ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Ενσωματωμένα Συστήματα – Υλοποίηση του SDES σε Hardware

June 1

2012

Κεχαγιάς Απόστολος

ΑΕΜ:134

Επιβλέπων Καθηγητής: Μηνάς Δασυγένης
http://arch.icte.uowm.gr

Table of Contents

Ο αλγόριθμος

Ο απλοποιημένος συμμετρικός αλγόριθμος S-DES παίρνει σαν είσοδο ένα 8-bit απλό κείμενο και ένα κλειδί 10-bit και παράγει ένα 8-bit κρυπτογράφημα σαν έξοδο. Πχ

Είσοδος: 00010101

Κλειδί: 0101101000

Έξοδος : 11001111

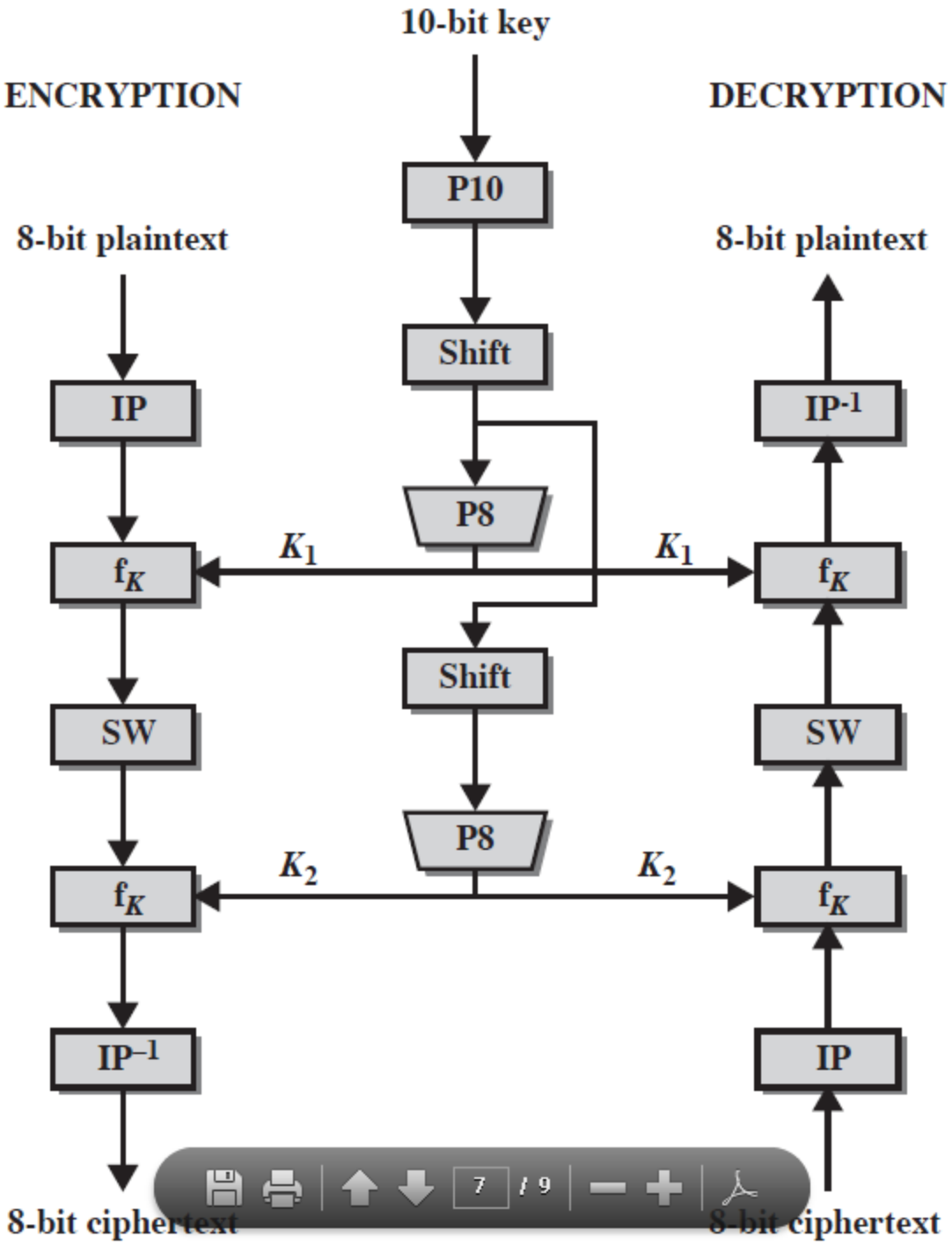
Ο S-DES εσωκλείει 5 συναρτήσεις για τη διαδικασία κρυπτογράφησης:

- Συνάρτηση 1: αρχική αντιμετάθεση (initial permutation, IP).
- Συνάρτηση 2: σύνθετη συνάρτηση f_k (περιλαμβάνει αντιμετάθεση και αλλαγή και εξαρτάται από το κλειδί εισόδου).
- Συνάρτηση 3: απλή συνάρτηση αντιμετάθεση των δύο μισών εισόδου (switch, SW).
- Συνάρτηση 4: τη σύνθετη συνάρτηση f_k και πάλι.
- Συνάρτηση 5: τελική αντιμετάθεση , που είναι η αντίστροφη της αρχικής αντιμετάθεσης (IP^{-1}).

Επίσης εσωκλείει 5 βήματα για την παραγωγή των δύο υποκλειδιών:

- Βήμα1 : αντιμετάθεση P_{10}
- Βήμα2: αριστερή ολίσθηση LS-1
- Βήμα3: αντιμετάθεση P_8
- Βήμα4: Διπλή αριστερή ολίσθηση LS-2
- Βήμα5: αντιμετάθεση P_8 (ξανά).

Σχηματικά αυτή η διαδικασία φαίνεται στην παρακάτω εικόνα.



Οι διαδικασίες P_8 και P_{10} είναι απλές αντιμεταθέσεις.

Είναι φανερό ότι από το γράφημα χρησιμοποιούνται δύο κλειδιά, το κλειδί K_1 και το κλειδί K_2, τα οποία προκύπτουν από το αρχικό κλειδί K.

Το αρχικό κλειδί έχει εύρος 10-bit, ενώ τα υποκλειδιά K_1 και K_2 έχουν εύρος 8-bit το καθένα.

Η κρυπτογράφηση τροφοδοτείται πρώτα με το υποκλειδί K_1 και στη συνέχεια με το υποκλειδί K_2.

