ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

M806 Space Data Systems

Space Technologies, Applications and Services

Οδηγός χρήσης των βασικών λειτουργιών του Vivado IDE της XILINX

Νεκτάριος Κρανίτης, Αντώνης Πασχάλης,
Διονύσης Βασιλόπουλος, Ιωάννης Σίδερης

Εαρινό Εξάμηνο
2019 – 2020

Πίνακας Περιεχομένων

Εισαγωγικά στοιχεία

Στον παρόντα οδηγό χρήσης θα παρουσιάσουμε αναλυτικά τις βασικές λειτουργίες του εργαλείου **Vivado IDE** (Integrated Development Environment) της Xilinx μέσα από τη σχεδίαση απλών ψηφιακών κυκλωμάτων (συνδυαστικών και ακολουθιακών) με τη χρήση της γλώσσας περιγραφής υλικού VHDL και την υλοποίησή τους στην αναπτυξιακή κάρτα **ZedBoard**, που διαθέτει τη διάταξη FPGA της XILINX **Zynq XC7Z020-1CLG484** All Programmable SoC (AP SoC).

Αρχικά, θα δούμε τα βασικά βήματα της εισαγωγής του κώδικα περιγραφής της σχεδίασης (design entry), της εισαγωγής του προγράμματος δοκιμής (testbench entry), της προσομοίωσης (simulation) για την επαλήθευση της ορθής σχεδίασης σύμφωνα με τις προδιαγραφές, της σύνθεσης (synthesis) και της υλοποίησης (implementation) που απαιτούνται για τη σχεδίαση ενός ψηφιακού κυκλώματος, χρησιμοποιώντας τις προκαθορισμένες ρυθμίσεις και αναλύοντας τα αποτελέσματα που προκύπτουν σε κάθε βήμα, χωριστά. Αυτά είναι και τα βήματα που θα ακολουθήσετε κατά τη σχεδίαση του επεξεργαστή στο πλαίσιο του μαθήματος.

Τέλος, θα δούμε πώς παράγεται το bitstream, ώστε να μπορεί να «κατέβει» η σχεδίαση στη διάταξη FPGA της αναπτυξιακής κάρτας Zedboard για την τελική δοκιμή της λειτουργίας του ψηφιακού μας κυκλώματος. Το τελευταίο στάδιο είναι προαιρετικό υπό τις παρούσες συνθήκες.

0-1. **Μαθησιακοί στόχοι**

Με την εκμάθηση του οδηγού χρήσης θα είστε σε θέση να:

- δημιουργείτε ένα **Project** στο Vivado IDE και να δηλώστε ως target device, τη διάταξη Zynq που βρίσκεται στην αναπτυξιακή κάρτα ZedBoard,
- δημιουργείτε ένα κατάλληλο αρχείο τύπου **Xilinx Design Constraint (XDC)** για τη δήλωση του σήματος του ρολογιού (CLK), του σήματος RESET, των χρονικών περιορισμών (timing constraints), καθώς και φυσικών περιορισμών (physical constraints) που σχετίζονται με τη χρησιμοποιούμενη αναπτυξιακή κάρτα, όπως οι αντιστοιχήσεις των χρησιμοποιούμενων ακροδεκτών,
- εισάγεται τον **κώδικα περιγραφής υλικού** σε γλώσσα VHDL ενός συνδυαστικού ή ακολουθιακού ψηφιακού κυκλώματος (δημιουργία αρχείων τύπου .vhd), ώστε το εργαλείο Vivado IDE να παράγει το **behavioral (elaborated design) model**, που προκύπτει μετά την ανάλυση στο επίπεδο RTL του κώδικα VHDL,
- εισάγετε το κατάλληλο **πρόγραμμα δοκιμής (testbench)** σε γλώσσα VHDL (δημιουργία αρχείων τύπου .vhd), ώστε το εργαλείο Vivado IDE να δύναται να προχωρήσει στα στάδια της προσομοίωσης.
- εκτελείτε **όλα τα στάδια της προσομοίωσης** (αναφέρονται αναλυτικά στη συνέχεια) με τον Simulator XSIM που διατίθεται μέσω του εργαλείου Vivado IDE,

- εκτελείτε τη διαδικασία της **σύνθεσης**, ώστε το εργαλείο Vivado IDE να παράγει το **synthesized design model**, που προκύπτει μετά τη σύνθεση του κώδικα VHDL, και να αναλύεται τα αποτελέσματα που προκύπτουν στο στάδιο της σύνθεσης,
- εκτελείτε τη διαδικασία της **υλοποίησης**, ώστε το εργαλείο Vivado IDE να παράγει το **implemented design model**, που προκύπτει μετά την υλοποίηση σε συγκεκριμένη τεχνολογία FPGA του synthesized design model, και να αναλύεται τα αποτελέσματα που προκύπτουν στο στάδιο της υλοποίησης,
- παράγετε το **αρχείο προγραμματισμού (bitstream)** της διάταξης FPGA και προγραμματίζετε τη **διάταξη Zynq** που βρίσκεται στην αναπτυξιακή κάρτα ZedBoard χρησιμοποιώντας το αρχείο bitstream (απαιτείται πρόσβαση στην κάρτα).

0-2. Πιθανές προσομοιώσεις μέσω του εργαλείου Vivado IDE

Μετά τη δημιουργία του κατάλληλου **προγράμματος δοκιμής (testbench)** στη γλώσσα VHDL δύναστε να εκτελέσετε τις ακόλουθες πιθανές προσομοιώσεις:

Behavioral simulation

Λογική προσομοίωση που εφαρμόζεται στο **behavioral (elaborated design) model** που προκύπτει μετά την ανάλυση στο επίπεδο RTL του κώδικα VHDL που έχετε εισάγει.

Post synthesis functional simulation

Λογική προσομοίωση που εφαρμόζεται στο **synthesized design model** που προκύπτει μετά τη σύνθεση του κώδικα VHDL. (Προσοχή! πρέπει να ταυτίζεται με το behavioral simulation).

Post synthesis timing simulation

Χρονική προσομοίωση που εφαρμόζεται στο **synthesized design model** που προκύπτει μετά τη σύνθεση του κώδικα VHDL.

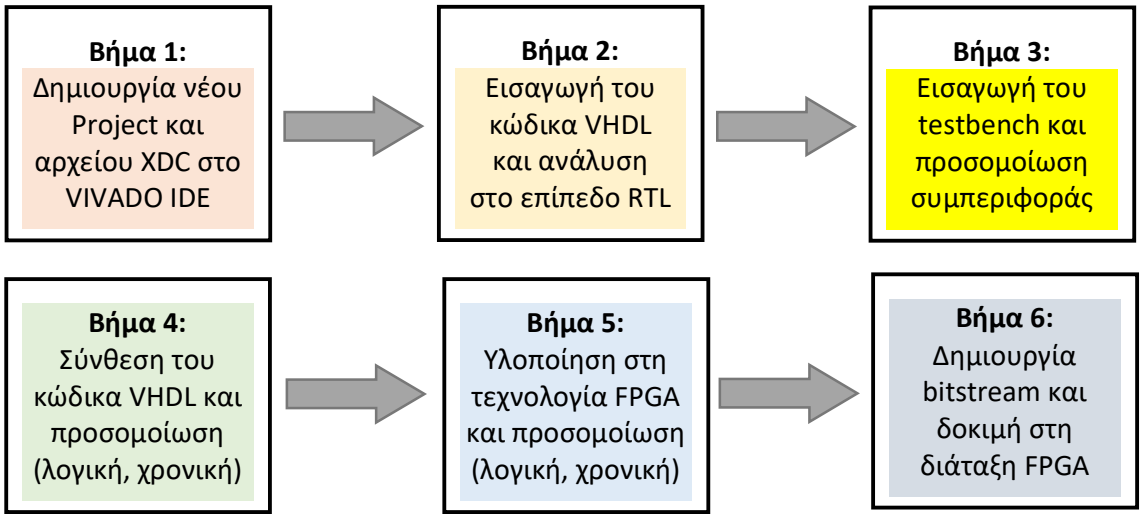
Post implementation functional simulation

Λογική προσομοίωση που εφαρμόζεται στο **implemented design model** που προκύπτει μετά την υλοποίηση σε συγκεκριμένη τεχνολογία FPGA του synthesized design model. (Προσοχή! πρέπει να ταυτίζεται με το behavioral simulation).

Post synthesis timing simulation

Χρονική προσομοίωση που εφαρμόζεται στο **implemented design model** που προκύπτει μετά την υλοποίηση σε συγκεκριμένη τεχνολογία FPGA του synthesized design model.

0-3. **Γενική ροή σχεδίασης και υλοποίησης ενός ψηφιακού κυκλώματος**



Το Βήμα 6 θα περιγραφθεί σε επόμενη έκδοση του οδηγού.

0-4. **Ψηφιακά κυκλώματα που θα σχεδιάσουμε και θα υλοποιήσουμε**

Στην παρούσα εργαστηριακή άσκηση θα επικεντρωθούμε στα ακόλουθα χαρακτηριστικά ψηφιακά κυκλώματα:

1. Παραμετροποιημένος καταχωρητής με RESET και WE των N bit.
2. Παραμετροποιημένος αθροιστής με Cout και OV των N bit.
3. Αθροιστής των 8 bit με καταχωρητές εισόδου και εξόδου ως σύγχρονο ακολουθιακό κύκλωμα.
4. Ανιχνευτής ακολουθίας 2 διαδοχικών bit ως μηχανή πεπερασμένων καταστάσεων (FSM) τύπου Moore.

Βήμα 1: Δημιουργία νέου Project και αρχείου τύπου XDC στο VIVADO IDE

Σημείωση: Οι οδηγίες για τη δημιουργία του νέου project εμφανίζουν μικρές διαφοροποιήσεις ανάλογα με το λειτουργικό σύστημα στο οποίο έχουμε εγκαταστήσει το VIVADO IDE (Linux ή MS Windows).

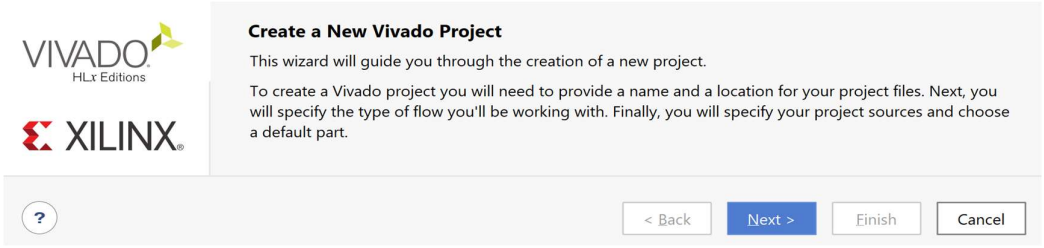
1-1. Εκτέλεση της εφαρμογής VIVADO IDE

1-1.1. Ανοίξτε το Vivado πατώντας στο **εικονίδιο του Vivado** που είναι διαθέσιμο στο Desktop ή εμφανίζεται όταν αναζητήσετε την εφαρμογή "Vivado". Η έκδοση πιθανώς να διαφέρει (π.χ. Vivado 2019.2).

1-2. Δημιουργία ενός νέου project στο VIVADO IDE

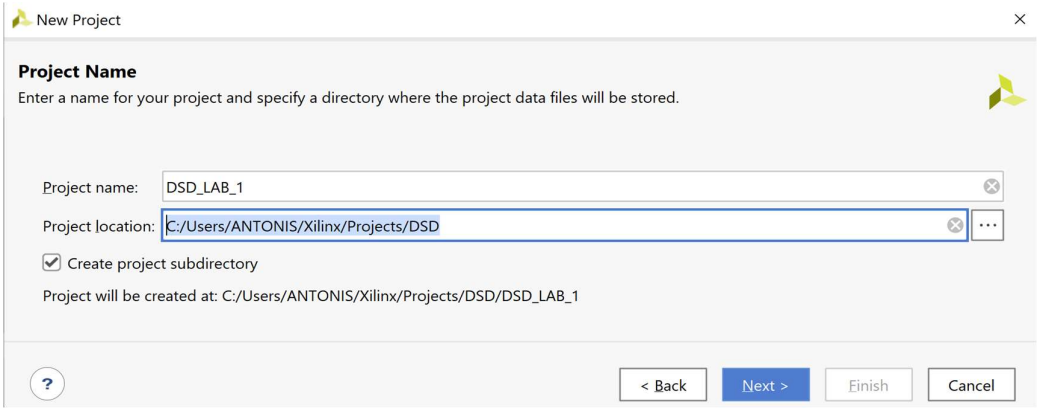
1-2.1. Πατήστε **Create Project** στο παράθυρο *Quick Start* για να ξεκινήσετε τον wizard δημιουργίας ενός νέου project. Εάν το project έχει ήδη δημιουργηθεί μπορούμε να το επιλέξουμε με το Project_name είτε από το **Open Project** στο παράθυρο *Quick Start*, είτε από το παράθυρο *Recent Project*.

1-2.2. Θα δείτε ένα παράθυρο διαλόγου *Create a New Vivado Project*. Πατήστε **Next**. Το κυκλάκι με το ερωτηματικό παρέχει τις απαραίτητες οδηγίες (είναι πολύ χρήσιμο εάν έχετε απορίες).

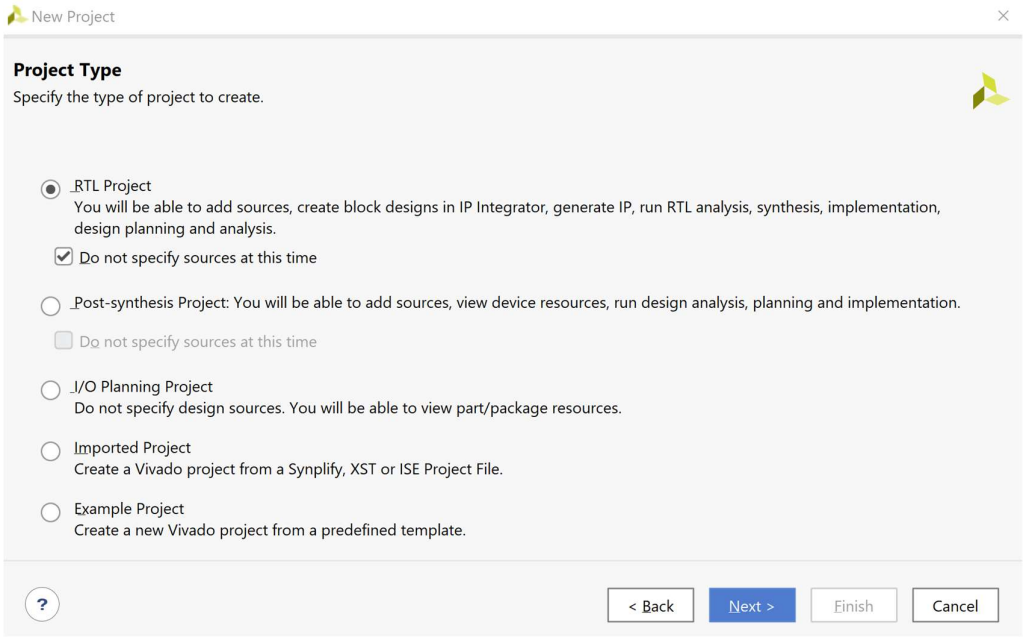


1-2.3. Στο παράθυρο διαλόγου *Project name* πατήστε το κουτάκι με τις τελίτσες του πεδίου **Project location**, ώστε να κάνετε browse και να δημιουργήσετε τον φάκελο που θα τοποθετήσετε το project σας (π.χ. C:/Users/ANTONIS/Xilinx/Projects/DSD), και πατήστε **Select**.

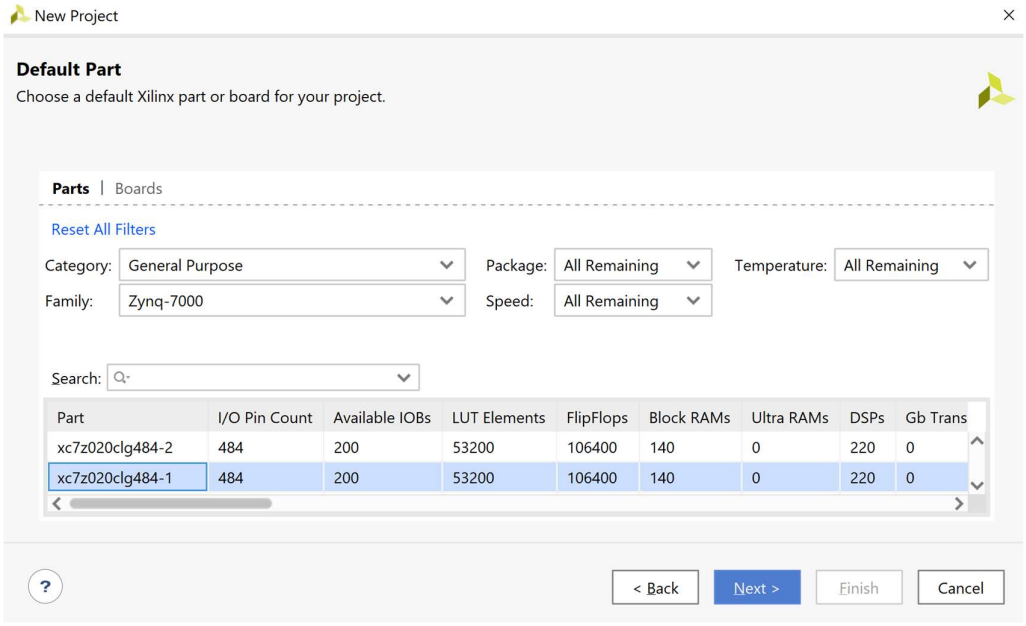
1-2.4. Στη συνέχεια, δηλώστε το Project_name (π.χ. DSD_LAB_1) στο πεδίο **Project name**. Βεβαιωθείτε ότι το κουτί **Create Project Subdirectory box** είναι επιλεγμένο. Πατήστε **Next**.



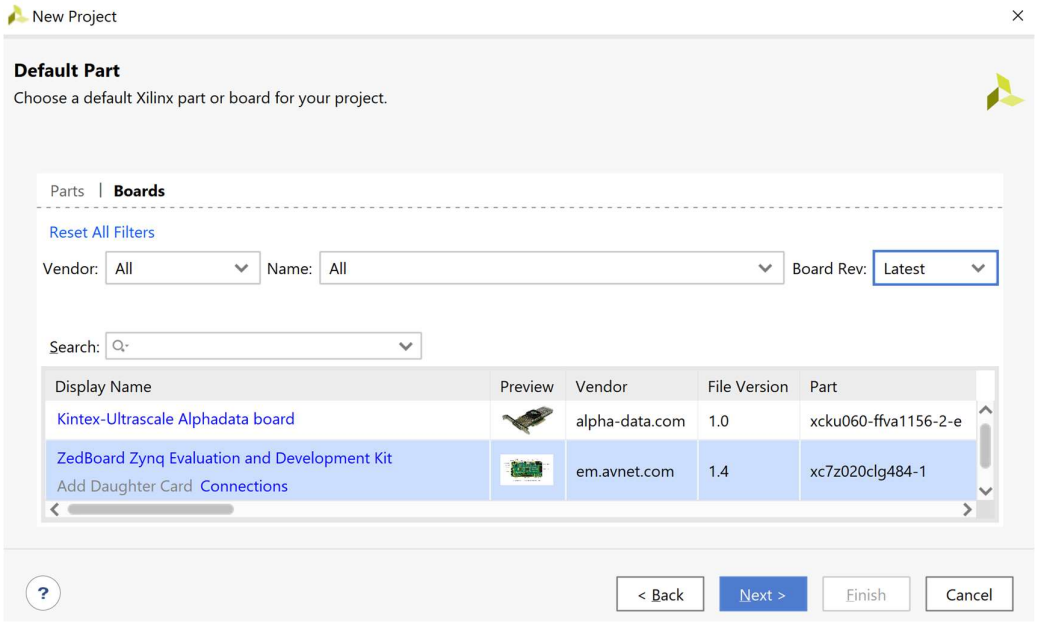
1-2.5. Στο παράθυρο διαλόγου *Project Type* πατήστε την επιλογή **RTL Project**. Βεβαιωθείτε ότι το κουτί **Do not specify sources at this time** είναι επιλεγμένο, ώστε να μην εισάγουμε πηγαία αρχεία VHDL κατά τη διάρκεια της δημιουργίας του project. Πατήστε **Next**.



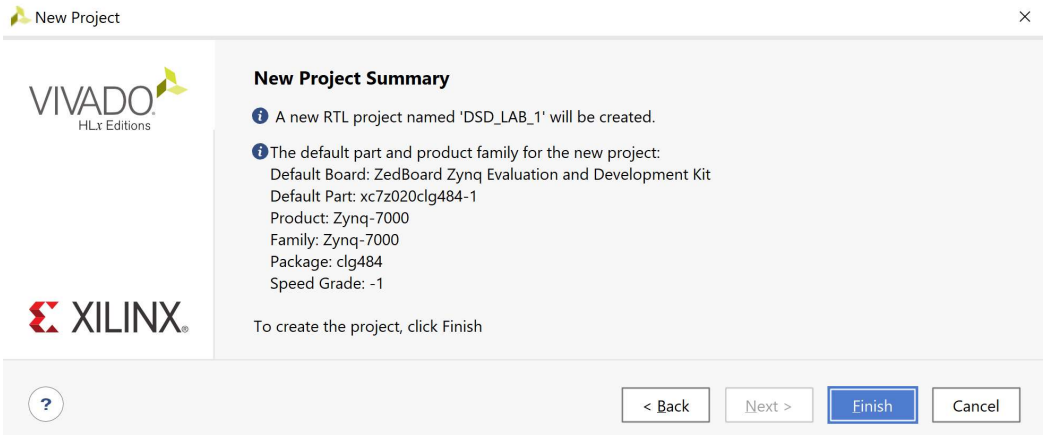
1-2.6. Στο παράθυρο διαλόγου *Default Part* κρατήστε την επιλογή **Parts** και επιλέξτε στο **Category** (π.χ. General Purpose) και στο **Familly** (π.χ. Zynq-7000). Στη συνέχεια επιλέξτε το Part **xc7z020clg484-1** (Zynq-7020, με 484 ακροδέκτες και ταχύτητα (speed grade) -1 (το πιο αργό της οικογένειας) που βρίσκεται στην αναπτυξιακή κάρτα ZedBoard. Παρατηρήστε τα διαθέσιμα resources του επιλεγμένου part (IOB = 200, LUT Elements = 53.200, Flip-Flops = 106.400 (διπλάσια από τα LUT Elements) Block RAMs = 140 και DSPs = 220). Πατήστε **Next**.



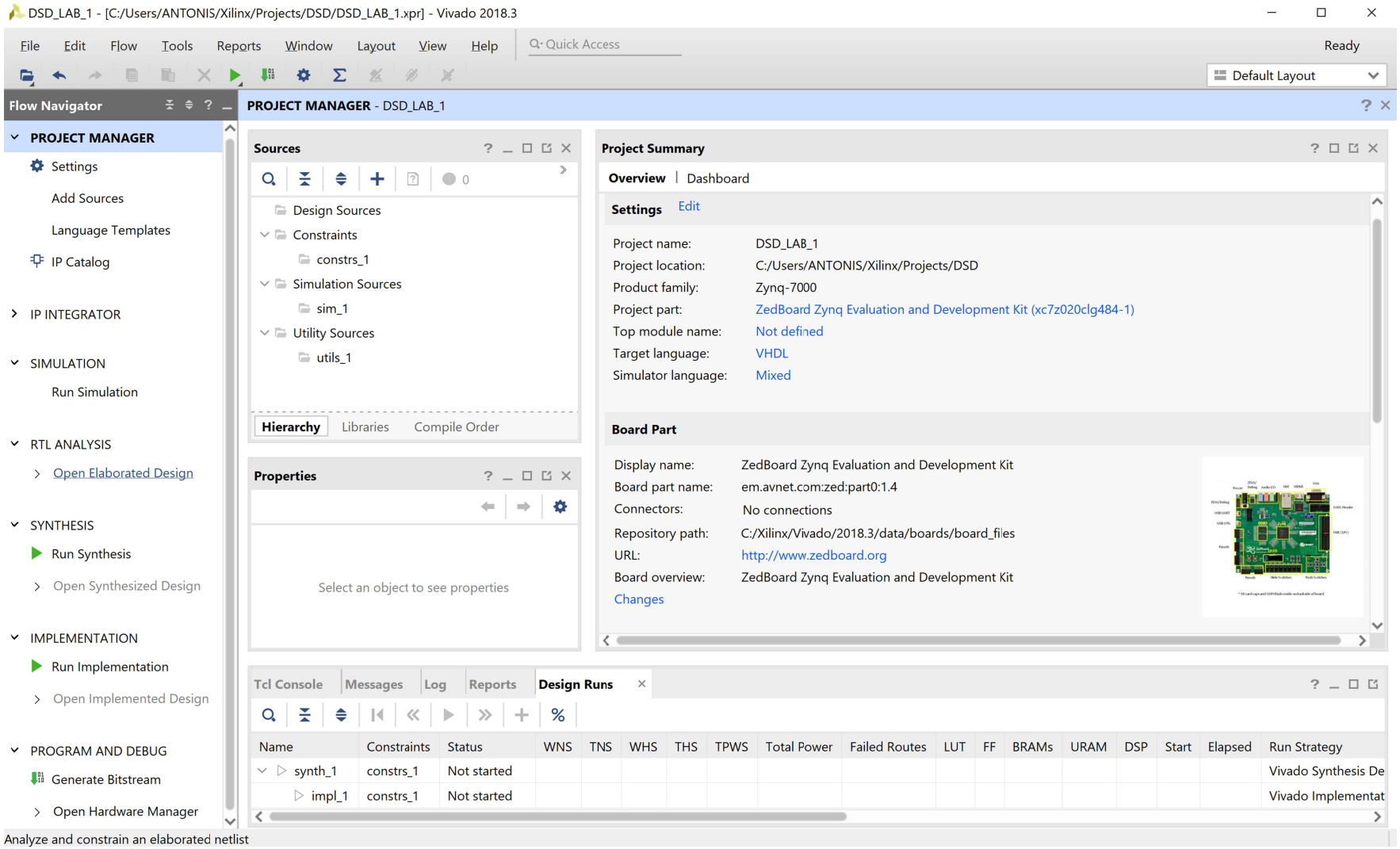
1-2.7. Εναλλακτικά, στο παράθυρο διαλόγου *Default Part* πατήστε την επιλογή **Boards** και επιλέξτε την αναπτυξιακή κάρτα **ZedBoard**. Πατήστε **Next**.



1-2.8. Στο παράθυρο διαλόγου *New Project Summary* βλέπετε το project_name και τα χαρακτηριστικά της επιλεγμένης (default) διάταξης FPGA. Πατήστε **Finish** για να δημιουργηθεί το νέο σας project που ονομάζεται **DSD_LAB_1**.



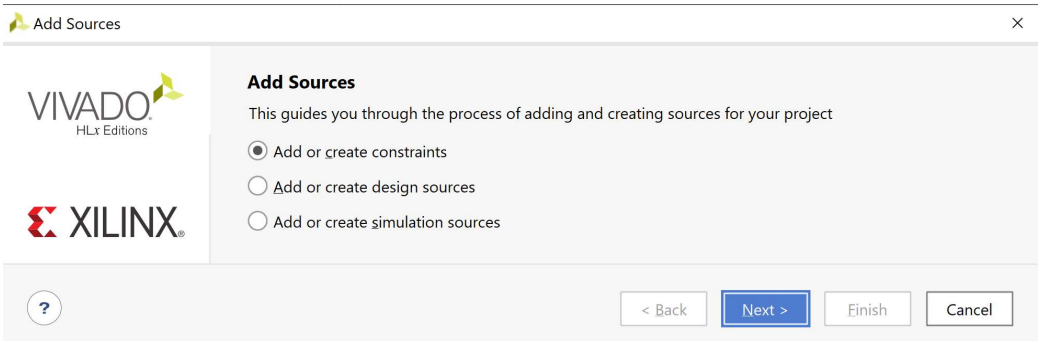
1-2.9. Στην επόμενη σελίδα φαίνεται το περιβάλλον του **PROJECT MANAGER** πάνω στο οποίο θα σχεδιάσουμε τα ψηφιακά κυκλώματά μας. Αναγνωρίζουμε την οριζόντια μπάρα επάνω με επιλογές File, Edit, Flow, Tools, Reports, Window, Layout, View και Help. Το κατακόρυφο παράθυρο αριστερά του Flow Navigator, όπου με κατάλληλη επιλογή εκτελούνται όλα τα βήματα της ψηφιακής σχεδίασης. Το παράθυρο Sources, όπου φαίνονται όλα τα πηγαία αρχεία του project διαρθρωμένα στα directories: Design Sources, Constraints, Simulation Sources και Utility Sources (αρχικά είναι άδειο). Το παράθυρο Properties. Το παράθυρο Project Summary (σε κάθε βήμα της ψηφιακής σχεδίασης διαφοροποιείται κατάλληλα). Τέλος, το κάτω οριζόντιο παράθυρο πολλαπλών χρήσεων, που χρησιμεύει μεταξύ άλλων ως Tcl Console και για την ανάλυση των αποτελεσμάτων της σύνθεσης και της υλοποίησης.


DSD_LAB_1 - [C:/Users/ANTONIS/Xilinx/Projects/DSD/DSD_LAB_1.xpr] - Vivado 2018.3
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
PROJECT MANAGER - DSD_LAB_1
Sources
Design Sources
Constraints
constrs_1
Simulation Sources
sim_1
Utility Sources
utils_1
Hierarchy Libraries Compile Order
Properties
Select an object to see properties
Project Summary
Overview | Dashboard
Settings Edit
Project name: DSD_LAB_1
Project location: C:/Users/ANTONIS/Xilinx/Projects/DSD
Product family: Zynq-7000
Project part: ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1)
Top module name: Not defined
Target language: VHDL
Simulator language: Mixed
Board Part
Display name: ZedBoard Zynq Evaluation and Development Kit
Board part name: em.avnet.com:zed:part0:1.4
Connectors: No connections
Repository path: C:/Xilinx/Vivado/2018.3/data/boards/board_files
URL: http://www.zedboard.org
Board overview: ZedBoard Zynq Evaluation and Development Kit
Changes
Tcl Console Messages Log Reports Design Runs
Name Constraints Status WNS TNS WHS THS TPWS Total Power Failed Routes LUT FF BRAMs URAM DSP Start Elapsed Run Strategy
synth_1 constrs_1 Not started Vivado Synthesis De
impl_1 constrs_1 Not started Vivado Implementat
Analyze and constrain an elaborated netlist

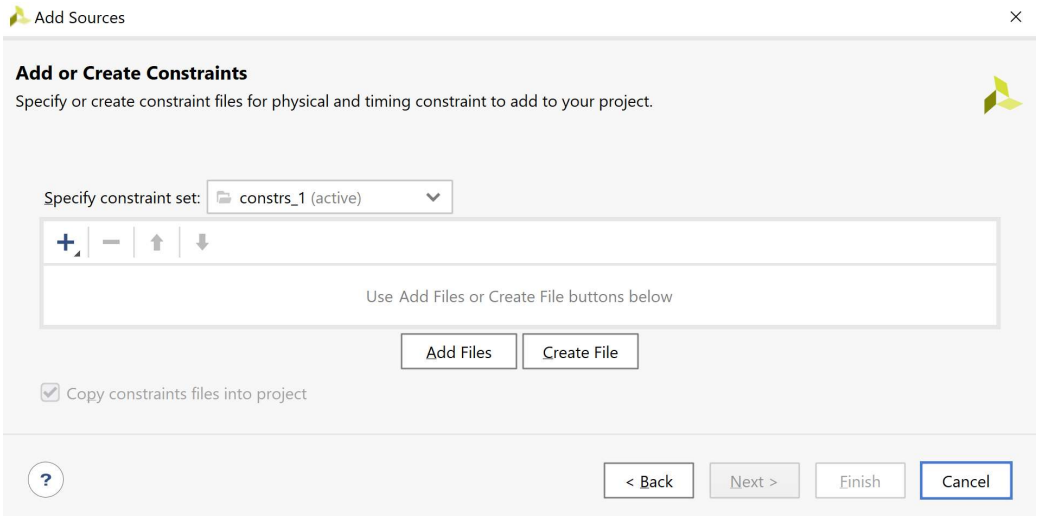
1-3. Δημιουργία του αρχείου τύπου XDC στο VIVADO IDE

1-3.1. Πατήστε το **+** στο παράθυρο *Sources* του PROJECT MANAGER για να ξεκινήσετε τον wizard δημιουργίας ενός νέου αρχείου (source).

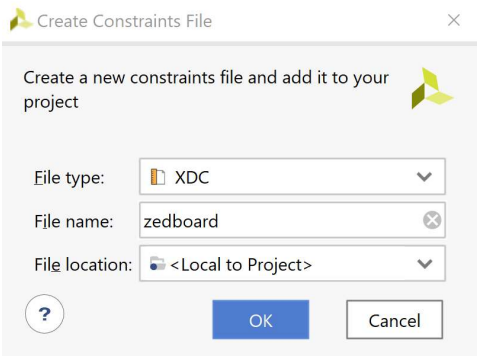
1-3.2. Στο παράθυρο διαλόγου *Add Sources* επιλέξτε το Add or create constraints, ώστε να δημιουργήσετε ένα αρχείο περιγραφής περιορισμών (constraint file) τύπου XDC. Σε αυτό το αρχείο, που αρχικά είναι άδειο, δηλώνουμε το σήμα του ρολογιού (CLK), το σήμα RESET, τους χρονικούς περιορισμούς (timing constraints), καθώς και τους φυσικούς περιορισμούς (physical constraints) που σχετίζονται με τη χρησιμοποιούμενη αναπτυξιακή κάρτα, όπως οι αντιστοιχίσεις των χρησιμοποιούμενων ακροδεκτών. Πατήστε **Next**.



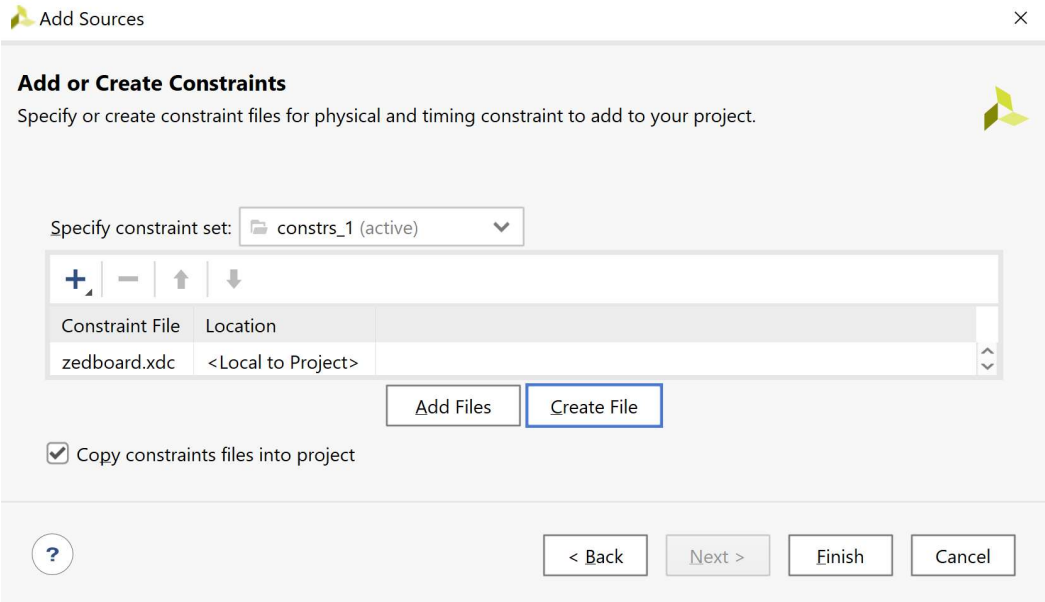
1-3.3. Στο παράθυρο διαλόγου *Add or Create Constraints* πατήστε την επιλογή **Create File** για τη δημιουργία του άδειου αρχείου τύπου XDC, που θα τοποθετηθεί στο ήδη καθορισμένο constraint set *constrs_1*.



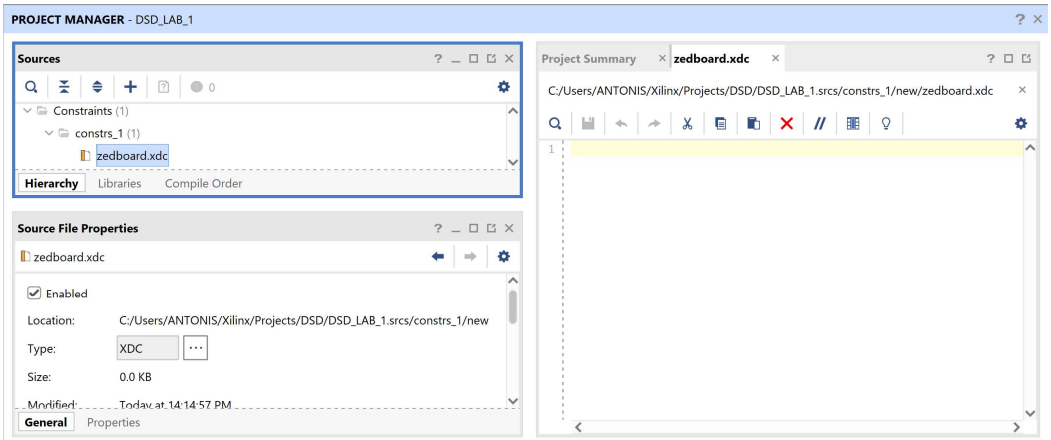
1-3.4. Στο παράθυρο διαλόγου *Create Constraints File* δηλώστε το όνομα του αρχείου XDC (π.χ. zedboard), αφού οι περιορισμοί σχετίζονται με την αναπτυξιακή κάρτα Zedboard. Πατήστε **OK**.



1-3.5. Επιστρέφετε στο παράθυρο διαλόγου *Add or Create Constraints*, όπου φαίνεται ότι έχει δημιουργηθεί το αρχείο **zedboard.xdc** και έχει συμπεριληφθεί στα αρχεία του project που ονομάζεται **DSD_LAB_1**. Πατήστε **Finish**.



1-3.6. Επιβεβαιώστε τη δημιουργία του αρχείου **zedboard.xdc** στο παράθυρο *Sources* του PROJECT MANAGER και ανοίξτε το με διπλό κλικ. Αρχικά είναι άδειο.



1-3.7. Με copy-paste συμπληρώνουμε το αρχείο **zedboard.xdc** με βάση το **πρότυπο zedboard.xdc**, που παρατίθεται στο Παράρτημα Γ από το κατασκευαστή (AVNET) της αναπτυξιακής κάρτας **Zedboard**. Αφαιρούμε το σύμβολο (**#**), όπου απαιτείται. Τέλος κάνουμε **Save**.

Στη συνέχεια φαίνεται το περιεχόμενο του **zedboard.xdc**, που καλύπτει τις εργαστηριακές μας ανάγκες (δήλωση και χρονικοί περιορισμοί (περίοδος 10 ns) για το σήμα του ρολογιού CLK, φυσικοί περιορισμοί για σήμα CLK, τα 5 GPIO push buttons, που μας επιτρέπουν να παράγουμε σήματα RESET και EN με το πάτημα του αντίστοιχου κουμπιού, τα 8 DIP Switches για αρχικές σταθερές τιμές σε εισόδους σημάτων και τα 8 LEDs για παρατήρηση τιμών σε εξόδους σημάτων).

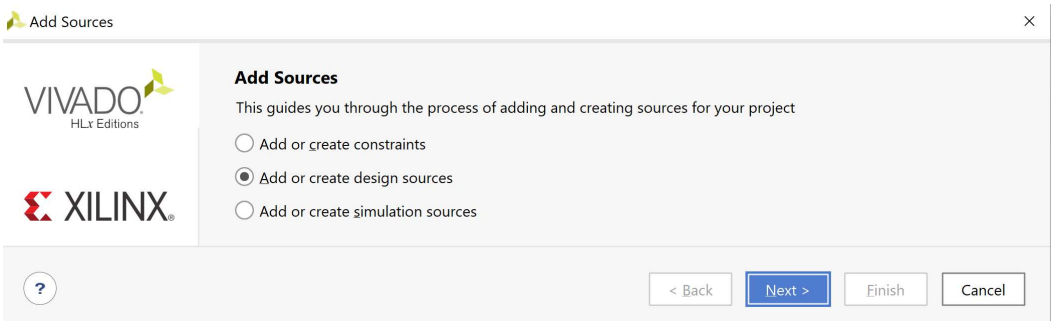
```
##############################################################
# ZedBoard Pin Assignments
##############################################################

# CLK - Zedboard 100MHz oscillator
set_property -dict { PACKAGE_PIN Y9 IOSTANDARD LVCMOS33 } [get_ports {CLK}]

# User GPIO push button for RESET and EN purposes
#set_property PACKAGE_PIN P16 [get_ports {BTNC}];  # "BTNC" central
#set_property PACKAGE_PIN R16 [get_ports {BTND}];  # "BTND" down
#set_property PACKAGE_PIN N15 [get_ports {BTNL}];  # "BTNL" left
#set_property PACKAGE_PIN R18 [get_ports {BTNR}];  # "BTNR" right
#set_property PACKAGE_PIN T18 [get_ports {BTNU}];  # "BTNU" up

# User DIP Switches – 8 bit user input
#set_property PACKAGE_PIN F22 [get_ports {SW0}];  # "SW0"
#set_property PACKAGE_PIN G22 [get_ports {SW1}];  # "SW1"
#set_property PACKAGE_PIN H22 [get_ports {SW2}];  # "SW2"
#set_property PACKAGE_PIN F21 [get_ports {SW3}];  # "SW3"
#set_property PACKAGE_PIN H19 [get_ports {SW4}];  # "SW4"
#set_property PACKAGE_PIN H18 [get_ports {SW5}];  # "SW5"
#set_property PACKAGE_PIN H17 [get_ports {SW6}];  # "SW6"
#set_property PACKAGE_PIN M15 [get_ports {SW7}];  # "SW7"

# User LEDs – 8 bit user output
#set_property PACKAGE_PIN T22 [get_ports {LD0}];  # "LD0"
#set_property PACKAGE_PIN T21 [get_ports {LD1}];  # "LD1"
#set_property PACKAGE_PIN U22 [get_ports {LD2}];  # "LD2"
#set_property PACKAGE_PIN U21 [get_ports {LD3}];  # "LD3"
#set_property PACKAGE_PIN V22 [get_ports {LD4}];  # "LD4"
#set_property PACKAGE_PIN W22 [get_ports {LD5}];  # "LD5"
#set_property PACKAGE_PIN U19 [get_ports {LD6}];  # "LD6"
#set_property PACKAGE_PIN U14 [get_ports {LD7}];  # "LD7"


##########################################################
##ZedBoard Timing Constraints
##########################################################

# define clock and period
create_clock -period 10 -name CLK -waveform {0.000 5.000} [get_ports {CLK}]
```

Βήμα 2: Εισαγωγή του κώδικα VHDL και ανάλυση στο επίπεδο RTL

2-1. Δημιουργία νέου αρχείου τύπου VHD στο VIVADO IDE

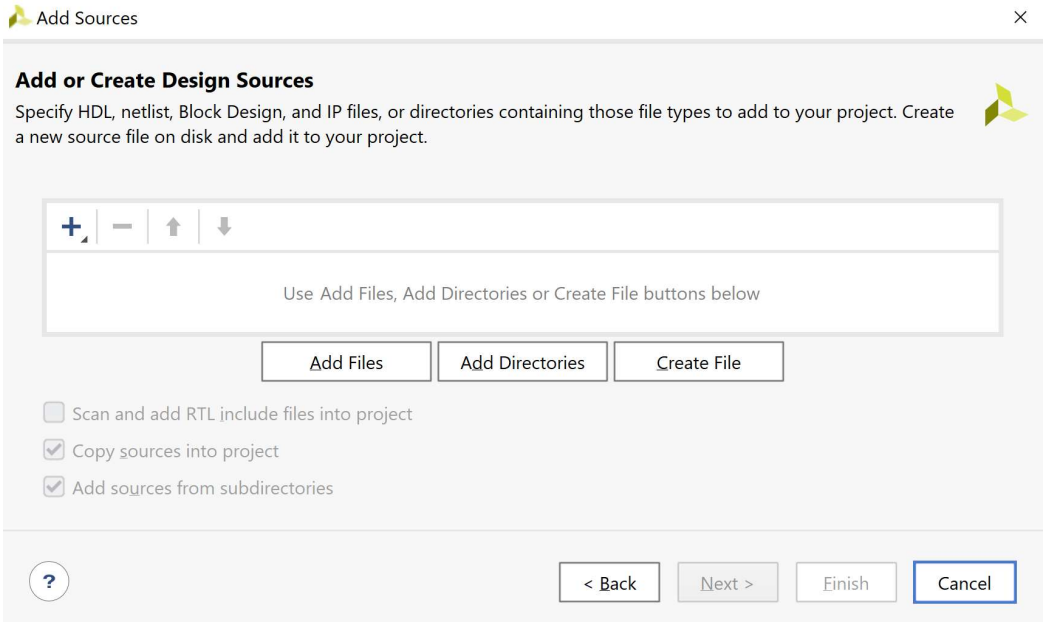
2-1.1. Πατήστε το **+** στο παράθυρο *Sources* του PROJECT MANAGER για να ξεκινήσετε τον wizard δημιουργίας ενός νέου αρχείου (source).

2-1.2. Στο παράθυρο διαλόγου *Add Sources* επιλέξτε το Add or create design sources, ώστε να δημιουργήσετε ένα νέο πηγαίο αρχείο περιγραφής στο επίπεδο RTL ενός ψηφιακού κυκλώματος (ή συστήματος, ανάλογα με την πολυπλοκότητα της σχεδίασης) στη γλώσσα VHDL (design source file) τύπου VHD. Σε αυτό το αρχείο εισάγουμε τον κώδικα VHDL. Πατήστε **Next**.

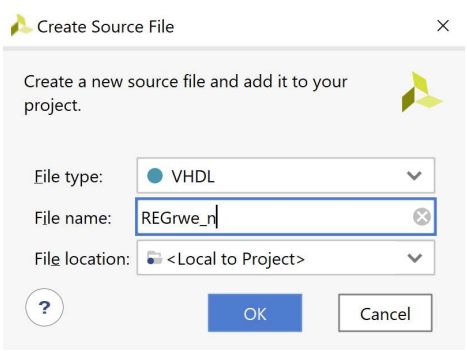


2-1.3. Στο παράθυρο διαλόγου *Add or Create Design Sources* πατήστε την επιλογή **Create File** για τη δημιουργία του design source file που θα τοποθετηθεί στο ήδη καθορισμένο design source set *sources_1*.

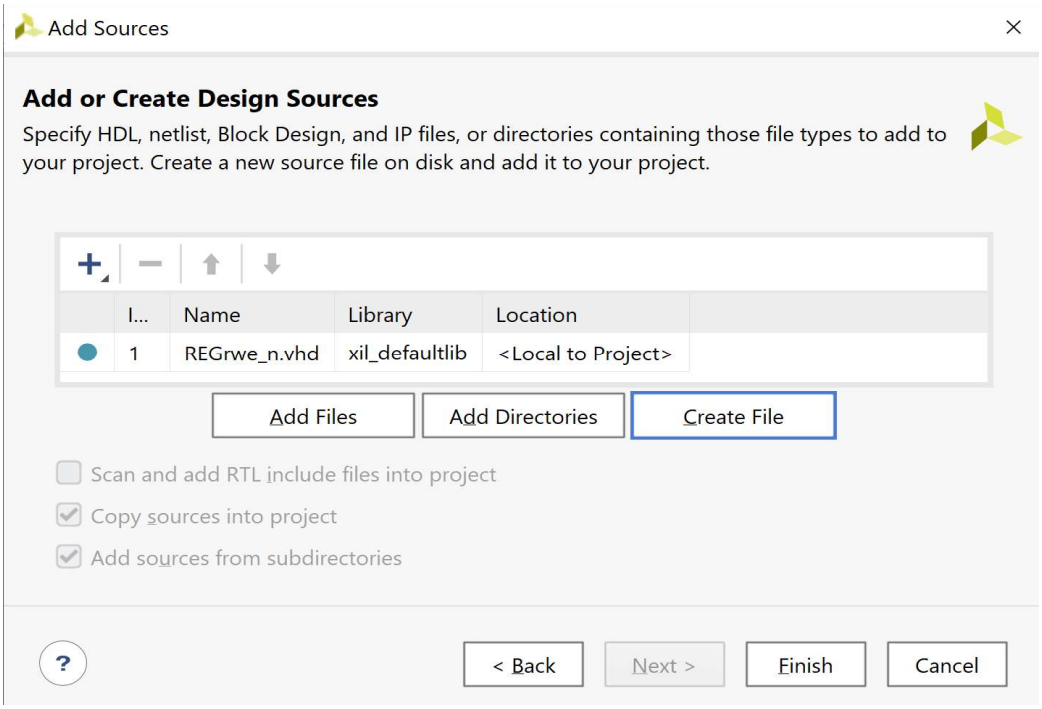
Εάν απαιτείται επιλέξτε την **VHDL** ως *Target Language* και το **Mixed** ως *Simulator Language*.



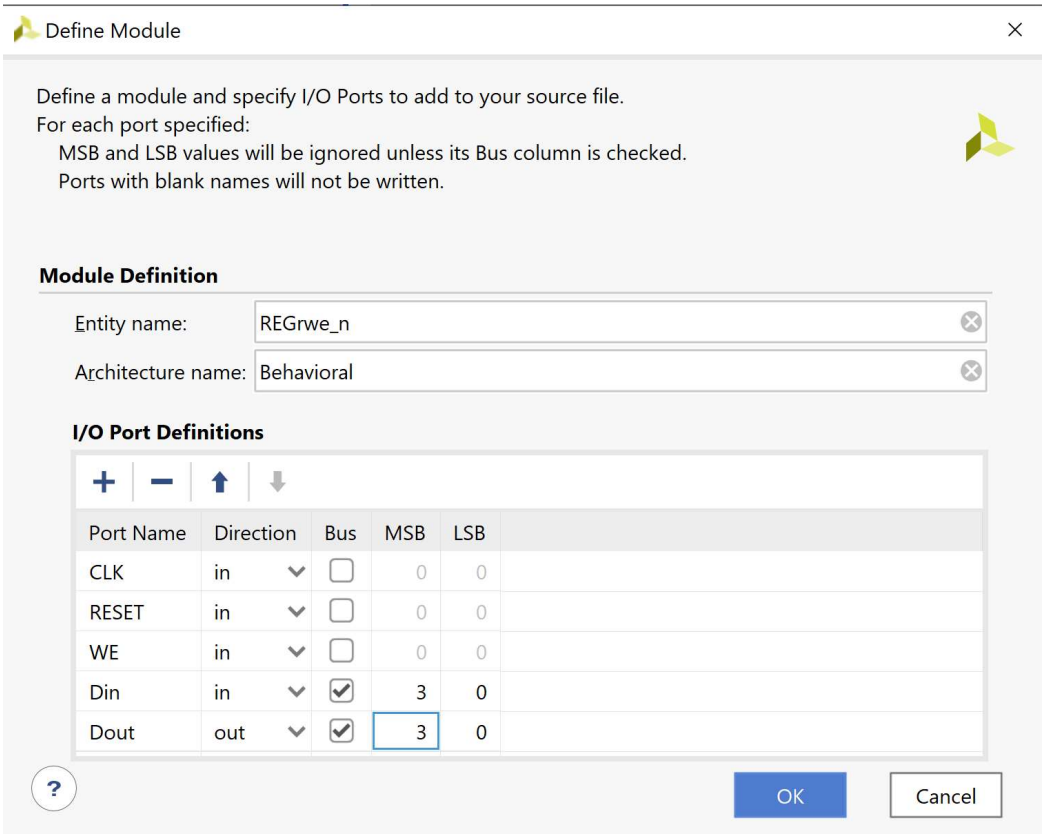
2-1.4. Στο παράθυρο διαλόγου *Create Source File* δηλώστε το όνομα του αρχείου VHD (π.χ. **REGrwe_n**), αφού σε αυτό το αρχείο θα περιγράψετε στη γλώσσα VHDL έναν παραμετροποιημένο καταχωρητή των N bit με σύγχρονη επαναφορά στο 0 με την ενεργοποίηση του σήματος RESET (RESET = 1) και με έγκριση εγγραφής με την ενεργοποίηση του σήματος ελέγχου WE (WE = 1). Πατήστε **OK**.



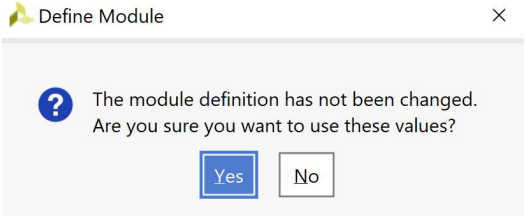
2-1.5. Επιστρέφετε στο παράθυρο διαλόγου *Add or Create Design Sources*, όπου φαίνεται ότι έχει δημιουργηθεί το αρχείο **REGrwe_n.vhd** και έχει συμπεριληφθεί στα αρχεία του project που ονομάζεται DSD_LAB_1. Πατήστε **Finish**.



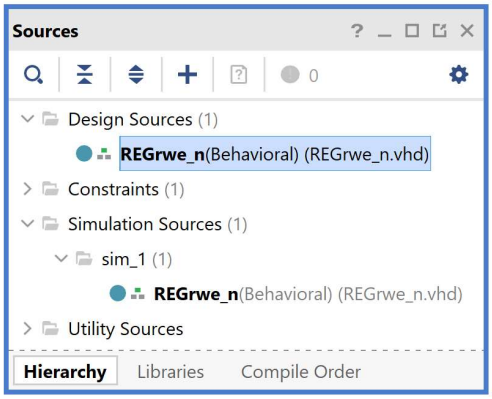
2-1.6. Αμέσως μετά εμφανίζεται το παράθυρο διαλόγου *Define Module*, όπου σας παρέχεται η δυνατότητα της δήλωσης του ονόματος της οντότητας, του ονόματος της αρχιτεκτονικής και των ports της οντότητας. Ως όνομα της οντότητας διατηρείτε το ίδιο όνομα που είχατε δώσει στο αρχείο (**REGrwe_n**). Επιλέξτε ως όνομα της αρχιτεκτονικής κάποιο από τα ονόματα *behavioral, structural, dataflow, mixed* ανάλογα με το είδος της περιγραφής που θα κάνετε στη γλώσσα VHDL. Στη συνέχεια, προαιρετικά δηλώστε τα ports στο **I/O Port Definitions**. Στη συγκεκριμένη περίπτωση επιλέξτε το όνομα **behavioral**. Ας υποθέσουμε ότι αρχικά δηλώνουμε τα ports ενός μη παραμετροποιημένου καταχωρητή των 4 bit με εισόδους **CLK**, **RESET**, **WE**, **Din[3:0]** και έξοδο **Dout[3:0]**. Εισάγετε το όνομα του σήματος ή της αρτηρίας στο port name. Για τις αρτηρίες επιλέξτε επιπλέον το Bus και ορίστε την τιμή του MSB (3) και του LSB (0). Πατήστε το **+** για δήλωση επιπλέον ports με τον ίδιο τρόπο. Τέλος, πατήστε **OK**.



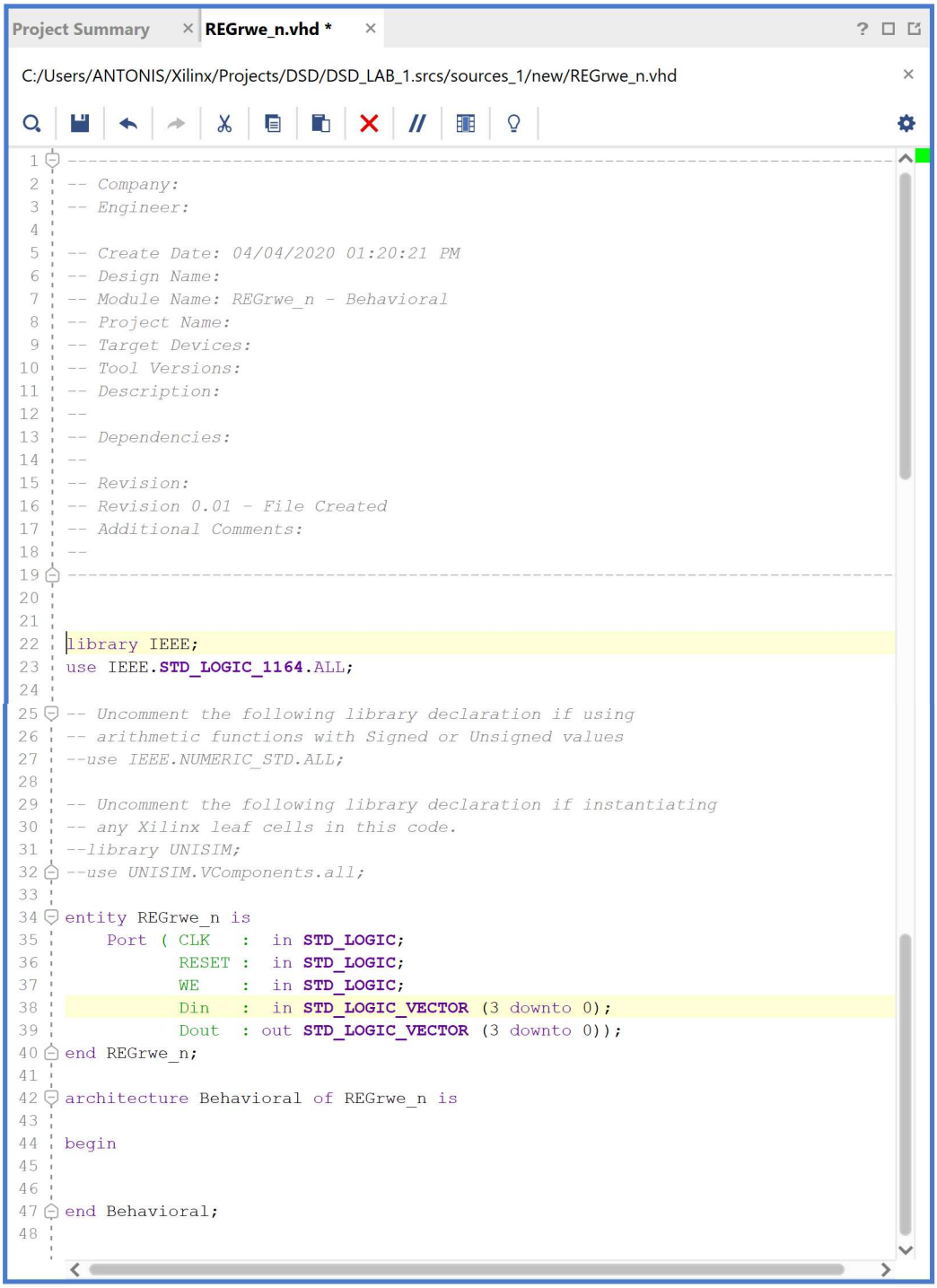
Μπορούμε να παρακάμψουμε τη δήλωση των ports. Στο παράθυρο προειδοποίησης *Define Module* που θα εμφανισθεί πατήστε **Yes**.



2-1.7. Επιβεβαιώστε τη δημιουργία του αρχείου **REGrwe_n.vhd** στο παράθυρο *Sources* του PROJECT MANAGER (επίσης φαίνεται το όνομα της οντότητας **REGrwe_n** και το όνομα της αρχιτεκτονικής **Behavioral**). Το αρχείο αυτό είναι αποθηκευμένο και στο design source set *sources_1* και στο simulation source set *sim_1* του project DSD_LAB1. Επειδή δεν υπάρχει άλλη οντότητα διαθέσιμη στα design sources του project, η οντότητα **REGrwe** ορίζεται αυτόματα ως η κορυφαία οντότητα της ιεραρχίας (**top**) επί της οποίας θα εκτελεσθούν στη συνέχεια τα επόμενα βήματα της σχεδίασης (RTL analysis, Synthesis, Implementation, Program and debug). Η κορυφαία οντότητα (**top design source)** διαφοροποιείται από τις υπόλοιπες οντότητες του design source set *sources_1* με το σύμβολο και τα **έντονα** (bold) γράμματα στο όνομα της οντότητας.

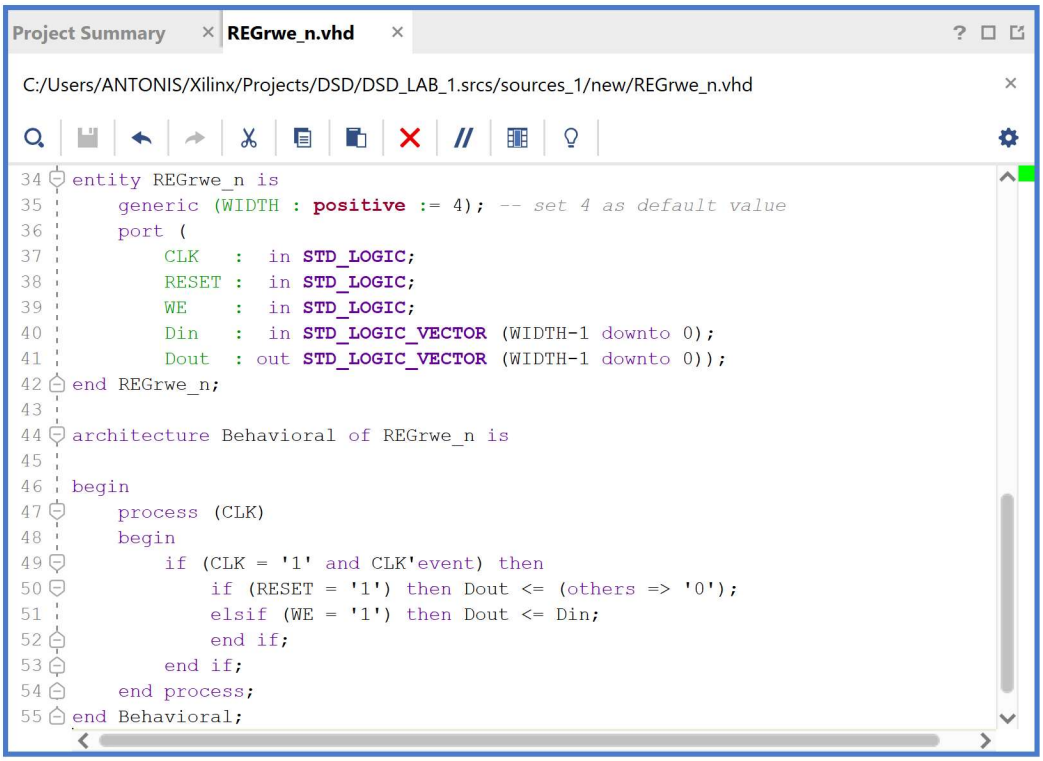


2-1.8. Ανοίξτε το αρχείο **REGrwe_n.vhd** με διπλό κλικ στο επιλεγμένο αρχείο. Τα ports της οντότητας έχουν ήδη ορισθεί, αλλά λείπει ο ορισμός της αρχιτεκτονικής της οντότητας.



Στις δυνατότητες του VHDL editor να λάβετε υπόψη σας το *Toggle Line Comments* και το *Toggle Column Selection Mode* (ιδιαίτερα χρήσιμο στην αντιγραφή ονομάτων σημάτων που έχετε στοιχίσει εκ των προτέρων) που βρίσκονται μετά το ✕.

2-1.9. Παραμετροποιείστε τις αρτηρίες Din και Dout με τη δήλωση generic πριν τη δήλωση των ports και συμπληρώστε τον ορισμό της αρχιτεκτονικής με βάση τον κώδικα που δίδεται στη σελίδα 301 του Κεφαλαίου 4 «Γλώσσες περιγραφής υλικού – Πλήρης έκδοση» των παραδόσεων του μαθήματος. Τέλος πατήστε **Save File**.

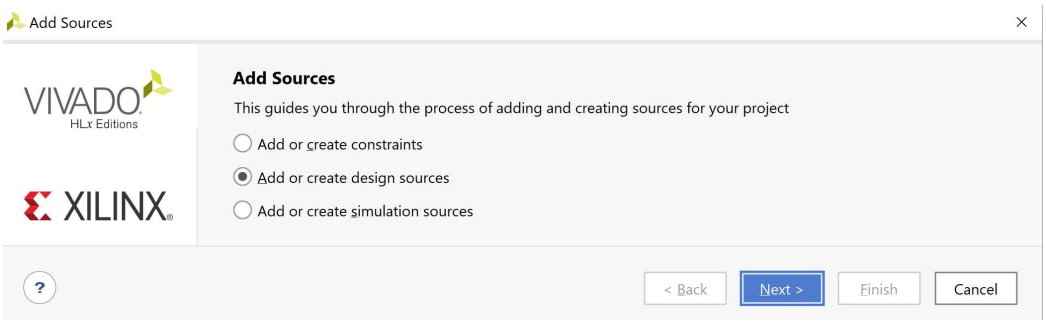


2-1.10. Χρησιμοποιήστε τον File Manager και κοιτάξτε στο φάκελο που ορίσατε το project. Θα βρείτε ότι έχει δημιουργηθεί εντός αυτού, μεταξύ άλλων φακέλων, και ο φάκελος **DSD_LAB_1.srcs**, καθώς και το Vivado Project File **DSD_LAB_1**. Μέσα στον φάκελο **DSD_LAB_1.srcs** θα βρείτε τους υποφακέλους **constrs_1** και **sources_1**. Στον υποφάκελο **constrs_1/new** θα βρείτε το **ZedBoard.xdc** (constraints file), ενώ στον υποφάκελο **sources_1/new** θα βρείτε το **REGrwe_n.vhd** (design source file).

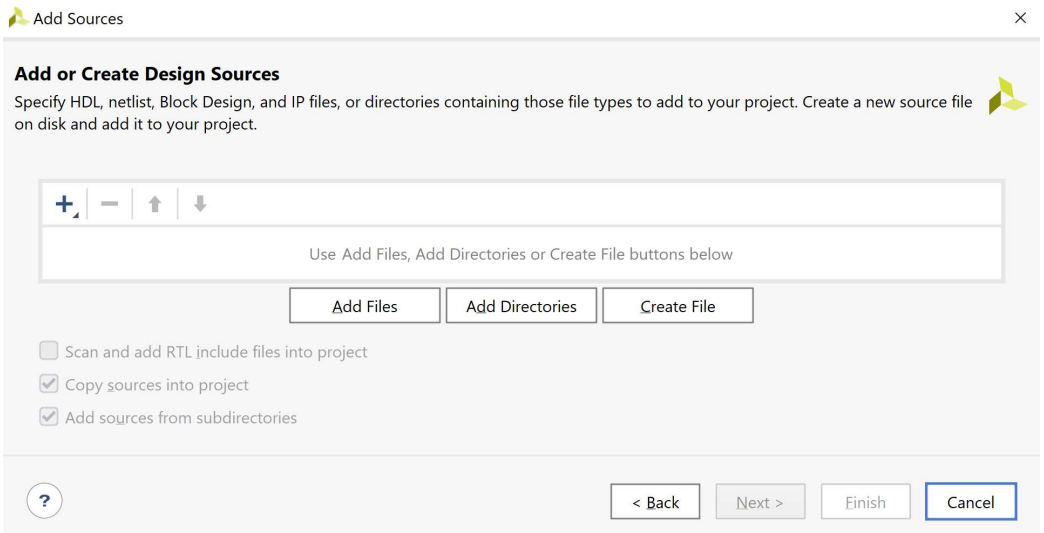
2-2. Προσθήκη υπάρχοντος αρχείου τύπου VHD στο VIVADO IDE

2-2.1. Πατήστε το **+** στο παράθυρο *Sources* του PROJECT MANAGER για να ξεκινήσετε τον wizard δημιουργίας ενός νέου αρχείου (source).

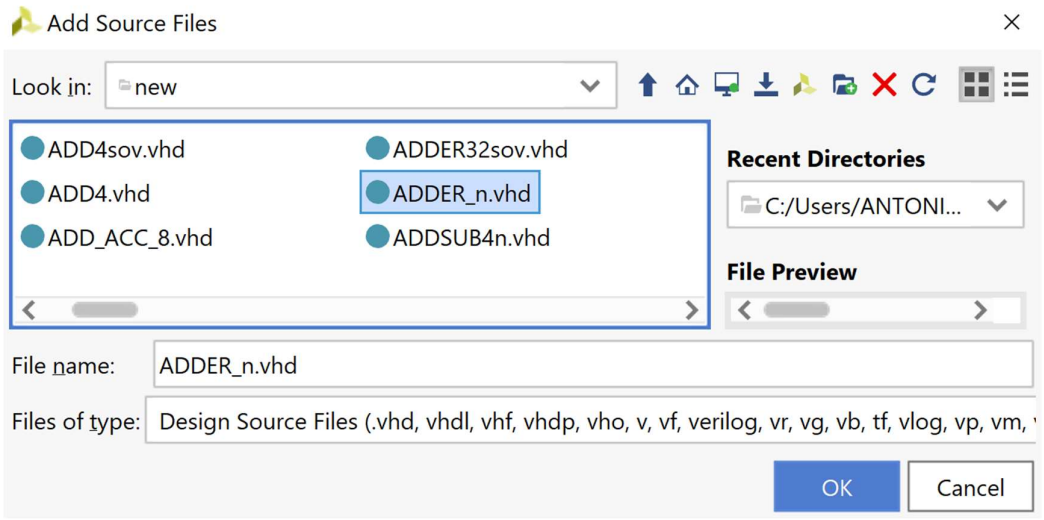
2-2.2. Στο παράθυρο διαλόγου *Add Sources* επιλέξτε το Add or create design sources, ώστε να προσθέσετε ένα υπάρχον πηγαίο αρχείο περιγραφής στο επίπεδο RTL ενός ψηφιακού κυκλώματος (ή συστήματος, ανάλογα με την πολυπλοκότητα της σχεδίασης) στη γλώσσα VHDL (design source file) τύπου VHD. Πατήστε **Next**.



