ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ

ΕΛΛΑΔΑ
ΥΠ.Ε.Π.Θ.

ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

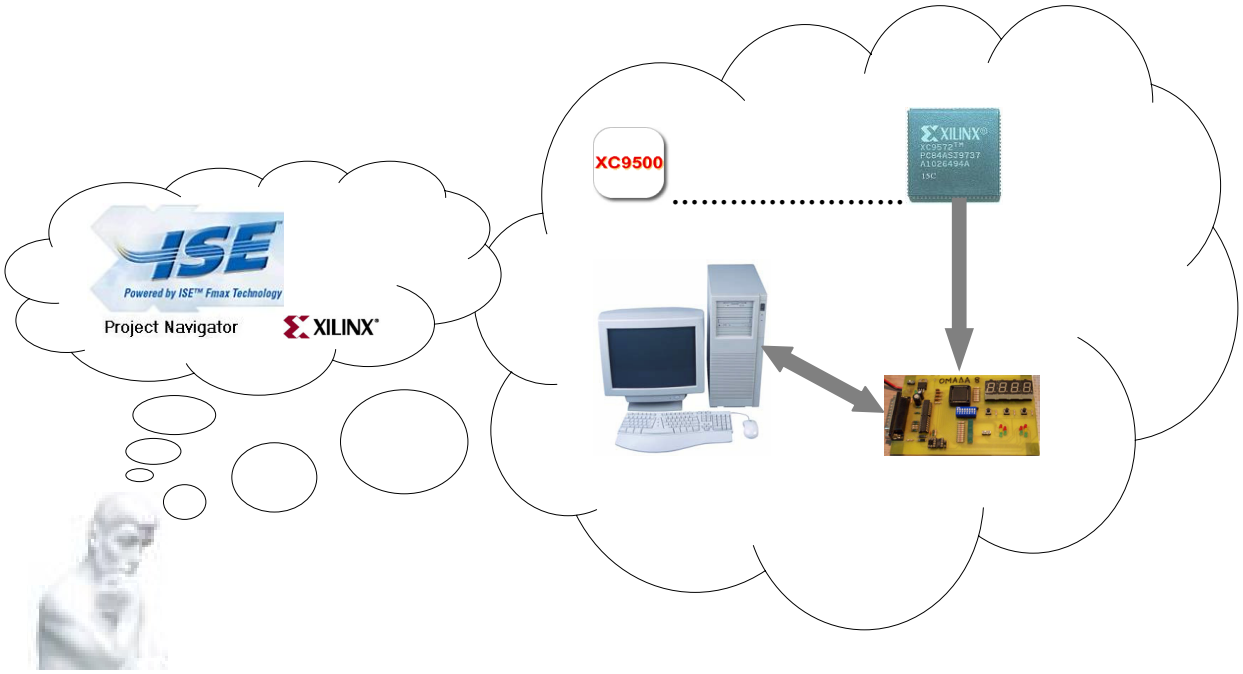
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Του φοιτητή Κωνσταντίνου Κοκίδη (Κ.Α.Σ.: 599023)

Τίτλος: «Δημιουργία τράπεζας Ψηφιακών Κυκλωμάτων με χρήση τεχνολογίας P.L.D.»

Επιβλέπων Καθηγητής: Χρήστος Τζίκας – Καθηγητής Εφαρμογών



ΘΕΣΣΑΛΟΝΙΚΗ 2009

ΠΙΝΑΚΑΣ ΑΚΑΔΗΜΑΪΚΩΝ ΣΤΟΙΧΕΙΩΝ

Πτυχιακή εργασία

Φοιτητής: Κωνσταντίνος Κοκίδης (Κ.Α.Σ.: 599023)
Επιβλέπων Καθηγητής: Χρήστος Τζίκας – Καθηγητής Εφαρμογών
Τίτλος: «Δημιουργία τράπεζας Ψηφιακών Κυκλωμάτων με χρήση τεχνολογίας P.L.D.» Κωδικός: 08110ΥΜ.
Ημερομηνία ανάληψης: 05/03/2008. Ημερομηνία περάτωσης: 10/09/2009.

ΠΕΡΙΛΗΨΗ

……Το περιεχόμενο της πτυχιακής εργασίας περιλαμβάνει μια συλλογή Ψηφιακών Κυκλωμάτων, συνδυαστικών και ακολουθιακών προγραμματιζομένων με την εφαρμογή της εταιρείας Xilinx. Για τον προγραμματισμό σε CPLD ή FPGA χρησιμοποιήθηκαν δύο γλώσσες. Πρώτα χρησιμοποιήθηκε η σχηματική γλώσσα και κατόπιν η περιγραφική (VHDL). Πιστεύω ό,τι αυτή η πτυχιακή εργασία θα βοηθήσει τους φοιτητές δίνοντάς τους κατευθυντική γραμμή σκέψης και αντίδρασης στην νέα τεχνολογία των P.L.D. (Προγραμματιζομένων Λογικών Συσκευών). Η μελλοντική μου ευχή είναι κάποιος άλλος φοιτητής να συνεχίσει την προσπάθεια, συνθέτοντας περισσότερα συνδυαστικά ψηφιακά κυκλώματα με το μοντέλο που έχω προετοιμάσει…..

