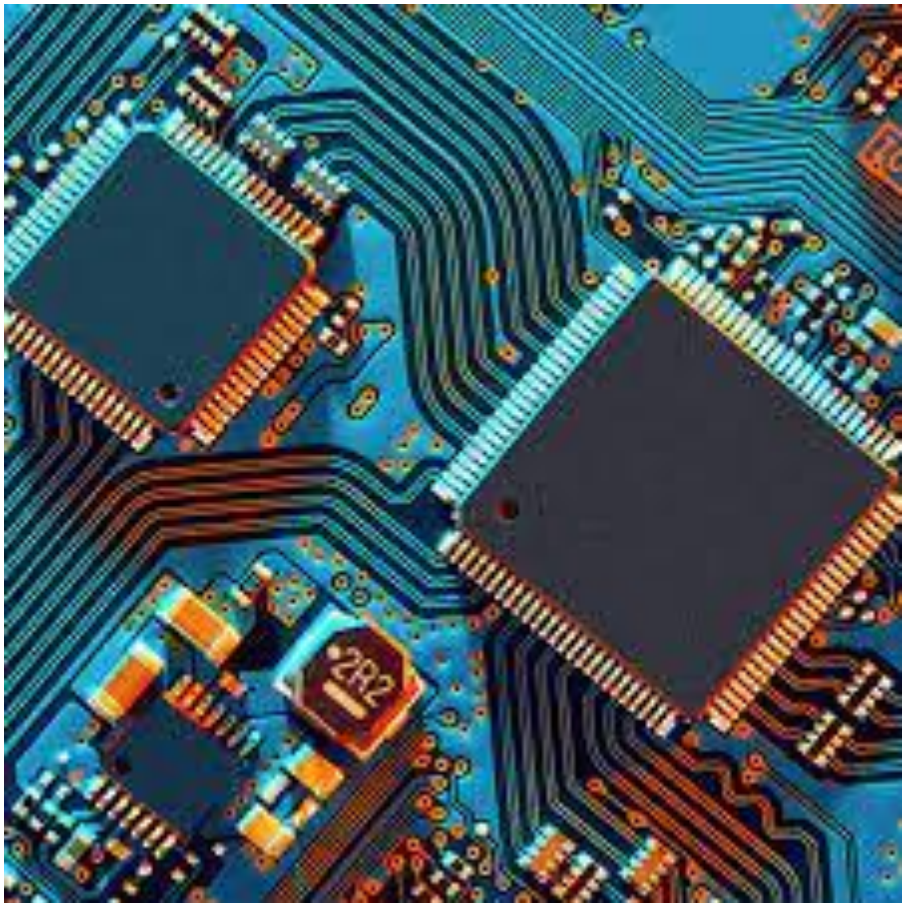


Gost 28147-89

HARDWARE IMPLEMENTATION

ΟΜΑΔΑ 13 | ΣΧΕΔΙΑΣΜΟΣ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ | 02/03/2023

ΟΜΑΔΑ 13

ΚΑΚΑΒΟΥΛΗΣ ΠΑΝΑΓΙΩΤΗΣ
Α.Μ. 1059406

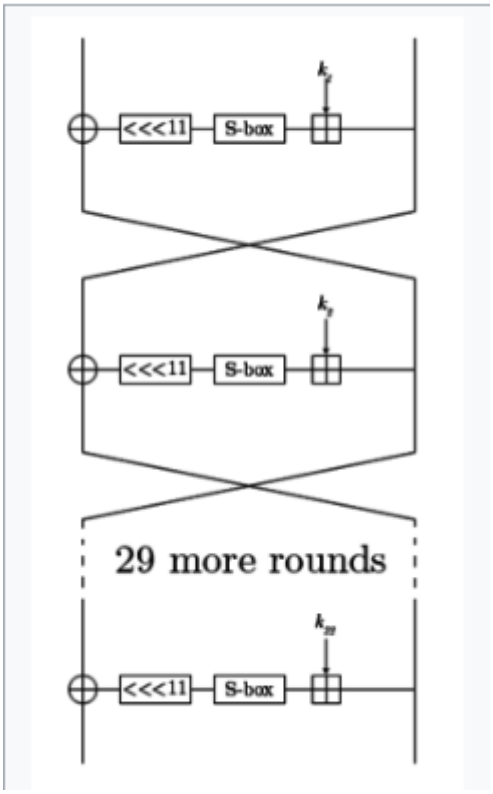
ΣΑΡΑΜΠΑΛΗΣ ΓΙΩΡΓΟΣ
Α.Μ. 1063479

Περιεχόμενα

Gost 28147-89 2
S_BOX 3
Αρχιτεκτονική 3
rAM 3
Gost_CiPher_fun 4
fsm 5
FINAL 7
VIVADO 9

Gost 28147-89

O Gost 28147-89 , είναι ένας αλγόριθμος κρυπτογράφησης μπλοκ δεδομένων των 64 bit με βάση ένα κλειδί των 256 bit. Συγκεκριμένα ο αλγόριθμος χρησιμοποιεί δίκτυο Feistel με αποτέλεσμα να είναι συμμετρικός, δίνοντας έτσι την δυνατότητα κρυπτογράφησης καθώς και αποκρυπτογράφησης των δεδομένων με το ίδιο ακριβώς κλειδί .Ο αλγόριθμος εκτελείται 32 φορές και επιστρέφει την κρυπτογράφηση της εισόδου.



