Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Μικροϋπολογιστων και Ψηφιακων Συστηματων

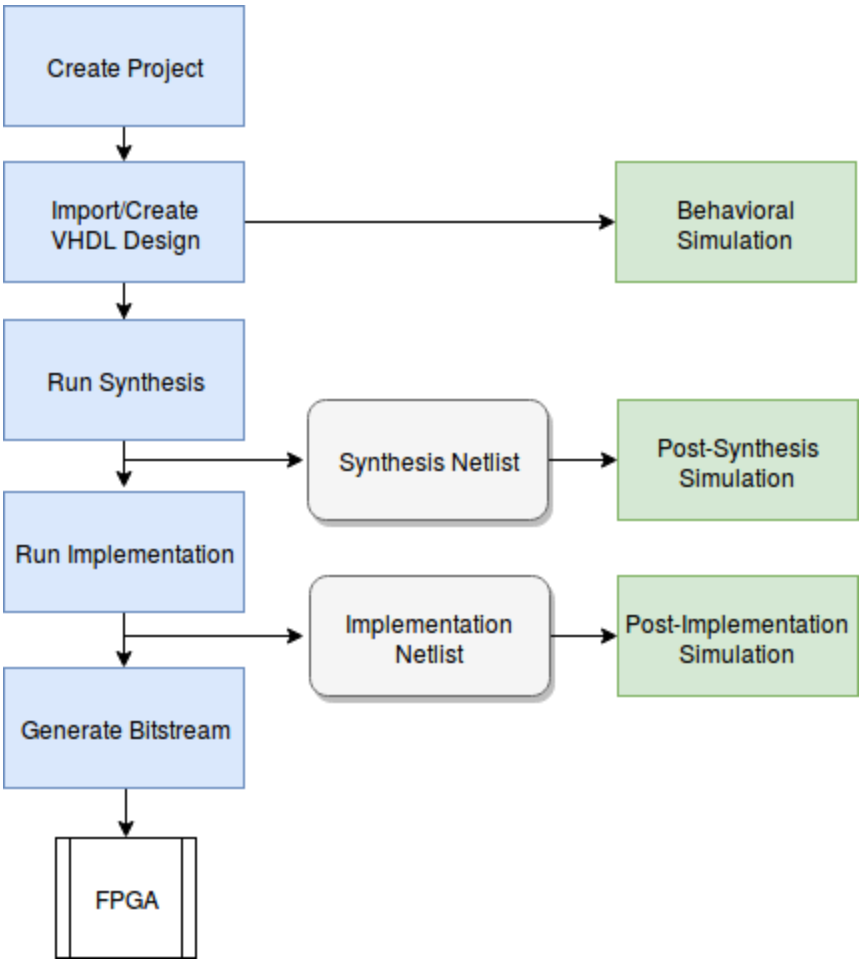
Ψηφιακά Συστήματα VLSI

Οδηγός Χρήσης του Εργαλείου Xilinx Vivado 2014.2

Αθήνα, 2018

1. Ροή Σχεδίασης σε FPGA

Η τυπική ροή που ακολουθείται για την σχεδίαση κυκλωμάτων σε FPGA παρουσιάζεται στο ακόλουθο σχήμα.

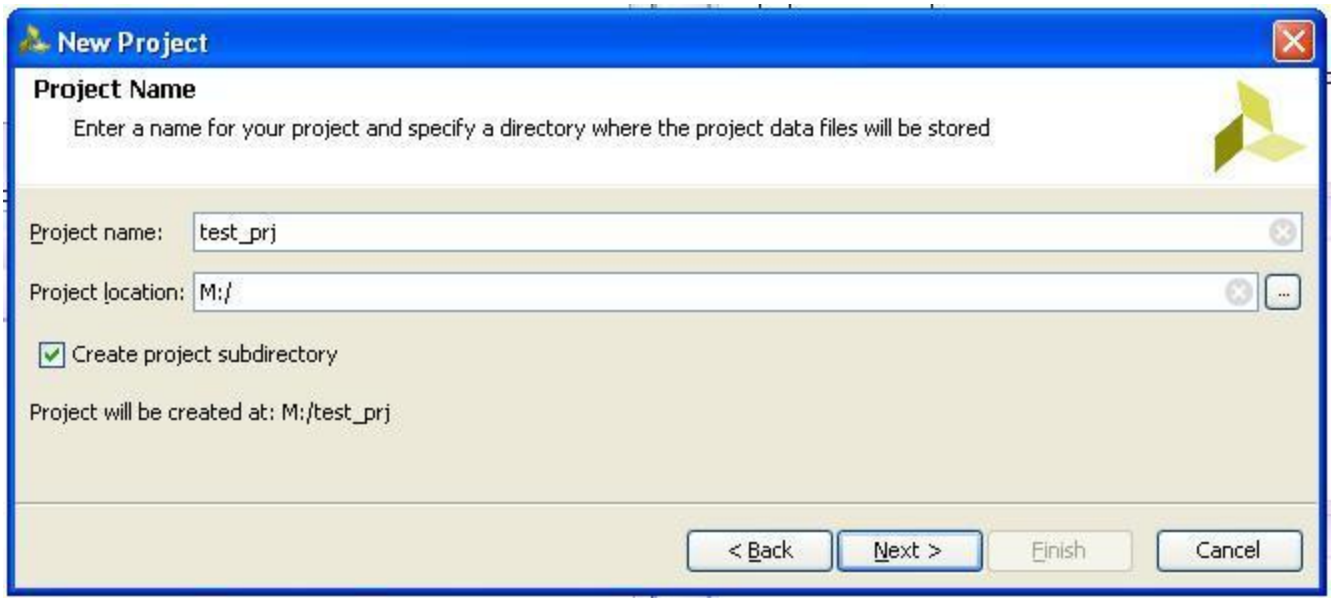


Στη συνέχεια, αναλύουμε και παρουσιάζουμε ξεχωριστά το κάθε ένα από τα παραπάνω βήματα, με χρήση του εργαλείου **Xilinx Vivado 2014.2** και της αναπτυξιακής FPGA πλακέτας **Digilent Zybo Zynq-7000**.

2. Δημιουργία Νέου Project

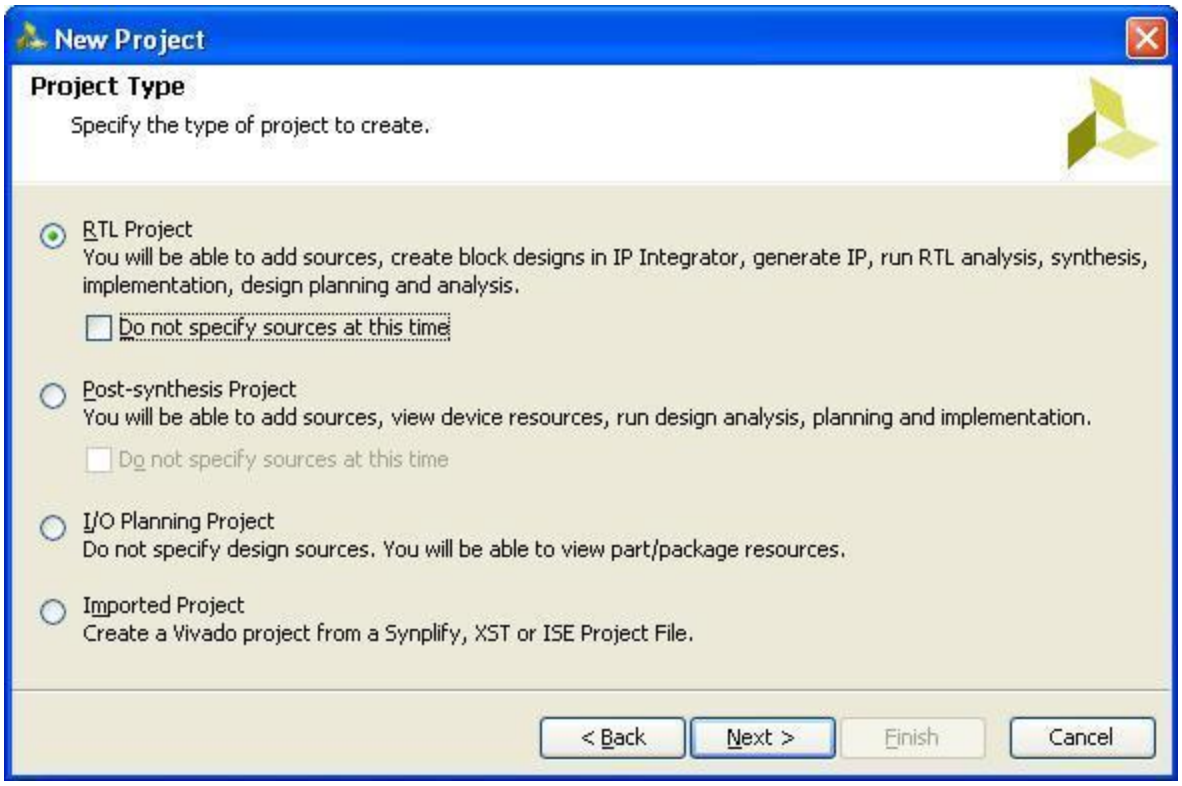
Αφού έχουμε ανοίξει το εργαλείο Vivado, για να ξεκινήσουμε να εργαζόμαστε σε αυτό θα πρέπει να δημιουργήσουμε ένα καινούριο project. Η δημιουργία ενός νέου project συνοψίζεται στα ακόλουθα βήματα:

1. Πατάμε **File > New Project…**, επιλέγουμε **Next** και εμφανίζεται το ακόλουθο παράθυρο:



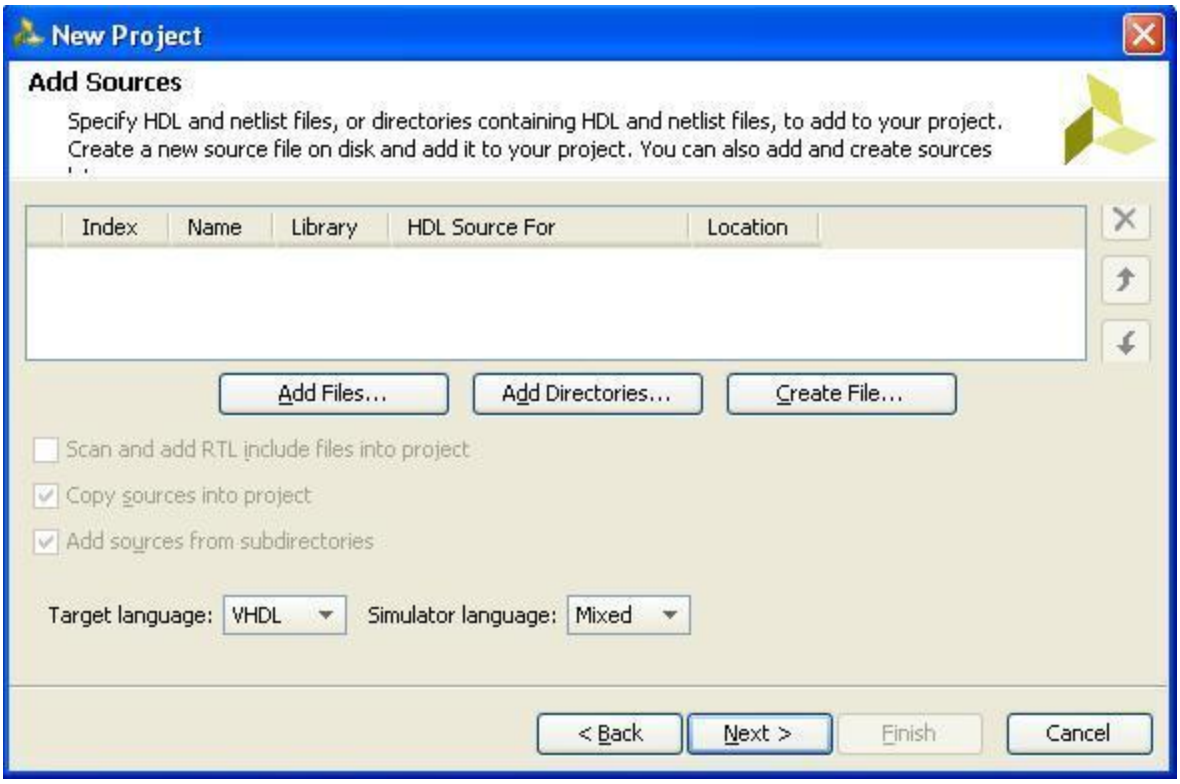
   Εισάγουμε το όνομα που θέλουμε να έχει το project (**Project name**), και μετά επιλέγουμε το path του directory στο οποίο θέλουμε αυτό να δημιουργηθεί (**Project location**), δημιουργώντας έτσι ένα subdirectory στο οποίο θα στηθεί το project.

2. Πατάμε **Next** και εμφανίζεται ένα παράθυρο επιλογής του τύπου project που θέλουμε να δημιουργήσουμε:



Επιλέγουμε τον τύπο **RTL Project**, δηλώνουμε δηλαδή ότι θα δημιουργήσουμε ένα project στο οποίο θα μπορούμε να εισάγουμε αρχεία περιγραφής του σχεδιασμού μας (σε γλώσσα περιγραφής υλικού HDL) και να δημιουργήσουμε IP blocks. Δηλώνουμε, επίσης, ότι θα ακολουθήσουμε την τυπική ροή σχεδίασης (RTL ανάλυση, σύνθεση, υλοποίηση, προσομοίωση, κτλ.).
Σε περίπτωση που δεν θέλουμε να εισάγουμε σε αυτό το σημείο τα αρχεία του σχεδιασμού μας, τικάρουμε την επιλογή **Do not specify sources at this time**. Η εισαγωγή αρχείων σε αυτό το σημείο είναι προαιρετική.