Η κρυπτογράφηση μπορεί να εκφραστεί ως εξής:

$C = IP^{-1}(f_{k2}(SW(f_{k1}(IP(m)))))$ όπου C είναι το κρυπτογράφημα και m το απλό κείμενο ενώ η ίδια διαδικασία αντιστρέφοντας τα κλειδιά που χρησιμοποιούνται δίνει το αρχικό μήνυμα.

Η παραγωγή των υποκλειδιών αναλύεται :

$K_1 = P8(Shift(P10(K)))$

$K_2 = P8(Shift(Shift(P10(K))))$

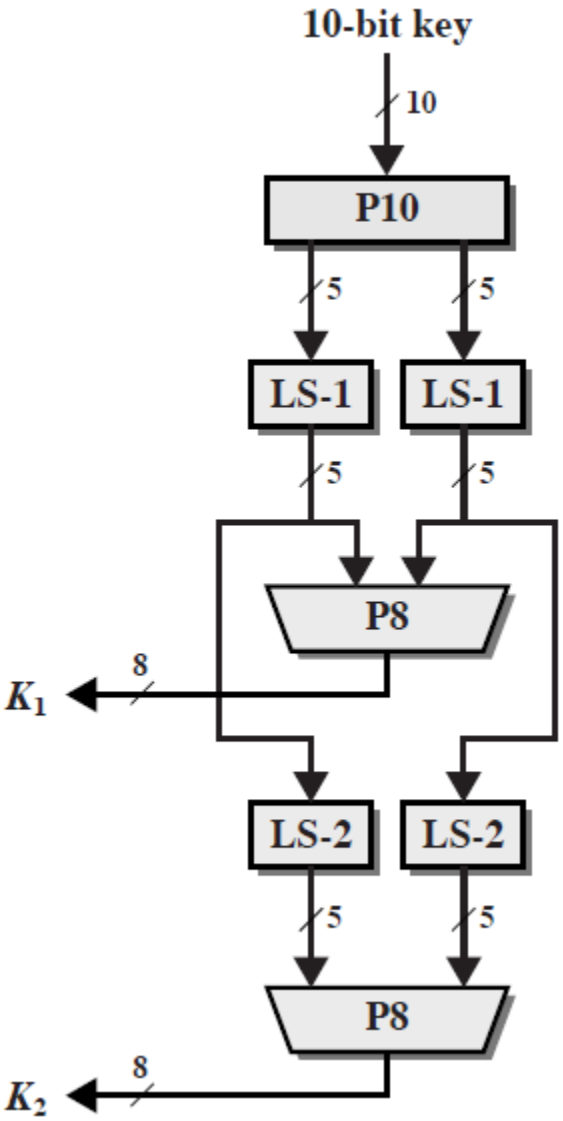


Figure 1 - Παραγωγή υποκλειδιών

Η κρυπτογράφηση του S-DES αποτελείται από δύο συνεχόμενα βήματα όπου γίνεται διπλή εφαρμογή της συνάρτησης f_k, πρώτα με είσοδο το υποκλειδί K_1 και μετά με είσοδο το υποκλειδί K_2.

Η συνάρτηση f_k μπορεί να περιγραφεί:

$f_k(L,R) = (L \text{ XOR } F(R,SK),R)$

Όπου:

L: το αριστερό 4-bit μέρος του απλού μηνύματος.

R: το δεξί 4-bit μέρος του απλού μηνύματος.

F: η ενδιάμεση διαδικασία αντιμετάθεσης και μετατόπισης (εισάγονται 4-bit και εξάγονται 4-bit).

SK: το υποκλειδί

Αρχικά το απλό μήνυμα των 8-bit εισάγεται για κρυπτογράφηση. Το αρχικό απλό μήνυμα αντιμετατίθεται σύμφωνα με τη συνάρτηση IP. Έπειτα το μήνυμα χωρίζεται στο αριστερό (L) και στο δεξί (R) μέρος του. Στη συνέχεια το δεξιό μέρος R εισάγεται στη συνάρτηση E/P. Η συνάρτηση E/P λαμβάνει 4-bit είσοδο και παράγει 8-bit έξοδος. Τέλος η 8-bit έξοδος συνδυάζεται με το 8-bit υποκλειδί K_1 με την πράξη XOR. Το αριστερό μέρος εισάγεται στο κουτί So και το δεξιό στο κουτί S1.

Τα κουτιά So και S1 δέχονται 4-bit εισόδους και παράγουν 2-bit εξόδους. Τα κουτιά αποτελούν δισδιάστατους πίνακες 4χ4 που περιέχουν δεκαδικούς αριθμούς από το ο έως και 3.

Η 4-bit είσοδος «σπάει» στη μέση και το πρώτο μέρος δηλώνει το δεκαδικό αριθμό σειράς και το δεύτερο μέρος δηλώνει το δεκαδικό αριθμό στήλης. Έτσι επιλέγεται ως έξοδος η αντίστοιχη εγγραφή του πίνακα.

$$S0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \qquad S1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

Έπειτα τα δύο τμήματα εισάγονται ενοποιημένα στην συνάρτηση αντιμετάθεσης P_4. Η συνάρτηση P_4 δέχεται δύο τμήματα των 2-bit και εξάγει ένα 4-bit τμήμα. Αύτο το 4-bit τμήμα συνδυάζεται με πράξη XOR με το αρχικό τμήμα L το οποίο με την σειρά του εισάγεται στην συνάρτηση SW σαν αριστερό μέλος. Σαν δεξιό μέλος εισάγεται το τμήμα R που είχε δημιουργηθεί σε προηγούμενο βήμα. Το αποτέλεσμα της SW μπαίνει σε ακόμη μία συνάρτηση f_k χρησιμοποιώντας ως υποκλειδί το K_2.