Το κλειδί χωρίζεται σε 8 διανύσματα των 32 bit τα οποία αποθηκεύονται σε μία μνήμη. Σε κάθε εκτέλεση του αλγορίθμου χρησιμοποιείται και διαφορετικό διάνυσμα κλειδιού .Η ακολουθία των διανυσμάτων αυτών για την διαδικασία κρυπτογράφησης είναι η ακόλουθη (ΔΚ0..ΔΚ7) (ΔΚ0..ΔΚ7) (ΔΚ0..ΔΚ7)(ΔΚ7..ΔΚ0). Στις τελευταίες 8 επαναλήψεις τα διανύσματα κλειδιού λαμβάνονται με την αντίθετη φορά.

Για την διαδικασία αποκρυπτογράφησης χρησιμοποιούνται ακριβώς τα ίδια διανύσματα αλλά με αντίστροφη φορά . Έτσι η ακολουθία θα είναι (ΔΚ0..ΔΚ7)(ΔΚ7..ΔΚ0) (ΔΚ7..ΔΚ0) (ΔΚ7..ΔΚ0).

Το δίκτυο το οποίο επαναλαμβάνεται 32 φορές φαίνεται στην εικόνα και οι πράξεις οι οποίες εκτελούνται είναι οι παρακάτω.

Αρχικά , ο αλγόριθμος διαχωρίζει την 64 bit-η είσοδο σε δύο διανύσματα των 32 bit το κάθε ένα. Το διάνυσμα το οποίο διαθέτει τα 32 λιγότερο σημαντικά bit αθροίζεται ακέραια με το διάνυσμα κλειδιού το οποίο αντιστοιχεί στην συγκεκριμένη επανάληψη. Στην συνέχεια, η έξοδος του αθροιστή εισέρχεται σε έναν πίνακα ο οποίος λέγεται S_Box και παράγει μία διαφορετική τιμή. Τέλος , η έξοδος από το S_BOX περιστρέφεται λογικά κατά 11 θέσεις και εκτελείται η πράξη XOR ανά bit με το διάνυσμα το οποίο περιέχει τα 32 πιο σημαντικά bit. Το αποτέλεσμα το οποίο υπολογίστηκε μαζί με τα 32 λιγότερο σημαντικά bit της εισόδου αντιμετατίθενται και προκύπτει η 64 bit-η είσοδος της επόμενης επανάληψης.

Το κλειδί καθώς και ο πίνακας S_BOX είναι τα κρυφά στοιχεία του αλγορίθμου και έτσι θα πρέπει να είναι γνωστά τα στοιχεία αυτά μεταξύ μονάδων για να συνεργαστούν.

Μία επίθεση κλειδιού μπορεί να υποκλέψει τα περιεχόμενα του S_BOX σε 2^{32} κρυπτογραφήσεις και επειδή ακριβώς ο GOST είναι ένας γνωστός αλγόριθμος δεν χρησιμοποιείται σε εφαρμογές που η ασφάλεια των δεδομένων κρίνεται σημαντική.

S_BOX

Το S_BOX το οποίο είναι ένα από τα κρυφά στοιχεία του δικτύου Feistel που αξιοποιεί ο GOST είναι ένας πίνακας στον οποίο εισέρχεται μία 32 bit-η λέξη. Η λέξη αυτή διαχωρίζεται σε διανύσματα των 4 bit .Με βάση τον αριθμό του διανύσματος (τα 4 λιγότερο σημαντικά bit της λέξης εισόδου έχουν τον αριθμό μηδέν) επιλέγεται η γραμμή του πίνακα από τον οποίο θα γίνει η αντικατάσταση ενώ, με βάση τον ακέραιο αριθμό τον οποίο αντιπροσωπεύουν τα 4 αυτά bit επιλέγεται η στήλη. Έτσι για παράδειγμα η είσοδος (F1F1F1F1)HEX θα έχει σαν αποτέλεσμα η μονάδα S_BOX να βγάλει σαν αποτέλεσμα (3BBD2ACF)HEX με βάση το S_BOX της εικόνας. Συγκεκριμένα το S_BOX της εικόνας είναι το S_BOX το οποίο χρησιμοποιούσε η CBR(Central Bank of Russian Federation) ,και το οποίο χρησιμοποιήθηκε στην υλοποίηση.