ABSTRACT

……..This final project includes a number of basic Digital Circuits, combinational and sequential prepared by Xilinx program. I used two languages for programming in to a CPLD or FPGA. First the schematic language and second the VHDL. I believe that this final project will help the Students by giving them a direction how to think and how to react with the new technology of PLD's (Programmable Logic Devices). My proposal for the future is that another student will continue the project, by synthesizing more combinational digital circuits using the models I had prepared……

ΠΡΟΛΟΓΟΣ

Στην παρούσα πτυχιακή εργασία περιλαμβάνεται ένας αριθμός ψηφιακών κυκλωμάτων που ανήκουν στην ύλη διδασκαλίας των εργαστηρίων των Ψηφιακών Κυκλωμάτων Ι και ΙΙ, όπως αυτά διδάσκονται στο τμήμα Ηλεκτρονικής του Α.Τ.Ε.Ι. Θεσσαλονίκης, σε διάρκεια δύο εξαμήνων. Με το σύνολο των κυκλωμάτων αυτών προετοιμάστηκε να προγραμματιστεί τελικά το ολοκληρωμένο κύκλωμα xc9572-pc44 τεχνολογίας C.P.L.D. της σειράς xc9500 της εταιρείας Xilinx® στην έκδοση της χρονοκαθυστέρησης των 15 nsec. Ο προγραμματισμός έγινε με την εφαρμογή Xilinx® ISE™ 9.1.03i. Το συγκεκριμένο ολοκληρωμένο κύκλωμα είναι εγκατεστημένο στην αναπτυξιακή πλακέτα του εργαστηρίου των Ψηφιακών Κυκλωμάτων Ι και ΙΙ όπου έγινε και ο προγραμματισμός του.

Στο πρώτο κεφάλαιο περιλαμβάνονται συνολικές εργασίες που αφορούν τα συνδυαστικά κυκλώματα. Συγκεκριμένα περιλαμβάνονται ο πολυπλέκτης 8:1, ο αποπολυπλέκτης 3:8, ο κωδικοποιητής 8 σε 3, ο αποκωδικοποιητής 3 σε 8, ο συγκριτής μεγέθους προσημασμένων αριθμών τεσσάρων δυαδικών ψηφίων σε συμπλήρωμα του 2 (2's), ο προσθαφαιρέτης προσημασμένων αριθμών τεσσάρων δυαδικών ψηφίων σε συμπλήρωμα του 2 (2's).

Στο δεύτερο κεφάλαιο περιλαμβάνονται συνολικές εργασίες που αφορούν τα ακολουθιακά κυκλώματα. Περιλαμβάνονται λοιπόν ο δυαδικός μετρητής mod-16 στις τρεις μορφές του αύξοντα, φθίνοντα και αύξοντα/φθίνοντα. Οι καταχωρητές ολίσθησης δεξιάς κατεύθυνσης χωρητικότητας οκτώ ψηφίων στις μορφές της σειριακής εισόδου - σειριακής εξόδου, σειριακής εισόδου –

παράλληλης εξόδου, παράλληλης εισόδου - σειριακής εξόδου, παράλληλης εισόδου - παράλληλης εξόδου. Οι μετρητές Johnson και κυκλικός σαν ειδικές περιπτώσεις των καταχωρητών ολίσθησης.

Βασικό σημείο και στοιχείο στήριξης με τακτικά βήματα τόσο για τη μελέτη και κατανόηση του προγράμματος Xilinx® ISE™ 9.1.03i όσο και του κορμού ανάπτυξης των παραπάνω συνολικών εργασιών αποτελεί η συνολική εργασία του πολυπλέκτη 8:1. Στην συνολική εργασία αυτή υπάρχει λοιπόν μεθοδική και βηματική ανάλυση των διαφόρων μερικών εργασιών που ακολουθούμε με την εφαρμογή της Xilinx® που θα οδηγήσει στον προγραμματισμό του ολοκληρωμένου μας τόσο στην σχηματική γλώσσα (schematic) όσο και στην περιγραφική γλώσσα VHDL. Ο τρόπος αυτός γραφής θα μπορούσε να χρησιμοποιηθεί σαν ένα είδος εκμάθησης του προγράμματος πάντα σε αντιπαραβολή με τις εικόνες που υπάρχουν στο παράρτημα Α για παραπάνω υποστήριξη στην προσπάθεια αυτή. Να σημειωθεί επίσης ό,τι η υποστήριξη συνεχίζεται με παροχή αρχείων κινουμένης εικόνας (video) στην ηλεκτρονική έκδοση της συγκεκριμένης πτυχιακής εργασίας.