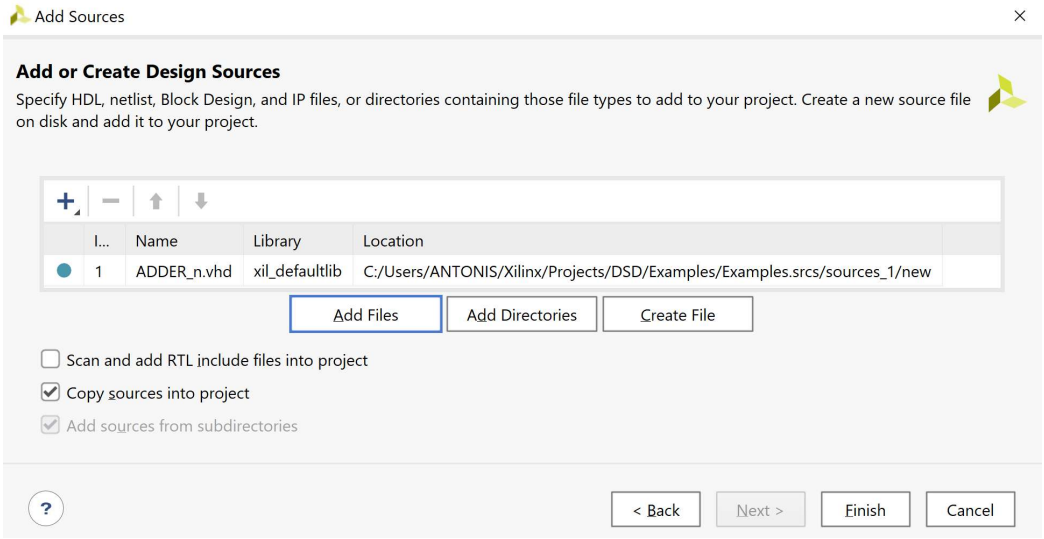
2-2.3. Στο παράθυρο διαλόγου *Add or Create Design Sources* πατήστε την επιλογή **Add Files** για την προσθήκη υπαρχόντων design source files που θα τοποθετηθούν στο ήδη καθορισμένο design source set *sources_1*.



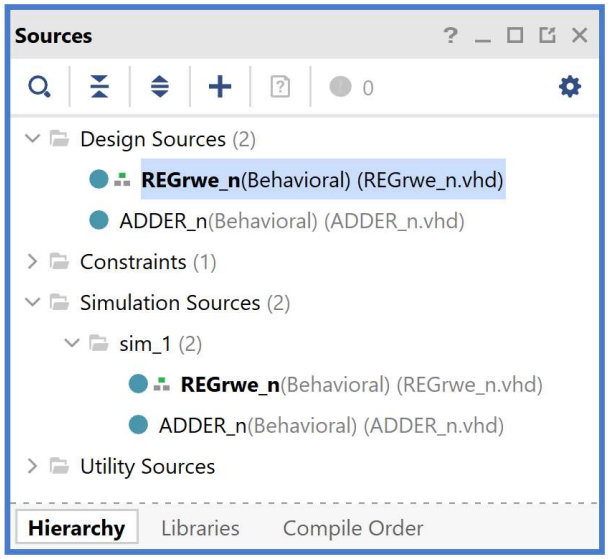
2-2.4. Στο παράθυρο διαλόγου *Add Source File* επιλέξτε το αρχείο **ADDER_n** που περιγράφει έναν παραμετροποιημένο προσημασμένο αθροιστή με κρατούμενο εξόδου Cout και υπερχείλιση OV των N bit με βάση τον κώδικα που δίδεται στη σελίδα 303 του Κεφαλαίου 4 «Γλώσσες περιγραφής υλικού – Πλήρης έκδοση» των παραδόσεων του μαθήματος. Τέλος πατήστε **OK**.



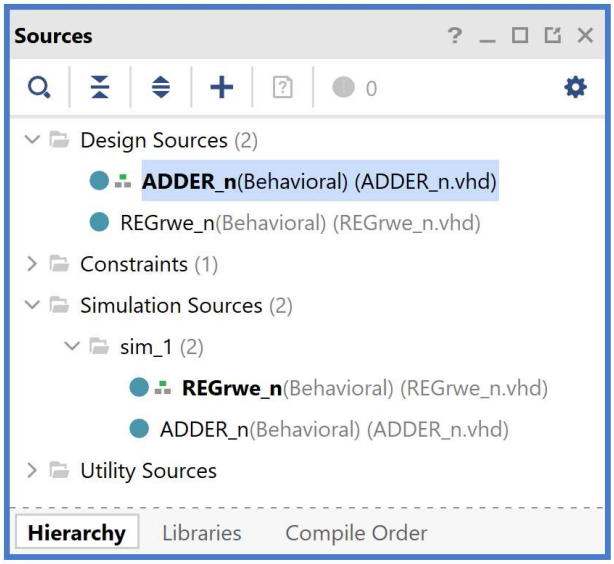
2-2.5. Επιστρέφετε στο παράθυρο διαλόγου *Add or Create Design Sources*, όπου φαίνεται ότι έχει προσδιορισθεί η θέση του αρχείου **ADDER_n.vhd** που θα αντιγραφεί στα αρχεία του project που ονομάζεται DSD_LAB_1. Βεβαιωθείτε ότι το κουτί **Copy sources into project** είναι επιλεγμένο, ώστε να επιτευχθεί η αντιγραφή του αρχείου. Πατήστε **Finish**.



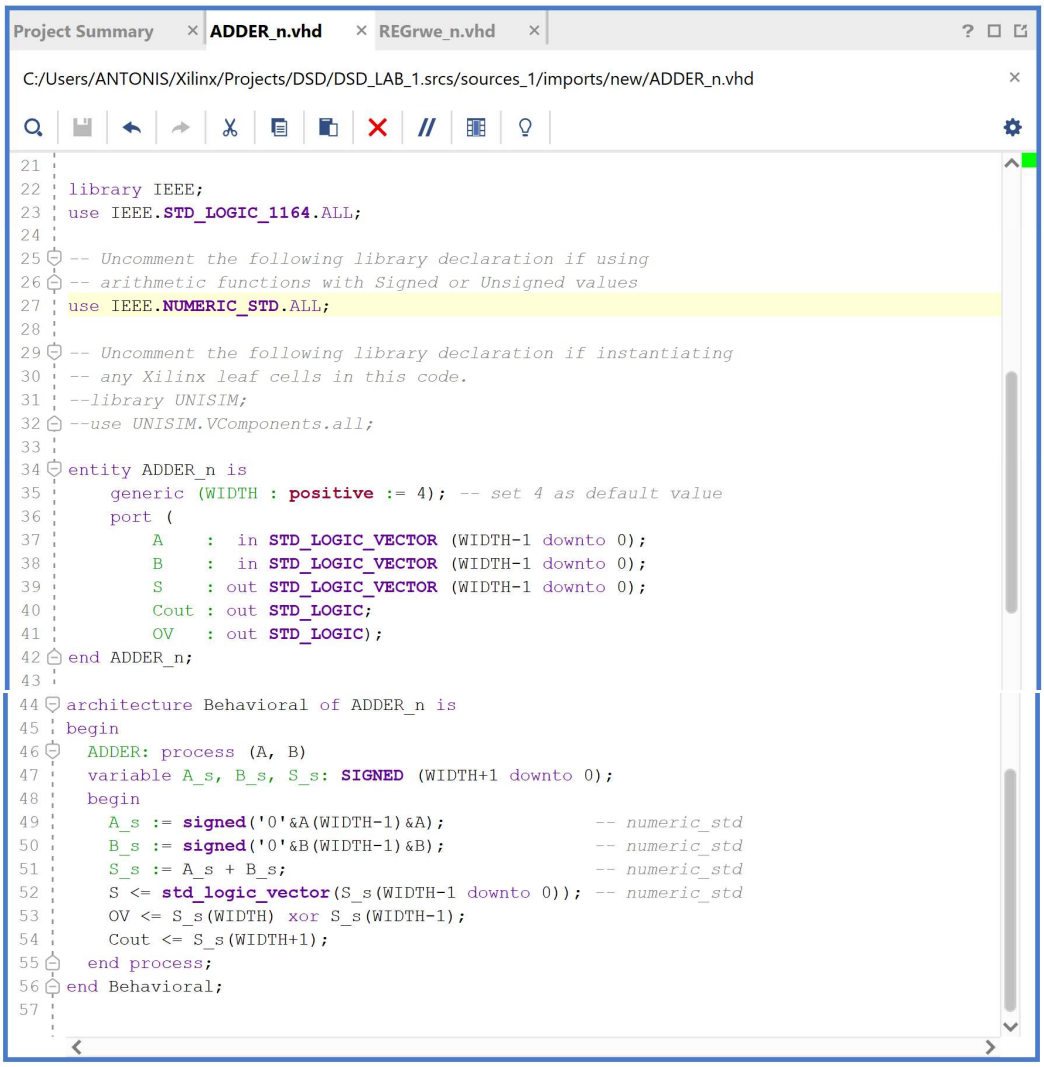
2-2.6. Επιβεβαιώστε τη δημιουργία του αρχείου **ADDER_n.vhd** στο παράθυρο *Sources* του PROJECT MANAGER (επίσης φαίνεται το όνομα της οντότητας **Adder_n** και το όνομα της αρχιτεκτονικής **Behavioral**). Η οντότητα **Adder_n** είναι αποθηκευμένη και στο design source set sources_1 και στο simulation source set sim_1 του project DSD_LAB1. Επί του παρόντος, η οντότητα **REGrwe_n** είναι ορισμένη ως η κορυφαία οντότητα της ιεραρχίας (**top**) επί της οποίας θα εκτελεσθούν στη συνέχεια τα επόμενα βήματα της σχεδίασης (RTL analysis, Synthesis, Implementation, Program and debug).



2-2.7. Για να ορισθεί η οντότητα **ADDER_n** ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design sources, θα πρέπει να κάνετε δεξί κλικ επί του ονόματος στο παράθυρο *Sources* του PROJECT MANAGER και να επιλέξετε το **Set as Top**. Η κορυφαία οντότητα **(top design source)** διαφοροποιείται από τις υπόλοιπες οντότητες του design source set *sources*_1 με το σύμβολο και τα **έντονα** (bold) γράμματα στο όνομα της οντότητας.



2-2.8. Ανοίξτε το αρχείο **ADDER_n.vhd** με διπλό κλικ στο επιλεγμένο αρχείο.

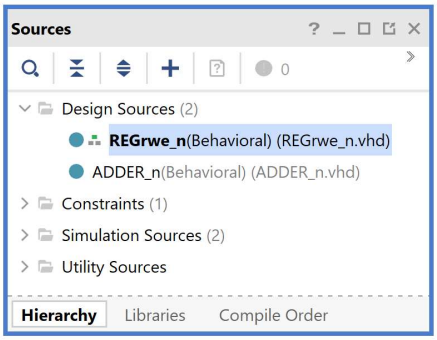


Προσέξτε ιδιαίτερα τη δήλωση του πακέτου **NUMERIC.STD** και τον τρόπο που παράγονται τα σήματα Cout και OV στους προσημασμένους αθροιστές.

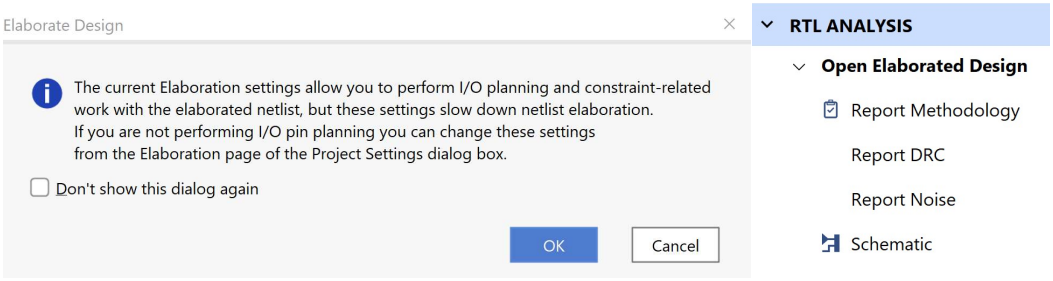
2-3. Ανάλυση του κώδικα VHDL στο επίπεδο RTL.

Τα πηγαία αρχεία περιγραφής συμπεριφοράς στη γλώσσα VHDL του καταχωρητή (**REGrwe_n.vhd**) και του αθροιστή (**ADDER_n.vhd**) αναλύονται στη συνέχεια στο **επίπεδο RTL**. Η ανάλυση περιορίζεται στη μελέτη του σχηματικού διαγράμματος στο επίπεδο RTL.

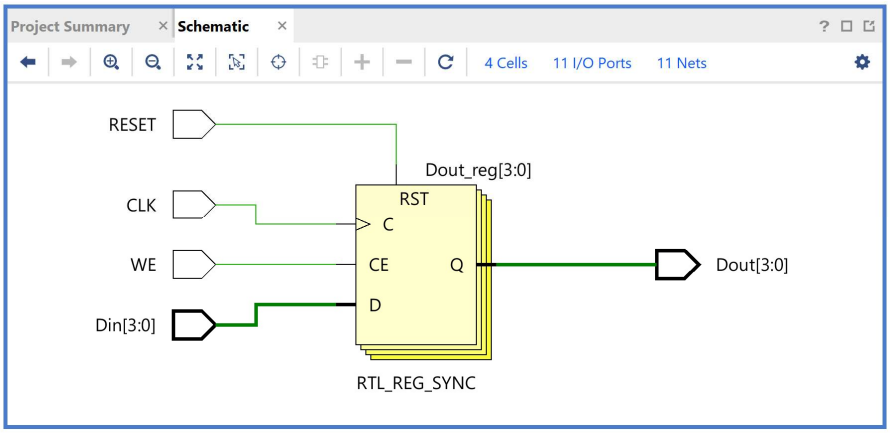
2-3.1. Αρχικά, στο παράθυρο *Sources* του PROJECT MANAGER επιλέξτε την οντότητα **REGrwe_n** ως την κορυφαία οντότητα της ιεραρχίας (**top**), κάνοντας δεξί κλικ επί του επιλεγμένου ονόματος και επιλέγοντας το **Set as Top**. Το παράθυρο *Sources* διαμορφώνεται όπως το βλέπετε δεξιά.



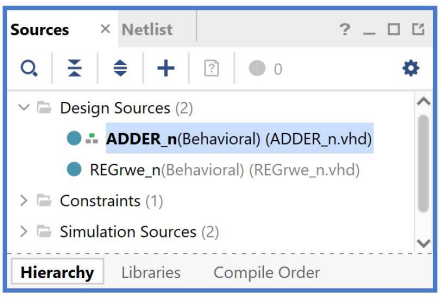
2-3.2. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Open Elaborated Design**, ώστε να εμφανισθούν οι δυνατότητες που παρέχει το εργαλείο Vivado IDE μετά την ανάλυση στο επίπεδο RTL (RTL ANALYSIS) του **behavioral (elaborated design) model** της κορυφαίας οντότητας **REGrwe_n** της ιεραρχίας (**top**). Εάν εμφανισθεί το παράθυρο προειδοποίησης *Elaborate Design*, πατήστε **OK**, ώστε να εκτελεσθεί η διαδικασία της ανάλυσης στο επίπεδο RTL.



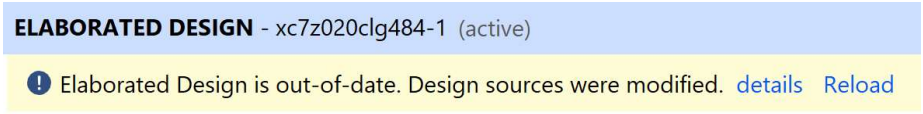
Εμφανίζεται το παράθυρο *Schematic* στη θέση του παραθύρου *Project Summary* με το σχηματικό διάγραμμα στο επίπεδο RTL του καταχωρητή των 4 bit με κοινές εισόδους CLK, RESET και EN. Ένα νέο παράθυρο *Schematic* εμφανίζεται κάθε φορά που επιλέγετε το **Schematic** στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*.



2-3.3. Στη συνέχεια, στο παράθυρο *Sources* του PROJECT MANAGER επιλέξτε την οντότητα **ADDER_n** ως την κορυφαία οντότητα της ιεραρχίας (**top**), κάνοντας δεξί κλικ επί του επιλεγμένου ονόματος και επιλέγοντας το **Set as Top**. Το παράθυρο *Sources* διαμορφώνεται όπως το βλέπετε δεξιά.

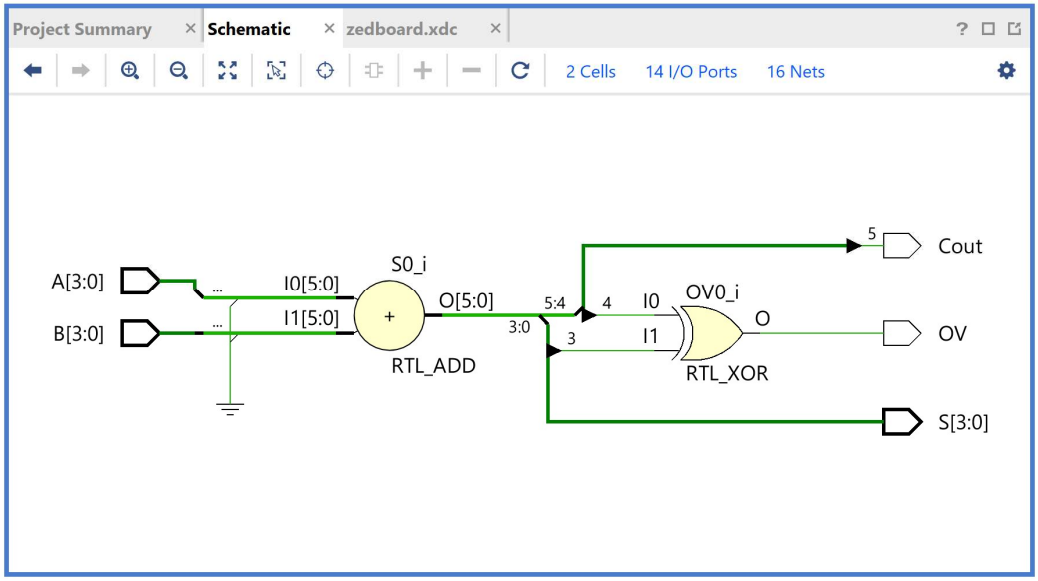


Προσοχή! Επειδή είναι ενεργή η διαδικασία της ανάλυσης στο επίπεδο RTL, θα εμφανισθεί η προειδοποίηση αλλαγής του **behavioral (elaborated design) model**.



2-3.4. Με επιλογή του **details** επιβεβαιώστε την έγκριση του αρχείου **ADDER_n.vhd**.

2-3.5. Με επιλογή του **Reload** εκτελέστε τη διαδικασία της ανάλυσης RTL της οντότητας **ADDER_n** (πατήστε **OK** σε όποιο παράθυρο προειδοποίησης εμφανισθεί). Εμφανίζεται στο νέο παράθυρο *Schematic* το σχηματικό διάγραμμα στο επίπεδο RTL του αθροιστή των 4 bit με εξόδους Cout και OV.

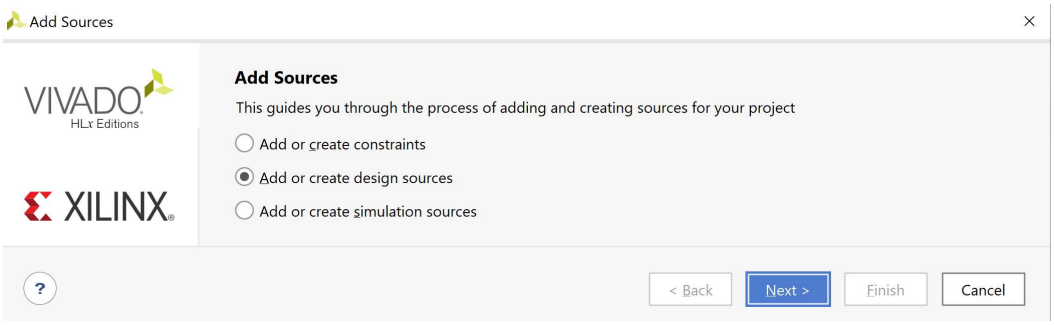


Επιβεβαιώστε το elaborated design. Συμπεριλαμβάνει έναν αθροιστή των 6 bit και μια πύλη XOR των 2 bit με έξοδο OV. Η έξοδος Cout είναι η έξοδος O[5] του αθροιστή.

2-4. Δημιουργία ιεραρχικής δομής τύπου VHD στο VIVADO IDE

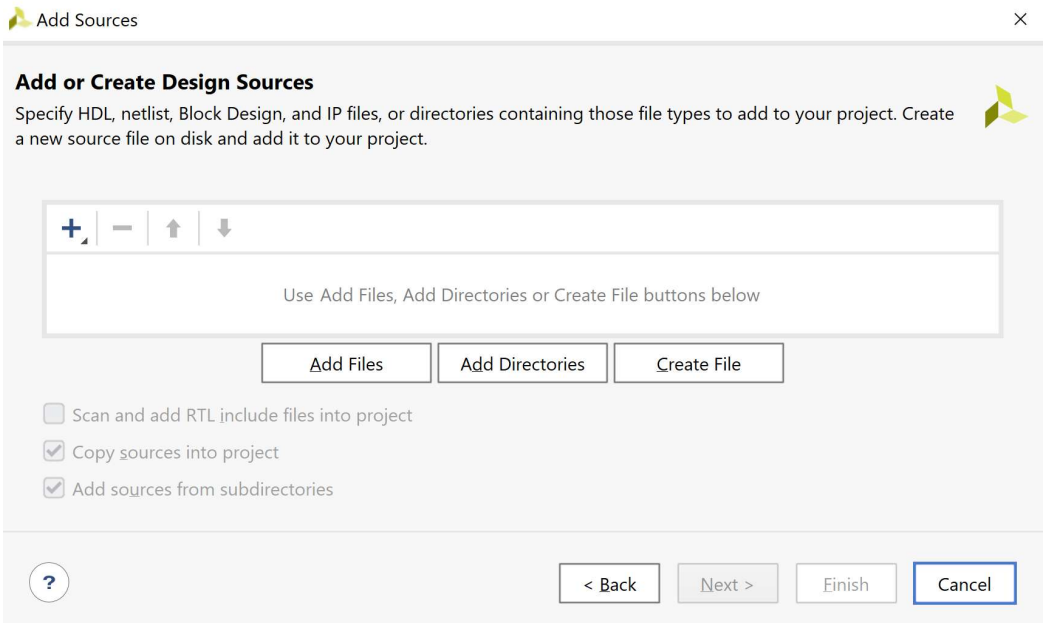
2-4.1. Πατήστε το **+** στο παράθυρο *Sources* του PROJECT MANAGER για να ξεκινήσετε τον wizard δημιουργίας της ιεραρχικής δομής ως πηγαίου αρχείου (source) του αθροιστή των 8 bit με καταχωρητές εισόδου και εξόδου (**ADDER_REG_8**) που χρησιμοποιεί ως στοιχεία (components) τις οντότητες (entities) **REGrwe_n** και **ADDER_n**, που ήδη έχετε δημιουργήσει.

2-4.2. Στο παράθυρο διαλόγου *Add Sources* επιλέξτε το Add or create design sources, ώστε να δημιουργήσετε ένα νέο πηγαίο αρχείο περιγραφής στο επίπεδο RTL ενός ψηφιακού κυκλώματος (ή συστήματος, ανάλογα με την πολυπλοκότητα της σχεδίασης) στη γλώσσα VHDL (design source file) τύπου VHD. Σε αυτό το αρχείο εισάγουμε τον κώδικα VHDL. Πατήστε **Next**.

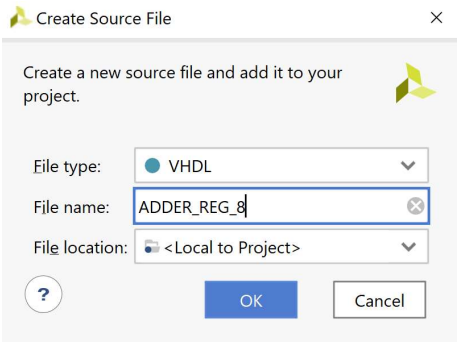


2-4.3. Στο παράθυρο διαλόγου *Add or Create Design Sources* πατήστε την επιλογή **Create File** για τη δημιουργία του νέου design source file που θα τοποθετηθεί στο ήδη καθορισμένο design source set *sources_1*.

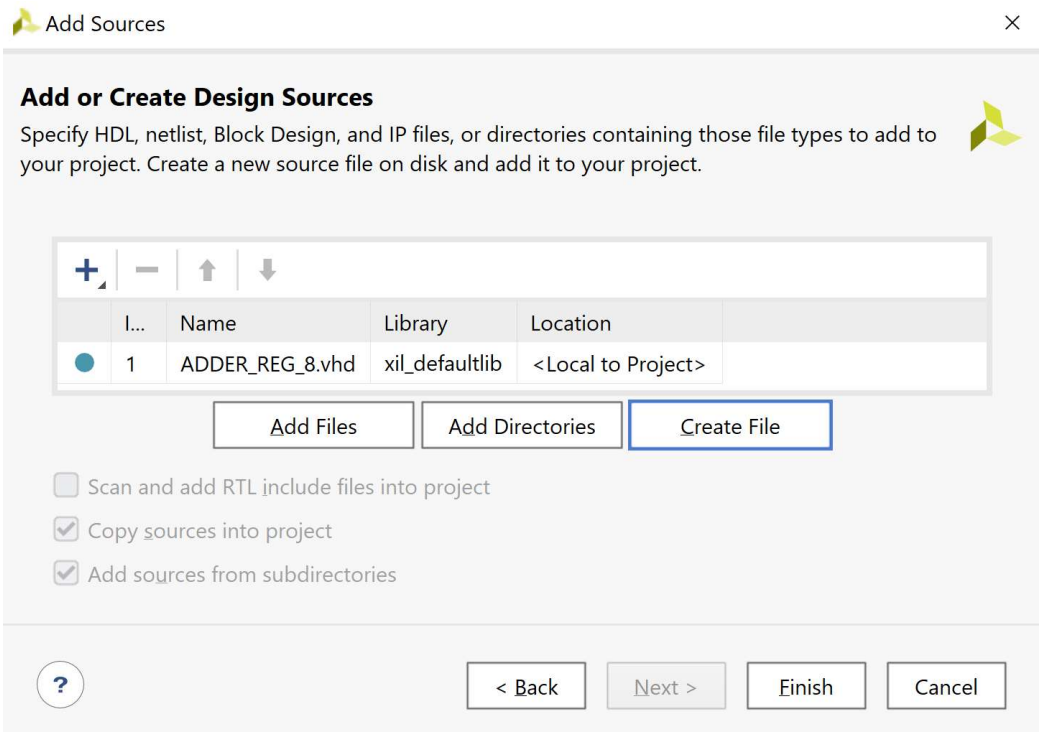
Εάν απαιτείται επιλέξτε την **VHDL** ως *Target Language* και το **Mixed** ως *Simulator Language*.



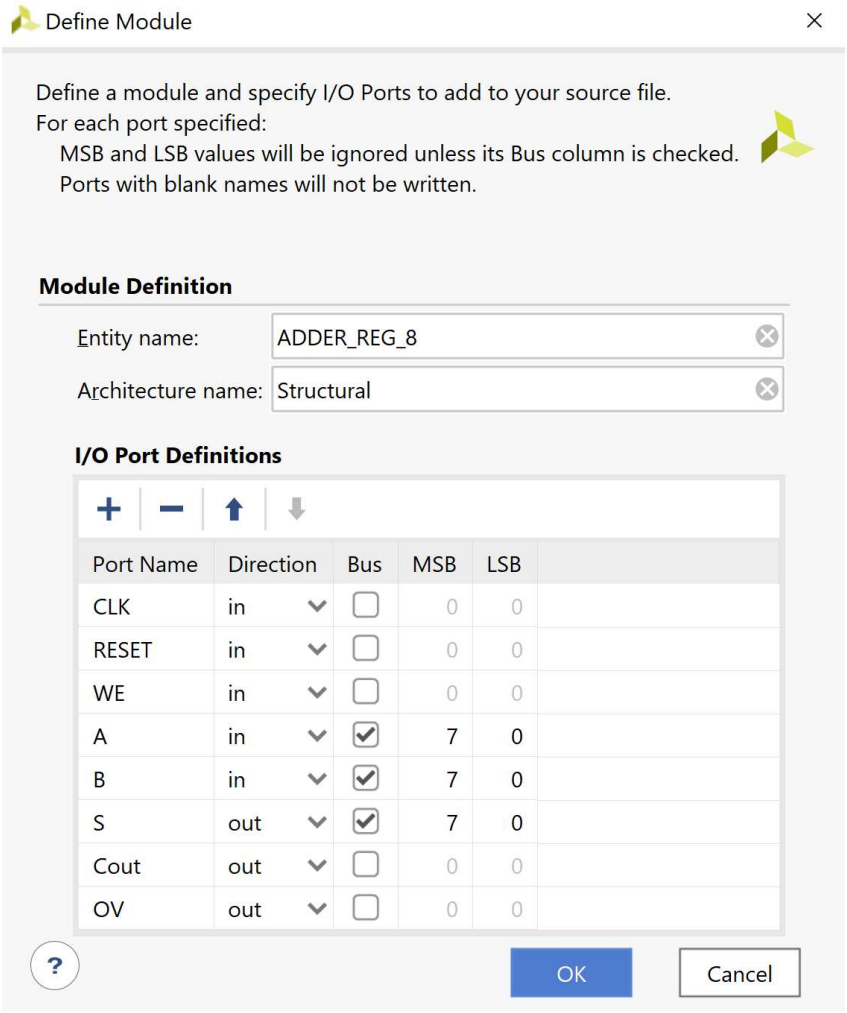
2-4.4. Στο παράθυρο διαλόγου *Create Source File* δηλώστε το όνομα του αρχείου VHD (π.χ. **ADDER_REG_8**), αφού σε αυτό το αρχείο θα περιγράψετε στη γλώσσα VHDL έναν αθροιστή των 8 bit με καταχωρητές εισόδου και εξόδου, το οποίο θεωρείται σύγχρονο ακολουθιακό κύκλωμα. Πατήστε **OK**.



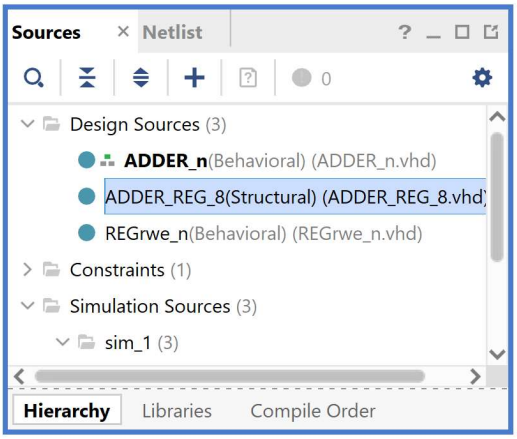
2-4.5. Επιστρέφετε στο παράθυρο διαλόγου *Add or Create Design Sources*, όπου φαίνεται ότι έχει δημιουργηθεί το αρχείο **ADDER_REG_8.vhd** και έχει συμπεριληφθεί στα αρχεία του project που ονομάζεται DSD_LAB_1. Πατήστε **Finish**.



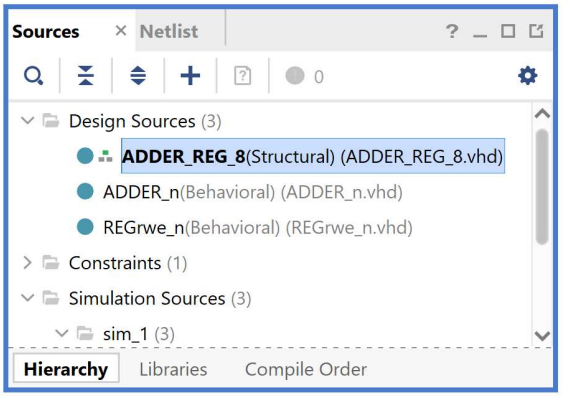
2-4.6. Εμφανίζεται το παράθυρο διαλόγου *Define Module*, όπου σας παρέχεται η δυνατότητα της δήλωσης του ονόματος της οντότητας, του ονόματος της αρχιτεκτονικής και των ports της οντότητας. Ως όνομα της οντότητας διατηρείστε το ίδιο όνομα που είχατε δώσει στο αρχείο (**ADDER_REG_8**). Επιλέξτε ως όνομα της αρχιτεκτονικής το όνομα *structural* αφού θα κάνετε περιγραφή δομής στη γλώσσα VHDL. Στη συνέχεια, προαιρετικά δηλώστε τα ports στο **I/O Port Definitions**. As υποθέσουμε ότι αρχικά δηλώνουμε τα ports του αθροιστή με καταχωρητές εισόδου και εξόδου των 8 bit που έχει εισόδους CLK, RESET, WE (για έγκριση εγγραφής στον καταχωρητή εισόδων του αθροιστή), A[7:0] και B[7:0], καθώς και εξόδους S[7:0], Cout και OV (όλες registered). Εισάγετε το όνομα του σήματος ή της αρτηρίας στο port name. Για τις αρτηρίες επιλέξτε επιπλέον το Bus και ορίστε την τιμή του MSB και του LSB. Πατήστε το **+** για δήλωση επιπλέον ports με τον ίδιο τρόπο. Τέλος, πατήστε **OK**.



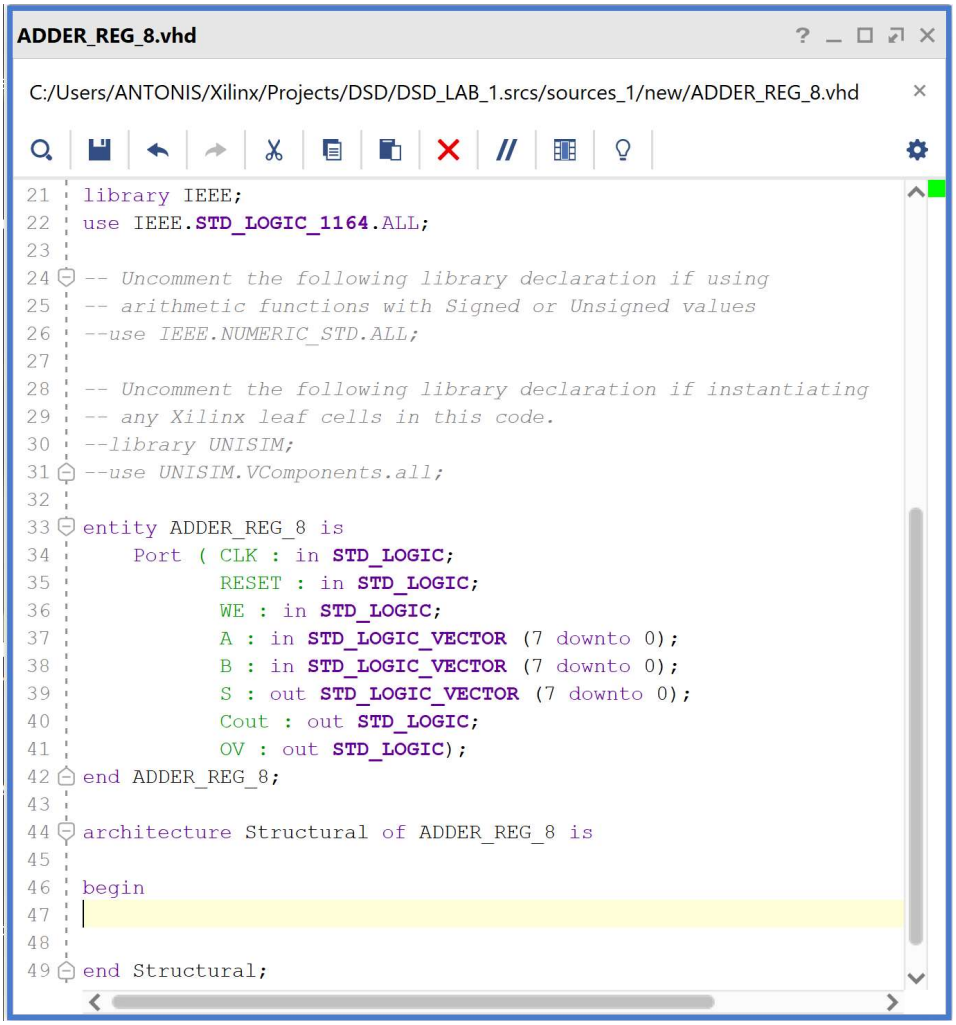
2-4.7. Επιβεβαιώστε τη δημιουργία του αρχείου **ADDER_REG_8.vhd** στο παράθυρο *Sources* του PROJECT MANAGER (επίσης φαίνεται το όνομα της οντότητας **ADDER_REG_8** και το όνομα της αρχιτεκτονικής **Structural**). Η οντότητα **ADDER_REG_8** αποθηκεύεται και στο design source set *sources_1* και στο simulation source set *sim_1* του project DSD_LAB1. Παρατηρείστε ότι η οντότητα **ADDER_n** παραμένει ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design sources, επί του παρόντος.



2-4.8. Για να ορισθεί η οντότητα **ADDER_REG_8** ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design sources, θα πρέπει να κάνετε δεξί κλικ επί του ονόματος στο παράθυρο *Sources* του PROJECT MANAGER και να επιλέξετε το **Set as Top**. Σε αυτό το σημείο, οι 3 οντότητες (τα 3 design sources) είναι ανεξάρτητα μεταξύ τους.

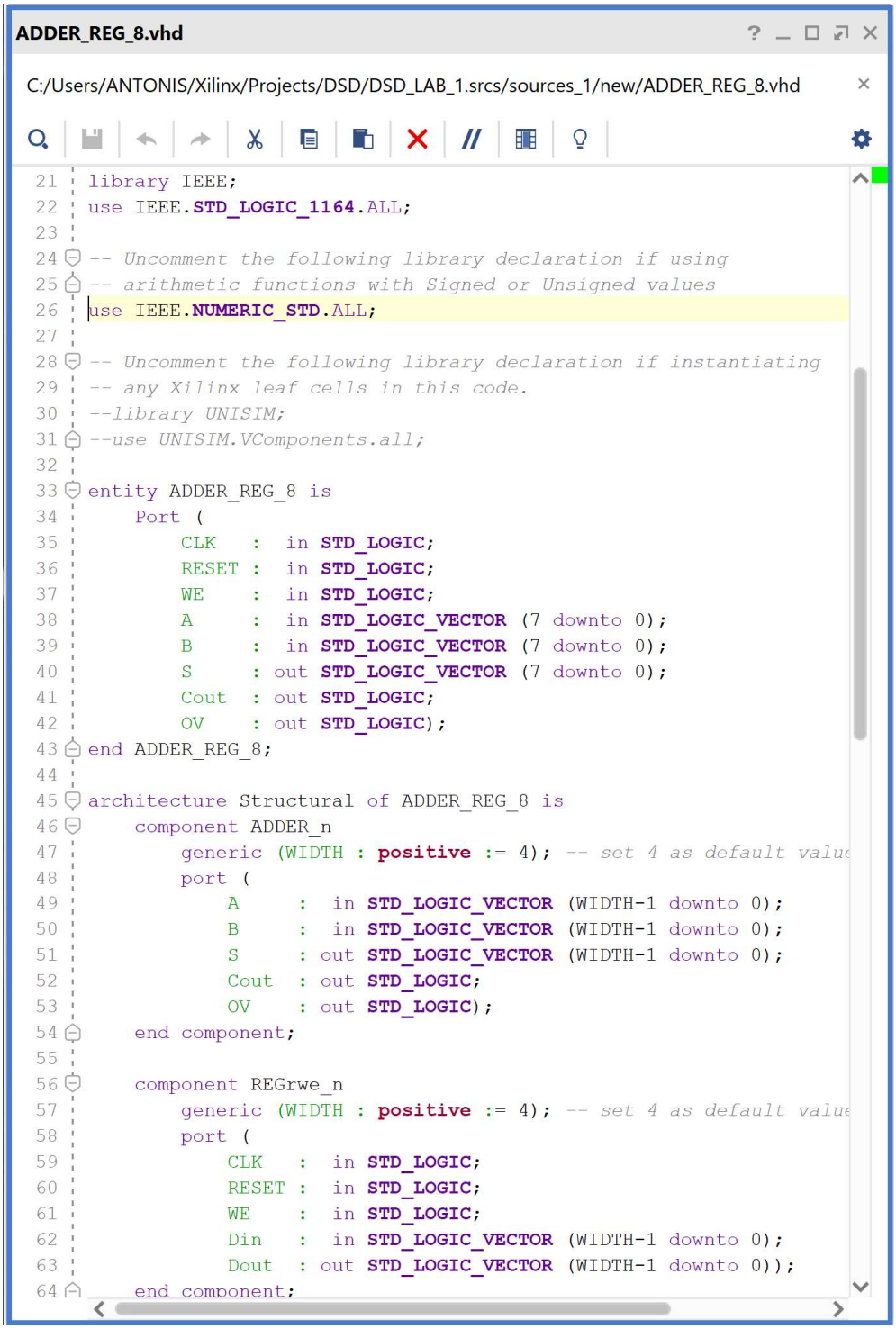


2-4.9. Ανοίξτε το πηγαίο αρχείο **ADDER_REG_8.vhd** με διπλό κλικ στο επιλεγμένο αρχείο. Τα ports της οντότητας έχουν ήδη ορισθεί, αλλά λείπει ο ορισμός της αρχιτεκτονικής της οντότητας. Μπορούμε να αυξήσουμε το μέγεθος του παραθύρου *ADDER_REG_8.vhd*, είτε σε πλάτος με την επιλογή του **maximize**, είτε με τη δημιουργία νέου παραθύρου με την επιλογή του **float**. Επιλέξτε το **float**.



Παρατηρείστε ότι το *use IEEE.NUMERIC_STD.ALL* εμφανίζεται σε γραμμή σχολίου, που πρέπει να αφαιρέσετε με την επιλογή του **Toggle Line Comments**. //

2-4.10. Συμπληρώστε την αρχιτεκτονική της οντότητας **ADDER_REG_8**. Στο τμήμα δηλώσεων της αρχιτεκτονικής (πριν το begin): δηλώστε τα components **ADDER_n** και **REGrwe_n** (με αντιγραφή των ports από τα αντίστοιχα entities) και τις εσωτερικές αρτηρίες **A_in[7:0]**, **B_in[7:0]**, **S_in[7:0]** και **FlagsI_in[1:0]**, **FlagsO_in[1:0]**. Στο σώμα της αρχιτεκτονικής (μετά το begin) περιγράψτε τις διασυνδέσεις των 5 components που απαρτίζουν την οντότητα **ADDER_REG_8** με τις **5 αντίστοιχες ταυτόχρονες εντολές στοιχείων**, που διαθέτουν και generic map και port map. Τέλος πατήστε **Save File**.

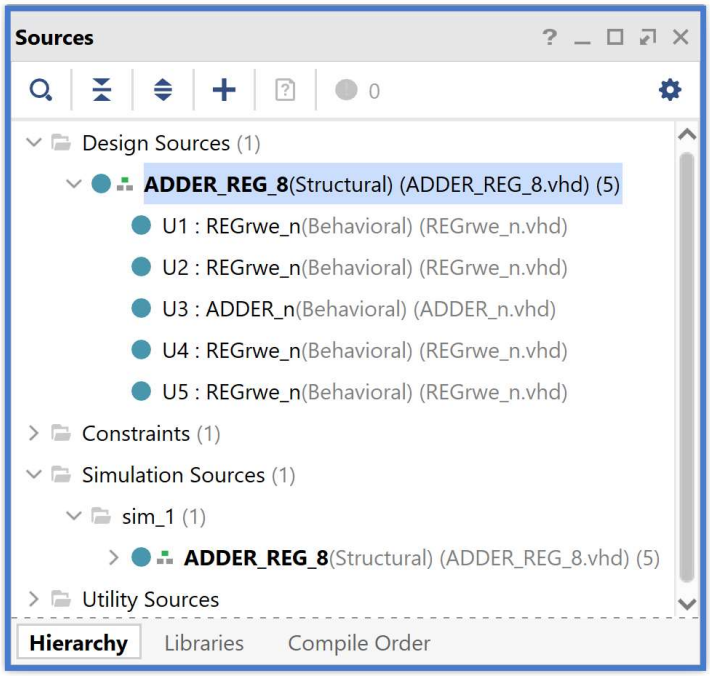


ADDER_REG_8.vhd

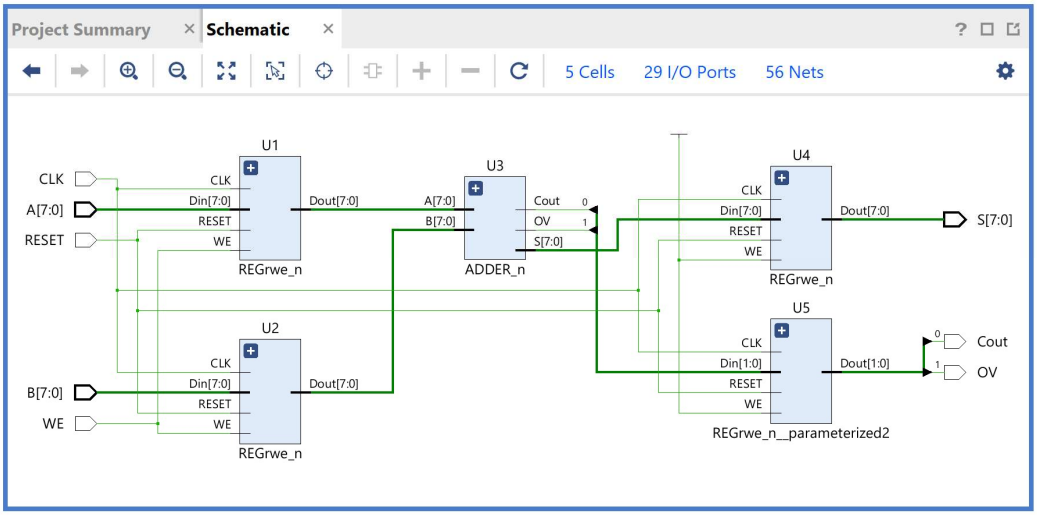
C:/Users/ANTONIS/Xilinx/Projects/DSD/DSD_LAB_1.srcs/sources_1/new/ADDER_REG_8.vhd

```
66      signal  A_in      : STD_LOGIC_VECTOR (7 downto 0);
67      signal  B_in      : STD_LOGIC_VECTOR (7 downto 0);
68      signal  S_in      : STD_LOGIC_VECTOR (7 downto 0);
69
70      signal  FlagsI_in : STD_LOGIC_VECTOR (1 downto 0);
71      signal  FlagsO_in : STD_LOGIC_VECTOR (1 downto 0);
72
73  begin
74
75      U1: REGrwe_n
76          generic map (WIDTH => 8)
77          port map (
78              CLK   => CLK,
79              RESET => RESET,
80              WE    => WE,
81              Din   => A,
82              Dout  => A_in);
83
84      U2: REGrwe_n
85          generic map (WIDTH => 8)
86          port map (
87              CLK   => CLK,
88              RESET => RESET,
89              WE    => WE,
90              Din   => B,
91              Dout  => B_in);
92
93      U3: ADDER_n
94          generic map (WIDTH => 8)
95          port map (
96              A     => A_in,
97              B     => B_in,
98              S     => S_in,
99              Cout  => FlagsI_in(0),
100             OV    => FlagsI_in(1));
101
102     U4: REGrwe_n
103         generic map (WIDTH => 8)
104         port map (
105             CLK   => CLK,
106             RESET => RESET,
107             WE    => '1',
108             Din   => S_in,
109             Dout  => S);
110
111     U5: REGrwe_n
112         generic map (WIDTH => 2)
113         port map (
114             CLK   => CLK,
115             RESET => RESET,
116             WE    => '1',
117             Din   => FlagsI_in,
118             Dout  => FlagsO_in);
119
120     Cout <= FlagsO_in(0);
121     OV   <= FlagsO_in(1);
122
123 end Structural;
```

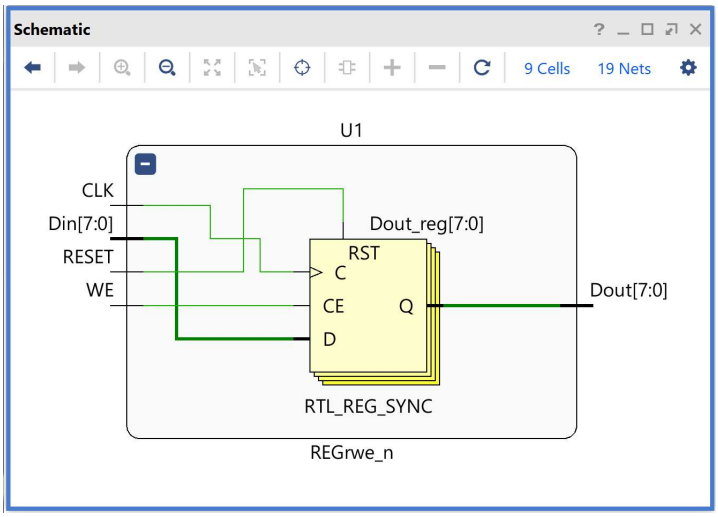
2-4.11. Παρατηρείστε το παράθυρο *Sources*, όπου έχει δημιουργηθεί αυτόματα η ιεραρχική δομή της οντότητας **ADDER_REG_8**. Επιλέξτε το επίπεδο της ιεραρχίας που επιθυμείτε να βλέπετε με κατάλληλη επιλογή πάνω στα > και ∨ (στην εικόνα φαίνονται δύο διαφορετικές επιλογές). Οι οντότητες **REGrwe_n** και **ADDER_n** έχουν ενταχθεί στην ιεραρχική δομή και δεν είναι πλέον ανεξάρτητες.

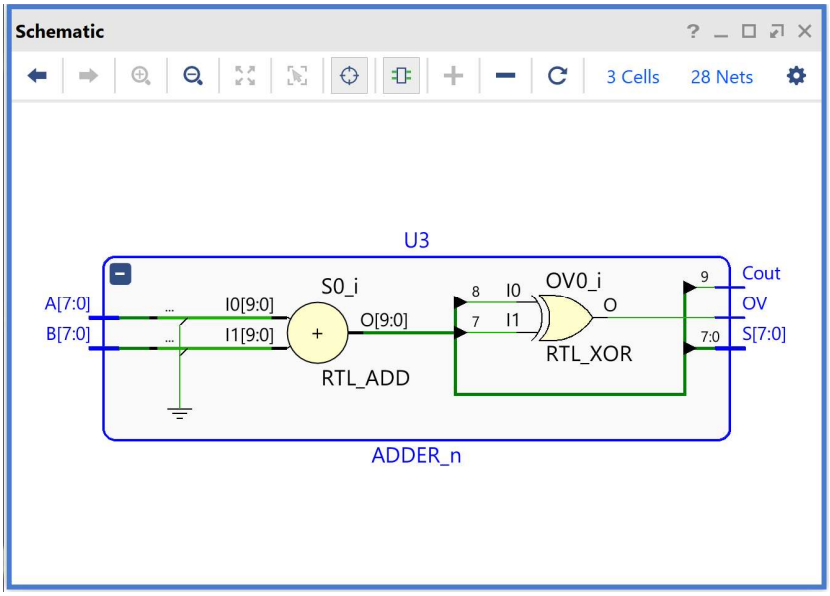


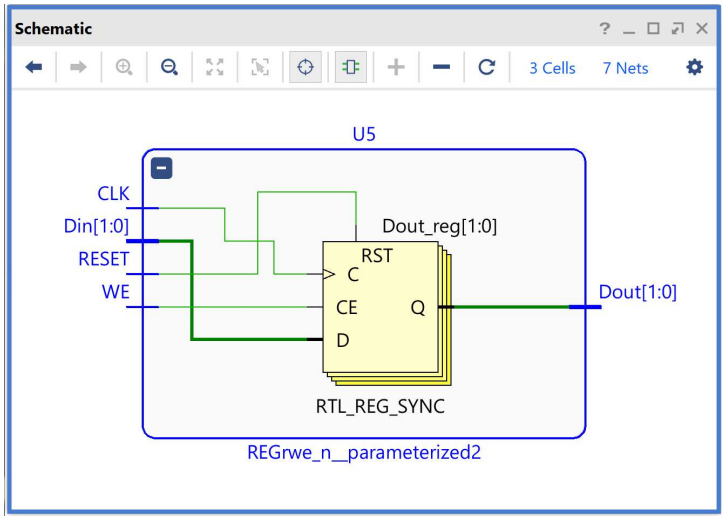
2-4.12. Με επιλογή του **Reload** εκτελέστε τη διαδικασία της ανάλυσης RTL της οντότητας **ADDER_REG_8**. Εμφανίζεται στο νέο παράθυρο *Schematic* το σχηματικό διάγραμμα στο επίπεδο RTL του αθροιστή των 8 bit με καταχωρητές εισόδου και εξόδου (ADDER_REG_8) που χρησιμοποιεί ως στοιχεία (components) τις οντότητες (entities) **REGrwe_n (U1, U2, U4, U5)** και **ADDER_n (U3)**, που ήδη έχετε δημιουργήσει.



Επιβεβαιώστε το elaborated design στο πιο κάτω ιεραρχικά επίπεδο με διπλό κλικ πάνω στα επιλεγμένα components U1 – U5. Τα components U1, U2 και U4 είναι ίδια.


Schematic
9 Cells
19 Nets
U1
CLK
Din[7:0]
RESET
WE
Dout_reg[7:0]
RST
C
CE
Q
D
Dout[7:0]
RTL_REG_SYNC
REGrwe_n


Schematic
3 Cells
28 Nets
U3
A[7:0]
B[7:0]
I0[9:0]
I1[9:0]
S0_i
O[9:0]
RTL_ADD
8
I0
7
I1
OV0_i
O
RTL_XOR
9
7:0
Cout
OV
S[7:0]
ADDER_n


Schematic
3 Cells
7 Nets
U5
CLK
Din[1:0]
RESET
WE
Dout_reg[1:0]
RST
C
CE
Q
D
Dout[1:0]
RTL_REG_SYNC
REGrwe_n__parameterized2

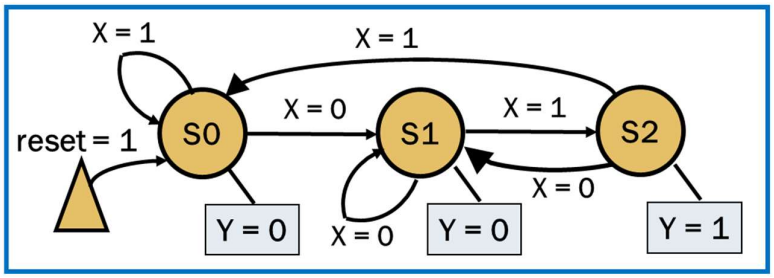
2-5. Δημιουργία του πηγαίου αρχείου PATTERN_FSM.VHD στο VIVADO IDE

που περιγράφει στη γλώσσα VHDL τον ανιχνευτή ακολουθίας 2 διαδοχικών bit ως μηχανή FSM τύπου Moor με βάση τον κώδικα που δίδεται στις σελίδες 324 – 330 του Κεφαλαίου 4 «Γλώσσες περιγραφής υλικού – Πλήρης έκδοση» των παραδόσεων του μαθήματος.

Ο ανιχνευτής ακολουθίας, πέραν των εισόδων CLK και RESET, έχει μία σειριακή είσοδο X και μία σειριακή έξοδο Y, που γίνεται 1 κάθε φορά που τα δύο τελευταία διαδοχικά bit του X είναι 01. Η συγκεκριμένη μηχανή FSM έχει τρεις καταστάσεις τύπου Moore ως εξής:

- S0 = αρχική κατάσταση, δεν έχει ανιχνευθεί κανένα ψηφίο, Y = 0,
- S1 = έχει ανιχνευθεί στην είσοδο X ένα bit 0, Y = 0,
- S2 = έχουν ανιχνευθεί στην είσοδο X δύο διαδοχικά bit 01, Y = 1.

Ο ανιχνευτής ακολουθίας υλοποιεί το ακόλουθο διάγραμμα μεταβολής κατάστασης:



2-5.1. Πατήστε το **+** στο παράθυρο *Sources* του PROJECT MANAGER για να ξεκινήσετε τον wizard δημιουργίας του πηγαίου αρχείου (source) του ανιχνευτή **PATTERN_FSM**.

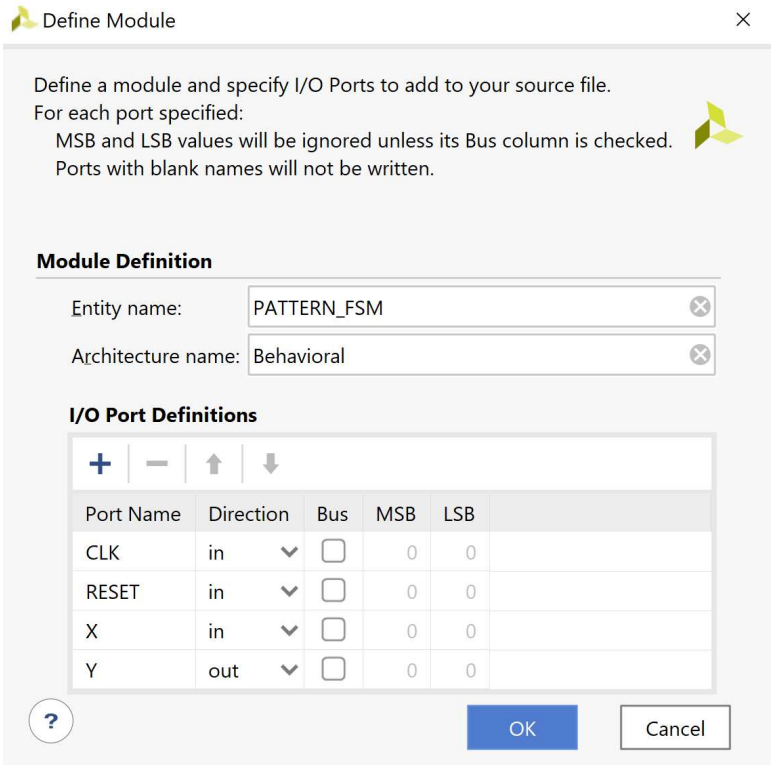
2-5.2. Στο παράθυρο διαλόγου *Add Sources* επιλέξτε το Add or create design sources και πατήστε **Next**.

2-5.3. Στο παράθυρο διαλόγου *Add or Create Design Sources* πατήστε την επιλογή **Create File** για τη δημιουργία του νέου design source file που θα τοποθετηθεί στο ήδη καθορισμένο design source set *sources_1*.

2-5.4. Στο παράθυρο διαλόγου *Create Source File* δηλώστε το όνομα του αρχείου VHD (π.χ. **PATTERN_FSM**) και πατήστε **OK**.

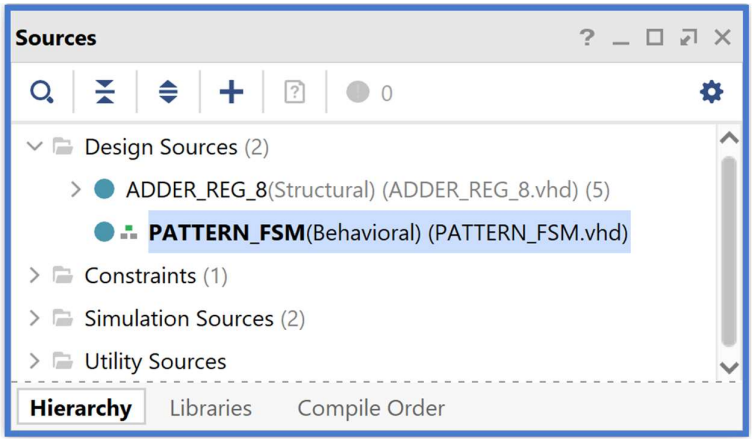
2-5.5. Επιστρέφετε στο παράθυρο διαλόγου *Add or Create Design Sources*, όπου φαίνεται ότι έχει δημιουργηθεί το αρχείο **PATTERN_FSM.vhd** και έχει συμπεριληφθεί στα αρχεία του project που ονομάζεται DSD_LAB_1. Πατήστε **Finish**.

2-5.6. Εμφανίζεται το παράθυρο διαλόγου *Define Module*, όπου σας παρέχεται η δυνατότητα της δήλωσης του ονόματος της οντότητας, του ονόματος της αρχιτεκτονικής και των ports της οντότητας. Ως όνομα της οντότητας διατηρείστε το ίδιο όνομα που είχατε δώσει στο αρχείο (**PATTERN_FSM**). Επιλέξτε ως όνομα της αρχιτεκτονικής το όνομα *behavioral* αφού θα κάνετε περιγραφή συμπεριφοράς στη γλώσσα VHDL. Στη συνέχεια, προαιρετικά δηλώστε τα ports στο **I/O Port Definitions** (**CLK**, **RESET**, **X**, **Y**). Τέλος, πατήστε **OK**.



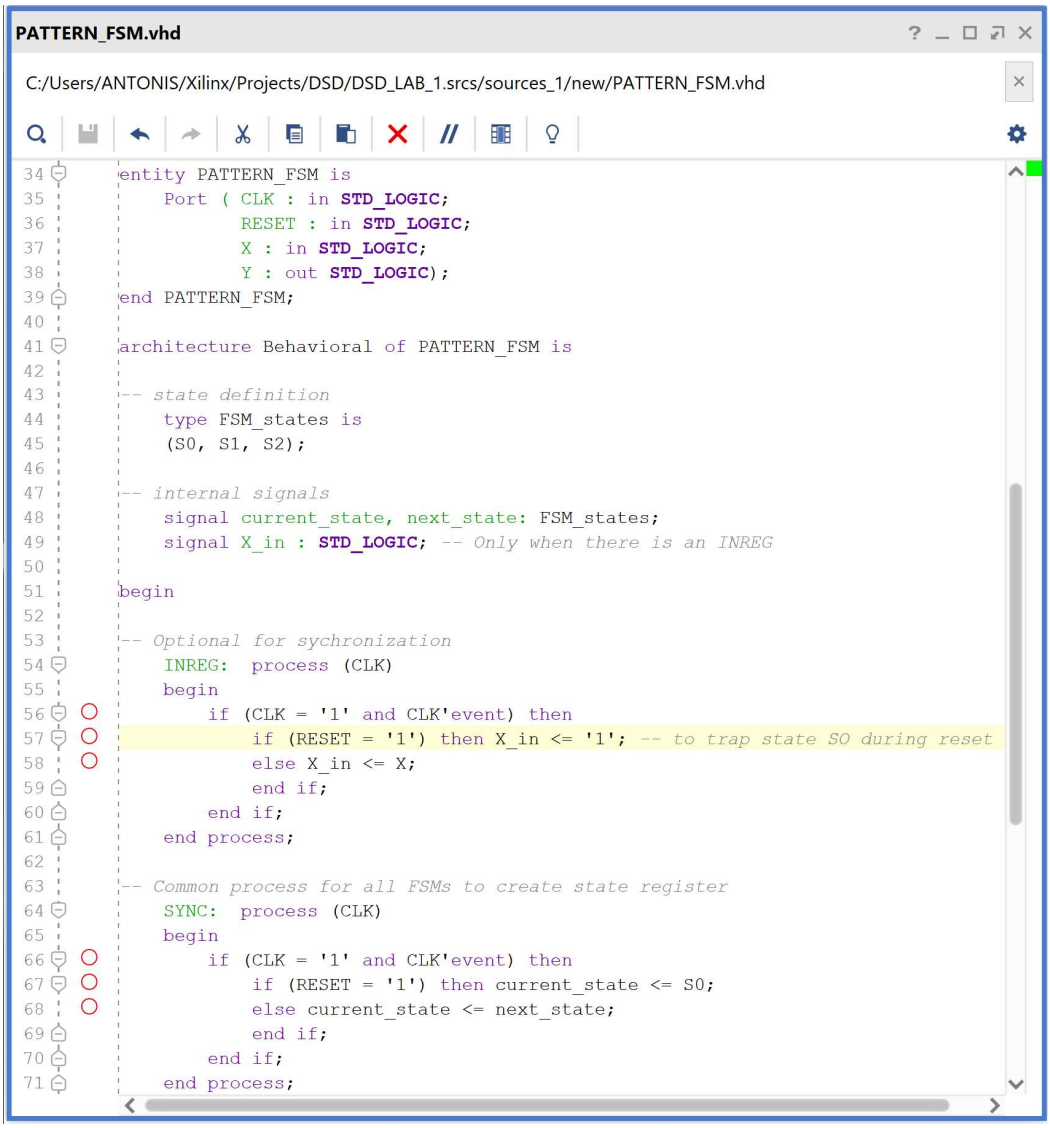
2-5.7. Επιβεβαιώστε τη δημιουργία του αρχείου **PATTERN_FSM.vhd** στο παράθυρο *Sources* του PROJECT MANAGER (επίσης φαίνεται το όνομα της οντότητας **PATTERN_FSM** και το όνομα της αρχιτεκτονικής **Behavioral**). Η οντότητα αυτή είναι αποθηκευμένη και στο design source set *sources_1* και στο simulation source set *sim_1* του project DSD_LAB1. Παρατηρείστε ότι η οντότητα **ADDER_REG_8** παραμένει ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design sources, επί του παρόντος.

2-5.8. Για να ορισθεί η οντότητα **PATTERN_FSM** ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design sources, θα πρέπει να κάνετε δεξί κλικ επί του ονόματος στο παράθυρο *Sources* του PROJECT MANAGER και να επιλέξετε το **Set as Top**.



2-5.9. Ανοίξτε το πηγαίο αρχείο **PATTERN_FSM.vhd** με διπλό κλικ στο επιλεγμένο αρχείο. Τα ports της οντότητας έχουν ήδη ορισθεί, αλλά λείπει ο ορισμός της αρχιτεκτονικής της οντότητας.

2-5.10. Συμπληρώστε την αρχιτεκτονική της οντότητας **PATTERN_FSM** με βάση τον κώδικα που δίδεται στις σελίδες 324 – 330 του Κεφαλαίου 4 «Γλώσσες περιγραφής υλικού – Πλήρης έκδοση» των παραδόσεων του μαθήματος. Οι καταστάσεις FSM_states περιγράφονται με τύπο απαρίθμησης ως S0, S1 και S2. Η είσοδος X αρχικά αποθηκεύεται σε D Flip-Flop που **τοποθετείται στο 1** με το σήμα RESET, ώστε να παγιδεύεται η μηχανή στην κατάσταση S0, όσο το σήμα RESET είναι ενεργό και μέχρι να εμφανιστεί η πρώτη είσοδος στην έξοδο του D Flip-Flop (X_in). Η μηχανή FSM περιγράφεται με δύο process: ένα σύγχρονο process (SYNC), που περιγράφει τον καταχωρητή κατάστασης που αρχικοποιείται στην κατάσταση S0 με το σήμα RESET, και ένα ασύγχρονο process (ASYNC), που περιγράφει τη λογική επόμενης κατάστασης και τη λογική εξόδου σύμφωνα με το διάγραμμα μεταβολής κατάστασης και υποστηρίζει ασφαλή λειτουργία στην περίπτωση βλάβης (fail-safe). Τέλος πατήστε **Save File**.

