ΚΑΤΑΧΩΡΗΤΕΣ 1

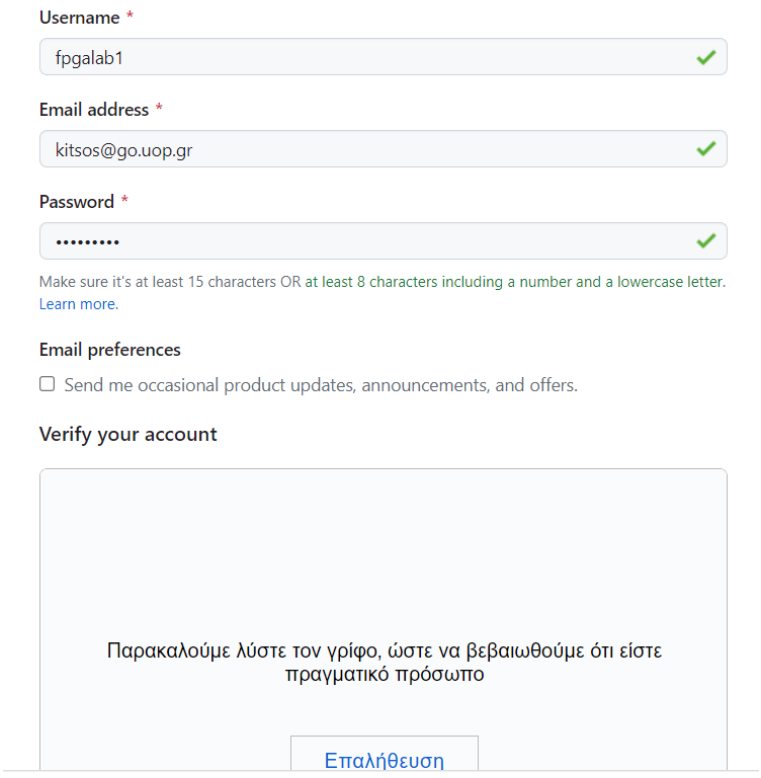
Σκοπός

Σε αυτή την εργαστηριακή άσκηση ο φοιτητής θα δημιουργήσει ένα αποθετήριο ώστε να αποθηκεύει τους κώδικες που θα συγγράφει στα πλαίσια του εργαστηρίου και έπειτα θα κάνει μια πρώτη γνωριμία με τη ροή σχεδιασμού ενός FPGA με τη χρήση του λογισμικού Quartus Prime Lite της Intel.

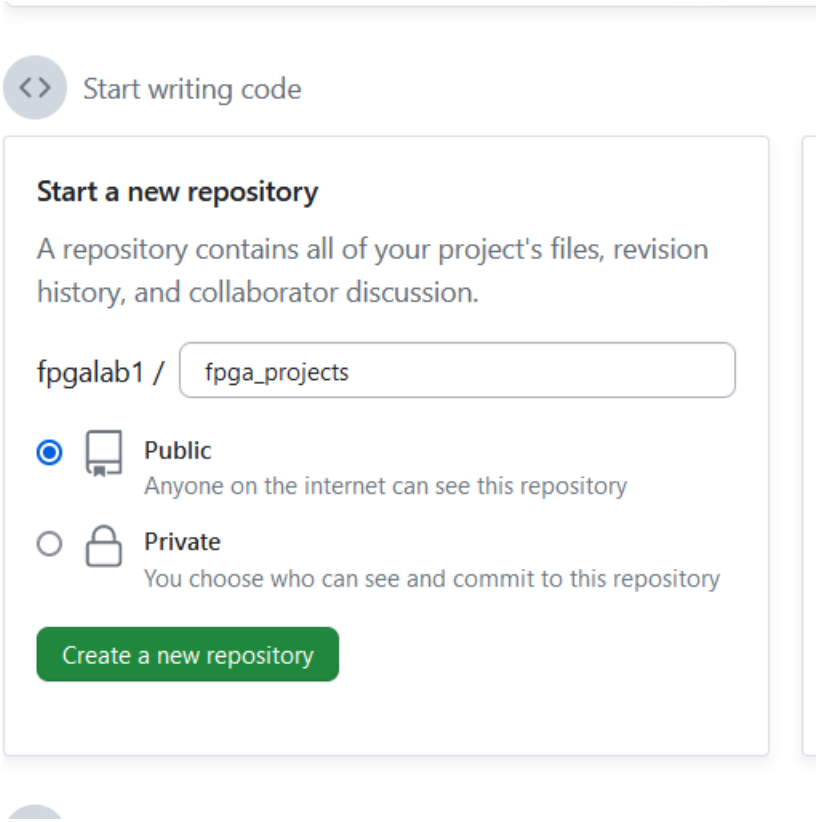
Δημιουργία ενός αποθετηρίου στο github

Για τη δημιουργία του χώρου σας στο github εκτελέστε τα παρακάτω βήματα.

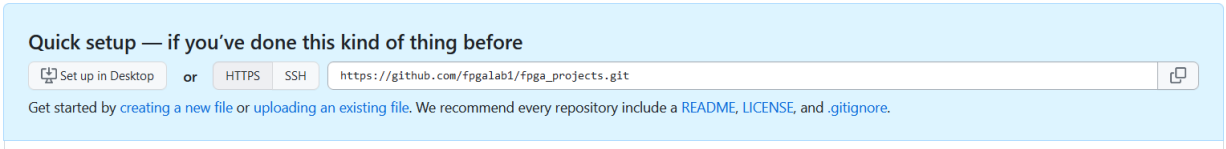
1) Δημιουργήστε ένα λογαριασμό στη σελίδα https://github.com/join με όνομα fpgalab (ή κάποιο παρόμοιο).



2) Έπειτα δημιουργήστε ένα δημόσιο (public) αποθετήριο με όνομα fpga_projects και επιλέγοντας Create a new repository