Τέλος, με το σύνολο των συνθέσιμων συνολικών εργασιών που επιμελήθηκα και κατέληξαν σε τελικό προγραμματισμό με τη συγκεκριμένη εφαρμογή, θέλω να πιστεύω ότι συμβάλλω στοιχειωδώς στην γρήγορη εκμάθησή της από τον κάθε αναγνώστη της πτυχιακής μου εργασίας.

Κωνσταντίνος Κοκίδης

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΣΥΝΔΥΑΣΤΙΚΑ ΚΥΚΛΩΜΑΤΑ

Ακολουθεί η ανάπτυξη σε σχηματική και περιγραφική γλώσσα (VHDL) των παρακάτω συνολικών εργασιών (project):

- πολυπλέκτης 8:1
- αποπολυπλέκτης 3:8
- κωδικοποιητής 8 σε 3
- αποκωδικοποιητής 3 σε 8
- μετατροπέας κώδικα (δεκαεξαδικού σε κώδικα για οδήγηση ενδείκτη 7 τομέων)
- συγκριτής μεγέθους προσημασμένων αριθμών (τεσσάρων ψηφίων)
- προσθέτης - αφαιρέτης τεσσάρων ψηφίων προσημασμένων αριθμών.

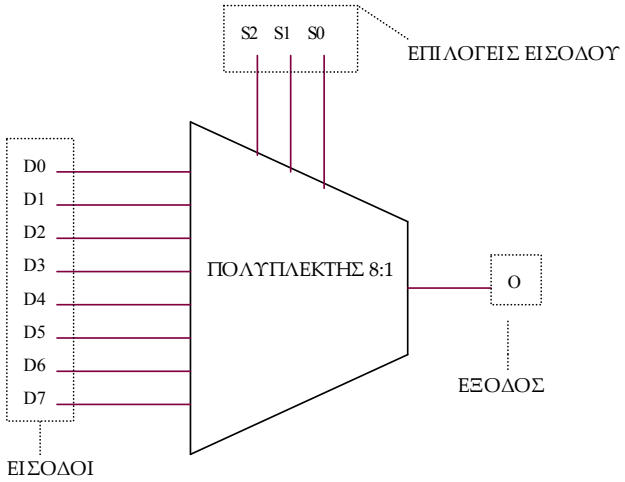
1.1 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ: ΠΟΛΥΠΛΕΚΤΗΣ 8:1

1.1.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

1.1.1.1 Σύντομη θεωρία

Ο ψηφιακός πολυπλέκτης (*mu*ltiple*x*er, *mux*) ή επιλογέας δεδομένων (data selector) αποτελεί ένα συνδυαστικό ψηφιακό κύκλωμα. Πρόκειται λοιπόν για ένα ψηφιακό μεταγωγέα πληροφορίας της μίας από τις εισόδους του στην επίσης μοναδική έξοδό του. Η επιλογή της εισόδου που θα μετάγει την πληροφορία της στην έξοδο καθορίζεται από τις γραμμές επιλογής εισόδου. Η ψηφιακή πληροφορία μεταφέρεται μέσω παράλληλων γραμμών αποτιμημένη σε δυαδικό ψηφίο (binary digit, bit) στην περίπτωση μονής γραμμής ή λεωφόρου (bus) στην περίπτωση περισσοτέρων γραμμών. Για τον γενικό πολυπλέκτη 2^n:1 η πληροφορία λεωφόρου n bits επιλογής εισόδου επιλέγει ποιο από τα 2^n bits της λεωφόρου δεδομένων μπορεί να μεταφερθεί στην έξοδο του ενός bit. Το σύμβολο 2^n:1 διαβάζεται «2^n σε 1» ή «1 από 2^n» και η παραπάνω διαδικασία ονομάζεται πολυπλεξία.

Για τον πολυπλέκτη 8:1 μία λεωφόρος n=3 bits δεδομένων επιλογής εισόδου μπορεί να επιλέγει (**πίνακας 1.1**) σε ποιο από τα 2^3=8 bits της λεωφόρου δεδομένων μπορεί να μεταφερθεί στην έξοδο του ενός bit. Τα δεδομένα επιλογής εισόδου συμβολίζονται με S0, S1 και S2 (MSB=S2). Τα δεδομένα εισόδου συμβολίζονται με D0, D1,…,D7 (MSB=D7). Η μοναδική έξοδος συμβολίζεται με O. Στον **πίνακα 1.1** απεικονίζεται ο συντετμημένος πίνακας αλήθειας ενός πολυπλέκτη 8:1 με γραφικό σύμβολο αυτό της **εικόνας 1.1**, λογική εξίσωση υλοποίησης σε άθροισμα ελαχίστων όρων (minterm) αυτή της **εικόνας 1.2**.