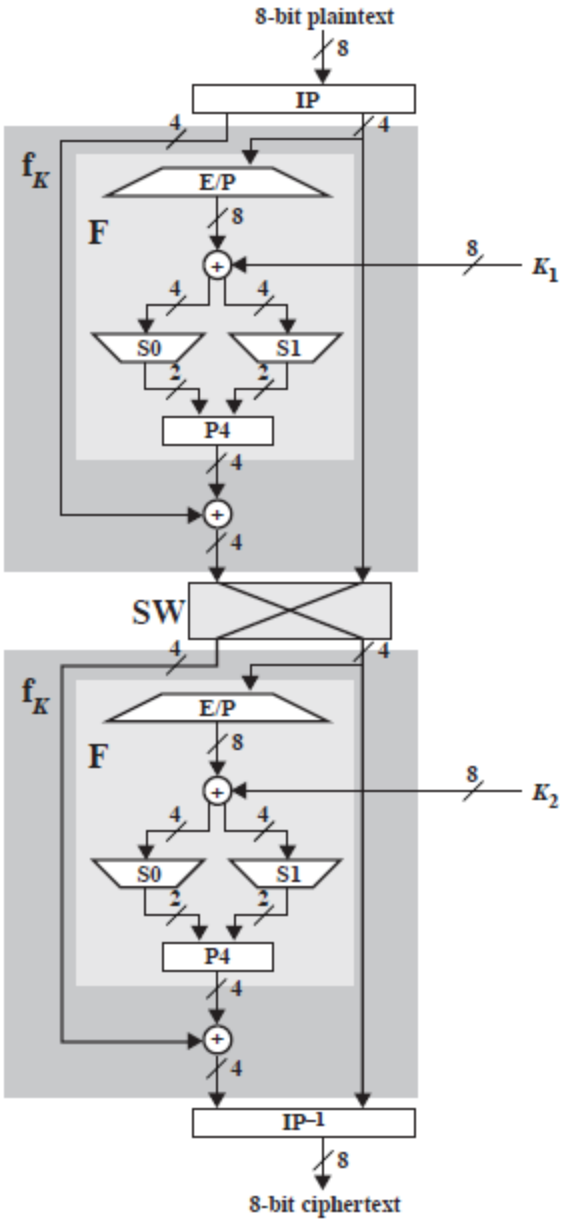


Figure 2 - Η κρυπτογράφηση σχηματικά

Υλοποίηση σε Hardware

Η υλοποίηση όπως φαίνεται και στην παρακάτω εικόνα ακολούθησε ακριβώς την τμηματική περιγραφή του αλγορίθμου. Αυτό σημαίνει πως για κάθε συνάρτηση που αναλύθηκε προηγουμένως δημιουργήθηκε μια αρχιτεκτονική. Έπειτα τα τμήματα ενώθηκαν χρησιμοποιώντας την τεχνική του pipelining χρησιμοποιώντας πεπερασμένες μηχανές καταστάσεων για την ακολουθιακή εκτέλεση του αλγορίθμου. Η σειρά εκτέλεσης μπορεί εύκολα να αναγνωριστεί από την σχηματική περιγραφή.

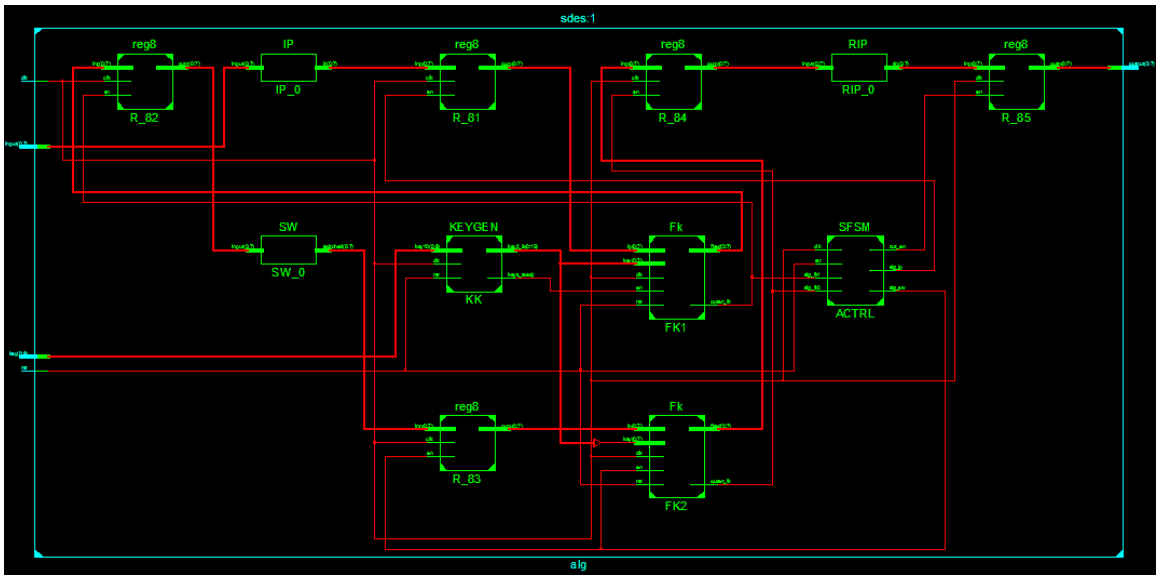


Figure 3 - sdes RTL

Τα components που υλοποιήθηκαν με τις εισόδους/εξόδους τους:

```
entity P10 is
    port (key: in std_logic_vector(0 to 9);
    p10: out std_logic_vector(0 to 9));
end;

entity P8 is
    port (input: in std_logic_vector(0 to 9);
    permute8 : out std_logic_vector(0 to 7));
end;

entity IP is
```

```
    port (input: in std_logic_vector(0 to 7);
    ip : out std_logic_vector(0 to 7));
end;

entity RIP is
    port (input: in std_logic_vector(0 to 7);
    rip : out std_logic_vector(0 to 7));
end;

entity FKFSM is
    port (clk, rst, en: in std_logic;
      en_f: in std_logic;
      out_en: out std_logic);
end;

entity Fk is
    port (clk, rst, en: in std_logic;
      ip: in std_logic_vector(0 to 7);
    key : in std_logic_vector(0 to 7);
    fked : out std_logic_vector(0 to 7);
      outen_fk: out std_logic);
end;

entity SW is
    port (input: in std_logic_vector(0 to 7);
    switched : out std_logic_vector(0 to 7));
end;

entity S0 is
    port(input: in std_logic_vector(0 to 3);
    output: out std_logic_vector(0 to 1));
end;

entity S1 is
    port(input: in std_logic_vector(0 to 3);
    output: out std_logic_vector(0 to 1));
end;

entity EP is
    port(input: in std_logic_vector(0 to 3);
    eped: out std_logic_vector(0 to 7));
end;

entity P4 is
    port(input: in std_logic_vector(0 to 3);
    p4ed: out std_logic_vector(0 to 3));
end;

entity FFSM is
    port (clk, rst, en_f: in std_logic;
      sig_ep, sig_xor, sig_s0s1: out std_logic;
      out_en: out std_logic);
end;

entity F is
    port (clk, rst, en_f:in std_logic;
      input: in std_logic_vector(0 to 3);
```

```
    key: in std_logic_vector(0 to 7);
    fed : out std_logic_vector(0 to 3);
      outen_f: out std_logic);
end;

entity XOR8 is
      port(
            inp1: in std_logic_vector(0 to 7);
            inp2: in std_logic_vector(0 to 7);
            outp: out std_logic_vector(0 to 7));
end;

entity XOR41 is
      port(
            inp1: in std_logic_vector(0 to 3);
            inp2: in std_logic_vector(0 to 3);
            outp: out std_logic_vector(0 to 3));
end;

entity LS1 is
      port (input: in std_logic_vector(0 to 9);
      ls1 : out std_logic_vector(0 to 9));
end;

entity LS2 is
      port (input: in std_logic_vector(0 to 9);
      ls2 : out std_logic_vector(0 to 9));
end;

entity KFSM is
    port (clk, rst: in std_logic;
      sig_p10, sig_p8, sig_ls1, sig_ls2: out std_logic;
      out_en: out std_logic);
end;

entity KEYGEN is
    port (clk, rst: in std_logic;
      key10: in std_logic_vector(0 to 9);
    key2_8: out std_logic_vector(0 to 15);
      keys_ready: out std_logic);
end;

entity SFSM is
    port (clk, rst: in std_logic;
      sig_fk1, sig_fk2: in std_logic;
      sig_ip, sig_sw: out std_logic;
      out_en: out std_logic);
end;

entity sdes is
    port(clk, rst: std_logic;
      input :in std_logic_vector(0 to 7);
    key: in std_logic_vector(0 to 9);
    output: out std_logic_vector(0 to 7));
end;

entity r_sdes is
```

```
    port(clk, rst: std_logic;
      input :in std_logic_vector(0 to 7);
    key: in std_logic_vector(0 to 9);
    output: out std_logic_vector(0 to 7));
end;
```