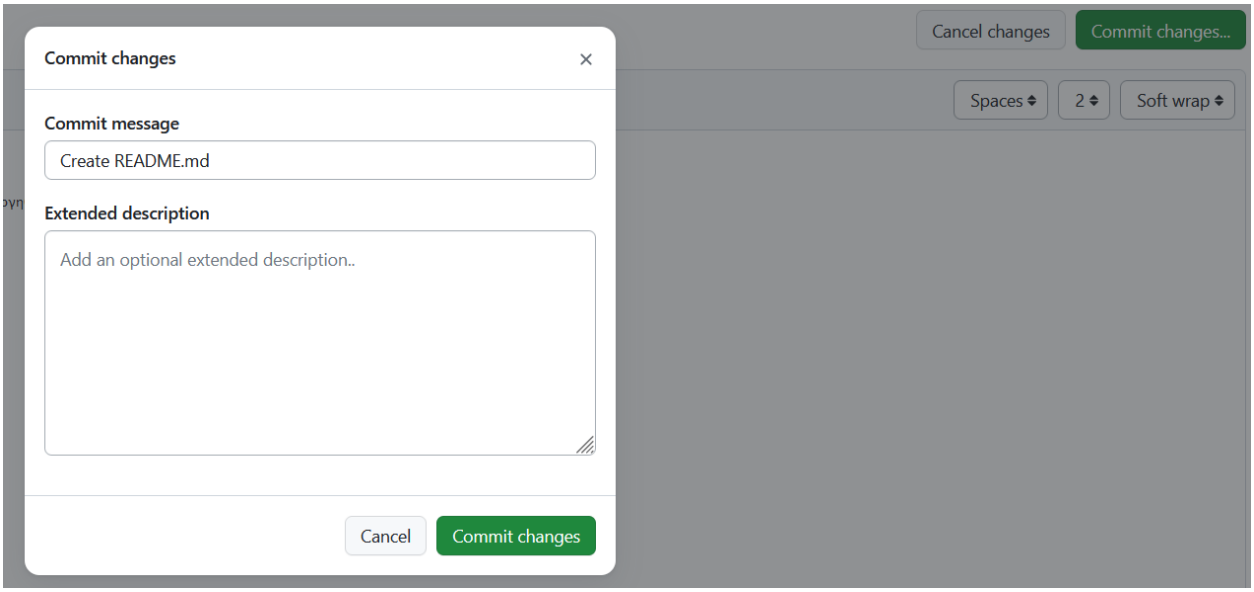


3) Δημιουργούμε επίσης ένα readme αρχείο με βασικές πληροφορίες του αποθετηρίου. Μπορεί να είναι πολύ αναλυτικό.



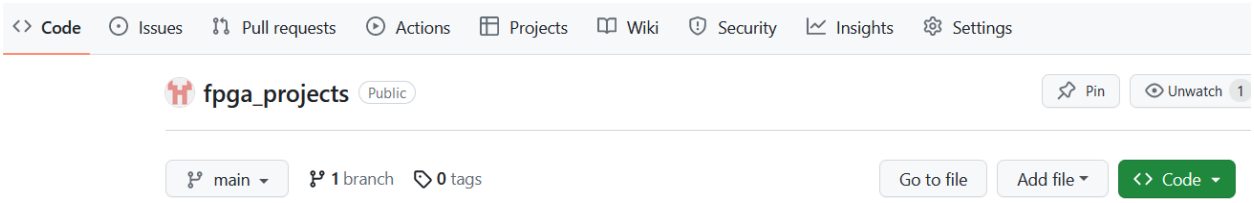
Το δικό μας αρχικά θα έχει πολύ βασικές πληροφορίες. Π.χ. Το αποθετήριο περιέχει όλα τα αρχεία VHDL των εργασιών που θα δημιουργηθούν στα πλαίσια του μαθήματος "Σχεδιασμός Ψηφιακών Συστημάτων σε FPGAs".

4) Επιλέγουμε Commit changes



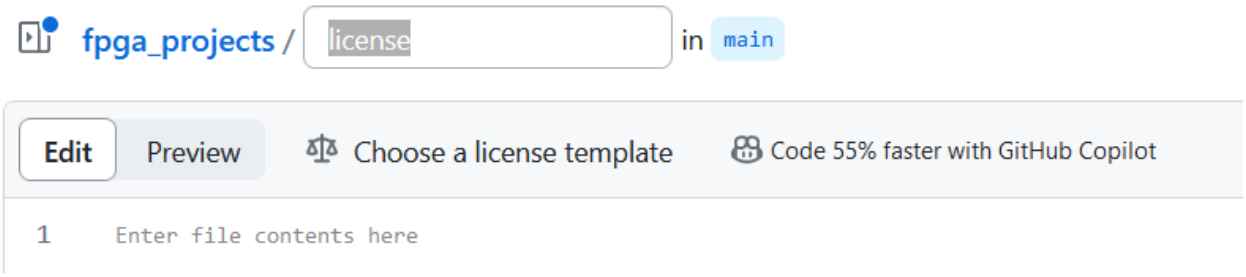
Και το αρχείο Readme.md είναι έτοιμο.

5) Έπειτα θα προσθέσουμε μια άδεια ανοιχτού κώδικα στο αποθετήριο μας. Αρχικά επιλέγετε Code.

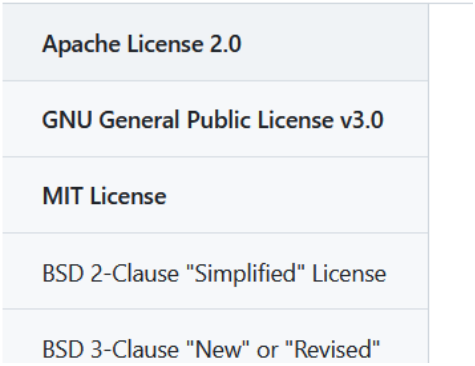


Επιλέγουμε Add file → Create new file και γράφουμε license.

Έπειτα επιλέγουμε Choose a license template



Από τις επιλογές, επιλέγουμε GNU General Public License v3.0

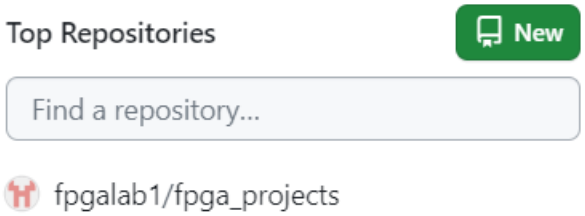


Έπειτα Review and submit και τέλος Commit change.

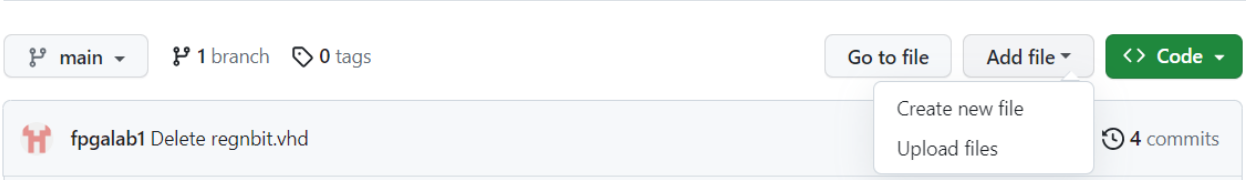
Το αποθετήριό σας είναι έτοιμο.

6) Μπορείτε να εισέρχεστε σε αυτό από τη σελίδα https://github.com/.