Εικόνα 1.1 Γραφικό σύμβολο πολυπλέκτη 8:1

Πίνακας 1.1 Συντετμημένος πίνακας αλήθειας πολυπλέκτη 8:1

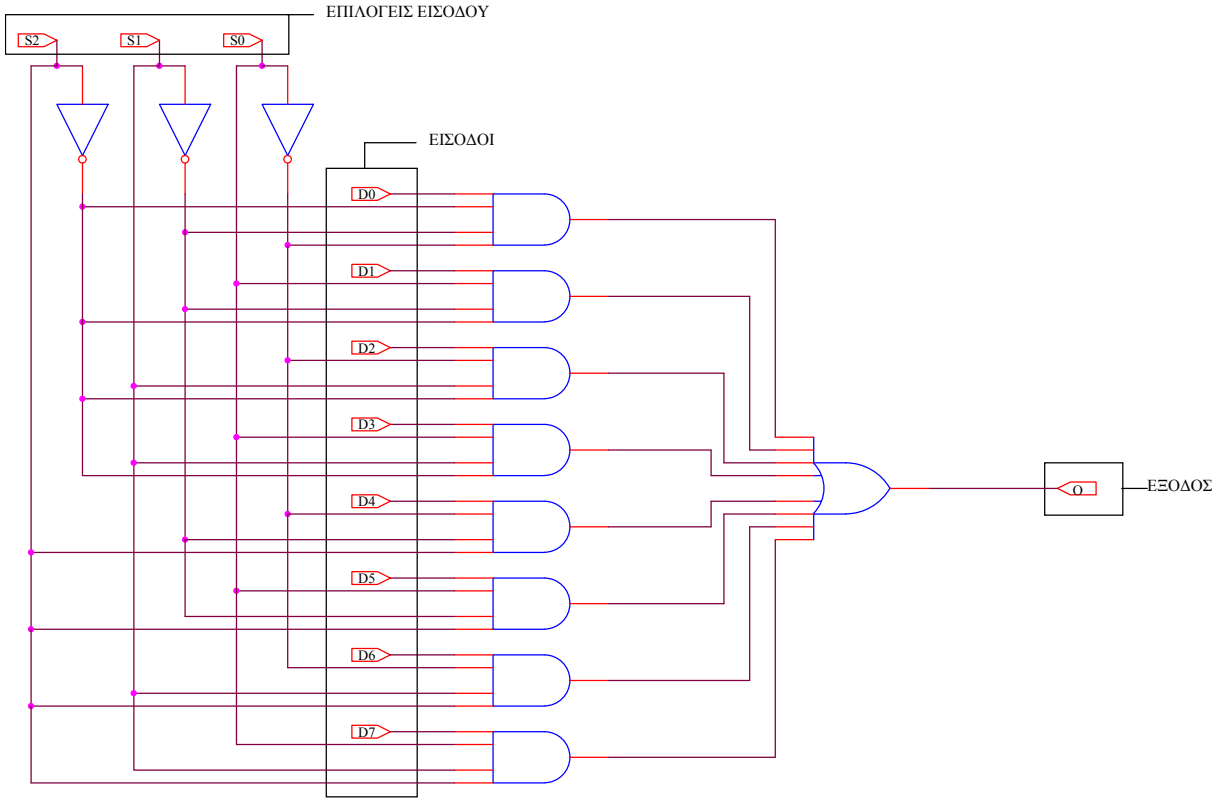
Επιλογή Εισόδων			**Έξοδος = Επιλεγόμενη Είσοδος**
MSB		**LSB**	
S2	**S1**	**S0**	**O**
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

$$O = D0 \cdot \overline{S2} \cdot \overline{S1} \cdot \overline{S0} + D1 \cdot \overline{S2} \cdot \overline{S1} \cdot S0 + D2 \cdot \overline{S2} \cdot S1 \cdot \overline{S0} + D3 \cdot \overline{S2} \cdot S1 \cdot S0 + \\ + D4 \cdot S2 \cdot \overline{S1} \cdot \overline{S0} + D5 \cdot S2 \cdot \overline{S1} \cdot S0 + D6 \cdot S2 \cdot S1 \cdot \overline{S0} + D7 \cdot S2 \cdot S1 \cdot S0$$

Εικόνα 1.2 Λογική εξίσωση πολυπλέκτη 8:1 σε άθροισμα ελαχίστων όρων (minterm)

1.1.1.2 Συμβατική ψηφιακή υλοποίηση

Η λογική εξίσωση της **εικόνας 1.2** παραπέμπει στο λογικό κύκλωμα υλοποίησης με συμβατικές λογικές πύλες της παρακάτω **εικόνας 1.3**.



Εικόνα 1.3 Λογικό κύκλωμα υλοποίησης για πολυπλέκτη 8:1

1.1.2 ΑΝΑΠΤΥΞΗ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

Οι **εικόνες** (**Α.1** ως **Α.75β**) βρίσκονται στο παράρτημα Α.