3. Πατάμε **Next** και μας εμφανίζεται το παράθυρο εισαγωγής των αρχείων του σχεδιασμού (σε περίπτωση που δεν έχουμε τικάρει την προηγούμενη επιλογή):



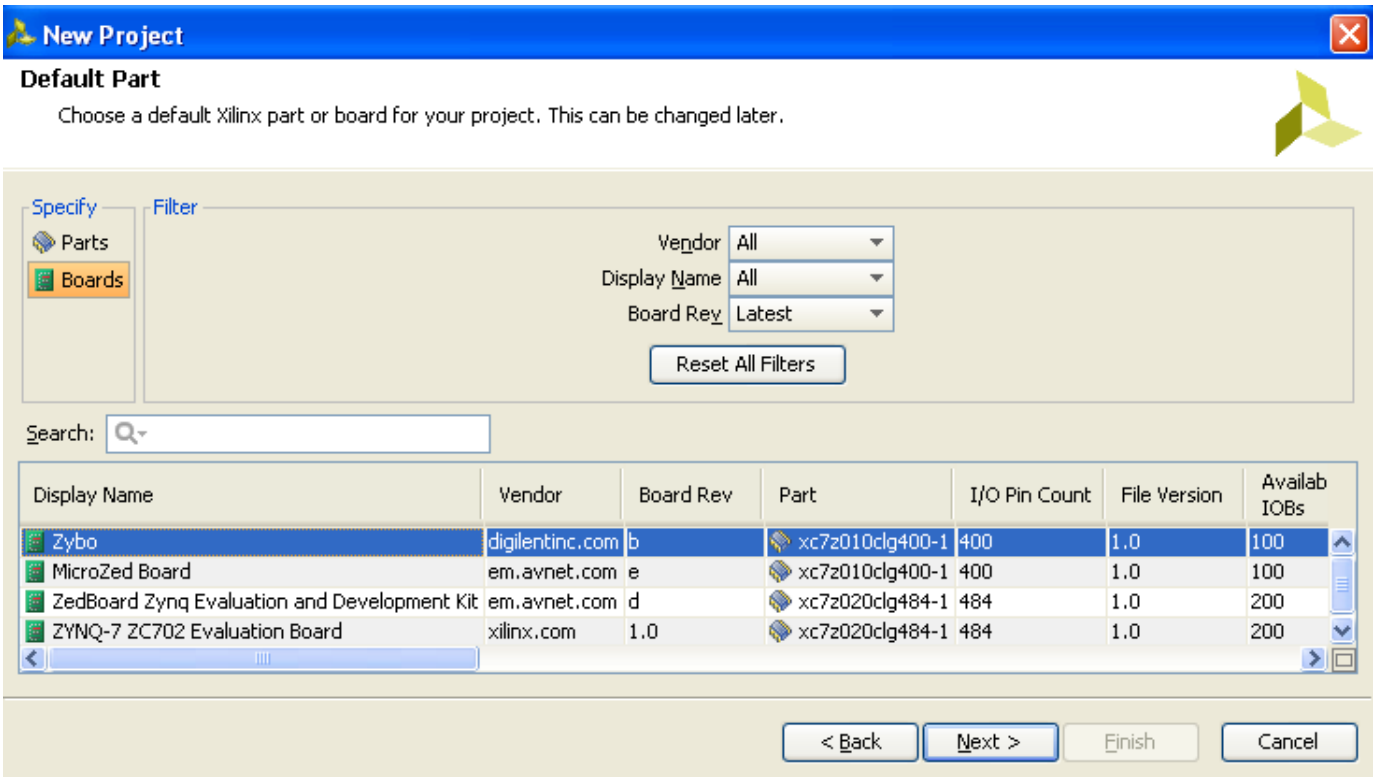
Μέσω της επιλογής **Add Files** μπορούμε να προσθέσουμε αρχεία HDL, ενώ μέσω της επιλογής **Create File** μπορούμε να δημιουργήσουμε νέα αρχεία.
Στα πλαίσια του Εργαστηρίου θα χρησιμοποιήσουμε τη γλώσσα περιγραφής υλικού **VHDL**, την οποία και επιλέγουμε στη φόρμα **Target language** και **Simulator language** (γλώσσα περιγραφής αρχείων σχεδιασμού και προσομοίωσης, αντίστοιχα).

4. Πατάμε **Next** και μας εμφανίζεται το παράθυρο εισαγωγής κάποιου υπάρχοντος IP. Η εισαγωγή αυτή είναι προαιρετική.

5. Πατάμε **Next** και μας εμφανίζεται το παράθυρο εισαγωγής αρχείων που να ορίζουν χρονικούς και φυσικούς περιορισμούς (*timing and physical constraints*):
   a. Στο αρχείο χρονικών περιορισμών δηλώνεται οτιδήποτε αφορά τους χρονισμούς του FPGA, όπως για παράδειγμα η στοχευμένη συχνότητα λειτουργίας του FPGA και οι χρονισμοί των εξωτερικών περιφερειακών.
   b. Στο αρχείο φυσικών περιορισμών ορίζονται πληροφορίες όπως οι φυσικές θέσεις των εισόδων/εξόδων του FPGA, καθώς και των διακοπτών και των LEDs που βρίσκονται στην πλακέτα του FPGA.

   Σε αυτό το σημείο, η εισαγωγή αυτών των αρχείων είναι προαιρετική. Οι περιορισμοί είναι απαραίτητοι να εισαχθούν σε περίπτωση που θέλουμε να κατεβάσουμε και να τρέξουμε τον σχεδιασμό μας στην πλακέτα.

6. Πατάμε **Next** και μας εμφανίζεται το παράθυρο επιλογής του FPGA και της πλακέτας που θα χρησιμοποιήσουμε:



Πηγαίνουμε στην καρτέλα **Boards**, και κατόπιν επιλέγουμε την αναπτυξιακή πλακέτα **Zybo**, η οποία θα χρησιμοποιηθεί στα πλαίσια του Εργαστηρίου.

7. Πατάμε **Next** και ανοίγει ένα παράθυρο όπου παρουσιάζεται μία σύνοψη των ιδιοτήτων του project που έχουμε δημιουργήσει.

8. Πατάμε **Finish** και το project μας έχει δημιουργηθεί!

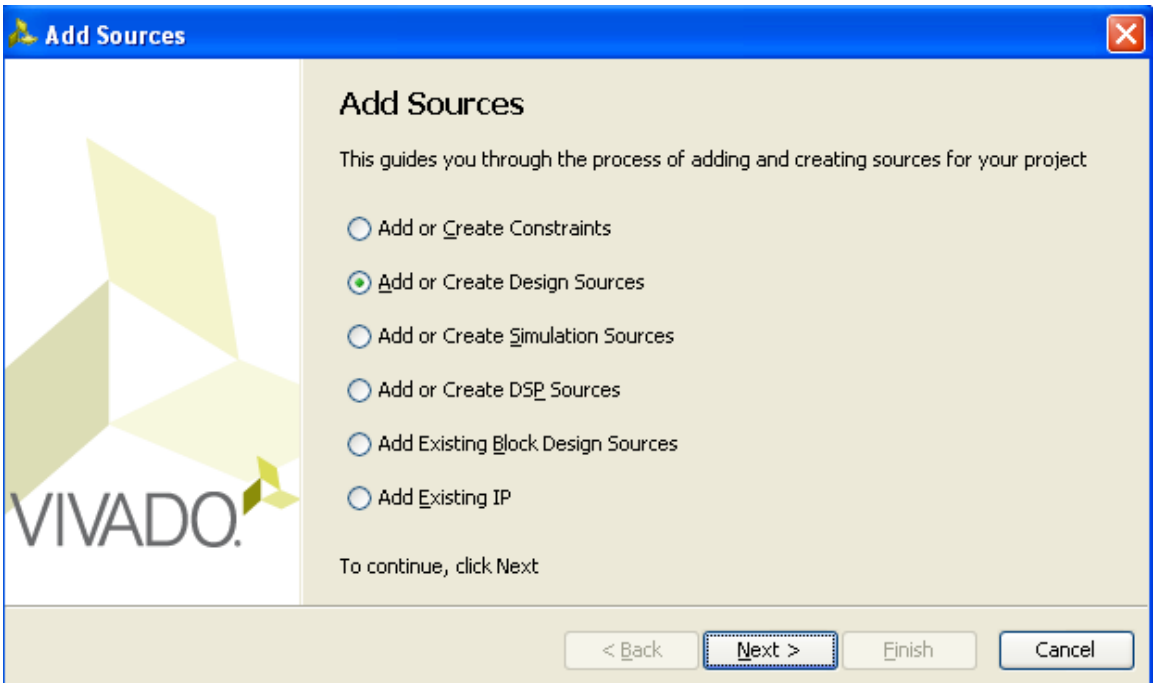
ΣΗΜΕΙΩΣΗ 1: Σε περίπτωση που θέλουμε να ανοίξουμε ένα project που έχουμε ήδη δημιουργήσει, πατάμε **File > Open Project…** και επιλέγουμε το project που θέλουμε (το αρχείο με κατάληξη **.xpr**). Επίσης, στο πλαίσιο **Recent Files** υπάρχουν τα πιο πρόσφατα project, και μπορούμε να τα ανοίξουμε επιλέγοντας τα.

ΣΗΜΕΙΩΣΗ 2: Μπορούμε να δημιουργήσουμε/ανοίξουμε ένα project και μέσω του πλαισίου **Quick Start**, επιλέγοντας **Create New Project**/**Open Project**.

3. Εισαγωγή/Δημιουργία HDL Αρχείων

Αφού έχουμε δημιουργήσει ένα νέο project, θα πρέπει να εισάγουμε HDL αρχεία που να περιγράφουν τον σχεδιασμό μας ή να δημιουργήσουμε καινούρια. Να σημειώσουμε πως αρχεία HDL μπορούν να εισαχθούν και κατά τη διάρκεια δημιουργίας του project, όπως δείξαμε πιο πάνω.
Στα πλαίσια του Εργαστηρίου, θα χρησιμοποιήσουμε τη γλώσσα περιγραφής υλικού VHDL. Για να εισάγουμε/δημιουργήσουμε ένα αρχείο VHDL κώδικα εκτελούμε τα ακόλουθα βήματα:

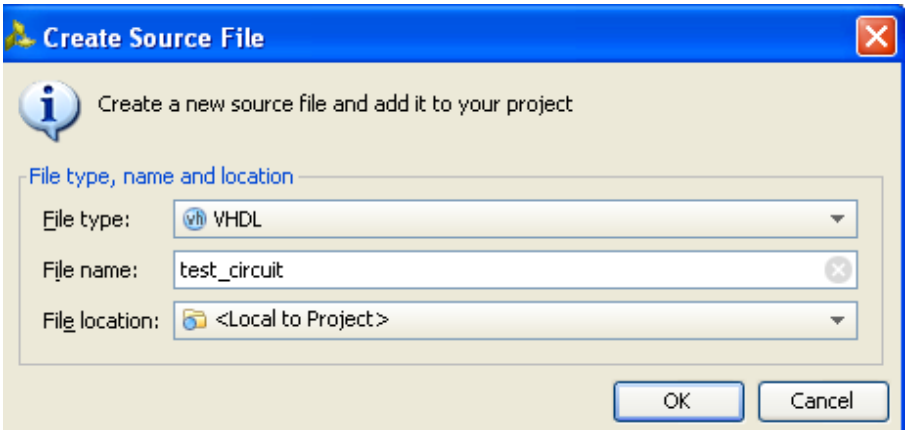
1. Στο περιβάλλον **Flow Navigator**, πατάμε **Project Manager > Add Sources** και εμφανίζεται το ακόλουθο παράθυρο:



Επιλέγουμε **Add or Create Design Sources**.

2. Πατάμε **Next** και εμφανίζεται ένα παράθυρο για να εισάγουμε (**Add Files...**) ή να δημιουργήσουμε (**Create File...**) VHDL αρχεία.

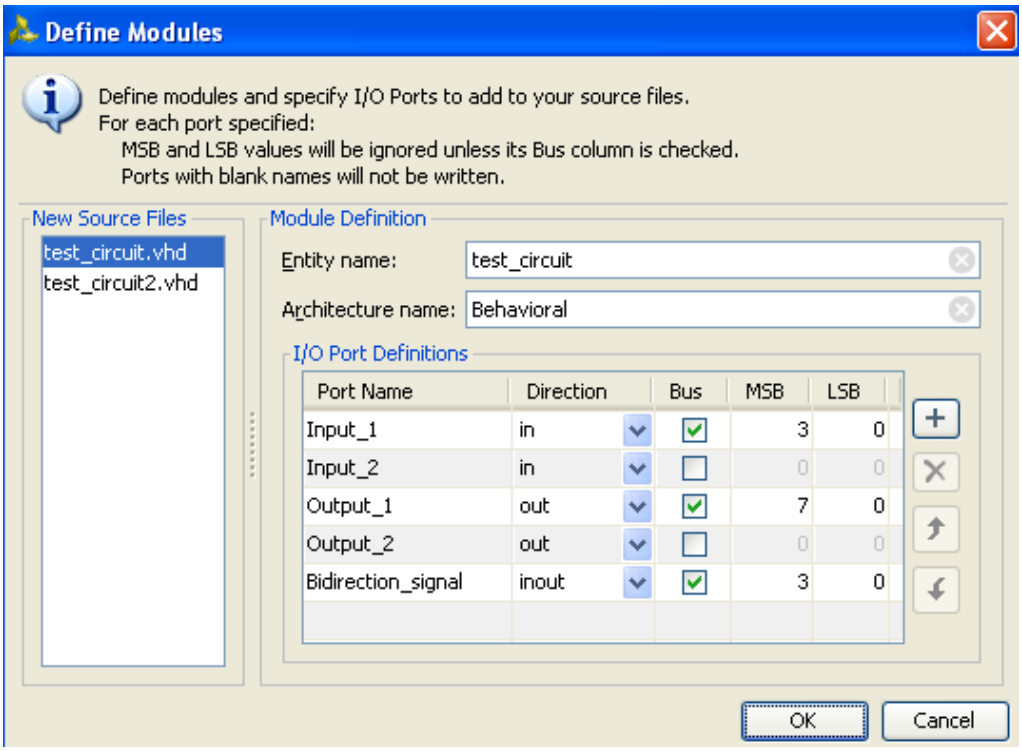
3. Αν θέλουμε να δημιουργήσουμε καινούρια αρχεία, πατώντας το **Create File…** εμφανίζεται το ακόλουθο παράθυρο:



Στο **File type** εισάγουμε **VHDL**, και στο **File name** εισάγουμε το όνομα που θέλουμε να έχει το αρχείο μας.

4. Πατάμε **OK** και το αρχείο δημιουργείται. Παρατηρούμε ότι έχει ενσωματωθεί στη λίστα με τα αρχεία που θέλουμε να προστεθούν στο project. Με αυτόν τον τρόπο, μπορούμε να εισάγουμε ή να δημιουργήσουμε όσα αρχεία θέλουμε να έχει ο σχεδιασμός μας.