Επιλέξτε fpgalab1/fpga_projects

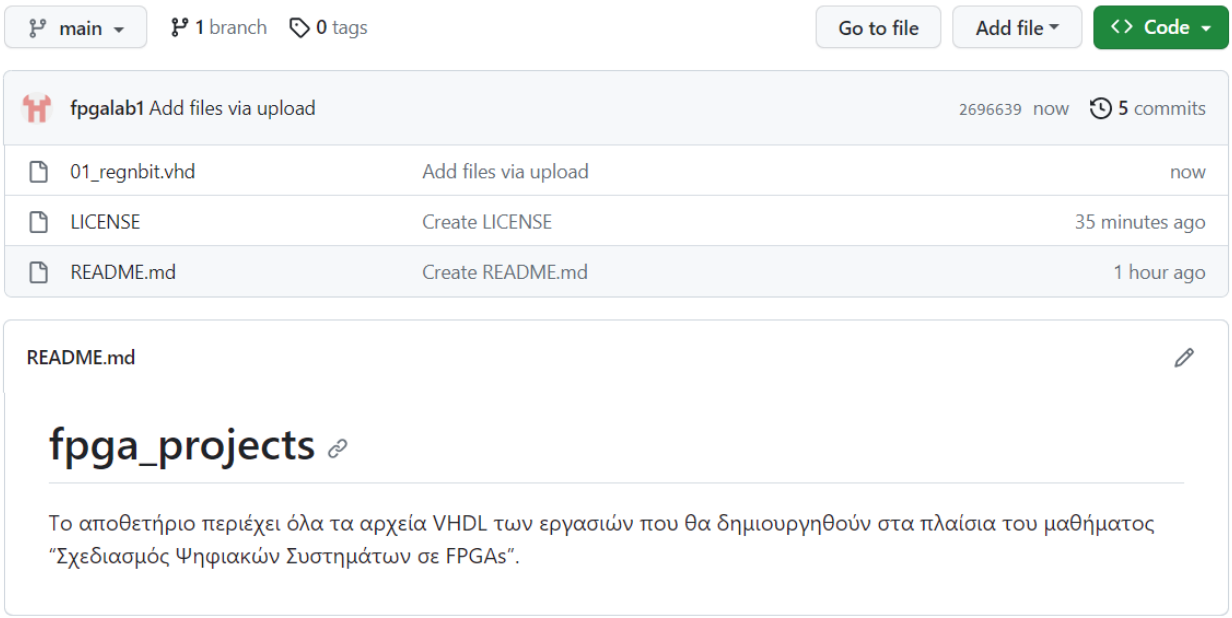


Έπειτα Add file→Upload files



και επιλέγουμε τα αρχεία που θέλουμε να αποθηκεύουμε στο αποθετήριό μας, π.χ. το αρχείο 01_regnbit.vhd και στο τέλος Commit changes.

7) Το αποθετήριο έχει τη μορφή που φαίνεται παρακάτω.



Καταχωρητές

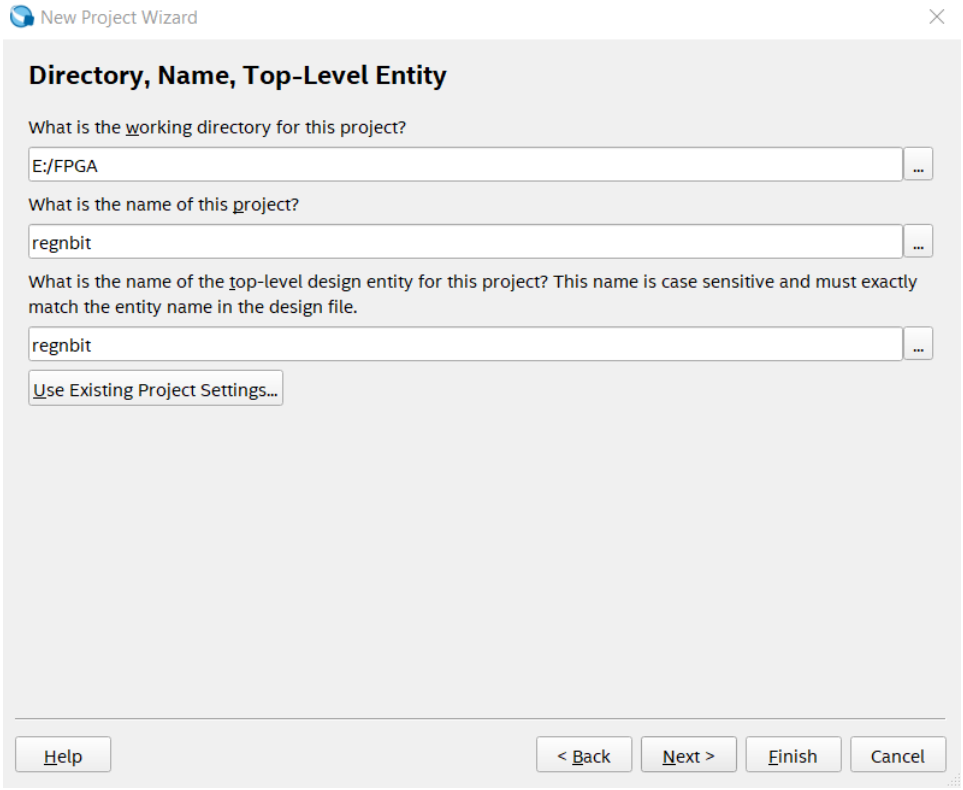
Θα χρησιμοποιήσουμε τον κώδικα ενός καταχωρητή εύρους n-bits με σήματα ελέγχου ld για τον έλεγχο παράλληλης φόρτωσης δεδομένων, inc για την αύξηση της τιμής του κατά ένα και rst για εκκαθάριση. Ο καταχωρητής αυτός θα πρέπει να χρησιμοποιεί τη δήλωση generic για τον ορισμό του μεγέθους του ώστε να μπορεί να χρησιμοποιηθεί για όλους τους καταχωρητές της σχετικά απλής ΚΜΕ.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all ;

entity regnbit is
generic (n: integer :=8);
port( din                       : in std_logic_vector(n-1 downto 0);
      clk,rst,ld          : in std_logic;
      inc                   : in std_logic;
      dout                  : out std_logic_vector(n-1 downto 0));
end regnbit;