Η λειτουργία όλων των παραπάνω components αναφέρεται στο έγγραφο που ορίζεται ο αλγόριθμος. Για περισσότερες λεπτομέρειες υπάρχει το αντίστοιχο link στην βιβλιογραφία.

Τα μόνα στοιχεία που δεν αναφέρονται είναι οι μηχανές καταστάσεων. Η λειτουργία όλων αυτών είναι να ενεργοποιούν ακολουθιακά και ανάλογα με τους κύκλους ρολογιού κάποια σήματα. Αυτά τα σήματα στην συνέχεια ενεργοποιούν registers οι οποίοι αποθηκεύουν την τρέχουσα κατάσταση ενός σήματος ή μια ομάδας σημάτων. Έτσι εξασφαλίζεται ότι θα είναι έτοιμα κάποια δεδομένα που απαιτούνται για την εκτέλεση ενός βήματος που βρίσκεται πιο χαμηλά σε μια υποτιθέμενη στοίβα εκτέλεσης.

Για την οπτικοποίηση των αποτελεσμάτων χρησιμοποιήθηκε η έξοδος vga οπότε υλοποιήθηκε ένας controller που λειτουργεί σε ανάλυση 640x480@25Mhz. Ο controller δέχεται ως είσοδο το μήνυμα το κλειδί το κρυπτογράφημα και την τρέχουσα θέση προς αλλαγή και εμφανίζει τα αποτελέσματα στην οθόνη χρησιμοποιώντας 4bit ανά χρώμα.

Είσοδοι και έξοδοι του controller

```
entity controller is
      port(
            clk50_in: in std_logic;
            input: inout std_logic_vector(0 to 17);
            red_out   : out std_logic_vector(3 downto 0);
            green_out : out std_logic_vector(3 downto 0);
            blue_out  : out std_logic_vector(3 downto 0);
            hs_out    : out std_logic;
            vs_out    : out std_logic;
            rotary_a     : in std_logic;
            rotary_b     : in std_logic;
            tgl_btn      : in std_logic;
            reset  : in std_logic);
end controller;
```

Τα γραφικά που εμφανίζονται δημιουργήθηκαν σε ένα πρόγραμμα επεξεργασίας εικόνας. Στην συγκεκριμένη περίπτωση χρησιμοποιήθηκε το GIMP. Δημιουργήθηκε το background που φαίνεται στην εικόνα και σε ξεχωριστά layers οι αριθμοί 0 και 1 με μέγεθος το εσωτερικό του κάθε κελιού που είναι 46x61.

