilization	Note(s)		
Number of bonded IOBs	386	560	68%			
Specific Feature Utilization	Used	Available	Utilization	Note(s)		
Number of BUFGCTRLs	1	32	3%			
Number used as BUFGs	1					
Total equivalent gate count for design	18,811					
Additional JTAG gate count for IOBs	18,528					

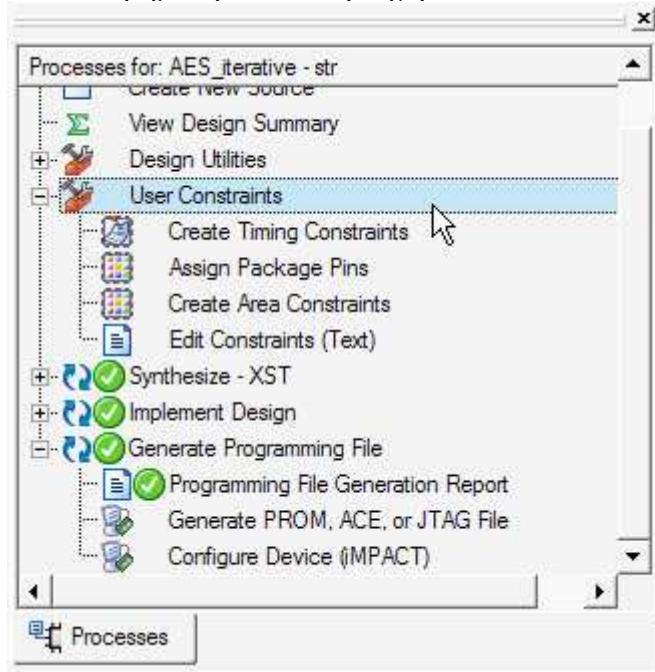
## Device Usage Page (device\_usage\_statistics.html)

This HTML page displays the device usage statistics that will be sent to Xilinx. The file also contains predefined XML tags used to simplify processing.