architecture  arc  of regnbit is
signal temp : std_logic_vector(n-1 downto 0);
begin
      process(clk,rst)
      begin
            if  rst='0' then -- changed to use push button
                  temp<=(others=>'0');
            elsIf rising_edge(clk) then
```

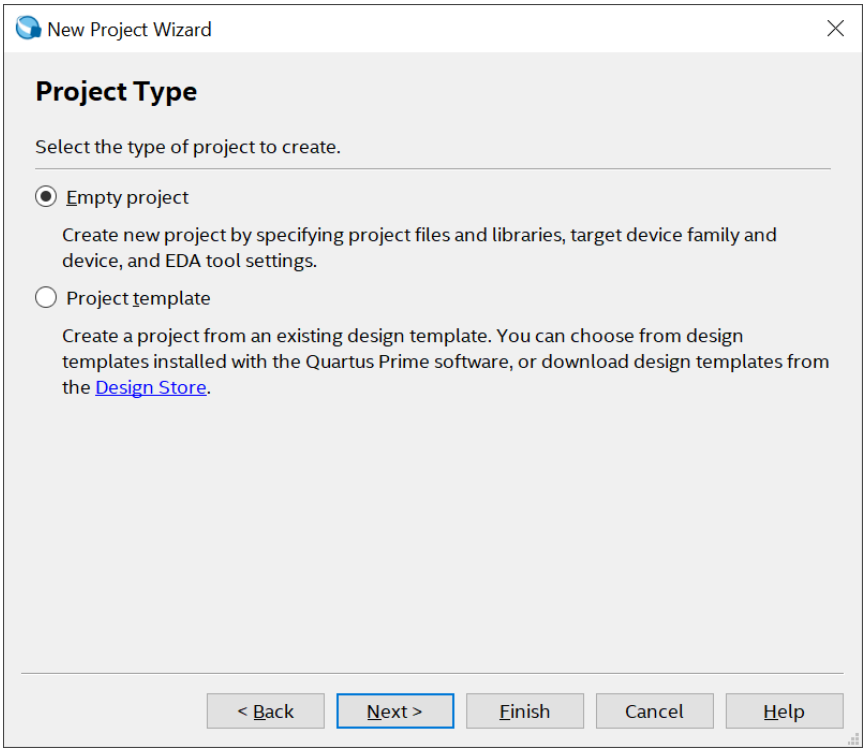
```
                    if ld='1' then
                        temp<=din;
                    elsif inc='1' then
                        temp<=temp+1;
                    end if;
                end if;
        end process;
dout<=temp;
end ;
```

βήμα 1: δημιουργία νέου project.

Δημιουργούμε ένα νέο project από το μενού **File|New Project Wizard**.

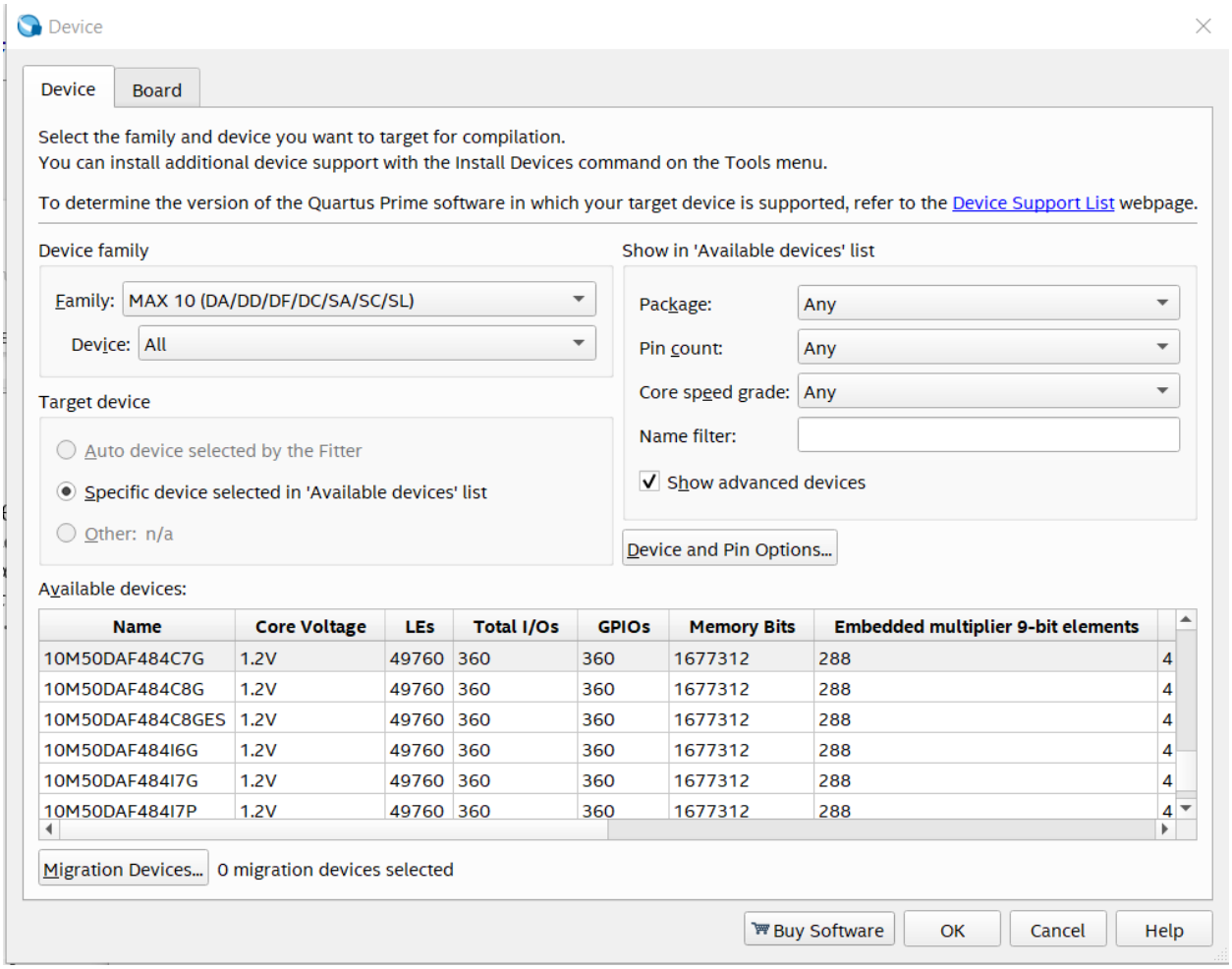


Εικόνα 1



Εικόνα 2

Αφού δώσουμε κάποια απαραίτητα στοιχεία όνομα, φάκελο που θα αποθηκευτεί το project μας κλπ. (Εικόνα 1) στη συνέχεια επιλέγουμε κενό project και αρχικά next (Εικόνα 2). Στη συνέχεια προσπερνάμε επίσης με next την επόμενη καρτέλα προσθήκης αρχείων και στη επιλογή συσκευής (Εικόνα 3) επιλέγουμε από την οικογένεια συσκευών MAX 10 και τη συσκευή 10M50DAF484C7G. Στο σημείο αυτό πατάμε finish για να ολοκληρώσουμε τη διαδικασία δημιουργίας του project.

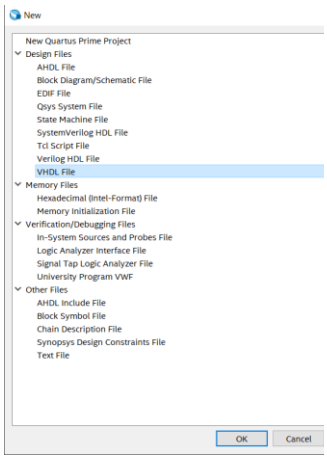


Εικόνα 3

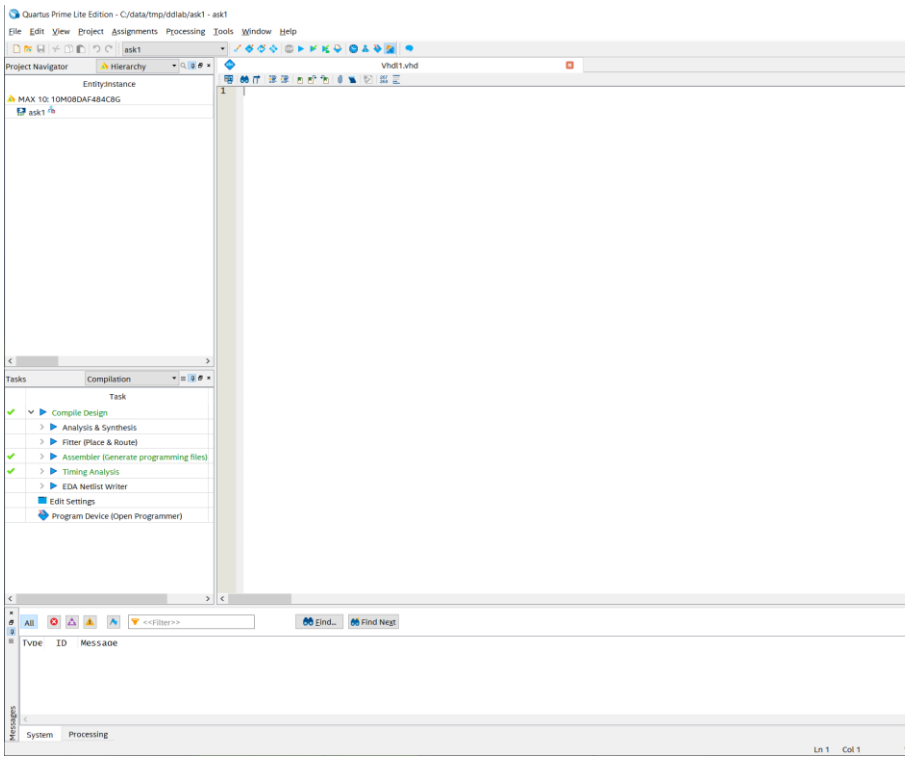
ΠΡΟΣΟΧΗ: *Δεν επιλέγουμε ποτέ τη προκαθορισμένη διαδρομή που μας προτείνει το Quartus Prime Lite αλλά κάποιο δικό μας φάκελο ΕΞΩ από το φάκελο εγκαταστάτης του λογισμικού.*

βήμα 2: προσθήκη σχεδιαστικού αρχείου – γλώσσα περιγραφής VHDL.

Δημιουργούμε ένα νέο αρχείο από το μενού **File|New** και επιλέγουμε VHDL File (Εικόνα 4).

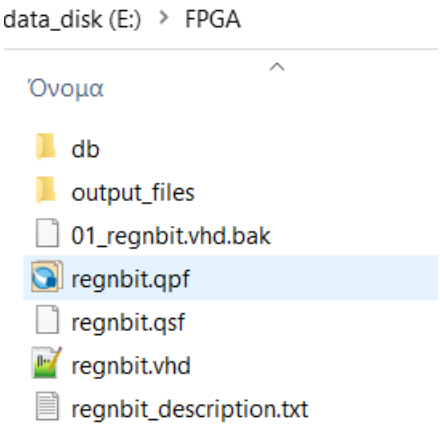


Εικόνα 4



Εικόνα 5

Στο περιβάλλον εργασίας που προκύπτει (Εικόνα 5), όπου πρόκειται ουσιαστικά για ένα κειμενογράφο, εισάγουμε το κύκλωμα μας. Τέλος ολοκληρώνουμε τη δημιουργία του σχεδιαστικού μας αρχείου αποθηκεύοντας το αρχείο μας στον ίδιο φάκελο που είναι το project μας (Εικόνα 6).