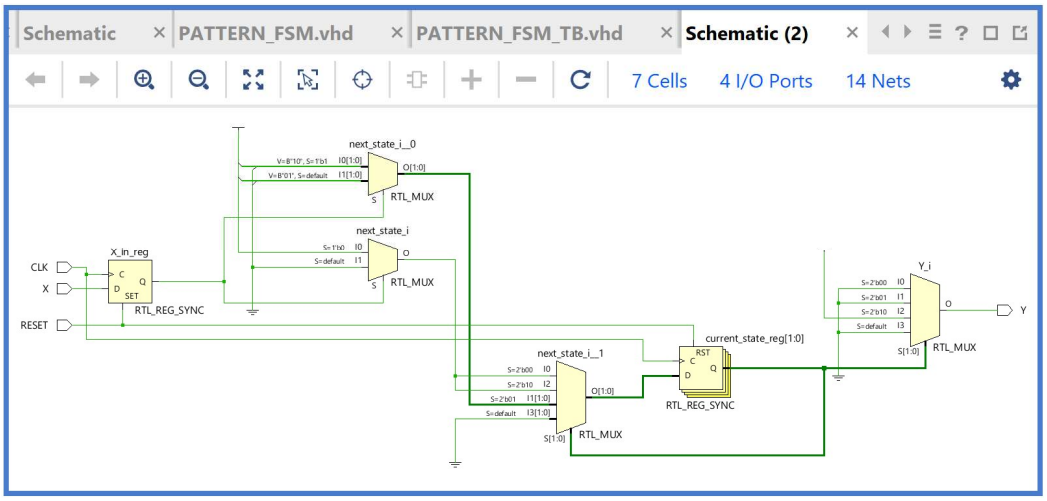


PATTERN_FSM.vhd

C:/Users/ANTONIS/Xilinx/Projects/DSD/DSD_LAB_1.srcs/sources_1/new/PATTERN_FSM.vhd

```
73      -- Process to create next state logic and output logic
74          ASYNC: process (current_state, X_in)   -- Moore
75          begin
76          -- FSM next state and output initialization
77              next_state <= S0;
78              Y <= '0';
79              case current_state is
80                  when S0 =>
81                      if (X_in = '0') then next_state <= S1;
82                      else next_state <= S0;
83                      end if;
84                  when S1 =>
85                      if (X_in = '1') then next_state <= S2;
86                      else next_state <= S1;
87                      end if;
88                  when S2 => Y <= '1';
89                      if (X_in = '0') then next_state <= S1;
90                      else next_state <= S0;
91                      end if;
92                  -- fail-safe behavior
93                  when others => next_state <= S0;
94              end case;
95          end process;
96
97      end Behavioral;
```

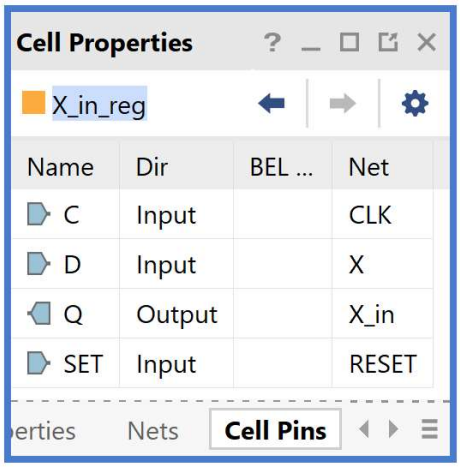
2-5.11. Με επιλογή του **Reload** εκτελέστε τη διαδικασία της ανάλυσης RTL του νέου πηγαίου αρχείου **PATTERN_FSM.vhd**. Εμφανίζεται στο νέο παράθυρο *Schematic* το σχηματικό διάγραμμα στο επίπεδο RTL του ανιχνευτή ακολουθίας 2 διαδοχικών bit ως μηχανή τύπου Moore.

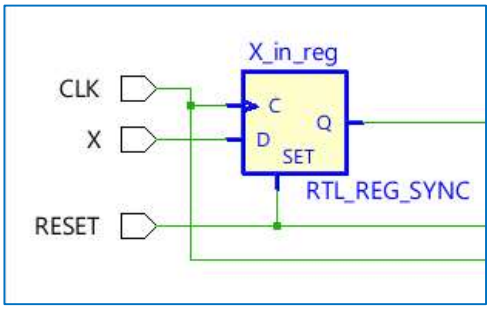


Επιβεβαιώστε το elaborated design. Μελετήστε την πληροφορία που παρέχεται στο παράθυρο *Cell Properties*, κάτω από το παράθυρο *Sources*. Αναλύστε χωριστά κάθε ένα από τα cells που φαίνονται παράθυρο schematic, με κατάλληλη επιλογή στα leaf cells είτε του παράθυρου *Netlist*, είτε του παραθύρου schematic. Παρατηρείστε ότι οι εντολές *if* και *case* της γλώσσας VHDL, που περιγράφουν συνδυαστική λογική, υλοποιούνται με πολυπλέκτες στο σχηματικό διάγραμμα στο επίπεδο RTL.

D Flip-Flop εισόδου X (X_in_reg):

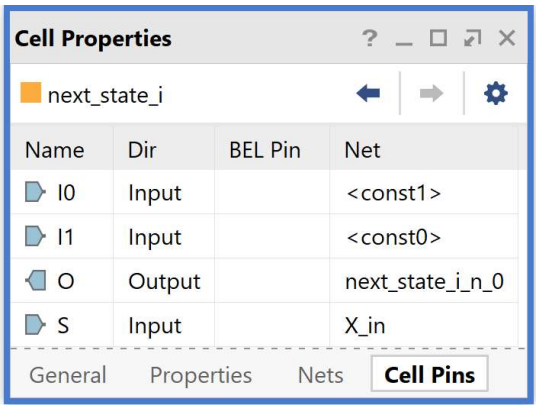
```
if (RESET = '1') then X_in <= '1'; else X_in <= X;
```

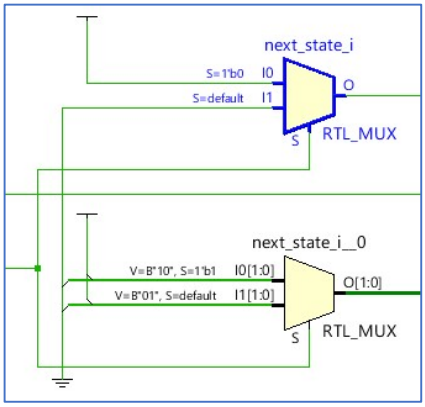




Πολυπλέκτης 2 σε 1 λογικής NOT (next_state_i):

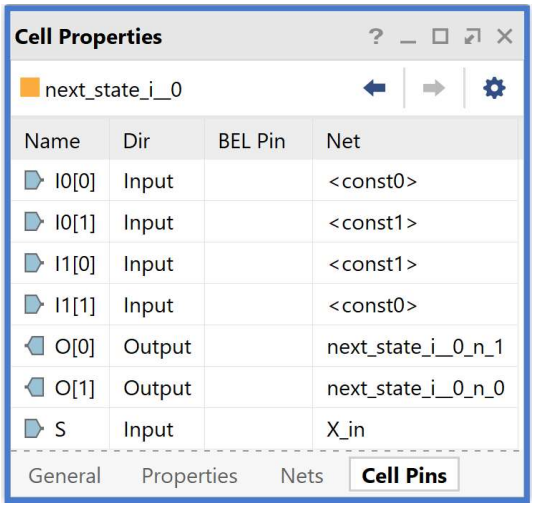
```
next_state_i_n_0 = not X_in
```

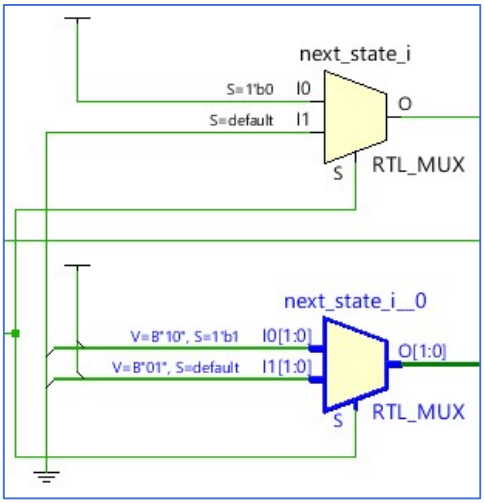




Πολυπλέκτης 2 σε 1 λογικής επόμενης κατάστασης (next_state_i_0):

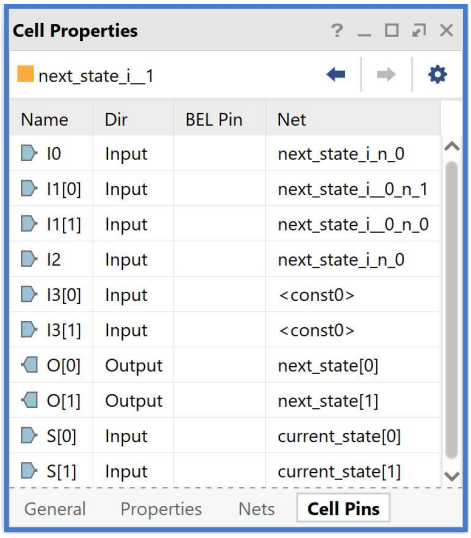
```
If (X_in = '1') then next_state_i_0 <= S2 = "10"
                else next_state_i_0 <= S1 = "01"
```

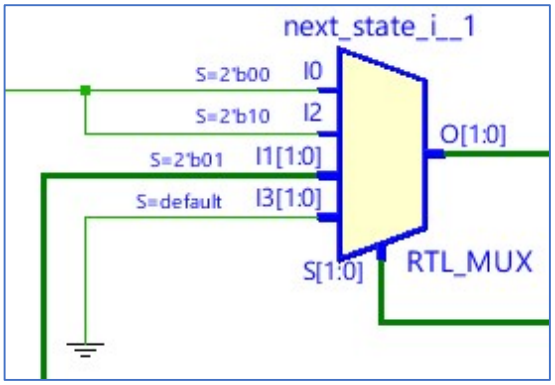




Πολυπλέκτης 4 σε 1 λογικής επόμενης κατάστασης (**next_state_i__1**):

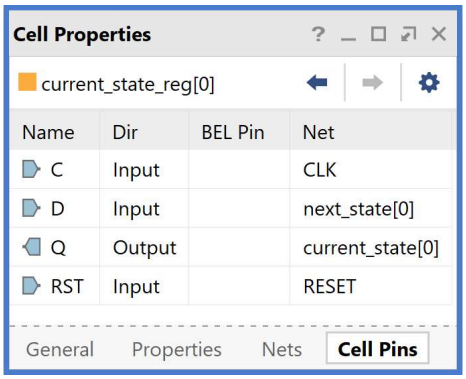
```
when (current_state = S0 = "00") =>
    next_state_i_n_0 = not X_in;

    next_state <= '0'&next_state_ i_n_0;

    If (X_in = '1') then next_state <= S0 = "00"
                    else next_state <= S1 = "01"

when (current_state = S2 = "10") =>
    next_state_i_n_0 = not X_in;

    next_state <= '0'&next_state_ i_n_0;

    If (X_in = '1') then next_state <= S0 = "00"
                    else next_state <= S1 = "01"

when (current_state = S1 = "01") =>
    next_state <= next_state_i_0;

    If (X_in = '1') then next_state <= S2 = "10"
                    else next_state <= S1 = "01"

when (current_state = "11") =>

    next_state <= S0 = "00"; (fail-safe operation)
```

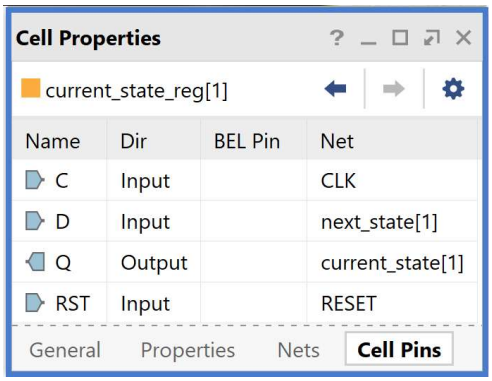


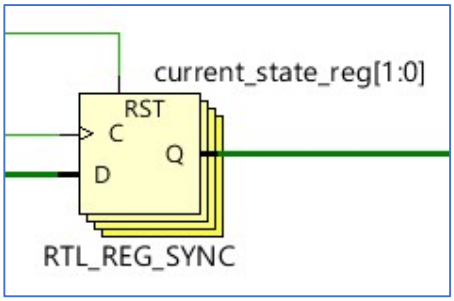


Καταχωρητής κατάστασης των 2 bit (current_state_reg_0/current_state_reg_1):

```
if (RESET = '1') then current_state <= S0 = "00";
                 else current_state <= next_state;
```

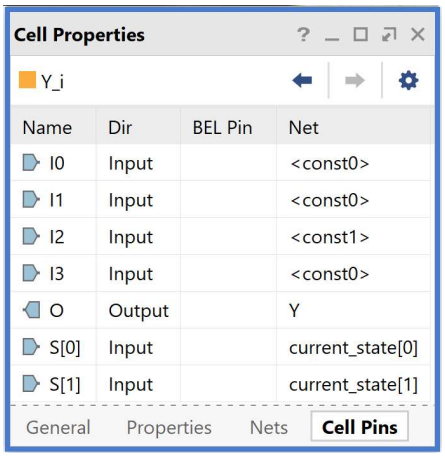


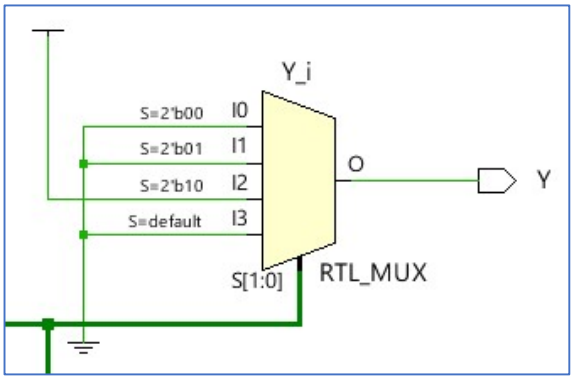




Πολυπλέκτης 2 σε 1 λογικής εξόδου:

```
if (current_state = S2 = "10") then Y <= '1'; else Y <= '0';
```





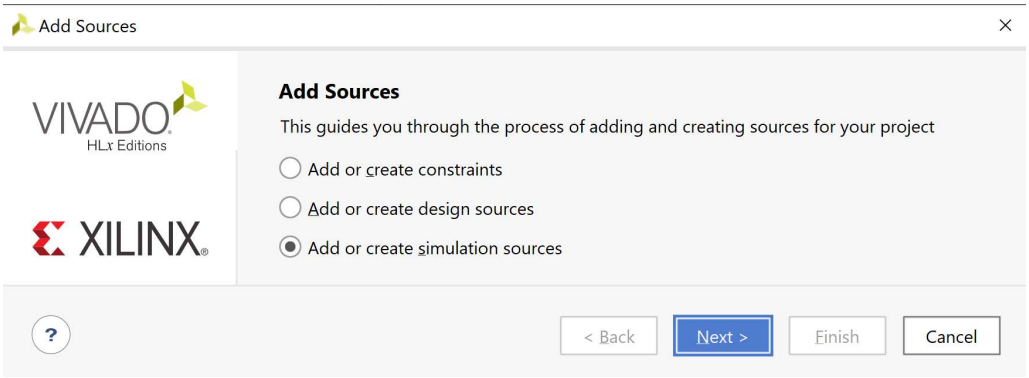
Η διαδικασία που ήδη περιγράψαμε στο Βήμα 2 «**Εισαγωγή του κώδικα VHDL και ανάλυση στο επίπεδο RTL**» εφαρμόζεται παρομοίως σε κάθε πιθανή υπομονάδα συνδυαστικής ή ακολουθιακής λογικής του επεξεργαστή.

Βήμα 3: Εισαγωγή του VHDL testbench και προσομοίωση συμπεριφοράς

3-1. Δημιουργία προγράμματος δοκιμών (testbench) τύπου VHD στο VIVADO IDE για συνδυαστική λογική ανάμεσα σε καταχωρητές

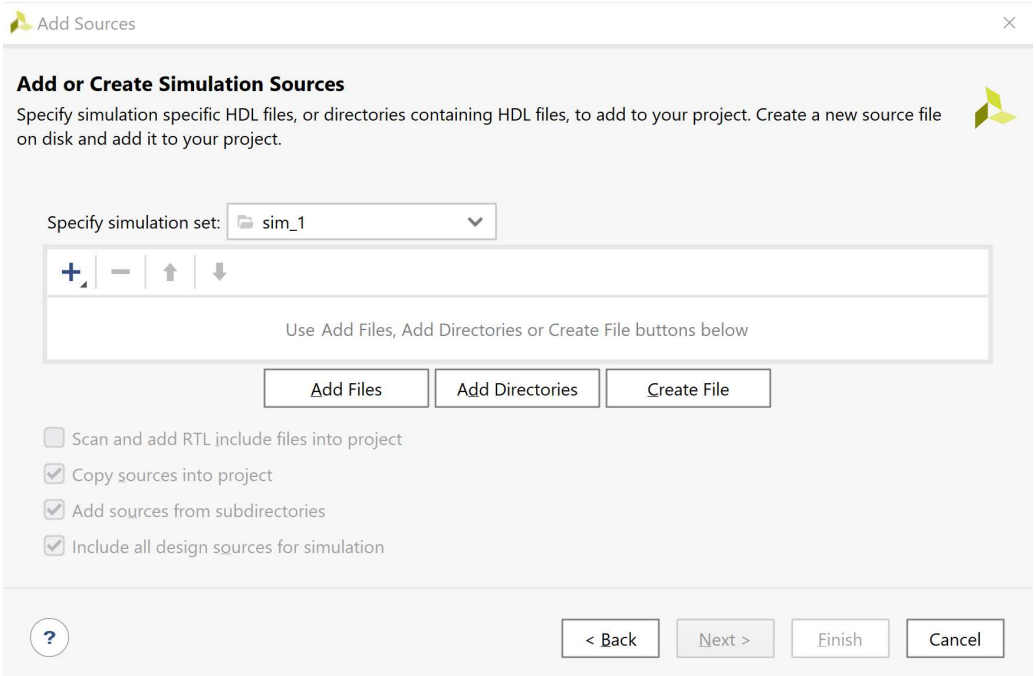
3-1.1. Πατήστε το **+** στο παράθυρο *Sources* του PROJECT MANAGER για να ξεκινήσετε τον wizard δημιουργίας ενός νέου αρχείου (source).

3-1.2. Στο παράθυρο διαλόγου *Add Sources* επιλέξτε το Add or create simulation sources, ώστε να δημιουργήσετε ένα νέο αρχείο προσομοίωσης (testbench) στη γλώσσα VHDL (simulation source file) τύπου VHD. Σε αυτό το αρχείο εισάγουμε τον κώδικα VHDL που περιγράφει τη λειτουργία του testbench. Πατήστε **Next**.

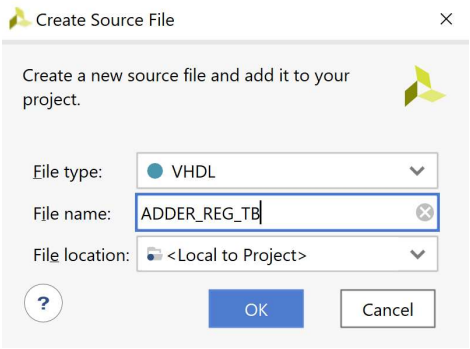


3-1.3. Στο παράθυρο διαλόγου *Add or Create Simulation Sources* πατήστε την επιλογή **Create File** για τη δημιουργία του design simulation file που θα τοποθετηθεί στο ήδη καθορισμένο design source set *sources_1*.

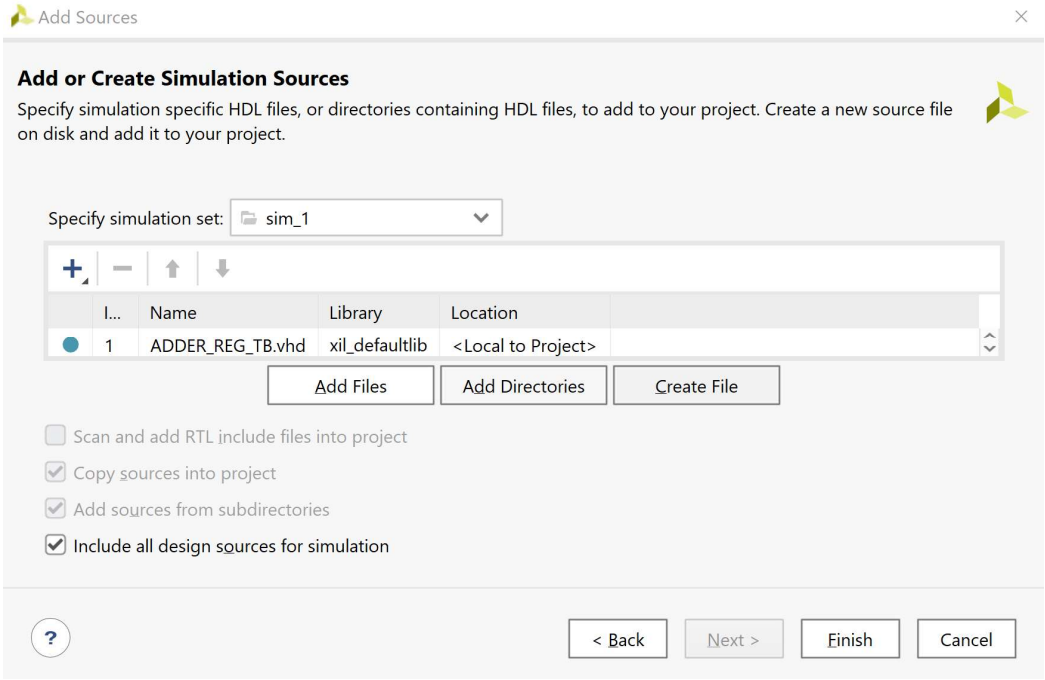
Εάν απαιτείται να καθοριστεί νέο simulation set (π.χ. sim_2), αρχικά δημιουργείστε το και στη συνέχεια ενεργοποιήστε το (επιλογή στο make active).



3-1.4. Στο παράθυρο διαλόγου *Create Source File* δηλώστε το όνομα του testbench αρχείου (_TB) VHD (π.χ. **ADDER_REG_8_TB**), αφού σε αυτό το αρχείο θα περιγράψετε στη γλώσσα VHDL την οντότητα **ADDER_REG_8_TB** που απαιτείται για τις ανάγκες της προσομοίωσης της οντότητας **ADDER_REG_8**. Πατήστε **OK**.

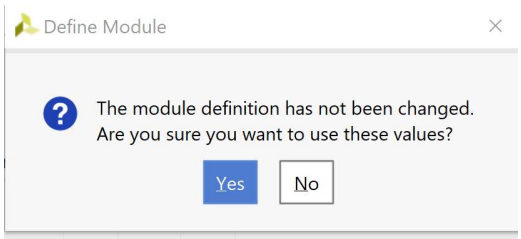


3-1.5. Επιστρέφετε στο παράθυρο διαλόγου *Add or Create Simulation Sources*, όπου φαίνεται ότι έχει δημιουργηθεί το αρχείο **ADD_REG_8_TB.vhd** και έχει συμπεριληφθεί στα αρχεία του project που ονομάζεται DSD_LAB_1. Διατηρείστε την επιλογή Include all design sources for simulation και πατήστε **Finish**.

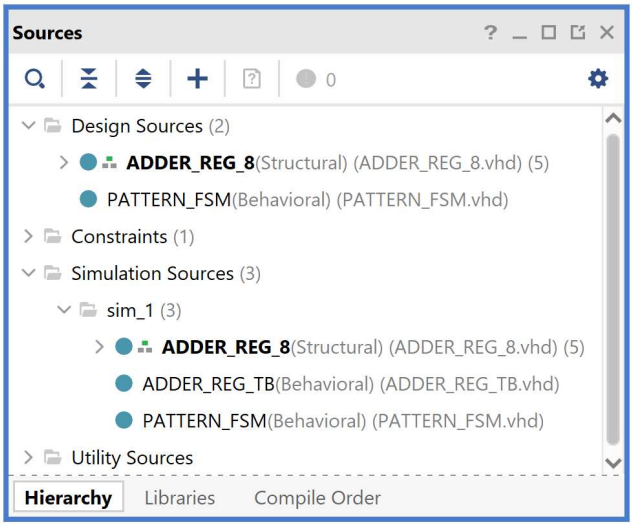


3-1.6. Εμφανίζεται το παράθυρο διαλόγου *Define Module*, όπου σας παρέχεται η δυνατότητα της δήλωσης του ονόματος της οντότητας (παραμένει **ADDER_REG_8_TB**), του ονόματος της αρχιτεκτονικής (επιλέξτε *behavioral*). Αυτή η οντότητα δεν έχει ports. Τέλος, πατήστε **OK**.

3-1.7. Εμφανίζεται το παράθυρο προειδοποίησης *Define Module* που προειδοποιεί για τη μη δήλωση των I/O ports. Πατήστε **Yes**.

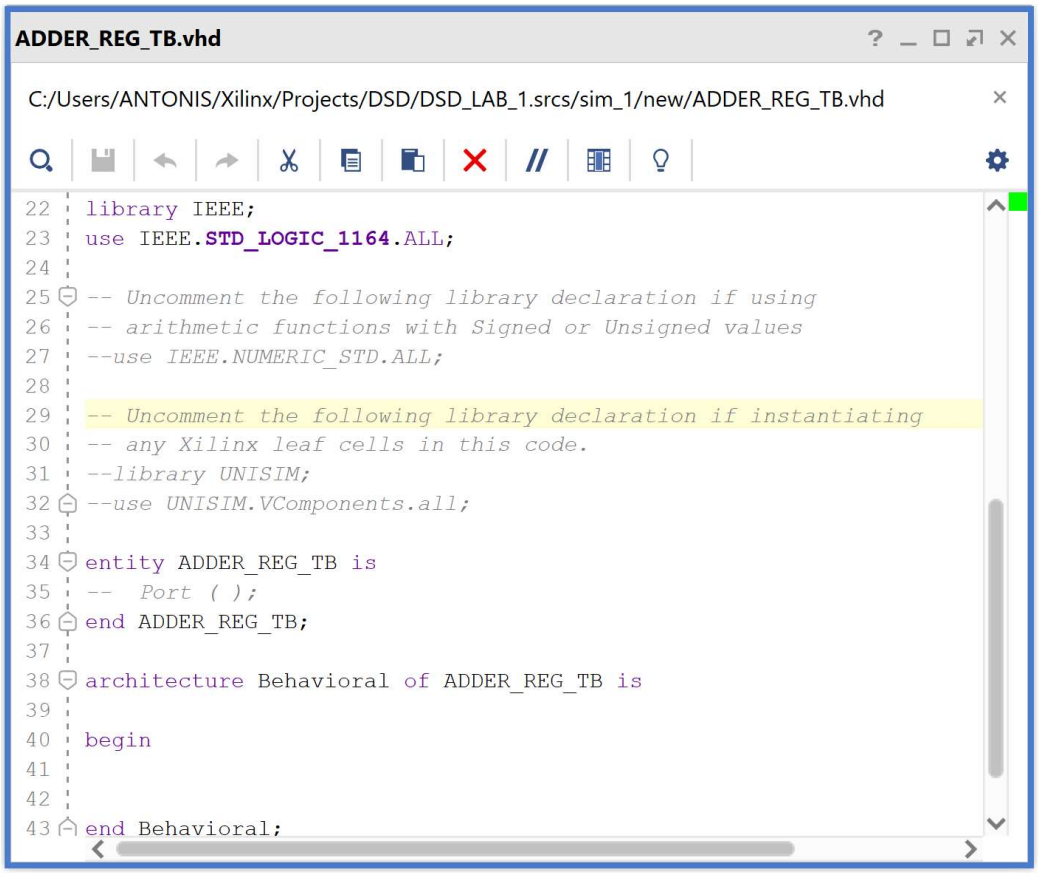


3-1.8. Επιβεβαιώστε τη δημιουργία του προγράμματος δοκιμών (testbench) **ADDER_REG_8_TB.vhd** στο παράθυρο *Sources* του PROJECT MANAGER (επίσης φαίνονται τα ονόματα της οντότητας **ADDER_REG_8_TB** και της αρχιτεκτονικής της **Behavioral**). Το αρχείο αυτό είναι αποθηκευμένο μόνο στο simulation source set *sim_1* του project DSD_LAB1.



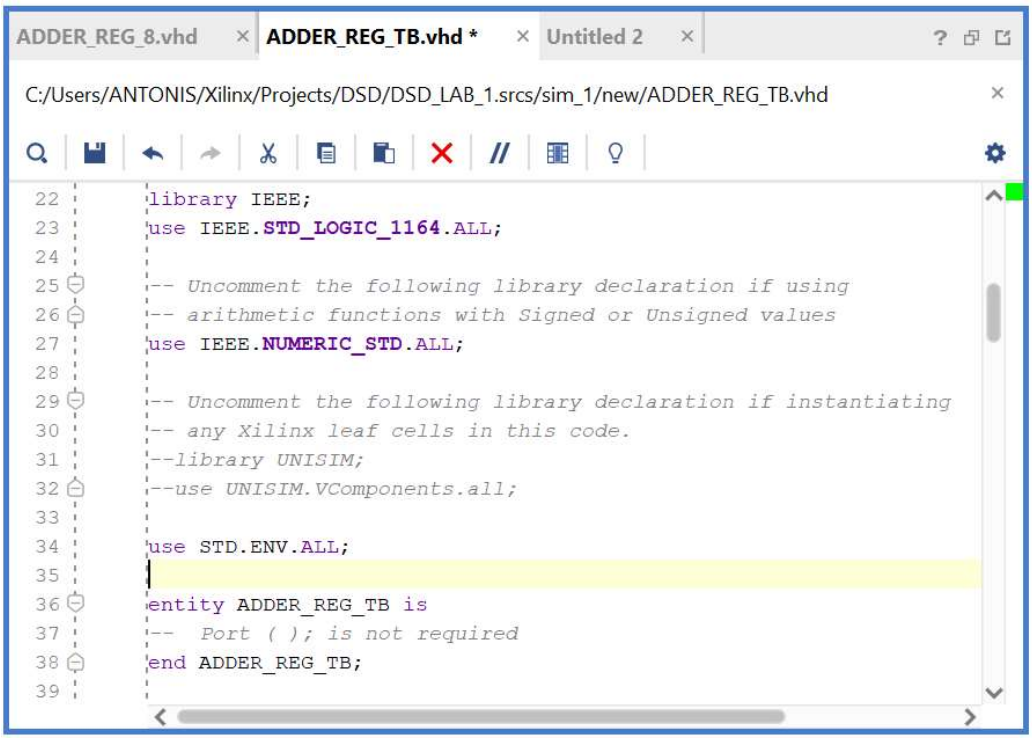
Η οντότητα **ADDER_REG_8** παραμένει ως η κορυφαία οντότητα της ιεραρχίας (**top**) των simulation resources επί του παρόντος.

3-1.9. Ανοίξτε το αρχείο **ADDER_REG_8_TB.vhd** με διπλό κλικ στο επιλεγμένο αρχείο. Λείπει ο ορισμός της αρχιτεκτονικής της οντότητας.

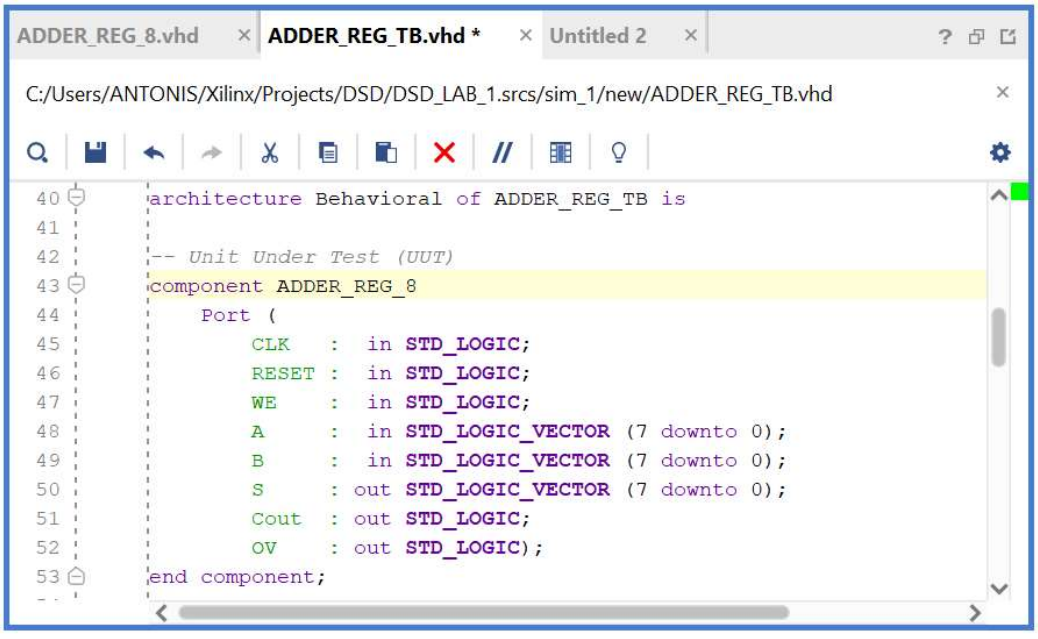


3-1.10. Συμπληρώστε το πρόγραμμα δοκιμών (testbench) **ADDER_REG_8_TB.vhd** και στο τέλος πατήστε **Save File**, αφού πρώτα ακολουθήσετε τα παρακάτω βήματα:

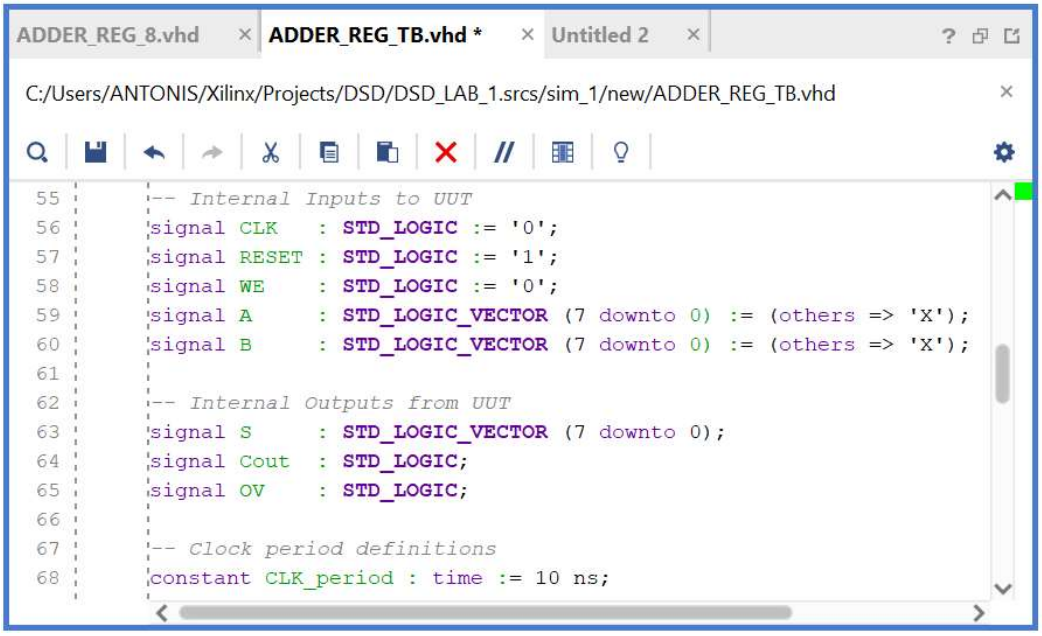
Βήμα 1: Ενεργοποιείστε τα πακέτα: **IEEE.NUMERIC_STD.ALL** και **STD.ENV.ALL**.



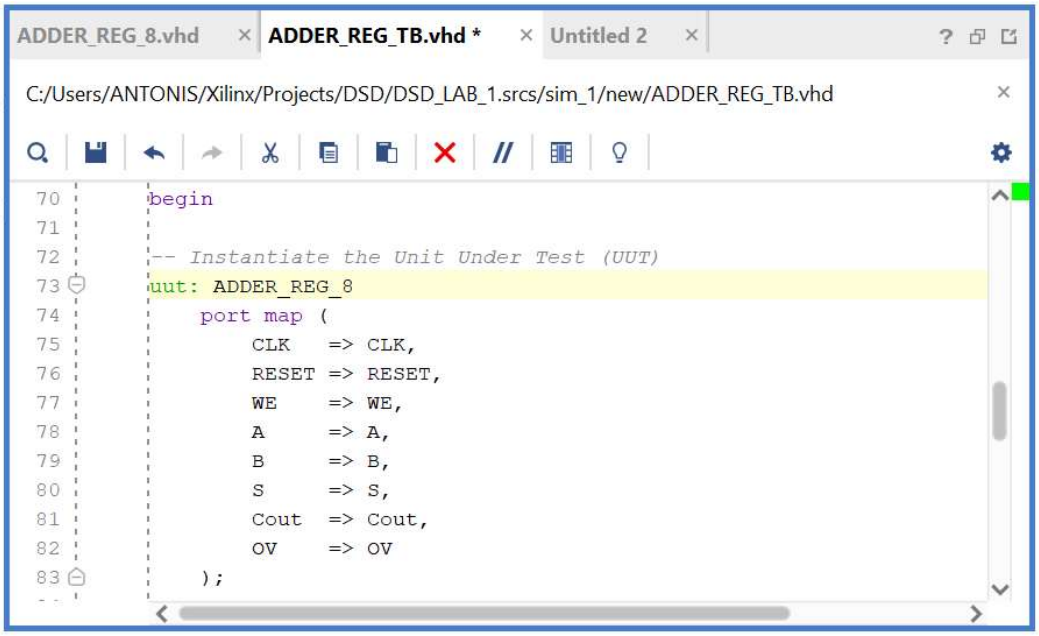
Βήμα 2: Στις δηλώσεις της αρχιτεκτονικής αρχικά προσθέστε το **component ADDER_REG_8** (προκύπτει από το **entity ADDER_REG_8** με τις κατάλληλες τροποποιήσεις) που θα είναι και το *Unit Under test (UUT)* της οντότητας **ADDER_REG_8_TB** του συγκεκριμένου προγράμματος δοκιμών (testbench).



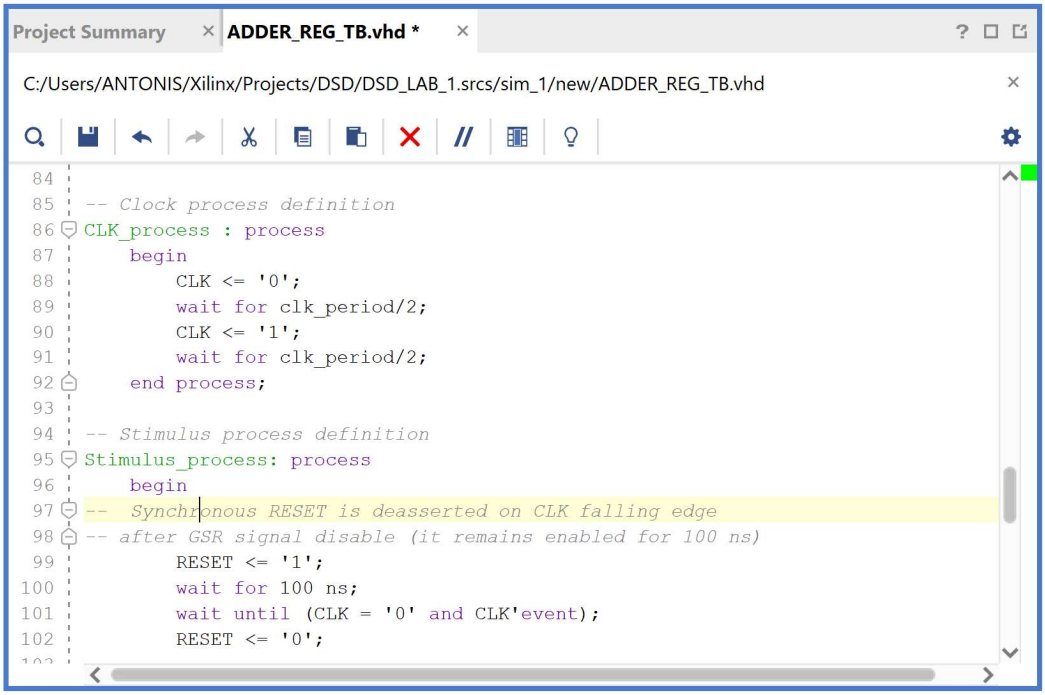
Βήμα 3: Στη συνέχεια προσθέστε τα εσωτερικά σήματα που θα χρησιμοποιηθούν ως είσοδοι και έξοδοι του *UUT*. Τέλος, ορίστε την περίοδο του CLK (έχουμε επιλέξει αρχικά τα **10 ns** λαμβάνοντας υπόψη το σήμα του ρολογιού της κάρτας που είναι στα 100 MHz). Μπορείτε να προσαρμόσετε την περίοδο του CLK στις απαιτήσεις της δικής σας υλοποίησης της σχεδίασης, για τις ανάγκες τις προσομοίωσης.



Βήμα 4: Στο σώμα της αρχιτεκτονικής μετά το begin, αρχικά, συνδέστε το *UUT* (**ADDER_REG_8**) με την οντότητα **ADDER_REG_8_TB** διατηρώντας τα ίδια ονόματα στα σήματα. Ιεραρχικά πλέον το *UUT* θα εμφανίζεται κάτω από τη συγκεκριμένη οντότητα.

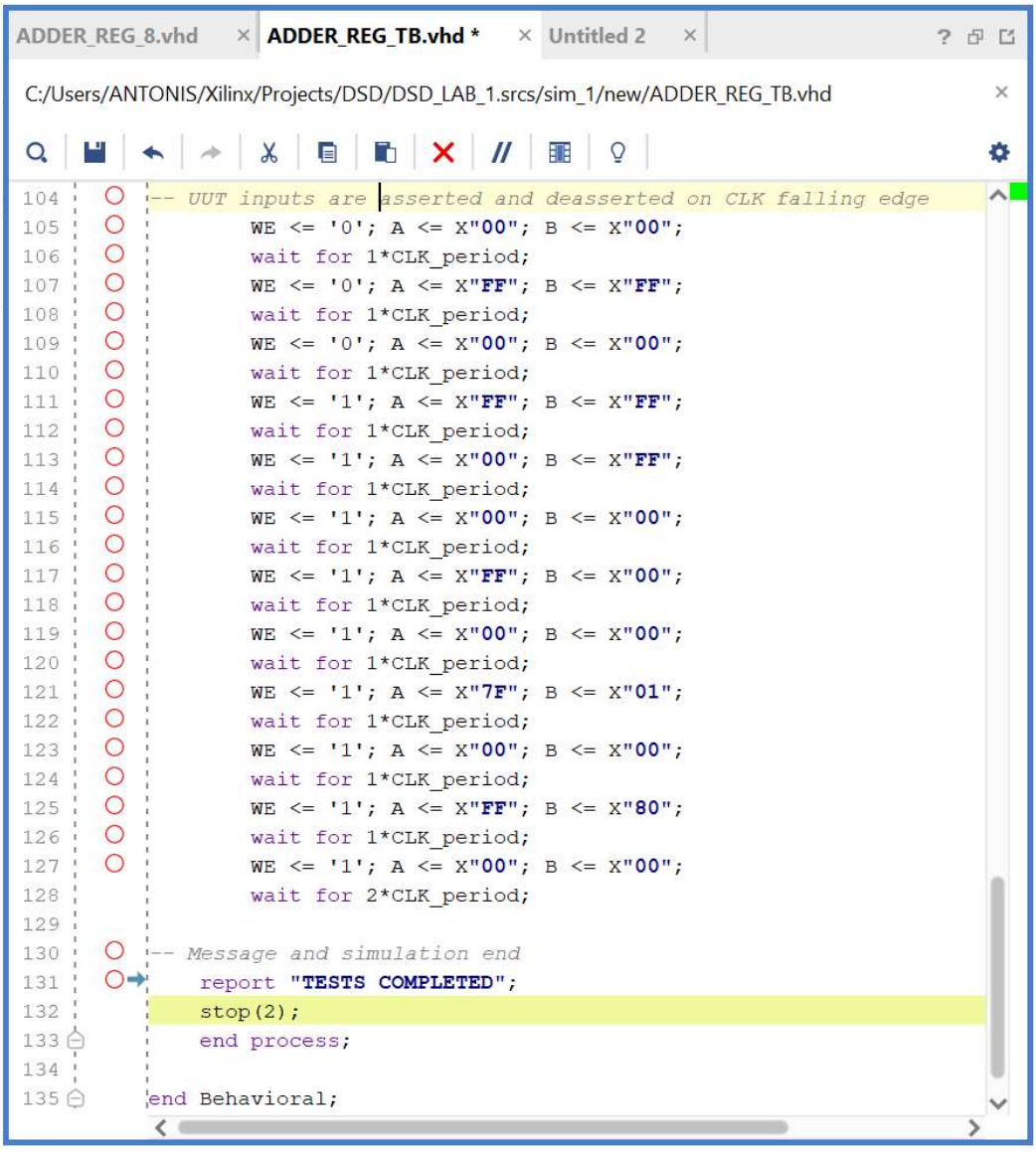


Βήμα 5: Στη συνέχεια περιγράψτε τη συμπεριφορά του CLK (στο *CLK_process*) και τη συμπεριφορά του σήματος RESET (στην αρχή του *Stimulus_process*). Για να υπάρχει συμβατότητα σε όλα τα είδη των προσομοιώσεων έχουμε επιλέξει το σήμα RESET (με τη εντολή *wait for 100 ns*) να παραμένει ενεργό για τουλάχιστον 100 ns, όσο διαρκεί η χρονική περίοδος που το FPGA επαναφέρει στην τιμή '0' όλους τους καταχωρητές και τις εξόδους του με το εσωτερικό σήμα Global Set/Reset (GSR), ώστε να μην χαθεί κάποια από τις εισόδους που θα δημιουργήσει το πρόγραμμα δοκιμών (testbench) κατά τη διάρκεια της προσομοίωσης των synthesized design models και implemented design models. Επιπλέον, έχουμε επιλέξει (με την εντολή *wait until (CLK = '0' and CLK'event)*) η απενεργοποίηση του σήματος RESET να γίνεται στην κατερχόμενη ακμή του CLK, ώστε να μην δημιουργούνται παραβιάσεις στους χρόνους σταθεροποίησης (set-up) και διατήρησης (hold) των καταχωρητών, ανεξάρτητα από την περίοδο του CLK.

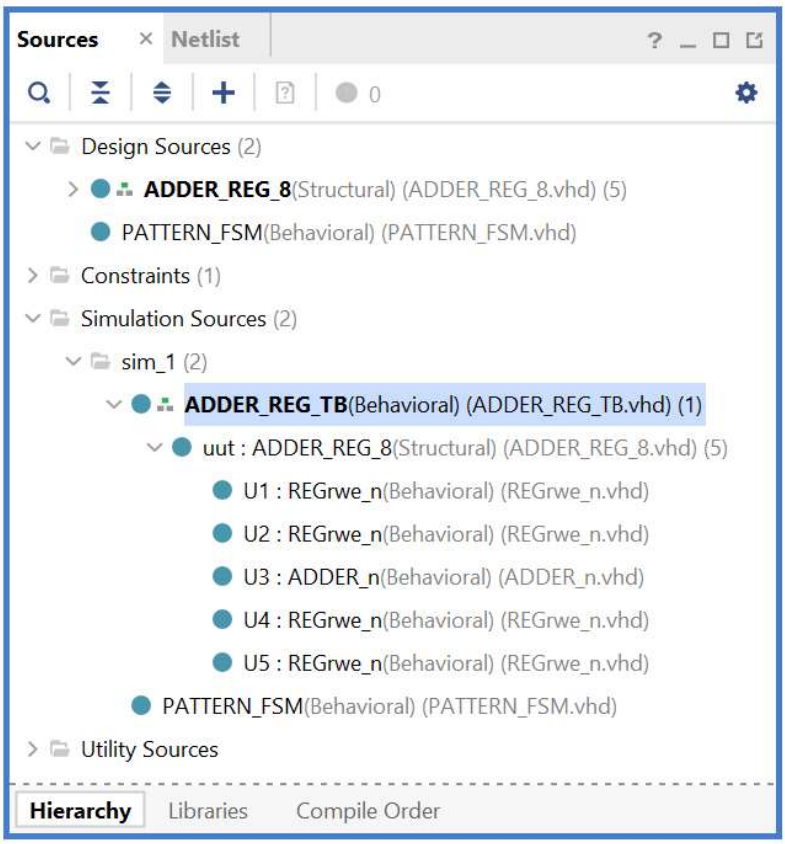


Βήμα 6: Στη συνέχεια, στο ίδιο *Stimulus_process*, περιγράψτε τις εισόδους δοκιμής που θα λάβει το *UUT* κατά τη διάρκεια της προσομοίωσης. Οι αναθέσεις τιμών σε όλες τις εισόδους του *UUT* γίνονται στην κατερχόμενη ακμή του CLK, ώστε να μην δημιουργούνται παραβιάσεις στους χρόνους σταθεροποίησης (set-up) και διατήρησης (hold) των καταχωρητών, ανεξάρτητα από την περίοδο του CLK.

Τέλος, δηλώστε μήνυμα ολοκλήρωσης της δοκιμής και διακοπής της προσομοίωσης με την εντολή *stop(2)*.



Βήμα 7: Επιβεβαιώστε την ιεραρχία της οντότητας του προγράμματος δοκιμών (testbench) **ADDER_REG_8_TB** σε σχέση με το *UUT* του (που είναι η οντότητα **ADDER_REG_8**) στο παράθυρο *Sources* του PROJECT MANAGER. Το αρχείο αυτό είναι αποθηκευμένο μόνο στο simulation source set *sim_1* του project DSD_LAB1 και πλέον αυτόματα η οντότητα **ADDER_REG_8_TB** έχει ήδη ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των simulation resources. Εάν δεν έχει γίνει αυτόματα, ορίστε το, οι ίδιοι. Πριν συνεχίσετε επιβεβαιώστε ότι η κορυφαία οντότητα της ιεραρχίας (**top**) των design resources είναι η οντότητα **ADDER_REG_8**.



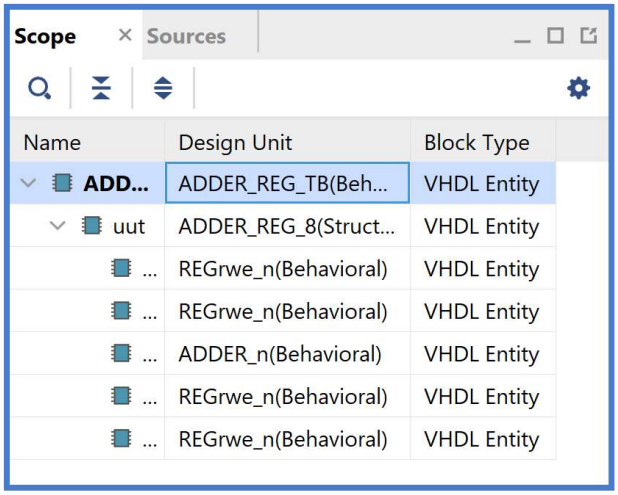
3-2. Εκτέλεση προσομοίωσης συμπεριφοράς στο VIVADO IDE για συνδυαστική λογική ανάμεσα σε καταχωρητές

Η προσομοίωση συμπεριφοράς εκτελείται στην οντότητα **ADDER_REG_8_TB** του προγράμματος δοκιμών (testbench) που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των simulation resources. Κατά την προσομοίωση, η οντότητα **ADDER_REG_8_TB** καλεί το *UUT* της (που είναι η οντότητα **ADDER_REG_8**).

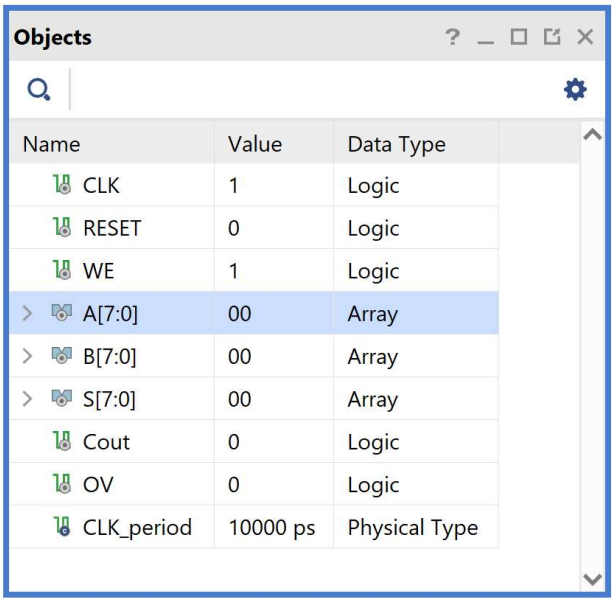
3-2.1. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Simulation**, ώστε να εμφανιστούν όλες οι πιθανές προσομοιώσεις που υποστηρίζει το Vivado IDE. Επιλέξτε **Run Behavioral Simulation**.

3-2.2. Το πρόγραμμα δοκιμής (testbench) και όλες οι οντότητες του *UUT* θα γίνουν compiled και θα τρέξει το Vivado simulator (εφόσον βέβαια δεν υπάρχουν σφάλματα). Θα εμφανιστεί το παράθυρο *SIMULATION* που απαρτίζεται από 4 παράθυρα:

Το παράθυρο *Scope*, όπου παρουσιάζεται η οντότητα **ADDER_REG_8_TB**, η οντότητα **ADDER_REG_8** (UUT) καθώς και οι υπόλοιπες οντότητες του *UUT* σύμφωνα με την ιεραρχία που έχουμε ορίσει.

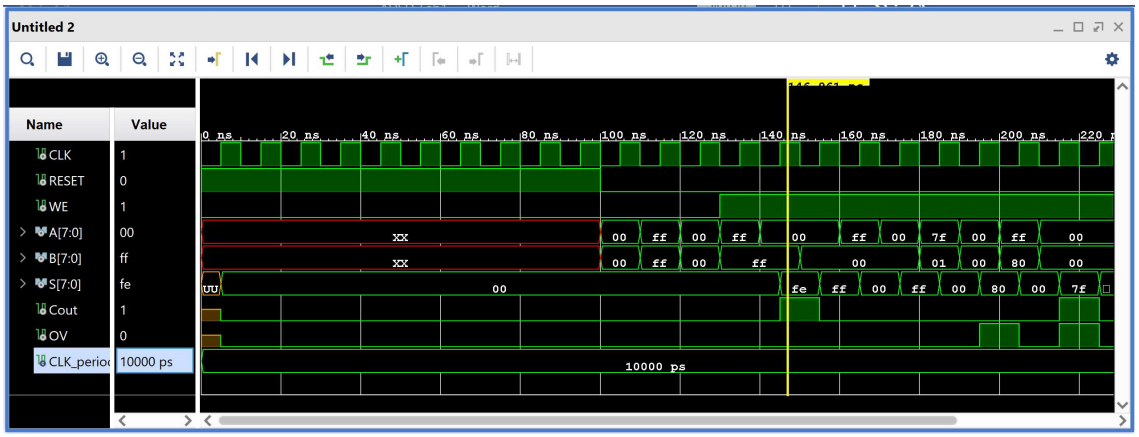


Το παράθυρο Objects, όπου εμφανίζονται τα σήματα top-level, δηλαδή οι είσοδοι και οι έξοδοι της οντότητας **ADDER_REG_8**, που είναι η κορυφαία οντότητα της ιεραρχίας του *UUT*, καθώς και η περίοδος του CLK (*CLK_period*). Οι αρτηρίες (array) αναλύονται στα σήματα που τις απαρτίζουν. Οι τιμές αντιστοιχούν στις τιμές που έχει σταματήσει η προσομοίωση της οντότητας **ADDER_REG_8_TB** (s*top (2)*).



Το παράθυρο *Tcl Console* με όλες τις διαδικασίες που εκτελούνται στο πλαίσιο της προσομοίωσης. Το Tcl Console καθαρίζει με την επιλογή *Clear*.

Το παράθυρο με τα διαγράμματα χρονισμού (μέχρι να αποθηκευτεί είναι untitled). Εάν δεν είναι εμφανές το παράθυρο *Untitled*, επιλέξτε το *Untitled*. Βλέπετε ολόκληρο το διάγραμμα χρονισμού με κατάλληλο **zoom out** ή επιλέγοντας το **zoom fit** . Για να επαναφέρετε το floating παράθυρο πίσω, απλά επιλέξτε το κουμπί Dock Window.

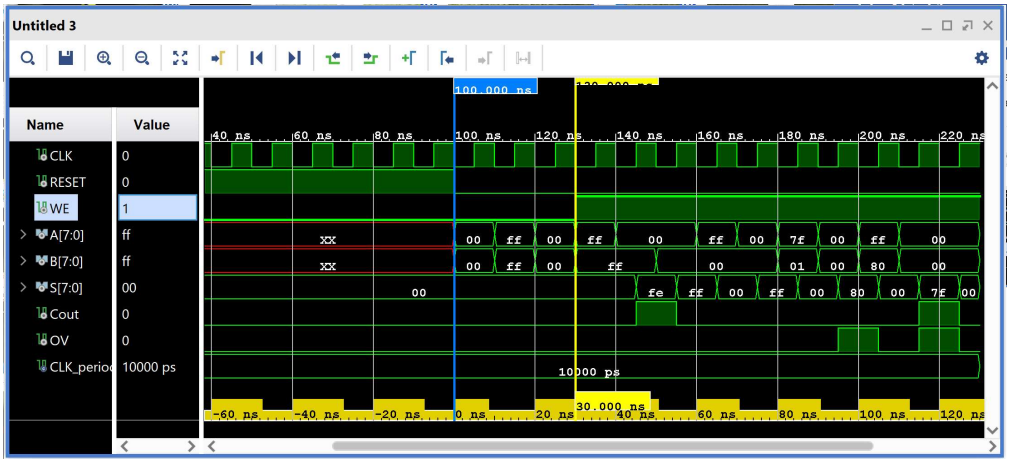


Πάνω απ' το παράθυρο με τα διαγράμματα χρονισμού, θα δείτε τα πιο κάτω κουμπιά:

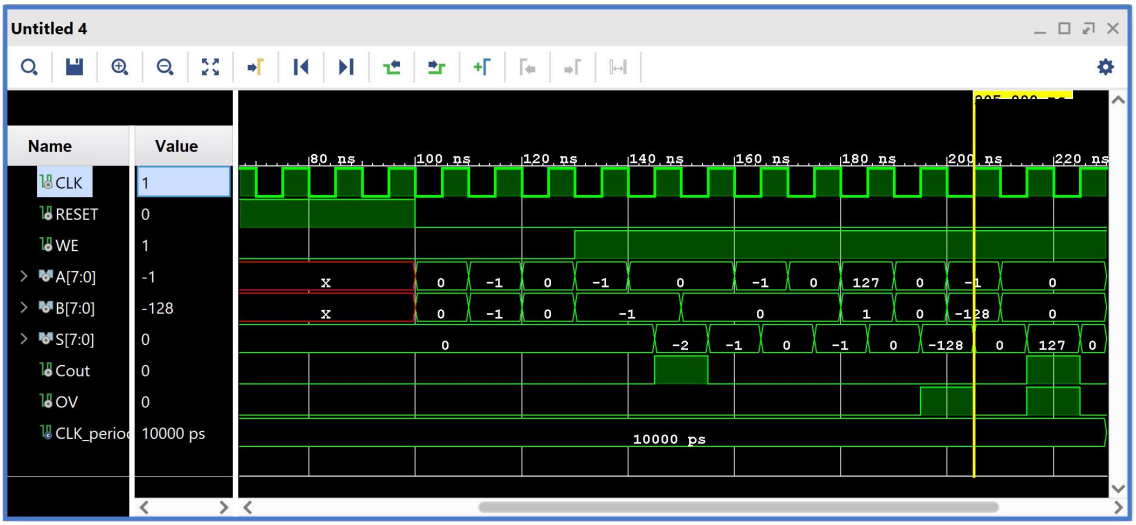
Οι τιμές (value) που βλέπουμε δίπλα από τα σήματα αντιστοιχούν στη συγκεκριμένη χρονική στιγμή που δείχνει ο κίτρινος marker. Ο κίτρινος marker μετακινείται κάνοντας κλικ σε ένα συγκεκριμένο σήμα και σε μία συγκεκριμένη χρονική στιγμή.

3-2.3. Επιλέξτε το σήμα που επιθυμείτε να μελετήσετε και μετακινείστε κατάλληλα τον κίτρινο marker, πατώντας το αντίστοιχο κουμπί: α) στην αρχή της κυμματομορφής (0 ns), β) στο τέλος της κυμματομορφής (230 ns), γ) στην προηγούμενη αλλαγή τιμής του επιλεγμένου σήματος, ή δ) στην επόμενη τιμή του επιλεγμένου σήματος, ώστε να μελετήσετε τις χρονικές στιγμές που το επιλεγμένο σήμα αλλάζει τιμή.

Προσθέστε και έναν δεύτερο marker, πατώντας το αντίστοιχο κουμπί, τον μπλε marker. Μετρήστε χρονικές αποστάσεις ανάμεσα στις αλλαγές τιμών δύο σημάτων. Για παράδειγμα, βάλτε τον μπλε marker στην κατερχόμενη ακμή του σήματος RESET και τον κίτρινο marker στην ανερχόμενη ακμή του σήματος WE. Απέχουν 30 ns.



3-2.4. Αλλάξτε την τιμή στις αρτηρίες A[7:0], B[7:0], S[7:0] από δεκαεξαδική σε προσημασμένη δεκαδική. Κάντε δεξί κλικ πάνω σε κάθε αρτηρία και επιλέξετε το Radix. Στο παράθυρο που εμφανίζεται διαλέξτε Signed Decimal.



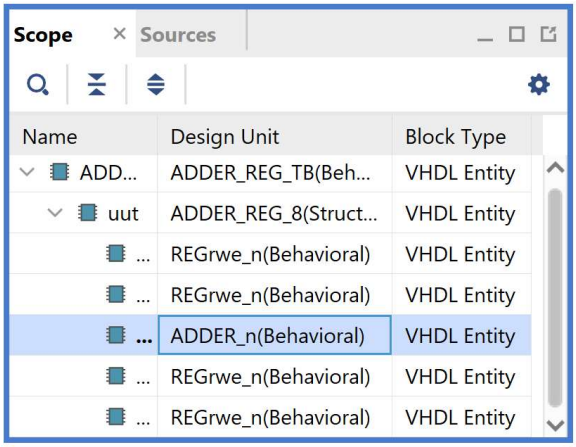
Παρατηρείστε ότι στην ανερχόμενη ακμή του CLK στο 200 ns, A = -1 και B = -128. Στην επόμενη ακμή του CLK εμφανίζεται η λανθασμένη τιμή στην έξοδο S = 127, Cout = 1, και OV = 1. Η καθυστέρηση του ενός κύκλου οφείλεται στην ύπαρξη καταχωρητών στην έξοδο του αθροιστή των 8 bit.

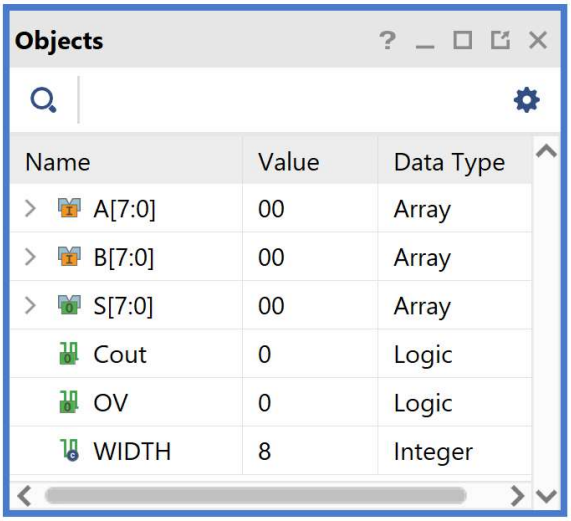
3-2.5. Προσθέστε περισσότερα **εσωτερικά σήματα** στο διάγραμμα χρονισμού της προσομοίωσης. Για παράδειγμα τις εισόδους και τις εξόδους της συνδυαστικής λογικής (της οντότητας **Adder_n**).

Στο παράθυρο *Scope* επιλέξτε την οντότητα **Adder_n**.

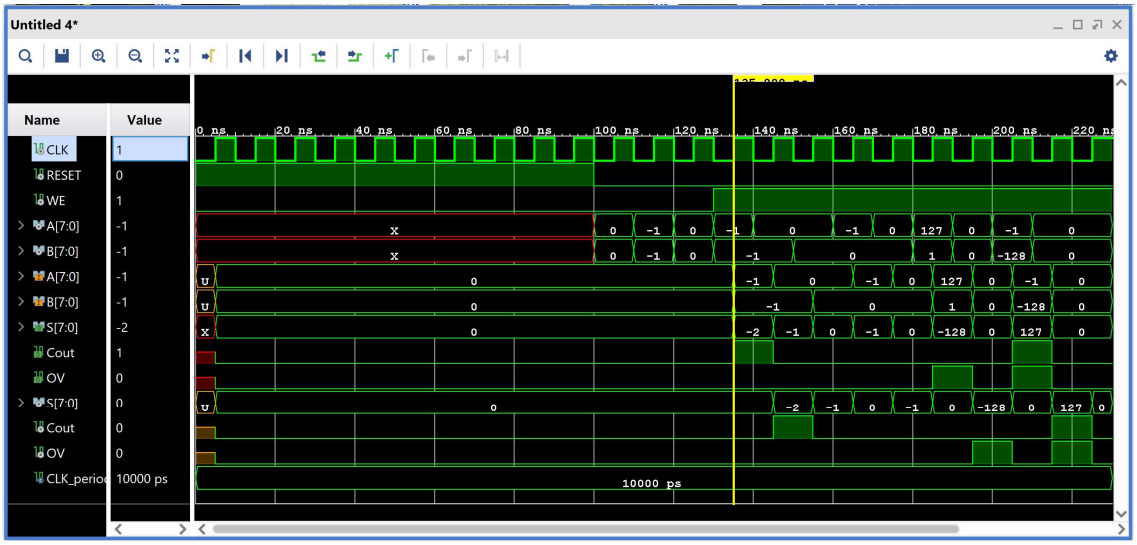
Παρατηρείστε ότι στο παράθυρο *Objects* εμφανίζονται τα σήματα της οντότητας **ADDER_n**.

Επιλέξτε το εσωτερικό σήμα A[7:0] και κάντε drag and drop στο παράθυρο με τα διαγράμματα χρονισμού, τοποθετώντας το κάτω από το υπάρχουν top-level σήμα B[7:0]. Επαναλάβετε την ίδια διαδικασία για τα εσωτερικά σήματα B[7:0], S[7:0] Cout και OV τοποθετώντας τα ακριβώς μετά το εσωτερικό σήμα A[7:0].





Επαναλάβετε τη διαδικασία της προσομοίωσης από την αρχή με επιλογή του κουμπιού **Restart** και στη συνέχεια του κουμπιού **Run All**. Και τα δύο κουμπιά βρίσκονται στην οριζόντια μπάρα στο πάνω μέρος.



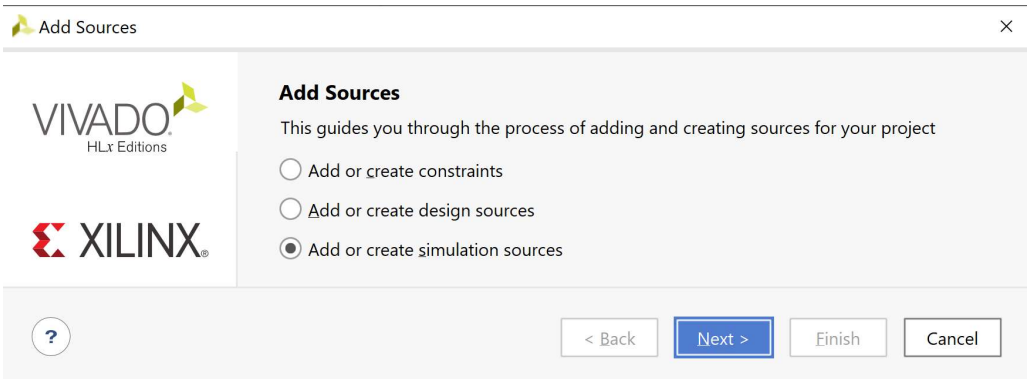
3-2.6. Κλείστε τον simulator επιλέγοντας το κουμπί × πάνω δεξιά στο παράθυρο του *SIMULATION*. Στο παράθυρο *Confirm Close* που εμφανίζεται πατήστε **OK**. Στο παράθυρο *Save Waveform Configuration* πατήστε **Save** για να αποθηκεύσετε το configuration του διαγράμματος χρονισμού στο οποίο έχετε καταλήξει, αλλιώς πατήστε **Discard**. Εάν πατήσετε **Save**, στο παράθυρο *Save Waveform* που εμφανίζεται διατηρείστε το όνομα **ADDER_REG_TB_behav.wcfg** και πατήστε **Save**. Στο παράθυρο *Waveform Configuration File* που εμφανίζεται πατήστε **Yes**.

Η διαδικασία που ήδη περιγράψαμε στα Βήματα 3-1 και 3-2 της «**Εισαγωγής του VHDL testbench και προσομοίωσης συμπεριφοράς**» εφαρμόζεται παρομοίως σε κάθε πιθανή υπομονάδα συνδυαστικής λογικής της διαδρομής δεδομένων και της μονάδας ελέγχου του επεξεργαστή.