1.1.2.1 Έναρξη εφαρμογής

- Με **διπλό** χτύπημα (κλικ) του ποντικιού πάνω στην συντόμευση της επιφάνειας εργασίας (**εικόνας Α.1)** ξεκινά η εφαρμογή **Project Navigator** (**εικόνα Α.2**) Άλλη μία εναλλακτική διαδρομή έναρξης της εφαρμογής για τα Windows XP είναι: **Start→Programs→Xilinx ISE 9.1i→Project Navigator (αγγλική έκδοση) ή Έναρξη→Όλα τα προγράμματα→Xilinx ISE 9.1i→Project Navigator (ελληνική έκδοση).**

Η εφαρμογή **Project Navigator** μας εισάγει στο αναπτυξιακό παραθυρικό περιβάλλον του προγράμματος για εργασίες (projects). (**Εικόνα Α.3**)

1.1.2.2 Δημιουργία νέας συνολικής εργασίας

- Στο νέο περιβάλλον επιλέγουμε την διαδρομή: **File→New Project.** (**Εικόνα Α.4**)
- Με το άνοιγμα του παραθύρου **New Project Wizard – Create New Project**

στην περιοχή **Project Name:** πληκτρολογούμε **mux_s**,

στην περιοχή **Project Location:** έχουμε αυτόματη δημιουργία του φάκελου **mux_s** στην προεπιλεγμένη διαδρομή της επιλογής μας**,**

στην περιοχή **Top-Level Source Type:** επιλέγουμε **Schematic** σαν την γλώσσα κορυφαίου επίπεδου ιεραρχίας όπου θα εργαστούμε.

Στη συνέχεια επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα Α.5**)

- Με το άνοιγμα του παράθυρου **New Project Wizard – Device Properties**

στην περιοχή **Device Family**: επιλέγουμε **XC9500 CPLDs**

στην περιοχή **Device:** επιλέγουμε **XC9572 (**ή **XC9536)**

στην περιοχή **Package:** επιλέγουμε **PC44**

στην περιοχή **Speed:** επιλέγουμε **-15**

στην περιοχή **Synthesis Tool:** επιλέγουμε **XST (VHDL/Verilog)**

στην περιοχή **Simulator:** επιλέγουμε **ISE Simulator (VHDL/Verilog).**

Στη συνέχεια επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα A.6**)

- Με το άνοιγμα του παράθυρου **New Project Wizard – Create New Source** επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **New Source**. (**Εικόνα A.7**)
- Με το άνοιγμα του παράθυρου **New Project Wizard – Select Source Type** με απλό κλικ του ποντικιού υπογραμμίζουμε το κουμπί **Add to Project**, με απλό κλικ του ποντικιού επιλέγουμε **Schematic**, στην περιοχή **File Name:** πληκτρολογούμε **mux_sch**.

Επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα A.8**)

- Με το άνοιγμα του παράθυρου **New Project Wizard – Summary** επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Finish**. (**Εικόνα A.9**)

Στην ερώτηση για δημιουργία του ανύπαρκτου καταλόγου με το όνομα που δώσαμε προηγουμένως επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Yes**. (**Εικόνα A.10**)

- Με το άνοιγμα του παράθυρου **New Project Wizard – Create New Source** επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα A.11**)
- Με το άνοιγμα του παράθυρου **New Project Wizard – Add Existing Sources** επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα A.12**)
- Με το άνοιγμα του παράθυρου **New Project Wizard – Project Summary** επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Finish**. (**Εικόνα A.13**) Τέλος της εισαγωγής στοιχείων της νέας εργασίας (project).

1.1.2.3 Νέο σχηματικό

1.1.2.3.1 Μεγιστοποίηση επιφάνειας εργασίας

◆ Στη συνέχεια εισερχόμαστε στο περιβάλλον ανάπτυξης εργασιών (project) στην σχηματική γλώσσα. (**Εικόνα Α.14**)

Για τη μεγιστοποίηση της επιφάνειας του περιβάλλοντος επιλέγουμε με απλό κλικ του ποντικιού το εικονίδιο **Float this window** από τη μπάρα εργαλείων. (**Εικόνα Α.15**)

Προκύπτοντας η μεγιστοποιημένη επιφάνεια εργασίας του σχηματικού με το δικό της παράθυρο. (**Εικόνα Α.16**)

1.1.2.3.2 Αντιστοίχηση και τοποθέτηση υλικών

◆ Για διαχείριση των υλικών επιλέγουμε με απλό κλικ του ποντικιού την ετικέτα **Symbols→Logic.** (**Εικόνες Α.17α** και **Α.17β**)

Στον **πίνακα Α.1** αντιστοιχίζονται τα υλικά από την συμβατική σχεδίαση με αυτά των βιβλιοθηκών του σχηματικού ώστε να είναι προσβάσιμα. Μια βέλτιστη αντιστοίχηση φαίνεται στον **πίνακα Α.2** με σκοπό τη μείωση του αριθμού των απαραίτητων υλικών για τη σχεδίαση που δανείζομαστε από τις βιβλιοθήκες.