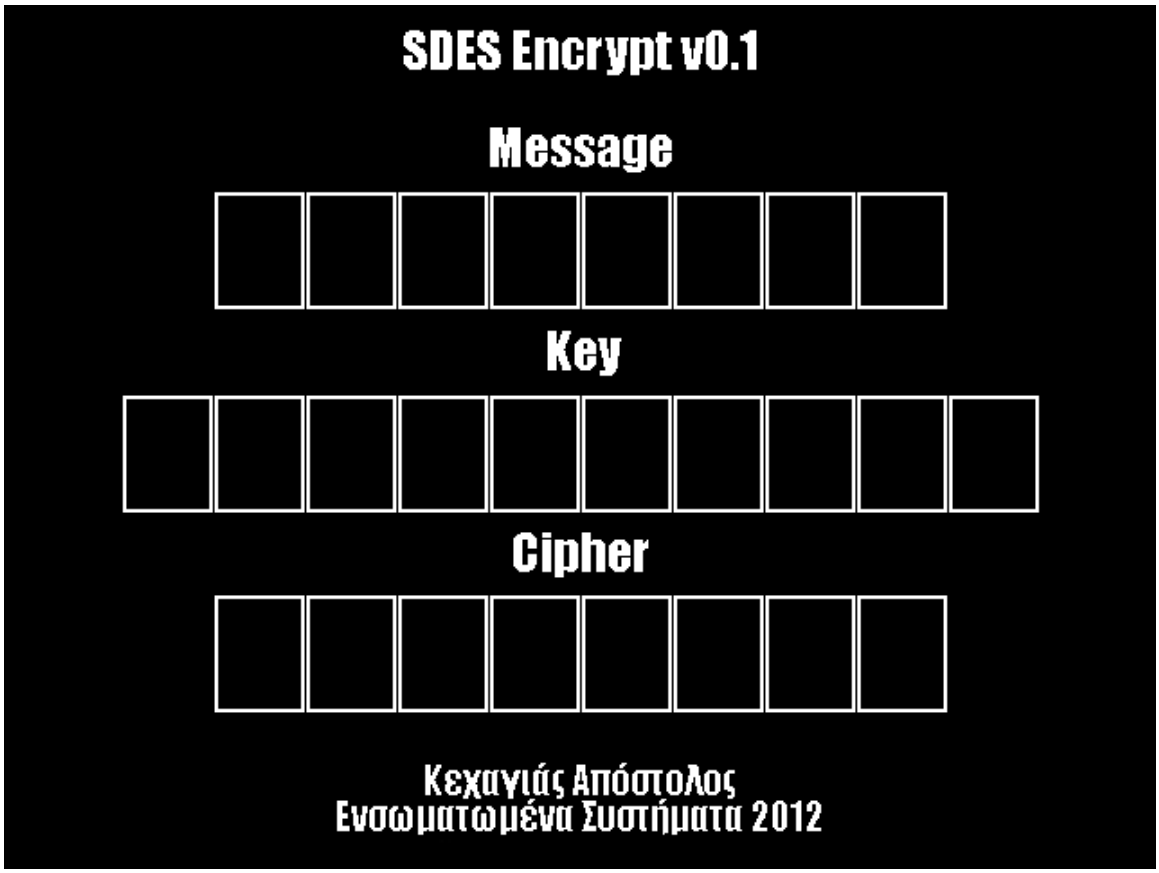


Figure 4 - GIMP ouput

Έπειτα σε όλες τις εικόνες ορίσαμε ένα κατώφλι ώστε να πάρουμε σαν αποτέλεσμα μια εικόνα με δύο χρωματικούς κώδικες. Εφόσον μιλάμε για rgb εικόνες αυτό είναι ο και 255. Στην συνέχεια χρησιμοποιήθηκε η matlab για την εξαγωγή του array. Συγκεκριμένα

Πχ για την εικόνα του αριθμού ο

```
z = imread('z.png');
z = rgb2gray(z) ;
z = z > 1;
dlmwrite('zero.txt',z);
```

Αυτό θα μας δώσει σαν έξοδο ένα αρχείο με ο και 1 διαχωρισμένα με κόμμα. Έτσι αν χρησιμοποιήσουμε το notepad++ πχ μπορούμε να αλλάξουμε τα ο σε '0', τα 1 σε '1' και τα \n σε),\n(ώστε να φτιάξουμε ένα vhdl array.

Αφού ορίστηκαν οι επιθυμητοί πίνακες τότε χρησιμοποιώντας τα σήματα hs και vs μπορούμε να ξέρουμε κάθε φορά σε ποιο pixel βρισκόμαστε και να το χρωματίσουμε ανάλογα. Ο vga controller ζωγραφίζει κάθε φορά την κύρια οθόνη και έπειτα ζωγραφίζει πάνω από αυτή τους χαρακτήρες ανάλογα με την είσοδο. Το ίδιο γίνεται και με τον κέρσορα.

Χρονισμός σημάτων VGA

Σύμφωνα με το datasheet για ανάλυση 640x480 προκύπτει ότι θα πρέπει να ορίσουμε τα σήματα hsync και vsync ως εξής.

Symbol	Parameter	Vertical Sync			Horizontal Sync	
		Time	Clocks	Lines	Time	Clocks
T_S	Sync pulse time	16.7 ms	416,800	521	32 µs	800
T_{DISP}	Display time	15.36 ms	384,000	480	25.6 µs	640
T_{PW}	Pulse width	64 µs	1,600	2	3.84 µs	96
T_{FP}	Front porch	320 µs	8,000	10	640 ns	16
T_{BP}	Back porch	928 µs	23,200	29	1.92 µs	48