3-3. Δημιουργία προγράμματος δοκιμών (testbench) τύπου VHD στο VIVADO IDE για μηχανές πεπερασμένων καταστάσεων (FSM)

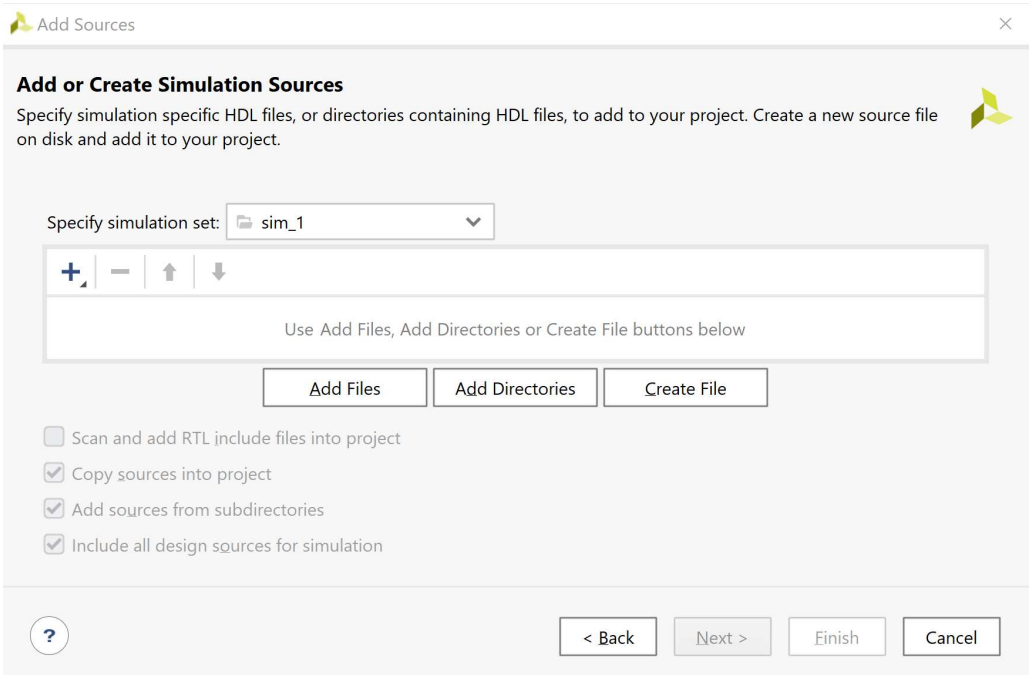
3-3.1. Πατήστε το **+** στο παράθυρο *Sources* του PROJECT MANAGER για να ξεκινήσετε τον wizard δημιουργίας ενός νέου αρχείου (source).

3-3.2. Στο παράθυρο διαλόγου *Add Sources* επιλέξτε το Add or create simulation sources, ώστε να δημιουργήσετε ένα νέο αρχείο προσομοίωσης (testbench) στη γλώσσα VHDL (simulation source file) τύπου VHD. Σε αυτό το αρχείο εισάγουμε τον κώδικα VHDL που περιγράφει τη λειτουργία του testbench. Πατήστε **Next**.

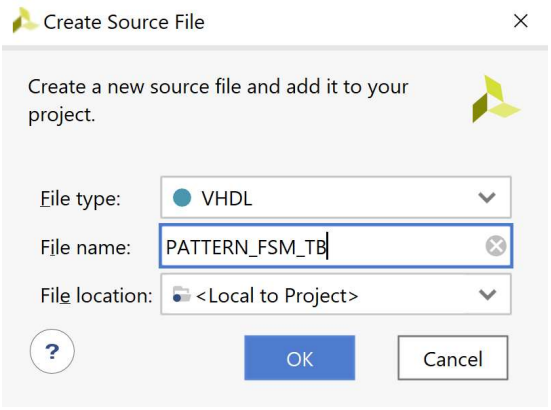


3-3.3. Στο παράθυρο διαλόγου *Add or Create Simulation Sources* πατήστε την επιλογή **Create File** για τη δημιουργία του design simulation file που θα τοποθετηθεί στο ήδη καθορισμένο design source set *sources_1*.

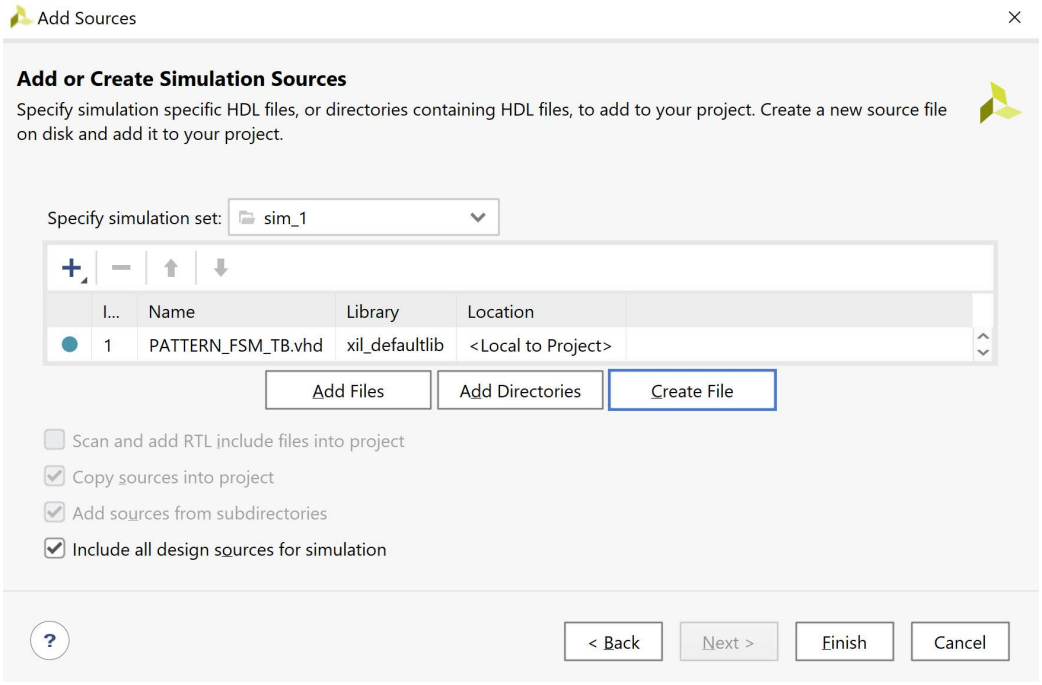
Εάν απαιτείται να καθοριστεί νέο simulation set (π.χ. sim_2), αρχικά δημιουργείστε το και στη συνέχεια ενεργοποιήστε το (επιλογή στο make active).



3-3.4. Στο παράθυρο διαλόγου *Create Source File* δηλώστε το όνομα του testbench αρχείου (_TB) VHD (π.χ. **PATTERN_FSM_TB**), αφού σε αυτό το αρχείο θα περιγράψετε στη γλώσσα VHDL την οντότητα **PATTERN_FSM_TB** που απαιτείται για τις ανάγκες της προσομοίωσης της οντότητας **PATTERN_FSM**. Πατήστε **OK**.

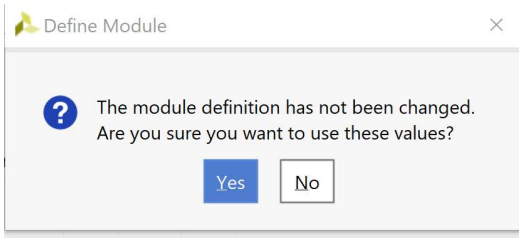


3-3.5. Επιστρέφετε στο παράθυρο διαλόγου *Add or Create Simulation Sources*, όπου φαίνεται ότι έχει δημιουργηθεί το αρχείο **PATTERN_FSM_TB.vhd** και έχει συμπεριληφθεί στα αρχεία του project που ονομάζεται DSD_LAB_1. Διατηρείστε την επιλογή Include all design sources for simulation και πατήστε **Finish**.

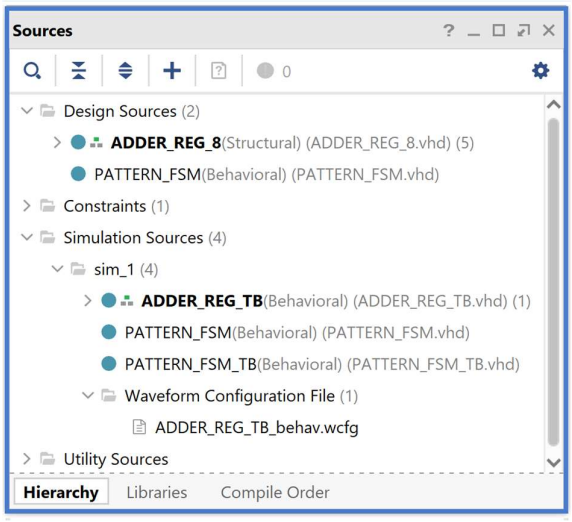


3-3.6. Εμφανίζεται το παράθυρο διαλόγου *Define Module*, όπου σας παρέχεται η δυνατότητα της δήλωσης του ονόματος της οντότητας (παραμένει **PATTERN_FSM_TB**), του ονόματος της αρχιτεκτονικής (επιλέξτε *behavioral*). Αυτή η οντότητα δεν έχει ports. Τέλος, πατήστε **OK**.

3-3.7. Εμφανίζεται το παράθυρο προειδοποίησης *Define Module* που προειδοποιεί για τη μη δήλωση των I/O ports. Πατήστε **Yes**.

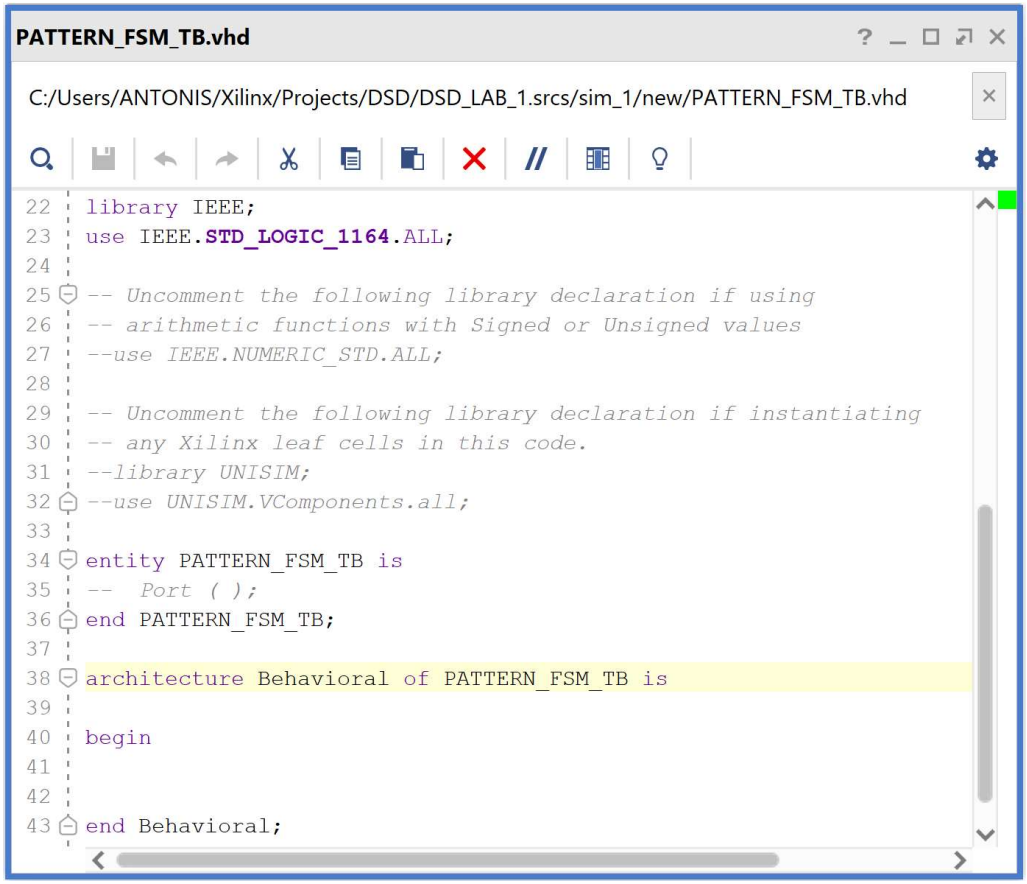


3-3.8. Επιβεβαιώστε τη δημιουργία του προγράμματος δοκιμών (testbench) **PATTERN_FSM_TB.vhd** στο παράθυρο *Sources* του PROJECT MANAGER (επίσης φαίνονται τα ονόματα της οντότητας **PATTERN_FSM_TB** και της αρχιτεκτονικής της **Behavioral**). Το αρχείο αυτό είναι αποθηκευμένο μόνο στο simulation source set *sim_1* του project DSD_LAB1.



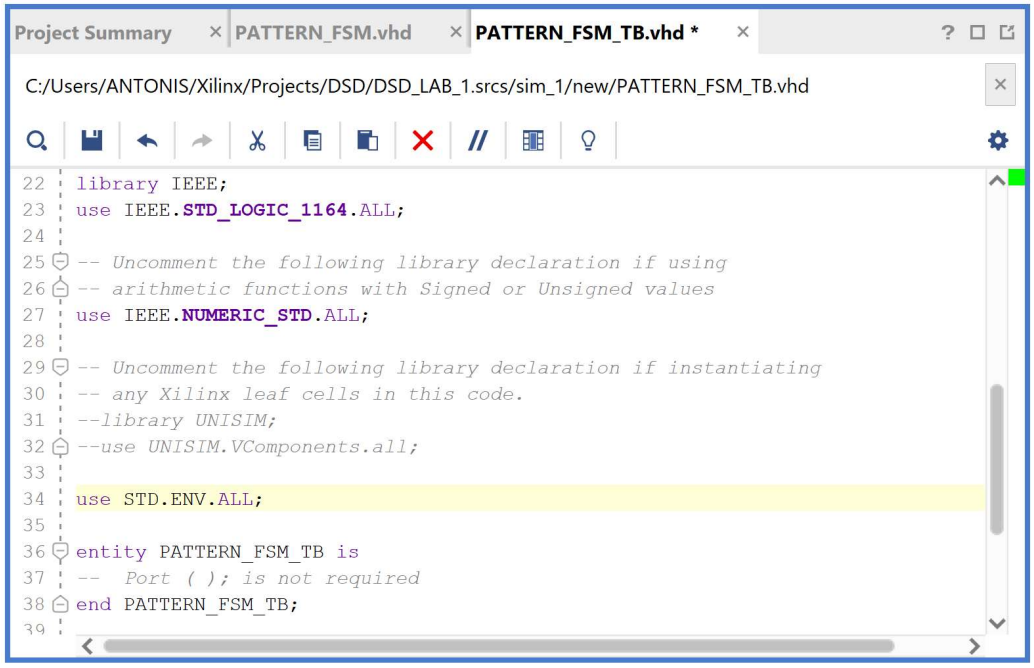
Η οντότητα **ADDER_REG_8_TB** παραμένει ως η κορυφαία οντότητα της ιεραρχίας (**top**) των simulation resources επί του παρόντος. Επίσης, βλέπουμε όλα τα αρχεία που έχουμε δημιουργήσει μέχρι τώρα.

3-3.9. Ανοίξτε το αρχείο **PATTERN_FSM_TB.vhd** με διπλό κλικ στο επιλεγμένο αρχείο. Λείπει ο ορισμός της αρχιτεκτονικής της οντότητας.

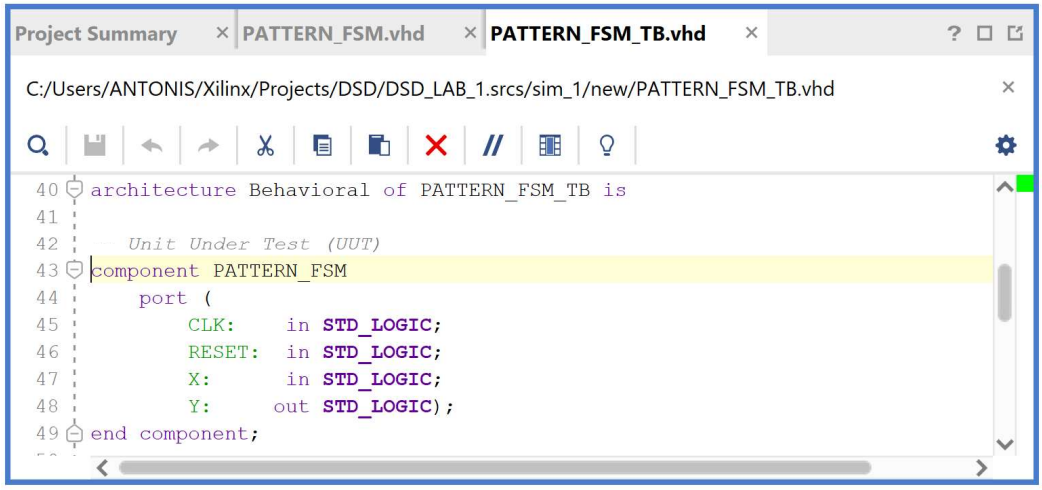


3-3.10. Συμπληρώστε το πρόγραμμα δοκιμών (testbench) **PATTERN_FSM_TB.vhd** και στο τέλος πατήστε **Save File**, αφού πρώτα ακολουθήσετε τα παρακάτω βήματα:

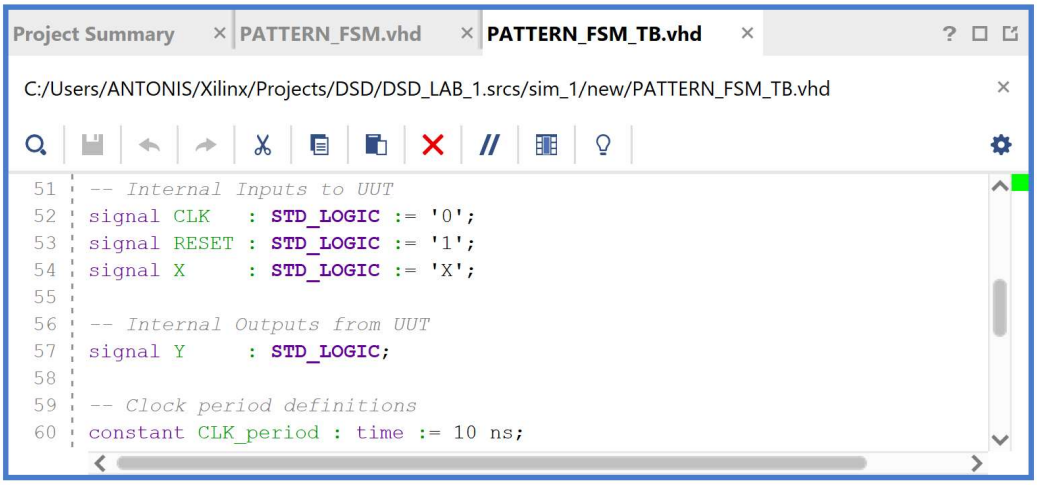
Βήμα 1: Ενεργοποιείστε τα πακέτα: **IEEE.NUMERIC_STD.ALL** και **STD.ENV.ALL**.



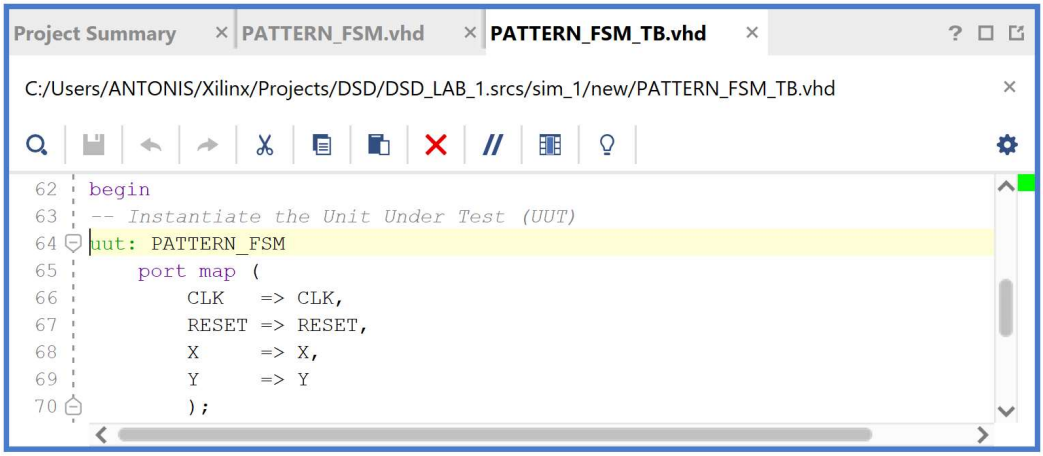
Βήμα 2: Στις δηλώσεις της αρχιτεκτονικής αρχικά προσθέστε το **component PATTERN_FSM** (προκύπτει από το **entity PATTERN_FSM** με τις κατάλληλες τροποποιήσεις) που θα είναι και το *Unit Under test (UUT)* της οντότητας **PATTERN_FSM _TB** του συγκεκριμένου προγράμματος δοκιμών (testbench).



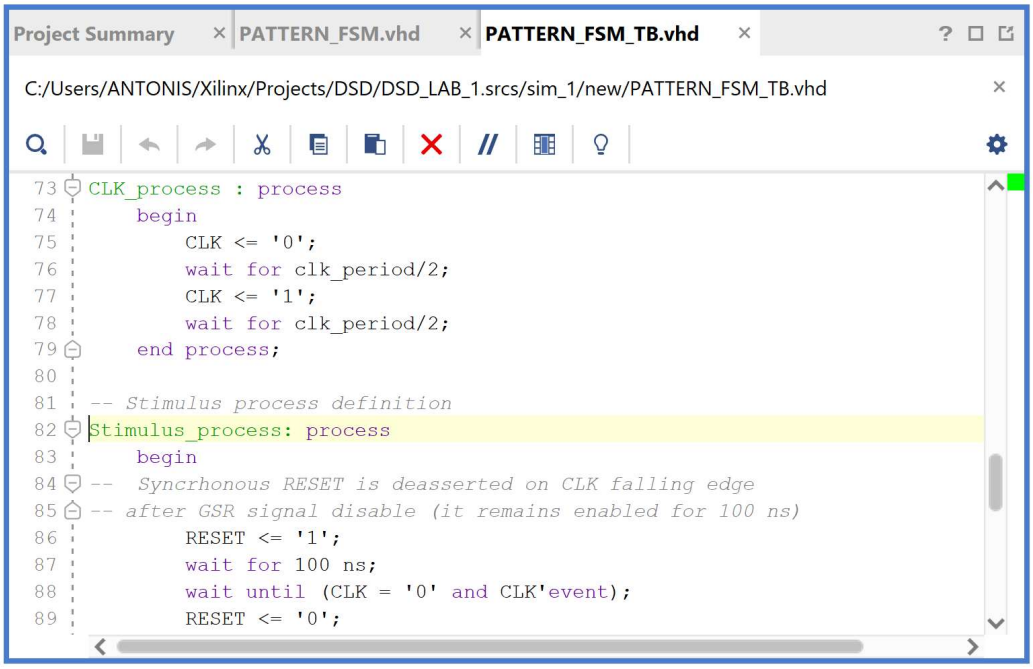
Βήμα 3: Στη συνέχεια προσθέστε τα εσωτερικά σήματα που θα χρησιμοποιηθούν ως είσοδοι και έξοδοι του *UUT*. Τέλος, ορίστε την περίοδο του CLK (έχουμε επιλέξει αρχικά τα **10 ns** λαμβάνοντας υπόψη το σήμα του ρολογιού της κάρτας που είναι στα 100 MHz). Μπορείτε να προσαρμόσετε την περίοδο του CLK στις απαιτήσεις της δικής σας υλοποίησης της σχεδίασης, για τις ανάγκες τις προσομοίωσης.



Βήμα 4: Στο σώμα της αρχιτεκτονικής μετά το begin, αρχικά, συνδέστε το *UUT* (**PATTERN_FSM**) με την οντότητα **PATTERN_FSM_TB** διατηρώντας τα ίδια ονόματα στα σήματα. Ιεραρχικά πλέον το *UUT* θα εμφανίζεται κάτω από τη συγκεκριμένη οντότητα.



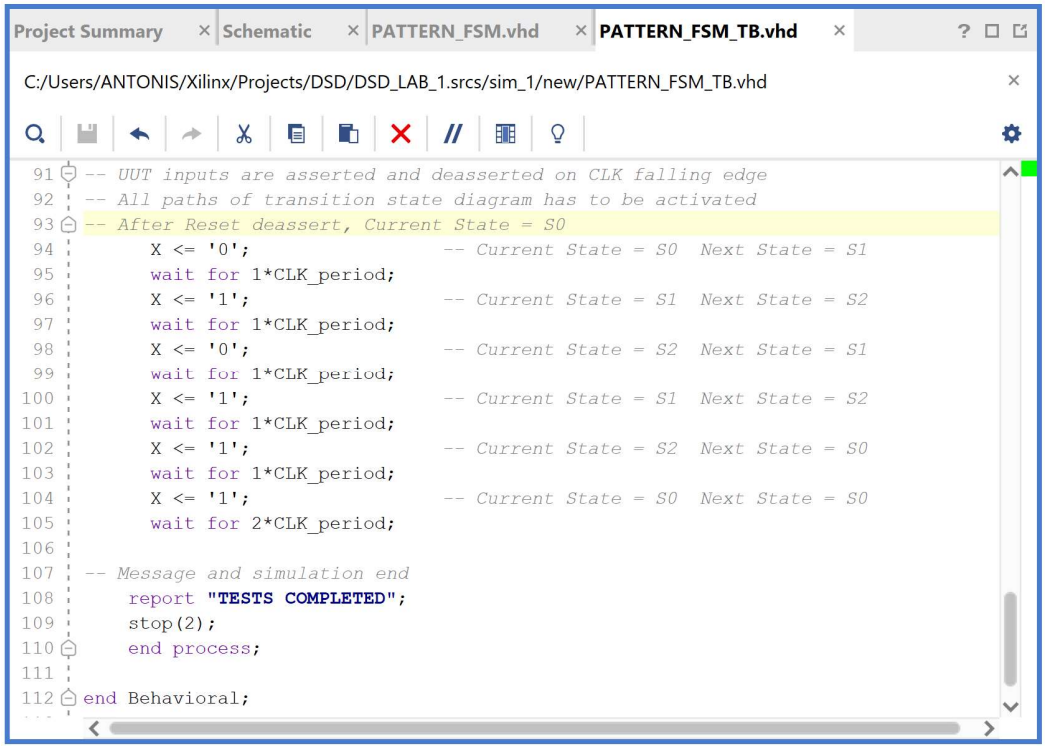
Βήμα 5: Στη συνέχεια περιγράψτε τη συμπεριφορά του CLK (στο *CLK_process*) και τη συμπεριφορά του σήματος RESET (στην αρχή του *Stimulus_process*). Για να υπάρχει συμβατότητα σε όλα τα είδη των προσομοιώσεων έχουμε επιλέξει το σήμα RESET (με τη εντολή *wait for 100 ns*) να παραμένει ενεργό για τουλάχιστον 100 ns, όσο διαρκεί η χρονική περίοδος που το FPGA επαναφέρει στην τιμή '0' όλους τους καταχωρητές και τις εξόδους του με το εσωτερικό σήμα Global Set/Reset (GSR), ώστε να μην χαθεί κάποια από τις εισόδους που θα δημιουργήσει το πρόγραμμα δοκιμών (testbench) κατά τη διάρκεια της προσομοίωσης των synthesized design models και implemented design models. Επιπλέον, έχουμε επιλέξει (με την εντολή *wait until (CLK = '0' and CLK'event)*) η απενεργοποίηση του σήματος RESET να γίνεται στην κατερχόμενη ακμή του CLK, ώστε να μην δημιουργούνται παραβιάσεις στους χρόνους σταθεροποίησης (set-up) και διατήρησης (hold) των καταχωρητών, ανεξάρτητα από την περίοδο του CLK.



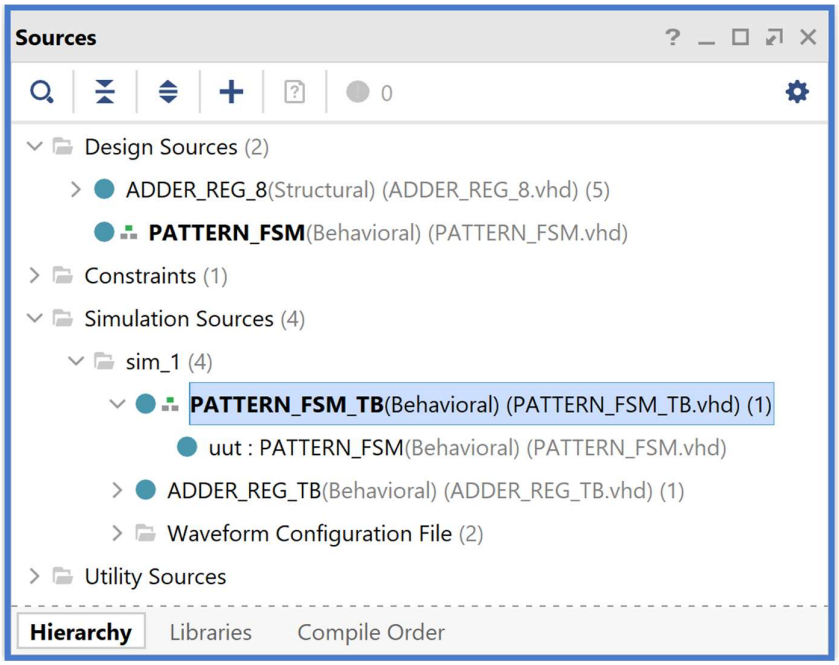
Βήμα 6: Στη συνέχεια, στο ίδιο *Stimulus_process*, περιγράψτε τις εισόδους δοκιμής που θα λάβει το *UUT* κατά τη διάρκεια της προσομοίωσης. Οι αναθέσεις τιμών σε όλες τις εισόδους του *UUT* γίνονται στην κατερχόμενη ακμή του CLK, ώστε να μην δημιουργούνται παραβιάσεις στους χρόνους σταθεροποίησης (set-up) και διατήρησης (hold) των καταχωρητών, ανεξάρτητα από την περίοδο του CLK.

Στην περίπτωση των μηχανών πεπερασμένων καταστάσεων (FSM) για κάθε τρέχουσα κατάσταση ορίζουμε την επόμενη κατάσταση με βάση τις τιμές στην είσοδο, ξεκινώντας από την κατάσταση S0 του FSM που προκύπτει με την ενεργοποίηση του σήματος RESET. Φροντίζουμε να ενεργοποιήσουμε όλες τις διαδρομές του διαγράμματος μεταβολής κατάστασης, που εμφανίζονται κατά την κανονική λειτουργία του FSM.

Τέλος, δηλώστε μήνυμα ολοκλήρωσης της δοκιμής και διακοπής της προσομοίωσης με την εντολή *stop(2)*.



Βήμα 7: Επιβεβαιώστε την ιεραρχία της οντότητας του προγράμματος δοκιμών (testbench) **PATTERN_FSM_TB** σε σχέση με το *UUT* του (που είναι η οντότητα **PATTERN_FSM**) στο παράθυρο *Sources* του PROJECT MANAGER. Το αρχείο αυτό είναι αποθηκευμένο μόνο στο simulation source set *sim_1* του project DSD_LAB1. Ορίστε την οντότητα **PATTERN_FSM_TB** ως την κορυφαία οντότητα της ιεραρχίας (**top**) των simulation resources. Επίσης, ορίστε την οντότητα **PATTERN_FSM** ως την κορυφαία οντότητα της ιεραρχίας (**top**) των design resources.



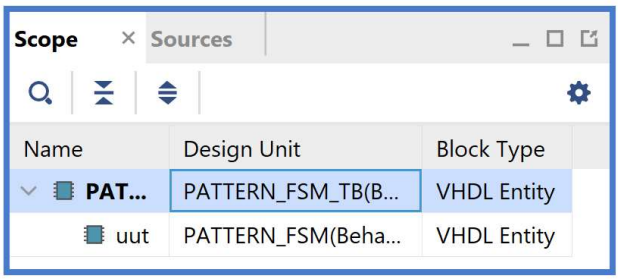
3-4. Εκτέλεση προσομοίωσης συμπεριφοράς στο VIVADO IDE για μηχανές πεπερασμένων καταστάσεων (FSM)

Η προσομοίωση συμπεριφοράς εκτελείται στην οντότητα **PATTERN_FSM_TB** του προγράμματος δοκιμών (testbench) που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των simulation resources. Κατά την προσομοίωση, η οντότητα **PATTERN_FSM_TB** καλεί το *UUT* της (που είναι η οντότητα **PATTERN_FSM**). Θα εμφανιστεί ένα νέο παράθυρο διαγραμμάτων χρονισμού χωρίς όνομα (untitled) και χωρίς σήματα.

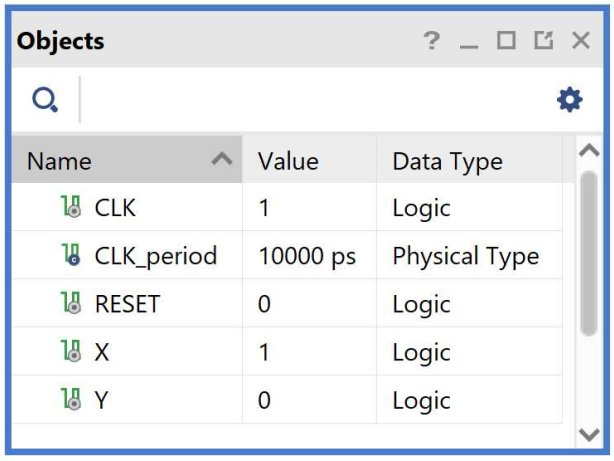
3-4.1. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Simulation**, ώστε να εμφανιστούν όλες οι πιθανές προσομοιώσεις που υποστηρίζει το Vivado IDE. Επιλέξτε **Run Behavioral Simulation**.

3-4.2. Το πρόγραμμα δοκιμής (testbench) και όλες οι οντότητες του *UUT* θα γίνουν compiled και θα τρέξει το Vivado simulator (εφόσον βέβαια δεν υπάρχουν σφάλματα). Θα εμφανιστεί το παράθυρο *SIMULATION* που απαρτίζεται από 4 παράθυρα:

Το παράθυρο *Scope*, όπου παρουσιάζεται η οντότητα **PATTERN_FSM_TB** και η οντότητα **PATTERN_FSM** (*UUT*).

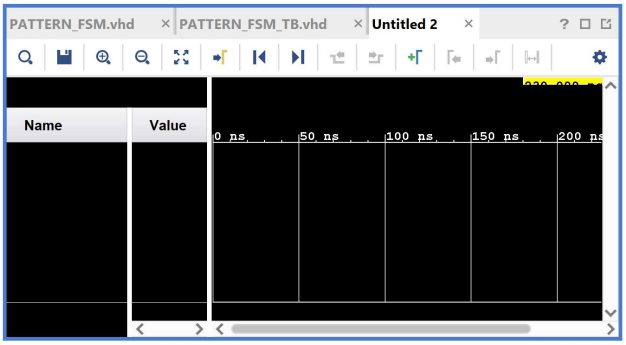


Το παράθυρο Objects, όπου εμφανίζονται τα σήματα top-level, δηλαδή οι είσοδοι και οι έξοδοι της οντότητας **PATTERN_FSM**, που είναι η κορυφαία οντότητα της ιεραρχίας του *UUT*, καθώς και η περίοδος του CLK (*CLK_period*). Οι τιμές αντιστοιχούν στις τιμές που έχει σταματήσει η προσομοίωση της οντότητας **PATTERN_FSM_TB** (st*op (2)*).

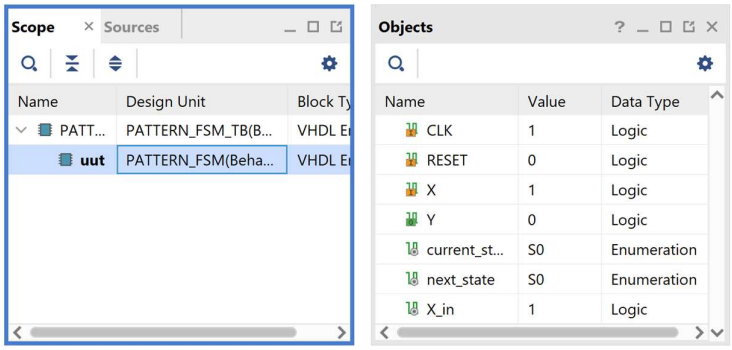


Το παράθυρο *Tcl Console* με όλες τις διαδικασίες που εκτελούνται στο πλαίσιο της προσομοίωσης. Το Tcl Console καθαρίζει με την επιλογή *Clear*.

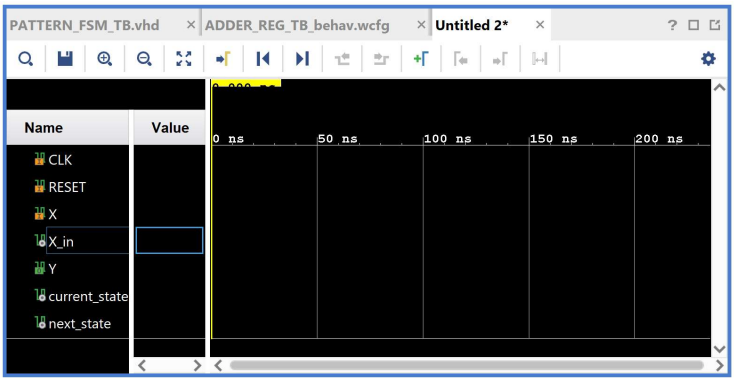
Το παράθυρο με τα διαγράμματα χρονισμού. **Προσοχή**! Εάν έχετε αποθηκεύσει κάποιο προηγούμενο *waveform configuration*, όπως για παράδειγμα το **ADDER_REG_TB_behav.wcfg**, που αφορά στην προσομοίωση της οντότητας **ADDER_REG_TB** θα πρέπει να το αποσυνδέστε από την εκτέλεση της προσομοίωσης της οντότητας **PATTERN_FSM_TB** επιλέγοντας το **File**, στη συνέχεια το **Simulation Waveform** και τέλος το **New Configuration**. Θα εμφανιστεί ένα κενό παράθυρο διαγράμματος χρονισμού χωρίς όνομα (*untitled*), όπως το ακόλουθο.



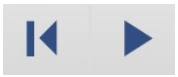
3-4.3. Επιλέξτε στο παράθυρο Scope την οντότητα **PATTERN_FSM** (*UUT*). Στο παράθυρο *Objects* θα εμφανιστούν όλα τα σήματα της οντότητας **PATTERN_FSM** (είσοδοι, έξοδοι και εσωτερικά σήματα).



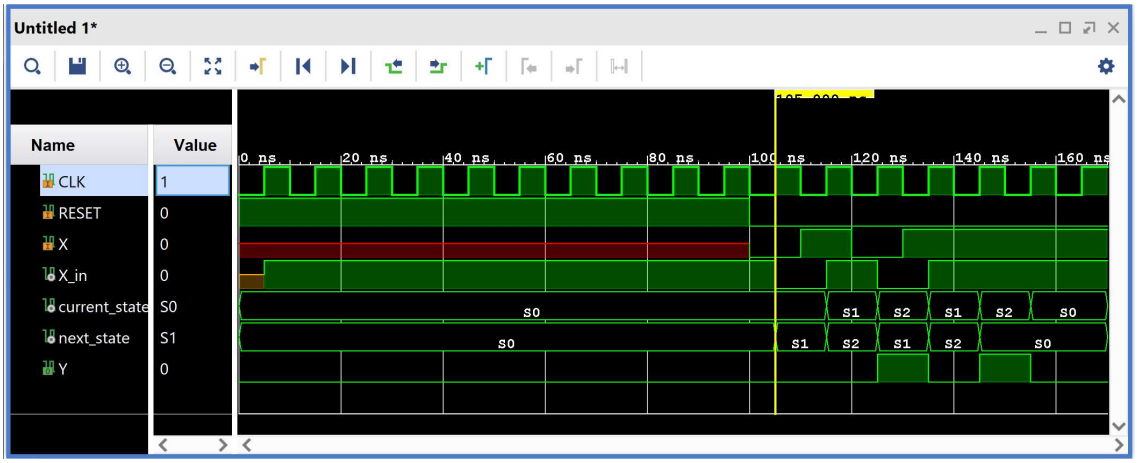
3-4.4. Επιλέξτε όλα τα διαθέσιμα σήματα στο παράθυρο *Objects* και κάντε drag and drop στο παράθυρο με τα διαγράμματα χρονισμού. Συμπεριλάβετε το εσωτερικό σήμα **X_in** (που είναι συγχρονισμένο στην ανερχόμενη ακμή του CLK ως έξοδος του καταχωρητή εισόδων INREG) κάτω από την είσοδο X, καθώς και τα εσωτερικά σήματα **current_state** και **next_state** που βοηθάνε στην αποσφαλμάτωση.



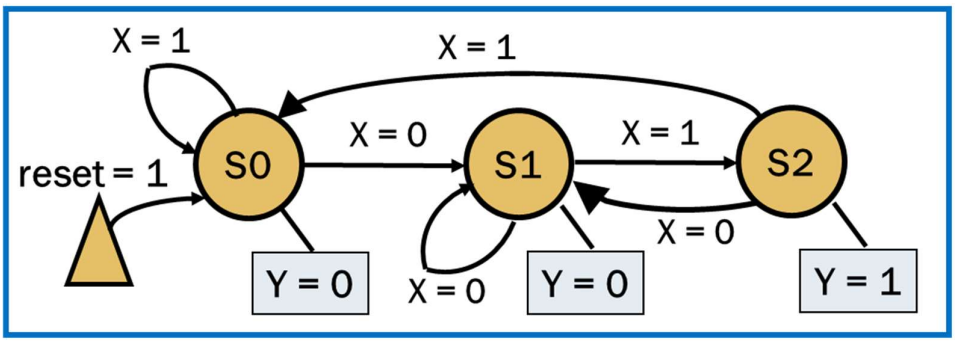
3-4.5. Επαναλάβετε τη διαδικασία της προσομοίωσης από την αρχή με επιλογή του κουμπιού Restart και στη συνέχεια του κουμπιού Run All. Και τα δύο κουμπιά βρίσκονται στην οριζόντια μπάρα στο πάνω μέρος.



Εάν δεν είναι εμφανές το παράθυρο *Untitled*, επιλέξτε το *Untitled*. Βλέπετε ολόκληρο το διάγραμμα χρονισμού με κατάλληλο **zoom out** ή επιλέγοντας το **zoom fit** . Για να επαναφέρετε το floating παράθυρο πίσω, απλά επιλέξτε το κουμπί Dock Window.



3-4.6. Συγκρίνετε το διάγραμμα χρονισμού με το διάγραμμα μεταβολής κατάστασης.



3-4.7. Κλείστε τον simulator επιλέγοντας το κουμπί × πάνω δεξιά στο παράθυρο του *SIMULATION*. Στο παράθυρο *Confirm Close* που εμφανίζεται πατήστε **OK**. Στο παράθυρο *Save Waveform Configuration* πατήστε **Save** για να αποθηκεύσετε το configuration του διαγράμματος χρονισμού στο οποίο έχετε καταλήξει, αλλιώς πατήστε **Discard**. Εάν πατήσετε **Save**, στο παράθυρο *Save Waveform* που εμφανίζεται διατηρείστε το όνομα **PATTERN_FSM_TB_behav.wcfg** και πατήστε **Save**. Στο παράθυρο *Waveform Configuration File* που εμφανίζεται πατήστε **Yes**.

Η διαδικασία που ήδη περιγράψαμε στα Βήματα 3-3 και 3-4 της «**Εισαγωγής του VHDL testbench και προσομοίωσης συμπεριφοράς**» εφαρμόζεται παρομοίως σε κάθε πιθανή μηχανή πεπερασμένων καταστάσεων (FSM), όπως αυτή που απαιτείται στη μονάδα ελέγχου του επεξεργαστή πολλών κύκλων.

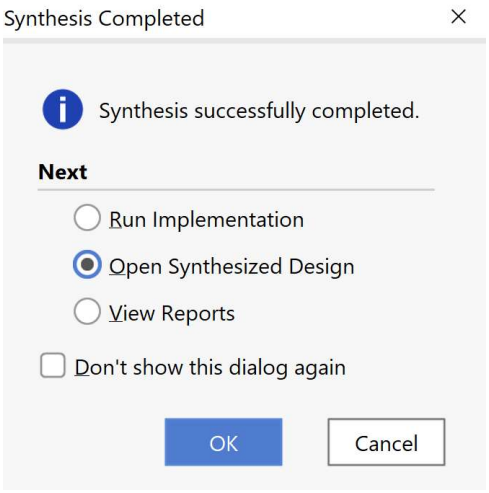
Βήμα 4: Σύνθεση του κώδικα VHDL και προσομοίωση (λογική, χρονική)

4-1. Εκτέλεση της διαδικασίας της σύνθεσης και ανάλυση των αποτελεσμάτων μετά τη σύνθεση στο VIVADO IDE για συνδυαστική λογική ανάμεσα σε καταχωρητές

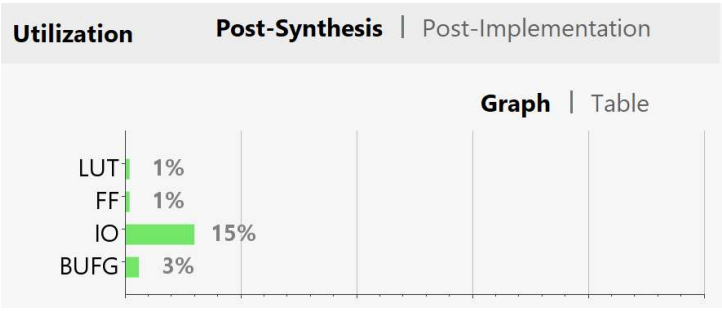
Η σύνθεση εκτελείται στην οντότητα **ADDER_REG_8** που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design resources, καθώς και σε όλα τα υπάρχοντα αρχεία της ιεραρχίας. Με τη σύνθεση το εργαλείο Vivado IDE παράγει το **synthesized design model** της οντότητας **ADDER_REG_8**.

4-1.1. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Synthesis**. Πατήστε **OK** στα παράθυρα προειδοποίησης που εμφανίζονται.

4-1.2. Στο παράθυρο *Synthesis Completed* υπάρχουν τρεις επιλογές. Επιλέξτε το **Open Synthesized Design** και πατήστε **OK** για να μελετήσετε το αποτέλεσμα της σύνθεσης πριν προχωρήσετε στο στάδιο της υλοποίησης (implementation). Πατήστε **Yes** για να κλείσετε το elaborated design, εάν εμφανιστεί το παράθυρο *Close Design*.



4-1.3. Αρχικά, επιλέξτε το παράθυρο *Project Summary* και μελετήστε τα διάφορα υποπαράθυρα. Εάν δεν το βλέπετε επιλέξτε το εικονίδιο Project Summary Σ . Μελετήστε στο υποπαράθυρο Utilization τους πόρους που χρησιμοποιεί η οντότητα **ADDER_REG_8** σε μορφή *Graph* και σε μορφή *Table* (στο κάτω μέρος αριστερά).

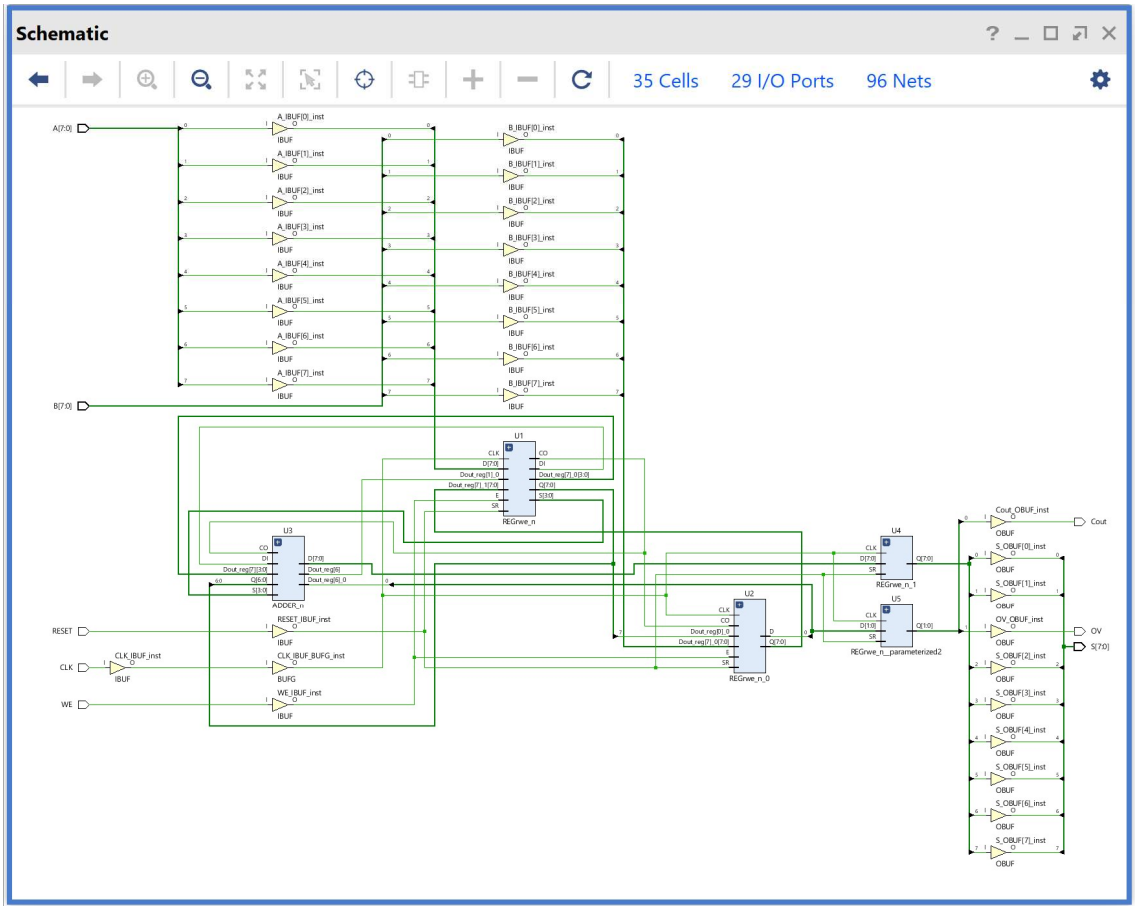


Utilization **Post-Synthesis** | Post-Implementation

Graph | **Table**

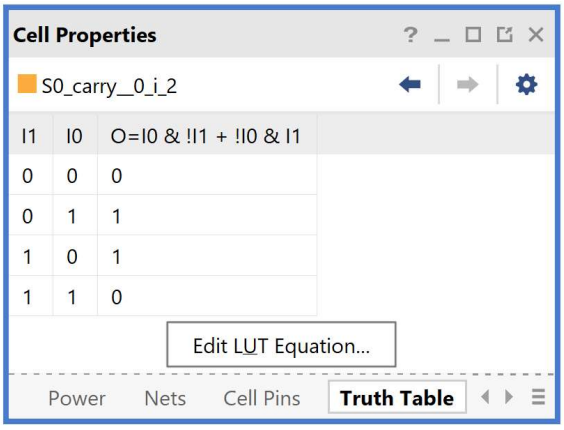
Resource	Estimation	Available	Utilizatio...
LUT	10	53200	0.02
FF	26	106400	0.02
IO	29	200	14.50
BUFG	1	32	3.13

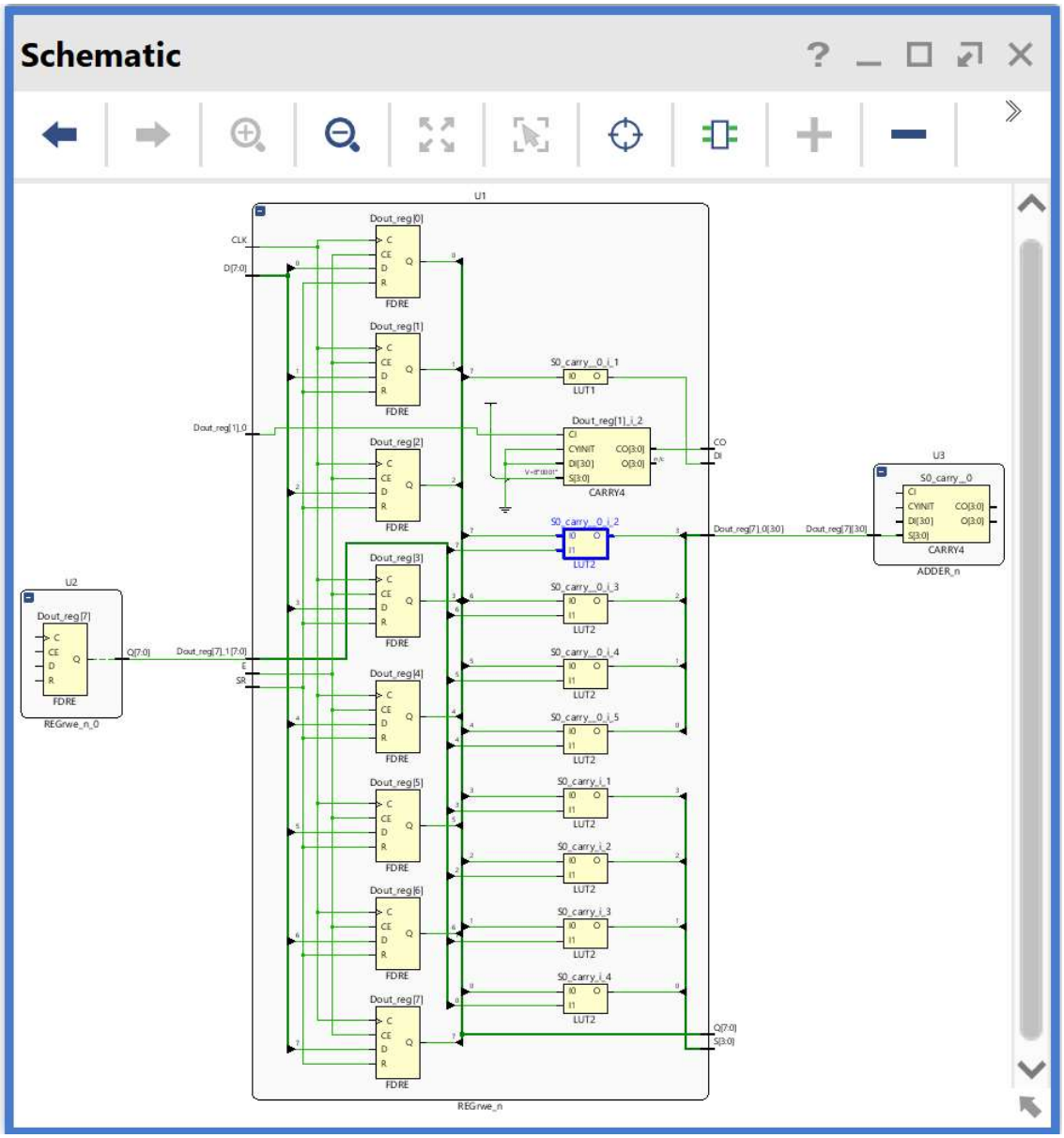
4-1.4. Στη συνέχεια, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Synthesized Design* επιλέξετε το **Schematic** για να δείτε το σχηματικό διάγραμμα του **synthesized design model**.



Παρατηρείστε ότι έχουν αυτόματα προστεθεί τα απαραίτητα IBUFs, OBUFs και BUFG primitives στο σχηματικό διάγραμμα, καθώς και ότι οι είσοδοι και οι έξοδοι από το FPGA είναι buffered. Επίσης, παρατηρείστε ότι έχει διατηρηθεί εν μέρει η ιεραρχία που έχει ορισθεί στην οντότητα **ADDER_REG_8**. Πατήστε το (+) σε όλες τις υπομονάδες U1-U5 και μελετήστε τις μία προς μία.

Υπομονάδα U1 (REGrwe_n): Συμπεριλαμβάνει τον καταχωρητή εισόδου A των 8 bit, 8 πύλες XOR (LUT2), έναν αντιστροφέα (LUT1) και μία μονάδα CARRY4. Επαληθεύστε τον πίνακα αλήθειας και την εξίσωση Boole ενός LUT, επιλέγοντας το συγκεκριμένο LUT2 (που αποκτά έντονο μπλε περίγραμμα) και ρυθμίζοντας την επιλογή *Truth Table* στο παράθυρο *Cell Properties*. Επίσης, φαίνονται οι είσοδοι και οι έξοδοι του συγκεκριμένου LUT2.





Υπομονάδα U2 (REGrwe_n_0): Συμπεριλαμβάνει τον καταχωρητή εισόδου B των 8 bit, 1 πύλη XOR (LUT2) και μία μονάδα CARRY4.

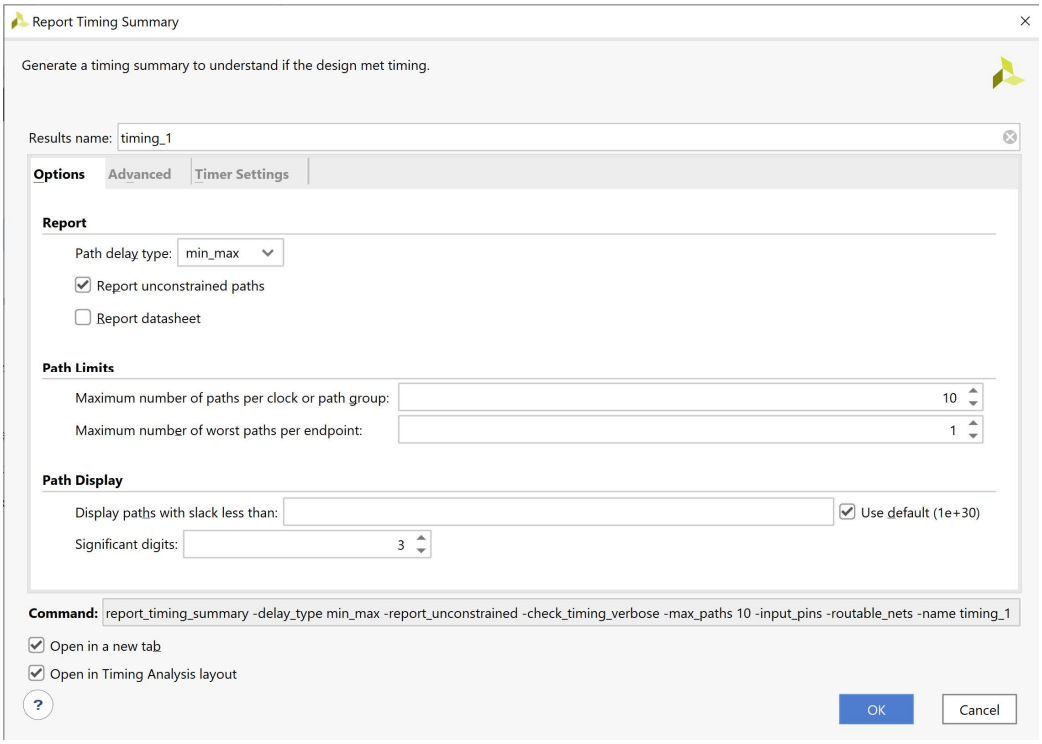
Υπομονάδα U3 (ADDER_n): Συμπεριλαμβάνει 1 πύλη XOR (LUT2) και 2 μονάδες CARRY4.

Υπομονάδα U4 (REGrwe_n_1): Συμπεριλαμβάνει τον καταχωρητή εξόδου S των 8 bit.

Υπομονάδα U5 (REGrwe_n_parameterized2): Συμπεριλαμβάνει τον καταχωρητή εξόδου των 2 bit για τις σημαίες Cout και OV.

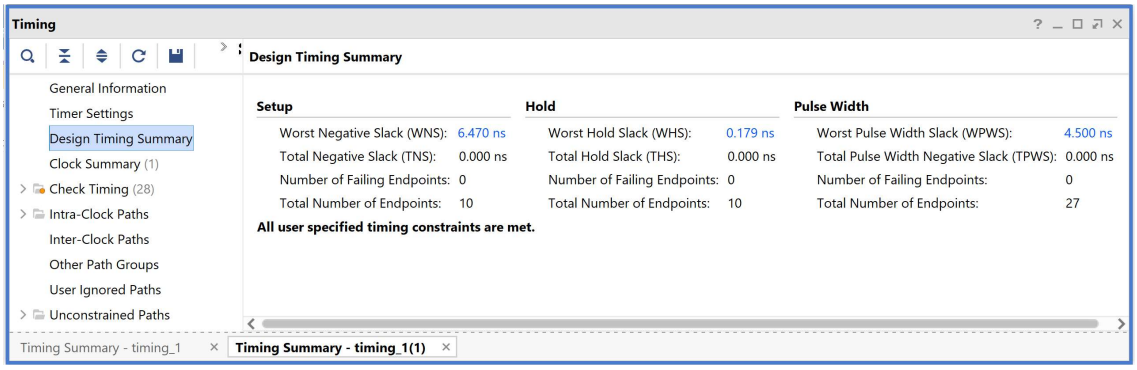
Προσοχή! Οι υπομονάδες **U1**, **U2** και **U3** είναι διαφορετικές από πλευράς λογικής στο **synthesized design model**, που προκύπτει μετά τη σύνθεση, σε σχέση με το **elaborated design model**. Αντίθετα, οι υπομονάδες **U4** και **U5** παραμένουν ίδιες από πλευράς λογικής.

4-1.5. Στη συνέχεια, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Synthesized Design* επιλέξετε το **Report Timing Summary** για να δείτε την ανάλυση χρονισμού που κάνει το εργαλείο Vivado IDE στο **synthesized design model**. Αν και η ανάλυση χρονισμού σε αυτό το επίπεδο δεν είναι ακριβής, χρησιμοποιείται ευρέως στις πολύπλοκες σχεδιάσεις για εξοικονόμηση χρόνου κατά την ανάπτυξη του κώδικα VHDL.



Στην επιλογή *Path delay type* ορίζεται ο τύπος της ανάλυσης που θα εκτελεσθεί. Το *max delay analysis* αφορά στην εύρεση της κρίσιμης διαδρομής (με τη μεγαλύτερη καθυστέρηση διάδοσης) και συνεπώς στην εύρεση της μέγιστης συχνότητας λειτουργίας χωρίς την παραβίαση του **χρόνου σταθεροποίησης** (setup time). Το *min delay analysis* αφορά στην εύρεση της σύντομης διαδρομής (με τη μικρότερη καθυστέρηση διάδοσης) χωρίς την παραβίαση του **χρόνου διατήρησης** (hold time).

4-1.6. Πατήστε **OK** για να παραχθεί το Timing_1 report.



Στη στήλη **Setup** παρουσιάζονται τα αποτελέσματα του *max delay analysis*.

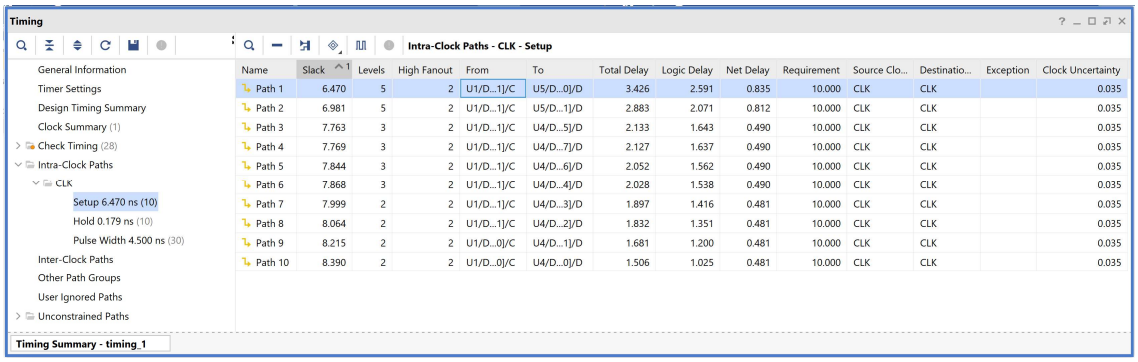
- Το *Worst Negative Slack* (**WNS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των κρίσιμων διαδρομών του max delay analysis. Μπορεί να είναι θετικό η αρνητικό. Ένα θετικό slack (6.470 ns) δηλώνει ότι η κρίσιμη διαδρομή ικανοποιεί την περίοδο του CLK. Ένα αρνητικό slack δηλώνει ότι παραβιάζεται ο χρόνος σταθεροποίησης (setup time).
- Το *Total Negative Slack* (**TNS**) είναι το άθροισμα όλων των αρνητικών WNS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου σταθεροποίησης. Το ψηφιακό κύκλωμα λειτουργεί στην επιλεγμένη συχνότητα λειτουργίας.
- Το *Number of Failing Endpoints* αφορά στον συνολικό αριθμό των endpoints που έχουν παραβιάσει το χρόνο σταθεροποίησης (αρνητικό WNS).
- Το *Total Number of Endpoints* αφορά στον συνολικό αριθμό των endpoints που έχουν αναλυθεί.

Στη στήλη **Hold** παρουσιάζονται τα αποτελέσματα του min delay analysis.

- Το *Worst Hold Slack* (**WHS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των σύντομων διαδρομών του min delay analysis. Μπορεί να είναι θετικό η αρνητικό. Ένα αρνητικό slack δηλώνει ότι παραβιάζεται ο χρόνος διατήρησης (hold time).
- Το *Total Hold Slack* (**THS**) είναι το άθροισμα όλων των αρνητικών WHS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου διατήρησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά.
- Το *Number of Failing Endpoints* αφορά στον συνολικό αριθμό των endpoints που έχουν παραβιάσει το χρόνο διατήρησης (αρνητικό WHS).
- Το *Total Number of Endpoints* αφορά στον συνολικό αριθμό των endpoints που έχουν αναλυθεί.

Στη στήλη **Pulse Width** παρουσιάζονται τα περιθώρια του σήματος CLK, όταν είναι HIGH ή LOW.

4-1.7. Επιλέξτε το **WNS link** και δείτε τις 10 χειρότερες κρίσιμες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο σταθεροποίησης. Η κρίσιμη διαδρομή έχει καθυστέρηση διάδοσης 3.426 ns, εκ των οποίων τα 2.591 ns αφορούν στη λογική (logic), ενώ τα 0.835 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.035 ns. Τα επίπεδα λογικής (logic level) είναι 5.

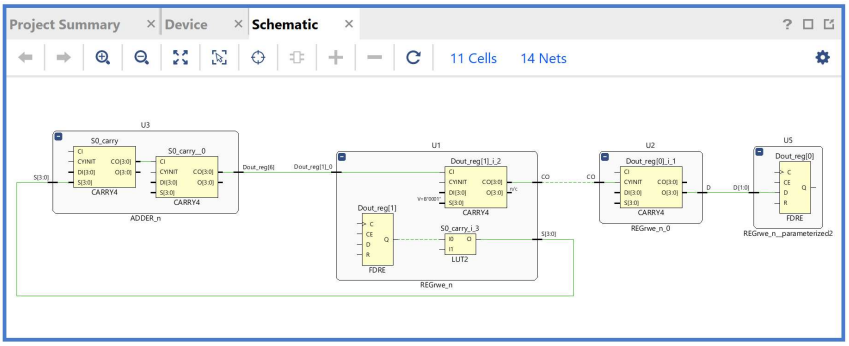


4-1.8. Επιλέξτε με διπλό κλικ το **Path 1**, ώστε να εμφανιστεί το παράθυρο *Path 1 – timing_1*. Η κρίσιμη διαδρομή περνάει από 4 μονάδες CARRY4 και 1 LUT2 (πύλη XOR). Υπολογίστε το WNS slack:

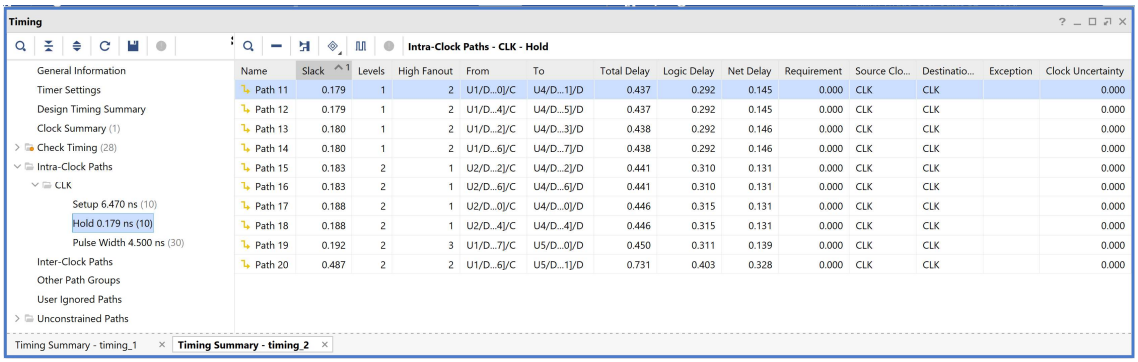
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U1) είναι 2.975 ns,
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή της υπομονάδας U1 και μέσω του LUT2 (U1) και των 4 μονάδων CARRY4 (U3-U1-U2) μέχρι την είσοδο D του καταχωρητή της υπομονάδας U5) είναι 3.426 ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι 6.401 ns,
- το **Required Time**, ως η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή 10.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U5) συν τον χρόνο σταθεροποίησης είναι 12.871 ns.

WNS slack = Required Time – Arrival Time = 12.871 – 6.401 = 6.470 ns

4-1.9. Επιλέξτε με δεξί κλικ στο **Path 1**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της κρίσιμης διαδρομής της οντότητας **ADDER_REG_8**.



4-1.10. Επιλέξτε το **WHS link** και δείτε τις 10 χειρότερες σύντομες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο διατήρησης. Η σύντομη διαδρομή έχει καθυστέρηση μόλυνσης 0.437 ns, εκ των οποίων τα 0.292 ns αφορούν στη λογική (logic), ενώ τα 0.145 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.000 ns. Τα επίπεδα λογικής (logic level) είναι 1.