#	S-box
1	4A92D80E6B1C7F53
2	EB4C6DFA23810759
3	581DA342EFC7609B
4	7DA1089FE46CB253
5	6C715FD84A9E03B2
6	4BA0721D36859CFE
7	DB413F590AE7682C
8	1FD057A4923E6B8C

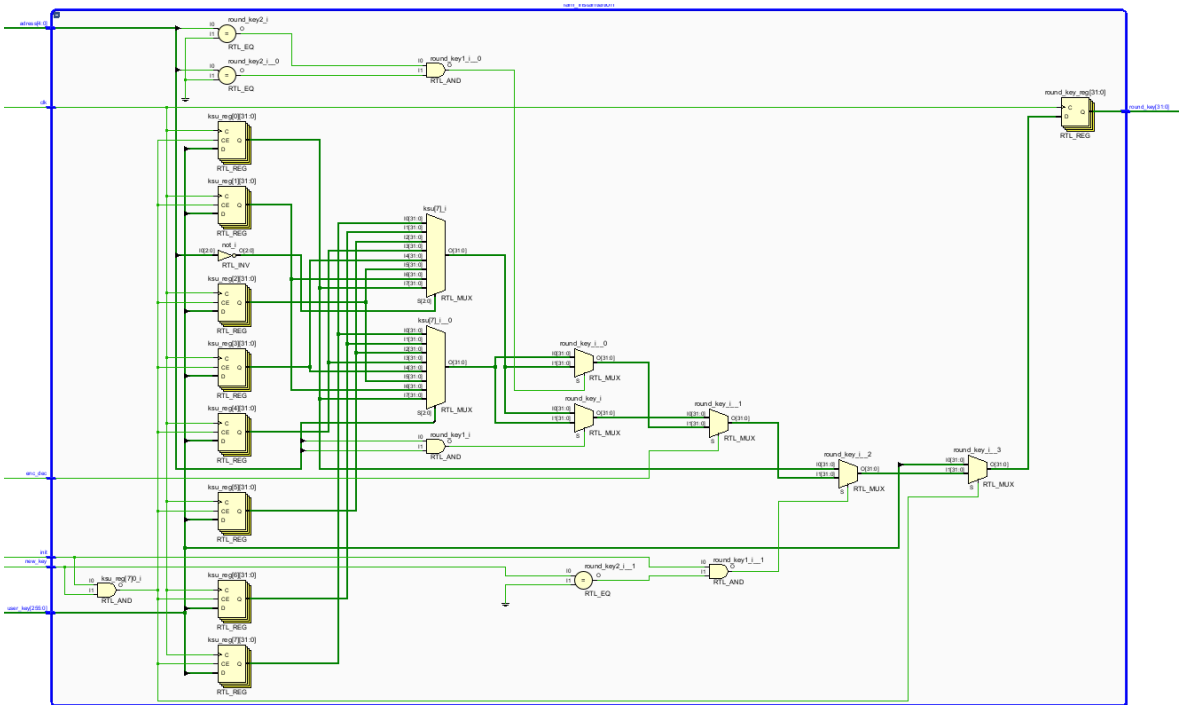
Αρχιτεκτονική

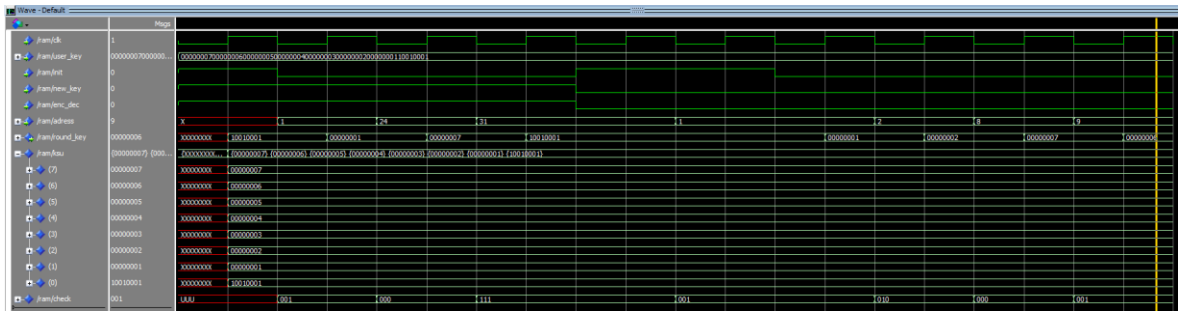
RAM

Η υπομονάδα RAM είναι η φυσική μνήμη του κυκλώματος στην οποία αποθηκεύεται ο πίνακας ksu . Ο πίνακας αυτός προκύπτει από την διαίρεση του κλειδιού μήκους 256 bit σε 8 διανύσματα των 32 bit. Η μονάδα , δέχεται σαν εισόδους τα σήματα user_key , clk, init, new_key, enc_dec, address και επιστρέφει στην έξοδο ένα σήμα μήκους 32bit που ονομάζεται round_key.