5. Ακολούθως, πατάμε **Finish**, και σε περίπτωση που έχουμε δημιουργήσει νέα αρχεία εμφανίζεται ένα παράθυρο για να ορίσουμε τα modules που θα περιγράψουμε, καθώς και τα σήματα εισόδου και εξόδου τους:

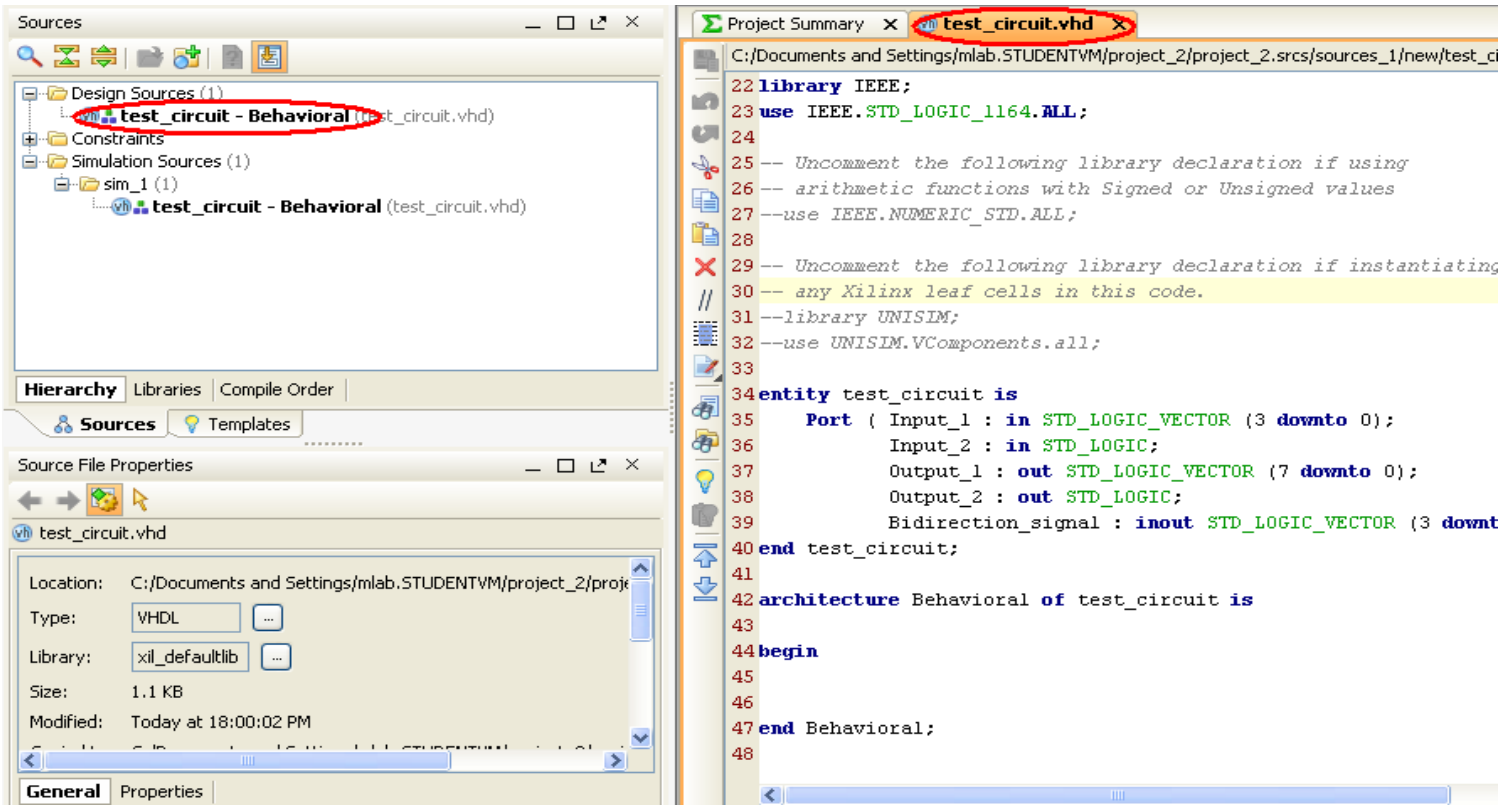


Στο πλαίσιο **New Source Files** επιλέγουμε το αρχείο του οποίου το module θέλουμε να ορίσουμε.
Στο **Entity name** εισάγουμε το όνομα που θέλουμε να έχει το module (καλό είναι να είναι το ίδιο με το όνομα του αρχείου), ενώ στο **Architecture name** εισάγουμε όνομα που να υποδηλώνει τον τρόπο με τον οποίο θα γράψουμε τον κώδικα (πχ. Behavioral, Structural, Data_flow, Gate_level, Mixed, ή οτιδήποτε διευκολύνει το χρήστη να υποδηλώσει την αρχιτεκτονική που θα χρησιμοποιήσει για να περιγράψει το module).
Στο **I/O Port Definitions** δηλώνουμε τα σήματα εισόδου και εξόδου του module. Συγκεκριμένα, στο **Port Name** εισάγουμε τα ονόματα των σημάτων, και στο **Direction** ορίζουμε αν αποτελεί σήμα εισόδου, εξόδου ή εισόδου-εξόδου (διπλής κατεύθυνσης). Επίσης, τικάρουμε το **Bus** σε περίπτωση που θέλουμε το σήμα να αποτελείται από περισσότερα από 1 bit (αρτηρία), ενώ στις 2 επόμενες στήλες (**MSB** και **LSB**) ορίζουμε το μήκος bit του σήματος, εισάγοντας το βάρος του MSB και του LSB (πχ. Για MSB=3 και LSB=0 δημιουργείται ένα 4-bit σήμα).

6. Αφού ορίσουμε το module που περιγράφει κάθε νέο αρχείο που δημιουργήσαμε, πατάμε **OK**.

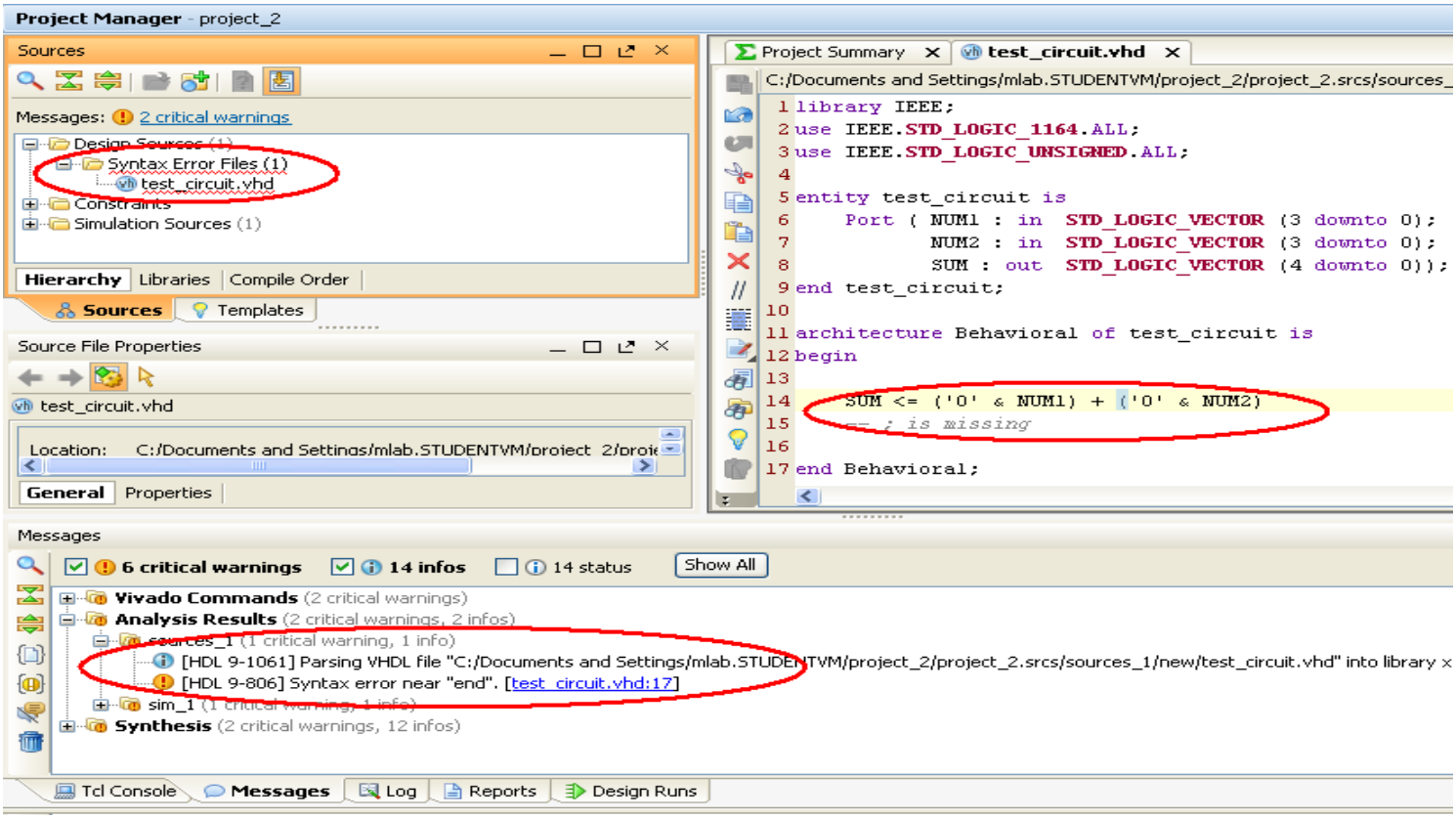
ΣΗΜΕΙΩΣΗ: Μία καλή πρακτική είναι κάθε VHDL αρχείο να περιγράφει μόνο ένα module, από το οποίο και θα παίρνει το όνομα του. Για παράδειγμα, το αρχείο <module1.vhd> να περιγράφει το module με entity name <module1>, το αρχείο <module2.vhd> να περιγράφει το module με entity name <module2>, κ.ο.κ.

Αφού έχουμε εισάγει τα αρχεία του σχεδιασμού μας, παρατηρούμε ότι στο πλαίσιο **Sources** και συγκεκριμένα στο **Design Sources**, έχουν προστεθεί τα modules που δημιουργήσαμε. Πατώντας διπλό κλικ σε κάθε module, ανοίγει το αρχείο .vhd που το περιγράφει. Στο αρχείο αυτό μπορούμε να εισάγουμε τον κώδικα που να περιγράφει το συγκεκριμένο module. Αφού περιγράψουμε το module, αποθηκεύουμε τις αλλαγές που κάναμε πατώντας **Ctrl+S**. Στην παρακάτω εικόνα, απεικονίζονται όσα αναφέραμε:



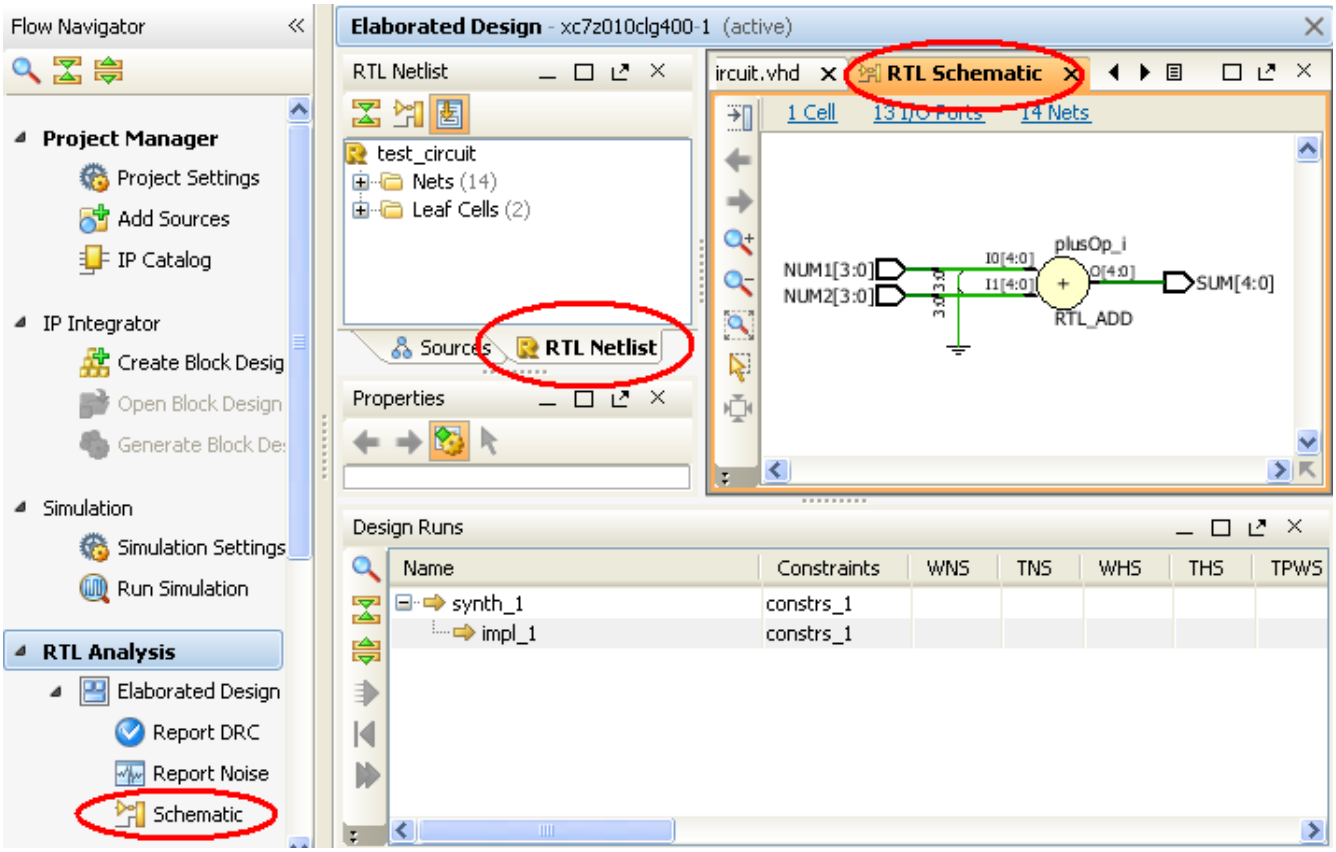
Να σημειώσουμε, ότι αν υπάρχουν πολλά modules στον σχεδιασμό, εμφανίζεται πρώτο το κορυφαίο σε ιεραρχία. Εάν θέλουμε να ορίσουμε ως κορυφαίο ένα άλλο module, πατάμε δεξί κλικ στο αρχείο περιγραφής του, και επιλέγουμε **Set as top**.

ΣΗΜΕΙΩΣΗ: Αν υπάρχουν συντακτικά λάθη στους κώδικες, το εργαλείο θα τα εντοπίσει αυτόματα μόλις πατήσουμε αποθήκευση, και θα μας ενημερώσει για αυτά, όπως φαίνεται στην παρακάτω εικόνα:



Αφού έχουμε εισάγει τα HDL αρχεία, μπορούμε να δούμε το κύκλωμα που έχουμε περιγράψει σε επίπεδο RTL. Το RTL σχηματικό είναι μια αφαιρετική αναπαράσταση, η οποία παρουσιάζει το κύκλωμα που έχουμε περιγράψει αναπαριστώντας τη ροή των σημάτων μεταξύ των καταχωρητών και τον αριθμητικών/λογικών μονάδων.