◆ Στην περίπτωση διαχείρισης πληροφορίας του κάθε υλικού το επιλέγουμε και στην συνέχεια ξαναεπιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Symbol Info**. (**Εικόνες Α.18α** και **Α.18β**) Εμφανίζεται λοιπόν ένα αρχείο μορφής pdf έξι σελίδων. (**Εικόνα Α.19**)

◆ Έχοντας επιλέξει με απλό κλικ του ποντικιού τη διαδρομή **Symbols→Logic** και το υλικό της επιλογής μας (εδώ **and4b1**) μπορούμε να το σύρουμε και ξανά με απλό κλικ του ποντικιού να το τοποθετήσουμε στην επιφάνεια εργασίας του σχηματικού. (**Πίνακας Α.3 με εικόνες Α.20α** και **Α.20β**)

Η παραπάνω ενέργεια σταματά πατώντας το πλήκτρο **Esc**.

◆ Επαναλαμβάνοντας την διαδικασία τοποθέτησης με όλα τα απαραίτητα προς σχεδίαση υλικά προκύπτει η **εικόνα Α.21**.

1.1.2.3.3 Τοποθέτηση απομονωτών εισόδου/εξόδου (I/O)

◆ Με τον παραπάνω τρόπο τοποθετούμε και τους απομονωτές εισόδου/εξόδου όπως φαίνονται στον **πίνακα Α.4**. Με την τελευταία ενέργεια προκύπτει η **εικόνα Α.22** για την επιφάνεια εργασίας.

1.1.2.3.4 Δημιουργία συνδέσεων

◆ Με απλό κλικ του ποντικιού επιλέγουμε το εργαλείο **Add Wire** (**εικόνα Α.23**) για τη δημιουργία συνδέσεων μεταξύ των υλικών στην επιφάνεια εργασίας. (**Πίνακας Α.5 με εικόνες Α.24α** ως **Α.24γ και πίνακας Α.6 με εικόνες Α.25α** ως **Α.25**γ).

Η παραπάνω ενέργεια σταματά πατώντας το πλήκτρο **Esc**.

1.1.2.3.4.1 Τοποθέτηση ονομάτων συνδέσεων

◆ Με απλό κλικ του ποντικιού επιλέγουμε το εργαλείο **Add Net Name** (**εικόνα Α.26**) και ονομάζουμε τις συνδέσεις μεταξύ των υλικών στην επιφάνεια εργασίας. Διαδικασία εξαιρετικά βοηθητική σε πολλούς κόμβους συνδέσεων όπως φαίνεται στις **εικόνες Α.27α** ως **Α.27**γ του **πίνακα Α.7** για τον κόμβο **S_2**. Με την τελευταία ενέργεια προκύπτει η **εικόνα Α.28** για την επιφάνεια εργασίας.

Η παραπάνω ενέργεια σταματά πατώντας το πλήκτρο **Esc**.

1.1.2.3.5 Τοποθέτηση I/O markers

◆ Επιλέγουμε με απλό κλικ του ποντικιού το εργαλείο **Add I/O Marker** (**εικόνα Α.29**), με νέο κλικ του ποντικιού τοποθετούμε το εργαλείο στις εισόδους/εξόδους αποδίδοντας επίσης το όνομα της επιλογής μας για την συγκεκριμένη είσοδο/έξοδο. (**Πίνακες Α.8 και Α.9 με εικόνες Α.30α** ως **Α.30δ** και **Α.31α** ως

Α.31β αντίστοιχα). Με την τελευταία ενέργεια προκύπτει η **εικόνα Α.32** για την επιφάνεια εργασίας.Η παραπάνω ενέργεια σταματά πατώντας το πλήκτρο **Esc**.

1.1.2.3.6 Αποθήκευση και έλεγχος λαθών

- Με απλό κλικ του ποντικιού επιλέγουμε το εργαλείο **Save** αποθηκεύοντας έτσι όλη την προηγούμενη δουλειά στην επιφάνεια εργασίας του σχηματικού. (**Εικόνα Α.33**)
- Με απλό κλικ του ποντικιού επιλέγουμε το εργαλείο **Dock this window** (**εικόνα Α.34**) όπου έχουμε μετάβαση του σχηματικού στο ίδιο παραθυρικό περιβάλλον (**εικόνα Α.35**) με την υπόλοιπη συνολική εργασία (project). Μεγιστοποιούμε το ελαχιστοποιημένο παράθυρο του σχηματικού (**εικόνα Α.36**). Με την τελευταία ενέργεια προκύπτει η **εικόνα Α.37** για την επιφάνεια εργασίας.
- Με απλό κλικ του ποντικιού επιλέγουμε το εργαλείο **Check Schematic** όπου ελέγχουμε το σχηματικό μας για πιθανά λάθη. (**Εικόνα Α.38**) Το σχηματικό λοιπόν δεν έχει λάθη. (**Εικόνα Α.39**)