Περιγραφή του κυκλώματος. Το κύκλωμα αυτό , σε κάθε ανερχόμενη παρυφή ρολογιού εξετάζει τα σήματα init και new_key. Με βάση τα σήματα αυτά γίνεται η αρχικοποίηση τόσο του πίνακα ksu όσο και του σήματος εξόδου. Έτσι εάν τόσο το σήμα init όσο και το σήμα new_key είναι σε υψηλό δυναμικό η μνήμη δέχεται το νέο κλειδί (user_key) και με βάση αυτό ενημερώνει τον πίνακα ksu και επιστρέφει τα 32 λιγότερο σημαντικά bit του κλειδιού στην έξοδο. Εάν πάλι το init είναι σε υψηλό δυναμικό αλλά το σήμα new_key είναι ίσο με το μηδέν αυτό σημαίνει ότι εισερχόμαστε σε έναν νέο κύκλο υπολογισμού στον οποίο ωστόσο το κλειδί είναι το ίδιο , με αποτέλεσμα να μην χρειάζεται να ενημερωθεί ο πίνακας. Τέλος εάν το init είναι ίσο με το μηδέν τότε το συνολικό κύκλωμα βρίσκεται σε εκτέλεση και επομένως με βάση το σήμα address η RAM ενημερώνει την έξοδο. Είναι σημαντικό να αναφερθεί ότι το σήμα address έχει μήκος 5 bit ενώ για την προσπέλαση 8 θέσεων μνήμης χρειάζονται μόλις 3bit . Αυτό συμβαίνει για να μπορεί το κύκλωμα να γνωρίζει σε ποιο κύκλο βρίσκεται και έτσι να βγάζει στην έξοδο το σωστό τμήμα της ksu . Έτσι , εάν το σήμα enc_dec είναι ίσο με 1 (το ολικό κύκλωμα εκτελεί κρυπτογράφηση) τότε κατά την $24^{η}$ εκτέλεση και μετά θα πρέπει η RAM να επιστρέψει τα κλειδιά με την εξής ακολουθία 7,6,..,0. Αυτό υλοποιείται με την εξέταση των bit : address(4) και address(3) . Εάν αυτά είναι ίσα με την μονάδα

τότε επιστρέφουμε στην έξοδο το συμπλήρωμα του αριθμού address(2)address(1)address(0). Όμοια λογική ακολουθείται και στην περίπτωση όπου το σήμα enc_dec βρίσκεται σε χαμηλό δυναμικό. Παρακάτω φαίνεται η κυκλωματική αναπαράσταση της υπομονάδας RAM καθώς και γραφικές οι οποίες επιδεικνύουν την λειτουργία της.





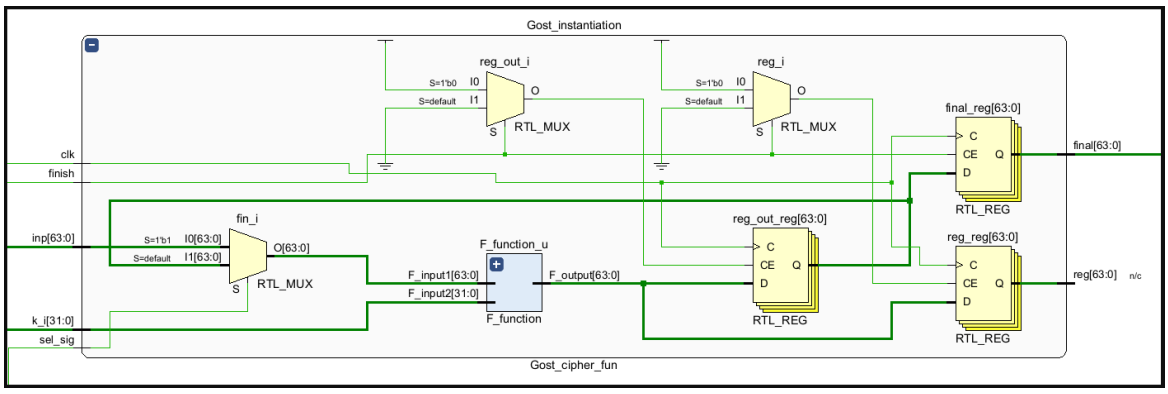
GOST_CIPHER_FUN

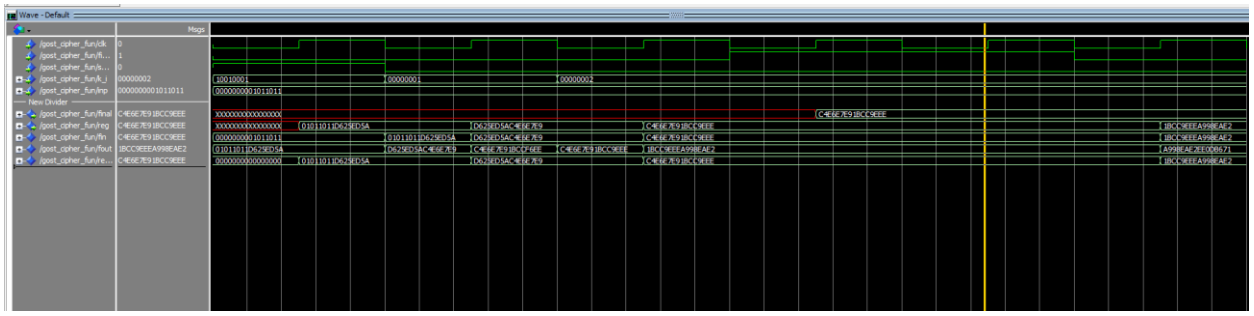
Η Gost_cipher_fun είναι η υπομονάδα στην οποία εκτελούνται οι επαναλήψεις κρυπτογράφησης . Η μονάδα αυτή αποτελείται από ένα στιγμιότυπο της συνάρτησης F_function έναν καταχωρητή καθώς και έναν πολυπλέκτη δύο σε ένα.