Πατώντας **Schematic** (βρίσκεται στο πλαίσιο **Flow Navigator > RTL Analysis > Elaborated Design**), το σχηματικό του κυκλώματος μας εμφανίζεται ακριβώς δίπλα, όπως φαίνεται στην ακόλουθη εικόνα:

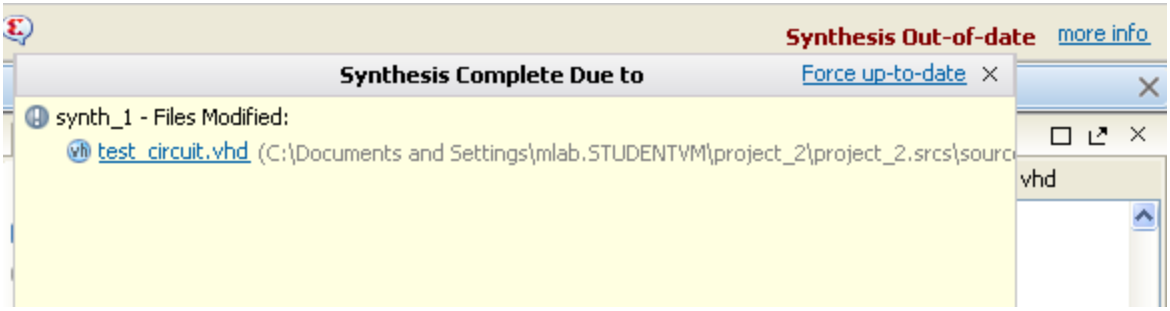


4. Σύνθεση του Σχεδιασμού

Αφού έχουμε δημιουργήσει τα αρχεία που περιγράφουν τον σχεδιασμό μας, θα πρέπει να πραγματοποιήσουμε τη σύνθεση (*synthesis*) του, η οποία είναι μία διαδικασία που παράγει τη λίστα διασυνδέσεων (*netlist*) του κυκλώματος σε επίπεδο λογικών πυλών. Για να πραγματοποιήσουμε τη σύνθεση, αρκεί να πατήσουμε το **Run Synthesis** (βρίσκεται στο πλαίσιο **Flow Navigator > Synthesis**) και το εργαλείο θα αρχίσει να την εκτελεί.
Το εργαλείο θα μας ενημερώσει με κατάλληλο μήνυμα για την επιτυχή ολοκλήρωση της σύνθεσης, και θα μας ρωτήσει ποιο βήμα θέλουμε να πραγματοποιήσουμε στη συνέχεια.

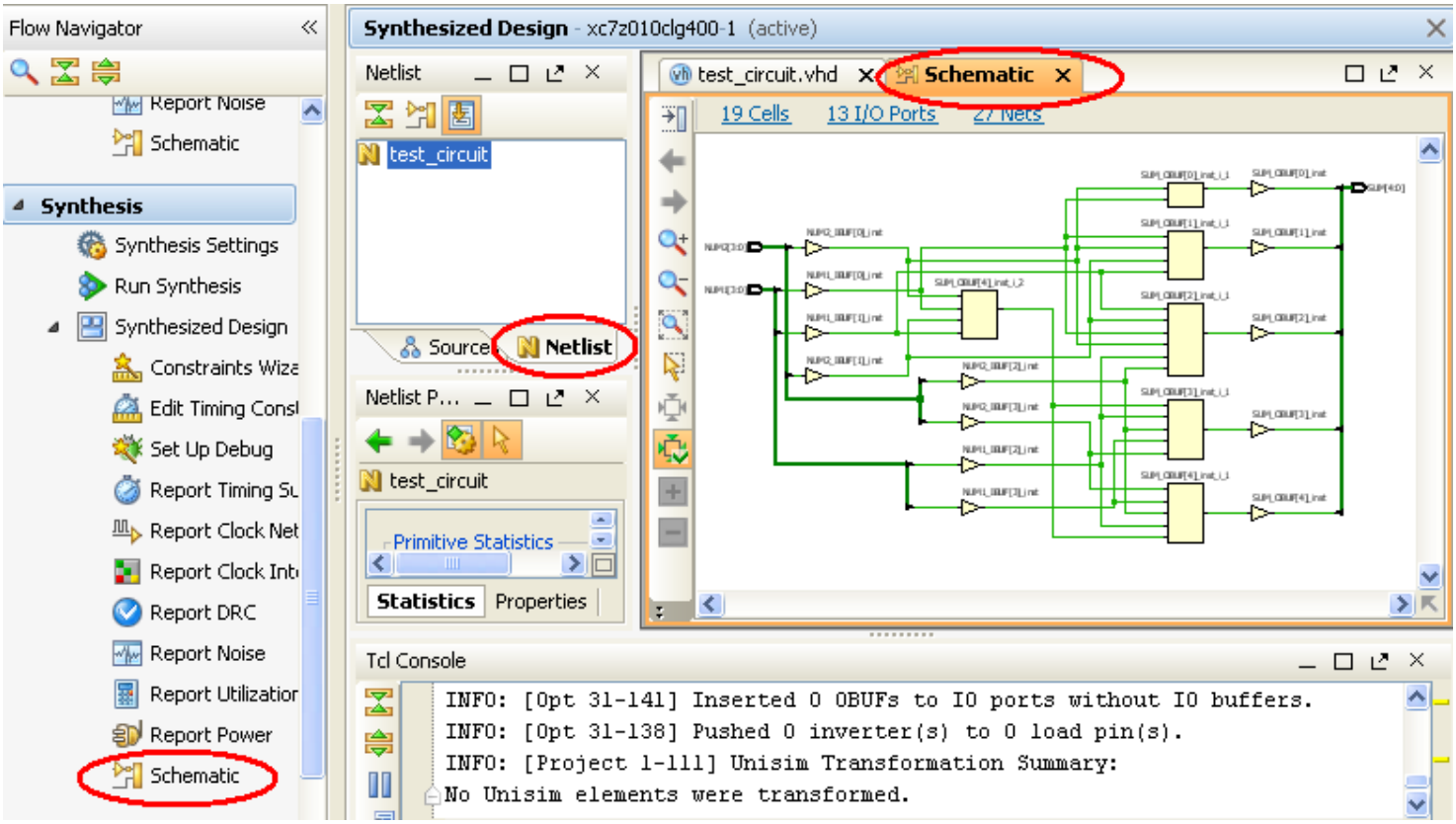
ΣΗΜΕΙΩΣΗ 1: Για να πραγματοποιηθεί η σύνθεση, θα πρέπει να μην υπάρχουν λάθη στα αρχεία του σχεδιασμού μας.

ΣΗΜΕΙΩΣΗ 2: Αν πραγματοποιήσουμε τη σύνθεση, και μετά κάνουμε κάποια αλλαγή στα VHDL αρχεία ή προσθέσουμε καινούρια, τότε το εργαλείο μας ενημερώνει ότι η σύνθεση που έχει πραγματοποιηθεί αφορά τα αρχεία πριν τις αλλαγές που κάναμε, δεν είναι δηλαδή up-to-date, όπως φαίνεται στην παρακάτω εικόνα:



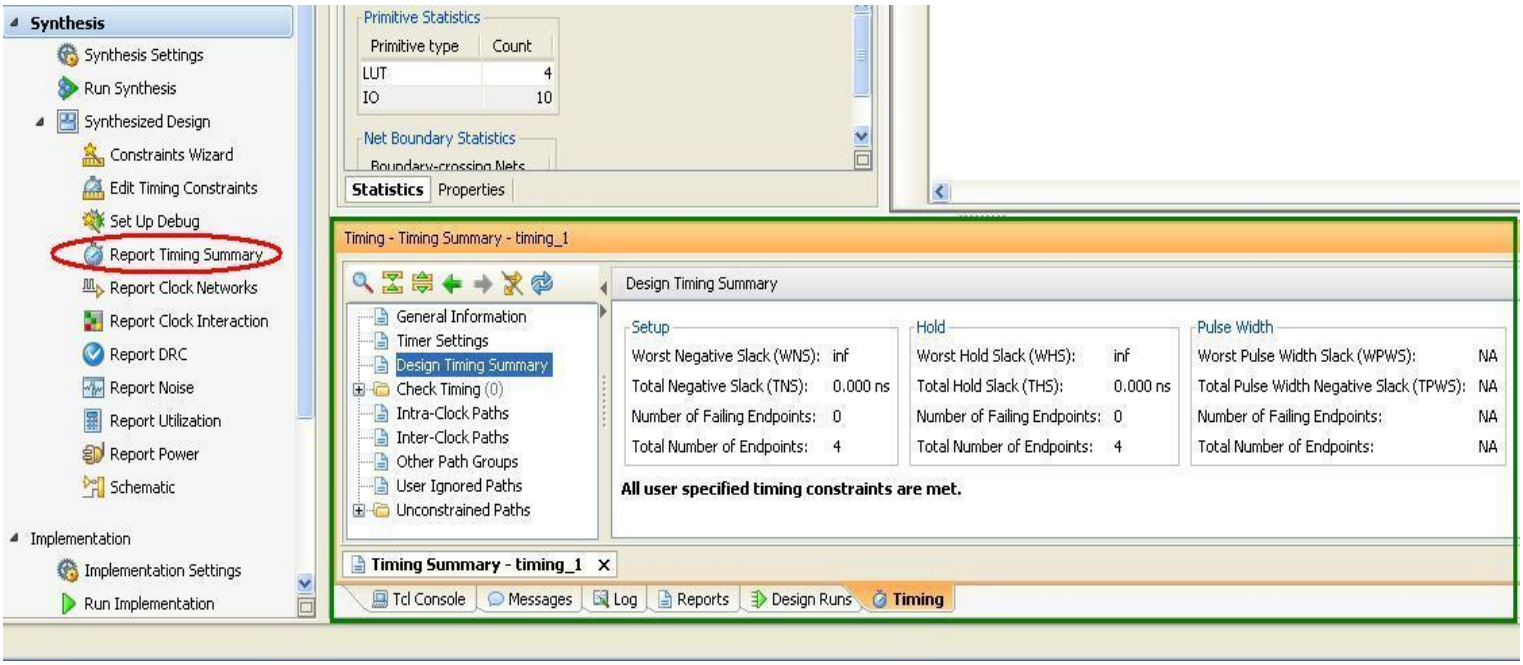
Πατώντας **more info**, εμφανίζεται ένα πλαίσιο που μας ενημερώνει για του που έχουν γίνει αλλαγές.

Αφού έχουμε ολοκληρώσει τη διαδικασία της σύνθεσης, μπορούμε να δούμε το κύκλωμα που παράχθηκε λαμβάνοντας υπόψη τους πόρους του FPGA.
Πατώντας **Schematic** (βρίσκεται στο πλαίσιο **Flow Navigator > Synthesis > Synthesized Design**), το σχηματικό της σύνθεσης εμφανίζεται ακριβώς δίπλα, όπως φαίνεται στην ακόλουθη εικόνα:



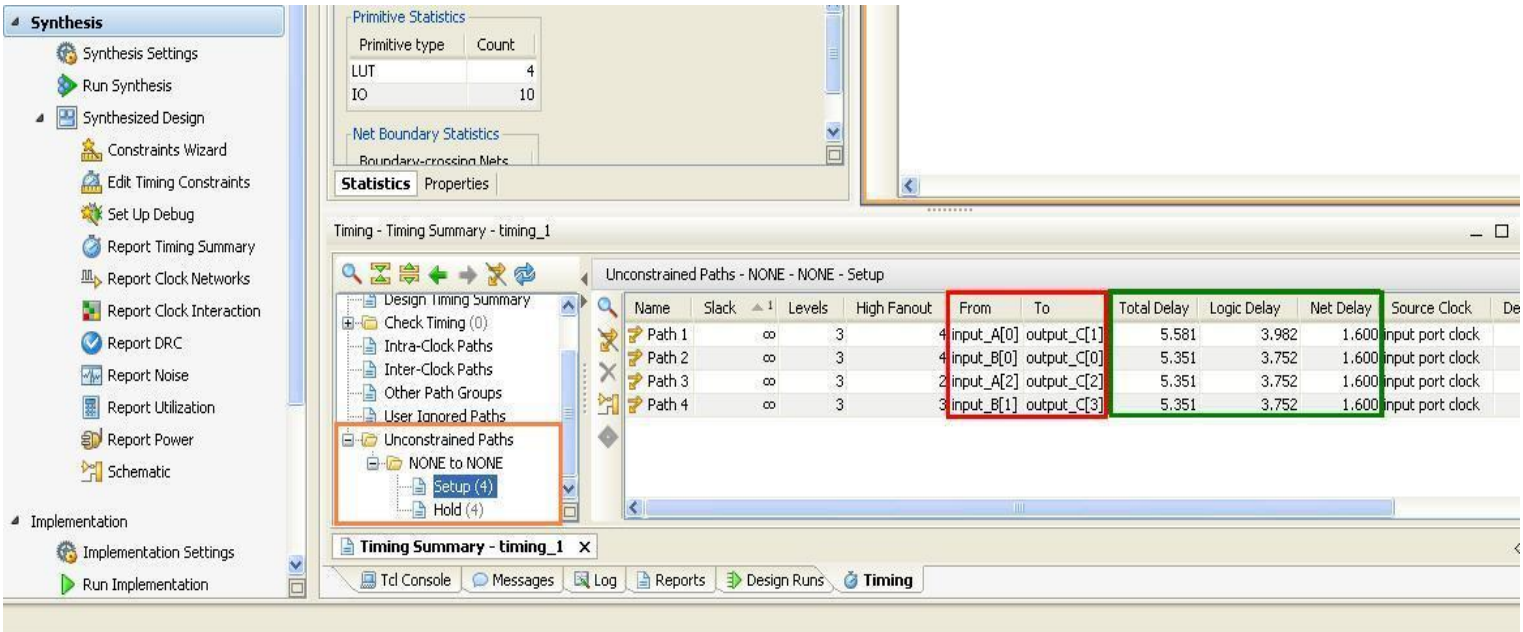
5. Εύρεση Κρίσιμου Μονοπατιού (Critical Path) του Σχεδιασμού

Στο σημείο αυτό υπάρχει η δυνατότητα να έχουμε μια εκτίμηση για τα μονοπάτια που είναι τα πιο αργά σε ένα κύκλωμα. Συγκεκριμένα αφού έχει ολοκληρωθεί η σύνθεση και έχουμε ανοίξει το σχηματικό, πατώντας **Report Timing Summary** (βρίσκεται στο πλαίσιο **Flow Navigator > Synthesis > Synthesized Design**), όπως φαίνεται στην παρακάτω εικόνα (κόκκινη έλλειψη):



Στο παράθυρο που θα ανοίξει αφήνουμε όλες τις προεπιλογές όπως είναι και πατάμε απλώς **OK**. Στο σημείο αυτό στο κάτω μέρος του εργαλείου Vivado θα ανοίξει μία καινούργια καρτέλα με το όνομα **Timing**. Ένα τέτοιο παράδειγμα δίνεται στην προηγούμενη εικόνα (πράσινο πλαίσιο).