1.1.2.3.7 Δημιουργία σχηματικού συμβόλου

- Με απλό κλικ του ποντικιού επιλέγουμε την ετικέτα **Source** (**πίνακας Α.10, εικόνα Α.40α**), κατόπιν παρομοίως την ετικέτα **Processes** (**πίνακας Α.10, εικόνα Α. 40β**) όπως και στο «**+**» δίπλα από το εικονίδιο **Design Utilities** για να φανούν τα περιεχόμενά του (**πίνακας Α.10,εικόνα Α.40γ**). Με διπλό κλικ του ποντικιού στο εργαλείο **Create Schematic Symbol** για ενεργοποίηση της εφαρμογής (**πίνακας Α.10, εικόνα Α.40δ**) που καταλήγει στη δημιουργία του σχηματικού συμβόλου. Ο έλεγχος για τη δημιουργία του σχηματικού σύμβολου δηλώνεται από το μήνυμα της ε**ικόνας Α.41**. Το σχηματικό σύμβολο αποθηκεύεται στην ίδια διαδρομή με το ίδιο όνομα με τα αντίστοιχα της συνολικής εργασίας (project). Σύντομα θα τα αποκαλούμε λοιπόν **διαδρομή και όνομα συμβόλου**.

◆ Με απλό κλικ του ποντικιού επιλέγουμε **Symbols→διαδρομή συμβόλου→όνομα συμβόλου** και τοποθετούμε το σχηματικό σύμβολο στην επιφάνεια εργασίας του σχηματικού με τον τρόπο των **εικόνων Α.42α** ως **Α.42στ** του **πίνακα Α.11**.

1.1.2.4 Προσομοίωση συμπεριφοράς (με ISE Simulator)

1.1.2.4.1 Ορισμός προσομοιωτή

◆ Θυμίζουμε ότι κατά τη δημιουργία του νέου project στην περιοχή **Simulator** επιλέξαμε **ISE Simulator (VHDL/Verilog)**. (**Εικόνα Α.6**). **Διαφορετικά** με **δεξί** κλικ του ποντικιού επιλέγουμε στην ετικέτα **Sources** το ολοκληρωμένο (**xc9572-15PC44**). (**Πίνακας Α.12, εικόνα Α.43α**). Μετά με νέο κλικ του ποντικιού επιλέγουμε **Properties**. (**Πίνακας Α.12, εικόνα Α.43β**) Με το άνοιγμα του παράθυρου **Project Properties** στην περιοχή **Simulator:** επιλέγουμε **ISE Simulator (VHDL/Verilog)**. (**Πίνακας Α.12, εικόνα Α.43γ**). Οπότε επιβεβαιώνουμε με νέο κλικ του ποντικιού στο κουμπί **OK**. (**Πίνακας Α.12, εικόνα Α.43δ**).

1.1.2.4.2 Κυματομορφή ελέγχου

1.1.2.4.2.1 Δημιουργία αρχείου κυματομορφής

◆ Με κλικ του ποντικιού στην περιοχή **Sources for:** του παραθύρου **Sources** επιλέγουμε **Behavioral Simulation** (**πίνακας Α.13**, **εικόνα Α.44α**). Με **δεξί** κλικ του ποντικιού επιλέγουμε το ολοκληρωμένο **xc9572-15PC44** (**πίνακας Α.13, εικόνα Α.44β**). Μετά με κλικ του ποντικιού επιλέγουμε **New Source…** (**Πίνακας Α.13**, **εικόνα Α.44γ**). Από το νέο παράθυρο **New Source Wizard – Select Source Type** που προκύπτει με κλικ του ποντικιού ξαναεπιλέγουμε **Test Bench Waveform** (**πίνακας Α.13**, **εικόνα Α.44δ**). Πληκτρολογούμε **mux_tbw** (όνομα κυματομορφής ελέγχου) και ξανά με κλικ του ποντικιού επιλέγουμε **Next**. (**Πίνακας Α.13**, **εικόνα Α.44ε**). Στο νέο παράθυρο **New Source Wizard – Associate Source** επαναλαμβάνουμε την επιλογή **Next** (**πίνακας Α.13**, **εικόνα Α.44στ**).

Οπότε στο παράθυρο **New Source Wizard – Summary** με κλικ του ποντικιού επιλέγουμε **Finish** (**πίνακας Α.13**, **εικόνα Α.44ζ**).

1.1.2.4.2.2 Χρονική αρχικοποίηση

- Στο παράθυρο **Initial Timing and Clock Wizard – Initialize Timing** με κλικ του ποντικιού επιλέγουμε διαδοχικά **Combinatorial (or internal clock)** (**πίνακας Α.14**, **εικόνα Α.45α**), αποεπιλέγουμε **GSR (FPGA)** (**πίνακας Α.14**, **εικόνα Α.45β**), επιλέγουμε **PRLD (CPLD)** (**εικόνα Α.45γ**), πληκτρολόγηση χρονικής τιμής στο πλαίσιο της περιοχής **Initial Length of Test Bench** (**πίνακας Α.14, εικόνα Α.45δ**) και τελικά επιλέγουμε **Finish** (**πίνακας Α.14, εικόνα Α.45ε**).