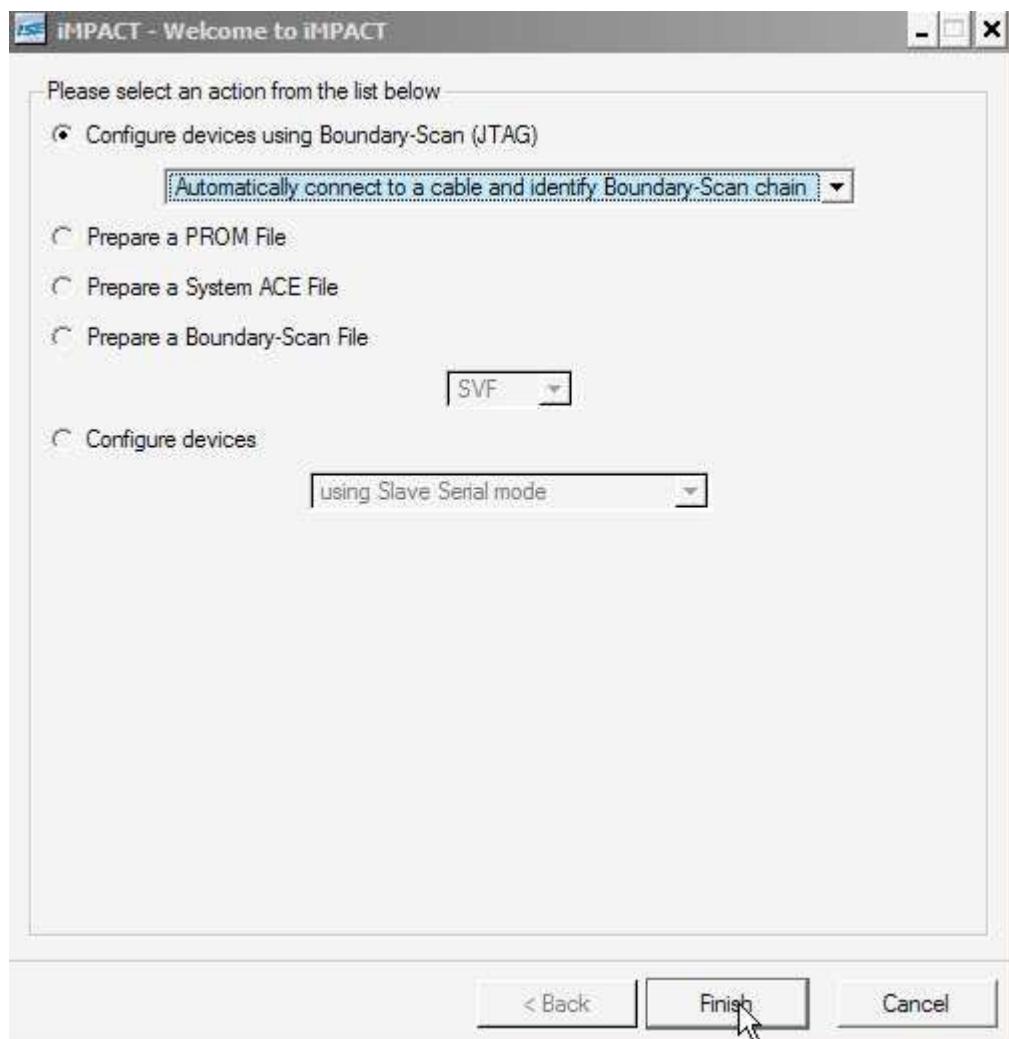
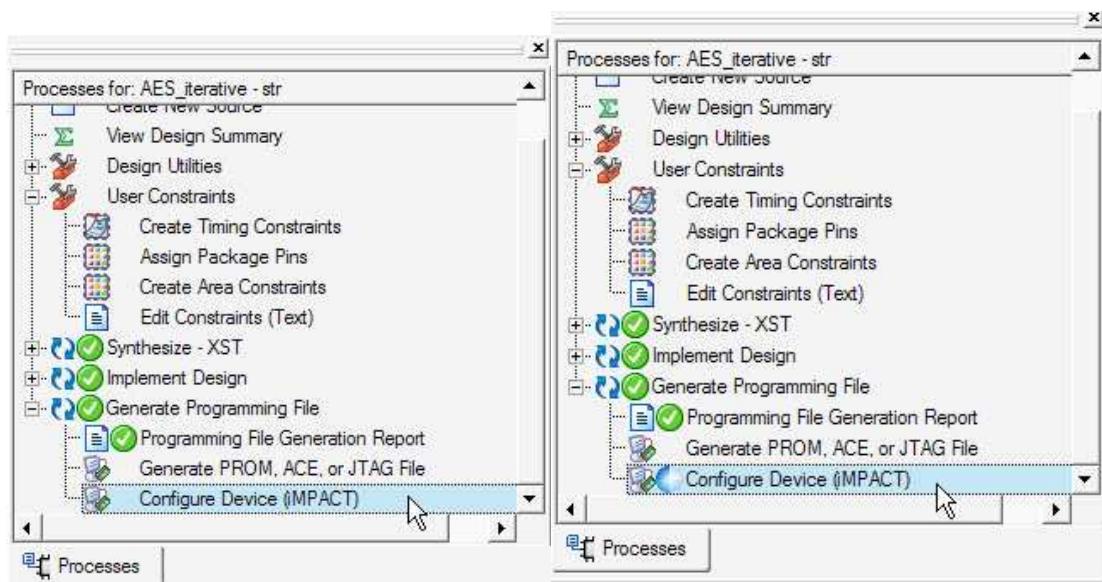
Please verify the contents are okay to send to Xilinx!

Software Version and Target Device			
Product Version:	ISE 9.2i	Target Family:	virtex5
OS Platform:	NT	Target Device:	xc5v8x50
Project ID (random number)	25675.1	Target Package:	ff1153
Date Generated	???.???.???? 23:21:30 2009	Target Speed:	-3
Device Usage Statistics			
Macro Statistics	Miscellaneous Statistics	Net Statistics	Site Usage
Adders/Subtractors=1  • 4-bit adder=1	MiscellaneousStatistics  • AGG_BONDED_JIO=386 • AGG_IO=386 • AGG_SLICE=881 • NUM_BONDED_JOB=386 • NUM_BUFG=1 • NUM_EQUIV_GATE=18781 • NUM_EQUIV_GLOBAL=54 • NUM_EQUIV_LOGIC_O6ONLY=2171 • NUM_EQUIV_SLICE=881 • NUM_EQUIV_WIRE=16 • NUM_HLONG=251 • NUM_INPUT=12174 • NUM_OUTPUT=9820 • NUM_SLICEL=881 • NUM_SLICE=2171 • NUM_WIRE=16 Registers=388  • Flip-Flops=388	NetStatistics  • NumNets_Active=2320 • NumNodesOfType_Active_BOUNCEACROSS=569 • NumNodesOfType_Active_BOUNCEIN=1699 • NumNodesOfType_Active_BUFGOUT=1 • NumNodesOfType_Active_CLKPIN=282 • NumNodesOfType_Active_CNTRLPIN=394 • NumNodesOfType_Active_DOUBLE=9820 • NumNodesOfType_Active_GENERIC=8 • NumNodesOfType_Active_GLOBAL=54 • NumNodesOfType_Active_HLONG=251 • NumNodesOfType_Active_INPUT=12174 • NumNodesOfType_Active_OUTPUT=9820 • NumNodesOfType_Active_BUFGOUT=1 • IOB_BUFG=1 • IOB_BUFG_BUFG=1 • IOB_IBUF=386 • IOB_INBUF=258 • IOB_OUTBUF=128 • IOB_PAD=386 • SLICEL=881 • SLICEL_A6LUT=660 • SLICEL_AFF=169 • SLICEL_B6LUT=560 • SLICEL_BFF=96 • SLICEL_C6LUT=485	SiteSummary  • BUFG=1 • BUFG_BUFG=1 • IOB=386 • IOB_IBUF=258 • IOB_OUTBUF=128 • IOB_PAD=386 • SLICEL=881 • SLICEL_A6LUT=660 • SLICEL_AFF=169 • SLICEL_B6LUT=560 • SLICEL_BFF=96 • SLICEL_C6LUT=485