Επιλέγουμε **Unconstrained Paths** και ανοίγοντας την ιεραρχία, πατάμε **Setup** (πορτοκαλί πλαίσιο στην εικόνα που ακολουθεί). Αυτή η κίνηση θα εμφανίσει τα πιο αργά μονοπάτια του κυκλώματος.



Η πληροφορία που μας ενδιαφέρει είναι οι στήλες **From** και **To** (κόκκινο πλαίσιο στην παρακάτω εικόνα), οι οποίες μας παρουσιάζουν όλα τα μονοπάτια του κυκλώματος από κάθε είσοδο σε κάθε έξοδο. Επιπλέον, οι στήλες **Total Delay**, **Logic Delay** και **Net Delay** μας δίνουν πληροφορίες σχετικά με την συνολική καθυστέρηση, την καθυστέρηση των πόρων του FPGA και την καθυστέρηση των καλωδίων διασύνδεσης του κυκλώματος για κάθε μονοπάτι. Το μονοπάτι με την μεγαλύτερη συνολική καθυστέρηση (**max Total Delay**) είναι και το κρίσιμο μονοπάτι (πιο αργό) του κυκλώματος.

6. Προσομοίωση του Σχεδιασμού

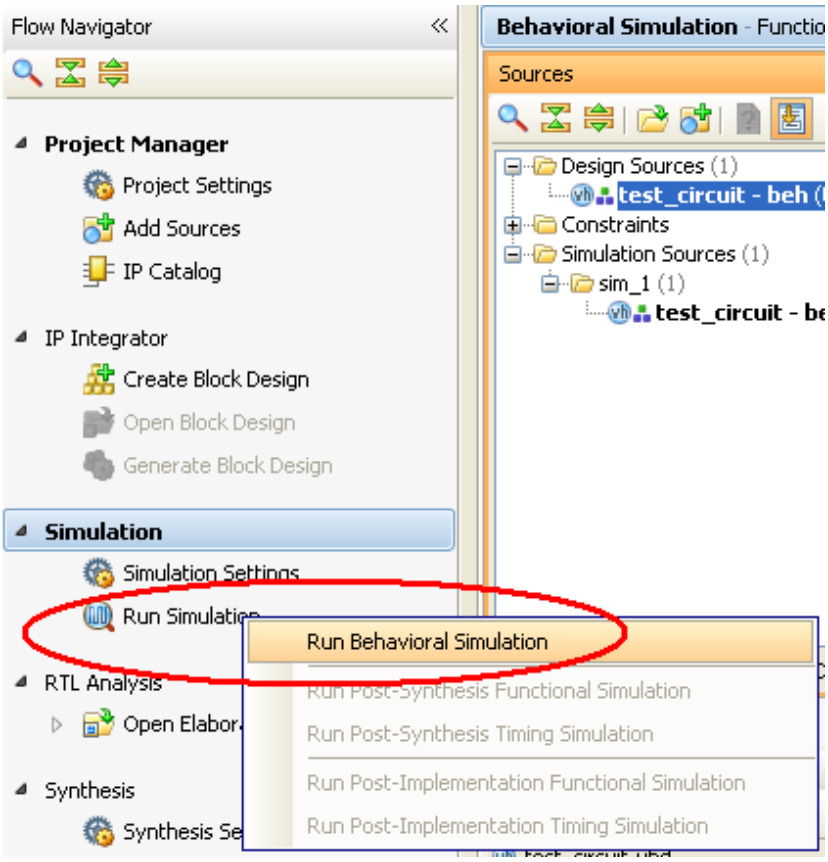
Ο έλεγχος της ορθής λειτουργίας του σχεδιασμού μας βασίζεται σε 2 άξονες: στην επαλήθευση της σωστής λειτουργίας του, και στη σωστή σχεδίαση του σε VHDL κώδικα. Στη γενική περίπτωση δημιουργείται ένα κατάλληλο αρχείο VHDL (testbench), το οποίο τροφοδοτεί με τιμές τις εισόδους του κυκλώματος, και μέσω προσομοιωτή παρατηρούνται οι τιμές των σημάτων που παράγονται στις εξόδους του κυκλώματος.
Η προσομοίωση μπορεί να πραγματοποιηθεί σε 3 επίπεδα:

- Στο **RTL** → ***Behavioral Προσομοίωση***: πραγματοποιείται προσομοίωση του σχεδιασμού που περιγράφεται από τα VHDL αρχεία (RTL netlist). Η προσομοίωση αυτή δεν επηρεάζεται από το FPGA και είναι τελείως ανεξάρτητη από αυτό.
- Μετά τη **Σύνθεση** → ***Post-Synthesis Προσομοίωση***: αποτελεί την προσομοίωση του σχεδιασμού που προέκυψε από τη σύνθεση των VHDL αρχείων (synthesis netlist).
- Μετά την **Υλοποίηση** → ***Post-Implementation Προσομοίωση***: αποτελεί την προσομοίωση του σχεδιασμού που προέκυψε από την υλοποίηση της λίστας διασυνδέσεων που προέκυψε από τη σύνθεση (implementation netlist). Λαμβάνεται υπόψη η αρχιτεκτονική του FPGA, χρονικές πληροφορίες, κτλ.

Στα πλαίσια του Εργαστηρίου θα ασχοληθούμε μόνο με την ***Behavioral Προσομοίωση***. Σκοπός της είναι να ελέγξουμε την ορθή λειτουργία του σχεδιασμού που έχουμε περιγράψει σε VHDL.
Στη συνέχεια θα παρουσιαστεί πρώτα ένας εναλλακτικός τρόπος προσομοίωσης, ο οποίος μπορεί να εφαρμοστεί σχετικά γρήγορα για κυκλώματα μικρής πολυπλοκότητας, και μετά η προσομοίωση του σχεδιασμού με χρήση testbench.

A. Χωρίς τη χρήση Testbench

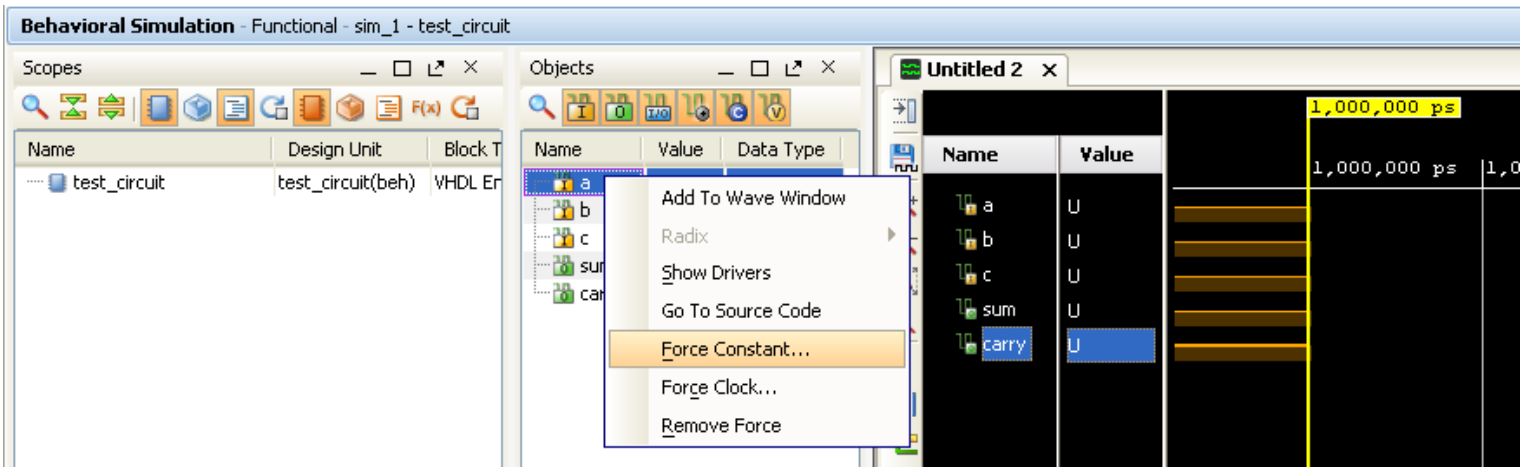
Επιλέγουμε **Run Simulation** (βρίσκεται στο πλαίσιο **Flow Navigator > Simulation**), και μετά πατάμε **Run Behavioral Simulation**, όπως φαίνεται στην ακόλουθη εικόνα:



Παρατηρούμε ότι οι επιλογές για Post-Synthesis και Post-Implementation προσομοιώσεις είναι ανενεργές, καθώς δεν έχουν εκτελεστεί ακόμα οι αντίστοιχες διαδικασίες.

Με το πάτημα του **Run Behavioral Simulation**, ανοίγει το περιβάλλον προσομοίωσης, καθώς και οι κυματομορφές των σημάτων εισόδου και εξόδου.

Στο σημείο αυτό τα σήματα εισόδου έχουν απροσδιόριστη τιμή (πορτοκαλί χρώμα) αφού δεν έχει εφαρμοστεί κάποια τιμή σε αυτά, όπως και τα σήματα εξόδου. Για να εφαρμοστεί μία τιμή σε ένα σήμα εισόδου, επιλέγουμε το σήμα από το πλαίσιο **Objects**, πατάμε δεξί κλικ, και εμφανίζεται μία λίστα επιλογών, όπως φαίνεται στην ακόλουθη εικόνα:

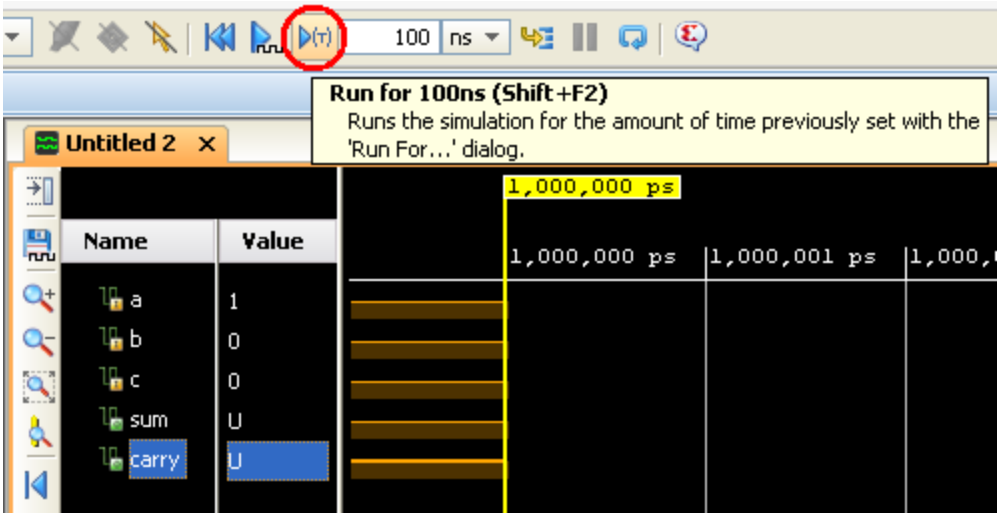


Σε αυτό το σημείο, οι επιλογές είναι δυο:

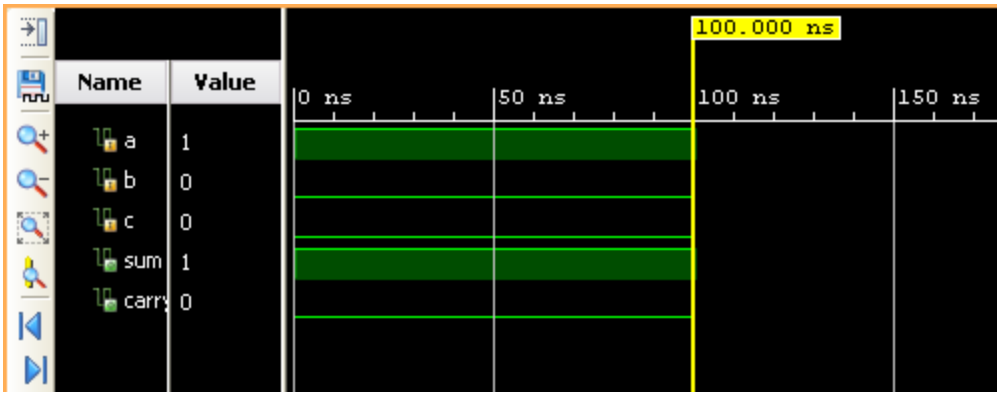
- Εφαρμογή σταθερής τιμής, πατώντας **Force Constant…**, και στη συνέχεια ορίζοντας την τιμή που θέλουμε να έχει το σήμα μας.
- Εφαρμογή εναλλαγής τιμών με κάποια περιοδικότητα, πατώντας **Force Clock…**, και στη συνέχεια ορίζοντας το κάθε πότε θέλουμε να αλλάζει τιμή το σήμα μας.

Αφού εφαρμόσουμε τιμές σε όλα τα σήματα εισόδου, είμαστε έτοιμοι να ξεκινήσουμε την προσομοίωση!