Περιγραφή F_function: η F_function είναι η συνδυαστική λογική ,είναι δηλαδή ο αθροιστής , το S_BOX καθώς και ο shifter – XOR συνδεδεμένα το ένα μετά το άλλο .Έτσι η F_function παίρνει σαν είσοδο κάθε φορά ένα 64bit σήμα καθώς και ένα 32bit-ο και επιστρέφει το αποτέλεσμα της παρούσας επανάληψης.

Περιγραφή Gost_cipher_fun : η Gost_cipher_fun μονάδα λοιπόν , δέχεται σαν εισόδους τα σήματα inp(64 bit), finish,sel_sig(control bits) ,clk, k_i(32 bit) και επιστρέφει στην έξοδο ένα σήμα final(64 bit). Με βάση το bit εισόδου sel_sig επιλέγεται το αντίστοιχο σήμα εισόδου στην F_function. Εάν το sel_sig είναι σε υψηλό δυναμικό τότε επιλέγεται σαν είσοδος το σήμα inp σηματοδοτώντας με αυτόν τον τρόπο την αρχή μίας νέας κρυπτογράφησης. Εάν το sel_sig είναι μηδέν τότε επιλέγεται σαν είσοδος στην F_function το σήμα reg_out το οποίο είναι η έξοδος του καταχωρητή και αντιπροσωπεύει το αποτέλεσμα της προηγούμενης επανάληψης. Όταν έρχεται ανερχόμενη παρυφή ρολογιού το κύκλωμα ελέγχει το περιεχόμενο του bit ελέγχου finish. Εάν αυτό βρεθεί σε υψηλό δυναμικό ενημερώνει την έξοδο final με το περιεχόμενο του καταχωρητή εκείνη τη στιγμή και στην συνέχεια δεν ενημερώνει τον καταχωρητή με νέα δεδομένα για όσο διάστημα παραμένει το finish στο 1. Αυτό συμβαίνει για να μπορεί ο αλγόριθμος να κάνει κρυπτογράφηση στο ήδη κρυπτογραφημένο σήμα. Εάν το σήμα γίνει μηδέν τότε και μόνο τότε ενημερώνεται εκ νέου ο καταχωρητής και ξεκινάει μία νέα διαδικασία κρυπτογράφησης.

Παρακάτω φαίνονται τόσο η κυκλωματική αναπαράσταση της υπομονάδας όσο και Κυματομορφές οι οποίες επιδεικνύουν την λειτουργία της.

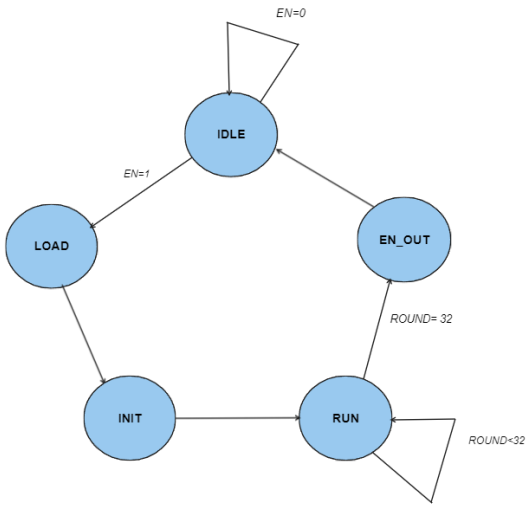




FSM

Η μονάδα FSM είναι υπεύθυνη για την παραγωγή και τον συγχρονισμό των υπόλοιπων μονάδων. Οι έξοδοι της μονάδας αυτής είναι συνάρτηση της τρέχουσας κατάστασης καθώς και των εισόδων επομένως αποτελεί μία Mealy FSM. Η είσοδοι αυτής της μονάδας είναι τα σήματα en, en_key, en_data τα οποία ορίζει ο χρήστης ανάλογα με το εάν θέλει να ξεκινήσει η διαδικασία κρυπτογράφησης και εάν θέλει να εισάγει νέες τιμές στο κλειδί ή στη λέξη κρυπτογράφησης .Η μηχανή FSM έχει 5 καταστάσεις οι οποίες

διακρίνονται : IDLE, LOAD, INIT, RUN, EN_OUT . Παρακάτω φαίνεται η σχηματική μετακίνηση μεταξύ των καταστάσεων.