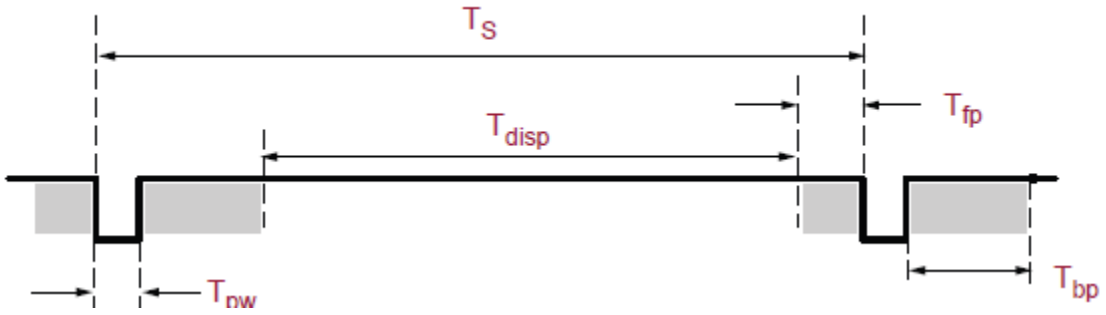


Figure 5 - VGA control Timing

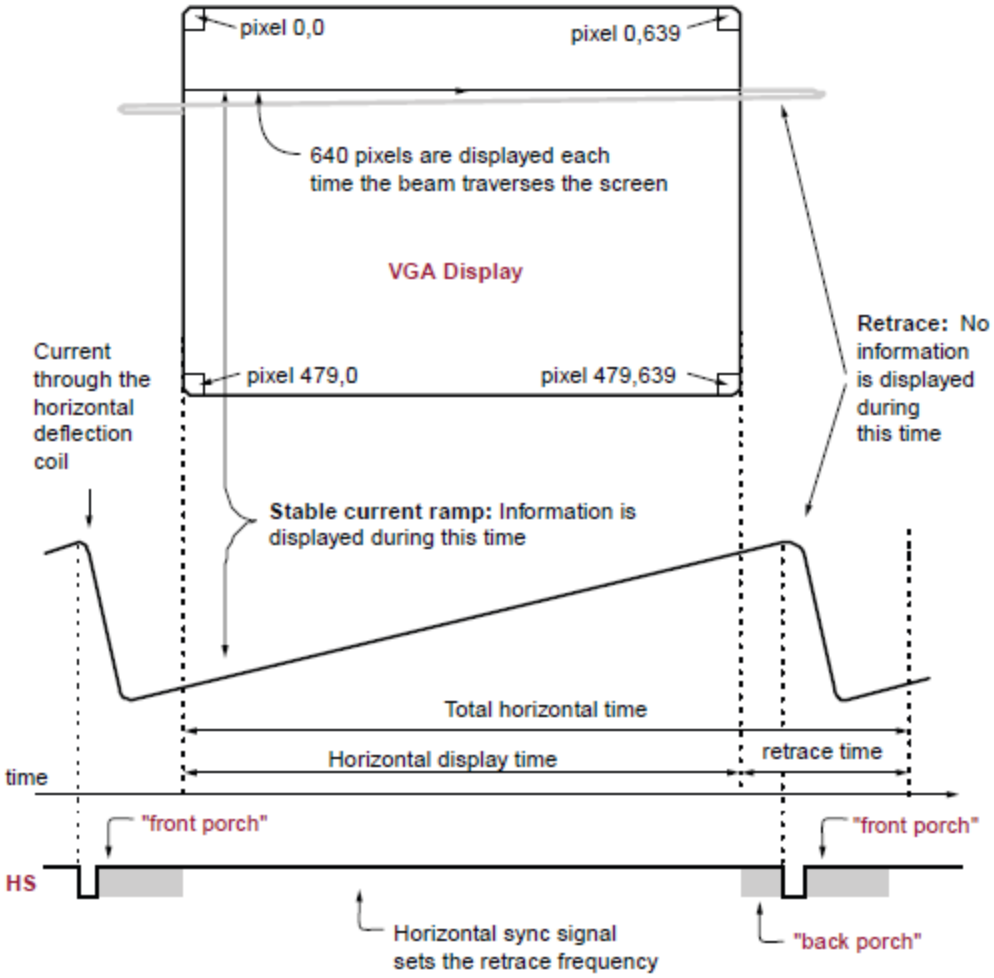


Figure 6 - Παράδειγμα

Δηλαδή ένας παλμός Tpw κάνει 96 κύκλους. Το Tbp(back porch) κάνει 48 κύκλους. Ο χρόνος εκτύπωσης είναι 640 κύκλοι ενώ το Tfp είναι 16 κύκλοι. Αυτό μας κάνει ένα σύνολο 800 κύκλων ώστε να τυπώσουμε σε μια οθόνη με 640 pixel μήκος.

Αντίστοιχα προκύπτει και ο χρονισμός του vs. Tpw 1600 κύκλοι(1600/800 = 2 γραμμές). Tbp 23200 κύκλοι(29 γραμμές). Χρόνος εκτύπωσης Tdisp 384000(480 γραμμές) και Tfp 8000(10 γραμμές) κύκλοι. Άρα ένα σύνολο 521 γραμμών χρειάζεται για τον χρονισμό του vs.

Έτσι ορίζοντας έναν counter για το κάθε ένα μπορούμε να βρούμε σε ποιο ακριβώς pixel βρισκόμαστε και να τυπώσουμε την κατάλληλη πληροφορία. Σύμφωνα με το σχήμα δηλαδή το μόνο που έχουμε να κάνουμε είναι:

α) για το hs να δίνουμε στην έξοδο πάντα 1 εκτός από τους κύκλους 0-97.

β) για το vs na δίνουμε στην έξοδο πάντα 1 εκτός από τους κύκλους 0-3.

Επαλήθευση

Τέλος για να δούμε ότι πράγματι δουλεύουν όλα σωστά κάναμε μια επαλήθευση με μια software υλοποίηση του αλγορίθμου.

```
C:\Users\tolis\Dropbox\Security\sDES>sdes -e -m 00001111 -k 1111100000

Message:        00001111:15
Key:            1111100000:992
Cipher:         10100101:165
```