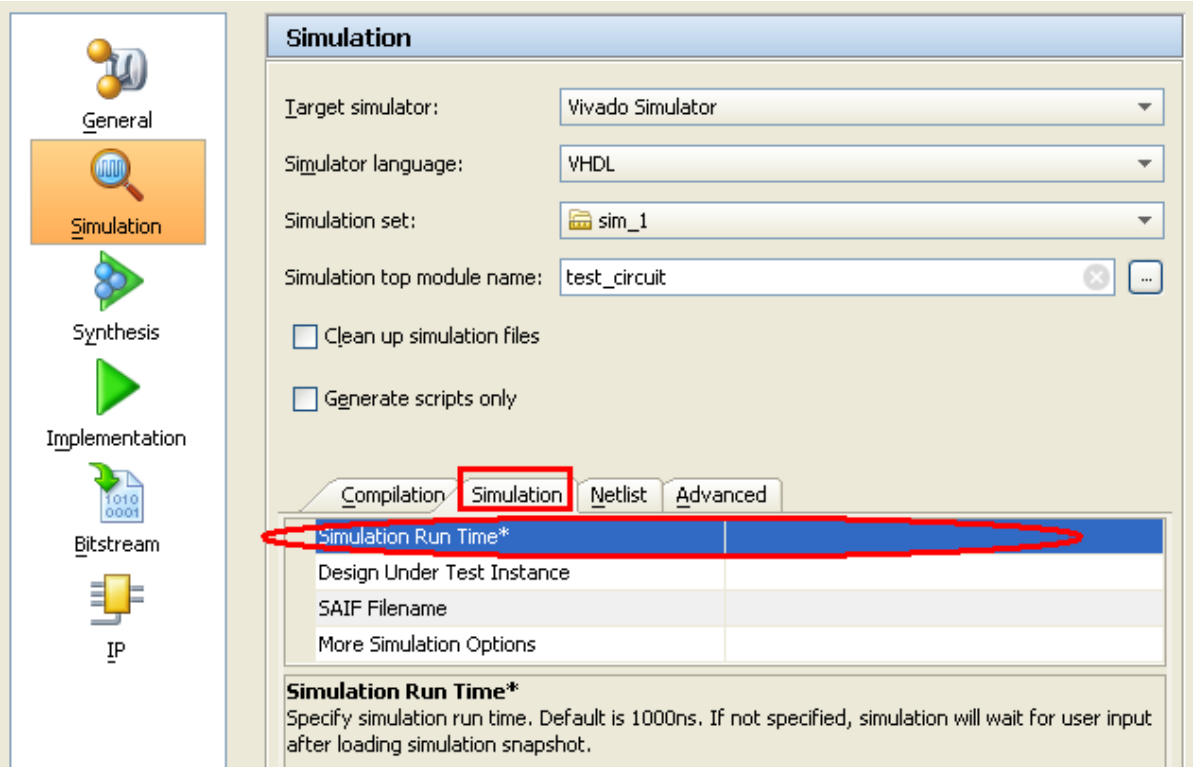
Η προσομοίωση αρχίζει μέσω των διαθέσιμων κουμπιών που βρίσκονται πάνω από τις κυματομορφές, όπως φαίνεται στην παρακάτω εικόνα:



Μέσω της επιλογής που έχει σημειωθεί με κόκκινο, επιλέγουμε να τρέξουμε την προσομοίωση για χρόνο ίσο με αυτόν που δηλώνουμε στις δύο διπλανές φόρμες (στο συγκεκριμένο παράδειγμα 100ns). Το αποτέλεσμα της συγκεκριμένης προσομοίωσης παρουσιάζεται στην ακόλουθη εικόνα:



ΣΗΜΕΙΩΣΗ 1: θα πρέπει να σημειώσουμε ότι πριν τρέξουμε την παραπάνω προσομοίωση, είχαμε απενεργοποιήσει την default τιμή εκτέλεσης της προσομοίωσης (1000 ns), η οποία ενεργοποιείται με το άνοιγμα του περιβάλλοντος προσομοίωσης. Αυτό το κάναμε μέσω της επιλογής **Simulation Settings** (βρίσκεται στο πλαίσιο **Flow Navigator > Simulation**), η οποία εμφανίζει το παράθυρο της ακόλουθης εικόνας, όπου υποδεικνύεται η απενεργοποίηση της λειτουργίας:



ΣΗΜΕΙΩΣΗ 2: επίσης αλλάξαμε από ps σε ns τη μονάδα αναπαράστασης του χρόνου των κυματομορφών, πατώντας δεξί κλικ πάνω στο σημείο που υποδεικνύεται από την παρακάτω εικόνα:

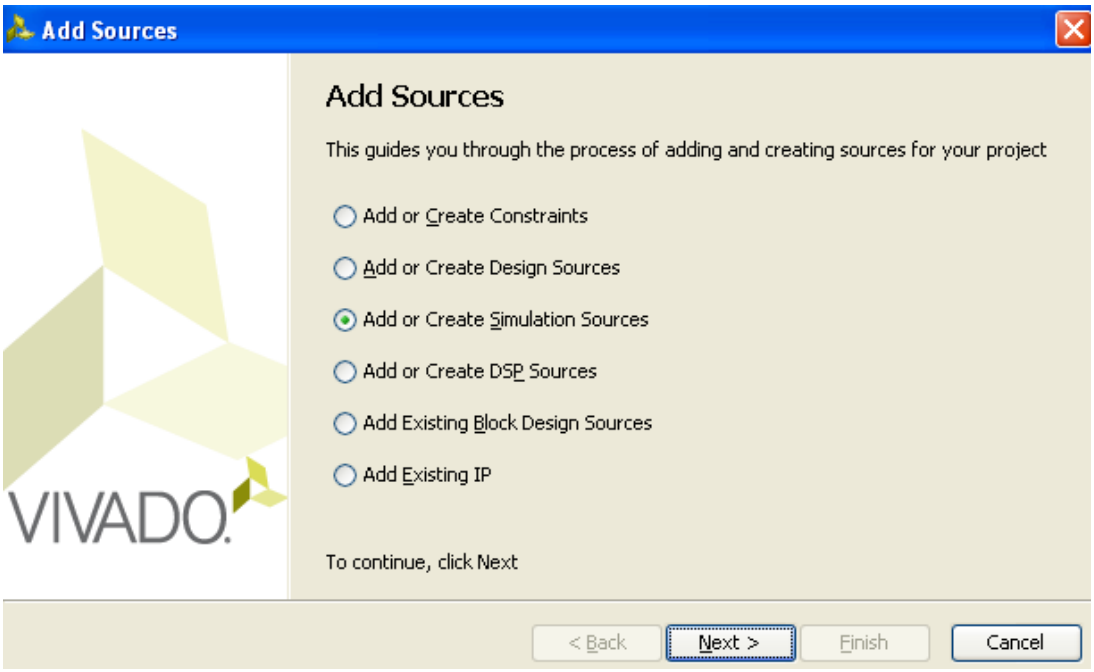
Β. Με τη χρήση Testbench

Το testbench αποτελεί μία περιγραφή σε VHDL, η οποία χρησιμοποιείται για την προσομοίωση του σχεδιασμού, την λειτουργικότητα του οποίου θέλουμε να ελέγξουμε (υπό δοκιμή μονάδα, *Unit Under Test* - UUT). Ένα testbench περιλαμβάνει:

- Ένα τουλάχιστον στιγμιότυπο της υπό δοκιμής μονάδας (UUT *instantiation*)
- Τις κυματομορφές που εφαρμόζονται στις εισόδους της υπό δοκιμής μονάδας

Δηλαδή, στο testbench ορίζουμε τις τιμές των διανυσμάτων εισόδου του σχεδιασμού μας, και το πως αυτές αλλάζουν στο χρόνο, και τις τροφοδοτούμε στο στιγμιότυπο που έχουμε δημιουργήσει.
Για να δημιουργήσουμε ένα testbench, ακολουθούμε τα παρακάτω βήματα:

1. Στο περιβάλλον **Flow Navigator**, πατάμε **Project Manager > Add Sources** και εμφανίζεται το ακόλουθο παράθυρο:



   Επιλέγουμε **Add or Create Simulation Sources**.

2. Πατάμε **Next** και εμφανίζεται ένα παράθυρο για να εισάγουμε (**Add Files...**) ή να δημιουργήσουμε (**Create File...**) VHDL αρχεία.

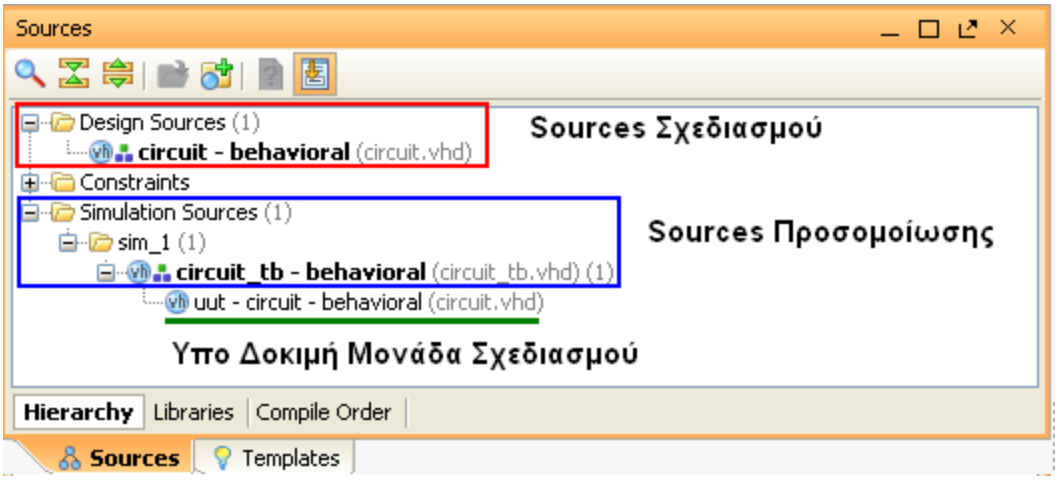
3. Ακολουθούμε την ίδια ακριβώς διαδικασία που περιγράψαμε στην ενότητα για την Εισαγωγή/Δημιουργία VHDL αρχείων.

ΣΗΜΕΙΩΣΗ_1: Μία καλή πρακτική είναι να δίνουμε όνομα στο αρχείο του testbench που να υποδηλώνει την υπο δοκιμή μονάδα. Αν για παράδειγμα θέλουμε να προσομοιώσουμε το <module1> (που βρίσκεται στο αρχείο <module1>.vhd), δίνουμε στο αρχείο testbench το όνομα <module1>_tb.vhd, και στο entity name του testbench το όνομα <module1>_tb.

ΣΗΜΕΙΩΣΗ_2: Όταν θέλουμε να προσομοιώσουμε ολόκληρο το σχεδιασμό μας, τότε δημιουργούμε testbench για το κορυφαίο σε ιεραρχία module.

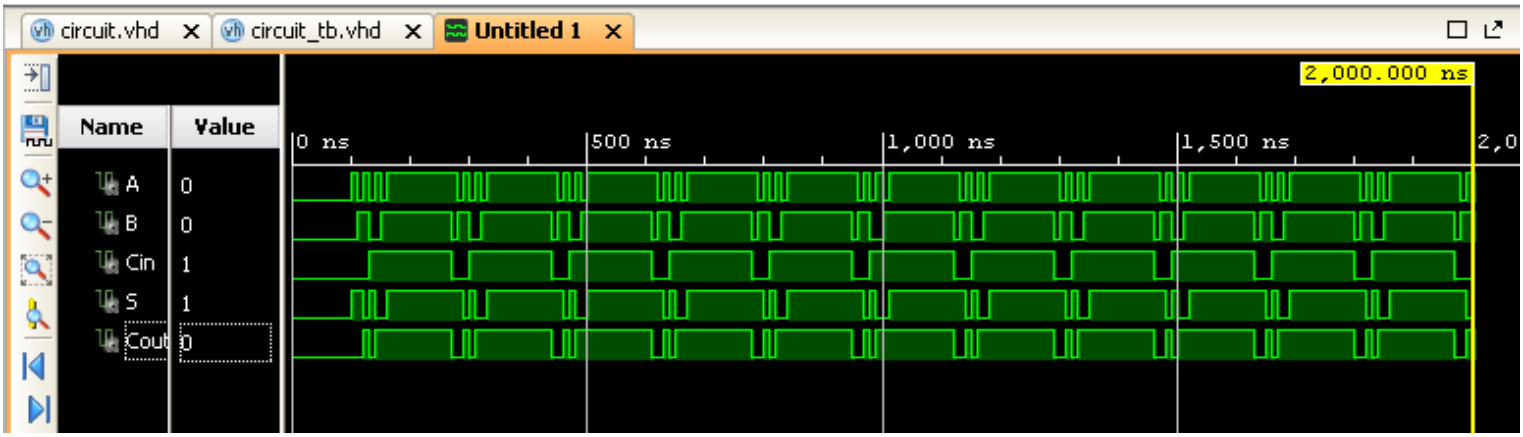
ΣΗΜΕΙΩΣΗ_3: Μπορούμε να δημιουργήσουμε testbenches και να ελέγξουμε την λειτουργία των χαμηλότερων σε ιεραρχία modules του σχεδιασμού μας.

Αφού έχουμε δημιουργήσει το testbench, παρατηρούμε ότι αυτό έχει προστεθεί στο πλαίσιο **Sources** και συγκεκριμένα στο **Simulation Sources**, όπως φαίνεται στην παρακάτω εικόνα:



Πατώντας διπλό κλικ στο module του testbench, μπορούμε να εισάγουμε τον κώδικα περιγραφής του.

Για να ξεκινήσουμε την προσομοίωση, επιλέγουμε **Run Simulation** (βρίσκεται στο πλαίσιο **Flow Navigator > Simulation**), και μετά πατάμε **Run Behavioral Simulation**.
Το testbench που έχουμε δημιουργήσει θα δώσει τιμές στις εισόδους της υπό δοκιμής μονάδας. Ξεκινάμε, λοιπόν, την προσομοίωση όπως και πριν (χρησιμοποιώντας τα μπλε κουμπιά για την εκκίνηση της προσομοίωσης), και αρχίζουν και παράγονται κυματομορφές όπως φαίνεται στην ακόλουθη εικόνα:



7. Κατανάλωση Πόρων του FPGA

Μία από τις σημαντικότερες παραμέτρους στην σχεδίαση κυκλωμάτων σε FPGA είναι ο αριθμός των βασικών δομικών του μπλοκ που θα χρησιμοποιηθούν προκειμένου να υλοποιηθεί η ζητούμενη λειτουργία του κυκλώματος. Εκτός από τον απόλυτο αριθμό, μας ενδιαφέρει και το ποσοστό χρήσης των διαθέσιμων πόρων.