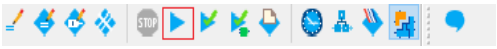
Εικόνα 6

ΠΡΟΣΟΧΗ: *Το όνομα με το οποίο αποθηκεύουμε το αρχείο μας θα πρέπει να είναι το ίδιο με το όνομα της οντότητας (entity).*

Σε αυτό το σημείο είναι απαραίτητο να δηλώσουμε το αρχείο που δημιουργήσαμε σαν το κύριο κύκλωμα του project μας. Αυτό επιτυγχάνεται, έχοντας ενεργό το παράθυρο με το αρχείο μας, από το μενού **Project|Set as Top-Level Entity**.

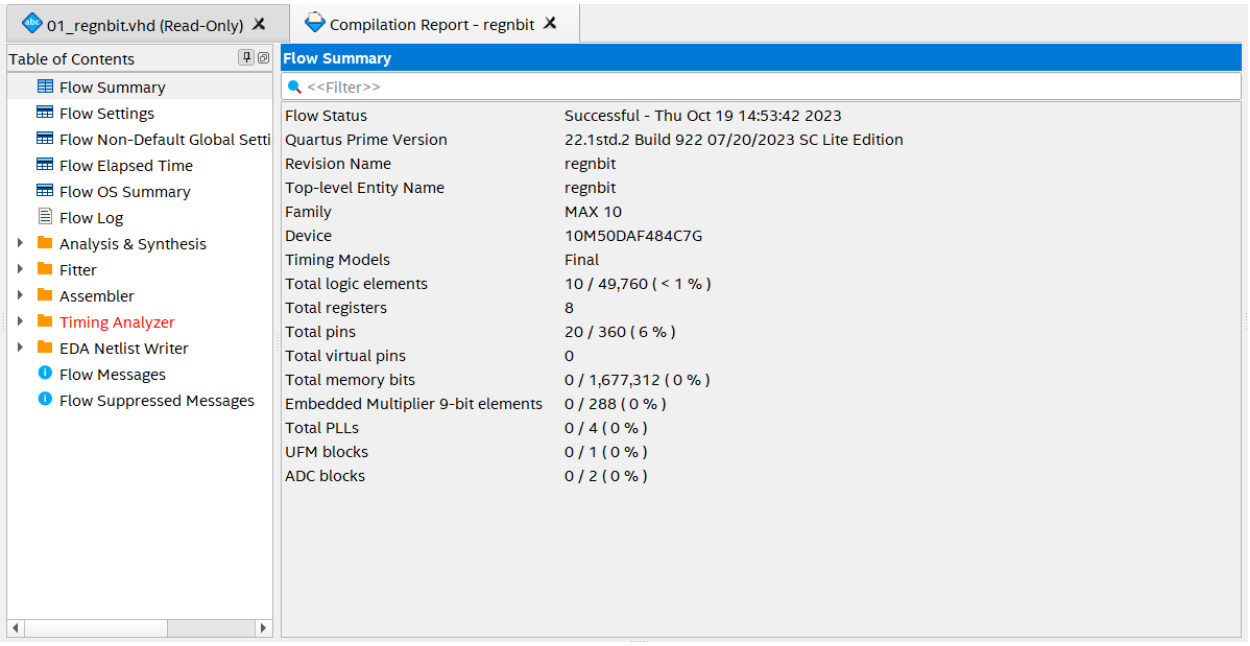
βήμα 3: συμβολομετάφραση – compilation του κυκλώματος.

Αφού ολοκληρώσουμε το βήμα 2, είναι αναγκαία η συμβολομετάφραση του κυκλώματος. Αυτό επιτυγχάνεται είτε με τη βοήθεια της γραμμής εργαλείων (Εικόνα 7) είτε από το μενού **Processing|Start Compilation**.



Εικόνα 7

Εφόσον δεν υπάρχει κάποιο πρόβλημα - λάθος στη σχεδίαση μας η συμβολομετάφραση ολοκληρώνεται και λαμβάνουμε την παρακάτω αναφορά (Εικόνα 8).



Εικόνα 8

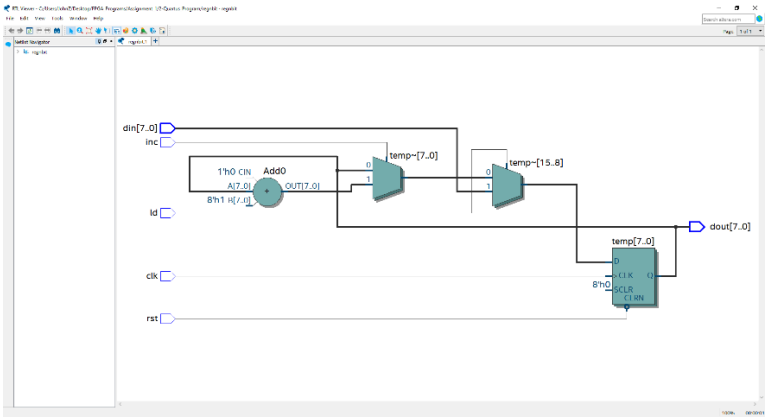
Εφόσον υπάρχει κάποιο λάθος επίσης λαμβάνουμε μία αναφορά λάθους ενώ με διπλό click πάνω στο μήνυμα λάθους μεταβαίνουμε αυτόματα στο σημείο περίπου που εντοπίζεται το λάθος αυτό. Σε αυτή την περίπτωση διορθώνουμε το λάθος και επαναλαμβάνουμε τη παραπάνω διαδικασία.

ΣΗΜΕΙΩΣΗ: *Είναι προφανές ότι λάθη που μπορούν να εντοπιστούν κατά τη συμβολομετάφραση, είναι συντακτικά εφόσον πρόκειται για αρχείο γλώσσας περιγραφής VHDL.*

Δεδομένου ότι τα FPGA είναι δομημένα εσωτερικά, μεταξύ άλλων, με Logic Elements (LEs) και Block RAM (BRAM) ενώ τα LEs υλοποιούνται με multiplexers και flip-flop πόσα και ποια από αυτά τα

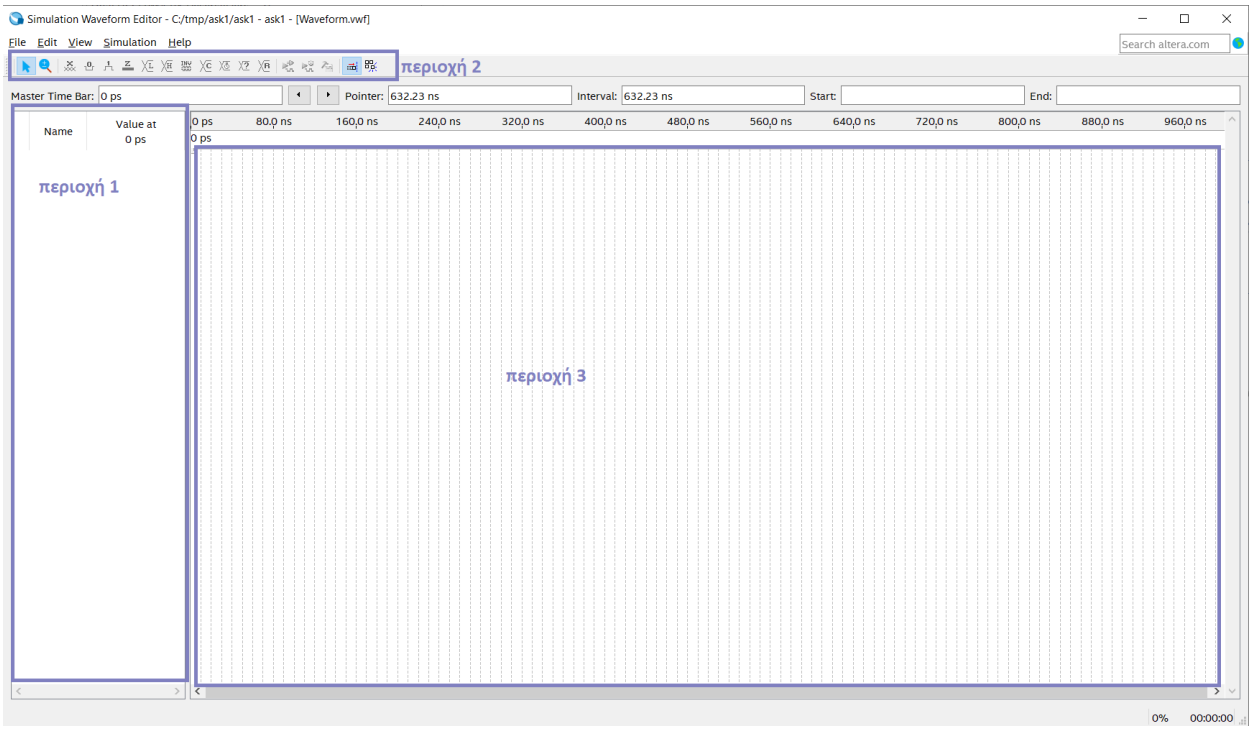
στοιχεία πιστεύετε έχουν χρησιμοποιηθεί για να υλοποιηθεί το παραπάνω κύκλωμα του καταχωρητή? Πως είναι συνδεδεμένα μεταξύ τους? Επιβεβαιώστε την απάντηση σας με τη χρήση του εργαλείου RTL Viewer από το μενού **Tools|Netlist Viewers|RTL Viewer**. Ήταν αυτό που υποψιαζόσασταν?

Τοποθετείστε εδώ τις παρατηρήσεις σας:



βήμα 4: λειτουργική εξομοίωση – simulation του κυκλώματος.