Figure 7 - Επάληθευση με software

Figure 8 - Η οπτικοποίηση του αλγορίθμου

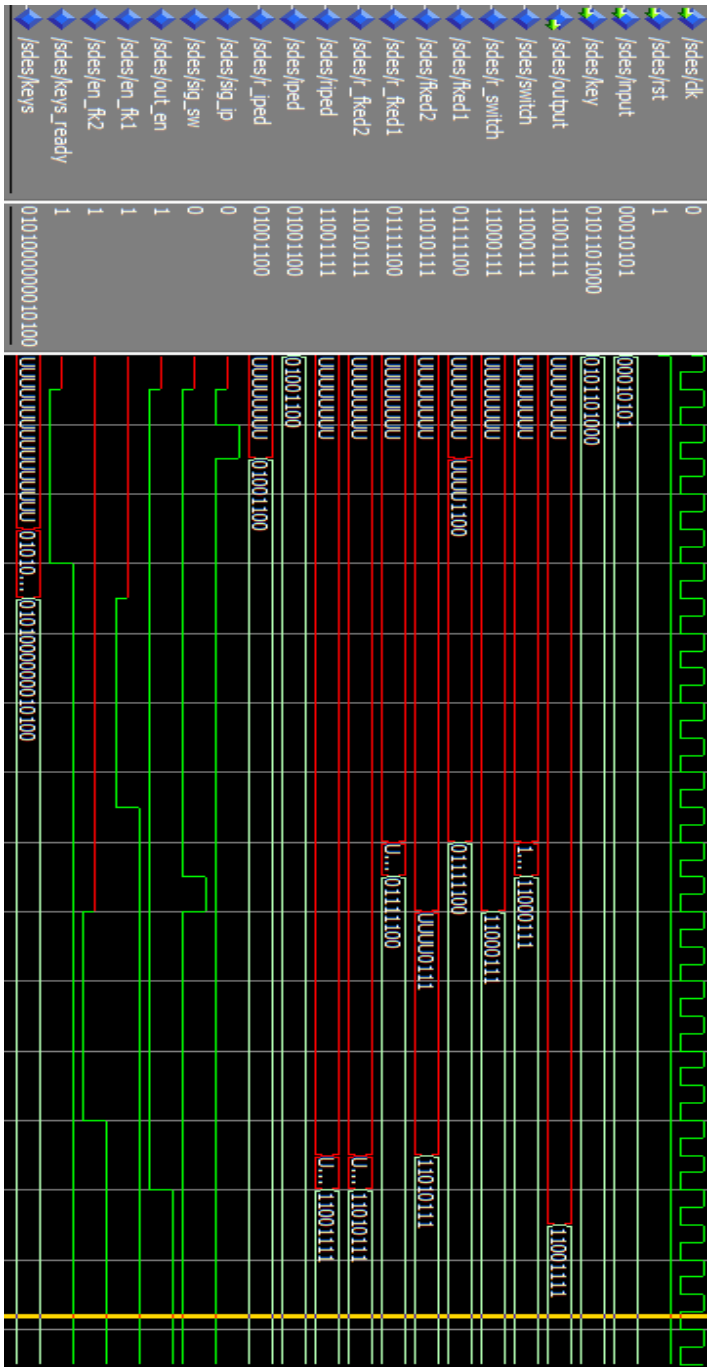


Figure 9 - προσομοίωση στο ModelSim

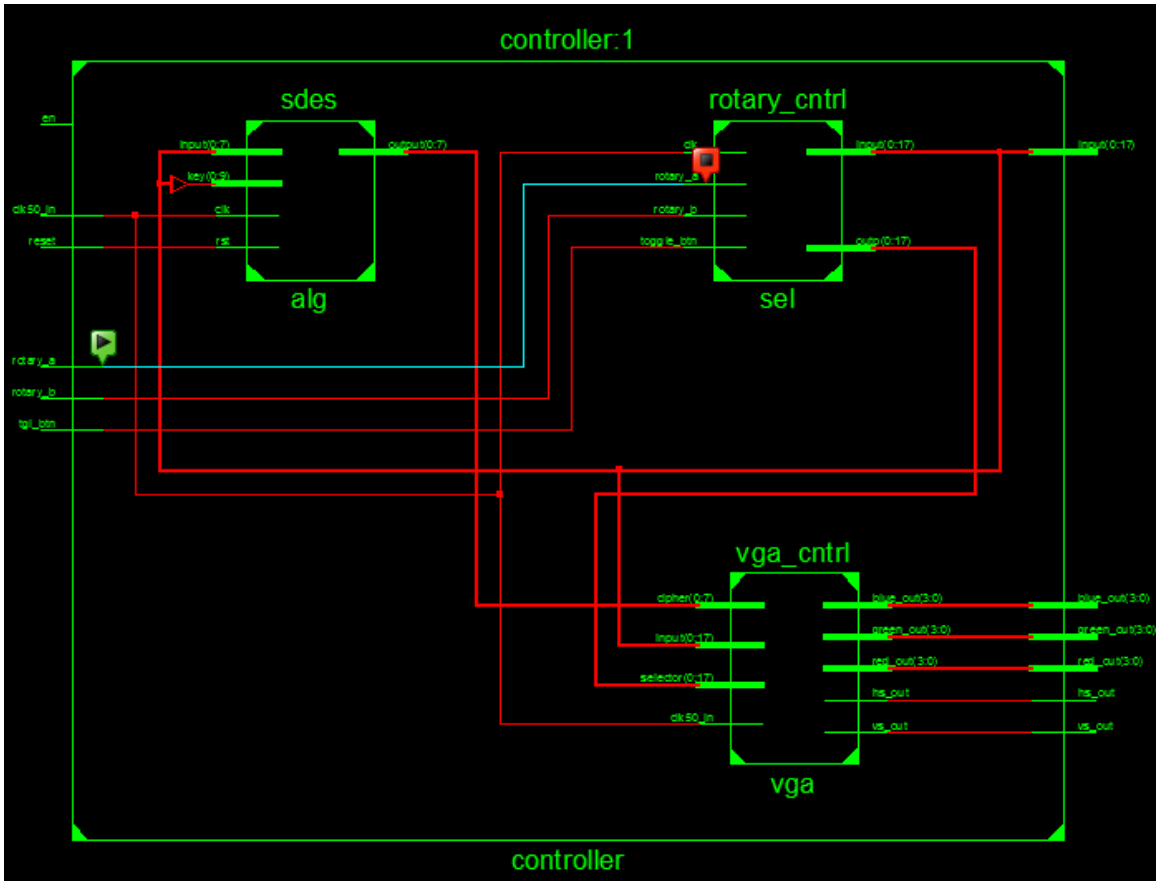


Figure 10 - controller RTL

Είσοδοι και έξοδοι vga_cntrl

```
entity vga_cntrl is
  port(clk50_in  : in std_logic;
            input    : in std_logic_vector(0 to 17);
            selector: in std_logic_vector(0 to 17);
            cipher     : in std_logic_vector(0 to 7);
            red_out    : out std_logic_vector(3 downto 0);
            green_out : out std_logic_vector(3 downto 0);
            blue_out   : out std_logic_vector(3 downto 0);
            hs_out     : out std_logic;
            vs_out     : out std_logic);
end vga_cntrl;
```

clk50_in : in std_logic;

50mhz clock

input : in std_logic_vector(0 to 17);

Δεδομένα προς εμφάνιση. Τα πρώτα 8 bit αφορούν το μήνυμα και τα υπόλοιπα το κρυπτογράφημα.

selector: in std_logic_vector(0 to 17);

Ένα από τα 17 bit είναι ενεργό κάθε φορά. Δείχνει σε ποια θέση θα εμφανιστεί ο κέρσορας.

cipher : in std_logic_vector(0 to 7);

Δεδομένα προς εμφάνιση. Το κρυπτογράφημα.

red_out : out std_logic_vector(3 downto 0);

Έξοδος για το κανάλι R(red).

green_out : out std_logic_vector(3 downto 0);

Έξοδος για το κανάλι G(green).

blue_out : out std_logic_vector(3 downto 0);

Έξοδος για το κανάλι B(blue).

hs_out : out std_logic;

Έξοδος horizontal sync.

vs_out : out std_logic

Έξοδος vertical sync.

Είσοδοι και έξοδοι rotary_cntrl

```
entity rotary_cntrl
port (
      outp : out std_logic_vector(0 to 17);
      toggle_btn : in std_logic;
      input : inout std_logic_vector(0 to 17);
      rotary_a : in std_logic;
      rotary_b : in std_logic;
      clk : in std_logic);
end entity;
```

outp : out std_logic_vector(0 to 17);

Ένα από τα 17 bit είναι ενεργό κάθε φορά. Δείχνει σε ποια θέση θα εμφανιστεί ο κέρσορας.

toggle_btn : in std_logic;

Αλλάζει κατάσταση στο επιλεγμένο bit.

input : inout std_logic_vector(0 to 17);

Τα πρώτα 8 bit αφορούν το μήνυμα και τα υπόλοιπα το κρυπτογράφημα.

rotary_a : in std_logic;

rotary_b : in std_logic;

Σήματα για ανίχνευση κίνησης του rotary knob.

Είσοδοι και έξοδοι sdes

```
entity sdes is
    port(clk, rst: std_logic;
    input :in std_logic_vector(0 to 7);
    key: in std_logic_vector(0 to 9);
    output: out std_logic_vector(0 to 7));
end;
```

input :in std_logic_vector(0 to 7);

Το 8 bit μήνυμα προς κρυπτογράφηση.

key: in std_logic_vector(0 to 9);

Το κλειδί που θα χρησιμοποιηθεί.

output: out std_logic_vector(0 to 7));

Το 8 bit κρυπτογράφημα.