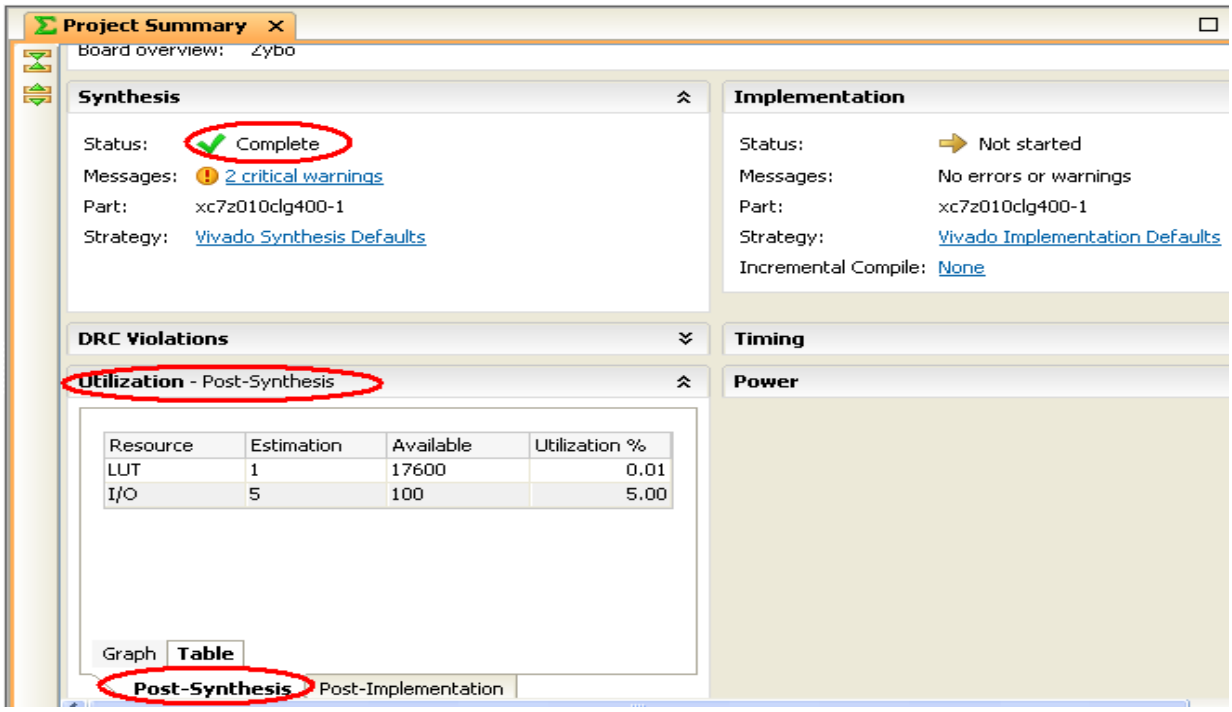
ΣΗΜΕΙΩΣΗ 1: Η χρήση των πόρων του FPGA σχετίζεται σε μεγάλο βαθμό με τον τρόπο με τον οποίο έχουμε περιγράψει τον σχεδιασμό μας (έχουμε γράψει δηλαδή τον HDL κώδικα). Για παράδειγμα, μία μη αποδοτική περιγραφή μπορεί να έχει ως αποτέλεσμα να ξεπεραστούν οι διαθέσιμοι πόροι (το εργαλείο θα κάνει σύνθεση τον σχεδιασμό, απλά θα μας αναφέρει ότι χρειάζεται περισσότερους πόρους για την υλοποίηση του, με άλλα λόγια ο σχεδιασμός δε θα “χωράει” στο FPGA).

ΣΗΜΕΙΩΣΗ 2: Ο χρήστης μπορεί να επιλέξει μέσω του εργαλείου ποιοι πόροι θέλει να χρησιμοποιηθούν και κάτω από ποιες συνθήκες (προαιρετικά).

Στο σημείο αυτό θα γίνει μια σύντομη περιγραφή των βασικών δομικών μονάδων ενός FPGA:

- ***Lookup Tables (LUTs)***: Σε γενικές γραμμές, ένα LUT είναι ένας πίνακας που καθορίζει ποια είναι η έξοδος του για κάθε δεδομένο συνδυασμό εισόδων. Στα πλαίσια της συνδυαστικής λογικής, είναι ουσιαστικά ο πίνακας αλήθειας μιας λογικής συνάρτησης. Αυτός ο πίνακας αλήθειας καθορίζει το πώς λειτουργεί η συνδυαστική λογική.
- ***Flip Flops (FFs)***: Τα FFs είναι στοιχεία μνήμης (καταχωρητές) που χρησιμοποιούνται για να συγχρονίσουν τη λογική και να αποθηκεύσουν λογικές καταστάσεις μεταξύ των κύκλων ρολογιού. Σε κάθε ακμή του ρολογιού, ένα FF αποθηκεύει την τιμή 1 ή 0 (αληθής ή ψευδής) στην είσοδο του, και κρατάει σταθερή την τιμή αυτή μέχρι την επόμενη ακμή του ρολογιού.
- ***Digital Signal Processors (DSPs)***: Τα DSPs αποτελούν προκατασκευασμένες ενσωματωμένες μονάδες ψηφιακής επεξεργασίας σήματος. Είναι κυκλώματα υψηλής απόδοσης, και μπορούν να χρησιμοποιηθούν για την υλοποίηση αριθμητικών (κυρίως MAC) και λογικών πράξεων. Επειδή ο αριθμός των μονάδων αυτών είναι σημαντικά περιορισμένος σε σχέση με τον αριθμό των LUTs ή των FFs σε ένα FPGA, η χρήση τους είναι ιδιαίτερα σημαντική, αφού η σωστή αξιοποίηση τους μπορεί να αυξήσει σημαντικά τις δυνατότητες ενός συστήματος. Αξίζει να σημειωθεί ότι οι πράξεις που υλοποιούν τα DSPs μπορούν να γίνουν και με χρήση LUTs και FFs, αλλά το τελικό κύκλωμα δεν θα έχει το ίδιο υψηλή απόδοση σε σχέση με το αντίστοιχο που κάνει χρήση των DSPs.
- ***RAM Blocks (RAMBs)***: Οι ενσωματωμένοι πόροι μνήμης (RAMBs) είναι άλλο ένα βασικό δομικό στοιχείο ενός FPGA. Κατανέμονται σε όλη την έκταση του FPGA και είναι χρήσιμες για την αποθήκευση συνόλων δεδομένων ή ενδιάμεσων τιμών μεταξύ παράλληλων λειτουργιών. Όπως και τα DSPs, ο αριθμός των RAMBs σε ένα FPGA είναι περιορισμένος, οπότε το αν θα πρέπει να γίνει χρήση τους είναι θέμα σωστής ανάλυσης της λειτουργίας ενός κυκλώματος.

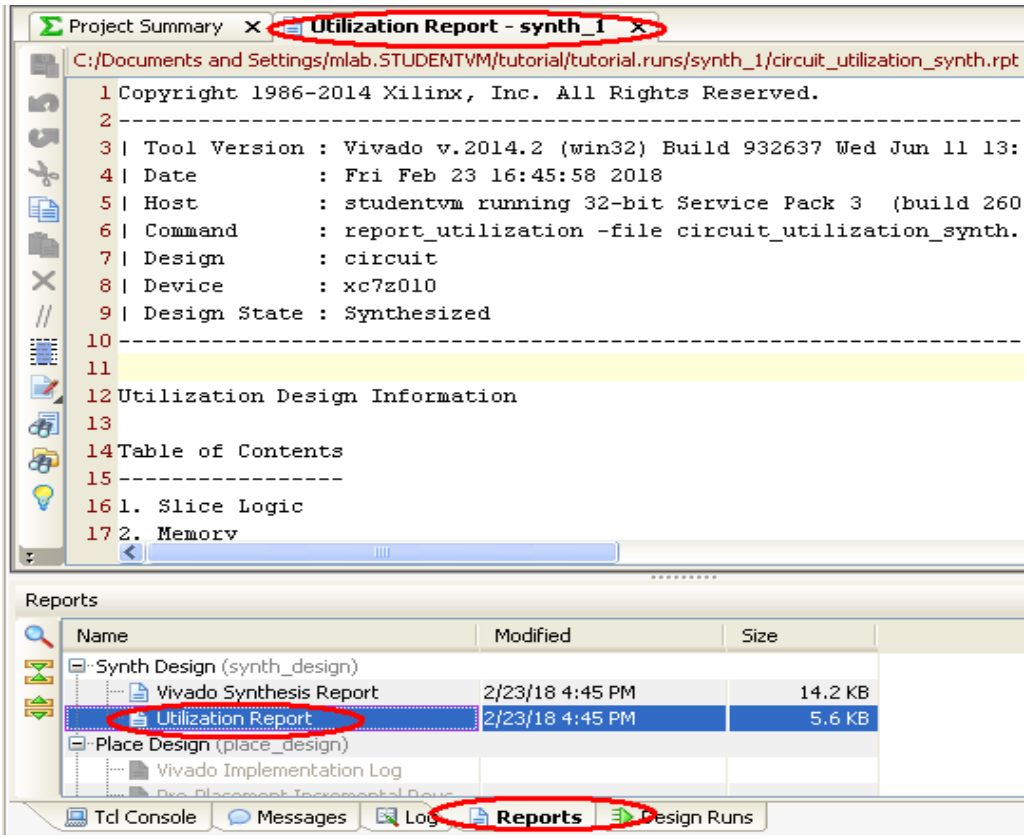
Αφού έχει εκτελεσθεί επιτυχώς η σύνθεση του σχεδιασμού, το εργαλείο δημιουργεί μία αναφορά που περιλαμβάνει μεταξύ άλλων τους πόρους αναμένεται να χρησιμοποιηθούν.
Μπορούμε να δούμε τους πόρους στο πλαίσιο **Project Summary**, όπως φαίνεται στην παρακάτω εικόνα:



Παρατηρούμε ότι η σύνθεση έχει ολοκληρωθεί επιτυχώς και στην καρτέλα **Utilization - Post-Synthesis** αναγράφονται οι πόροι. Το εργαλείο τους παρουσιάζει είτε με τη μορφή πίνακα (**Table**), είτε με τη μορφή γραφικής παράστασης (**Graph**).

ΣΗΜΕΙΩΣΗ: Οι πόροι που παραθέτει το εργαλείο μετά τη σύνθεση αποτελούν μία εκτίμηση του ακριβή αριθμού των πόρων που θα χρησιμοποιηθούν για την υλοποίηση του σχεδιασμού. Για αυτό τον λόγο αναφέρονται ως **Estimation**, όπως φαίνεται και στην παραπάνω εικόνα. Ο πραγματικός αριθμός των πόρων παρατίθεται μόλις ολοκληρωθεί και η διαδικασία της υλοποίησης (*implementation*).

Επίσης, το εργαλείο δημιουργεί μία πιο αναλυτική αναφορά για την σύνθεση. Αυτή είναι προσβάσιμη με τον τρόπο που απεικονίζεται στην παρακάτω εικόνα:



8. Υλοποίηση του Σχεδιασμού

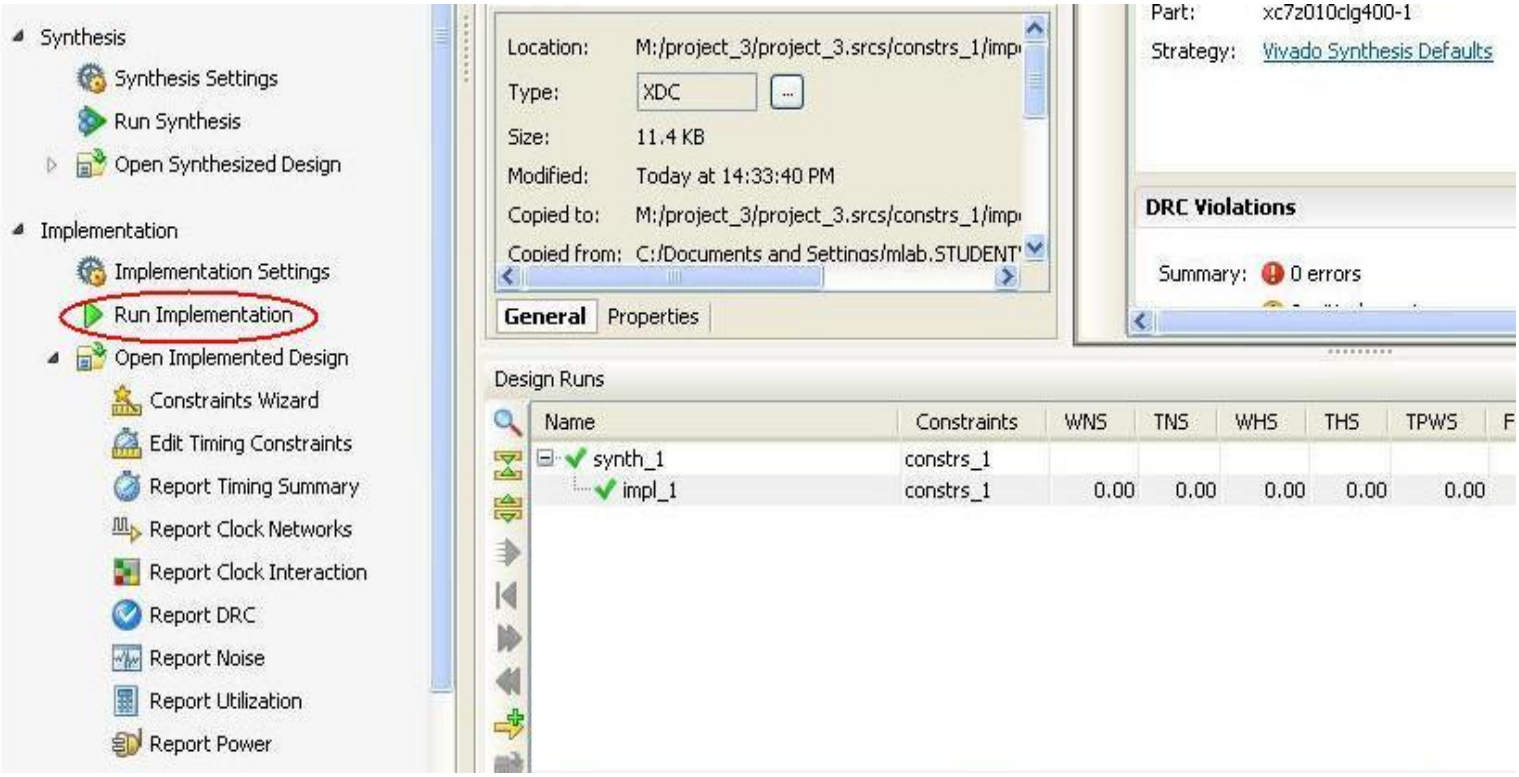
Μετά το στάδιο της σύνθεσης ακολουθεί το στάδιο της υλοποίησης του κυκλώματος (Implementation). Η υλοποίηση του κυκλώματος χωρίζεται σε δύο υποστάδια:

1) Της Τοποθέτησης του σχεδιασμού (**Place Design**). Κατά το στάδιο αυτό δοκιμάζονται διάφορες τοποθετήσεις του κυκλώματος στους διαθέσιμους πόρους του FPGA.