Δημιουργούμε ένα νέο αρχείο από το μενού **File|New** και επιλέγουμε University Program VWF. Το αρχείο που προκύπτει είναι ένα αρχείο κυματομορφών (Εικόνα 9) το οποίο θα χρησιμοποιήσουμε για να σχηματίσουμε τις κυματομορφές των εισόδων και μετά την εξομοίωση να λάβουμε τα αποτελέσματα δηλαδή τις κυματομορφές εξόδου.



Εικόνα 9

Εμφανίζουμε τις εισόδους και εξόδους του κυκλώματος μας κάνοντας δεξί click στη περιοχή 1, ακολουθώντας τις επιλογές **Insert Node or Bus|Node Finder|List** επιλέγοντας από τα διαθέσιμα σήματα (Nodes Found) όσα επιθυμούμε και μεταφέροντας τα στα δεξιά (Selected Nodes). Στη συνέχεια και αφού επιστρέψουμε στο αρχικό περιβάλλον εξομοίωσης πατώντας διαδοχικά οκ σχηματίζουμε τις κυματομορφές εισόδου. Αυτό επιτυγχάνεται επιλέγοντας όσο χρονικό διάστημα θέλουμε από όποιο σήμα επιθυμούμε στη **περιοχή 3** κρατώντας πατημένο το αριστερό πλήκτρο του mouse και δίνοντας τιμές από τα εργαλεία στη **περιοχή 2**. Τέλος για να «τρέξουμε» την εξομοίωση επιλέγουμε από το μενού simulation->Run Functional Simulation.

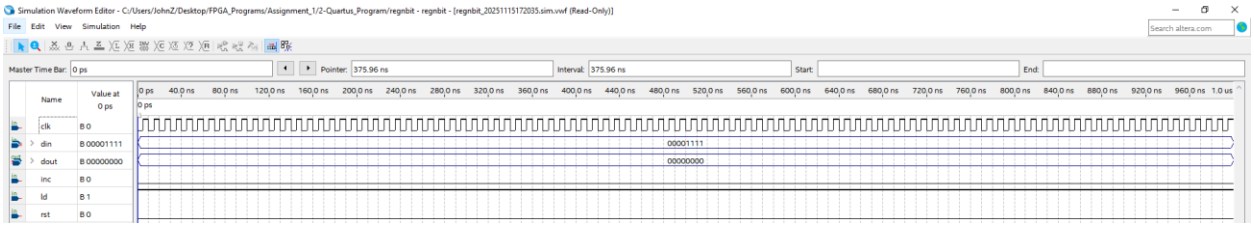
ΠΡΟΣΟΧΗ: *Για μια επιτυχημένη εξομοίωση θα πρέπει την πρώτη φορά να αφαιρέσετε την παράμετρο* ***-novopt*** *από τις ρυθμίσεις μέσω του μενού simulation->simulation settings*

Αφού ολοκληρώσετε την εξομοίωση, ακολουθώντας τα παραπάνω βήματα ελέγξτε την ορθή του κυκλώματος.

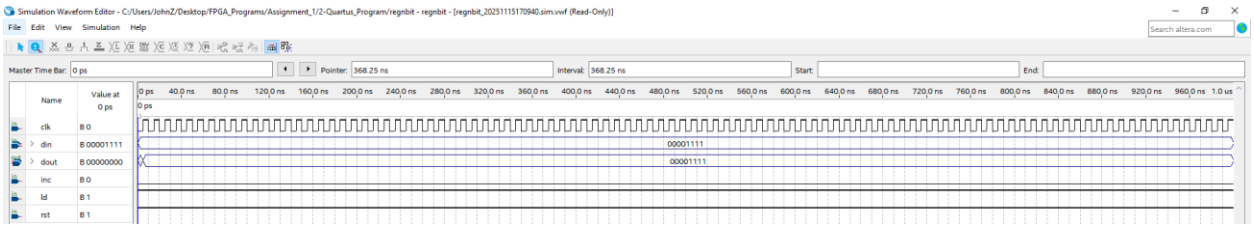
Τοποθετείστε εδώ τις παρατηρήσεις σας:

Για να μπορέσουμε να ελέγξουμε την λειτουργία του κυκλώματος, θα πρέπει να δοκιμάσουμε διαφορετικά σενάρια:

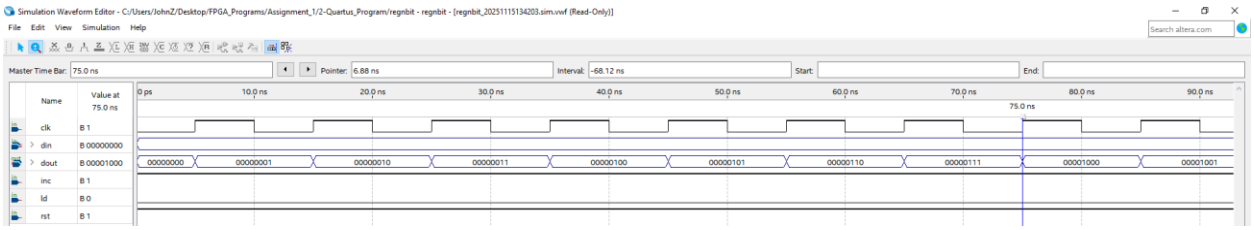
Σενάριο 1: ο καταχωρητής ***rst*** είναι 0. Αυτό ανεξαρτήτως με την είσοδο ***din***, θα θέτει τον καταχωρητή ***dout*** με μηδέν:



Σενάριο 2: Οι καταχωρητές ***rst = 1*** και ***ld = 1***, φορτώνει τον αριθμό στην είσοδο: $(00001111)_2 = (15)_{10}$ και μετά από ένα παλμό ρολογιού φορτώνει τον αριθμό στην έξοδο.