Όπως φαίνεται και από το σχήμα από οι μεταβάσεις LOAD->INIT, INIT->RUN και EN_OUT->IDLE γίνονται χωρίς να ισχύει κάποια συνθήκη. Ωστόσο είναι σημαντικό να αναφερθεί πώς εάν υπάρχει ασύγχρονο reset η κατάσταση αρχικοποιείται στην IDLE.

Η μονάδα FSM έχει δύο process τα οποία γίνονται triggered με το clock . Το πρώτο είναι υπεύθυνο για την ενημέρωση της κατάστασης και το δεύτερο είναι υπεύθυνο για την ενημέρωση των εξόδων.

Στην συνέχεια περιγράφεται η λειτουργία της μηχανής και των αντίστοιχων σημάτων εξόδου από την κατάσταση idle με en=1.

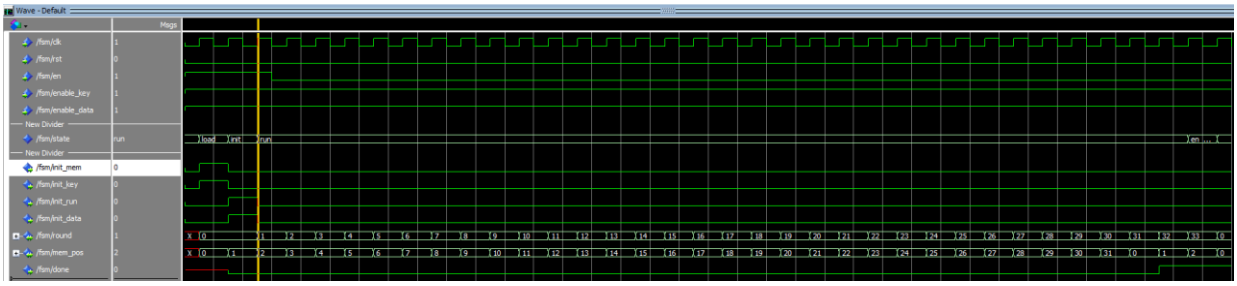
IDLE: Το πρώτο process σε ανερχόμενη παρυφή μετατρέπει την κατάσταση από idle σε load και με αυτόν τον τρόπο ενημερώνονται τα σήματα init_mem, init_key. Συγκεκριμένα το σήμα init_mem θα πάρει την μονάδα έτσι ώστε στο επόμενο clock η RAM να αρχικοποιηθεί, ενώ το σήμα init_key θα αρχικοποιηθεί ανάλογα με το σήμα εισόδου new_key. Το δεύτερο process θα μηδενίσει τα σήματα εισόδου-εξόδου mem_pos και round και αυτό γιατί δεν πρόλαβε την μετάβαση και έτσι εκτελείται για state = idle. Οι άλλες μονάδες ,δηλαδή, η RAM και η Gost_cipher, δεν θα εκτελεστούν διότι τα σήματα init_mem – init_run δεν έχουν ενημερωθεί στο πρώτο clock.

LOAD: Το πρώτο process θα μεταβάλει την κατάσταση σε INIT έτσι θα ενημερωθούν αντίστοιχα τα σήματα init_run , init_data, init_mem και init_key . Το δεύτερο process θα μηδενίσει το done και θα αυξήσει την τιμή του mem_pos κατά ένα δηλαδή mem_pos=1. Η RAM στο clock αυτό βλέπει τα σήματα init_mem και init_key από τον προηγούμενο κύκλο και αρχίζει την αρχικοποίηση φορτώνοντας στην έξοδό της το Round_key_0. Η Gost_cipher ακόμα δεν εκτελείται.

INIT: Το πρώτο process μεταβάλει την κατάσταση σε RUN και ενημερώνονται εκ νέου τα σήματα init_mem ,init_key, init_run και init_data στο μηδέν. Το δεύτερο process βλέποντας state INIT αυξάνει κατά ένα τα σήματα round και mem_pos. Η RAM επειδή βλέπει init_mem='0' και mem_pos='1' από το προηγούμενο clock φορτώνει στην έξοδο το Round_key_1 . Η Gost_cipher βλέπει το σήμα init_data από τον προηγούμενο κύκλο και αρχίζει την εκτέλεση με Round_key_0 επειδή αυτό είναι έτοιμο την ώρα του clock στην είσοδό του.

Αυτή η διαδικασία επαναλαμβάνεται μέχρις ότου το round=31 όπου το δεύτερο process σηκώνει σήμα done. Ωστόσο το σήμα done η Gost_cipher το βλέπει μετά από έναν ακόμα κύκλο για τον οποίο τρέχει τη τελευταία επανάληψη και όπου η FSM μεταβαίνει σε κατάσταση EN_OUT. Το σήμα done διατηρεί την τιμή 1 μέχρι να ξανά έρθει η fsm σε

κατάσταση init έτσι ώστε να μην αλλάξει η τιμή του καταχωρητή εξόδου της Gost_cipher.