Σε περίπτωση που επιθυμούμε στο device να χρησιμοποιούνται συγκεκριμένες θύρες, η επιλογή αυτή δίνεται :  
(αυτό το βήμα πρέπει να προηγηθεί όλων των άλλων)



Στην συνέχεια δίνεται η δυνατότητα να ενώσουμε το board με τον υπολογιστή και το πρόγραμμα να αναλάβει το κατέβασμα στο FPGA.



Πραγματική συχνότητα λειτουργίας :

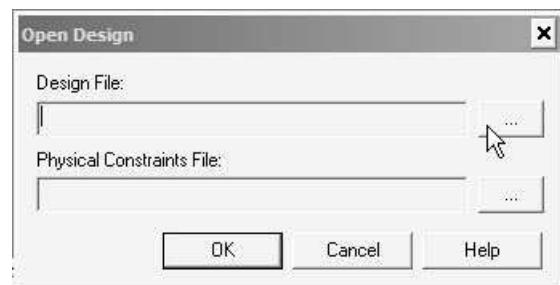
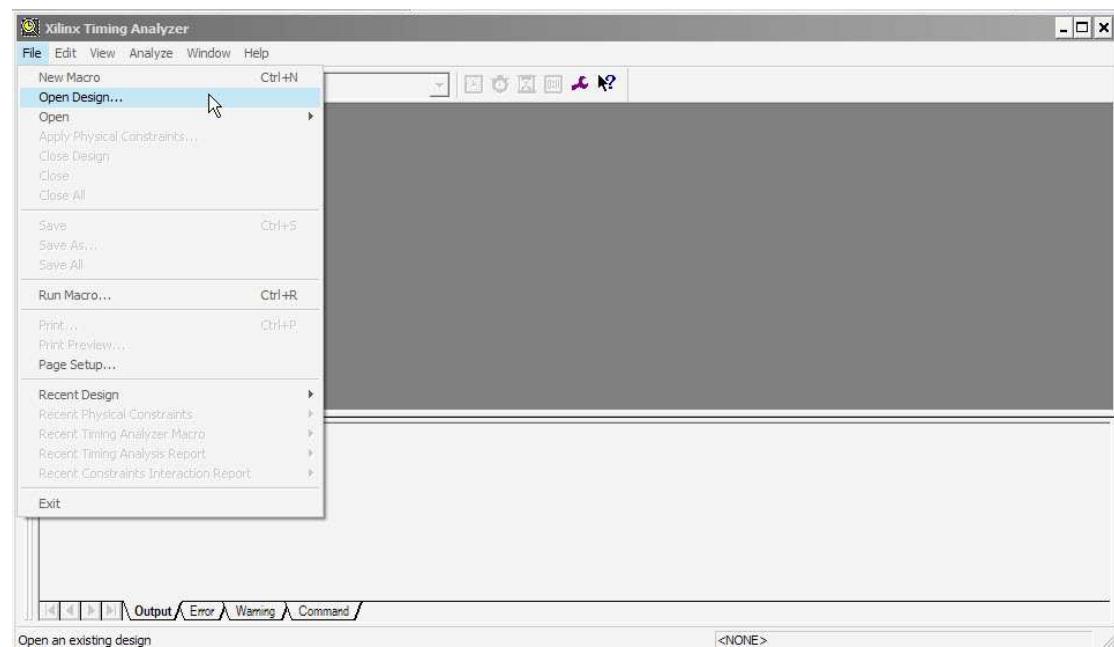
Η συχνότητα που δόθηκε παραπάνω είναι η συχνότητα έτσι όπως υπολογιστικέ από το πρόγραμμα βάζοντας καθυστερήσεις σε πύλες κτλ προκειμένου να δώσει μία εκτίμηση για την συχνότητα λειτουργίας και αποτελεί την συχνότητα με την οποία το κύκλωμα δουλεύει ορθά . Αν θέλουμε να μάθουμε την συχνότητα λειτουργία του κυκλώματος που θα κατέβει σε FPGA υπάρχει τρόπος.