Σενάριο 3: Οι καταχωρητές ***rst = 1, ld = 0,*** και ***inc = 1***, φορτώνει τον αριθμό που θέτουμε στην είσοδο: $(00000000)_2 = (0)_{10}$ και προσθέτει κατά ένα στην θετική ακμοπυροδότηση του ρολογιού όπως φαίνεται στην παρακάτω φωτογραφία:



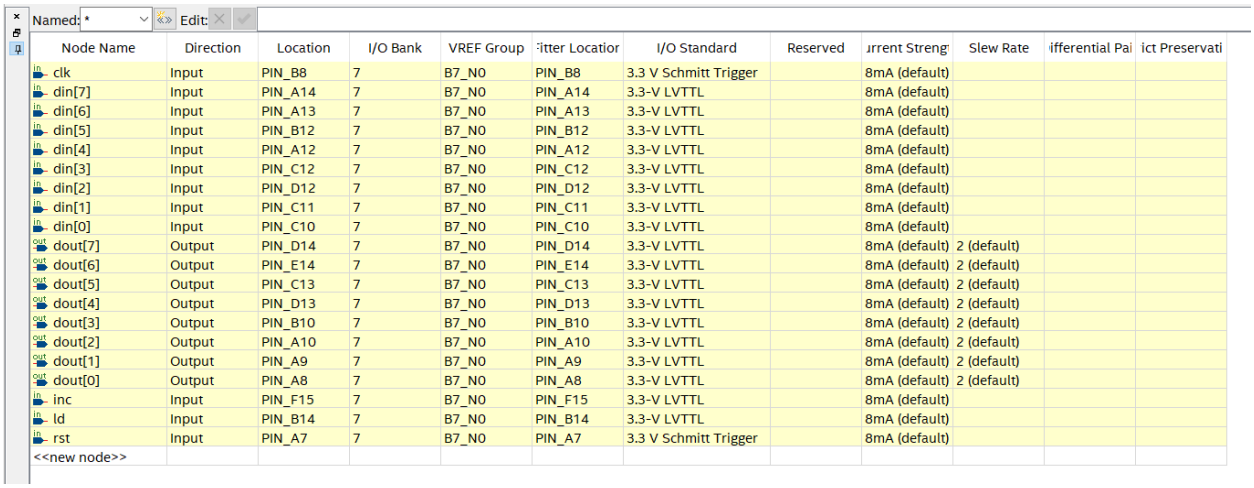
βήμα 5: έλεγχός του κυκλώματος στην αναπτυξιακή πλακέτα.

Ο τελικός έλεγχος ορθής λειτουργίας, θα πραγματοποιηθεί σε ένα ολοκληρωμένο κύκλωμα στο οποίο θα έχει προγραμματισθεί με το σχεδιαζόμενο κύκλωμα. Για το λόγο αυτό θα χρησιμοποιηθεί η αναπτυξιακή κάρτα DE10-Lite της Intel. Η αναπτυξιακή κάρτα DE10-Lite διαθέτει το ολοκληρωμένο κύκλωμα προγραμματιζόμενης λογικής, MAX10 που ανήκει στην κατηγορία FPGA. Για το λειτουργικό έλεγχο θα χρησιμοποιηθούν μια σειρά από βοηθητικά στοιχεία (push button, led, switch) τα οποία είναι διαθέσιμα στην αναπτυξιακή πλακέτα και τα οποία είναι απευθείας συνδεδεμένα στα pin του FPGA. Τα στοιχεία αυτά είναι συνδεδεμένα στο κύκλωμα του καταχωρητή όπως φαίνεται στο που ακολουθεί.

din	switch[0] .. switch[7]
clk	key0
rst	Key1
ld	switch[8]
inc	switch[9]
dout	led[0] .. led[7]

Πίνακας 1

Δεδομένου ότι τα διάφορα στοιχεία που αναφέραμε παραπάνω είναι συνδεδεμένα σταθερά μέσω τυπωμένου κυκλώματος σε συγκεκριμένα pin του FPGA θα πρέπει να οδηγήσουμε σε αυτά τις εισόδους και τις εξόδους του καταχωρητή μας. Αυτό επιτυγχάνεται με τη χρήση του εργαλείου Pin Planner το καλούμε μέσα από το μενού **Assignments|Pin Planner**. Μέσω του εργαλείου αυτού κάνουμε τις ρυθμίσεις που φαίνονται στην Εικόνα 10. Δώστε προσοχή στις στήλες location και I/O Standard.



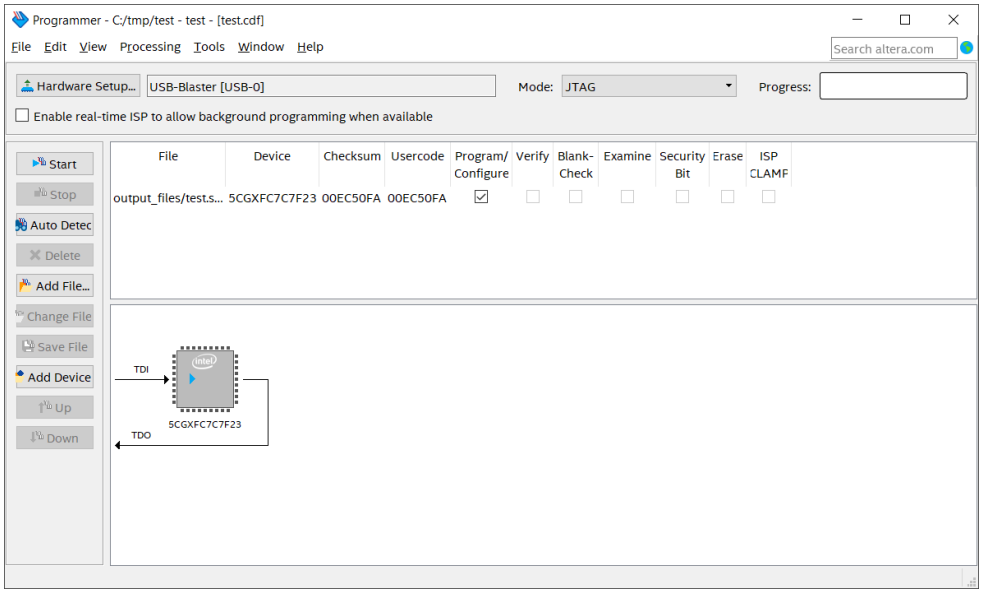
Named: * Edit:

Node Name	Direction	Location	I/O Bank	VREF Group	itter Locatior	I/O Standard	Reserved	urrent Streng	Slew Rate	ifferential Pai	ict Preservati
clk	Input	PIN_B8	7	B7_N0	PIN_B8	3.3 V Schmitt Trigger		8mA (default)			
din[7]	Input	PIN_A14	7	B7_N0	PIN_A14	3.3-V LVTTL		8mA (default)			
din[6]	Input	PIN_A13	7	B7_N0	PIN_A13	3.3-V LVTTL		8mA (default)			