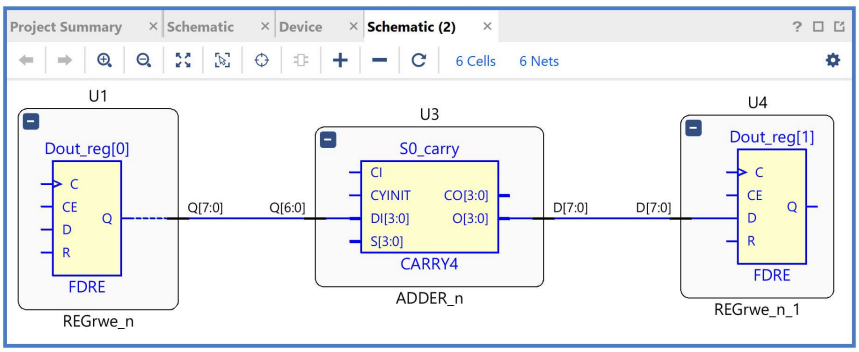
Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clo...	Destinatio...	Exception	Clock Uncertainty
Path 11	0.179	1	2	U1/D...0]/C	U4/D...1]/D	0.437	0.292	0.145	0.000	CLK	CLK		0.000
Path 12	0.179	1	2	U1/D...4]/C	U4/D...5]/D	0.437	0.292	0.145	0.000	CLK	CLK		0.000
Path 13	0.180	1	2	U1/D...2]/C	U4/D...3]/D	0.438	0.292	0.146	0.000	CLK	CLK		0.000
Path 14	0.180	1	2	U1/D...6]/C	U4/D...7]/D	0.438	0.292	0.146	0.000	CLK	CLK		0.000
Path 15	0.183	2	1	U2/D...2]/C	U4/D...2]/D	0.441	0.310	0.131	0.000	CLK	CLK		0.000
Path 16	0.183	2	1	U2/D...6]/C	U4/D...6]/D	0.441	0.310	0.131	0.000	CLK	CLK		0.000
Path 17	0.188	2	1	U2/D...0]/C	U4/D...0]/D	0.446	0.315	0.131	0.000	CLK	CLK		0.000
Path 18	0.188	2	1	U2/D...4]/C	U4/D...4]/D	0.446	0.315	0.131	0.000	CLK	CLK		0.000
Path 19	0.192	2	3	U1/D...7]/C	U5/D...0]/D	0.450	0.311	0.139	0.000	CLK	CLK		0.000
Path 20	0.487	2	2	U1/D...6]/C	U5/D...1]/D	0.731	0.403	0.328	0.000	CLK	CLK		0.000

4-1.11. Επιλέξτε με διπλό κλικ το **Path 11**, ώστε να εμφανιστεί το παράθυρο *Path 11 – timing_1*. Η σύντομη διαδρομή περνάει από 1 μονάδα CARRY4. Υπολογίστε το WHS slack:

- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U1) είναι 0.735 ns,
- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή της υπομονάδας U1 και μέσω της 1 μονάδας CARRY4 (U3) μέχρι την είσοδο D του καταχωρητή της υπομονάδας U4) είναι 0.438 ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι 1.173 ns,
- το **Required Time**, ως η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U4) συν τον χρόνο διατήρησης, είναι 0.993 ns.

WHS slack = Arrival Time – Required Time = 1.173 – 0.993 = 0.180 ns

4-1.12. Επιλέξτε με δεξί κλικ στο **Path 11**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της σύντομης διαδρομής της οντότητας **ADDER_REG_8**.



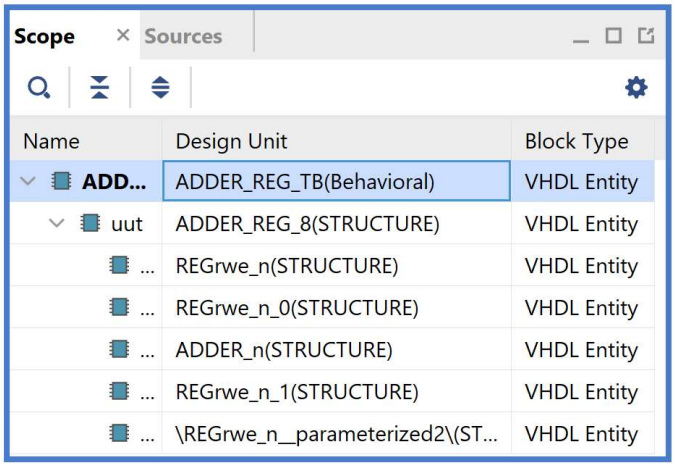
4-2. Εκτέλεση προσομοίωσης μετά τη σύνθεση (λογική και χρονική) στο VIVADO IDE για συνδυαστική λογική ανάμεσα σε καταχωρητές

Η προσομοίωση μετά τη σύνθεση (λογική και χρονική) εκτελείται στην οντότητα **ADDER_REG_8_TB** του προγράμματος δοκιμών (testbench) που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των simulation resources. Κατά την προσομοίωση, η οντότητα **ADDER_REG_8_TB** καλεί το *UUT* της (που είναι το **synthesized design model** της οντότητας **ADDER_REG_8**).

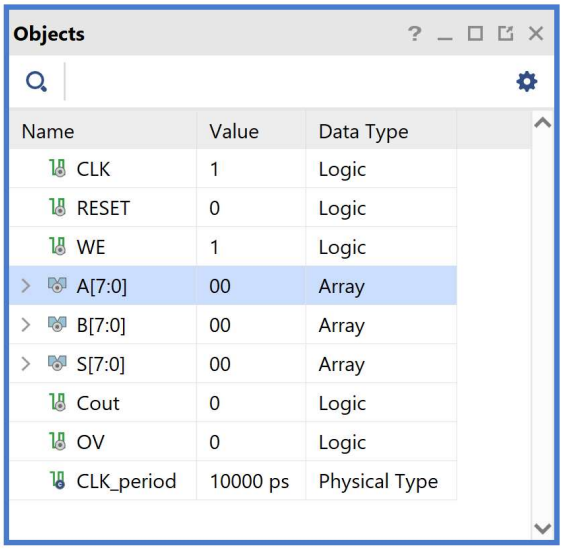
4-2.1. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Simulation**, ώστε να εμφανιστούν όλες οι πιθανές προσομοιώσεις που υποστηρίζει το Vivado IDE. Επιλέξτε **Run Post-Synthesis Functional Simulation**.

4-2.2. Το πρόγραμμα δοκιμής (testbench) και όλες οι οντότητες του *UUT* θα γίνουν compiled και θα τρέξει το Vivado simulator (εφόσον βέβαια δεν υπάρχουν σφάλματα). Θα εμφανιστεί το παράθυρο *SIMULATION* που απαρτίζεται από 4 παράθυρα:

Το παράθυρο *Scope*, όπου παρουσιάζεται η οντότητα **ADDER_REG_8_TB**, το **synthesized design model** της οντότητας **ADDER_REG_8** (*UUT*) καθώς και οι υπόλοιπες νέες οντότητες του *UUT* που προκύπτουν μετά τη σύνθεση. Όλες οι οντότητες που απαρτίζουν πλέον το *UUT* είναι *structural*.

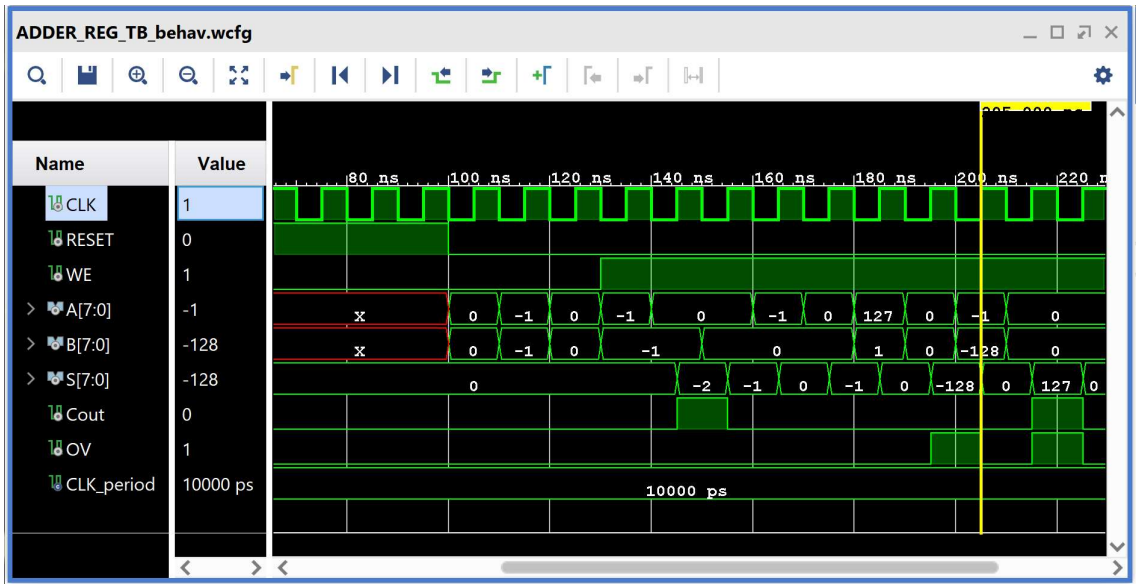


Το παράθυρο *Objects*, όπου εμφανίζονται τα σήματα top-level, δηλαδή οι είσοδοι και οι έξοδοι της οντότητας **ADDER_REG_8**, που είναι η κορυφαία οντότητα της ιεραρχίας του *UUT*, καθώς και η περίοδος του CLK (*CLK_period*). Οι αρτηρίες (array) αναλύονται στα σήματα που τις απαρτίζουν. Οι τιμές αντιστοιχούν στις τιμές που έχει σταματήσει η προσομοίωση της οντότητας **ADDER_REG_8_TB** (st*op (2)*).

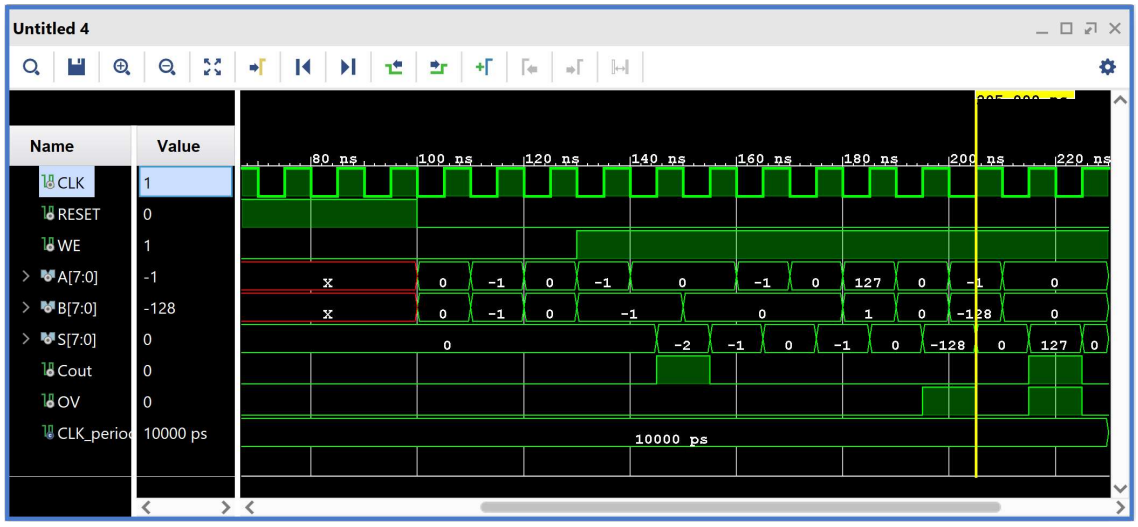


Το παράθυρο *Tcl Console* με όλες τις διαδικασίες που εκτελούνται στο πλαίσιο της προσομοίωσης. Το Tcl Console καθαρίζει με την επιλογή *Clear*.

Επιλέξτε το παράθυρο *ADDER_REG_TB_behav.wcfg*, που δημιουργήσατε για την προσομοίωση συμπεριφοράς. Βλέπετε ολόκληρο το διάγραμμα χρονισμού της λογικής προσομοίωσης μετά τη σύνθεση με κατάλληλο **zoom out** ή επιλέγοντας το **zoom fit** . Για να επαναφέρετε το floating παράθυρο πίσω, απλά επιλέξτε το κουμπί Dock Window. Μελετήστε μόνο top-level εισόδους/εξόδους.

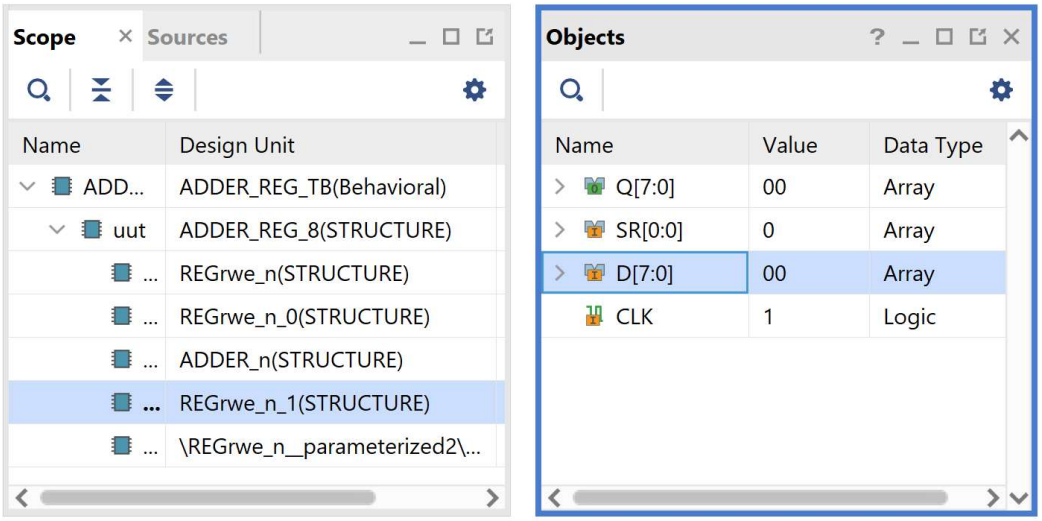


4-2.3. Συγκρίνετε τα διαγράμματα χρονισμού της λογικής προσομοίωσης μετά τη σύνθεση και της προσομοίωσης συμπεριφοράς (υπάρχει ταύτιση στο επίπεδο των top-level εισόδων/εξόδων).

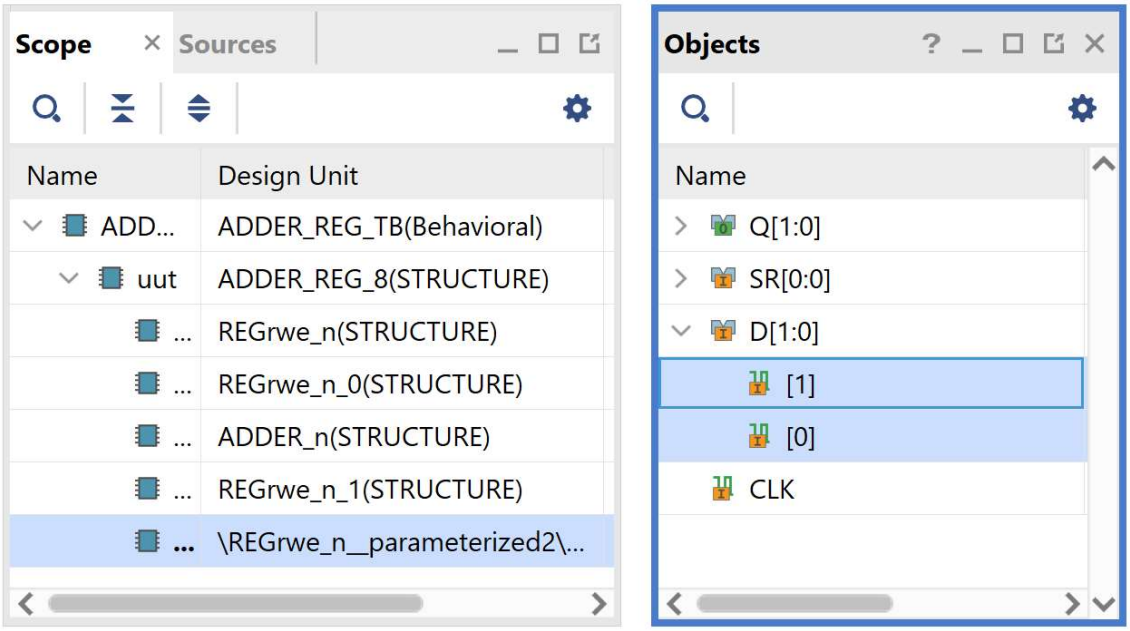


4-2.4. Προσθέστε περισσότερα **εσωτερικά σήματα** στο διάγραμμα χρονισμού της προσομοίωσης μετά τη σύνθεση. Μετά από μελέτη του σχηματικού διαγράμματος του **synthesized design model** προκύπτει ότι οι είσοδοι **D[7:0]** και **D[1:0]** των υπομονάδων **U4** και **U5**, αντίστοιχα, αντιστοιχούν στις εξόδους **S[7:0]** και **Cout**, **OV** της οντότητας **Adder_n** (του **elaborated design model** πριν τη σύνθεση) και για αυτό το λόγο τις προσθέτουμε στο διάγραμμα χρονισμού της προσομοίωσης μετά τη σύνθεση.

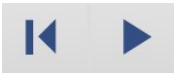
Στο παράθυρο *Scope* επιλέξτε την οντότητα **REGrwe_n_1**. Παρατηρείστε ότι στο παράθυρο Objects εμφανίζονται τα σήματα της οντότητας **REGrwe_n_1**. Επιλέξτε το εσωτερικό σήμα **D[7:0]** (που αντιστοιχεί στο S[7:0]) και κάντε drag and drop στο παράθυρο με τα διαγράμματα χρονισμού, τοποθετώντας το κάτω από το υπάρχουν top-level σήμα **B[7:0]**.



Στο παράθυρο *Scope* επιλέξτε την οντότητα **REGrwe_n_parameterized2**. Παρατηρείστε ότι στο παράθυρο Objects εμφανίζονται τα σήματα αυτής της οντότητας. Επιλέξτε τα εσωτερικά σήματα **D[0]** (που αντιστοιχεί στο Cout) και **D[1]** (που αντιστοιχεί στο OV) και κάντε drag and drop στο παράθυρο με τα διαγράμματα χρονισμού, τοποθετώντας το κάτω από το σήμα **D[7:0]**.

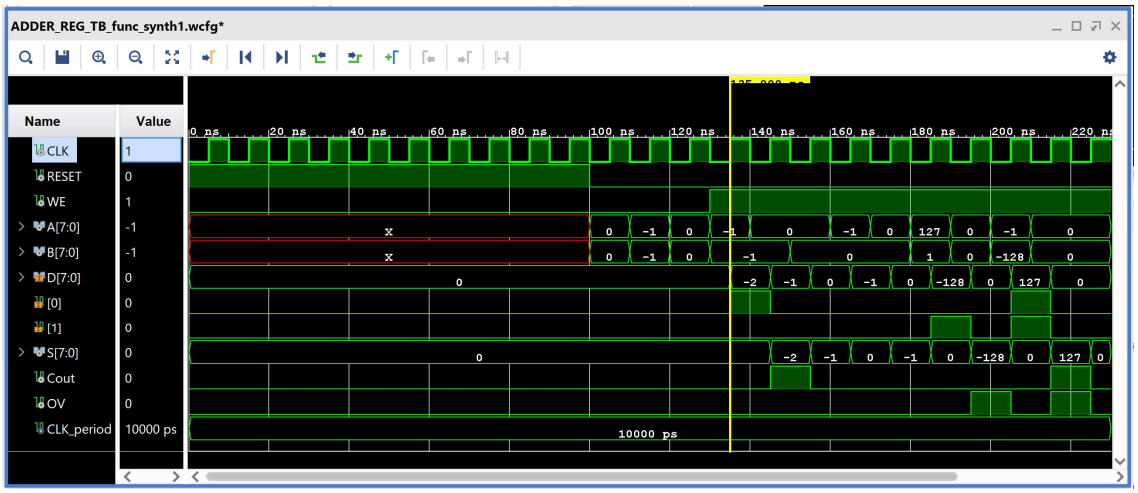


Επαναλάβετε τη διαδικασία της προσομοίωσης από την αρχή με επιλογή του κουμπιού **Restart** και στη συνέχεια του κουμπιού **Run All**. Και τα δύο κουμπιά βρίσκονται στην οριζόντια μπάρα στο πάνω μέρος.

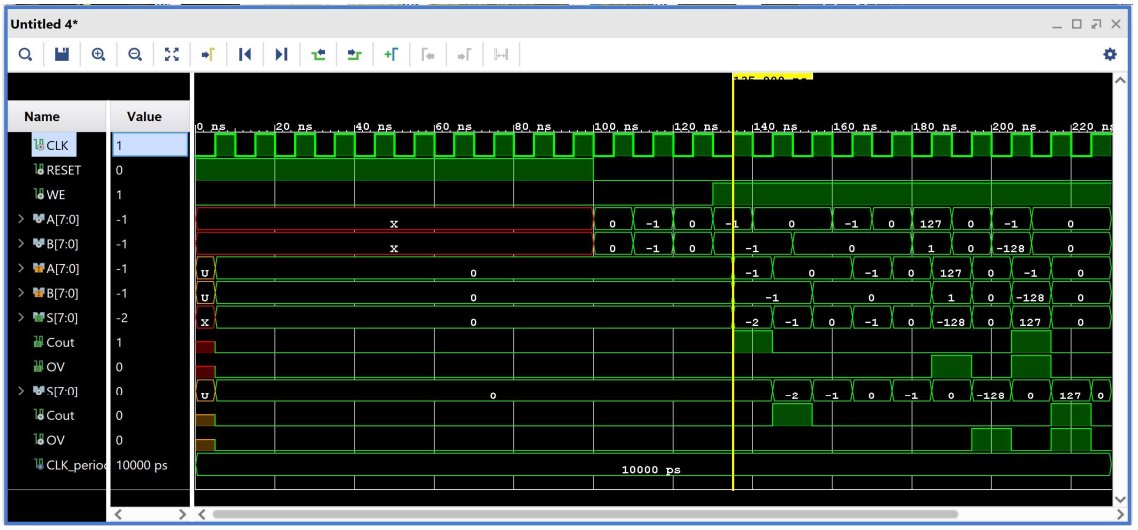


Το εσωτερικό σήμα **D[7:0]** του διαγράμματος χρονισμού της προσομοίωσης μετά τη σύνθεση αντιστοιχεί στο εσωτερικό σήμα **S[7:0]** του διαγράμματος χρονισμού της προσομοίωσης συμπεριφοράς. Ομοίως, το εσωτερικό σήμα **[0]** αντιστοιχεί στο **Cout** και το εσωτερικό σήμα **[1]** αντιστοιχεί στο **OV**.

Διάγραμμα χρονισμού της λογικής προσομοίωσης μετά τη σύνθεση:



Διάγραμμα χρονισμού της προσομοίωσης συμπεριφοράς:



4-2.5. Συγκρίνετε τη λογική προσομοίωση μετά τη σύνθεση με την προσομοίωση συμπεριφοράς (υπάρχει ταύτιση στο επίπεδο εισόδων/εξόδων και των επιλεγμένων εσωτερικών σημάτων).

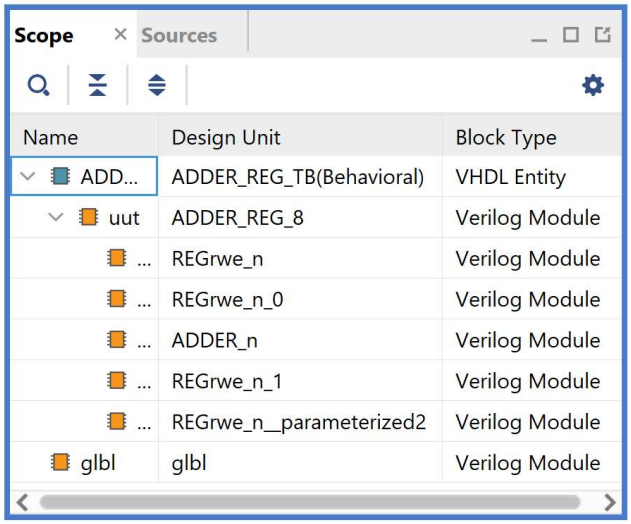
4-2.6. Επιλέξτε αρχικά **File** στη συνέχεια **Simulation Waveform** και τέλος **Save Configuration as ..** για να αποθηκεύσετε το configuration του διαγράμματος χρονισμού στο οποίο έχετε καταλήξει με νέο όνομα (π.χ. **ADDER_REG_TB_func_synth1.wcfg**). Στο παράθυρο *Waveform Configuration File* που εμφανίζεται πατήστε **Yes**.

4-2.7. Κλείστε τον simulator επιλέγοντας το κουμπί × πάνω δεξιά στο παράθυρο του *SIMULATION*. Στο παράθυρο *Confirm Close* που εμφανίζεται πατήστε **OK**.

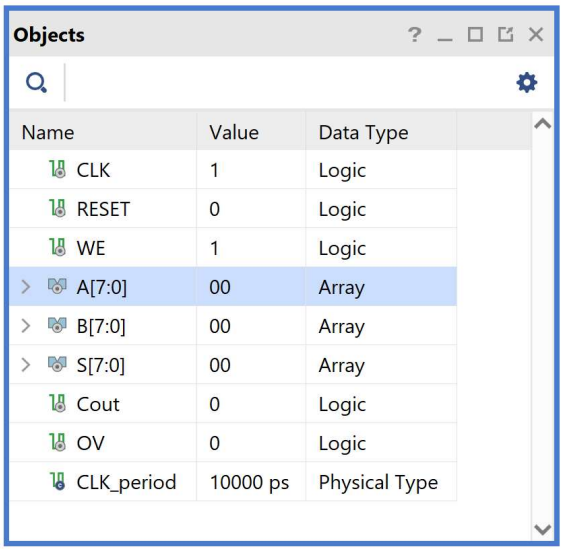
4-2.8. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Simulation**, ώστε να εμφανιστούν όλες οι πιθανές προσομοιώσεις που υποστηρίζει το Vivado IDE. Επιλέξτε **Run Post-Synthesis Timing Simulation**.

4-2.9. Το πρόγραμμα δοκιμής (testbench) και όλες οι οντότητες του *UUT* θα γίνουν compiled και θα τρέξει το Vivado simulator (εφόσον βέβαια δεν υπάρχουν σφάλματα). Θα εμφανιστεί το παράθυρο *SIMULATION* που απαρτίζεται από 4 παράθυρα:

Το παράθυρο *Scope*, όπου παρουσιάζεται η οντότητα **ADDER_REG_8_TB**, το **synthesized design model** της οντότητας **ADDER_REG_8** (*UUT*) καθώς και οι υπόλοιπες νέες οντότητες του *UUT* που προκύπτουν μετά τη σύνθεση για χρονική προσομοίωση. Όλες οι οντότητες που απαρτίζουν πλέον το *UUT* είναι *Verilog Module!*



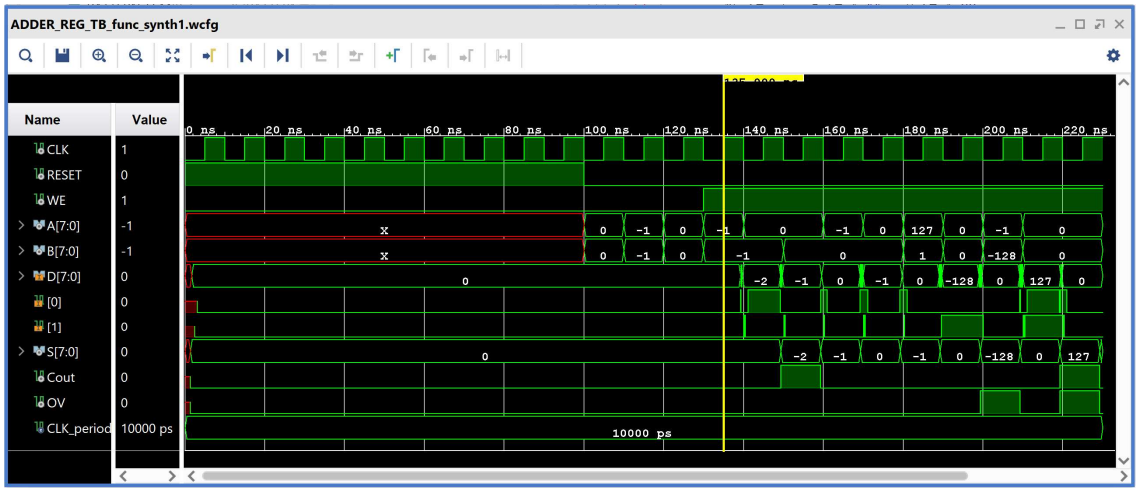
Το παράθυρο *Objects*, όπου εμφανίζονται τα σήματα top-level, δηλαδή οι είσοδοι και οι έξοδοι της οντότητας **ADDER_REG_8**, που είναι η κορυφαία οντότητα της ιεραρχίας του *UUT*, καθώς και η περίοδος του CLK (*CLK_period*). Οι αρτηρίες (array) αναλύονται στα σήματα που τις απαρτίζουν. Οι τιμές αντιστοιχούν στις τιμές που έχει σταματήσει η προσομοίωση της οντότητας **ADDER_REG_8_TB** (*stop (2)*).



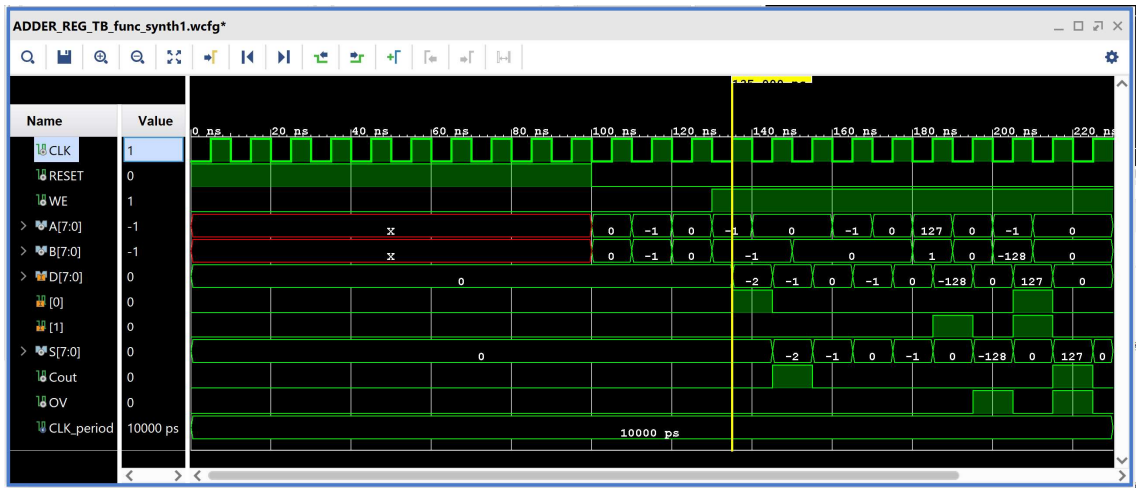
Το παράθυρο *Tcl Console* με όλες τις διαδικασίες που εκτελούνται στο πλαίσιο της προσομοίωσης. Το Tcl Console καθαρίζει με την επιλογή *Clear*.

Επιλέξτε το παράθυρο *ADDER_REG_TB_func_synth1.wcfg*. Βλέπετε ολόκληρο το διάγραμμα χρονισμού της χρονικής προσομοίωσης μετά τη σύνθεση με κατάλληλο **zoom out** ή επιλέγοντας το **zoom fit** . Για να επαναφέρετε το floating παράθυρο πίσω, απλά επιλέξτε το κουμπί Dock Window.

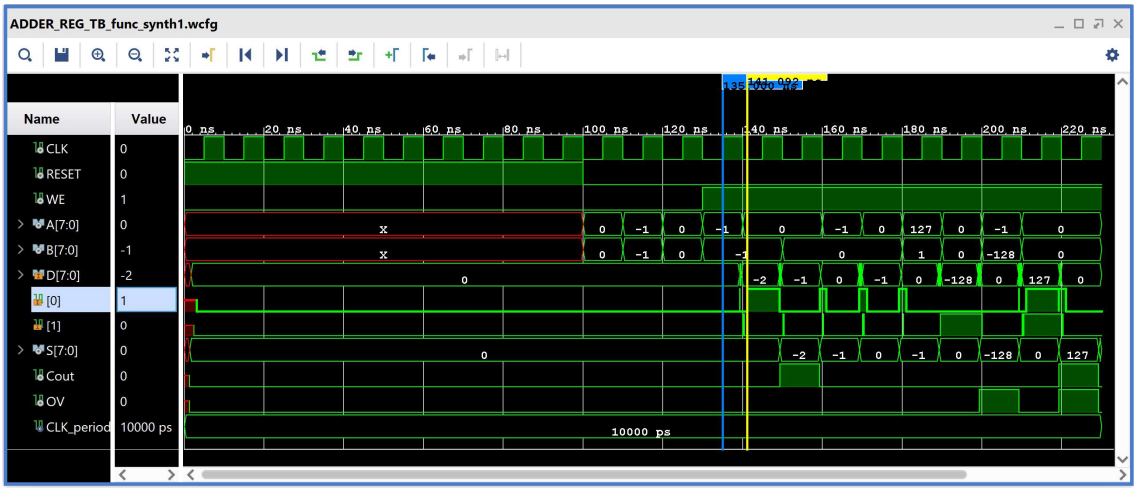
Προσοχή! Εάν δεν θέλετε να βλέπετε άλλα ενεργοποιημένα παράθυρα διαγραμμάτων χρονισμού κατά τη διάρκεια της προσομοίωσης μπορείτε να τα απενεργοποιήσετε. Στο παράθυρο *Sources* επιλέξτε με δεξί κλικ το παράθυρο διαγραμμάτων χρονισμού που θέλετε να απενεργοποιήσετε και πατήστε **Disable File**. Τα απενεργοποιημένα παράθυρα διαγραμμάτων χρονισμού μεταφέρονται στα *Disabled Sources*. Για να ενεργοποιήσετε πάλι κάποιο παράθυρο διαγραμμάτων χρονισμού επιλέξτε το με δεξί κλικ και πατήστε **Enable File**.



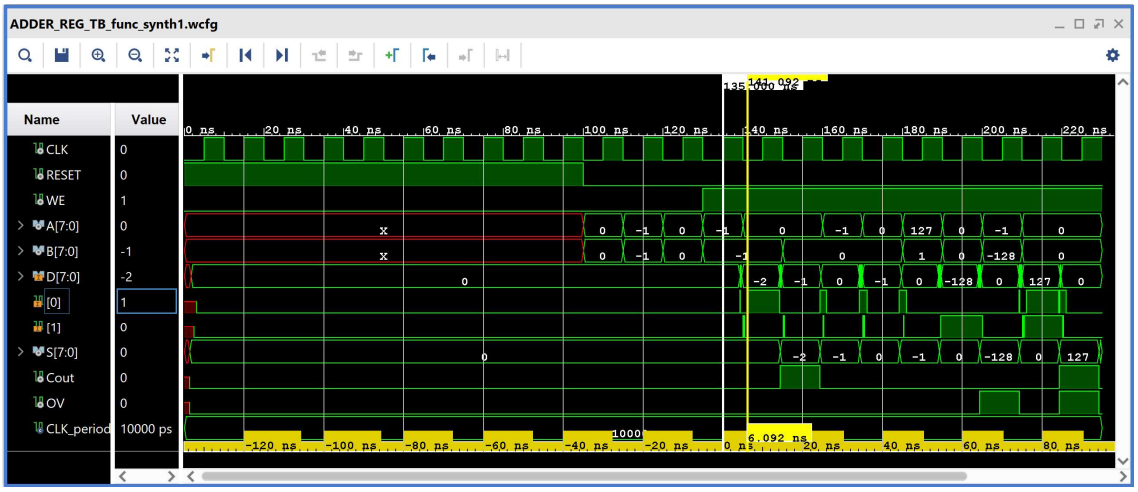
4-2.10. Συγκρίνετε τα διαγράμματα χρονισμού της χρονικής προσομοίωσης μετά τη σύνθεση και της λογικής προσομοίωσης μετά τη σύνθεση. Είναι εμφανείς οι καθυστερήσεις διάδοσης στο πρώτο διάγραμμα χρονισμού.



4-2.11. Υπολογίστε το **Arrival Time** ενός σήματος στο διάγραμμα χρονισμού της χρονικής προσομοίωσης μετά τη σύνθεση με τη χρήση των marker. Για παράδειγμα, βάλτε τον μπλε marker στην ανερχόμενη ακμή του **CLK** στα 135.000 ns και τον κίτρινο marker στην ανερχόμενη ακμή του εσωτερικού σήματος **[0]** (που αντιστοιχεί στο Cout) στα 141.092 ns. Απέχουν 6.092 ns. Συγκρίνετε με το **Arrival Time** που είχατε βρει κατά τη χρονική ανάλυση για την κρίσιμη διαδρομή, που ήταν 6.401 ns.



Με κλικ πάνω στον μπλε marker (γίνεται λευκός), ορίζουμε τη θέση του στα 0.000 ns και είναι πλέον εμφανές ότι ο κίτρινος marker απέχει 6.092 ns.



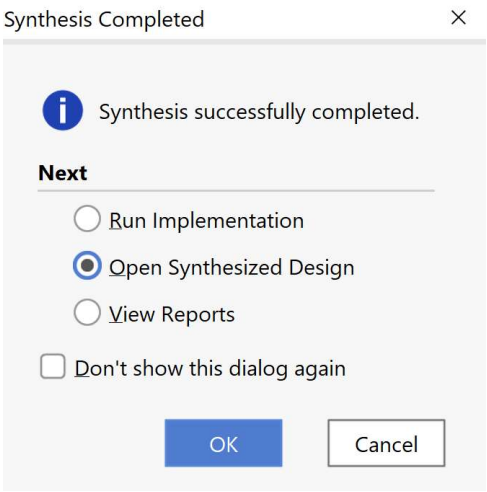
4-2.12. Κλείστε τον simulator επιλέγοντας το κουμπί ✕ πάνω δεξιά στο παράθυρο του *SIMULATION*. Στο παράθυρο *Confirm Close* που εμφανίζεται πατήστε **OK**. Εάν επιθυμείτε να μη σώσετε ένα επιπλέον waveform configuration, στο παράθυρο *Save Waveform Configuration* επιλέξτε **Discard**.

Η διαδικασία που ήδη περιγράψαμε στα Βήματα 4-1 και 4-2 της «**Σύνθεσης του κώδικα VHDL και προσομοίωση (λογική, χρονική)**» εφαρμόζεται παρομοίως σε κάθε πιθανή υπομονάδα συνδυαστικής λογικής της διαδρομής δεδομένων του επεξεργαστή.

4-3. Εκτέλεση της διαδικασίας της σύνθεσης και ανάλυση των αποτελεσμάτων μετά τη σύνθεση στο VIVADO IDE για μηχανή πεπερασμένων καταστάσεων (FSM)

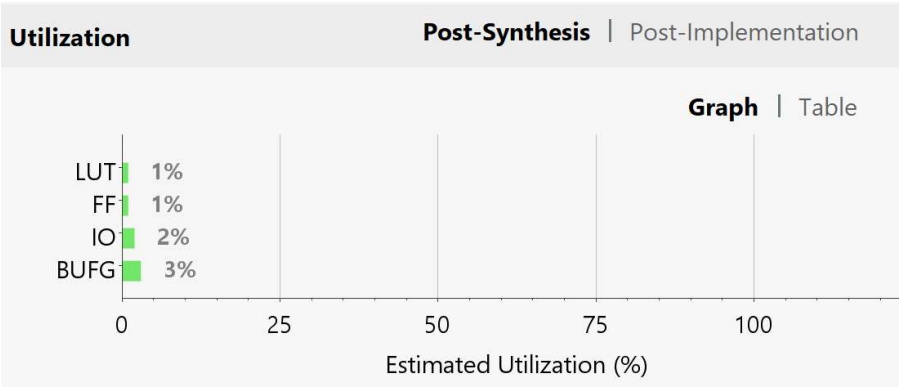
Η σύνθεση εκτελείται στην οντότητα **PATTERN_FSM** που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design resources, καθώς και σε όλα τα υπάρχοντα αρχεία της ιεραρχίας. Με τη σύνθεση το εργαλείο Vivado IDE παράγει το **synthesized design model** της οντότητας **PATTERN_FSM**.

4-3.1. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Synthesis**. Πατήστε **OK** στα παράθυρα προειδοποίησης που εμφανίζονται.



4-3.2. Στο παράθυρο *Synthesis Completed* υπάρχουν τρεις επιλογές. Επιλέξτε το **Open Synthesized Design** και πατήστε **OK** για να μελετήσετε το αποτέλεσμα της σύνθεσης πριν προχωρήσετε στο στάδιο της υλοποίησης (implementation). Πατήστε **Yes** για να κλείσετε το elaborated design, εάν εμφανιστεί το παράθυρο *Close Design*.

4-3.3. Αρχικά, επιλέξτε το παράθυρο *Project Summary* και μελετήστε τα διάφορα υποπαράθυρα. Εάν δεν το βλέπετε επιλέξτε το εικονίδιο Project Summary Σ . Μελετήστε στο υποπαράθυρο Utilization τους πόρους που χρησιμοποιεί η οντότητα **PATTERN_FSM** σε μορφή *Graph* και σε μορφή *Table* (στο κάτω μέρος αριστερά).

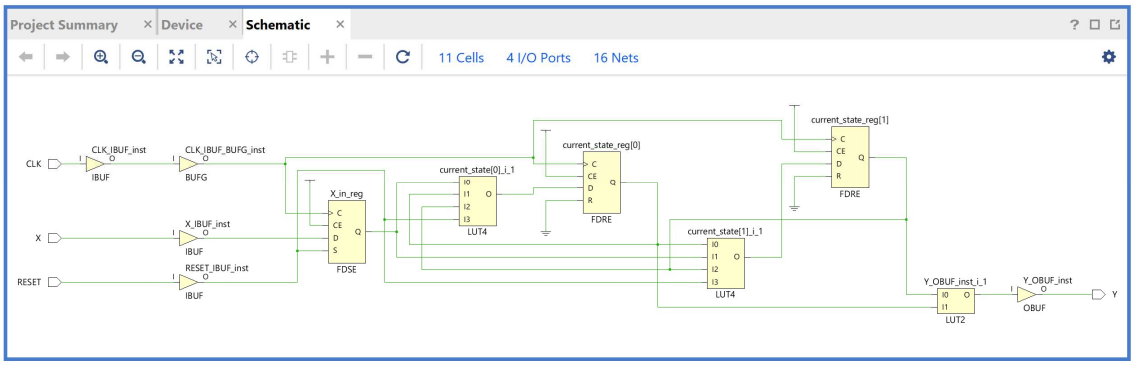


Utilization — **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	2	53200	0.01
FF	3	106400	0.01
IO	4	200	2.00
BUFG	1	32	3.13

4-3.4. Στη συνέχεια, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Synthesized Design* επιλέξετε το **Schematic** για να δείτε το σχηματικό διάγραμμα του **synthesized design model**.

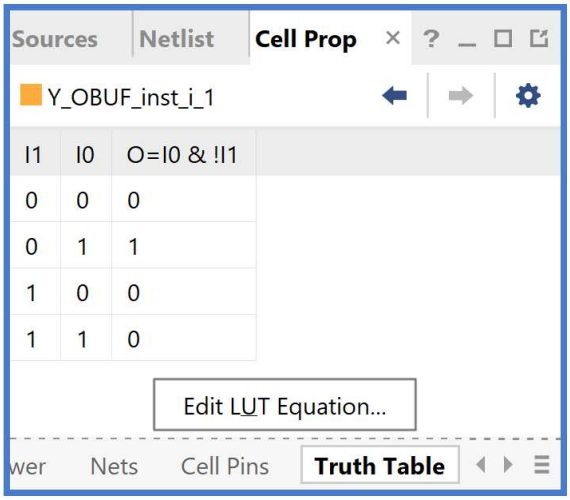


Παρατηρείστε ότι έχουν αυτόματα προστεθεί τα απαραίτητα IBUFs, OBUFs και BUFG primitives στο σχηματικό διάγραμμα.

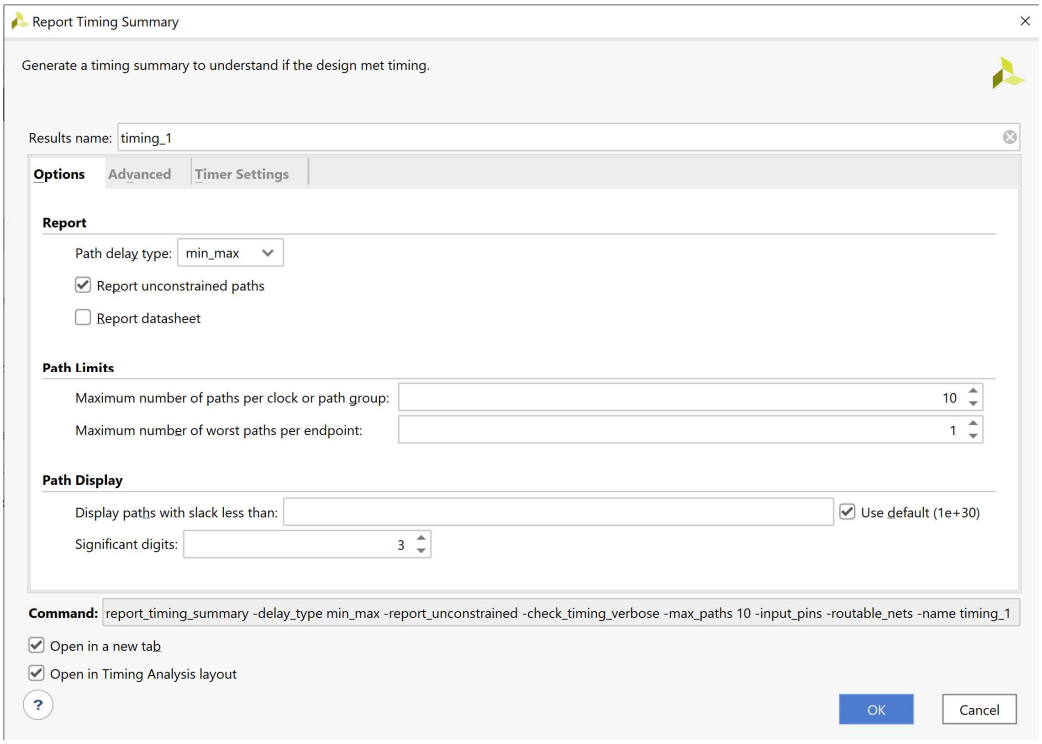
Με επιλογή των LUTs μπορείτε να επαληθεύσετε τη λογική που έχετε περιγράψει στον κώδικα VHDL. Για παράδειγμα, η λογική εξόδου έχει περιγραφθεί ως εξής:

```
if (current_state = S2 = "10") then Y <= '1'; else Y <= '0';
```

Επιλέγοντας το LUT2 με έξοδο Y (που αποκτά μπλε περίγραμμα), επαληθεύουμε τη λογική ρυθμίζοντας την επιλογή *Truth Table* στο παράθυρο *Cell Properties*. Μελετώντας το σχηματικό διάγραμμα παρατηρούμε ότι το I1 είναι η έξοδος του current_state_reg[0], ενώ το I0 είναι η έξοδος του current_state_reg[1].

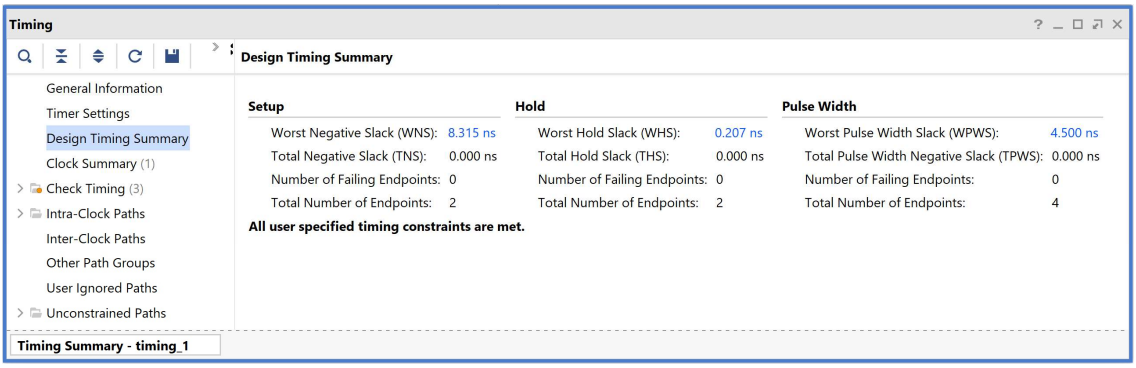


4-3.5. Στη συνέχεια, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Synthesized Design* επιλέξετε το **Report Timing Summary** για να δείτε την ανάλυση χρονισμού που κάνει το εργαλείο Vivado IDE στο **synthesized design model**. Αν και η ανάλυση χρονισμού σε αυτό το επίπεδο δεν είναι ακριβής, χρησιμοποιείται ευρέως στις πολύπλοκες σχεδιάσεις για εξοικονόμηση χρόνου κατά την ανάπτυξη του κώδικα VHDL.



Στην επιλογή *Path delay type* ορίζεται ο τύπος της ανάλυσης που θα εκτελεσθεί. Το *max delay analysis* αφορά στην εύρεση της κρίσιμης διαδρομής (με τη μεγαλύτερη καθυστέρηση διάδοσης) και συνεπώς στην εύρεση της μέγιστης συχνότητας λειτουργίας χωρίς την παραβίαση του **χρόνου σταθεροποίησης** (setup time). Το *min delay analysis* αφορά στην εύρεση της σύντομης διαδρομής (με τη μικρότερη καθυστέρηση διάδοσης) χωρίς την παραβίαση του **χρόνου διατήρησης** (hold time).

4-3.6. Πατήστε **OK** για να παραχθεί το Timing_1 report.



Στη στήλη **Setup** παρουσιάζονται τα αποτελέσματα του *max delay analysis*.

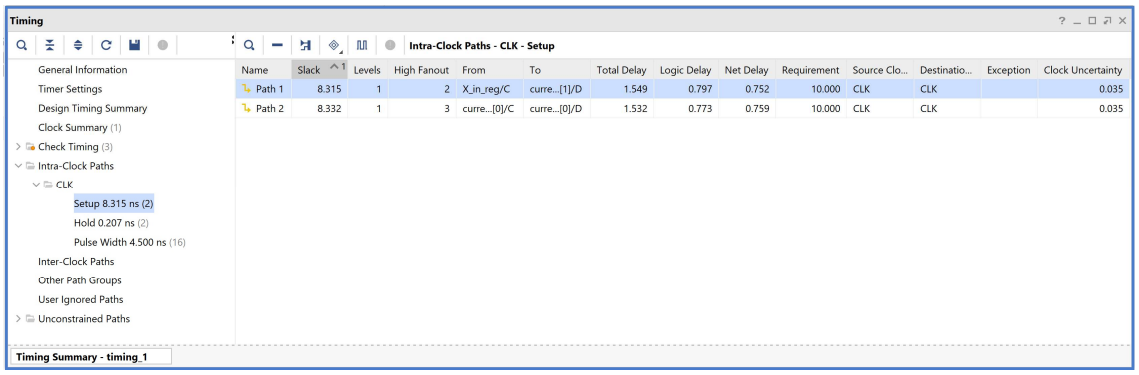
- Το *Worst Negative Slack* (**WNS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των κρίσιμων διαδρομών του max delay analysis. Μπορεί να είναι θετικό η αρνητικό. Ένα θετικό slack (8.315 ns) δηλώνει ότι η κρίσιμη διαδρομή ικανοποιεί την περίοδο του CLK. Ένα αρνητικό slack δηλώνει ότι παραβιάζεται ο χρόνος σταθεροποίησης (setup time).
- Το *Total Negative Slack* (**TNS**) είναι το άθροισμα όλων των αρνητικών WNS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου σταθεροποίησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά στην επιλεγμένη συχνότητα λειτουργίας.
- Το *Number of Failing Endpoints* αφορά στον συνολικό αριθμό των endpoints που έχουν παραβιάσει το χρόνο σταθεροποίησης (αρνητικό WNS).
- Το *Total Number of Endpoints* αφορά στον συνολικό αριθμό των endpoints που έχουν αναλυθεί.

Στη στήλη **Hold** παρουσιάζονται τα αποτελέσματα του min delay analysis.

- Το *Worst Hold Slack* (**WHS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των σύντομων διαδρομών του min delay analysis. Μπορεί να είναι θετικό η αρνητικό. Ένα αρνητικό slack δηλώνει ότι παραβιάζεται ο χρόνος διατήρησης (hold time).
- Το *Total Hold Slack* (**THS**) είναι το άθροισμα όλων των αρνητικών WHS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου διατήρησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά.
- Το *Number of Failing Endpoints* αφορά στον συνολικό αριθμό των endpoints που έχουν παραβιάσει το χρόνο διατήρησης (αρνητικό WHS).
- Το *Total Number of Endpoints* αφορά στον συνολικό αριθμό των endpoints που έχουν αναλυθεί.

Στη στήλη **Pulse Width** παρουσιάζονται τα περιθώρια του σήματος CLK, όταν είναι HIGH ή LOW.

4-3.7. Επιλέξτε το **WNS link** και δείτε τις 2 διαθέσιμες κρίσιμες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο σταθεροποίησης. Η κρίσιμη διαδρομή έχει καθυστέρηση διάδοσης 1.549 ns, εκ των οποίων τα 0.797 ns αφορούν στη λογική (logic), ενώ τα 0.752 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.035 ns. Τα επίπεδα λογικής (logic level) είναι 1.

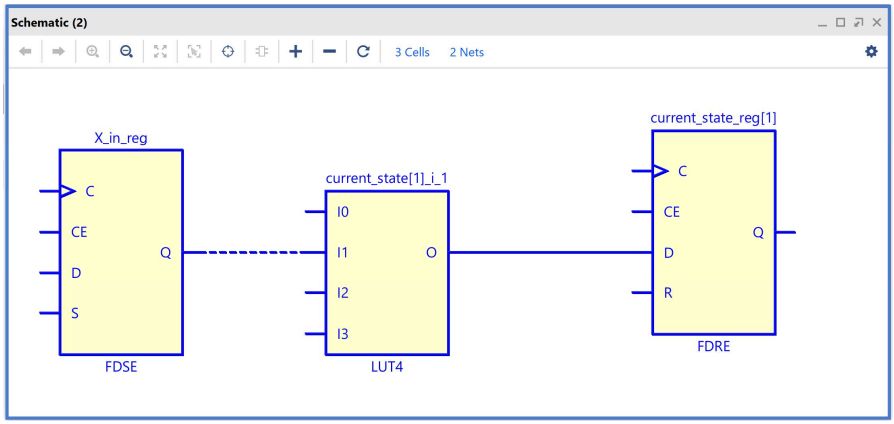


4-3.8. Επιλέξτε με διπλό κλικ το **Path 1**, ώστε να εμφανιστεί το παράθυρο *Path 1 – timing_1*. Η κρίσιμη διαδρομή περνάει από 1 LUT4. Υπολογίστε το WNS slack:

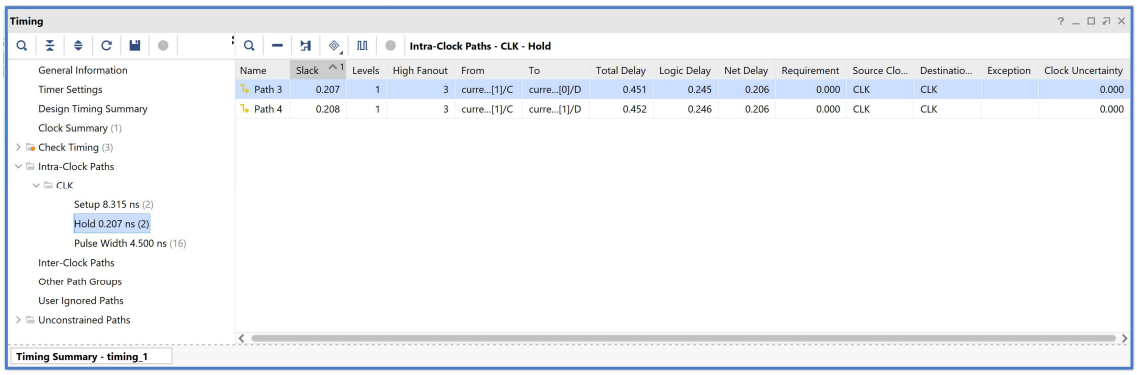
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή εισόδων) είναι 2.975 ns,
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή εισόδων και μέσω του LUT4 μέχρι την είσοδο D του καταχωρητή κατάστασης 1) είναι 1.549 ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι 4.524 ns,
- το **Required Time**, ως η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή 10.000 ns μέχρι την είσοδο CLK του καταχωρητή κατάστασης 1) συν τον χρόνο σταθεροποίησης είναι 12.839 ns.

WNS slack = Required Time – Arrival Time = 12.839 – 4.524 = 8.315 ns

4-3.9. Επιλέξτε με δεξί κλικ στο **Path 1**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της κρίσιμης διαδρομής της οντότητας **PATTERN_FSM**.



4-3.10. Επιλέξτε το **WHS link** και δείτε τις 2 σύντομες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο διατήρησης. Η σύντομη διαδρομή έχει καθυστέρηση μόλυνσης 0.451 ns, εκ των οποίων τα 0.245 ns αφορούν στη λογική (logic), ενώ τα 0.206 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.000 ns. Τα επίπεδα λογικής (logic level) είναι 1.

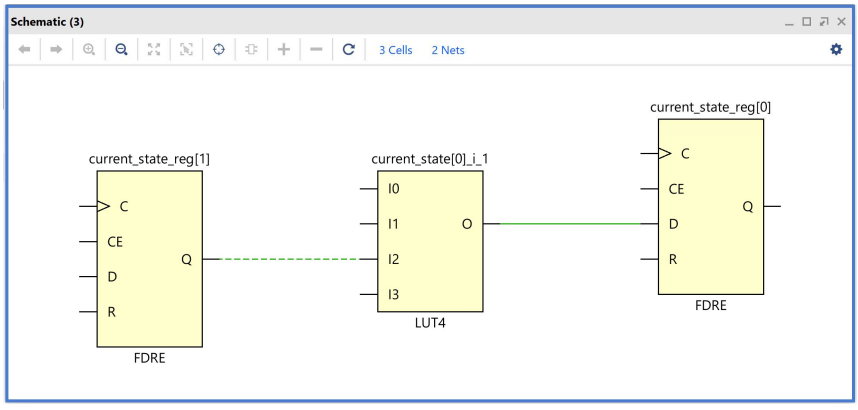


4-3.11. Επιλέξτε με διπλό κλικ το **Path 3**, ώστε να εμφανιστεί το παράθυρο *Path 3 – timing_1*. Η σύντομη διαδρομή περνάει από 1 μονάδα LUT4. Υπολογίστε το WHS slack:

- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή κατάστασης 1) είναι 0.735 ns,
- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή κατάστασης 1 και μέσω του LUT4 μέχρι την είσοδο D του καταχωρητή κατάστασης 0) είναι 0.451 ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι 1.186 ns,
- το **Required Time**, ως η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK στον καταχωρητή κατάστασης 0) συν τον χρόνο διατήρησης, είναι 0.979 ns.

WHS slack = Arrival Time – Required Time = 1.186 – 0.979 = 0.207 ns

4-3.12. Επιλέξτε με δεξί κλικ στο **Path 3**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της σύντομης διαδρομής της οντότητας **PATTERN_FSM**.



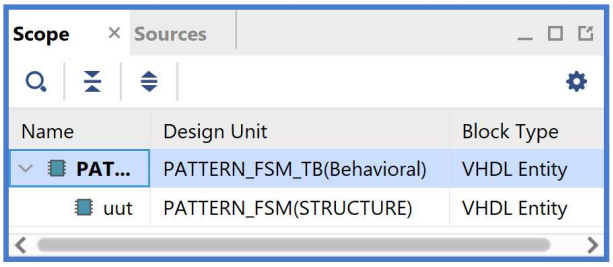
4-4. Εκτέλεση προσομοίωσης μετά τη σύνθεση (λογική και χρονική) στο VIVADO IDE για μηχανή πεπερασμένων καταστάσεων (FSM)

Η προσομοίωση μετά τη σύνθεση (λογική και χρονική) εκτελείται στην οντότητα **PATTERN_FSM_TB** του προγράμματος δοκιμών (testbench) που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των simulation resources. Κατά την προσομοίωση, η οντότητα **PATTERN_FSM_TB** καλεί το *UUT* της (που είναι το **synthesized design model** της οντότητας **PATTERN_FSM**).

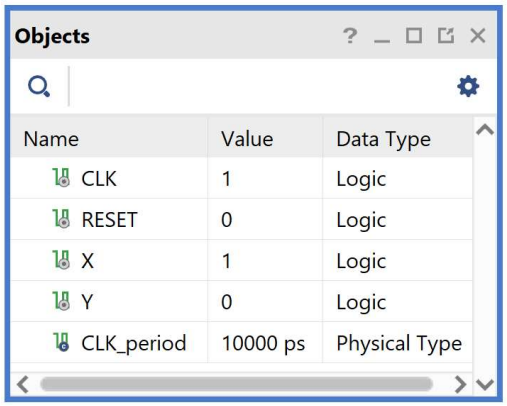
4-4.1. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Simulation**, ώστε να εμφανιστούν όλες οι πιθανές προσομοιώσεις που υποστηρίζει το Vivado IDE. Επιλέξτε **Run Post-Synthesis Functional Simulation**.

4-4.2. Το πρόγραμμα δοκιμής (testbench) και όλες οι οντότητες του *UUT* θα γίνουν compiled και θα τρέξει το Vivado simulator (εφόσον βέβαια δεν υπάρχουν σφάλματα). Θα εμφανιστεί το παράθυρο *SIMULATION* που απαρτίζεται από 4 παράθυρα:

Το παράθυρο *Scope*, όπου παρουσιάζεται η οντότητα **PATTERN_FSM_TB** και το **synthesized design model** της οντότητας **PATTERN_ FSM** (*UUT*) που προκύπτει μετά τη σύνθεση, η οποία πλέον είναι *structural*.

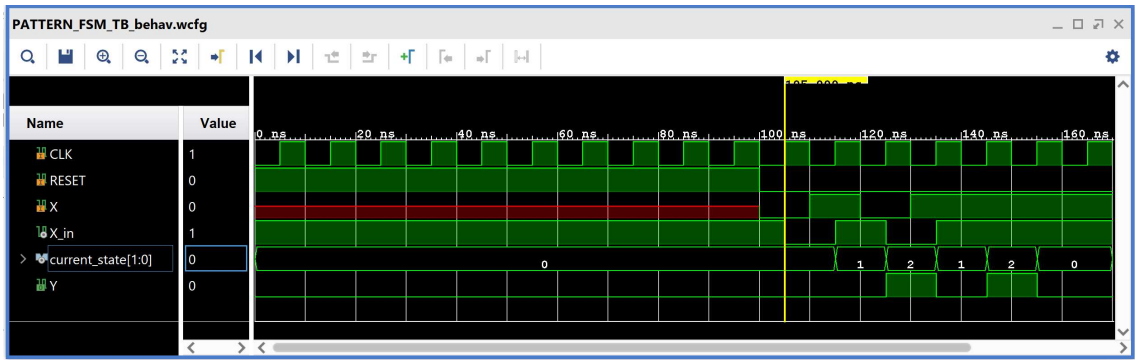


Το παράθυρο *Objects*, όπου εμφανίζονται τα σήματα top-level, δηλαδή οι είσοδοι και οι έξοδοι της οντότητας **PATTERN_ FSM**, που είναι η κορυφαία οντότητα της ιεραρχίας του *UUT*, καθώς και η περίοδος του CLK (*CLK_period*). Οι αρτηρίες (array) αναλύονται στα σήματα που τις απαρτίζουν. Οι τιμές αντιστοιχούν στις τιμές που έχει σταματήσει η προσομοίωση της οντότητας **PATTERN_ FSM_TB** (st*op (2)*).

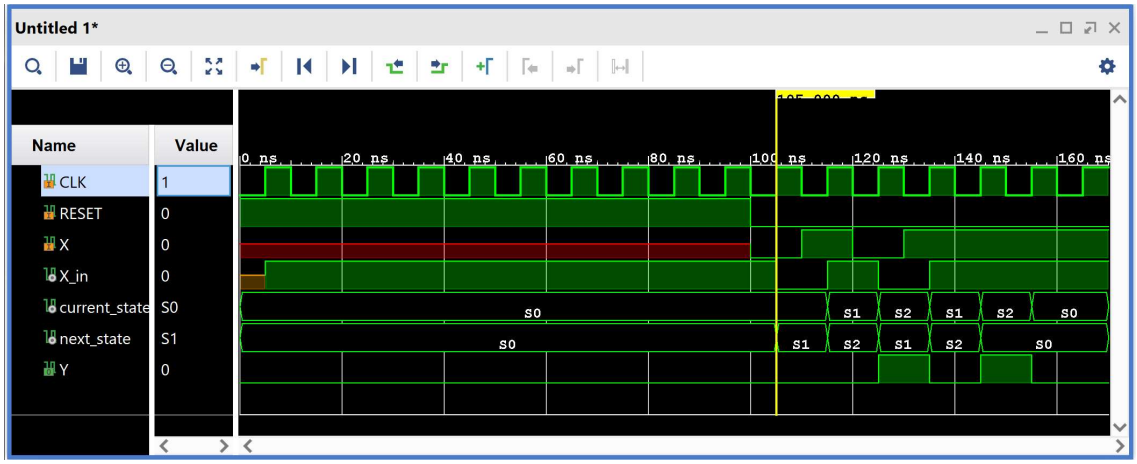


Το παράθυρο *Tcl Console* με όλες τις διαδικασίες που εκτελούνται στο πλαίσιο της προσομοίωσης. Το Tcl Console καθαρίζει με την επιλογή *Clear*.

Επιλέξτε το παράθυρο *PATTERN_ FSM_TB_behav.wcfg*, που δημιουργήσατε για την προσομοίωση συμπεριφοράς. Βλέπετε ολόκληρο το διάγραμμα χρονισμού της λογικής προσομοίωσης μετά τη σύνθεση με κατάλληλο **zoom out** ή επιλέγοντας το **zoom fit** . Για να επαναφέρετε το floating παράθυρο πίσω, απλά επιλέξτε το κουμπί Dock Window.

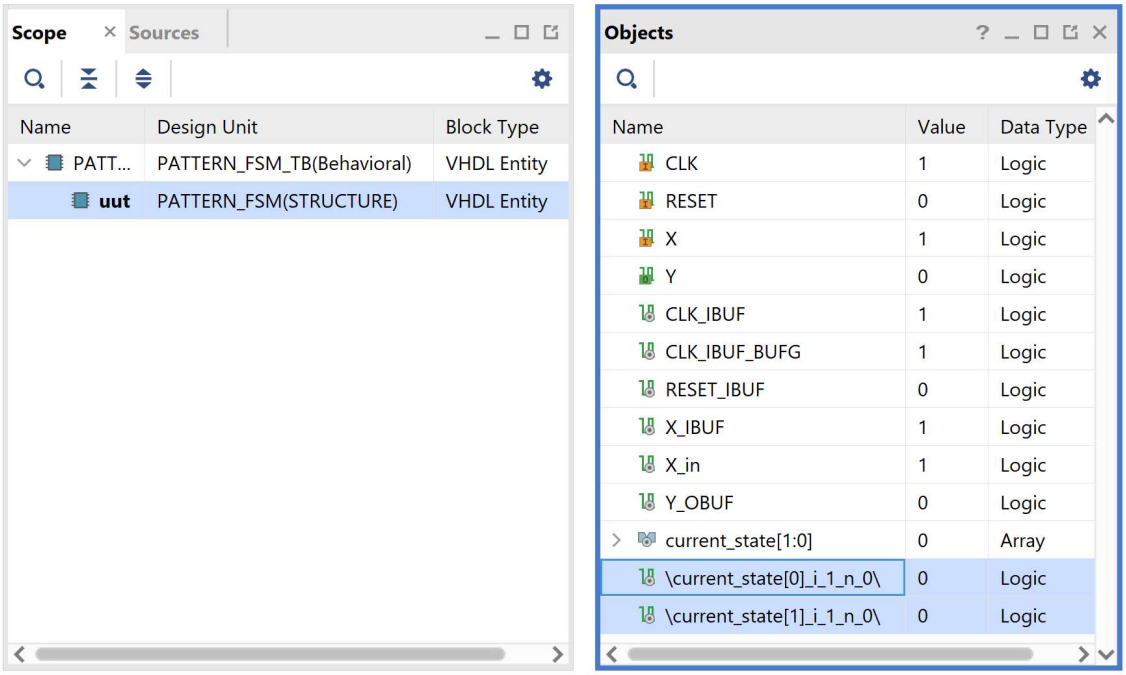


4-4.3. Συγκρίνετε τα διαγράμματα χρονισμού της λογικής προσομοίωσης μετά τη σύνθεση και της προσομοίωσης συμπεριφοράς (υπάρχει ταύτιση στο επίπεδο εισόδων/ εξόδων και εσωτερικών σημάτων X_in και current_state, όπου S0 = 0, S1 = 1, S2 = 2).