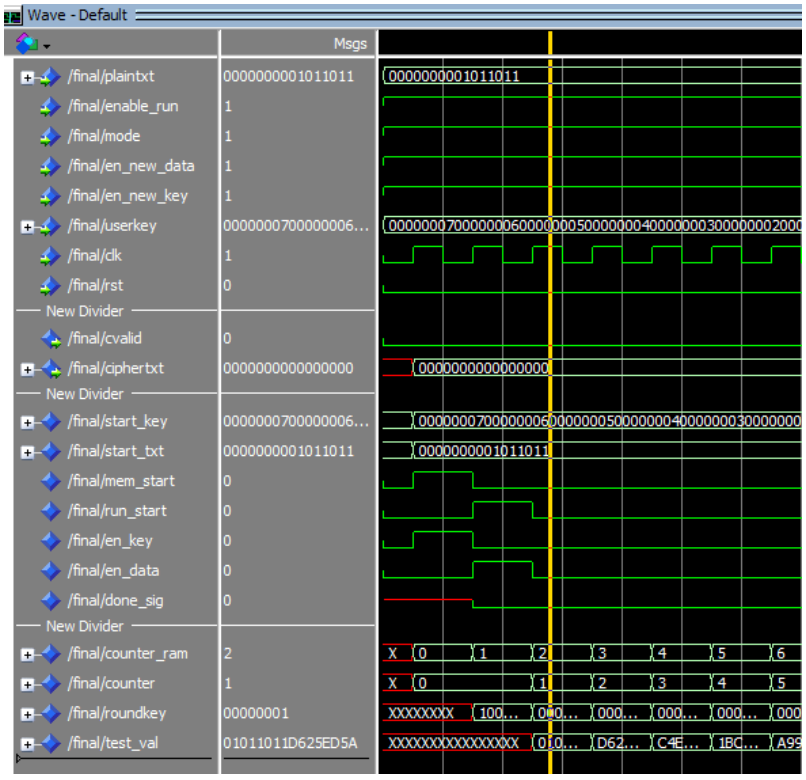
FINAL

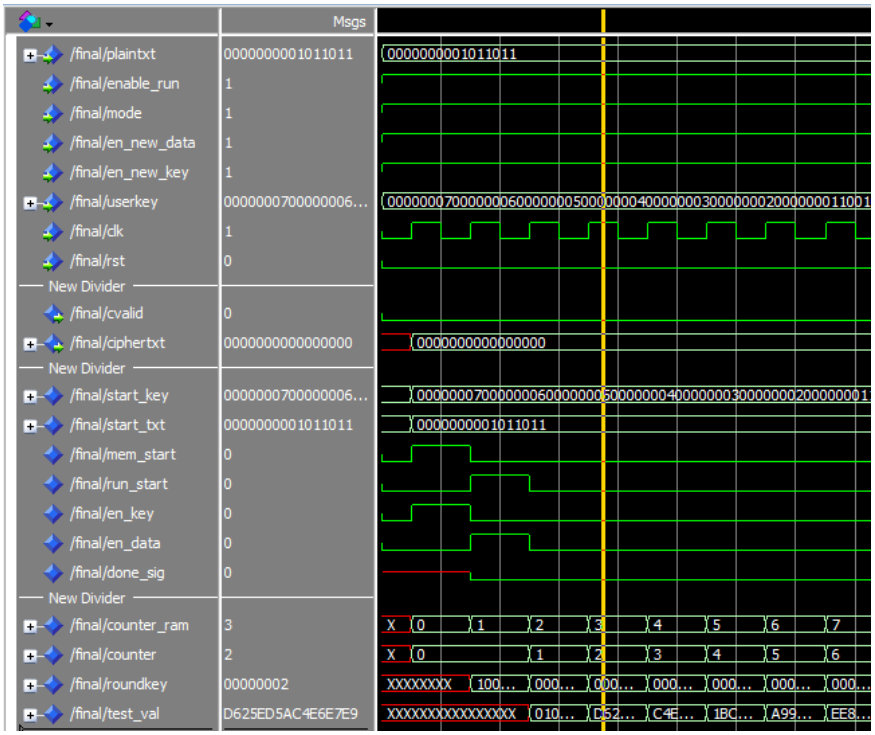
Για την ολοκληρωμένη υλοποίηση δημιουργούμε σε ένα νέο entity στο οποίο δημιουργούμε στιγμιότυπα των μονάδων RAM,Gost_cipher_fun,FSM και συνδέουμε τις μονάδες μεταξύ τους με σήματα. Επίσης , δημιουργούμε δύο καταχωρητές εισόδου στους οποίους εγγράφονται τα σήματα plaintext και userkey πριν περάσουν στην λογική.