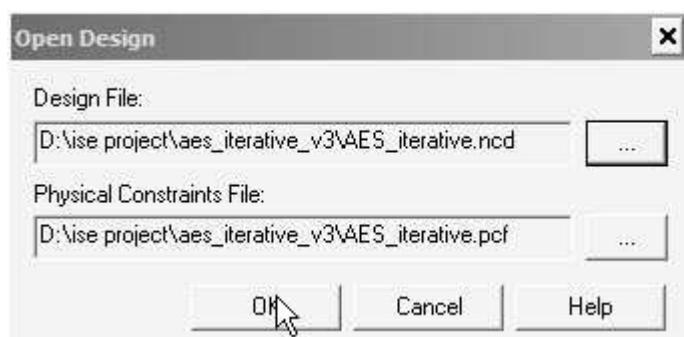
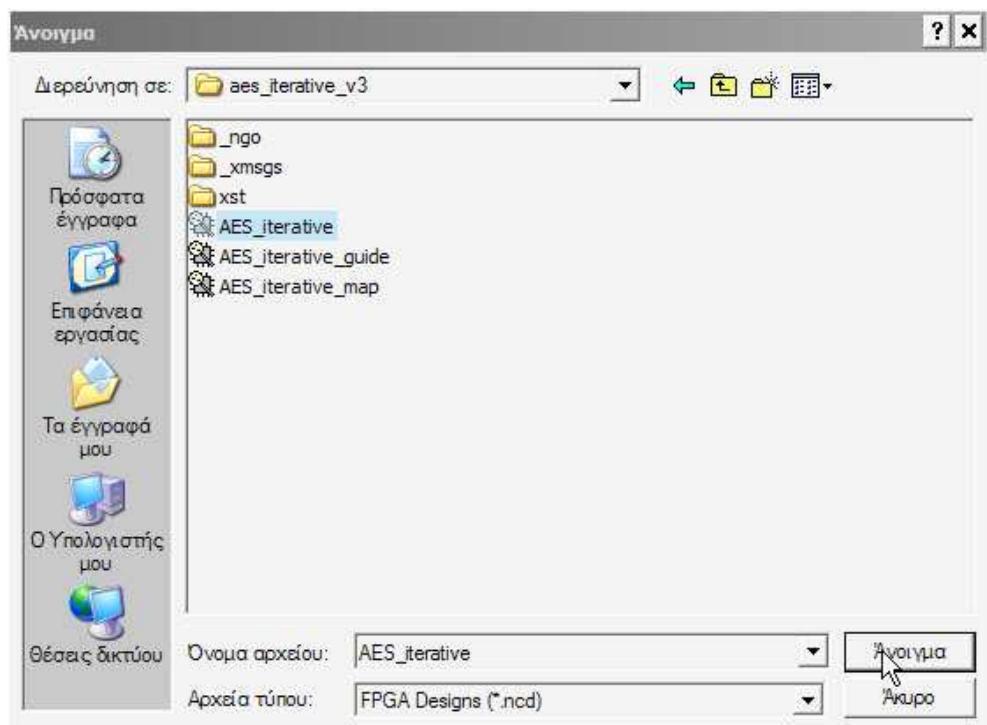
**Σημείωση : Η συχνότητα αυτή είναι το κατώφλι λειτουργίας !**

Σε καμία περίπτωση το κύκλωμα δεν θα δουλέψει με συχνότητα μεγαλύτερη από αυτή. Επίσης η συχνότητα αυτή αποτελεί την καλύτερη περίπτωση σε ιδανικές θερμοκρασίες και με ιδανικές τάσεις. Δεν θα πρέπει να θεωρηθεί η συχνότητα αυτή ως η ασφαλής συχνότητα λειτουργίας..

Απαραίτητη προϋπόθεση είναι ο κύκλωμα να έχει φτάσει μέχρι το στάδιο place and route επιτυχώς.