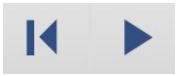


4-4.4. Προσθέστε περισσότερα **εσωτερικά σήματα** στο διάγραμμα χρονισμού της προσομοίωσης μετά τη σύνθεση. Μετά από μελέτη του σχηματικού διαγράμματος του **synthesized design model** προκύπτει ότι οι είσοδοι **D** των FDRE **current_state _reg[0]** (**current_state[0]_i_1_n_0**) και **current_state _reg[1]** (**current_state[1] _i_1_n_0**) αντιστοιχούν στο **next_state** της οντότητας **PATTERN_ FSM** (του **elaborated design model** πριν τη σύνθεση) και για αυτό το λόγο τις προσθέτουμε στο διάγραμμα χρονισμού της προσομοίωσης μετά τη σύνθεση.

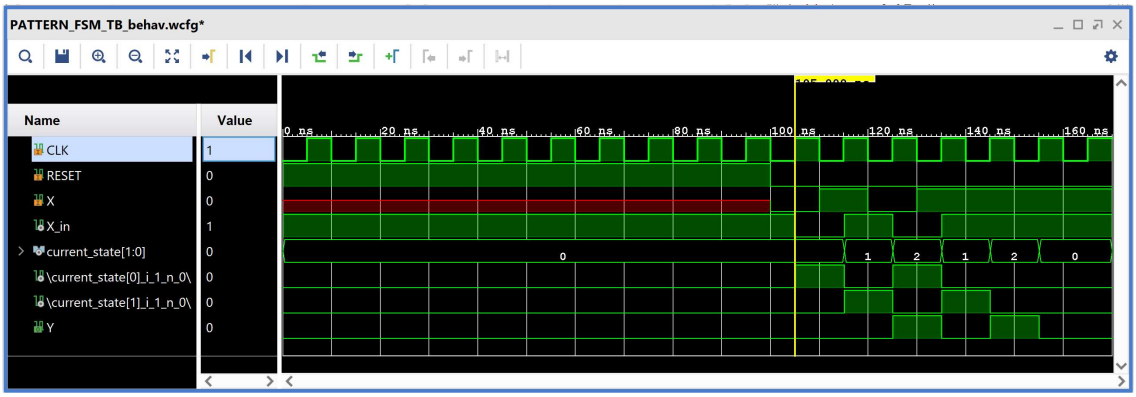
Στο παράθυρο *Scope* επιλέξτε την οντότητα **PATTERN_ FSM** (*UUT*). Παρατηρείστε ότι στο παράθυρο Objects εμφανίζονται όλα τα σήματα του *UUT*. Επιλέξτε τα εσωτερικά σήματα **current_state[0]_i_1_n_0** (αντιστοιχεί στο next_state[0]) και **current_ state[1]_i_1_n_0** (αντιστοιχεί στο next_state[1]) και κάντε drag and drop στο παράθυρο με τα διαγράμματα χρονισμού, τοποθετώντας τα κάτω από το **current_state**.



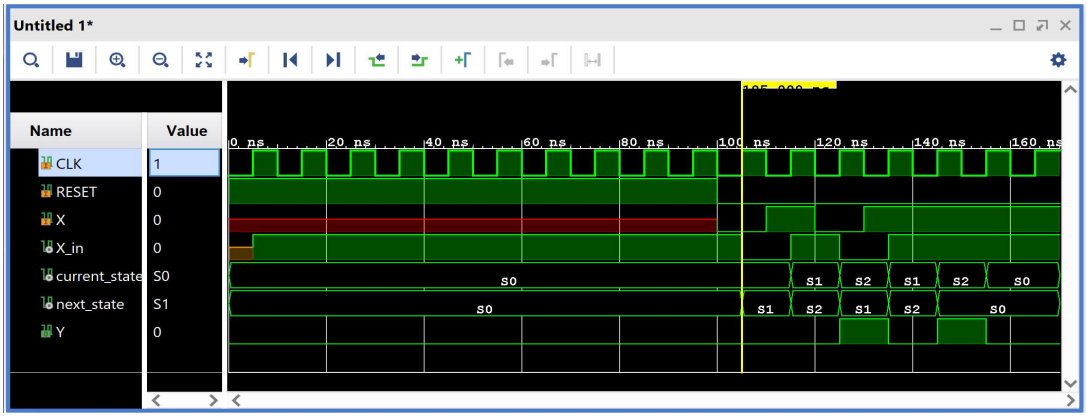
Επαναλάβετε τη διαδικασία της προσομοίωσης από την αρχή με επιλογή του κουμπιού **Restart** και στη συνέχεια του κουμπιού **Run All**. Και τα δύο κουμπιά βρίσκονται στην οριζόντια μπάρα στο πάνω μέρος.



Διάγραμμα χρονισμού της λογικής προσομοίωσης μετά τη σύνθεση:



Διάγραμμα χρονισμού της προσομοίωσης συμπεριφοράς:



4-4.5. Συγκρίνετε τη λογική προσομοίωση μετά τη σύνθεση με την προσομοίωση συμπεριφοράς (υπάρχει ταύτιση στο επίπεδο εισόδων/εξόδων και επιλεγμένων εσωτερικών σημάτων).

4-4.6. Επιλέξτε αρχικά **File** στη συνέχεια **Simulation Waveform** και τέλος **Save Configuration as ..** για να αποθηκεύσετε το configuration του διαγράμματος χρονισμού στο οποίο έχετε καταλήξει με νέο όνομα (π.χ. **PATTERN_FSM_TB _func_synth.wcfg**). Στο παράθυρο *Waveform Configuration File* που εμφανίζεται πατήστε **Yes**.

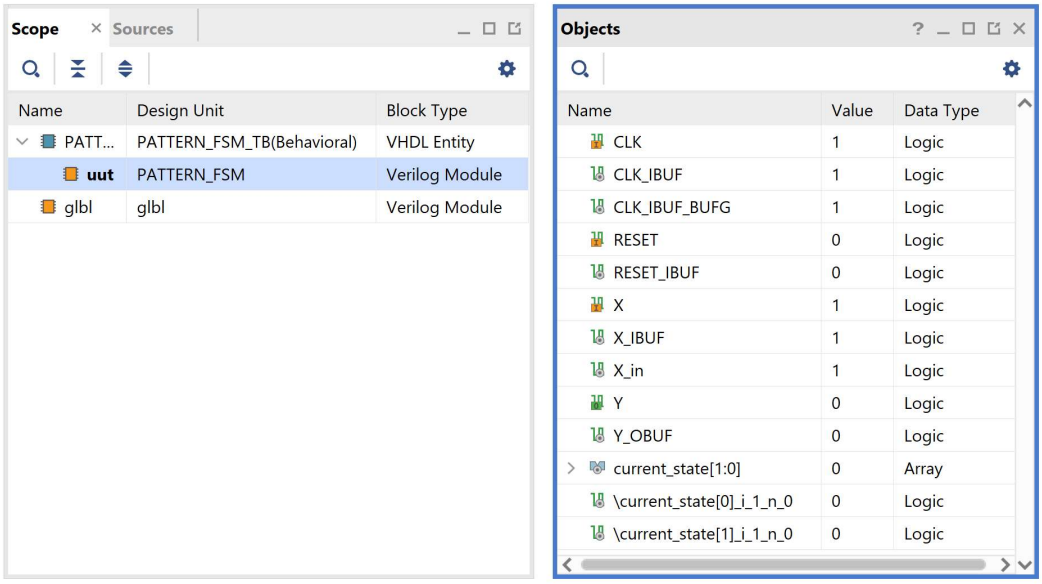
4-4.7. Κλείστε τον simulator επιλέγοντας το κουμπί × πάνω δεξιά στο παράθυρο του *SIMULATION*. Στο παράθυρο *Confirm Close* που εμφανίζεται πατήστε **OK**.

4-4.8. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Simulation**, ώστε να εμφανιστούν όλες οι πιθανές προσομοιώσεις που υποστηρίζει το Vivado IDE. Επιλέξτε **Run Post-Synthesis Timing Simulation**.

4-4.9. Το πρόγραμμα δοκιμής (testbench) και όλες οι οντότητες του *UUT* θα γίνουν compiled και θα τρέξει το Vivado simulator (εφόσον βέβαια δεν υπάρχουν σφάλματα). Θα εμφανιστεί το παράθυρο *SIMULATION* που απαρτίζεται από 4 παράθυρα:

Το παράθυρο *Scope*, όπου παρουσιάζεται η οντότητα **PATTERN_ FSM_TB** και το **synthesized design model** της οντότητας **PATTERN_ FSM** (*UUT*) που προκύπτει μετά τη σύνθεση για χρονική προσομοίωση. Όλες οι οντότητες που απαρτίζουν πλέον το *UUT* είναι *Verilog Module!*

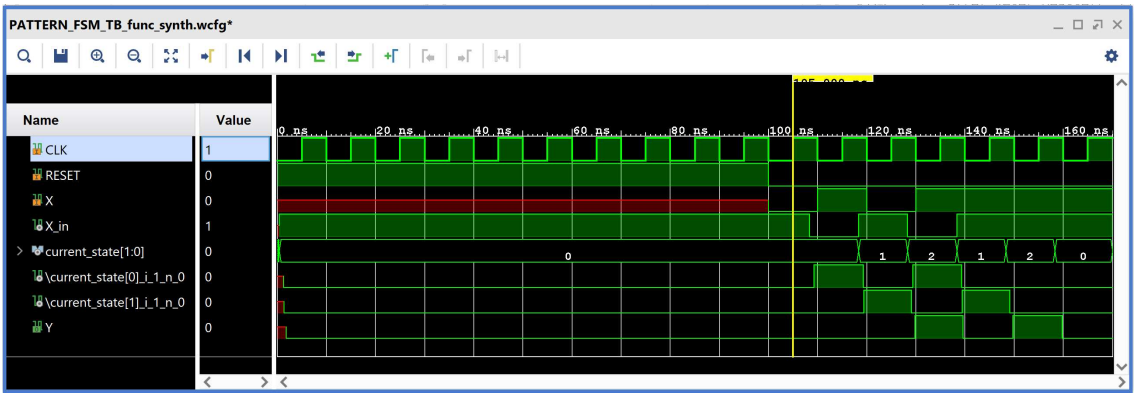
Το παράθυρο *Objects*, όπου εμφανίζονται τα σήματα της οντότητας **PATTERN_ FSM** (*UUT*). Οι αρτηρίες (array) αναλύονται στα σήματα που τις απαρτίζουν. Οι τιμές αντιστοιχούν στις τιμές που έχει σταματήσει η προσομοίωση της οντότητας **PATTERN_ FSM_TB** (st*op (2)*).



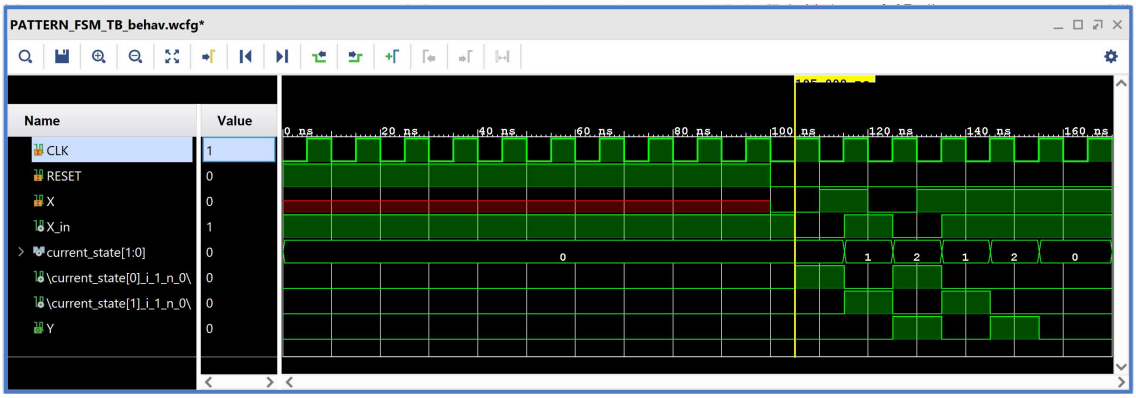
Το παράθυρο *Tcl Console* με όλες τις διαδικασίες που εκτελούνται στο πλαίσιο της προσομοίωσης. Το Tcl Console καθαρίζει με την επιλογή *Clear*.

Επιλέξτε το παράθυρο *PATTERN_ FSM_TB_func_synth.wcfg*. Βλέπετε ολόκληρο το διάγραμμα χρονισμού της χρονικής προσομοίωσης μετά τη σύνθεση με κατάλληλο **zoom out** ή επιλέγοντας το **zoom fit**.

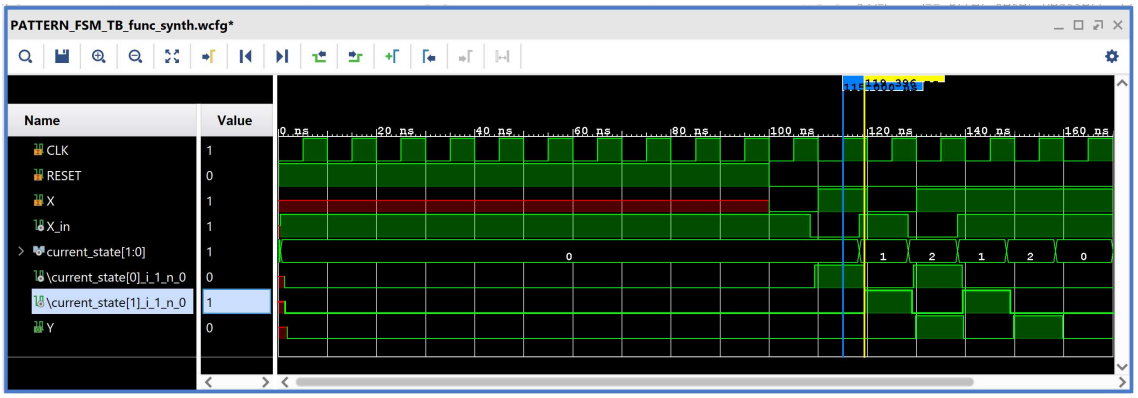
Σημείωση: Εάν δεν θέλετε να βλέπετε άλλα ενεργοποιημένα παράθυρα διαγραμμάτων χρονισμού κατά τη διάρκεια της προσομοίωσης μπορείτε να τα απενεργοποιήσετε. Στο παράθυρο *Sources* επιλέξτε με δεξί κλικ το παράθυρο διαγραμμάτων χρονισμού που θέλετε να απενεργοποιήσετε και πατήστε **Disable File**. Τα απενεργοποιημένα παράθυρα διαγραμμάτων χρονισμού μεταφέρονται στα *Disabled Sources*. Για να ενεργοποιήσετε πάλι κάποιο παράθυρο διαγραμμάτων χρονισμού επιλέξτε το με δεξί κλικ και πατήστε **Enable File**.



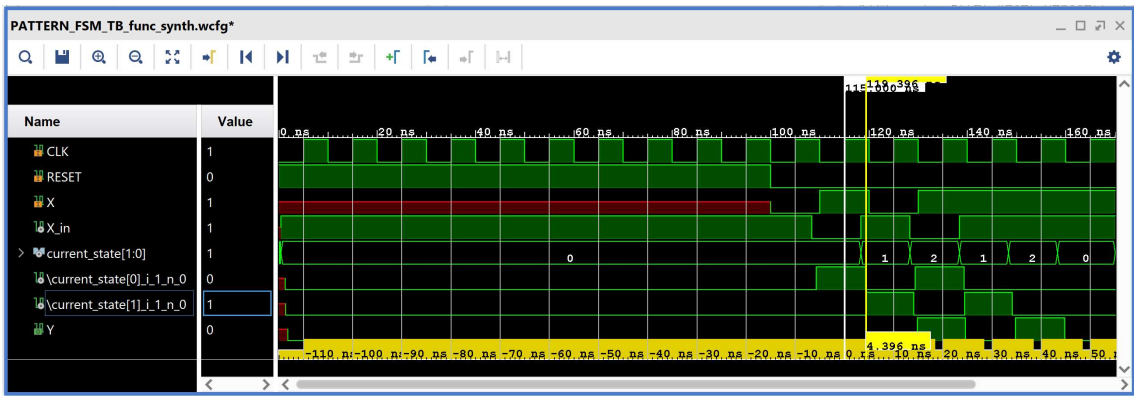
4-4.10. Συγκρίνετε τα διαγράμματα χρονισμού της χρονικής και της λογικής προσομοίωσης μετά τη σύνθεση. Είναι εμφανείς οι καθυστερήσεις διάδοσης στο πρώτο διάγραμμα χρονισμού.



4-4.11. Υπολογίστε το **Arrival Time** ενός σήματος στο διάγραμμα χρονισμού της χρονικής προσομοίωσης μετά τη σύνθεση με τη χρήση των marker. Για παράδειγμα, βάλτε τον μπλε marker στην ανερχόμενη ακμή του **CLK** στα 115.000 ns και τον κίτρινο marker στην ανερχόμενη ακμή του εσωτερικού σήματος **current_state[1]_i_1_n_0** (που αντιστοιχεί στο next_state[1]) στα 119.396 ns. Απέχουν 4.396 ns. Συγκρίνετε με το **Arrival Time** που είχατε βρει κατά τη χρονική ανάλυση για την κρίσιμη διαδρομή, που ήταν 4.524 ns.



Με κλικ πάνω στον μπλε marker (γίνεται λευκός), ορίζουμε τη θέση του στα 0.000 ns και είναι πλέον εμφανές ότι ο κίτρινος marker απέχει 4.396 ns.



4-4.12. Κλείστε τον simulator επιλέγοντας το κουμπί ✕ πάνω δεξιά στο παράθυρο του *SIMULATION*. Στο παράθυρο *Confirm Close* που εμφανίζεται πατήστε **OK**. Εάν επιθυμείτε να μη σώσετε ένα επιπλέον waveform configuration, στο παράθυρο *Save Waveform Configuration* επιλέξτε **Discard**.

Η διαδικασία που ήδη περιγράψαμε στα Βήματα 4-3 και 4-4 της «**Σύνθεσης του κώδικα VHDL και προσομοίωση (λογική, χρονική)**» εφαρμόζεται παρομοίως σε κάθε πιθανή μηχανή πεπερασμένων καταστάσεων (FSM).

Βήμα 5: Υλοποίηση στη τεχνολογία FPGA και προσομοίωση (λογική, χρονική)

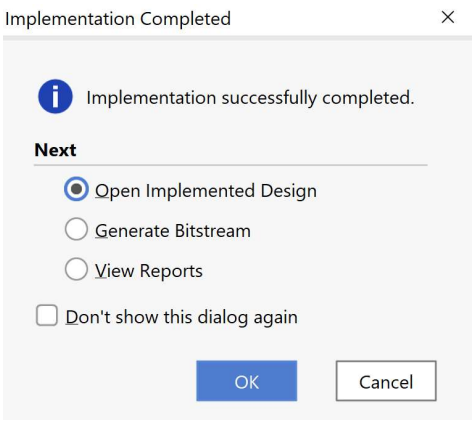
5-1. Εκτέλεση της διαδικασίας της υλοποίησης και ανάλυση των αποτελεσμάτων μετά την υλοποίηση στο VIVADO IDE για συνδυαστική λογική ανάμεσα σε καταχωρητές

Σε μία διαδικασία bottom-up σχεδίασης και υλοποίησης ενός ψηφιακού συστήματος, η υλοποίηση σε συγκεκριμένη τεχνολογία FPGA εφαρμόζεται κυρίως στα ανώτερα ιεραρχικά επίπεδα, όταν η σχεδίαση έχει πλέον ωριμάσει στο επίπεδο της αποσφαλμάτωσης και της βελτιστοποίησης. Η διαδικασία της υλοποίησης εφαρμόζεται στο **synthesized design model** της οντότητας που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design resources και μετά από την ολοκλήρωση της υλοποίησης παράγεται το αντίστοιχο **implemented design model**.

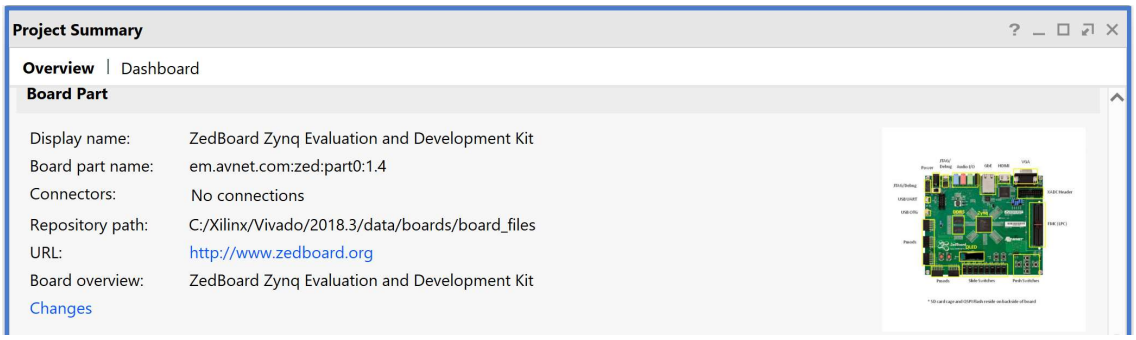
Η υλοποίηση στην τεχνολογία **Zynq-7000 (device xc7z020clg484-1)** εκτελείται στο **synthesized design model** της οντότητας **ADDER_REG_8** που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design resources, καθώς και σε όλα τα υπάρχοντα αρχεία της ιεραρχίας. Με τη σύνθεση το εργαλείο Vivado IDE παράγει το **implemented design model** της οντότητας **ADDER_REG_8**.

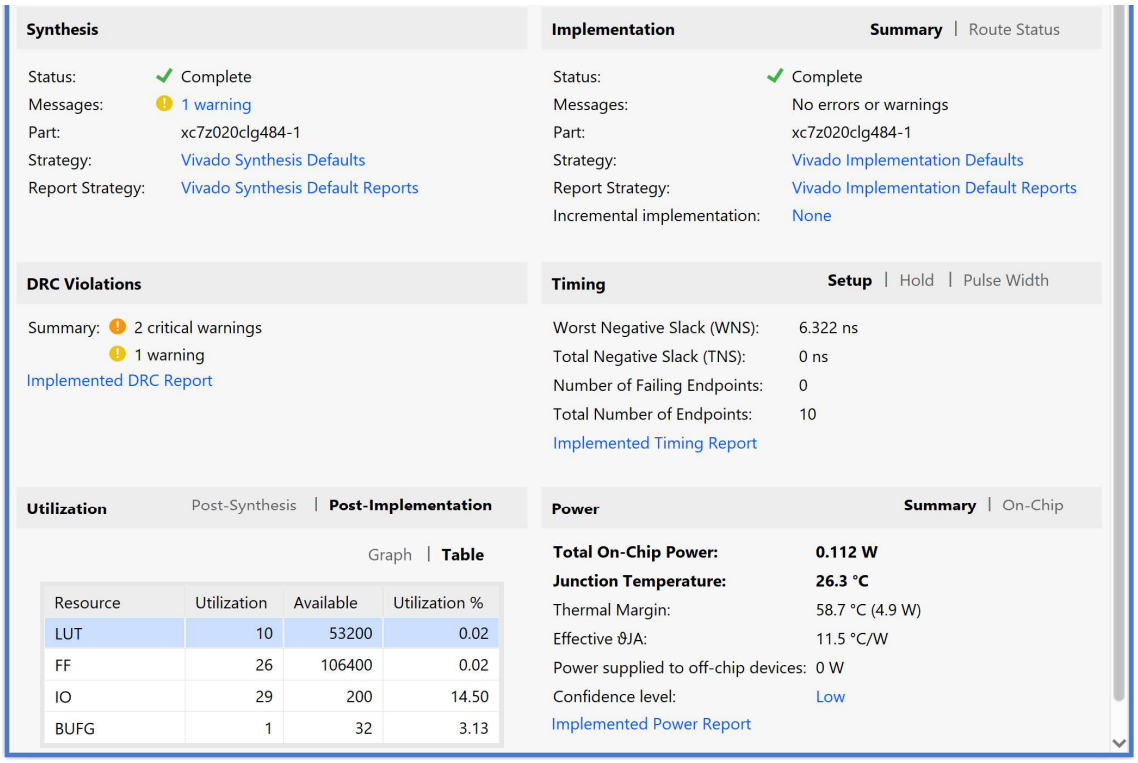
5-1.1. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Implementation**. Πατήστε **OK** στα παράθυρα προειδοποίησης που εμφανίζονται.

5-1.2. Στο παράθυρο *Implementation Completed* υπάρχουν τρεις επιλογές. Επιλέξτε το **Open Implemented Design** και πατήστε **OK** για να μελετήσετε το αποτέλεσμα της υλοποίησης (implementation).

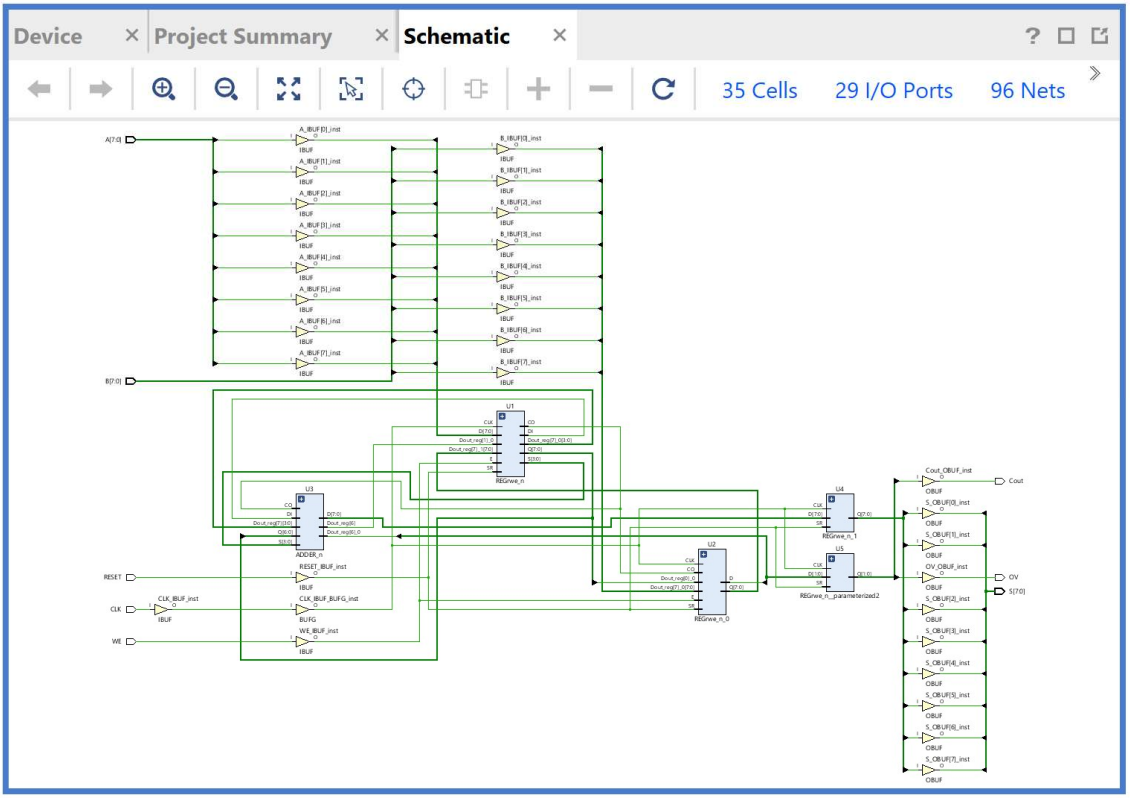


5-1.3. Αρχικά, επιλέξτε το παράθυρο *Project Summary* και μελετήστε τα διάφορα υπο-παράθυρα.





5-1.4. Στη συνέχεια, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Implemented Design* επιλέξετε το **Schematic** για να δείτε το σχηματικό διάγραμμα του **implemented design model**.



Παρατηρείστε ότι έχουν αυτόματα προστεθεί τα απαραίτητα IBUFs, OBUFs και BUFG primitives στο σχηματικό διάγραμμα, καθώς και ότι οι είσοδοι και οι έξοδοι από το FPGA είναι buffered. Επίσης, παρατηρείστε ότι έχει διατηρηθεί εν μέρει η ιεραρχία που έχει ορισθεί στην οντότητα **ADDER_REG_8**. Πατήστε το (+) σε όλες τις υπομονάδες U1-U5 και μελετήστε τις μία προς μία. Είναι όμοιες με τις υπομονάδες που παράχθηκαν μετά τη σύνθεση.

Υπομονάδα U1 (REGrwe_n): Συμπεριλαμβάνει τον καταχωρητή εισόδου A των 8 bit, 8 πύλες XOR (LUT2), έναν αντιστροφέα (LUT1) και μία μονάδα CARRY4.

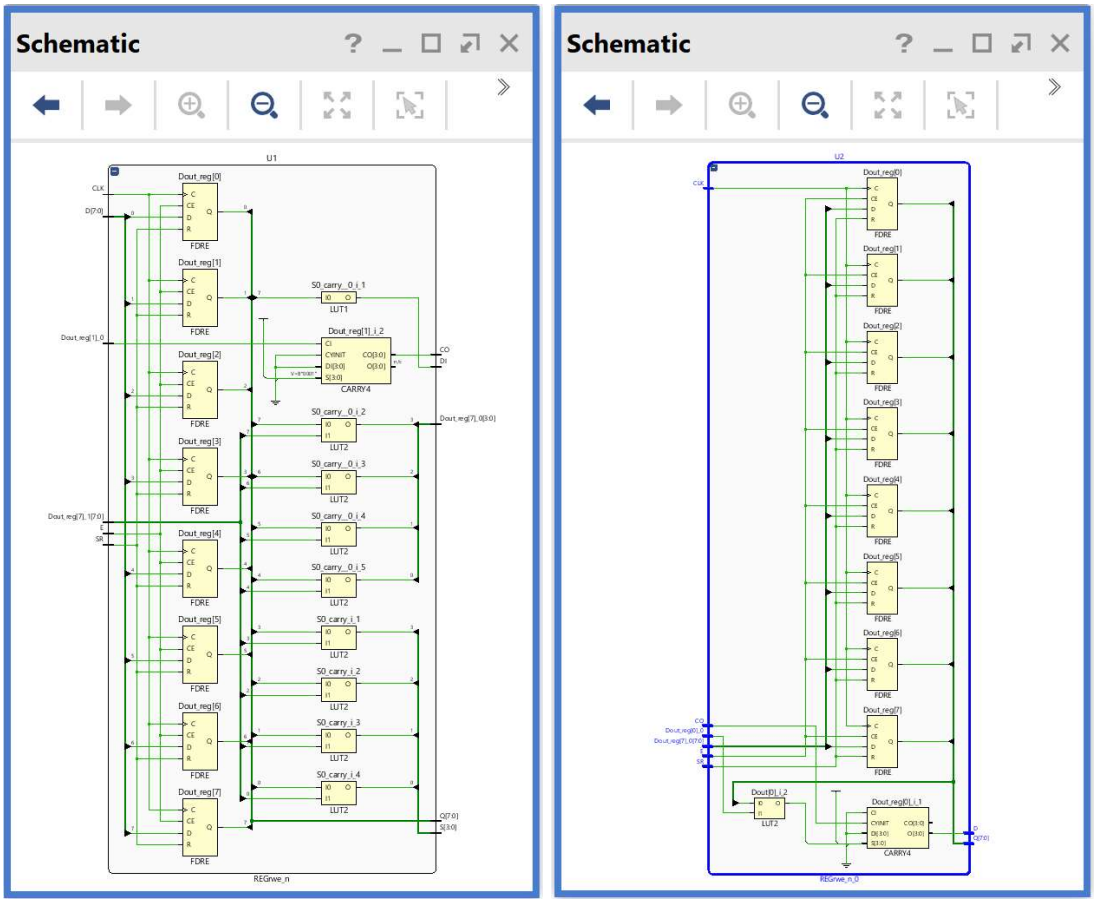
Υπομονάδα U2 (REGrwe_n_0): Συμπεριλαμβάνει τον καταχωρητή εισόδου B των 8 bit, 1 πύλη XOR (LUT2) και μία μονάδα CARRY4.

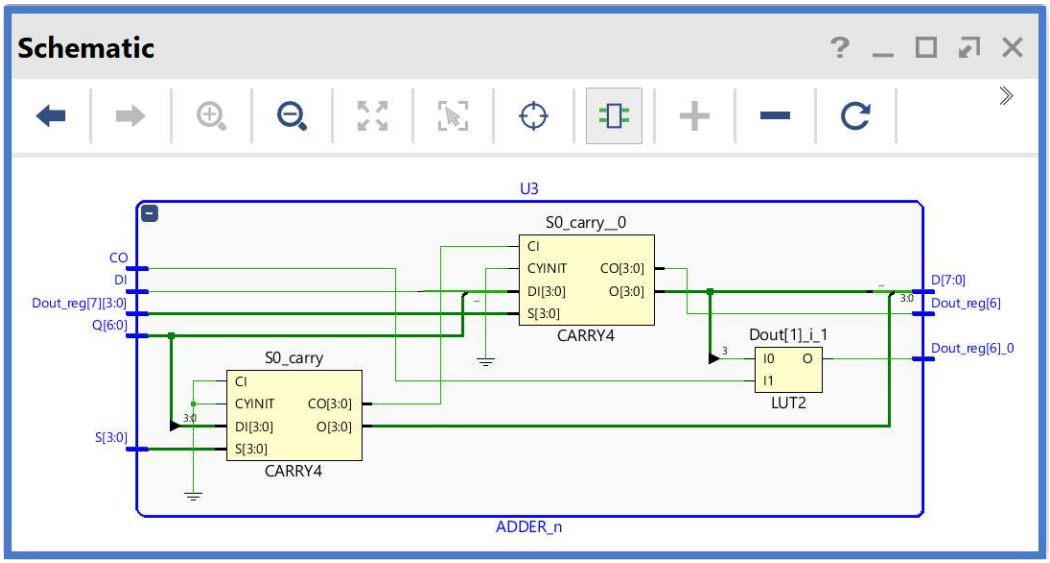
Υπομονάδα U3 (ADDER_n): Συμπεριλαμβάνει 1 πύλη XOR (LUT2) και 2 μονάδες CARRY4.

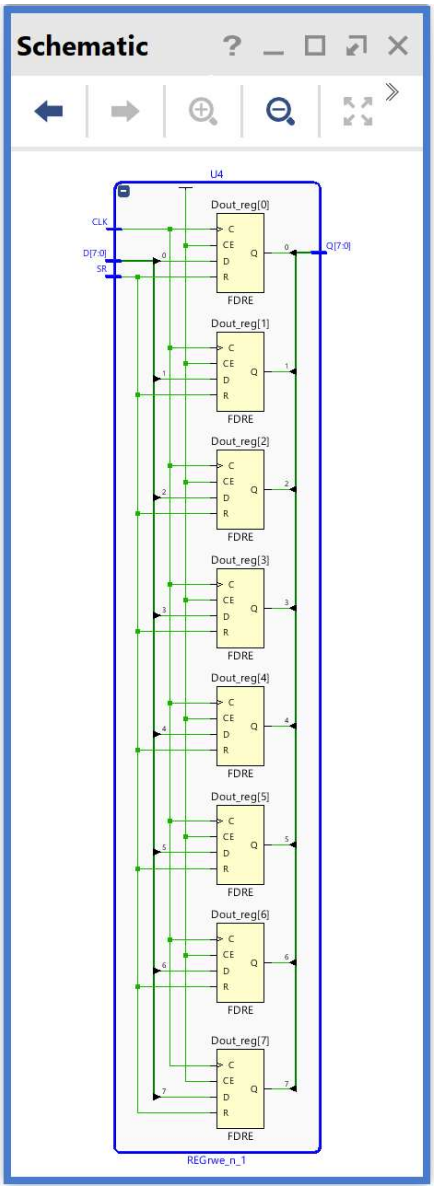
Υπομονάδα U4 (REGrwe_n_1): Συμπεριλαμβάνει τον καταχωρητή εξόδου S των 8 bit.

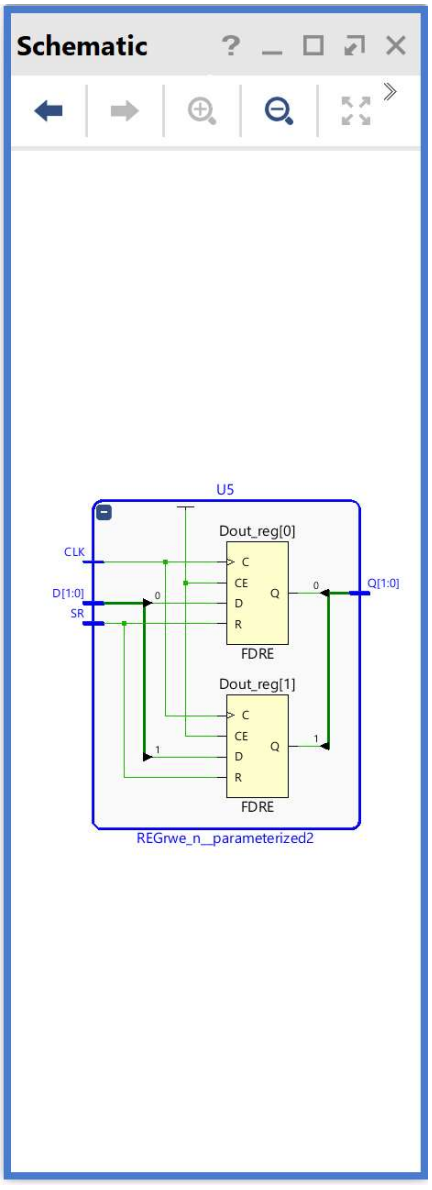
Υπομονάδα U5 (REGrwe_n_parameterized2): Συμπεριλαμβάνει τον καταχωρητή εξόδου των 2 bit για τις σημαίες Cout και OV.

Προσοχή! Οι υπομονάδες **U1**, **U2** και **U3** είναι διαφορετικές από πλευράς λογικής στο **synthesized (implemented) design model**, που προκύπτει μετά τη σύνθεση (υλοποίηση), σε σχέση με το **elaborated design model**. Αντίθετα, οι υπομονάδες **U4** και **U5** παραμένουν ίδιες από πλευράς λογικής.



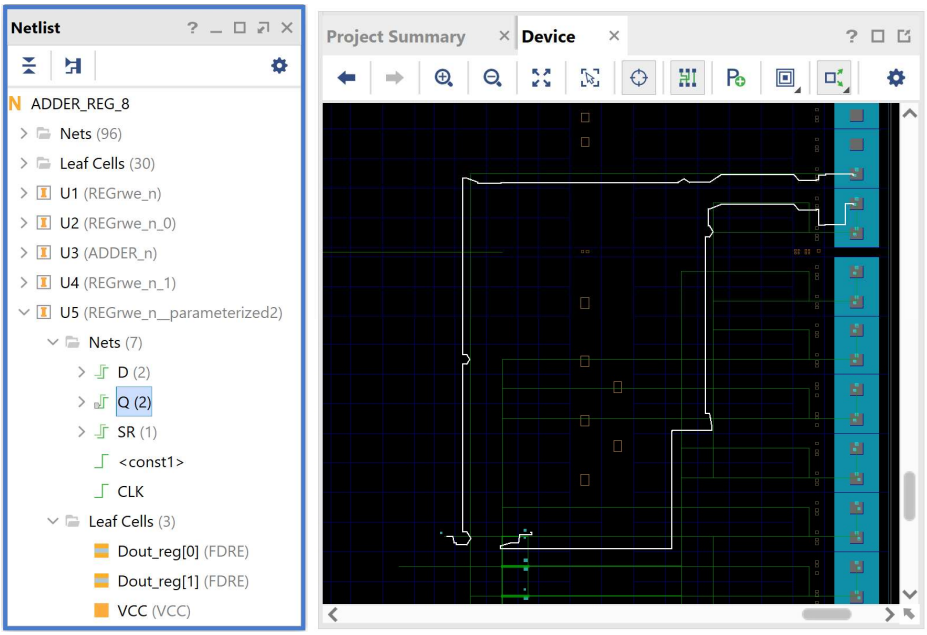

Schematic
U3
S0_carry__0
CI
CYINIT
DI[3:0]
S[3:0]
CO[3:0]
O[3:0]
CARRY4
CO
DI
Dout_reg[7][3:0]
Q[6:0]
S0_carry
S[3:0]
D[7:0]
Dout_reg[6]
Dout[1]_i_1
I0
I1
O
LUT2
Dout_reg[6]_0
ADDER_n


Schematic
U4
Dout_reg[0]
Dout_reg[1]
Dout_reg[2]
Dout_reg[3]
Dout_reg[4]
Dout_reg[5]
Dout_reg[6]
Dout_reg[7]
CLK
D[7:0]
SR
Q[7:0]
C
CE
D
R
Q
FDRE
REGrwe_n_1

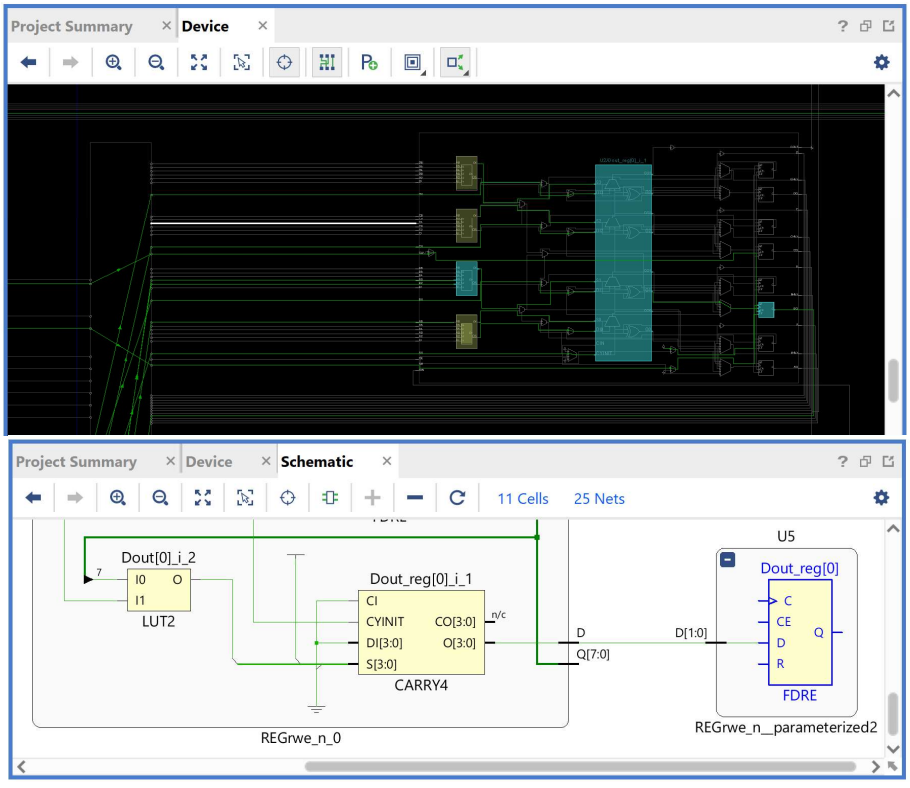

Schematic
U5
Dout_reg[0]
Dout_reg[1]
CLK
D[1:0]
SR
Q[1:0]
C
CE
D
R
Q
FDRE
REGrwe_n__parameterized2

5-1.5. Στη συνέχεια μελετήστε το παράθυρο *Device* έχοντας πατήσει τα κουμπιά *auto-fit selection*, *routing resources* και *show cell connections*.

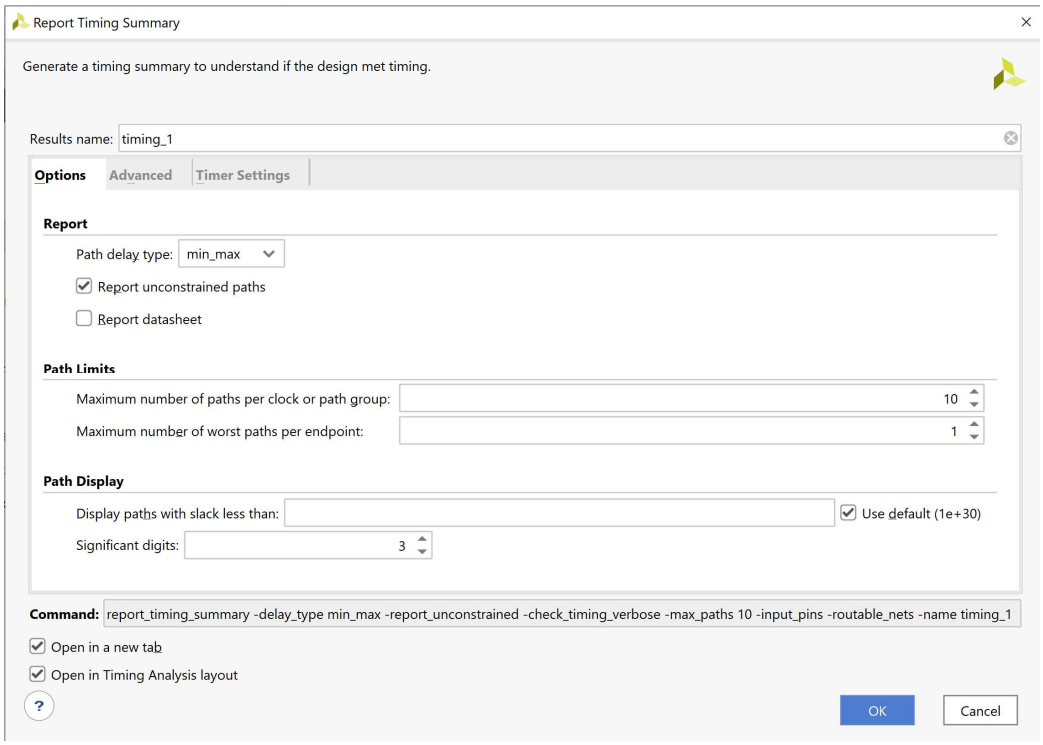
Στη μελέτη σας επιλέξτε κάποιο *net*, όπως για παράδειγμα το **Q[1:0]** (OV, Cout) της υπομονάδας U5 με κατάλληλη επιλογή στο παράθυρο *Netlist*.



5-1.6. Επικεντρώστε στη δημιουργία του Q[0] (Cout) με παράλληλη μελέτη των παραθύρων *Device* και *Schematic*, εναλλάξ. Απαιτείται εντοπισμένη μεγέθυνση. Φαίνονται: το LUT2 (U2), το CARRY4 (U2) και το FDRE (U5) στο SLICE_X113Y15!

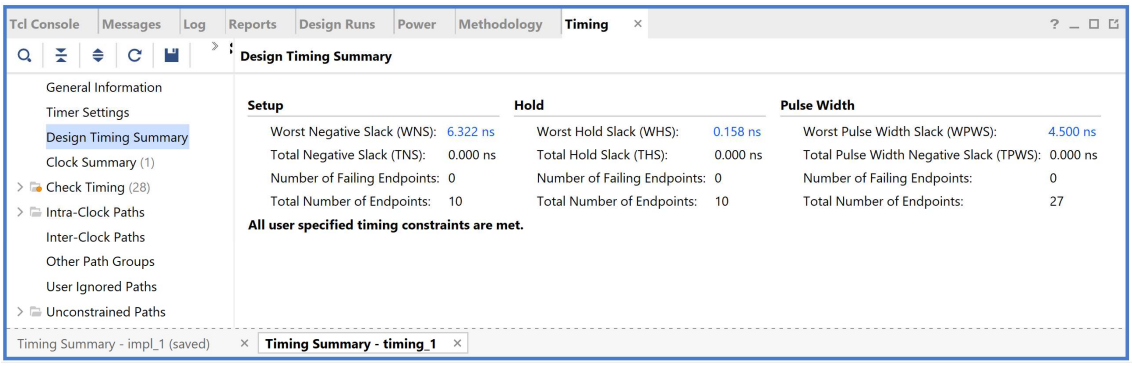


5-1.7. Στη συνέχεια, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Implemented Design* επιλέξετε το **Report Timing Summary** για να δείτε την ανάλυση χρονισμού που κάνει το εργαλείο Vivado IDE στο **implemented design model**. Αυτή είναι η πιο ακριβής ανάλυση χρονισμού που έχουμε διαθέσιμη.

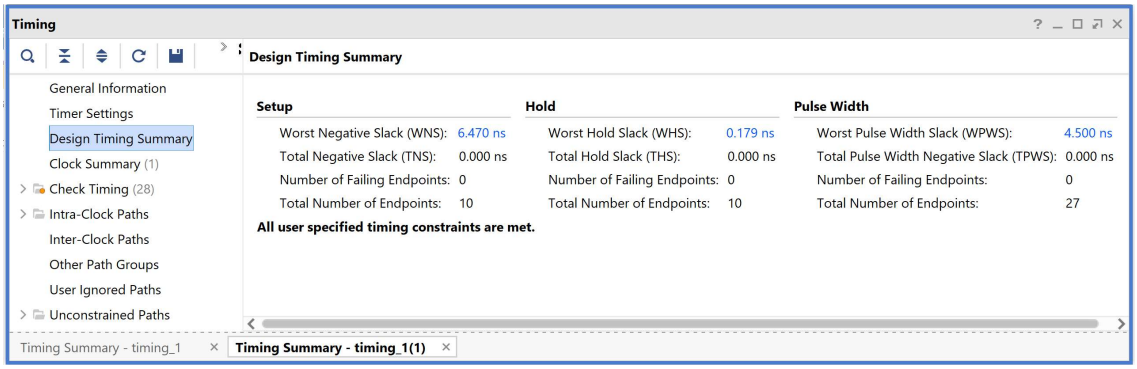


Στην επιλογή *Path delay type* ορίζεται ο τύπος της ανάλυσης που θα εκτελεσθεί. Το *max delay analysis* αφορά στην εύρεση της κρίσιμης διαδρομής (με τη μεγαλύτερη καθυστέρηση διάδοσης) και συνεπώς στην εύρεση της μέγιστης συχνότητας λειτουργίας χωρίς την παραβίαση του χρόνου σταθεροποίησης (setup time). Το *min delay analysis* αφορά στην εύρεση της σύντομης διαδρομής (με τη μικρότερη καθυστέρηση διάδοσης) και συνεπώς στην πιθανή παραβίαση του χρόνου διατήρησης (hold time).

5-1.8. Πατήστε **OK** για να παραχθεί το Timing_1 report.



Συγκρίνετε με το Timing_1 report που παράχθηκε μετά τη σύνθεση.



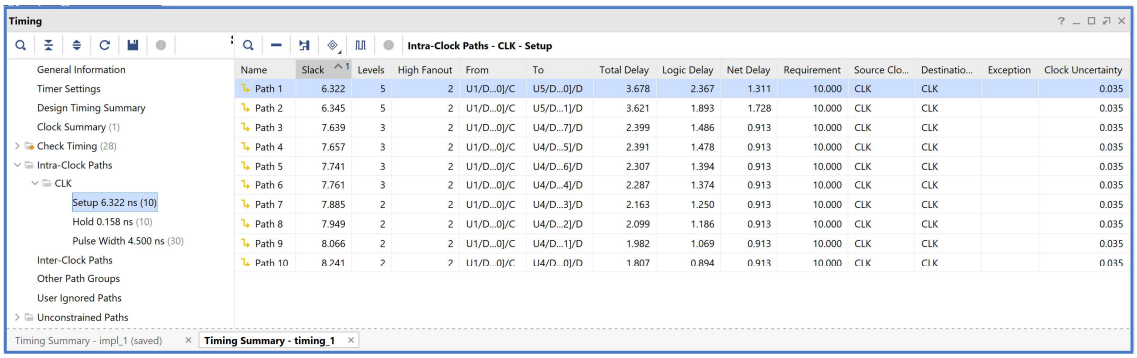
Στη στήλη **Setup** παρουσιάζονται τα αποτελέσματα του *max delay analysis*.

- Το *Worst Negative Slack* (**WNS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των κρίσιμων διαδρομών του max delay analysis. Μπορεί να είναι θετικό η αρνητικό. Ένα θετικό slack (6.322 ns) δηλώνει ότι η κρίσιμη διαδρομή ικανοποιεί την περίοδο του CLK. Ένα αρνητικό slack δηλώνει ότι παραβιάζεται ο χρόνος σταθεροποίησης (setup time).
- Το *Total Negative Slack* (**TNS**) είναι το άθροισμα όλων των αρνητικών WNS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου σταθεροποίησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά στην επιλεγμένη συχνότητα λειτουργίας.

Στη στήλη **Hold** παρουσιάζονται τα αποτελέσματα του min delay analysis.

- Το *Worst Hold Slack* (**WHS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των σύντομων διαδρομών του min delay analysis. Μπορεί να είναι θετικό η αρνητικό. Ένα αρνητικό slack δηλώνει πόσο παραβιάζεται ο χρόνος διατήρησης (hold time).
- Το *Total Hold Slack* (**THS**) είναι το άθροισμα όλων των αρνητικών WHS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου διατήρησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά.

5-1.9. Επιλέξτε το **WNS link** και δείτε τις 10 χειρότερες κρίσιμες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο σταθεροποίησης. Η κρίσιμη διαδρομή έχει καθυστέρηση διάδοσης 3.678 ns, εκ των οποίων τα 2.367 ns αφορούν στη λογική (logic), ενώ τα 1.311 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.035 ns. Τα επίπεδα λογικής (logic level) είναι 5.



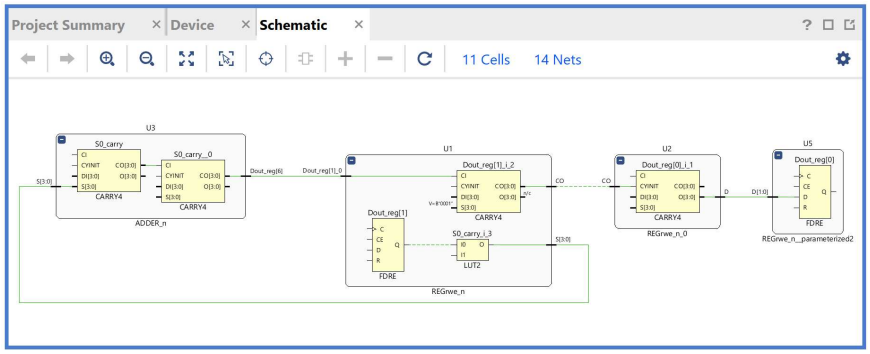
Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clo...	Destinatio...	Exception	Clock Uncertainty
Path 1	6.322	5	2	U1/D...0]/C	U5/D...0]/D	3.678	2.367	1.311	10.000	CLK	CLK		0.035
Path 2	6.345	5	2	U1/D...0]/C	U5/D...1]/D	3.621	1.893	1.728	10.000	CLK	CLK		0.035
Path 3	7.639	3	2	U1/D...0]/C	U4/D...7]/D	2.399	1.486	0.913	10.000	CLK	CLK		0.035
Path 4	7.657	3	2	U1/D...0]/C	U4/D...5]/D	2.391	1.478	0.913	10.000	CLK	CLK		0.035
Path 5	7.741	3	2	U1/D...0]/C	U4/D...6]/D	2.307	1.394	0.913	10.000	CLK	CLK		0.035
Path 6	7.761	3	2	U1/D...0]/C	U4/D...4]/D	2.287	1.374	0.913	10.000	CLK	CLK		0.035
Path 7	7.885	2	2	U1/D...0]/C	U4/D...3]/D	2.163	1.250	0.913	10.000	CLK	CLK		0.035
Path 8	7.949	2	2	U1/D...0]/C	U4/D...2]/D	2.099	1.186	0.913	10.000	CLK	CLK		0.035
Path 9	8.066	2	2	U1/D...0]/C	U4/D...1]/D	1.982	1.069	0.913	10.000	CLK	CLK		0.035
Path 10	8.241	2	2	U1/D...0]/C	U4/D...0]/D	1.807	0.894	0.913	10.000	CLK	CLK		0.035

5-1.10. Επιλέξτε με διπλό κλικ το **Path 1**, ώστε να εμφανιστεί το παράθυρο *Path 1 – timing_1*. Η κρίσιμη διαδρομή περνάει από 4 μονάδες CARRY4 και 1 LUT2 (πύλη XOR). Υπολογίστε το WNS slack (σε παρένθεση τα αποτελέσματα της σύνθεσης):

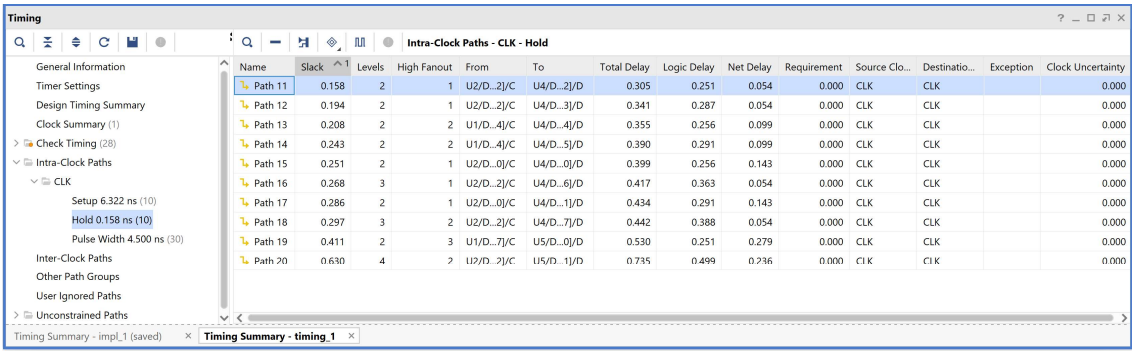
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U1) είναι **5.637** (2.975) ns,
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή της υπομονάδας U1 και μέσω του LUT2 (U1) και των 4 μονάδων CARRY4 (U3-U1-U2) μέχρι την είσοδο D του καταχωρητή της υπομονάδας U5) είναι **3.678** (3.426) ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι **9.315** (6.401) ns,
- το **Required Time**, ως η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή **10.000** ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U5) συν τον χρόνο σταθεροποίησης είναι **15.637** (12.871) ns.

WNS slack = Required Time – Arrival Time = 15.637 – 9.315 = 6.322 ns

5-1.11. Επιλέξτε με δεξί κλικ στο **Path 1**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της κρίσιμης διαδρομής της οντότητας **ADDER_REG_8**. (Δεν έχει αλλάξει η διαδρομή).



5-1.12. Επιλέξτε το **WHS link** και δείτε τις 10 χειρότερες σύντομες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο διατήρησης. Η σύντομη διαδρομή έχει καθυστέρηση μόλυνσης 0.305 ns, εκ των οποίων τα 0.251 ns αφορούν στη λογική (logic), ενώ τα 0.054 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.000 ns. Τα επίπεδα λογικής (logic level) είναι 2 (ήταν 1 μετά τη σύνθεση).

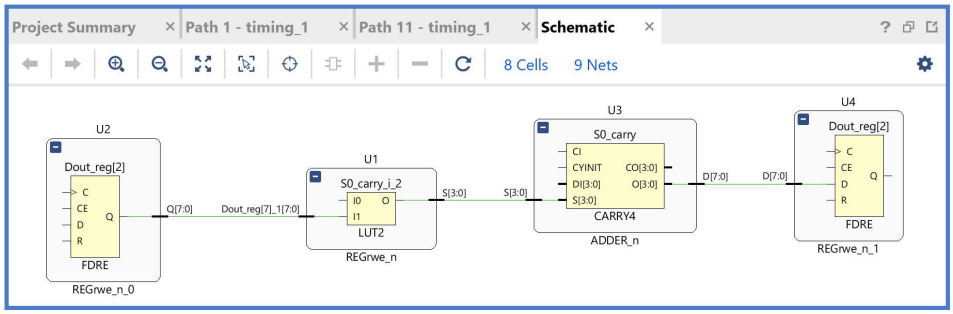


5-1.13. Επιλέξτε με διπλό κλικ το **Path 11**, ώστε να εμφανιστεί το παράθυρο *Path 11 – timing_1*. Η σύντομη διαδρομή περνάει από 1 μονάδα CARRY4 και 1 LUT2. Υπολογίστε το WHS slack (σε παρένθεση τα αποτελέσματα της σύνθεσης):

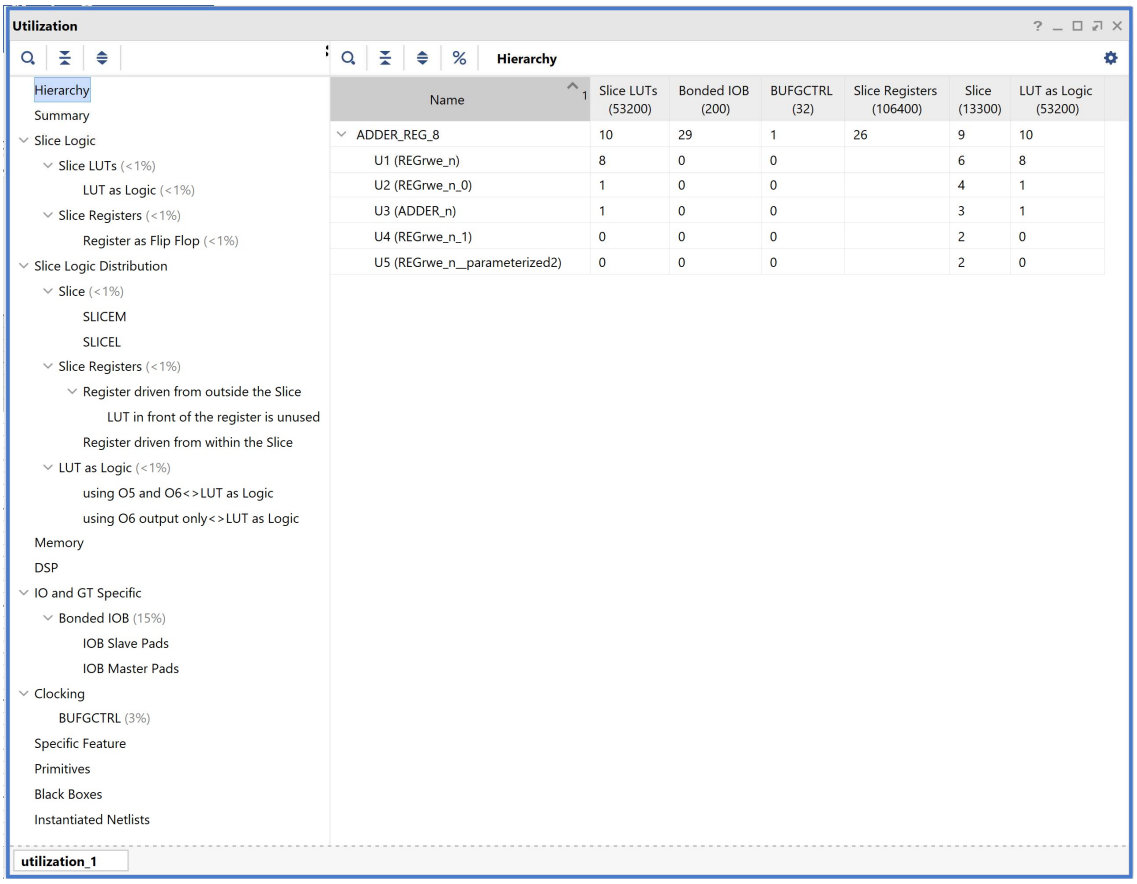
- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U2) είναι **1.583** (0.735) ns,
- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή U2 και μέσω του LUT2 (U1) και 1 μονάδας CARRY4 (U3) μέχρι την είσοδο D του καταχωρητή U4) είναι **0.305** (0.438) ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι **1.888** (1.173) ns,
- το **Required Time**, ως η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U4) συν τον χρόνο διατήρησης, είναι **1.730** (0.993) ns.

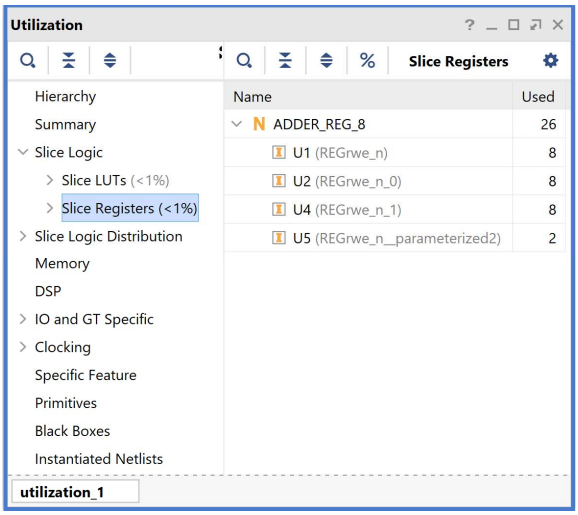
WHS slack = Arrival Time – Required Time = 1.888 – 1.730 = 0.158 ns

5-1.14. Επιλέξτε με δεξί κλικ στο **Path 11**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της σύντομης διαδρομής της οντότητας **ADDER_REG_8**. (Έχει αλλάξει η διαδρομή).



5-1.15. Τέλος, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Implemented Design* επιλέξετε το **Report Utilization** και μελετήστε τους πόρους που χρησιμοποιεί η οντότητα **ADDER_REG_8**. Πατήστε **OK**. (Μπορείτε να μελετήσετε και άλλα ενδιαφέροντα reports, εάν επιλέξετε το παράθυρο *Reports*).





5-2. Εύρεση συχνότητας λειτουργίας και ρύθμιση του σήματος CLK στο VIVADO IDE για συνδυαστική λογική ανάμεσα σε καταχωρητές

Όταν η διαδικασία bottom-up σχεδίασης και υλοποίησης ενός ψηφιακού συστήματος ολοκληρωθεί (η σχεδίαση έχει πλέον ωριμάσει στο επίπεδο της αποσφαλμάτωσης και της βελτιστοποίησης) και έχουμε φθάσει στη σχεδίαση και υλοποίηση της κορυφαίας οντότητας της ιεραρχίας του ψηφιακού συστήματός μας, προχωράμε στη διαδικασία της εύρεσης της μέγιστης συχνότητας λειτουργίας και της ρύθμισης του σήματος **CLK**. Αυτή η διαδικασία εφαρμόζεται στο **implemented design model** της οντότητας που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design resources.

Η συχνότητα λειτουργίας εξαρτάται από την τεχνολογία FPGA (π.χ. οικογένεια, speed grade), το περιβάλλον υλοποίησης (π.χ. θερμοκρασία, κατανάλωση ισχύος, ροή αέρα) και τους περιορισμούς που τίθενται σε επίπεδο board για τη διατήρηση του σήματος.

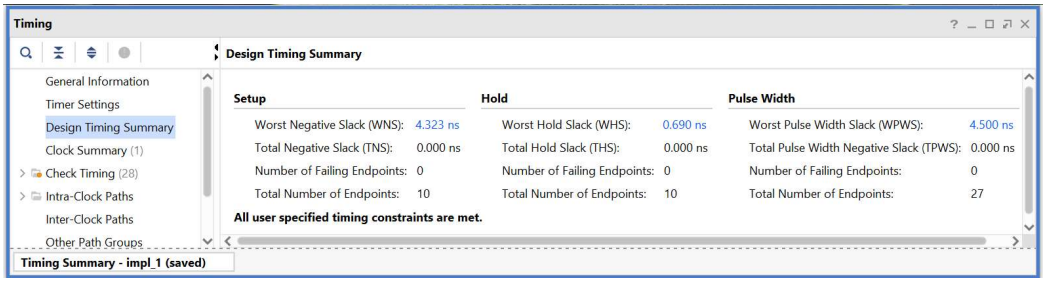
Για την ορθή εμφάνιση των εξόδων στο χρονικό διάγραμμα της χρονικής προσομοίωσης του **implemented design model** επιλέγουμε να τοποθετήσουμε τους **καταχωρητές εξόδου στα IOB** του FPGA.

5-2.1. Αρχικά, επιλέξτε να τοποθετήσετε τους καταχωρητές εξόδου στα **IOB** του FPGA. Στο παράθυρο *Sources*, επιλέξτε το constraint αρχείο **zedboard.xdc** και ανοίξτε το. Στο τέλος του αρχείου προσθέστε τον περιορισμό που υποχρεώνει το εργαλείο Vivado IDE να θέσει τους καταχωρητές εξόδου στα **IOB** του FPGA και πατήστε **Save**.