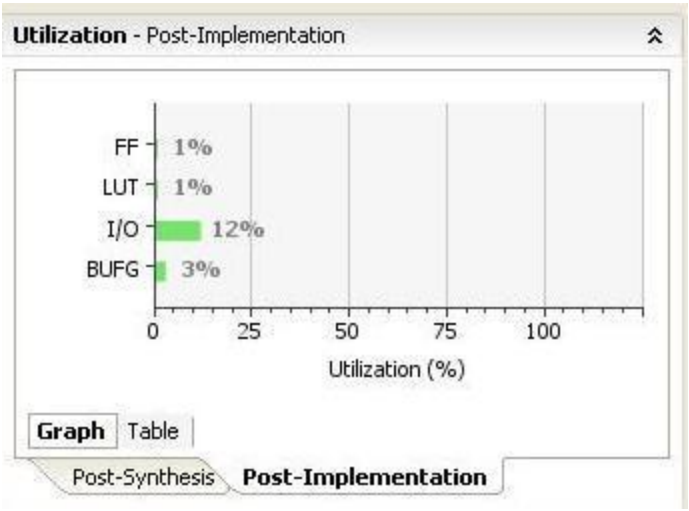
2) Της Δρομολόγησης του σχεδιασμού (**Route Design**). Στο στάδιο αυτό για κάθε πιθανή τοποθέτηση του σχεδιασμού στους πόρους του FPGA δοκιμάζονται διάφορες δρομολογήσεις των διασυνδέσεων μεταξύ των πόρων προκειμένου να υλοποιηθεί η λειτουργία που έχει περιγραφεί.

Οι διαδικασίες **Place & Route** απαιτούν πολλά βήματα καθώς υπάρχουν τοποθετήσεις για τις οποίες δεν μπορεί να επιτευχθεί δρομολόγηση, καθώς επίσης και τοποθετήσεις στις οποίες δεν υπάρχει δρομολόγηση ώστε να πληρούνται οι χρονικοί περιορισμοί. Επίσης λαμβάνονται υπόψη και χρονικοί και χωρικοί περιορισμοί που έχει δώσει ο χρήστης. Γενικά, η τελική κάλυψη του χώρου και η καθυστέρηση κρίσιμου μονοπατιού (**critical path delay**) είναι μεγαλύτερες από τις εκτιμήσεις της σύνθεσης, αλλά και η πιο ακριβείς.

Για να πραγματοποιήσουμε την υλοποίηση, αρκεί να πατήσουμε το **Run Implementation** (βρίσκεται στο πλαίσιο **Flow Navigator > Implementation**) και το εργαλείο θα αρχίσει να την εκτελεί. Το εργαλείο θα μας ενημερώσει με κατάλληλο μήνυμα για την επιτυχή ολοκλήρωση της υλοποίησης.



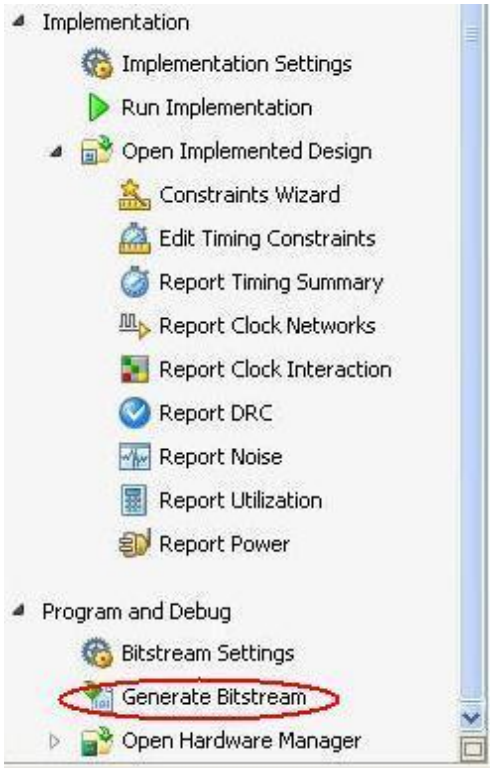
Μετά την επιτυχή ολοκλήρωση της υλοποίησης ενός σχεδιασμού το εργαλείο θα μας δώσει την τελική και πιο ακριβή αναφορά σχετικά με τους πόρους του FPGA που έχουν χρησιμοποιηθεί. Ένα τέτοιο παράδειγμα αναφοράς (**Utilization**) παρουσιάζεται στην επόμενη εικόνα:



Επιπλέον, στο συγκεκριμένο σημείο ο σχεδιαστής είναι σε θέση να δει και την ακριβή συχνότητα λειτουργίας του συστήματος που έχει υλοποιηθεί στην συγκεκριμένη συσκευή που επέλεξε κατά την διάρκεια δημιουργίας του project. Η πληροφορία αυτή εμφανίζεται στην καρτέλα **Project Summary** και συγκεκριμένα στην περιοχή **Timing**. Η πληροφορία που παρέχει το Vivado για την συχνότητα λειτουργίας είναι με βάση την περίοδο του ρολογιού που έχει τεθεί από τον σχεδιαστή όταν δημιουργεί του περιορισμούς (**constraints**) του σχεδιασμού. Συγκεκριμένα, η πληροφορία που ενδιαφέρει είναι η **Worst Negative Slack (WNS)**. Εάν η τιμή που αναφέρει το εργαλείο είναι θετική, σημαίνει πως δίνατε να μειωθεί η περίοδος του ρολογιού κατά αυτήν την ποσότητα. Διαφορετικά αν η τιμή που αναφέρεται είναι αρνητική, τότε θα πρέπει να αυξηθεί η περίοδος κατά αυτήν την ποσότητα.

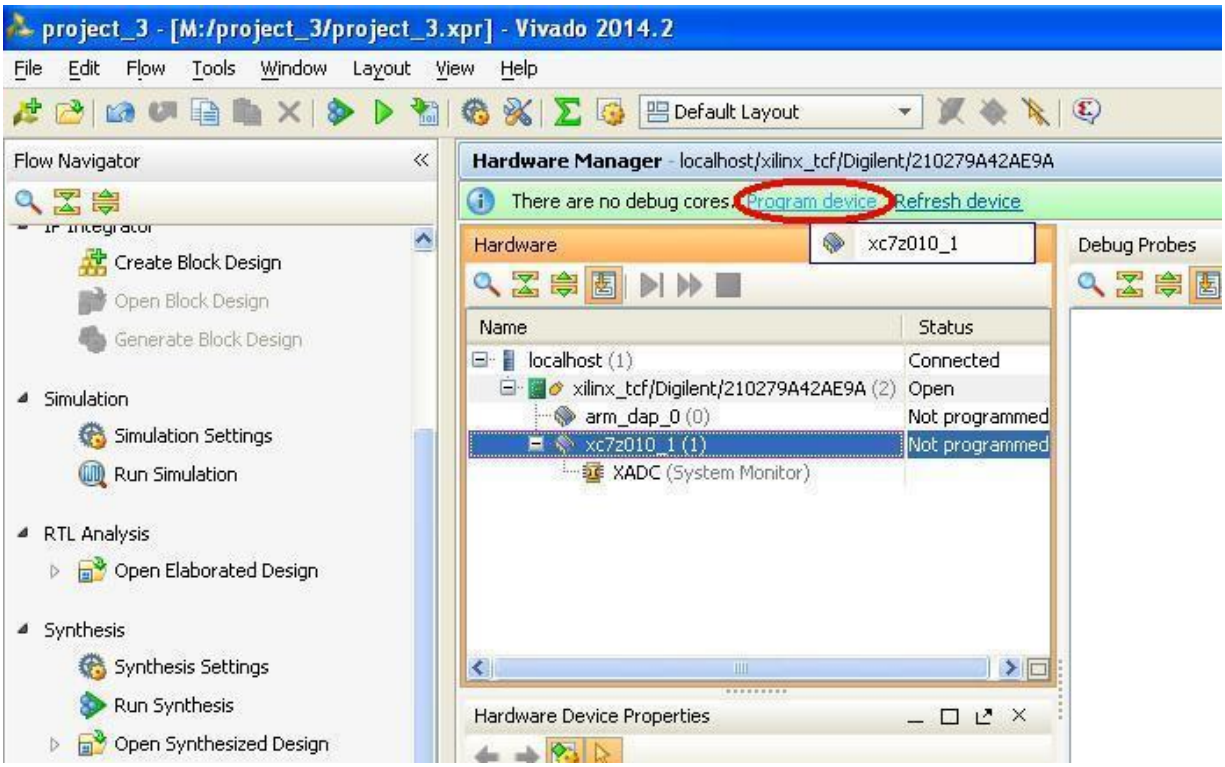
9. Προγραμματισμός του FPGA

Το τελικό στάδιο στη ροή σχεδίασης και υλοποίησης ενός κυκλώματος σε FPGA είναι η παραγωγή του αρχείου προγραμματισμού επιλέγοντας **Generate Bitstream** (βρίσκεται στο πλαίσιο **Flow Navigator > Program and Debug**). Η διαδικασία αυτή απλά δημιουργεί το **bitstream** αρχείο (π.χ. **design.bit**) προγραμματισμού του FPGA για την τελική τοποθέτηση-δρομολόγηση που επιλέχθηκε από το εργαλείο.

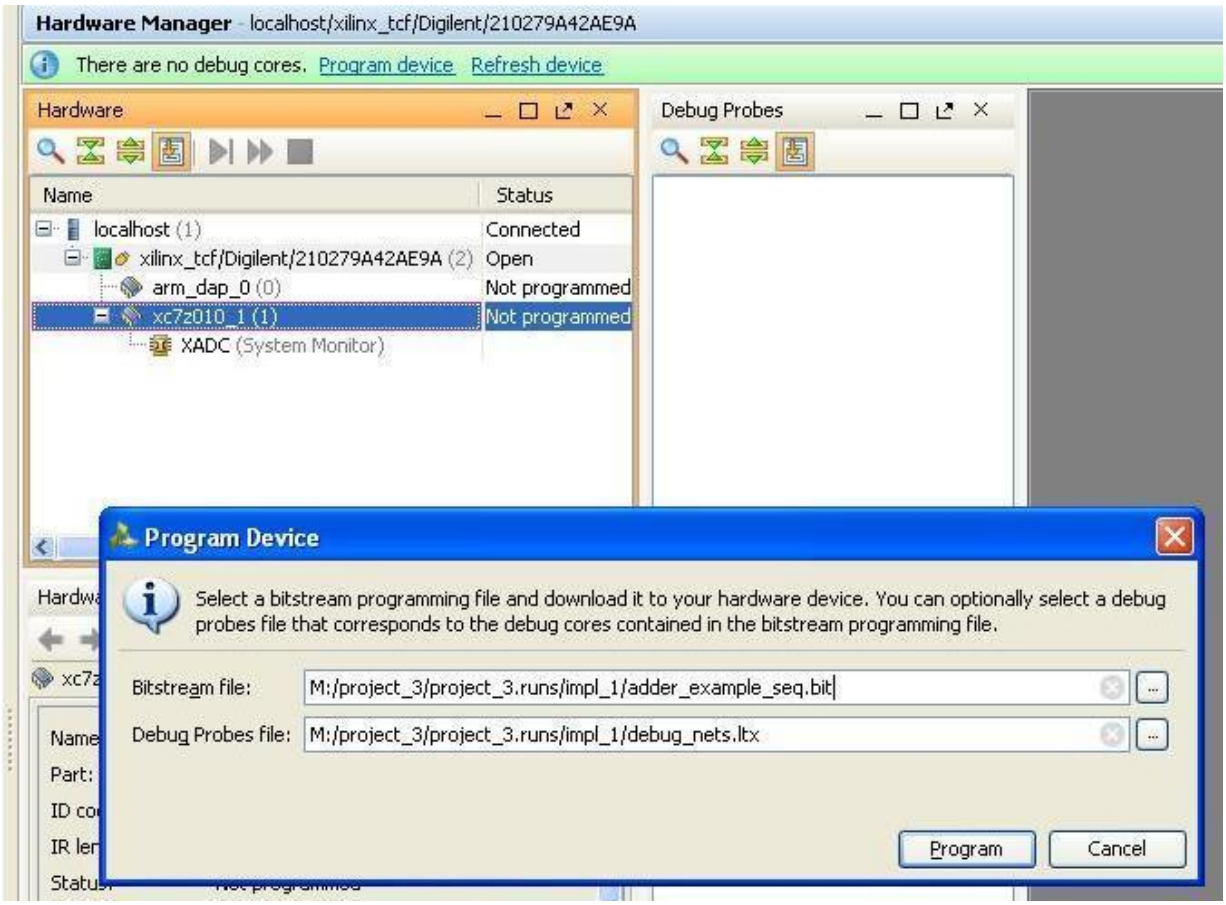


Ο προγραμματισμός του FPGA γίνεται μέσω του Vivado. Τα βήματα για τον προγραμματισμό του FPGA είναι:

1) Αφού έχει ολοκληρωθεί η διαδικασία παραγωγής του αρχείου προγραμματισμού (**file_name.bit**), συνδέουμε την αναπτυξιακή κάρτα **Zybo** με το USB καλώδιο στο PC και ανοίγουμε την τροφοδοσία.
2) Εκτελούμε την διεργασία **Open Hardware Manager** που βρίσκεται στο **Flow Navigator > Program and Debug**.
3) Στο σημείο αυτό επιλέγουμε **Open a new hardware target** από την πράσινη μπάρα που έχει εμφανιστεί στη δεξιά πλευρά του κεντρικού παραθύρου του Vivado. Στο παράθυρο που θα εμφανιστεί πατάμε **Next** μέχρι να εμφανιστεί η επιλογή **Finish**. Εάν όλα έχουν γίνει σωστά και δεν εμφανιστεί κάποιο πρόβλημα θα πρέπει να δούμε την παρακάτω εικόνα:



4) Στη συνέχεια, πατώντας το **Program device** εμφανίζεται ένα πλαίσιο με το μοντέλο του FPGA που έχει συνδεθεί. Επιλέγοντας το, ανοίγει ένα νέο παράθυρο που φαίνεται το **bitsteam** αρχείο (π.χ. <design>.bit) που θα χρησιμοποιηθεί για να προγραμματιστεί το FPGA. Πατώντας **Program** το Vivado θα προγραμματίσει το FPGA, το οποίο πλέον είναι έτοιμο να εκτελέσει τη λειτουργία που έχει υλοποιήσει/περιγράψει ο σχεδιαστής.



10. Χρήσιμες Αναφορές

[1] Εισαγωγικό βιβλίο για VHDL: Από τη σελίδα του μαθήματος στο mycources **Εργαλεία > Έγγραφα > VLSI_2018_Lab > Guides_Tutorials_Books_etc. > free_range_vhdl.pdf**
[2] Οδηγίες εγκατάστασης board files για το ZYBO: Board files