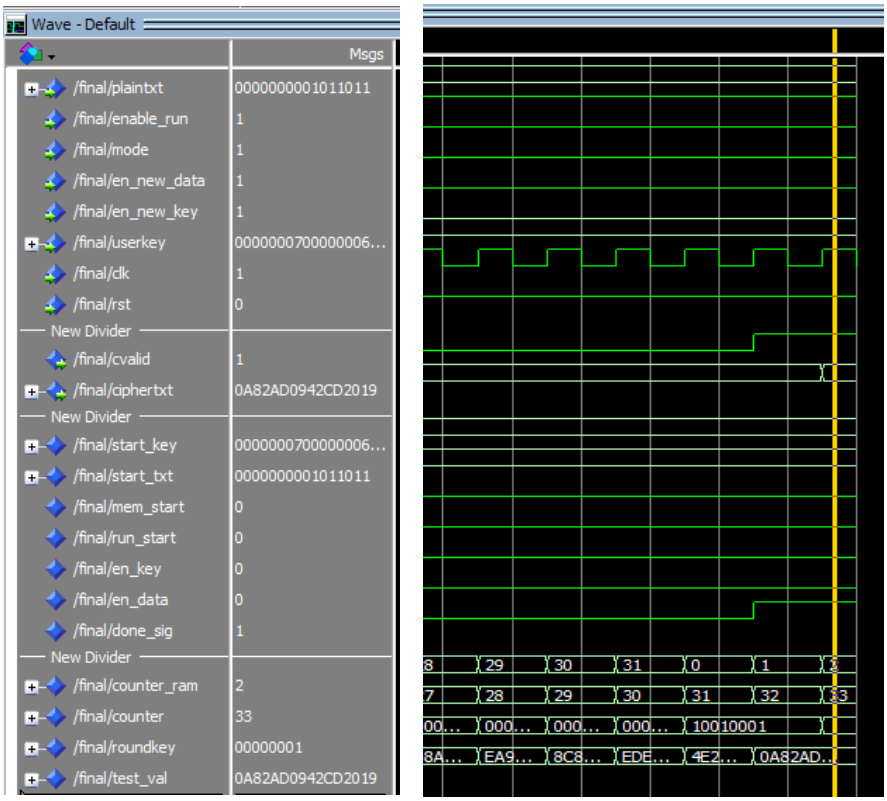
Παρακάτω φαίνονται Κυματομορφές που αποδεικνύουν την ορθή λειτουργία.

Για εισόδους plaintext : (0000000001011011)hex και

User_key : (00000007000000060000000500000004000000030000000200000001100101001)hex



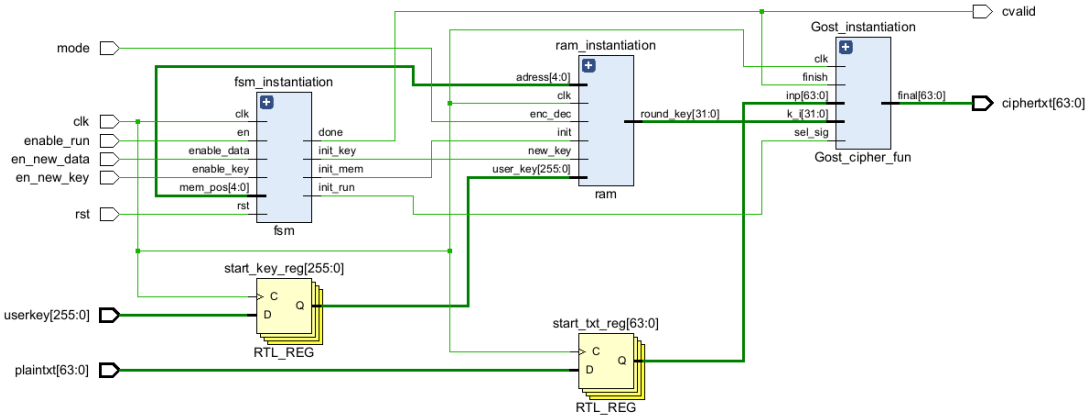




Να σημειωθεί ότι οι αντίστοιχες Κυματομορφές έχουν γίνει για την διαδικασία της αποκρυπτογράφησης.

VIVADO

Χρησιμοποιήσαμε το Vivado για την synthesis και το implementation το κυκλώματός μας. Το υλικό που επιλέχθηκε είναι το xc7z100ffg1156-1 της οικογενείας Zynq-7000 λόγω των επαρκών I/O που χρειάζονταν για το κύκλωμα.



Εικόνα 1 Αρχιτεκτονική GOST στο Vivado

Χρησιμοποιήσαμε τις default στρατηγικές για το synthesis και το implementation.

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	405	277400	0.15
FF	753	554800	0.14
IO	390	400	97.50
BUFG	1	32	3.13

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0,577 ns	Worst Hold Slack (WHS):	0,067 ns	Worst Pulse Width Slack (WPWS):	4,600 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	1218	Total Number of Endpoints:	1218	Total Number of Endpoints:	754

All user specified timing constraints are met.

Στο timing constraints θέσαμε την περίοδο του ρολογιού στα 10 ns οπότε και η συχνότητα λειτουργίας είναι στα 100 MHz. Με μικρότερα ρολόγια παρατηρούνταν setup violations.

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.278 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25,4°C**
Thermal Margin:	59,6°C (41,2 W)
Effective ϑJA:	1,4°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power