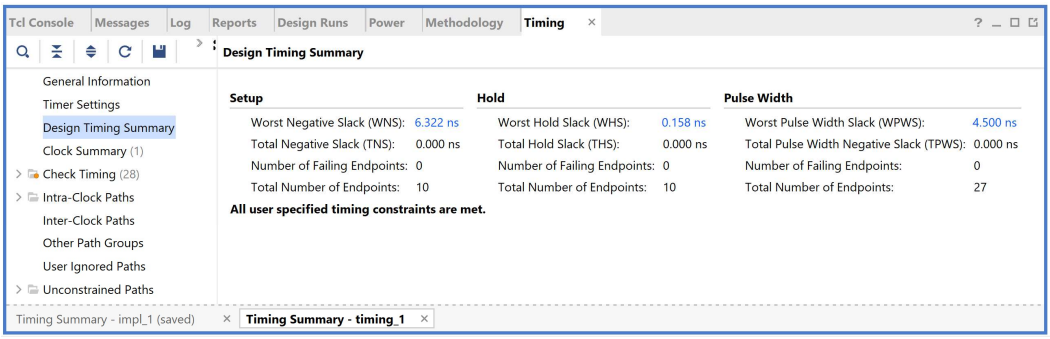
```
44 # Set output registers to IOB
45 set_property IOB TRUE [all_outputs]
```

5-2.2. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Open Implemented Design** και στη συνέχεια επιλέξτε το **Run Implementation**. Επαναλάβετε τη διαδικασία της σύνθεσης και της υλοποίησης πατώντας **Yes** και **OK**.

5-2.3. Στο παράθυρο *Implementation Completed* κρατήστε την επιλογή **Open Implemented Design** και πατήστε **OK**. Στο παράθυρο *Timing* βλέπετε την ανάλυση χρονισμού που κάνει το εργαλείο Vivado IDE στο **implemented design model** με τη χρήση των **IOB** του FPGA για περίοδο **CLK** στα **10.000** ns.



Συγκρίνετε με το timing_1 report μετά την υλοποίηση, χωρίς τη χρήση των **IOB**, με περίοδο **CLK** στα **10.000** ns.



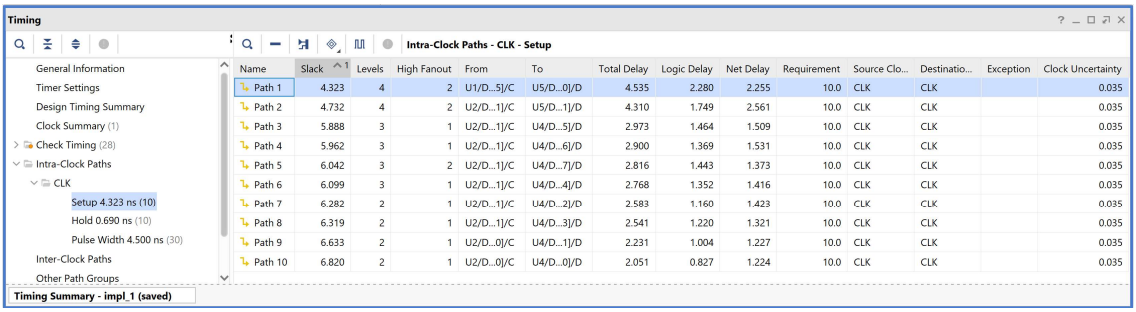
Στη στήλη **Setup** παρουσιάζονται τα αποτελέσματα του *max delay analysis*.

- Το *Worst Negative Slack* (**WNS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των κρίσιμων διαδρομών του max delay analysis. Μπορεί να είναι θετικό η αρνητικό. Ένα θετικό slack (4.323 ns) δηλώνει ότι η κρίσιμη διαδρομή ικανοποιεί την περίοδο του CLK. Η τοποθέτηση των καταχωρητών εξόδου στα **IOB** του FPGA **μειώνουν** το **WNS** από **6.322** σε **4.323**!
- Το *Total Negative Slack* (**TNS**) είναι το άθροισμα όλων των αρνητικών WNS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου σταθεροποίησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά στην επιλεγμένη συχνότητα λειτουργίας.

Στη στήλη **Hold** παρουσιάζονται τα αποτελέσματα του min delay analysis.

- Το *Worst Hold Slack* (**WHS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των σύντομων διαδρομών του min delay analysis. Μπορεί να είναι θετικό η αρνητικό. Ένα θετικό slack δηλώνει ότι δεν παραβιάζεται ο χρόνος διατήρησης (hold time). Η τοποθέτηση των καταχωρητών εξόδου στα **IOB** του FPGA **αυξάνουν** το **WHS** από **0.158** σε **0.690**!
- Το *Total Hold Slack* (**THS**) είναι το άθροισμα όλων των αρνητικών WHS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου διατήρησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά.

5-2.4. Επιλέξτε το **WNS link** και δείτε τις 10 χειρότερες κρίσιμες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο σταθεροποίησης. Η κρίσιμη διαδρομή έχει καθυστέρηση διάδοσης 4.535 ns, εκ των οποίων τα 2.280 ns αφορούν στη λογική (logic), ενώ τα 2.255 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.035 ns. Τα επίπεδα λογικής (logic level) είναι πλέον 4 (δεν συμπεριλαμβάνεται μία μονάδα CARRY4).



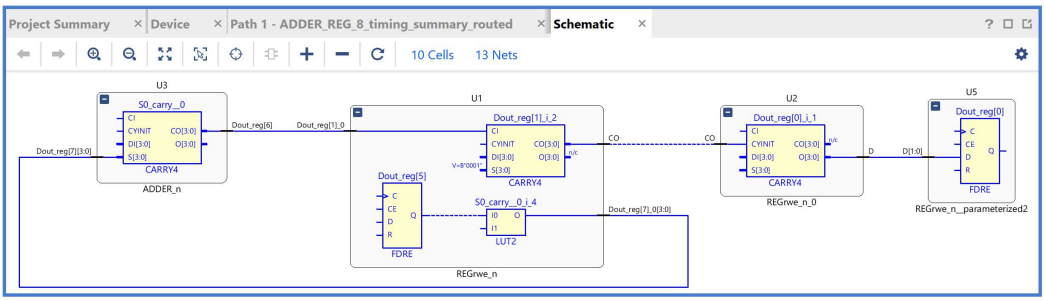
Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clo...	Destinatio...	Exception	Clock Uncertainty
Path 1	4.323	4	2	U1/D...5]/C	U5/D...0]/D	4.535	2.280	2.255	10.0	CLK	CLK		0.035
Path 2	4.732	4	2	U2/D...1]/C	U5/D...1]/D	4.310	1.749	2.561	10.0	CLK	CLK		0.035
Path 3	5.888	3	1	U2/D...1]/C	U4/D...5]/D	2.973	1.464	1.509	10.0	CLK	CLK		0.035
Path 4	5.962	3	1	U2/D...1]/C	U4/D...6]/D	2.900	1.369	1.531	10.0	CLK	CLK		0.035
Path 5	6.042	3	2	U2/D...1]/C	U4/D...7]/D	2.816	1.443	1.373	10.0	CLK	CLK		0.035
Path 6	6.099	3	1	U2/D...1]/C	U4/D...4]/D	2.768	1.352	1.416	10.0	CLK	CLK		0.035
Path 7	6.282	2	1	U2/D...1]/C	U4/D...2]/D	2.583	1.160	1.423	10.0	CLK	CLK		0.035
Path 8	6.319	2	1	U2/D...1]/C	U4/D...3]/D	2.541	1.220	1.321	10.0	CLK	CLK		0.035
Path 9	6.633	2	1	U2/D...0]/C	U4/D...1]/D	2.231	1.004	1.227	10.0	CLK	CLK		0.035
Path 10	6.820	2	1	U2/D...0]/C	U4/D...0]/D	2.051	0.827	1.224	10.0	CLK	CLK		0.035

5-2.5. Επιλέξτε με διπλό κλικ το **Path 1**, ώστε να εμφανιστεί το παράθυρο *Path 1 – ADDER_REG_8_timing_summary_routed*. Η κρίσιμη διαδρομή περνάει από 3 μονάδες CARRY4 και 1 LUT2. Υπολογίστε το WNS slack (σε παρένθεση τα αποτελέσματα της υλοποίησης, χωρίς τη χρήση IOB):

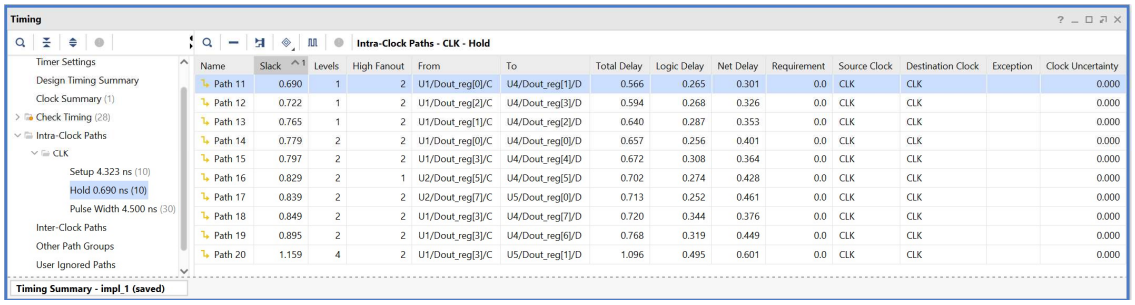
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U1) είναι **5.632** (5.637) ns,
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή της υπομονάδας U1, του LUT2 (U1) και των 3 μονάδων CARRY4 (U3-U1-U2) μέχρι την είσοδο D του καταχωρητή της υπομονάδας U5) είναι **4.535** (3.678) ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι **10.167** (9.315) ns,
- το **Required Time**, ως η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή **10.000** ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U5) συν τον χρόνο σταθεροποίησης (-1.016) είναι **14.490** (15.637) ns.

WNS slack = Required Time – Arrival Time = 14.490 – 10.167 = 4.323 ns

5-2.6. Επιλέξτε με δεξί κλικ στο **Path 1**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της κρίσιμης διαδρομής της οντότητας **ADDER_REG_8**. (Έχει αλλάξει η διαδρομή).



5-2.7. Επιλέξτε το **WHS link** και δείτε τις 10 χειρότερες σύντομες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο διατήρησης. Η σύντομη διαδρομή έχει καθυστέρηση μόλυνσης 0.566 ns, εκ των οποίων τα 0.265 ns αφορούν στη λογική (logic), ενώ τα 0.301 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.000 ns. Τα επίπεδα λογικής (logic level) είναι 1 (ήταν 2 μετά την υλοποίηση, χωρίς τη χρήση IOB).



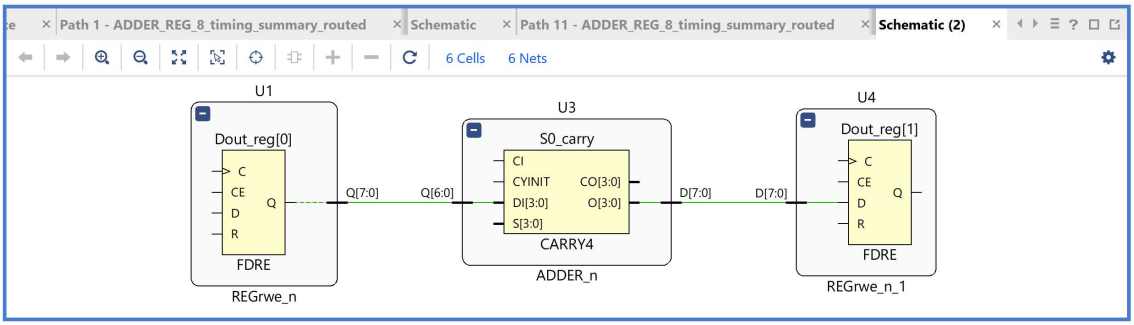
Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 11	0.690	1	2	U1/Dout_reg[0]/C	U4/Dout_reg[1]/D	0.566	0.265	0.301	0.0	CLK	CLK		0.000
Path 12	0.722	1	2	U1/Dout_reg[2]/C	U4/Dout_reg[3]/D	0.594	0.268	0.326	0.0	CLK	CLK		0.000
Path 13	0.765	1	2	U1/Dout_reg[1]/C	U4/Dout_reg[2]/D	0.640	0.287	0.353	0.0	CLK	CLK		0.000
Path 14	0.779	2	2	U1/Dout_reg[0]/C	U4/Dout_reg[0]/D	0.657	0.256	0.401	0.0	CLK	CLK		0.000
Path 15	0.797	2	2	U1/Dout_reg[3]/C	U4/Dout_reg[4]/D	0.672	0.308	0.364	0.0	CLK	CLK		0.000
Path 16	0.829	2	1	U2/Dout_reg[5]/C	U4/Dout_reg[5]/D	0.702	0.274	0.428	0.0	CLK	CLK		0.000
Path 17	0.839	2	2	U2/Dout_reg[7]/C	U5/Dout_reg[0]/D	0.713	0.252	0.461	0.0	CLK	CLK		0.000
Path 18	0.849	2	2	U1/Dout_reg[3]/C	U4/Dout_reg[7]/D	0.720	0.344	0.376	0.0	CLK	CLK		0.000
Path 19	0.895	2	2	U1/Dout_reg[3]/C	U4/Dout_reg[6]/D	0.768	0.319	0.449	0.0	CLK	CLK		0.000
Path 20	1.159	4	2	U1/Dout_reg[3]/C	U5/Dout_reg[1]/D	1.096	0.495	0.601	0.0	CLK	CLK		0.000

5-2.8. Επιλέξτε με διπλό κλικ το **Path 11**, ώστε να εμφανιστεί το παράθυρο *Path 11 – ADDER_REG_8_timing_summary_routed*. Η σύντομη διαδρομή περνάει από 1 μονάδα CARRY4 της υπομονάδας U3. Υπολογίστε το WHS slack (σε παρένθεση τα αποτελέσματα της υλοποίησης, χωρίς τη χρήση IOB):

- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U1) είναι **1.580** (1.583) ns,
- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή της υπομονάδας U2 και μέσω του LUT2 (U1) και της 1 μονάδας CARRY4 (U3) μέχρι την είσοδο D του καταχωρητή της υπομονάδας U4) είναι **0.566** (0.305) ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι **2.146** (1.888) ns,
- το **Required Time**, ως η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U4) συν τον χρόνο διατήρησης (-0.155), είναι **1.456** (1.730) ns.

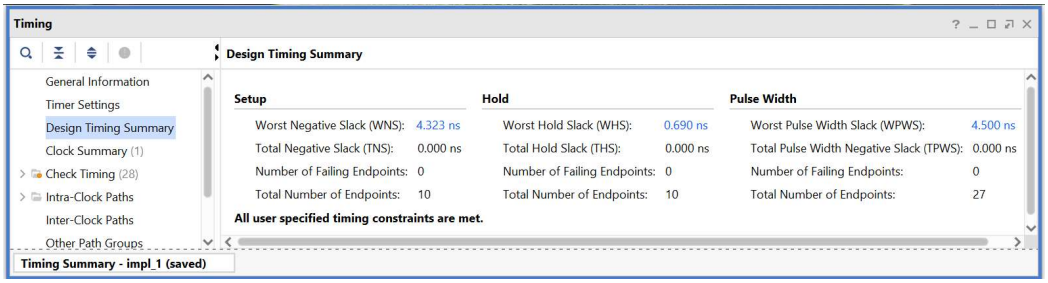
WHS slack = Arrival Time – Required Time = 2.146 – 1.456 = 0.690 ns

5-2.9. Επιλέξτε με δεξί κλικ στο **Path 11**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της σύντομης διαδρομής της οντότητας **ADDER_REG_8**. (Έχει αλλάξει διαδρομή).

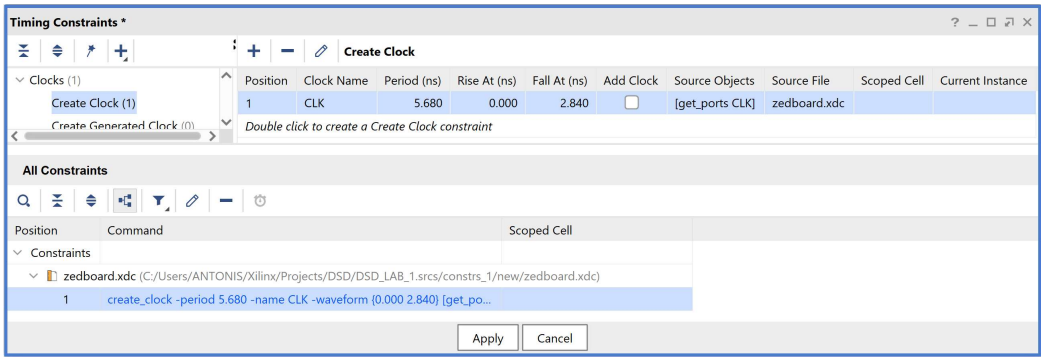


5-2.10. Στη συνέχεια, με βάση το **Design Timing Summary** του παράθυρου *timing* ορίστε τη νέα περίοδο του σήματος **CLK**:

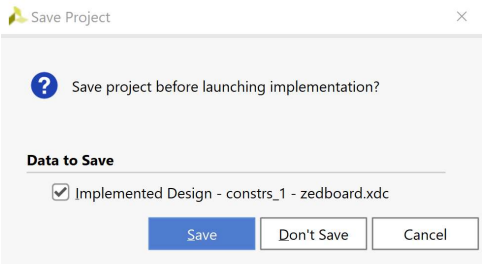
New_CLK_period = CLK_period – WNS = 10.000 – 4.320 = 5.680 ns



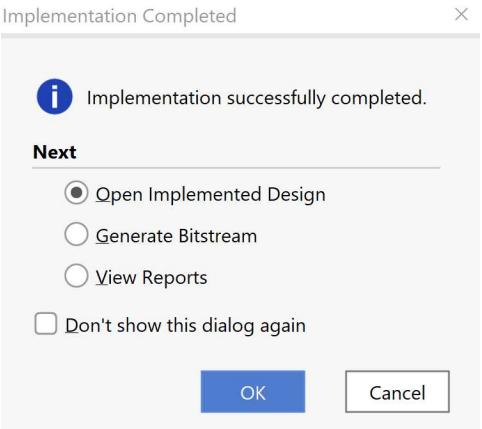
5-2.11. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Open Implemented Design** και στη συνέχεια επιλέξτε το **Edit Timing Constraints**, ώστε να δημιουργήσετε ένα νέο clock constraint. Στο παράθυρο *Timing Constraints* αλλάξτε με διπλό κλικ το **Period** από 10.000 ns σε **5.680 ns** (10.000 - 4.320 ns) και το **Fall At** από 5.000 ns σε **2.840 ns**. Πατήστε **Apply**.



5-2.12. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Implementation**. Στο παράθυρο *Save Project* επιλέξτε **Save**, για να αποθηκευτεί η νέα περίοδος του σήματος **CLK**. Στη συνέχεια πατήστε **OK** και **Update**. Επαναλάβετε τη διαδικασία της σύνθεσης και της υλοποίησης πατώντας **Yes** και **OK**.

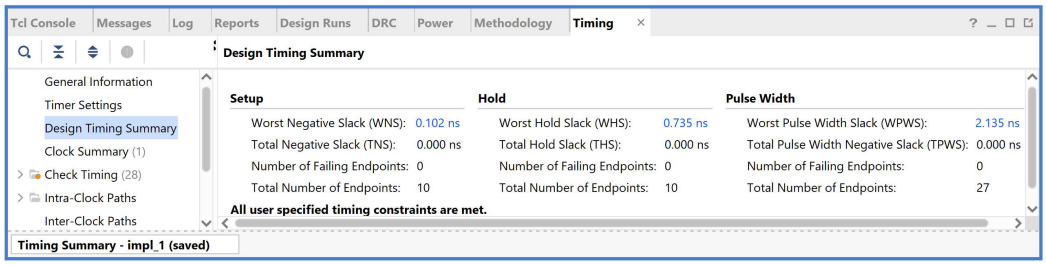


5-2.13. Στο παράθυρο *Implementation Completed*, διατηρήστε την επιλογή **Open Implemented Design** και πατήστε **OK**.

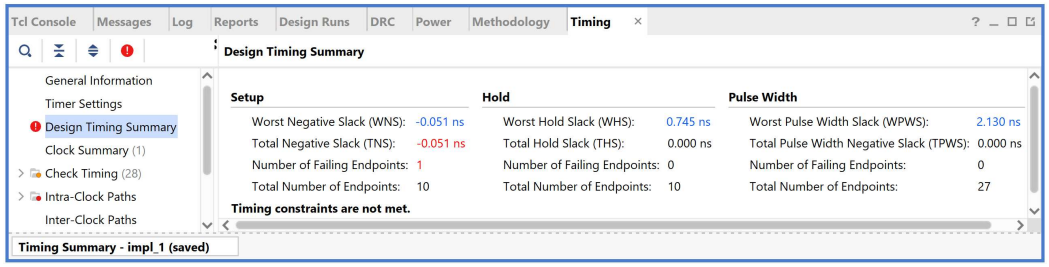


5-2.14. Επαναλάβετε τα βήματα **5-2.11-14** μέχρι να προκύψει ένα αρνητικό WNS. Επιλέξτε, τη μικρότερη περίοδο με θετικό WNS. Δοκιμάστε τις περιόδους: **5.680 ns** (WNS = 0.287), **5.400 ns** (WNS = 0.135), **5.270 ns** (WNS = 0.102), **5.260 ns** (WNS = -0.051).

Μικρότερη περίοδος με θετικό WNS (**5.270 ns**)



Μεγαλύτερη περίοδος με αρνητικό WNS (**5.260 ns**)



Με βάση τα impl_1 report επιλέγουμε την περίοδο των **5.270 ns** ως μία ικανοποιητική περίοδο για το σήμα **CLK**. Μέγιστη συχνότητα λειτουργίας: **189.8 MHz**.

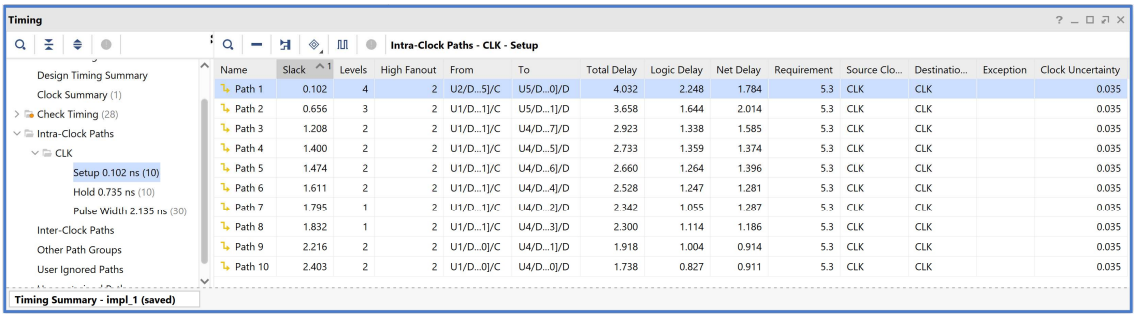
Στη στήλη **Setup** παρουσιάζονται τα αποτελέσματα του *max delay analysis*.

- Το *Worst Negative Slack* (**WNS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των κρίσιμων διαδρομών του max delay analysis. Ένα θετικό slack (0.102 ns) δηλώνει ότι η κρίσιμη διαδρομή ικανοποιεί την περίοδο του CLK.
- Το *Total Negative Slack* (**TNS**) είναι το άθροισμα όλων των αρνητικών WNS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου σταθεροποίησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά στην επιλεγμένη συχνότητα λειτουργίας.

Στη στήλη **Hold** παρουσιάζονται τα αποτελέσματα του min delay analysis.

- Το *Worst Hold Slack* (**WHS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των σύντομων διαδρομών του min delay analysis. Ένα θετικό slack (0.735) δηλώνει ότι δεν παραβιάζεται ο χρόνος διατήρησης (hold time).
- Το *Total Hold Slack* (**THS**) είναι το άθροισμα όλων των αρνητικών WHS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου διατήρησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά.

5-2.15. Επιλέξτε το **WNS link** και δείτε τις 10 χειρότερες κρίσιμες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο σταθεροποίησης. Η κρίσιμη διαδρομή έχει καθυστέρηση διάδοσης 4.032 ns, εκ των οποίων τα 2.248 ns αφορούν στη λογική (logic), ενώ τα 1.784 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.035 ns. Τα επίπεδα λογικής (logic level) είναι 4.

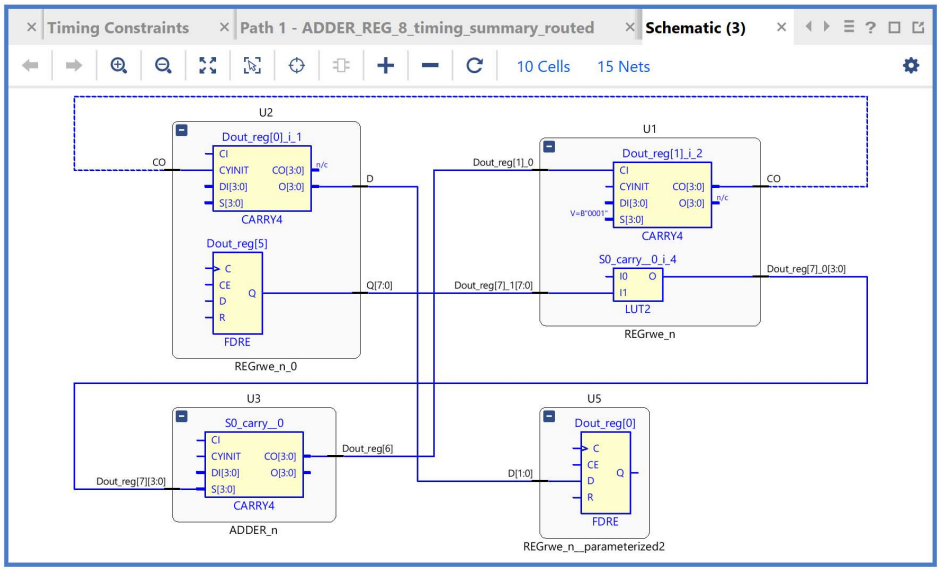


5-2.16. Επιλέξτε με διπλό κλικ το **Path 1**, ώστε να εμφανιστεί το παράθυρο *Path 1 – ADDER_REG_8_timing_summary_routed*. Η κρίσιμη διαδρομή περνάει από 3 μονάδες CARRY4 και 1 LUT2. Υπολογίστε το WNS slack (σε παρένθεση τα αποτελέσματα της υλοποίησης με σήμα **CLK** στα 10.000 ns):

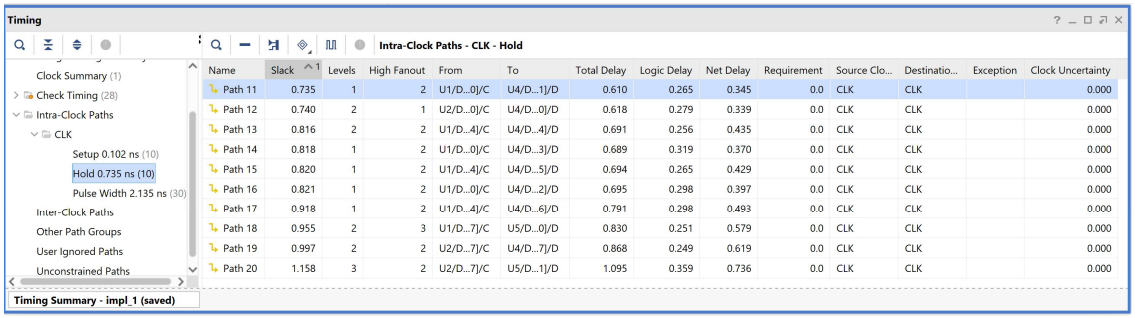
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U2) είναι **5.629** (5.632) ns,
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή της υπομονάδας U2, το LUT2 (U1) και των 3 μονάδων CARRY4 (U3-U1-U2) μέχρι την είσοδο D του καταχωρητή της υπομονάδας U5) είναι **4.032** (4.535) ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι **9.661** (10.167) ns,
- το **Required Time**, ως η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή **5.270** ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U5) συν τον χρόνο σταθεροποίησης (-1.013) είναι **9.763** (14.490) ns.

WNS slack = Required Time – Arrival Time = 9.763 – 9.661 = 0.102 ns

5-2.17. Επιλέξτε με δεξί κλικ στο **Path 1**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της κρίσιμης διαδρομής της οντότητας **ADDER_REG_8**. (Έχει αλλάξει η διαδρομή).



5-2.18. Επιλέξτε το **WHS link** και δείτε τις 10 χειρότερες σύντομες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο διατήρησης). Η σύντομη διαδρομή έχει καθυστέρηση μόλυνσης 0.610 ns, εκ των οποίων τα 0.265 ns αφορούν στη λογική (logic), ενώ τα 0.345 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.000 ns. Τα επίπεδα λογικής (logic level) είναι 1.

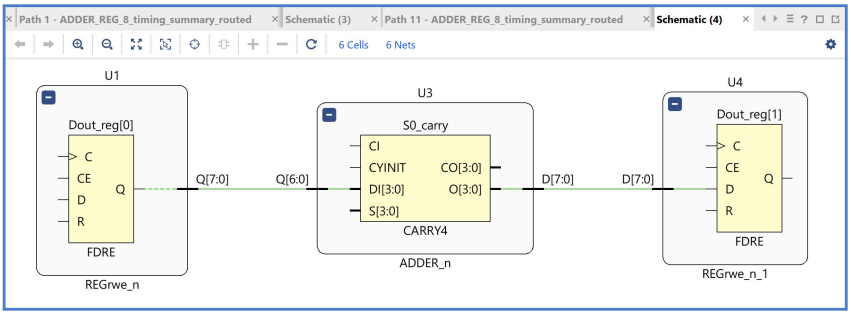


5-2.19. Επιλέξτε με διπλό κλικ το **Path 11**, ώστε να εμφανιστεί το παράθυρο *Path 11 – ADDER_REG_8_timing_summary_routed*. Η σύντομη διαδρομή περνάει από 1 μονάδα CARRY4. Υπολογίστε το WHS slack (σε παρένθεση τα αποτελέσματα της υλοποίησης με σήμα **CLK** με περίοδο των 10.000 ns):

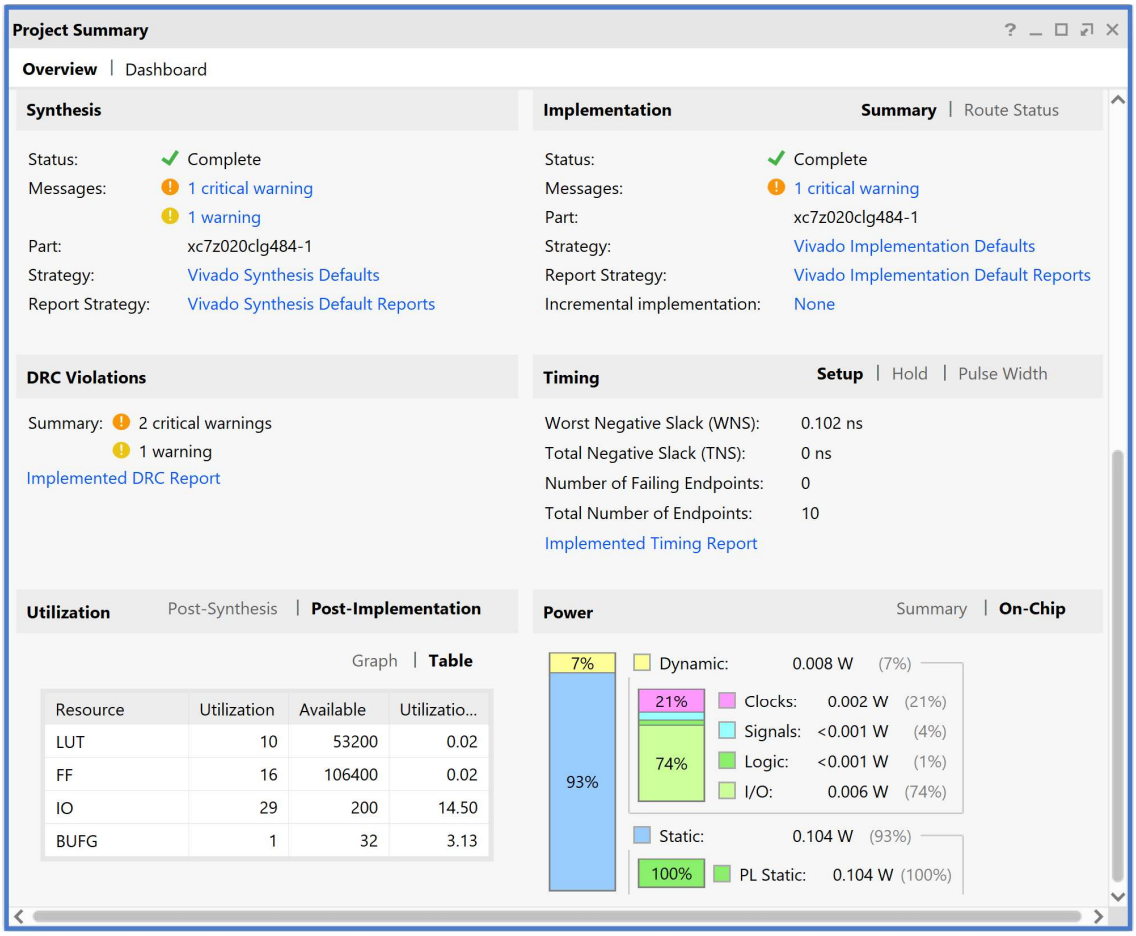
- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U1) είναι **1.581** (1.580) ns,
- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή της υπομονάδας U1 και μέσω της 1 μονάδας CARRY4 (U3) μέχρι την είσοδο D του καταχωρητή της υπομονάδας U4) είναι **0.610** (0.566) ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι **2.191** (2.146) ns,
- το **Required Time**, ως η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή της υπομονάδας U4) συν τον χρόνο διατήρησης (-0.155), είναι **1.456** (1.456) ns.

WHS slack = Arrival Time – Required Time = 2.191 – 1.456 = 0.735 ns

5-2.20. Επιλέξτε με δεξί κλικ στο **Path 11**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της σύντομης διαδρομής της οντότητας **ADDER_REG_8**. (Έχει αλλάξει η διαδρομή).

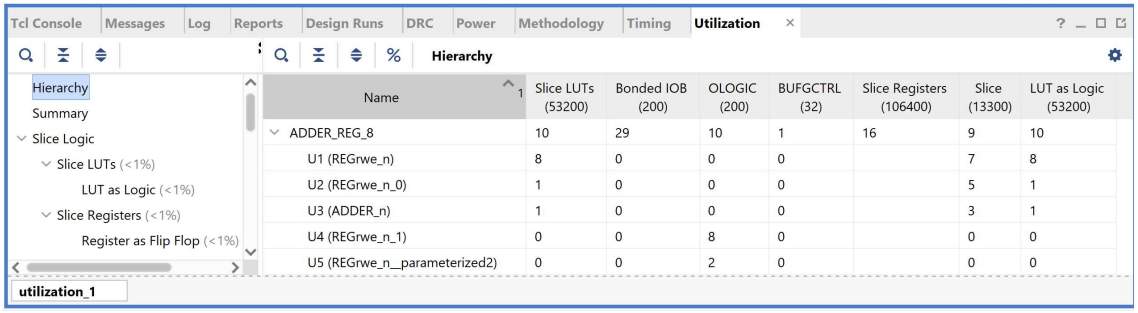


5-2.21. Στη συνέχεια, επιλέξτε το παράθυρο *Project Summary* και μελετήστε τα διάφορα υπο-παράθυρα.



5-2.22. Στη συνέχεια, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Implemented Design* επιλέξετε το **Schematic** για να δείτε το σχηματικό διάγραμμα του **implemented design model**. Δεν έχει αλλάξει με την αλλαγή της περιόδου του σήματος **CLK**.

5-2.23. Τέλος, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Implemented Design* επιλέξετε το **Report Utilization** και μελετήστε τους πόρους που χρησιμοποιεί η οντότητα **ADDER_REG_8**. Πατήστε **OK**. (Μπορείτε να μελετήσετε και άλλα ενδιαφέροντα reports, εάν επιλέξετε το παράθυρο *Reports*).



5-3. Εκτέλεση προσομοίωσης μετά την υλοποίηση (λογική και χρονική) στο VIVADO IDE για συνδυαστική λογική ανάμεσα σε καταχωρητές

Η προσομοίωση μετά την υλοποίηση (λογική και χρονική) εκτελείται στην οντότητα **ADDER_REG_8_TB** του προγράμματος δοκιμών (testbench) που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των simulation resources. Κατά την προσομοίωση, η οντότητα **ADDER_REG_8_TB** καλεί το *UUT* της (που είναι το **implemented design model** της οντότητας **ADDER_REG_8**).

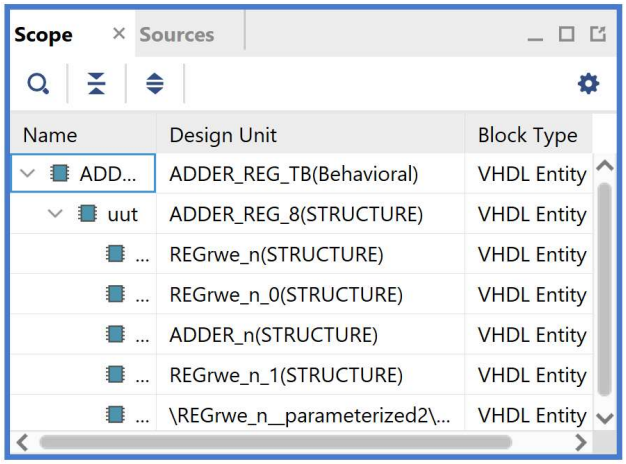
5-3.1. Στο αρχείο **ADDER_REG_8_TB.vhd** αλλάξτε την περίοδο του σήματος **CLK** στη βέλτιστη περίοδο που βρήκατε στο Βήμα 5-2. Πατήστε **Save**.

```
67 ¦ -- Clock period definitions
68 ¦ constant CLK_period : time := 5.270 ns;
```

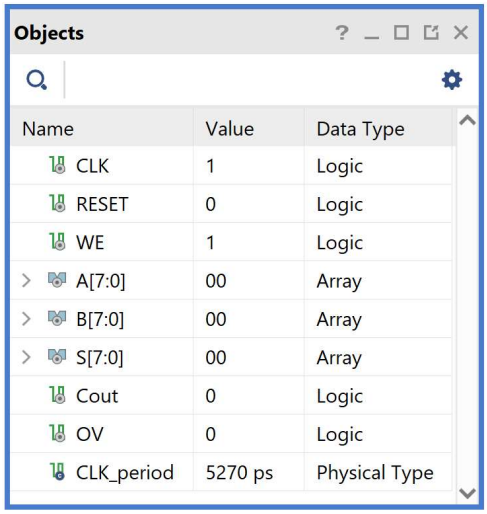
5-3.2. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Simulation**, ώστε να εμφανιστούν όλες οι πιθανές προσομοιώσεις που υποστηρίζει το Vivado IDE. Επιλέξτε **Run Post-Implementation Functional Simulation**.

5-3.3. Το πρόγραμμα δοκιμής (testbench) και όλες οι οντότητες του *UUT* θα γίνουν compiled και θα τρέξει το Vivado simulator (εφόσον βέβαια δεν υπάρχουν σφάλματα). Θα εμφανιστεί το παράθυρο *SIMULATION* που απαρτίζεται από 4 παράθυρα:

Το παράθυρο *Scope*, όπου παρουσιάζεται η οντότητα **ADDER_REG_8_TB**, το **implemented design model** της οντότητας **ADDER_ REG_8** (*UUT*) καθώς και οι υπόλοιπες νέες οντότητες του *UUT* που προκύπτουν μετά την υλοποίηση. Όλες οι οντότητες που απαρτίζουν το *UUT* είναι *structural*.

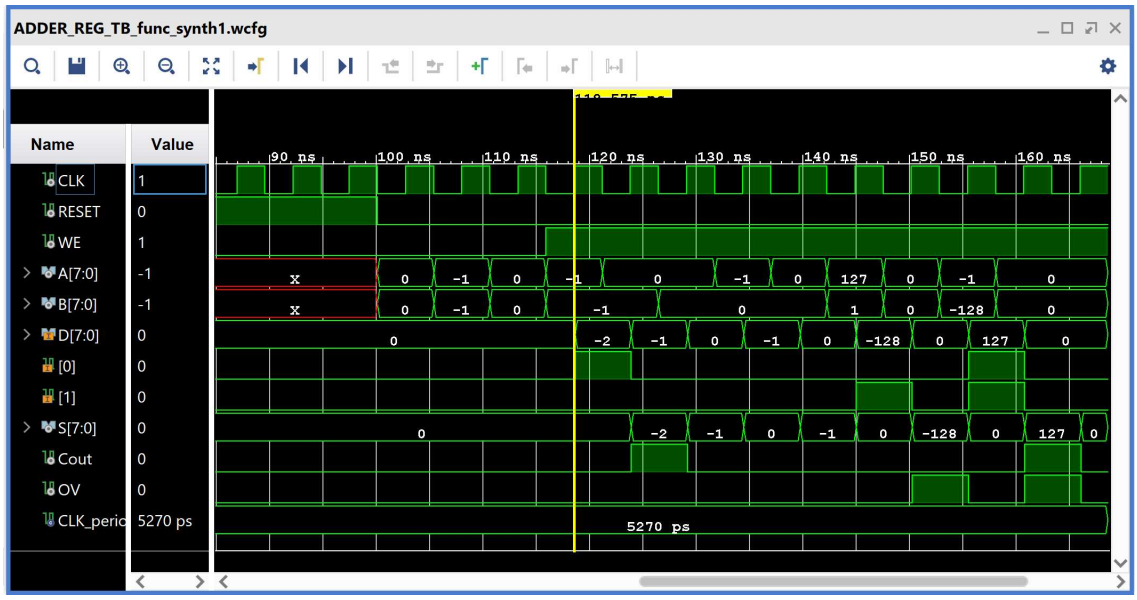


Το παράθυρο *Objects*, όπου εμφανίζονται τα σήματα top-level, δηλαδή οι είσοδοι και οι έξοδοι της οντότητας **ADDER_REG_8**, που είναι η κορυφαία οντότητα της ιεραρχίας του *UUT*, καθώς και η περίοδος του CLK (*CLK_period*). Οι τιμές αντιστοιχούν στις τιμές που έχει σταματήσει η προσομοίωση της οντότητας **ADDER_REG_8_TB** (st*op (2)*).



Το παράθυρο *Tcl Console* με όλες τις διαδικασίες που εκτελούνται στο πλαίσιο της προσομοίωσης. Το Tcl Console καθαρίζει με την επιλογή *Clear*.

Επιλέξτε το παράθυρο *ADDER_REG_TB_func_synth1.wcfg*, που δημιουργήσατε για την προσομοίωση του **synthesized design model**, όπου συμπεριλαμβάνονται πέραν των top-level εισόδων/εξόδων και τα απαραίτητα **εσωτερικά σήματα** της εξόδου του αθροιστή (**D[7:0]** που αντιστοιχεί στην έξοδο **S[7:0]**, **[0]** που αντιστοιχεί στο **Cout** και **[1]** που αντιστοιχεί στο **OV**. Βλέπετε ολόκληρο το διάγραμμα χρονισμού της λογικής προσομοίωσης μετά τη σύνθεση με κατάλληλο **zoom out** και επιλέγοντας το **zoom fit**.

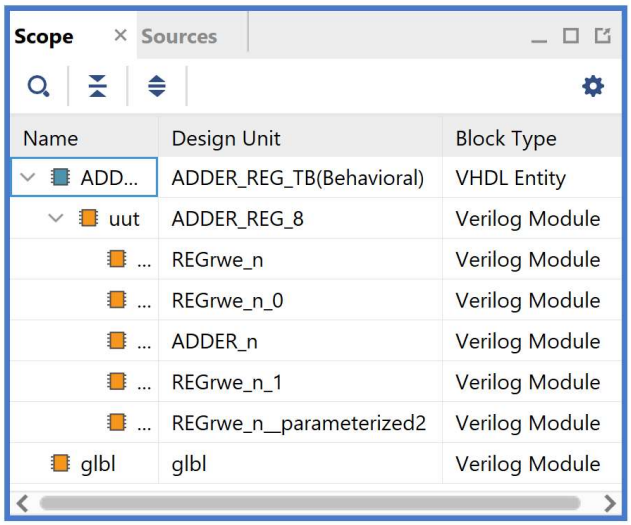


5-3.4. Κλείστε τον simulator επιλέγοντας το κουμπί × πάνω δεξιά στο παράθυρο του *SIMULATION*. Στο παράθυρο *Confirm Close* που εμφανίζεται πατήστε **OK**.

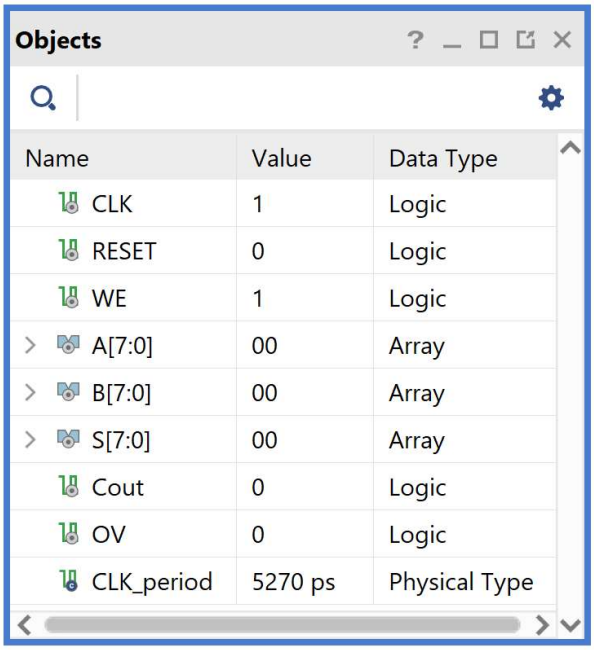
5-3.5. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Simulation**, ώστε να εμφανιστούν όλες οι πιθανές προσομοιώσεις που υποστηρίζει το Vivado IDE. Επιλέξτε **Run Post-Implementation Timing Simulation**.

5-3.6. Το πρόγραμμα δοκιμής (testbench) και όλες οι οντότητες του *UUT* θα γίνουν compiled και θα τρέξει το Vivado simulator (εφόσον βέβαια δεν υπάρχουν σφάλματα). Θα εμφανιστεί το παράθυρο *SIMULATION* που απαρτίζεται από 4 παράθυρα:

Το παράθυρο *Scope*, όπου παρουσιάζεται η οντότητα **ADDER_REG_8_TB**, το **implemented design model** της οντότητας **ADDER_REG_8** (*UUT*) καθώς και οι υπόλοιπες νέες οντότητες του *UUT* που προκύπτουν μετά τη σύνθεση για χρονική προσομοίωση. Όλες οι οντότητες που απαρτίζουν πλέον το *UUT* είναι *Verilog Module!*

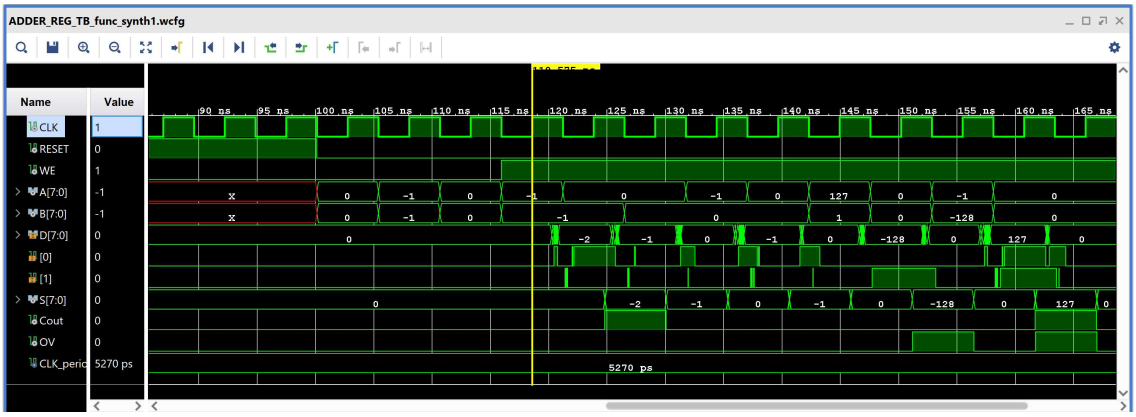


Το παράθυρο *Objects*, όπου εμφανίζονται τα σήματα top-level, δηλαδή οι είσοδοι και οι έξοδοι της οντότητας **ADDER_REG_8**, που είναι η κορυφαία οντότητα της ιεραρχίας του *UUT*, καθώς και η περίοδος του CLK (*CLK_period*). Οι αρτηρίες (array) αναλύονται στα σήματα που τις απαρτίζουν. Οι τιμές αντιστοιχούν στις τιμές που έχει σταματήσει η προσομοίωση της οντότητας **ADDER_REG_8_TB** (st*op (2)*).

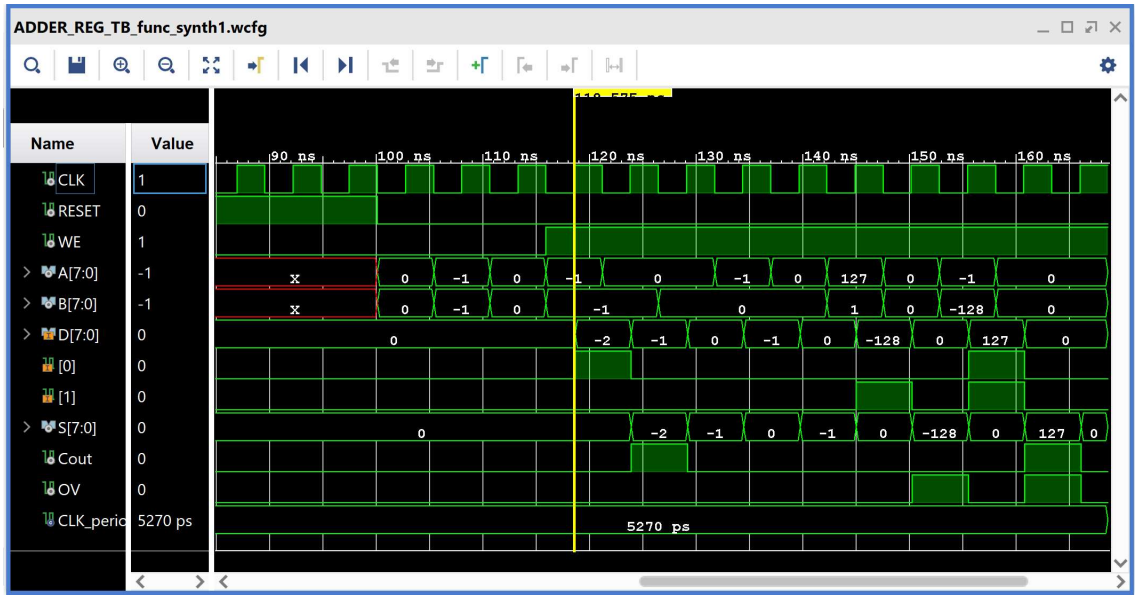


Το παράθυρο *Tcl Console* με όλες τις διαδικασίες που εκτελούνται στο πλαίσιο της προσομοίωσης. Το Tcl Console καθαρίζει με την επιλογή *Clear*.

Επιλέξτε το παράθυρο *ADDER_REG_TB_func_synth1.wcfg*. Βλέπετε ολόκληρο το διάγραμμα χρονισμού της χρονικής προσομοίωσης μετά την υλοποίηση με κατάλληλο **zoom out** ή επιλέγοντας το **zoom fit** .

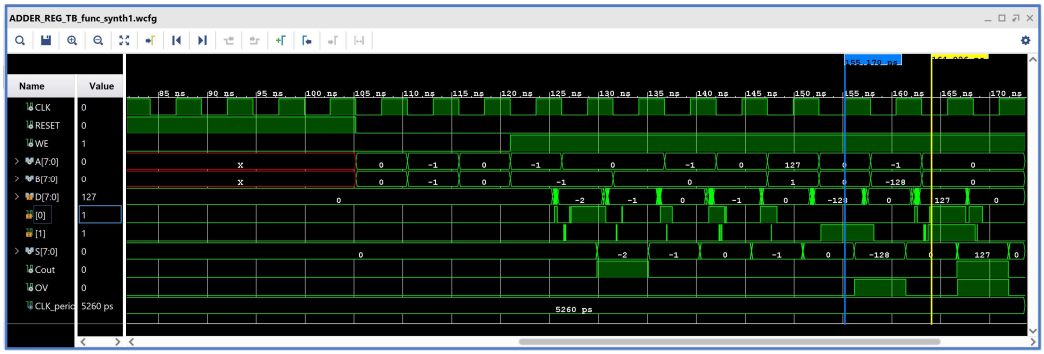


5-3.7. Συγκρίνετε τα διαγράμματα χρονισμού της χρονικής και λογικής προσομοίωσης μετά την υλοποίηση. Είναι εμφανείς οι καθυστερήσεις διάδοσης στο πρώτο διάγραμμα χρονισμού.

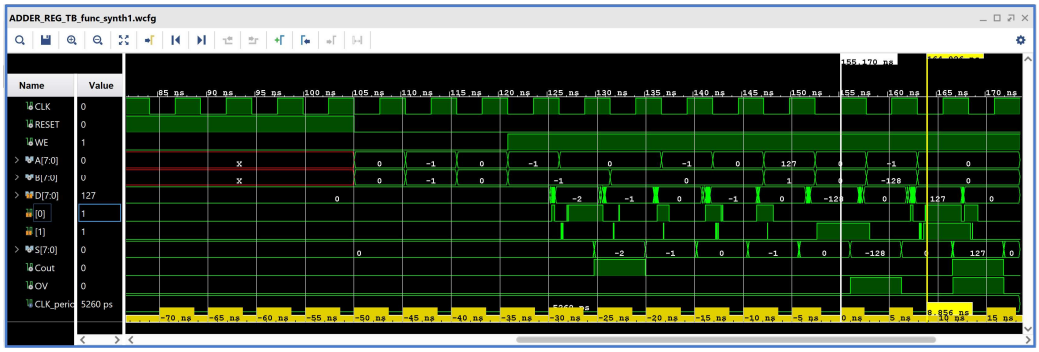


5-3.8. Υπολογίστε το **Arrival Time** ενός σήματος στο διάγραμμα χρονισμού της χρονικής προσομοίωσης μετά την υλοποίηση με τη χρήση των marker. Για παράδειγμα, βάλτε τον μπλε marker στην ανερχόμενη ακμή του **CLK** στα 155.170 ns και τον κίτρινο marker στην ανερχόμενη ακμή του εσωτερικού σήματος **[0]** (που αντιστοιχεί στο Cout) στα 164.026 ns. Απέχουν 8.856 ns. Συγκρίνετε με το **Arrival Time** που είχατε βρει κατά τη χρονική ανάλυση για την κρίσιμη διαδρομή, που ήταν 9.661 ns.

Προσοχή! Πρέπει να βάλουμε τον μπλε marker **έναν κύκλο πριν**, επειδή η περίοδος του **CLK** είναι **μικρότερη** του **Arrival Time**.



Με κλικ πάνω στον μπλε marker (γίνεται λευκός), ορίζουμε τη θέση του στα 0.000 ns και είναι πλέον εμφανές ότι ο κίτρινος marker απέχει 8.856 ns.



5-3.9. Κλείστε τον simulator επιλέγοντας το κουμπί ✕ πάνω δεξιά στο παράθυρο του *SIMULATION*. Στο παράθυρο *Confirm Close* που εμφανίζεται πατήστε **OK**. Εάν επιθυμείτε να μη σώσετε ένα επιπλέον waveform configuration, στο παράθυρο *Save Waveform Configuration* επιλέξτε **Discard**.

Η διαδικασία που ήδη περιγράψαμε στα Βήματα 5-1, 5-2 και 5-3 της «**Υλοποίησης στη τεχνολογία FPGA και προσομοίωση (λογική, χρονική)**» εφαρμόζεται παρομοίως σε κάθε πιθανή υπομονάδα συνδυαστικής λογικής της διαδρομής δεδομένων του επεξεργαστή.

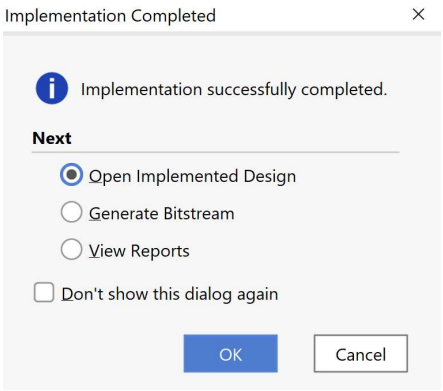
Προσοχή! Στην περίπτωση του επεξεργαστή πολλών κύκλων υπάρχει το ενδεχόμενο κάποια διαδρομή μέσω της μονάδας ελέγχου να χαρακτηρίζεται εσφαλμένως ως κρίσιμη διαδρομή. Στην περίπτωση αυτή με δεξί κλικ πάνω στο **path** επιλέγουμε αρχικά **Set Multicycle Path** και στη συνέχεια **Source Clock to Destination Clock**, προσδιορίζοντας και τον αριθμό **N** των κύκλων ρολογιού. Περισσότερες πληροφορίες στο *Vivado Design Suite User Guide: Using Constraints* (UG903), Multicycle Paths.

5-4. Εκτέλεση της διαδικασίας της υλοποίησης και ανάλυση των αποτελεσμάτων μετά την υλοποίηση στο VIVADO IDE για μηχανή πεπερασμένων καταστάσεων (FSM)

Η διαδικασία της υλοποίησης εφαρμόζεται στο **synthesized design model** της οντότητας που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design resources και μετά από την ολοκλήρωση της υλοποίησης παράγεται το αντίστοιχο **implemented design model**.

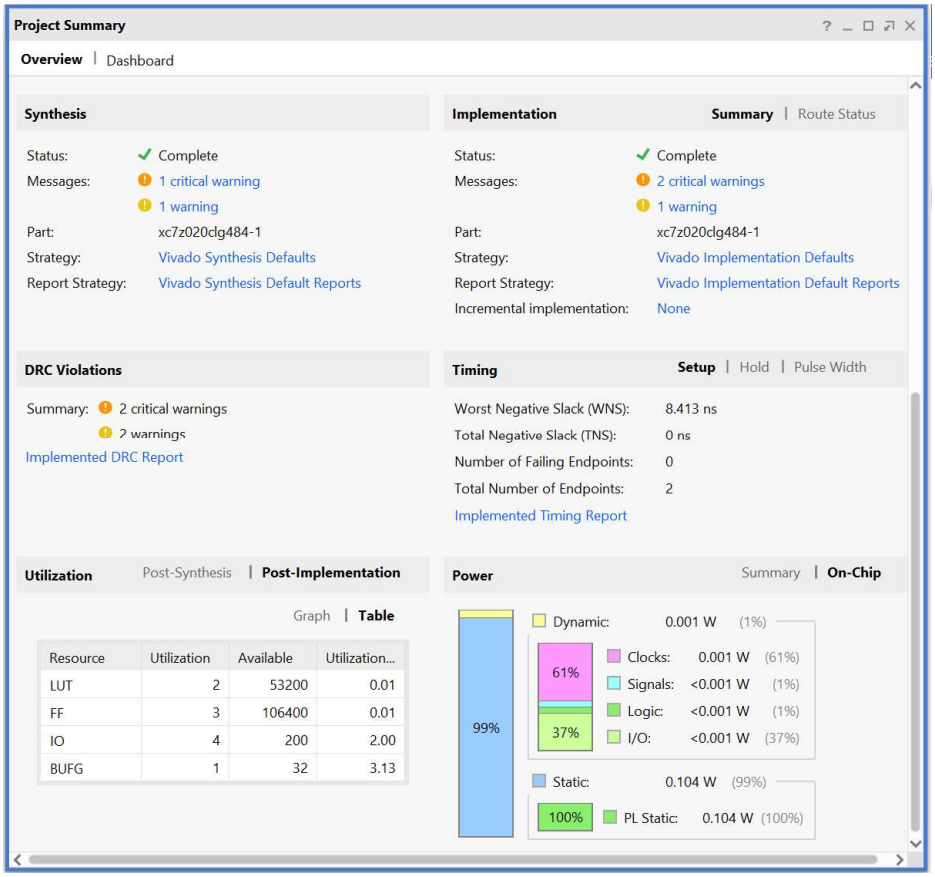
Η υλοποίηση στην τεχνολογία **Zynq-7000 (device xc7z020clg484-1)** εκτελείται στο **synthesized design model** της οντότητας **PATTERN_FSM** που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των design resources. Με τη υλοποίηση το εργαλείο Vivado IDE παράγει το **implemented design model** της οντότητας **PATTERN_FSM**.

5-4.1. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Implementation**. Πατήστε **OK** στα παράθυρα προειδοποίησης που εμφανίζονται.

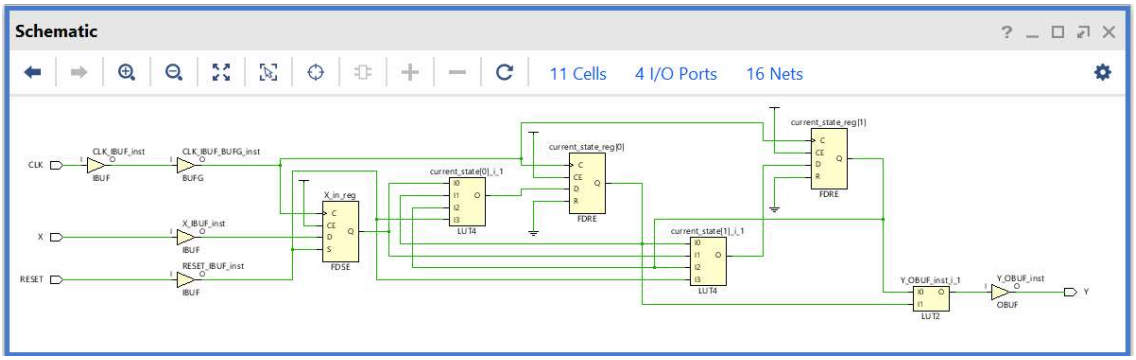


5-4.2. Στο παράθυρο *Implementation Completed* υπάρχουν τρεις επιλογές. Επιλέξτε το **Open Implemented Design** και πατήστε **OK** για να μελετήσετε το αποτέλεσμα της υλοποίησης (implementation).

5-4.3. Αρχικά, επιλέξτε το παράθυρο *Project Summary* και μελετήστε τα διάφορα υπο-παράθυρα.



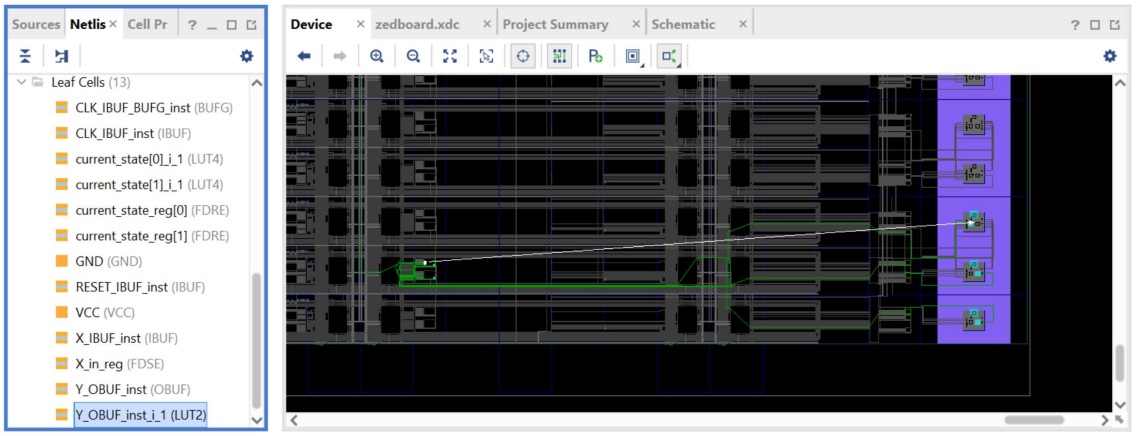
5-4.4. Στη συνέχεια, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Implemented Design* επιλέξετε το **Schematic** για να δείτε το σχηματικό διάγραμμα του **implemented design model**.



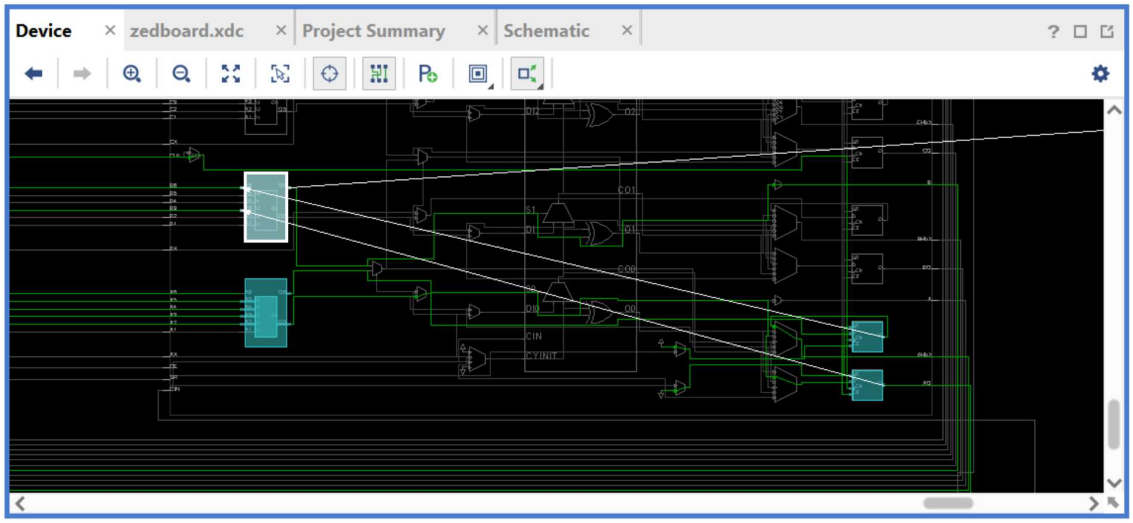
Παρατηρείστε ότι έχουν αυτόματα προστεθεί τα απαραίτητα IBUFs, OBUFs και BUFG primitives στο σχηματικό διάγραμμα, καθώς και ότι οι είσοδοι και οι έξοδοι από το FPGA είναι buffered. Παραμένει ίδιο με το σχηματικό διάγραμμα μετά τη σύνθεση.

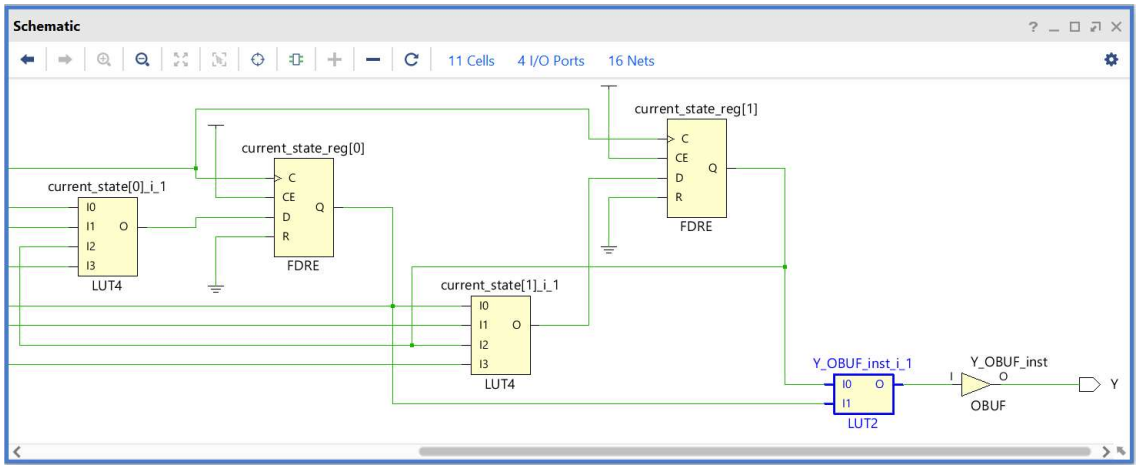
5-4.5. Στη συνέχεια μελετήστε το παράθυρο *Device* έχοντας πατήσει τα κουμπιά *auto-fit selection*, *routing resources* και *show cell connections*. 

Στη μελέτη σας επιλέξτε κάποιο *leaf cell*, όπως για παράδειγμα το LUT2 που παράγει το Y με κατάλληλη επιλογή στο παράθυρο *Netlist*.