Χειρισμός

Η λειτουργία του αλγορίθμου είναι πολύ απλή. Χρησιμοποιούμε το rotary knob ώστε να επιλέξουμε ποιο bit θέλουμε να τροποποιήσουμε. Αυτό γίνεται ορατό από τον κέρσορα που βρίσκεται κάτω από την τρέχουσα θέση. Έπειτα χρησιμοποιούμε το btn_south για να αλλάξουμε κατάσταση στο bit. Αφού έχουμε βάλει τις επιθυμητές εισόδους για τον υπολογισμό του κρυπτογραφήματος πρέπει να θέσουμε τον διακόπτη SW3 σε θέση logic 1. Η ίδια διαδικασία μπορεί να γίνει πολλές φορές αρκεί πρώτα να κάνουμε reset τον αλγόριθμο θέτοντας τον διακόπτη SW3 σε θέση logic 0.

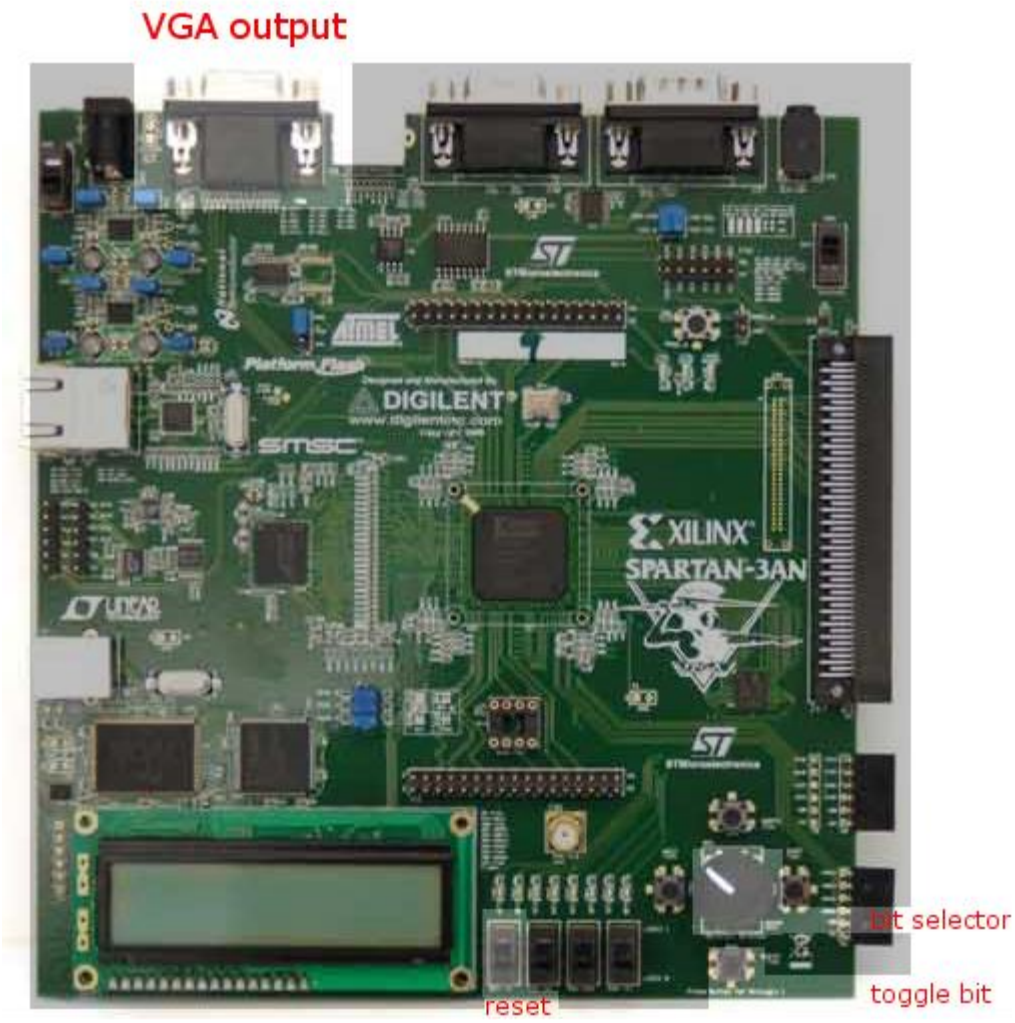


Figure 11 - Χειρισμός

Power/Utilization/Timing

Οι παρακάτω τιμές που αναφέρονται έχουν πραγματοποιηθεί με design goal: balanced στο ISE 13.2. Η μεγαλύτερη συχνότητα επιτεύχθηκε με αυτή την επιλογή (105mhz). Η ίδια συχνότητα προκύπτει και για minimum runtime. Η επιλογή για power optimization ρίχνει την συχνότητα στα 90mhz.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	284	11,776	2%	
Number used as Flip Flops	282			
Number used as Latches	2			
Number of 4 input LUTs	2,790	11,776	23%	
Number of occupied Slices	1,591	5,888	27%	
Number of Slices containing only related logic	1,591	1,591	100%	
Number of Slices containing unrelated logic	0	1,591	0%	
Total Number of 4 input LUTs	2,813	11,776	23%	
Number used as logic	2,762			
Number used as a route-thru	23			
Number used as Shift registers	28			
Number of bonded IOBs	37	372	9%	
Number of BUFGMUXs	2	24	8%	
Average Fanout of Non-Clock Nets	3.72			

Figure 12 - Utilization summary

Device	
Family	Spartan3a
Part	xc3s700a
Package	fg484
Grade	Commercial
Process	Typical
Speed Grade	-4

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Characterization	
PRODUCTION	v1.1,06-26-09

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.009	2	--	--
Logic	0.000	2810	11776	24
Signals	0.000	2913	--	--
IOs	0.000	37	372	10
Leakage	0.033			
Total	0.042			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	22.3	84.1	25.9

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.021	0.008	0.013
Vccaux	2.500	0.006	0.000	0.006
Vcco33	3.300	0.000	0.000	0.000
Vcco25	2.500	0.000	0.000	0.000

	Total	Dynamic	Quiescent
Supply Power (W)	0.042	0.009	0.033

Figure 13 - Power analysis

```
Clock to Setup on destination clock clk50_in
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk50_in       |    9.501|         |         |         |
---------------+---------+---------+---------+---------+


Timing summary:
---------------

Timing errors: 0  Score: 0  (Setup/Max: 0, Hold: 0)

Constraints cover 937 paths, 0 nets, and 775 connections

Design statistics:
   Minimum period:   9.501ns{1}   (Maximum frequency: 105.252MHz)
```

Figure 14 - Delay/Frequency

Βιβλιογραφία/links

https://bitbucket.org/kechap/sdes/src

https://bitbucket.org/kechap/simple-des/src

http://xilinx.com/products/boards/s3estarter/files/s3esk_rotary_encoder_interface.zip

http://mercury.webster.edu/aleshunas/COSC%205130/G-SDES.pdf