Start → All programs → Xilinx ISE 9.2i → Accessories → Timing Analyzer





Βλέπουμε ότι φορτώνει απευθείας το σχέδιο που θα κατέβει στο FPGA.  
Αποτελεί έτσι την πλέον κοντινή προσέγγιση όσο αφορά την συχνότητα λειτουργίας

Στην συνέχεια :



Και μας βγάζει το παράθυρο :



Όταν τελειώσει η διαδικασία παίρνουμε τα αποτελέσματα.

```
=====
Timing constraint: Default period analysis for net "clk_BUFGP"
76888 items analyzed, 0 timing errors detected. (0 setup errors, 0 hold errors)
Minimum period is 5.000ns.

Delay: 5.000ns (data path - clock path skew + uncertainty)
Source: j_2 (FF)
Destination: cells_b_5_5 (FF)
Data Path Delay: 4.996ns (Levels of Logic = 2)
Clock Path Skew: 0.031ns
Source Clock: clk_BUFGP rising
Destination Clock: clk_BUFGP rising
Clock Uncertainty: 0.035ns

Data Path: j_2 to cells_b_5_5
  Delay type      Delay(ns)  Logical Resource(s)
  -----
  Tcko            0.326    j_2
  net (Fanout=364) 3.336    j<2>
  Tilo            0.080    cells_b_5_mux0001<85>63
  net (Fanout=1)   1.226    cells_b_5_mux0001<85>_map14
  Tas             0.028    cells_b_5_mux0001<85>167
                           cells_b_5_5
  -----
  Total           4.996ns (0.434ns logic, 4.562ns route)
```

Υπάρχουν πολλές επιλογές στο πρόγραμμα όπως να μπορεί να δει κάποιος τον χρόνο από slice σε slice ή να πρόκειται για vhdl το χρόνο του κάθε component . Περισσότερες πληροφορίες στο manual.

## Βιβλιογραφία

- [1] NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation—Methods and Techniques*, December 2001.
- [2] NIST Special Publication 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, November, 2007.  
<http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>
- [3] FIPS Publication 197, *the Advanced Encryption Standard (AES)*, U.S. DoC/NIST, November, 2001.  
<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- [4] D. McGrew, J. Viega, *The Galois/Counter Mode of Operation (GCM)*, May 31, 2005. <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>
- [5] William Stallings, *Cryptography and Network Security* fourth edition. Upper Saddle River, N.J. : Prentice Hall, c2006.
- [6] *High Speed Architecture for Galois/Counter Mode of Operation (GCM)* ,Bo Yang, Sambit Mishra, Ramesh Karri ,ECE Department, Polytechnic University, Brooklyn, NY
- [7] NIST Special Publication 800-38B, *Recommendation for Block Cipher Modes of Operation: the CMAC Authentication Mode*. U.S. DoC/NIST, October 2003.  
[http://csrc.nist.gov/publications/nistpubs/800-38B/SP\\_800-38B.pdf](http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf)
- [8] NIST Special Publication 800-38C, *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*. U.S. DoC/NIST, May 2004.  
[http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C\\_updated-July20\\_2007.pdf](http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf)
- [9] Nicolas Sklavos, Xinmiao Zhang, "Wireless Security and Cryptography: Specifications and Implementations" 2007-03-30.
- [10] A. Satoh. High-speed hardware architectures for authenticated encryption mode gcm. IEEE International Symposium on Circuits and Systems, 2006.
- [11] A. Satoh, T. Sugawara, and T. Aoki. High-Speed Pipelined Hardware Architecture for Galois Counter Mode. In: J. Garay et al. (Eds.) ISC, 2007.
- [12] A. Satoh. High-Speed Parallel Hardware Architecture for Galois Counter Mode. IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007.
- [13] S. Lemsitzer, J. Wolkerstorfer, N. Felber, and M. Braendli. Multi-gigabit GCM-AES Architecture Optimized for FPGAs.

[14] G. Zhou, H. Michalik, and L. Hinsenkamp. Efficient and High-Throughput Implementations of AES-GCM on FPGAs. International Conference on Field-Programmable Technology, 2007.

[15] Xilinx, Virtex-II Platform FPGA User Guide  
<http://www.xilinx.com/support/documentation/virtex-ii.htm>

[16] Test vectors from:  
[http://csrc.nist.gov/groups/ST/toolkit/documents/Examples/AES\\_GCM.pdf](http://csrc.nist.gov/groups/ST/toolkit/documents/Examples/AES_GCM.pdf)