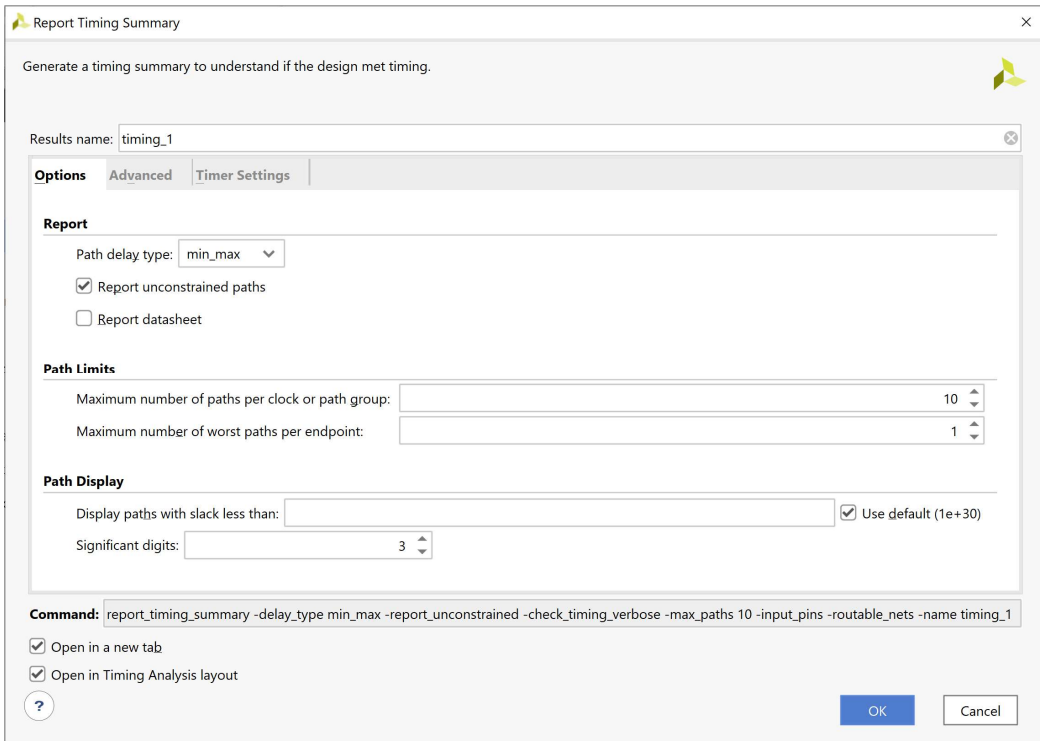


5-4.6. Επικεντρώστε στη δημιουργία του Y με παράλληλη μελέτη των παραθύρων *Device* και *Schematic*, εναλλάξ. Απαιτείται εντοπισμένη μεγέθυνση. Στο SLICE_X113Y1 φαίνονται: το **LUT2** (Y_OBUF_inst_i_1), το **LUT4** (current_state_[0]_i_1), το **FDRE** (current_state_reg[1]) και το **FDRE** (current_state_reg[0]).



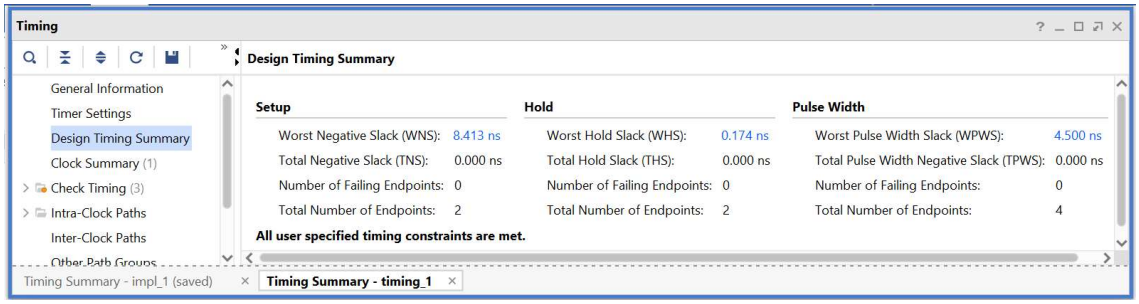


5-4.7. Στη συνέχεια, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Implemented Design* επιλέξετε το **Report Timing Summary** για να δείτε την ανάλυση χρονισμού που κάνει το εργαλείο Vivado IDE στο **implemented design model**. Αυτή είναι η πιο ακριβής ανάλυση χρονισμού που έχουμε διαθέσιμη.

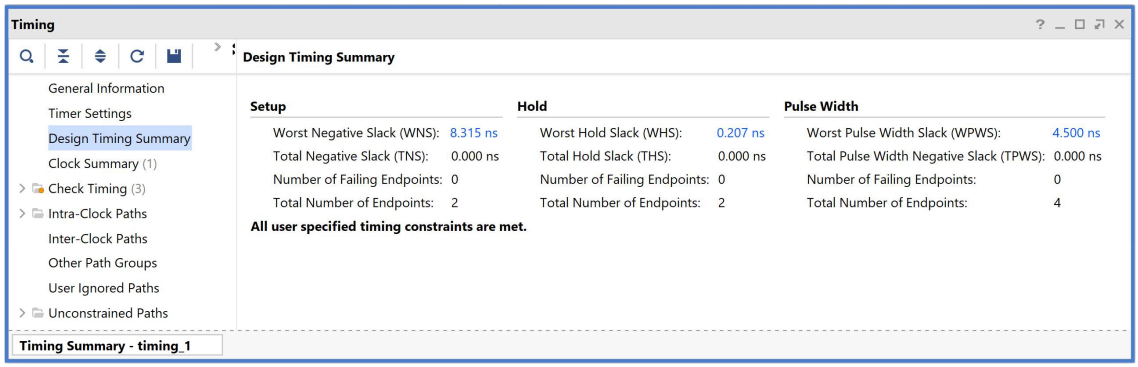


Στην επιλογή *Path delay type* ορίζεται ο τύπος της ανάλυσης που θα εκτελεσθεί. Το *max delay analysis* αφορά στην εύρεση της κρίσιμης διαδρομής (με τη μεγαλύτερη καθυστέρηση διάδοσης) και συνεπώς στην εύρεση της μέγιστης συχνότητας λειτουργίας χωρίς την παραβίαση του **χρόνου σταθεροποίησης** (setup time). Το *min delay analysis* αφορά στην εύρεση της σύντομης διαδρομής (με τη μικρότερη καθυστέρηση διάδοσης) χωρίς την παραβίαση του **χρόνου διατήρησης** (hold time).

5-4.8. Πατήστε **OK** για να παραχθεί το Timing_1 report.



Συγκρίνετε με το Timing_1 report που είχε παραχθεί μετά τη σύνθεση.



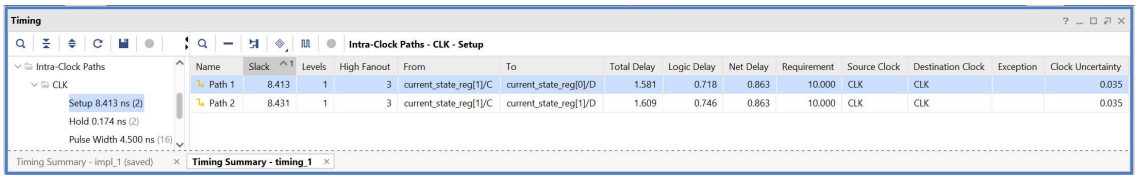
Στη στήλη **Setup** παρουσιάζονται τα αποτελέσματα του *max delay analysis*.

- Το *Worst Negative Slack* (**WNS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των κρίσιμων διαδρομών του max delay analysis. Μπορεί να είναι θετικό η αρνητικό. Ένα θετικό slack (8.413 ns) δηλώνει ότι η κρίσιμη διαδρομή ικανοποιεί την περίοδο του CLK. Ένα αρνητικό slack δηλώνει ότι παραβιάζεται ο χρόνος σταθεροποίησης (setup time).
- Το *Total Negative Slack* (**TNS**) είναι το άθροισμα όλων των αρνητικών WNS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου σταθεροποίησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά στην επιλεγμένη συχνότητα λειτουργίας.

Στη στήλη **Hold** παρουσιάζονται τα αποτελέσματα του min delay analysis.

- Το *Worst Hold Slack* (**WHS**) είναι μια τιμή (με link) που αντιστοιχεί στο μικρότερο διαθέσιμο περιθώριο (slack) που προκύπτει από την ανάλυση όλων των σύντομων διαδρομών του min delay analysis. Μπορεί να είναι θετικό η αρνητικό. Ένα αρνητικό slack δηλώνει ότι παραβιάζεται ο χρόνος διατήρησης (hold time).
- Το *Total Hold Slack* (**THS**) είναι το άθροισμα όλων των αρνητικών WHS για κάθε timing path endpoint. Η τιμή είναι 0.000 ns όταν δεν υπάρχουν παραβιάσεις του χρόνου διατήρησης. Το ψηφιακό κύκλωμα λειτουργεί κανονικά.

5-4.9. Επιλέξτε το **WNS link** και δείτε τις 2 διαθέσιμες κρίσιμες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο σταθεροποίησης. Η κρίσιμη διαδρομή έχει καθυστέρηση διάδοσης 1.581 ns, εκ των οποίων τα 0.718 ns αφορούν στη λογική (logic), ενώ τα 0.863 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.035 ns. Τα επίπεδα λογικής (logic level) είναι 1.

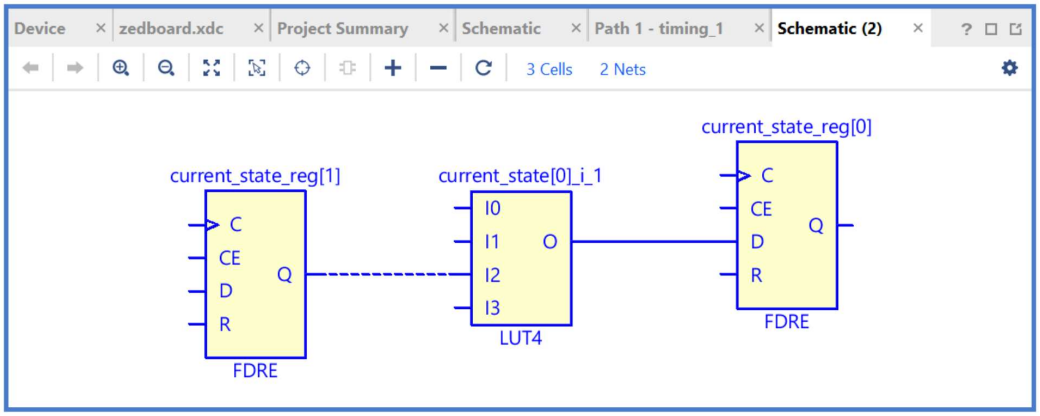


5-4.10. Επιλέξτε με διπλό κλικ το **Path 1**, ώστε να εμφανιστεί το παράθυρο *Path 1 – timing_1*. Η κρίσιμη διαδρομή περνάει από 1 LUT4. Υπολογίστε το WNS slack (σε παρένθεση το αποτέλεσμα της σύνθεσης):

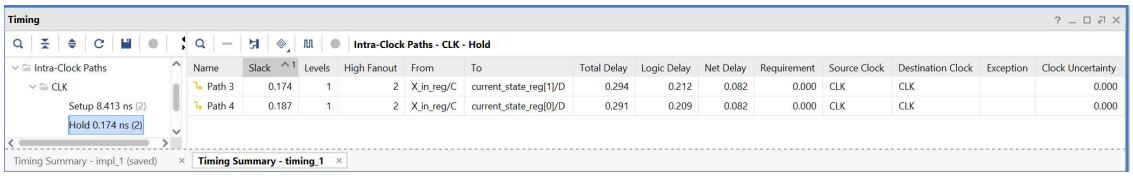
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή κατάστασης 1) είναι **5.642** (2.975) ns,
- η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή κατάστασης 1 και μέσω του LUT4 μέχρι την είσοδο D του καταχωρητή κατάστασης 0) είναι **1.581** (1.549) ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι **7.223** (4.524) ns,
- το **Required Time**, ως η καθυστέρηση διάδοσης της κρίσιμης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή 10.000 ns μέχρι την είσοδο CLK του καταχωρητή κατάστασης 0) συν τον χρόνο σταθεροποίησης είναι **15.636** (12.839) ns.

WNS slack = Required Time – Arrival Time = 15.636 – 7.223 = 8.413 ns

5-4.11. Επιλέξτε με δεξί κλικ στο **Path 1**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της κρίσιμης διαδρομής της οντότητας **PATTERN_FSM**. (Δεν έχει αλλάξει η διαδρομή).



5-4.12. Επιλέξτε το **WHS link** και δείτε τις 2 σύντομες διαδρομές (δηλαδή με το μικρότερο θετικό slack) που δεν παραβιάζουν το χρόνο διατήρησης. Η σύντομη διαδρομή έχει καθυστέρηση μόλυνσης 0.294 ns, εκ των οποίων τα 0.212 ns αφορούν στη λογική (logic), ενώ τα 0.082 ns αφορούν στη δικτύωση (net). Η αβεβαιότητα του CLK εκτιμάται στα 0.000 ns. Τα επίπεδα λογικής (logic level) είναι 1.

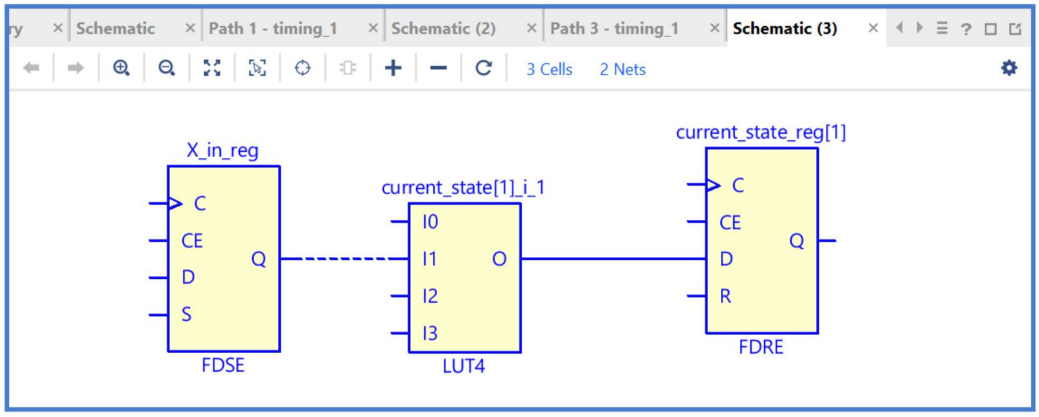


5-4.13. Επιλέξτε με διπλό κλικ το **Path 3**, ώστε να εμφανιστεί το παράθυρο *Path 3 – timing_1*. Η σύντομη διαδρομή περνάει από 1 μονάδα LUT4. Υπολογίστε το WHS slack (σε παρένθεση το αποτέλεσμα της σύνθεσης):

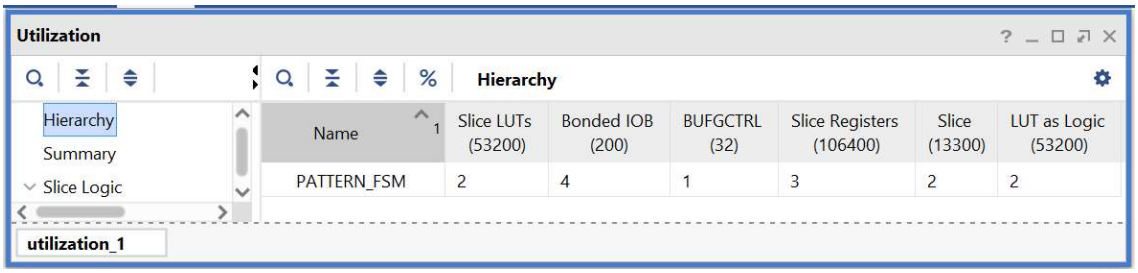
- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Source Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK του καταχωρητή εισόδου) είναι **1.588** (0.735) ns,
- η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Data Path* (από την είσοδο CLK μέχρι την έξοδο Q του καταχωρητή εισόδου και μέσω του LUT4 μέχρι την είσοδο D του καταχωρητή κατάστασης 1) είναι **0.294** (0.451) ns,
- το **Arrival Time**, ως άθροισμα των ανωτέρων χρόνων, είναι **1.882** (1.186) ns,
- το **Required Time**, ως η καθυστέρηση μόλυνσης της σύντομης διαδρομής του *Destination Clock Path* (από την πηγή του CLK τη χρονική στιγμή 0.000 ns μέχρι την είσοδο CLK στον καταχωρητή κατάστασης 1) συν τον χρόνο διατήρησης, είναι **1.708** (0.979) ns.

WHS slack = Arrival Time – Required Time = 1.882 – 1.708 = 0.174 ns

5-4.14. Επιλέξτε με δεξί κλικ στο **Path 3**, το **Schematic**. Μελετήστε το σχηματικό διάγραμμα της σύντομης διαδρομής της οντότητας **PATTERN_FSM**. (Έχει αλλάξει η διαδρομή).



5-4.15. Τέλος, στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, μέσα στο *Open Implemented Design* επιλέξετε το **Report Utilization** και μελετήστε τους πόρους που χρησιμοποιεί η οντότητα **PATTERN_FSM**. Πατήστε **OK**. (Μπορείτε να μελετήσετε και άλλα ενδιαφέροντα reports, εάν επιλέξετε το παράθυρο *Reports*).



5-5. Εκτέλεση προσομοίωσης μετά την υλοποίηση (λογική και χρονική) στο VIVADO IDE για μηχανή πεπερασμένων καταστάσεων (FSM)

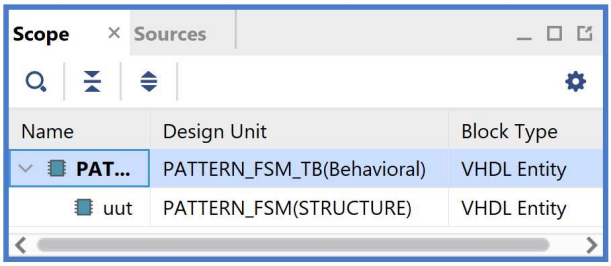
Η προσομοίωση μετά την υλοποίηση (λογική και χρονική) εκτελείται στην οντότητα **PATTERN_FSM_TB** του προγράμματος δοκιμών (testbench) που έχει ορισθεί ως η κορυφαία οντότητα της ιεραρχίας (**top**) των simulation resources. Κατά την προσομοίωση, η οντότητα **PATTERN_FSM_TB** καλεί το *UUT* της (που είναι το **implemented design model** της οντότητας **PATTERN_FSM**).

5-5.1. Ενεργοποιήστε το waveform configuration file *PATTERN_ FSM_TB_func_synth.wcfg* στην περίπτωση που είναι απενεργοποιημένο. Στο παράθυρο *Sources* κάντε δεξί κλικ και επιλέξτε **Enable File**. Απενεργοποιήστε όλα τα υπόλοιπα waveform configuration file. Στο παράθυρο *Sources* κάντε δεξί κλικ και επιλέξτε **Disable File**.

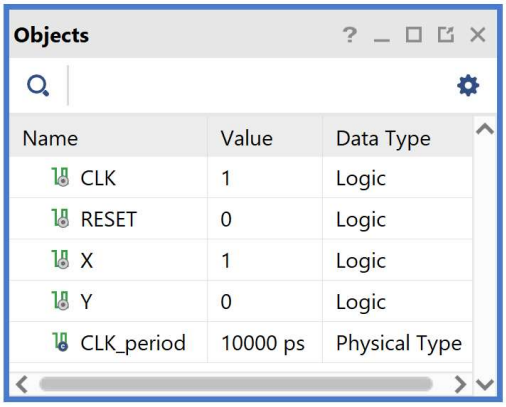
5-5.2. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Simulation**, ώστε να εμφανιστούν όλες οι πιθανές προσομοιώσεις που υποστηρίζει το Vivado IDE. Επιλέξτε **Run Post-Implementation Functional Simulation**.

5-5.3. Το πρόγραμμα δοκιμής (testbench) και όλες οι οντότητες του *UUT* θα γίνουν compiled και θα τρέξει το Vivado simulator (εφόσον βέβαια δεν υπάρχουν σφάλματα). Θα εμφανιστεί το παράθυρο *SIMULATION* που απαρτίζεται από 4 παράθυρα:

Το παράθυρο *Scope*, όπου παρουσιάζεται η οντότητα **PATTERN_FSM_TB** και το **implemented design model** της οντότητας **PATTERN_FSM** (*UUT*) μετά την υλοποίηση, η οποία είναι *structural*.

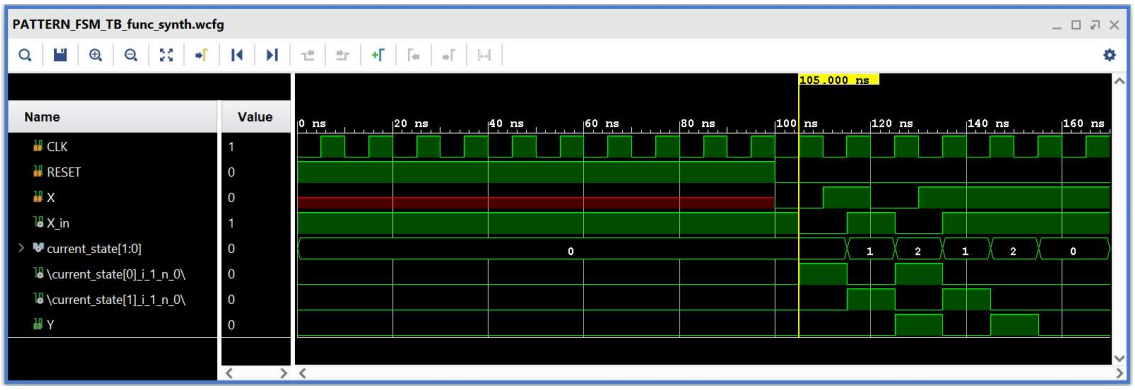


Το παράθυρο *Objects*, όπου εμφανίζονται τα σήματα top-level, δηλαδή οι είσοδοι και οι έξοδοι της οντότητας **PATTERN_FSM**, που είναι η κορυφαία οντότητα της ιεραρχίας του *UUT*, καθώς και η περίοδος του CLK (*CLK_period*). Οι τιμές αντιστοιχούν στις τιμές που έχει σταματήσει η προσομοίωση της οντότητας **PATTERN_FSM_TB** (st*op (2)*).



Το παράθυρο *Tcl Console* με όλες τις διαδικασίες που εκτελούνται στο πλαίσιο της προσομοίωσης. Το Tcl Console καθαρίζει με την επιλογή *Clear*.

Επιλέξτε το παράθυρο *PATTERN_FSM_TB_func_synth.wcfg*, που δημιουργήσατε για την προσομοίωση του **synthesized design model**, όπου συμπεριλαμβάνονται πέραν των top-level εισόδων/εξόδων και τα απαραίτητα **εσωτερικά σήματα current_state[0]_i_1_n_0** (αντιστοιχεί στο next_state[0]) και **current_ state[1]_i_1_n_0** (αντιστοιχεί στο next_state[1]). Βλέπετε ολόκληρο το διάγραμμα χρονισμού της λογικής προσομοίωσης μετά τη σύνθεση με κατάλληλο **zoom out** και επιλέγοντας το **zoom fit**. Τα διαγράμματα χρονισμού της λογικής προσομοίωσης μετά τη σύνθεση και μετά την υλοποίηση είναι ίδια.



5-5.4. Κλείστε τον simulator επιλέγοντας το κουμπί × πάνω δεξιά στο παράθυρο του *SIMULATION*. Στο παράθυρο *Confirm Close* που εμφανίζεται πατήστε **OK**.

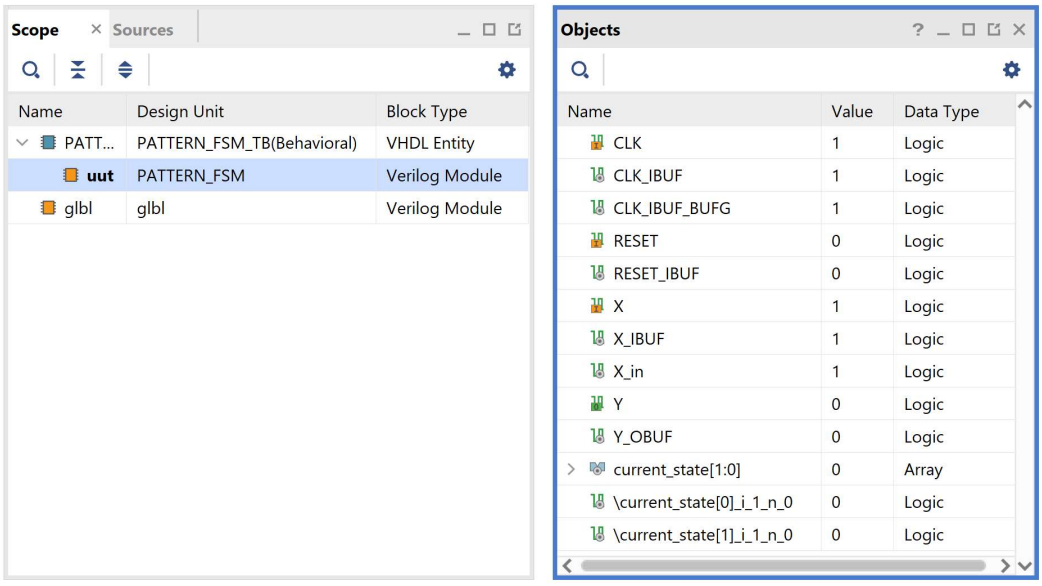
5-5.5. Στο κατακόρυφο παράθυρο αριστερά του *Flow Navigator*, επιλέξτε το **Run Simulation**, ώστε να εμφανιστούν όλες οι πιθανές προσομοιώσεις που υποστηρίζει το Vivado IDE. Επιλέξτε **Run Post-Implementation Timing Simulation**.

5-5.6. Το πρόγραμμα δοκιμής (testbench) και όλες οι οντότητες του *UUT* θα γίνουν compiled και θα τρέξει το Vivado simulator (εφόσον βέβαια δεν υπάρχουν σφάλματα). Θα εμφανιστεί το παράθυρο *SIMULATION* που απαρτίζεται από 4 παράθυρα:

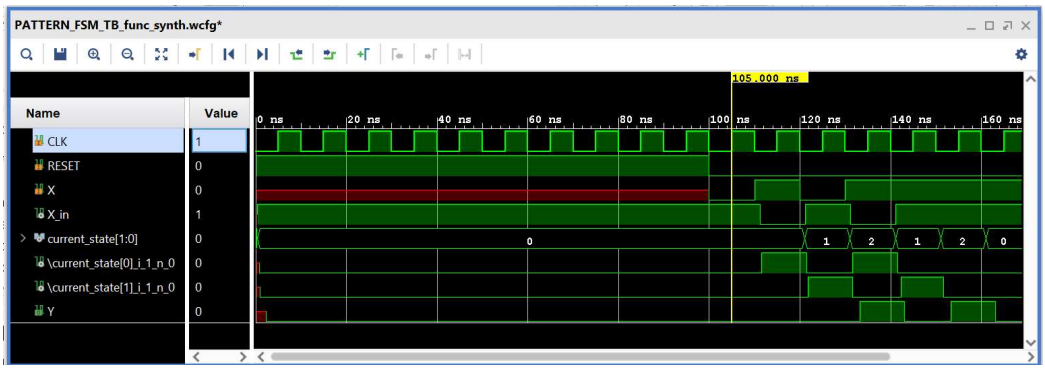
Το παράθυρο Scope, όπου παρουσιάζεται η οντότητα **PATTERN_ FSM_TB** και το **implemented design model** της οντότητας **PATTERN_ FSM** (*UUT*) που προκύπτει μετά την υλοποίηση για χρονική προσομοίωση. Όλες οι οντότητες που απαρτίζουν πλέον το *UUT* είναι *Verilog Module!*

Το παράθυρο *Objects*, όπου εμφανίζονται τα σήματα της οντότητας **PATTERN_ FSM** (*UUT*). Οι αρτηρίες (array) αναλύονται στα σήματα που τις απαρτίζουν. Οι τιμές αντιστοιχούν στις τιμές που έχει σταματήσει η προσομοίωση της οντότητας **PATTERN_ FSM_TB** (stop (2)).

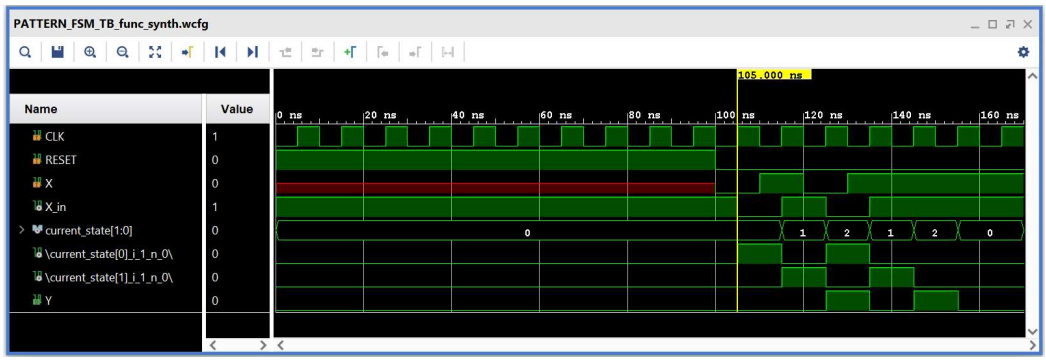
Το παράθυρο *Tcl Console* με όλες τις διαδικασίες που εκτελούνται στο πλαίσιο της προσομοίωσης. Το Tcl Console καθαρίζει με την επιλογή *Clear*.



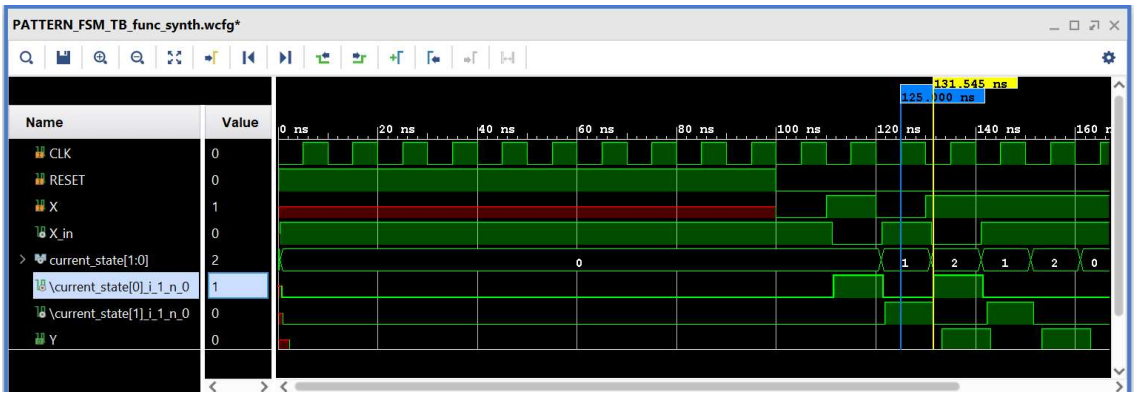
Επιλέξτε το παράθυρο *PATTERN_ FSM_TB_func_synth.wcfg*. Βλέπετε ολόκληρο το διάγραμμα χρονισμού της χρονικής προσομοίωσης μετά τη σύνθεση με κατάλληλο **zoom out** ή επιλέγοντας το **zoom fit**. (Εάν λείπουν τα εσωτερικά σήματα, τα μεταφέρετε από το παράθυρο *Objects* και επαναλάβετε τη διαδικασία της προσομοίωσης από την αρχή με επιλογή του κουμπιού **Restart** και στη συνέχεια του κουμπιού **Run All**.)



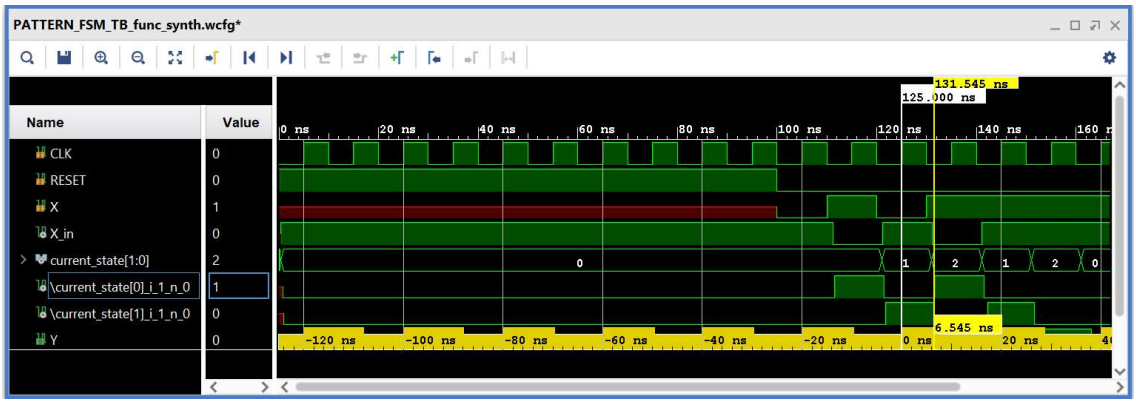
5-5.7. Συγκρίνετε τα διαγράμματα χρονισμού της χρονικής και της λογικής προσομοίωσης μετά την υλοποίηση. Είναι εμφανείς οι καθυστερήσεις διάδοσης στο πρώτο διάγραμμα χρονισμού.



5-5.8. Υπολογίστε το **Arrival Time** ενός σήματος στο διάγραμμα χρονισμού της χρονικής προσομοίωσης μετά την υλοποίηση με τη χρήση των marker. Για παράδειγμα, βάλτε τον μπλε marker στην ανερχόμενη ακμή του **CLK** στα 125.000 ns και τον κίτρινο marker στην ανερχόμενη ακμή του εσωτερικού σήματος **current_state[0]_i_1_n_0** (που αντιστοιχεί στο next_state[0]) στα 131.545 ns. Απέχουν 6.545 ns. Συγκρίνετε με το **Arrival Time** που είχατε βρει κατά τη χρονική ανάλυση για την κρίσιμη διαδρομή, που ήταν 7.223 ns.



Με κλικ πάνω στον μπλε marker (γίνεται λευκός), ορίζουμε τη θέση του στα 0.000 ns και είναι πλέον εμφανές ότι ο κίτρινος marker απέχει 6.545 ns.

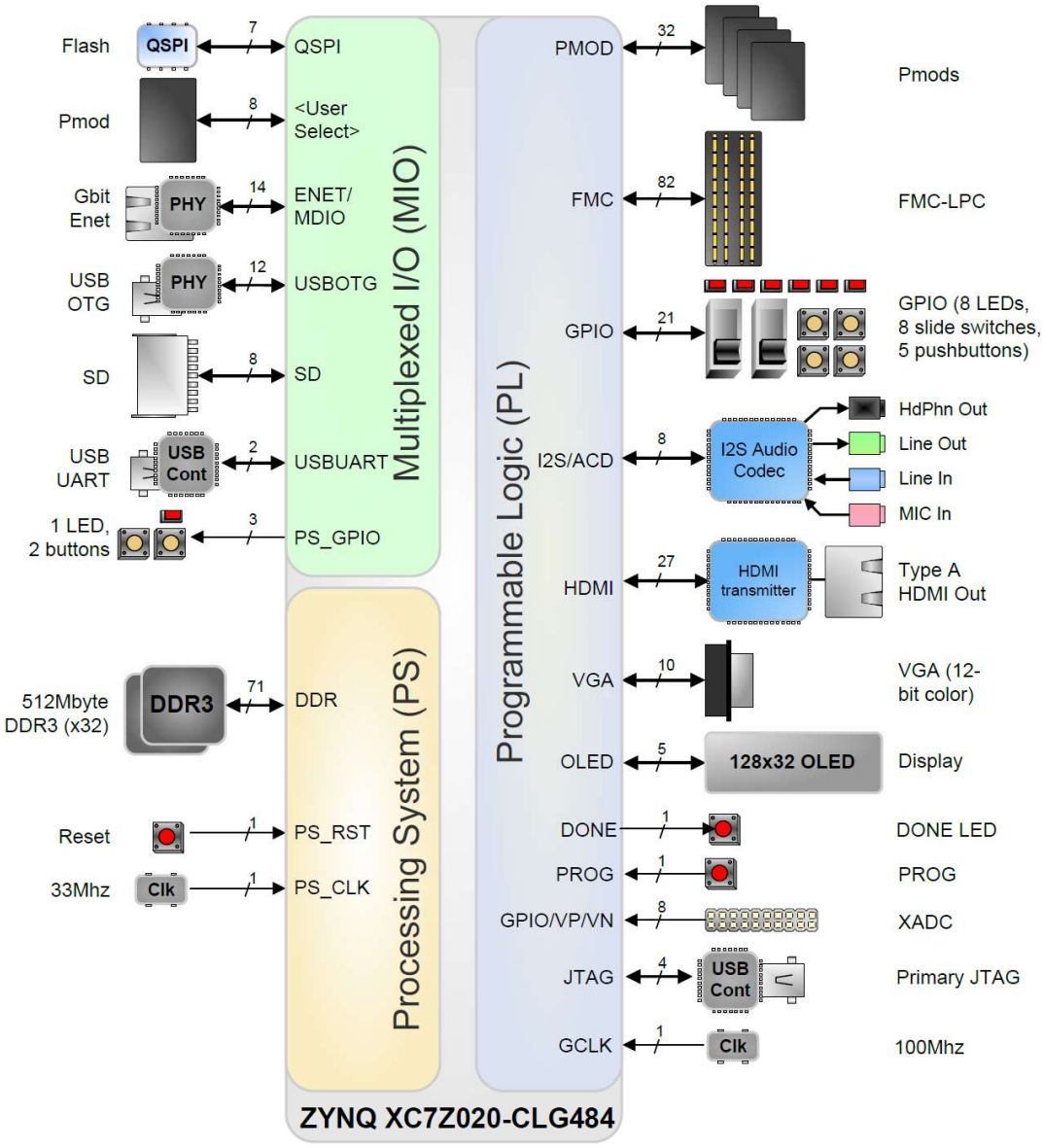


5-5.9. Κλείστε τον simulator επιλέγοντας το κουμπί ✕ πάνω δεξιά στο παράθυρο του *SIMULATION*. Στο παράθυρο *Confirm Close* που εμφανίζεται πατήστε **OK**. Εάν επιθυμείτε να μη σώσετε ένα επιπλέον waveform configuration, στο παράθυρο *Save Waveform Configuration* επιλέξτε **Discard**.

Η διαδικασία που ήδη περιγράψαμε στα Βήματα 5-4 και 5-5 της «**Υλοποίησης στη τεχνολογία FPGA και προσομοίωση (λογική, χρονική)**» εφαρμόζεται παρομοίως σε κάθε πιθανή μηχανή πεπερασμένων καταστάσεων (FSM).

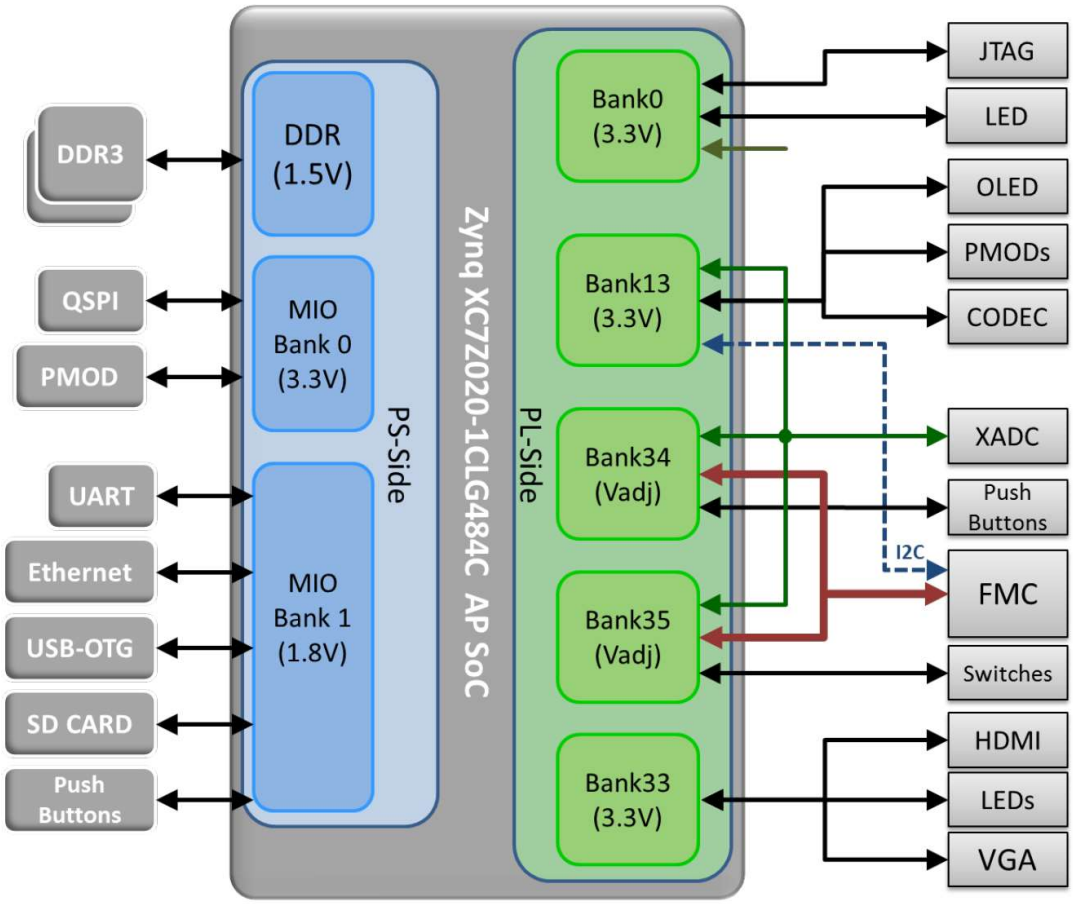
ΠΑΡΑΡΤΗΜΑ Α

Μπλοκ διάγραμμα της αναπτυξιακής κάρτας Zedboard



ΠΑΡΑΡΤΗΜΑ Β

Bank Assignments της διάταξης Zynq Z7020 της αναπτυξιακής κάρτας Zedboard



ΠΑΡΑΡΤΗΜΑ Γ

Πρότυπο zedboard.xdc από την AVNET για την κάρτα Zedboard

```
# ----------------------------------------------------------------------------
#     _____
#    /     \
#   /____   \____
#  / \===\    \==/
# /___\===\___\/  AVNET Design Resource Center
#      \======/         www.em.avnet.com/drc
#       \====/
# ----------------------------------------------------------------------------
#
#  Created With Avnet UCF Generator V0.4.0
#     Date: Saturday, June 30, 2012
#     Time: 12:18:55 AM
#
#  This design is the property of Avnet.  Publication of this
#  design is not authorized without written consent from Avnet.
#
#  Please direct any questions to:
#     ZedBoard.org Community Forums
#     http://www.zedboard.org
#
#  Disclaimer:
#     Avnet, Inc. makes no warranty for the use of this code or design.
#     This code is provided "As Is". Avnet, Inc assumes no responsibility for
#     any errors, which may appear in this code, nor does it make a commitment
#     to update the information contained herein. Avnet, Inc specifically
#     disclaims any implied warranties of fitness for a particular purpose.
#                      Copyright(c) 2012 Avnet, Inc.
#                              All rights reserved.
#
# ----------------------------------------------------------------------------
#
#  Notes:
#
#  10 August 2012
#     IO standards based upon Bank 34 and Bank 35 Vcco supply options of 1.8V,
#     2.5V, or 3.3V are possible based upon the Vadj jumper (J18) settings.
#     By default, Vadj is expected to be set to 1.8V but if a different
#     voltage is used for a particular design, then the corresponding IO
#     standard within this UCF should also be updated to reflect the actual
#     Vadj jumper selection.
#
#  09 September 2012
#     Net names are not allowed to contain hyphen characters '-' since this
#     is not a legal VHDL87 or Verilog character within an identifier.
#     HDL net names are adjusted to contain no hyphen characters '-' but
#     rather use underscore '_' characters.  Comment net name with the hyphen
#     characters will remain in place since these are intended to match the
#     schematic net names in order to better enable schematic search.
#
#  17 April 2014
#     Pin constraint for toggle switch SW7 was corrected to M15 location.
#
#  16 April 2015
#     Corrected the way that entire banks are assigned to a particular IO
#     standard so that it works with more recent versions of Vivado Design
#     Suite and moved the IO standard constraints to the end of the file
#     along with some better organization and notes like we do with our SOMs.
#
#   6 June 2016
#     Corrected error in signal name for package pin N19 (FMC Expansion Connector)
#
#
# ----------------------------------------------------------------------------
```

```
# ------------------------------------------------------------------------
# Audio Codec - Bank 13
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN AB1 [get_ports {AC_ADR0}];  # "AC-ADR0"
#set_property PACKAGE_PIN Y5  [get_ports {AC_ADR1}];  # "AC-ADR1"
#set_property PACKAGE_PIN Y8  [get_ports {SDATA_O}];  # "AC-GPIO0"
#set_property PACKAGE_PIN AA7 [get_ports {SDATA_I}];  # "AC-GPIO1"
#set_property PACKAGE_PIN AA6 [get_ports {BCLK_O}];  # "AC-GPIO2"
#set_property PACKAGE_PIN Y6  [get_ports {LRCLK_O}];  # "AC-GPIO3"
#set_property PACKAGE_PIN AB2 [get_ports {MCLK_O}];  # "AC-MCLK"
#set_property PACKAGE_PIN AB4 [get_ports {iic_rtl_scl_io}];  # "AC-SCK"
#set_property PACKAGE_PIN AB5 [get_ports {iic_rtl_sda_io}];  # "AC-SDA"

# ------------------------------------------------------------------------
# Clock Source - Bank 13
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN Y9 [get_ports {GCLK}];  # "GCLK"

# ------------------------------------------------------------------------
# JA Pmod - Bank 13
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN Y11  [get_ports {JA1}];  # "JA1"
#set_property PACKAGE_PIN AA8  [get_ports {JA10}];  # "JA10"
#set_property PACKAGE_PIN AA11 [get_ports {JA2}];  # "JA2"
#set_property PACKAGE_PIN Y10  [get_ports {JA3}];  # "JA3"
#set_property PACKAGE_PIN AA9  [get_ports {JA4}];  # "JA4"
#set_property PACKAGE_PIN AB11 [get_ports {JA7}];  # "JA7"
#set_property PACKAGE_PIN AB10 [get_ports {JA8}];  # "JA8"
#set_property PACKAGE_PIN AB9  [get_ports {JA9}];  # "JA9"

# ------------------------------------------------------------------------
# JB Pmod - Bank 13
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN W12 [get_ports {JB1}];  # "JB1"
#set_property PACKAGE_PIN W11 [get_ports {JB2}];  # "JB2"
#set_property PACKAGE_PIN V10 [get_ports {JB3}];  # "JB3"
#set_property PACKAGE_PIN W8 [get_ports {JB4}];  # "JB4"
#set_property PACKAGE_PIN V12 [get_ports {JB7}];  # "JB7"
#set_property PACKAGE_PIN W10 [get_ports {JB8}];  # "JB8"
#set_property PACKAGE_PIN V9 [get_ports {JB9}];  # "JB9"
#set_property PACKAGE_PIN V8 [get_ports {JB10}];  # "JB10"

# ------------------------------------------------------------------------
# JC Pmod - Bank 13
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN AB6 [get_ports {JC1_N}];  # "JC1_N"
#set_property PACKAGE_PIN AB7 [get_ports {JC1_P}];  # "JC1_P"
#set_property PACKAGE_PIN AA4 [get_ports {JC2_N}];  # "JC2_N"
#set_property PACKAGE_PIN Y4  [get_ports {JC2_P}];  # "JC2_P"
#set_property PACKAGE_PIN T6  [get_ports {JC3_N}];  # "JC3_N"
#set_property PACKAGE_PIN R6  [get_ports {JC3_P}];  # "JC3_P"
#set_property PACKAGE_PIN U4  [get_ports {JC4_N}];  # "JC4_N"
#set_property PACKAGE_PIN T4  [get_ports {JC4_P}];  # "JC4_P"

# ------------------------------------------------------------------------
# JD Pmod - Bank 13
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN W7 [get_ports {JD1_N}];  # "JD1_N"
#set_property PACKAGE_PIN V7 [get_ports {JD1_P}];  # "JD1_P"
#set_property PACKAGE_PIN V4 [get_ports {JD2_N}];  # "JD2_N"
#set_property PACKAGE_PIN V5 [get_ports {JD2_P}];  # "JD2_P"
#set_property PACKAGE_PIN W5 [get_ports {JD3_N}];  # "JD3_N"
#set_property PACKAGE_PIN W6 [get_ports {JD3_P}];  # "JD3_P"
#set_property PACKAGE_PIN U5 [get_ports {JD4_N}];  # "JD4_N"
#set_property PACKAGE_PIN U6 [get_ports {JD4_P}];  # "JD4_P"
```

```
# ------------------------------------------------------------------------
# OLED Display - Bank 13
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN U10  [get_ports {OLED_DC}];  # "OLED-DC"
#set_property PACKAGE_PIN U9   [get_ports {OLED_RES}];  # "OLED-RES"
#set_property PACKAGE_PIN AB12 [get_ports {OLED_SCLK}];  # "OLED-SCLK"
#set_property PACKAGE_PIN AA12 [get_ports {OLED_SDIN}];  # "OLED-SDIN"
#set_property PACKAGE_PIN U11  [get_ports {OLED_VBAT}];  # "OLED-VBAT"
#set_property PACKAGE_PIN U12  [get_ports {OLED_VDD}];  # "OLED-VDD"

# ------------------------------------------------------------------------
# HDMI Output - Bank 33
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN W18  [get_ports {HD_CLK}];  # "HD-CLK"
#set_property PACKAGE_PIN Y13  [get_ports {HD_D0}];  # "HD-D0"
#set_property PACKAGE_PIN AA13 [get_ports {HD_D1}];  # "HD-D1"
#set_property PACKAGE_PIN W13  [get_ports {HD_D10}];  # "HD-D10"
#set_property PACKAGE_PIN W15  [get_ports {HD_D11}];  # "HD-D11"
#set_property PACKAGE_PIN V15  [get_ports {HD_D12}];  # "HD-D12"
#set_property PACKAGE_PIN U17  [get_ports {HD_D13}];  # "HD-D13"
#set_property PACKAGE_PIN V14  [get_ports {HD_D14}];  # "HD-D14"
#set_property PACKAGE_PIN V13  [get_ports {HS_D15}];  # "HD-D15"
#set_property PACKAGE_PIN AA14 [get_ports {HD_D2}];  # "HD-D2"
#set_property PACKAGE_PIN Y14  [get_ports {HD_D3}];  # "HD-D3"
#set_property PACKAGE_PIN AB15 [get_ports {HD_D4}];  # "HD-D4"
#set_property PACKAGE_PIN AB16 [get_ports {HD_D5}];  # "HD-D5"
#set_property PACKAGE_PIN AA16 [get_ports {HD_D6}];  # "HD-D6"
#set_property PACKAGE_PIN AB17 [get_ports {HD_D7}];  # "HD-D7"
#set_property PACKAGE_PIN AA17 [get_ports {HD_D8}];  # "HD-D8"
#set_property PACKAGE_PIN Y15  [get_ports {HD_D9}];  # "HD-D9"
#set_property PACKAGE_PIN U16  [get_ports {HD_DE}];  # "HD-DE"
#set_property PACKAGE_PIN V17  [get_ports {HD_HSYNC}];  # "HD-HSYNC"
#set_property PACKAGE_PIN W16  [get_ports {HD_INT}];  # "HD-INT"
#set_property PACKAGE_PIN AA18 [get_ports {HD_SCL}];  # "HD-SCL"
#set_property PACKAGE_PIN Y16  [get_ports {HD_SDA}];  # "HD-SDA"
#set_property PACKAGE_PIN U15  [get_ports {HD_SPDIF}];  # "HD-SPDIF"
#set_property PACKAGE_PIN Y18  [get_ports {HD_SPDIFO}];  # "HD-SPDIFO"
#set_property PACKAGE_PIN W17  [get_ports {HD_VSYNC}];  # "HD-VSYNC"

# ------------------------------------------------------------------------
# User LEDs - Bank 33
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN T22 [get_ports {LD0}];  # "LD0"
#set_property PACKAGE_PIN T21 [get_ports {LD1}];  # "LD1"
#set_property PACKAGE_PIN U22 [get_ports {LD2}];  # "LD2"
#set_property PACKAGE_PIN U21 [get_ports {LD3}];  # "LD3"
#set_property PACKAGE_PIN V22 [get_ports {LD4}];  # "LD4"
#set_property PACKAGE_PIN W22 [get_ports {LD5}];  # "LD5"
#set_property PACKAGE_PIN U19 [get_ports {LD6}];  # "LD6"
#set_property PACKAGE_PIN U14 [get_ports {LD7}];  # "LD7"

# ------------------------------------------------------------------------
# VGA Output - Bank 33
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN Y21  [get_ports {VGA_B1}];  # "VGA-B1"
#set_property PACKAGE_PIN Y20  [get_ports {VGA_B2}];  # "VGA-B2"
#set_property PACKAGE_PIN AB20 [get_ports {VGA_B3}];  # "VGA-B3"
#set_property PACKAGE_PIN AB19 [get_ports {VGA_B4}];  # "VGA-B4"
#set_property PACKAGE_PIN AB22 [get_ports {VGA_G1}];  # "VGA-G1"
#set_property PACKAGE_PIN AA22 [get_ports {VGA_G2}];  # "VGA-G2"
#set_property PACKAGE_PIN AB21 [get_ports {VGA_G3}];  # "VGA-G3"
#set_property PACKAGE_PIN AA21 [get_ports {VGA_G4}];  # "VGA-G4"
#set_property PACKAGE_PIN AA19 [get_ports {VGA_HS}];  # "VGA-HS"
#set_property PACKAGE_PIN V20  [get_ports {VGA_R1}];  # "VGA-R1"
#set_property PACKAGE_PIN U20  [get_ports {VGA_R2}];  # "VGA-R2"
#set_property PACKAGE_PIN V19  [get_ports {VGA_R3}];  # "VGA-R3"
#set_property PACKAGE_PIN V18  [get_ports {VGA_R4}];  # "VGA-R4"
#set_property PACKAGE_PIN Y19  [get_ports {VGA_VS}];  # "VGA-VS"
```

```
# ------------------------------------------------------------------------
# User Push Buttons - Bank 34
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN P16 [get_ports {BTNC}];  # "BTNC"
#set_property PACKAGE_PIN R16 [get_ports {BTND}];  # "BTND"
#set_property PACKAGE_PIN N15 [get_ports {BTNL}];  # "BTNL"
#set_property PACKAGE_PIN R18 [get_ports {BTNR}];  # "BTNR"
#set_property PACKAGE_PIN T18 [get_ports {BTNU}];  # "BTNU"

# ------------------------------------------------------------------------
# USB OTG Reset - Bank 34
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN L16 [get_ports {OTG_VBUSOC}];  # "OTG-VBUSOC"

# ------------------------------------------------------------------------
# XADC GIO - Bank 34
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN H15 [get_ports {XADC_GIO0}];  # "XADC-GIO0"
#set_property PACKAGE_PIN R15 [get_ports {XADC_GIO1}];  # "XADC-GIO1"
#set_property PACKAGE_PIN K15 [get_ports {XADC_GIO2}];  # "XADC-GIO2"
#set_property PACKAGE_PIN J15 [get_ports {XADC_GIO3}];  # "XADC-GIO3"

# ------------------------------------------------------------------------
# Miscellaneous - Bank 34
# ------------------------------------------------------------------------
#set_property PACKAGE_PIN K16 [get_ports {PUDC_B}];  # "PUDC_B"

## ------------------------------------------------------------------------
## USB OTG Reset - Bank 35
## ------------------------------------------------------------------------
#set_property PACKAGE_PIN G17 [get_ports {OTG_RESETN}];  # "OTG-RESETN"

## ------------------------------------------------------------------------
## User DIP Switches - Bank 35
## ------------------------------------------------------------------------
#set_property PACKAGE_PIN F22 [get_ports {SW0}];  # "SW0"
#set_property PACKAGE_PIN G22 [get_ports {SW1}];  # "SW1"
#set_property PACKAGE_PIN H22 [get_ports {SW2}];  # "SW2"
#set_property PACKAGE_PIN F21 [get_ports {SW3}];  # "SW3"
#set_property PACKAGE_PIN H19 [get_ports {SW4}];  # "SW4"
#set_property PACKAGE_PIN H18 [get_ports {SW5}];  # "SW5"
#set_property PACKAGE_PIN H17 [get_ports {SW6}];  # "SW6"
#set_property PACKAGE_PIN M15 [get_ports {SW7}];  # "SW7"

## ------------------------------------------------------------------------
## XADC AD Channels - Bank 35
## ------------------------------------------------------------------------
#set_property PACKAGE_PIN E16 [get_ports {AD0N_R}];  # "XADC-AD0N-R"
#set_property PACKAGE_PIN F16 [get_ports {AD0P_R}];  # "XADC-AD0P-R"
#set_property PACKAGE_PIN D17 [get_ports {AD8N_N}];  # "XADC-AD8N-R"
#set_property PACKAGE_PIN D16 [get_ports {AD8P_R}];  # "XADC-AD8P-R"

## ------------------------------------------------------------------------
## FMC Expansion Connector - Bank 13
## ------------------------------------------------------------------------
#set_property PACKAGE_PIN R7 [get_ports {FMC_SCL}];  # "FMC-SCL"
#set_property PACKAGE_PIN U7 [get_ports {FMC_SDA}];  # "FMC-SDA"

## ------------------------------------------------------------------------
## FMC Expansion Connector - Bank 33
## ------------------------------------------------------------------------
#set_property PACKAGE_PIN AB14 [get_ports {FMC_PRSNT}];  # "FMC-PRSNT"

## ------------------------------------------------------------------------
## FMC Expansion Connector - Bank 34
## ------------------------------------------------------------------------
#set_property PACKAGE_PIN L19 [get_ports {FMC_CLK0_N}];  # "FMC-CLK0_N"
#set_property PACKAGE_PIN L18 [get_ports {FMC_CLK0_P}];  # "FMC-CLK0_P"
#set_property PACKAGE_PIN M20 [get_ports {FMC_LA00_CC_N}];  # "FMC-LA00_CC_N"
#set_property PACKAGE_PIN M19 [get_ports {FMC_LA00_CC_P}];  # "FMC-LA00_CC_P"
#set_property PACKAGE_PIN N20 [get_ports {FMC_LA01_CC_N}];  # "FMC-LA01_CC_N"
#set_property PACKAGE_PIN N19 [get_ports {FMC_LA01_CC_P}];  # "FMC-LA01_CC_P"
#set_property PACKAGE_PIN P18 [get_ports {FMC_LA02_N}];  # "FMC-LA02_N"
#set_property PACKAGE_PIN P17 [get_ports {FMC_LA02_P}];  # "FMC-LA02_P"
#set_property PACKAGE_PIN P22 [get_ports {FMC_LA03_N}];  # "FMC-LA03_N"
#set_property PACKAGE_PIN N22 [get_ports {FMC_LA03_P}];  # "FMC-LA03_P"
#set_property PACKAGE_PIN M22 [get_ports {FMC_LA04_N}];  # "FMC-LA04_N"
```

```
#set_property PACKAGE_PIN M21 [get_ports {FMC_LA04_P}];  # "FMC-LA04_P"
#set_property PACKAGE_PIN K18 [get_ports {FMC_LA05_N}];  # "FMC-LA05_N"
#set_property PACKAGE_PIN J18 [get_ports {FMC_LA05_P}];  # "FMC-LA05_P"
#set_property PACKAGE_PIN L22 [get_ports {FMC_LA06_N}];  # "FMC-LA06_N"
#set_property PACKAGE_PIN L21 [get_ports {FMC_LA06_P}];  # "FMC-LA06_P"
#set_property PACKAGE_PIN T17 [get_ports {FMC_LA07_N}];  # "FMC-LA07_N"
#set_property PACKAGE_PIN T16 [get_ports {FMC_LA07_P}];  # "FMC-LA07_P"
#set_property PACKAGE_PIN J22 [get_ports {FMC_LA08_N}];  # "FMC-LA08_N"
#set_property PACKAGE_PIN J21 [get_ports {FMC_LA08_P}];  # "FMC-LA08_P"
#set_property PACKAGE_PIN R21 [get_ports {FMC_LA09_N}];  # "FMC-LA09_N"
#set_property PACKAGE_PIN R20 [get_ports {FMC_LA09_P}];  # "FMC-LA09_P"
#set_property PACKAGE_PIN T19 [get_ports {FMC_LA10_N}];  # "FMC-LA10_N"
#set_property PACKAGE_PIN R19 [get_ports {FMC_LA10_P}];  # "FMC-LA10_P"
#set_property PACKAGE_PIN N18 [get_ports {FMC_LA11_N}];  # "FMC-LA11_N"
#set_property PACKAGE_PIN N17 [get_ports {FMC_LA11_P}];  # "FMC-LA11_P"
#set_property PACKAGE_PIN P21 [get_ports {FMC_LA12_N}];  # "FMC-LA12_N"
#set_property PACKAGE_PIN P20 [get_ports {FMC_LA12_P}];  # "FMC-LA12_P"
#set_property PACKAGE_PIN M17 [get_ports {FMC_LA13_N}];  # "FMC-LA13_N"
#set_property PACKAGE_PIN L17 [get_ports {FMC_LA13_P}];  # "FMC-LA13_P"
#set_property PACKAGE_PIN K20 [get_ports {FMC_LA14_N}];  # "FMC-LA14_N"
#set_property PACKAGE_PIN K19 [get_ports {FMC_LA14_P}];  # "FMC-LA14_P"
#set_property PACKAGE_PIN J17 [get_ports {FMC_LA15_N}];  # "FMC-LA15_N"
#set_property PACKAGE_PIN J16 [get_ports {FMC_LA15_P}];  # "FMC-LA15_P"
#set_property PACKAGE_PIN K21 [get_ports {FMC_LA16_N}];  # "FMC-LA16_N"
#set_property PACKAGE_PIN J20 [get_ports {FMC_LA16_P}];  # "FMC-LA16_P"

## ------------------------------------------------------------------------
## FMC Expansion Connector - Bank 35
## ------------------------------------------------------------------------
#set_property PACKAGE_PIN C19 [get_ports {FMC_CLK1_N}];  # "FMC-CLK1_N"
#set_property PACKAGE_PIN D18 [get_ports {FMC_CLK1_P}];  # "FMC-CLK1_P"
#set_property PACKAGE_PIN B20 [get_ports {FMC_LA17_CC_N}];  # "FMC-LA17_CC_N"
#set_property PACKAGE_PIN B19 [get_ports {FMC_LA17_CC_P}];  # "FMC-LA17_CC_P"
#set_property PACKAGE_PIN C20 [get_ports {FMC_LA18_CC_N}];  # "FMC-LA18_CC_N"
#set_property PACKAGE_PIN D20 [get_ports {FMC_LA18_CC_P}];  # "FMC-LA18_CC_P"
#set_property PACKAGE_PIN G16 [get_ports {FMC_LA19_N}];  # "FMC-LA19_N"
#set_property PACKAGE_PIN G15 [get_ports {FMC_LA19_P}];  # "FMC-LA19_P"
#set_property PACKAGE_PIN G21 [get_ports {FMC_LA20_N}];  # "FMC-LA20_N"
#set_property PACKAGE_PIN G20 [get_ports {FMC_LA20_P}];  # "FMC-LA20_P"
#set_property PACKAGE_PIN E20 [get_ports {FMC_LA21_N}];  # "FMC-LA21_N"
#set_property PACKAGE_PIN E19 [get_ports {FMC_LA21_P}];  # "FMC-LA21_P"
#set_property PACKAGE_PIN F19 [get_ports {FMC_LA22_N}];  # "FMC-LA22_N"
#set_property PACKAGE_PIN G19 [get_ports {FMC_LA22_P}];  # "FMC-LA22_P"
#set_property PACKAGE_PIN D15 [get_ports {FMC_LA23_N}];  # "FMC-LA23_N"
#set_property PACKAGE_PIN E15 [get_ports {FMC_LA23_P}];  # "FMC-LA23_P"
#set_property PACKAGE_PIN A19 [get_ports {FMC_LA24_N}];  # "FMC-LA24_N"
#set_property PACKAGE_PIN A18 [get_ports {FMC_LA24_P}];  # "FMC-LA24_P"
#set_property PACKAGE_PIN C22 [get_ports {FMC_LA25_N}];  # "FMC-LA25_N"
#set_property PACKAGE_PIN D22 [get_ports {FMC_LA25_P}];  # "FMC-LA25_P"
#set_property PACKAGE_PIN E18 [get_ports {FMC_LA26_N}];  # "FMC-LA26_N"
#set_property PACKAGE_PIN F18 [get_ports {FMC_LA26_P}];  # "FMC-LA26_P"
#set_property PACKAGE_PIN D21 [get_ports {FMC_LA27_N}];  # "FMC-LA27_N"
#set_property PACKAGE_PIN E21 [get_ports {FMC_LA27_P}];  # "FMC-LA27_P"
#set_property PACKAGE_PIN A17 [get_ports {FMC_LA28_N}];  # "FMC-LA28_N"
#set_property PACKAGE_PIN A16 [get_ports {FMC_LA28_P}];  # "FMC-LA28_P"
#set_property PACKAGE_PIN C18 [get_ports {FMC_LA29_N}];  # "FMC-LA29_N"
#set_property PACKAGE_PIN C17 [get_ports {FMC_LA29_P}];  # "FMC-LA29_P"
#set_property PACKAGE_PIN B15 [get_ports {FMC_LA30_N}];  # "FMC-LA30_N"
#set_property PACKAGE_PIN C15 [get_ports {FMC_LA30_P}];  # "FMC-LA30_P"
#set_property PACKAGE_PIN B17 [get_ports {FMC_LA31_N}];  # "FMC-LA31_N"
#set_property PACKAGE_PIN B16 [get_ports {FMC_LA31_P}];  # "FMC-LA31_P"
#set_property PACKAGE_PIN A22 [get_ports {FMC_LA32_N}];  # "FMC-LA32_N"
#set_property PACKAGE_PIN A21 [get_ports {FMC_LA32_P}];  # "FMC-LA32_P"
#set_property PACKAGE_PIN B22 [get_ports {FMC_LA33_N}];  # "FMC-LA33_N"
#set_property PACKAGE_PIN B21 [get_ports {FMC_LA33_P}];  # "FMC-LA33_P"
```

```
# ------------------------------------------------------------------------
# IOSTANDARD Constraints
#
# Note that these IOSTANDARD constraints are applied to all IOs currently
# assigned within an I/O bank.  If these IOSTANDARD constraints are
# evaluated prior to other PACKAGE_PIN constraints being applied, then
# the IOSTANDARD specified will likely not be applied properly to those
# pins.  Therefore, bank wide IOSTANDARD constraints should be placed
# within the XDC file in a location that is evaluated AFTER all
# PACKAGE_PIN constraints within the target bank have been evaluated.
#
# Un-comment one or more of the following IOSTANDARD constraints according to
# the bank pin assignments that are required within a design.
# ------------------------------------------------------------------------

# Note that the bank voltage for IO Bank 33 is fixed to 3.3V on ZedBoard.
# set_property IOSTANDARD LVCMOS33 [get_ports -of_objects [get_iobanks 33]];

# Set the bank voltage for IO Bank 34 to 1.8V by default.
# set_property IOSTANDARD LVCMOS33 [get_ports -of_objects [get_iobanks 34]];
# set_property IOSTANDARD LVCMOS25 [get_ports -of_objects [get_iobanks 34]];
# set_property IOSTANDARD LVCMOS18 [get_ports -of_objects [get_iobanks 34]];

# Set the bank voltage for IO Bank 35 to 1.8V by default.
# set_property IOSTANDARD LVCMOS33 [get_ports -of_objects [get_iobanks 35]];
# set_property IOSTANDARD LVCMOS25 [get_ports -of_objects [get_iobanks 35]];
# set_property IOSTANDARD LVCMOS18 [get_ports -of_objects [get_iobanks 35]];

# Note that the bank voltage for IO Bank 13 is fixed to 3.3V on ZedBoard.
# set_property IOSTANDARD LVCMOS33 [get_ports -of_objects [get_iobanks 13]];

############################################################################
set_property -dict {PACKAGE_PIN P16 IOSTANDARD LVCMOS33} [get_ports {reset_rtl}];
# "BTNC"
set_property -dict {PACKAGE_PIN Y9 IOSTANDARD LVCMOS33} [get_ports {sys_clock}];
# "GCLK"

# set_property -dict {PACKAGE_PIN T22 IOSTANDARD LVCMOS33} [get_ports {leds_8bits[0]}];
# "LD0"
# set_property -dict {PACKAGE_PIN T21 IOSTANDARD LVCMOS33} [get_ports {leds_8bits[1]}];
# "LD1"
# set_property -dict {PACKAGE_PIN U22 IOSTANDARD LVCMOS33} [get_ports {leds_8bits[2]}];
# "LD2"
# set_property -dict {PACKAGE_PIN U21 IOSTANDARD LVCMOS33} [get_ports {leds_8bits[3]}];
# "LD3"
# set_property -dict {PACKAGE_PIN V22 IOSTANDARD LVCMOS33} [get_ports {leds_8bits[4]}];
# "LD4"
# set_property -dict {PACKAGE_PIN W22 IOSTANDARD LVCMOS33} [get_ports {leds_8bits[5]}];
# "LD5"
# set_property -dict {PACKAGE_PIN U19 IOSTANDARD LVCMOS33} [get_ports {leds_8bits[6]}];
# "LD6"
# set_property -dict {PACKAGE_PIN U14 IOSTANDARD LVCMOS33} [get_ports {leds_8bits[7]}];
# "LD7"

# create_clock -period 10.0 -name sys_clock -waveform {0.000 5.000} [get_ports {sys_clock}];
```