din[5]	Input	PIN_B12	7	B7_N0	PIN_B12	3.3-V LVTTL		8mA (default)			
din[4]	Input	PIN_A12	7	B7_N0	PIN_A12	3.3-V LVTTL		8mA (default)			
din[3]	Input	PIN_C12	7	B7_N0	PIN_C12	3.3-V LVTTL		8mA (default)			
din[2]	Input	PIN_D12	7	B7_N0	PIN_D12	3.3-V LVTTL		8mA (default)			
din[1]	Input	PIN_C11	7	B7_N0	PIN_C11	3.3-V LVTTL		8mA (default)			
din[0]	Input	PIN_C10	7	B7_N0	PIN_C10	3.3-V LVTTL		8mA (default)			
dout[7]	Output	PIN_D14	7	B7_N0	PIN_D14	3.3-V LVTTL		8mA (default)	2 (default)		
dout[6]	Output	PIN_E14	7	B7_N0	PIN_E14	3.3-V LVTTL		8mA (default)	2 (default)		
dout[5]	Output	PIN_C13	7	B7_N0	PIN_C13	3.3-V LVTTL		8mA (default)	2 (default)		
dout[4]	Output	PIN_D13	7	B7_N0	PIN_D13	3.3-V LVTTL		8mA (default)	2 (default)		
dout[3]	Output	PIN_B10	7	B7_N0	PIN_B10	3.3-V LVTTL		8mA (default)	2 (default)		
dout[2]	Output	PIN_A10	7	B7_N0	PIN_A10	3.3-V LVTTL		8mA (default)	2 (default)		
dout[1]	Output	PIN_A9	7	B7_N0	PIN_A9	3.3-V LVTTL		8mA (default)	2 (default)		
dout[0]	Output	PIN_A8	7	B7_N0	PIN_A8	3.3-V LVTTL		8mA (default)	2 (default)		
inc	Input	PIN_F15	7	B7_N0	PIN_F15	3.3-V LVTTL		8mA (default)			
ld	Input	PIN_B14	7	B7_N0	PIN_B14	3.3-V LVTTL		8mA (default)			
rst	Input	PIN_A7	7	B7_N0	PIN_A7	3.3 V Schmitt Trigger		8mA (default)			
<<new node>>											

Εικόνα 10

ΣΗΜΕΙΩΣΗ: *Πρέπει να γνωρίζεται ότι τα switches δίνουν λογικό 1 όταν είναι τοποθετημένα στη θέση κοντύτερα στο FPGA και λογικό 0 στη άλλη θέση ενώ τα push button δίνουν λογικό 0 όταν είναι πατημένα και λογικό 1 ελεύθερα.*

Τέλος από την επιλογή **Tools|Programmer** του μενού γίνεται ο προγραμματισμός της συσκευής. Δώστε προσοχή στο σωστό όνομα του αρχείου προγραμματισμού (πρέπει να ταυτίζεται με το όνομα του project) και την σωστή συσκευή 10M50DAF484C7G . Αν όλα είναι σωστά από το πλήκτρο Start ξεκινά ο προγραμματισμός (Εικόνα 11).



Εικόνα 11

Δίνοντας τιμές στους διακόπτες και με τη χρήση των δύο button ελέγξτε την ορθή του κυκλώματος.