1.1.2.4.2.3 Ρύθμιση κυματομορφής

- Με κλικ του ποντικιού επιλέγουμε το εργαλείο **Zoom In** μεγεθύνοντας την επιφάνεια εργασίας της κυματομορφής ελέγχου που προκύπτει ώστε να είναι καλά ορατή (**πίνακας Α.15**, **εικόνες Α.46α** και **Α.46β**). Όπως επίσης φαίνεται (**πίνακας Α.16**, **εικόνες Α.47α** ως **Α.47γ**) με κλικ του ποντικιού επιλέγουμε τα χρονικά σημεία αλλαγής λογικής στάθμης της κυματομορφής ελέγχου για κατοπινό έλεγχο ανταπόκρισης της σχεδίασης στις ανάγκες μας. Εφόσον τελειώσουμε με τα σημεία αυτά αποθηκεύουμε τις επιλογές μας με κλικ του ποντικιού στο εργαλείο **Save** (**εικόνα Α.48**).

1.1.2.4.2.4 Ενεργοποίηση προσομοίωσης

- Η επιλογή με απλό κλικ του ποντικιού στο «+» δίπλα στο αρχείο κυματομορφής κατόπιν με κλικ του ποντικιού επιλέγουμε το όνομα του αρχείου κυματομορφής (**mux_tbw, πίνακας Α.17** και **εικόνα Α.49α**). Με κλικ του ποντικιού επιλέγουμε την ετικέτα **Processes**. (**Πίνακας Α.17**, **εικόνα Α.49β**). Η επιλογή με κλικ του ποντικιού στο «+» δίπλα στο εικονίδιο **Xilinx ISE Simulator**. (**Πίνακας Α.17**, **εικόνα Α.49γ**). Στην **εικόνα Α.49δ** του **πίνακα Α.17** φαίνεται η

επιλογή **Simulate Behavioral Model** όπου επιλέγεται με **διπλό** κλικ του ποντικιού και ξεκινά η εφαρμογή της προσομοίωσης σε νέο παράθυρο καταλήγοντας στην προσομοίωση της ε**ικόνας Α.50**.

1.1.2.4.2.5 Ενεργοποιημένη προσομοίωση

◆ Η επιλογή με απλό κλικ του ποντικιού στο εργαλείο για επανεκκίνηση της προσομοίωσης **Restart Simulation**. (**Πίνακας Α.18**, **εικόνα Α.51α**). Ξαναεπιλέγουμε με απλό κλικ του ποντικιού το εργαλείο για ενεργοποίηση της προσομοίωσης για επιλεγμένο χρονικό διάστημα (εδώ 10.000 nsec) **Run For Specific Time**. (**Πίνακας Α.18**, **εικόνα Α.51β**). Στο νέο παράθυρο (**εικόνα Α.52**) ενεργοποιείται προσομοίωση για το νέο χρονικό διάστημα. Στο ίδιο παράθυρο επιλέγουμε με κλικ του ποντικιού το εργαλείο **Float this window** για απομόνωση της εφαρμογής σε δικό της παράθυρο σε σχέση με την συνολική εργασία (project) που ήταν ενεργοποιημένη προηγουμένως. Η **εικόνα Α.53** δείχνει το απομονωμένο αυτό παράθυρο με την επιλογή της μελλοντικής μεγέθυνσής του όπως φαίνεται στην **εικόνα Α.54**. Η επιλογή με απλό κλικ του ποντικιού στο εργαλείο **Zoom In** (**εικόνα Α.55**) μεγεθύνει παραπάνω την κυματομορφή προσομοίωσης. Η επαναλαμβανόμενη χρήση του εργαλείου προκαλεί μεγέθυνση της επιλογής μας. (**Εικόνα Α.56**). Στην **εικόνα Α.57** επιλέγεται (με κλικ του ποντικιού) ένα χρονικό σημείο για έλεγχο της ορθότητας λογικών σταθμών σύμφωνα με τη σχεδίαση. Εφόσον μας ικανοποιεί το αποτέλεσμα του ελέγχου για όλα τα χρονικά σημεία της κυματομορφής προσομοίωσης γνωρίζουμε ότι το κύκλωμά μας ανταποκρίνεται ορθά. Τέλος στην **εικόνα Α.58** επιλέγουμε το εργαλείο **Dock this Window** για να επανέλθει το παράθυρο της προσομοίωσης στο αρχικό παραθυρικό περιβάλλον με τη συνολική εργασία (project). (**Εικόνα Α.59**)

1.1.2.5 Προγραμματισμός

1.1.2.5.1 Αντιστοίχιση ακίδων

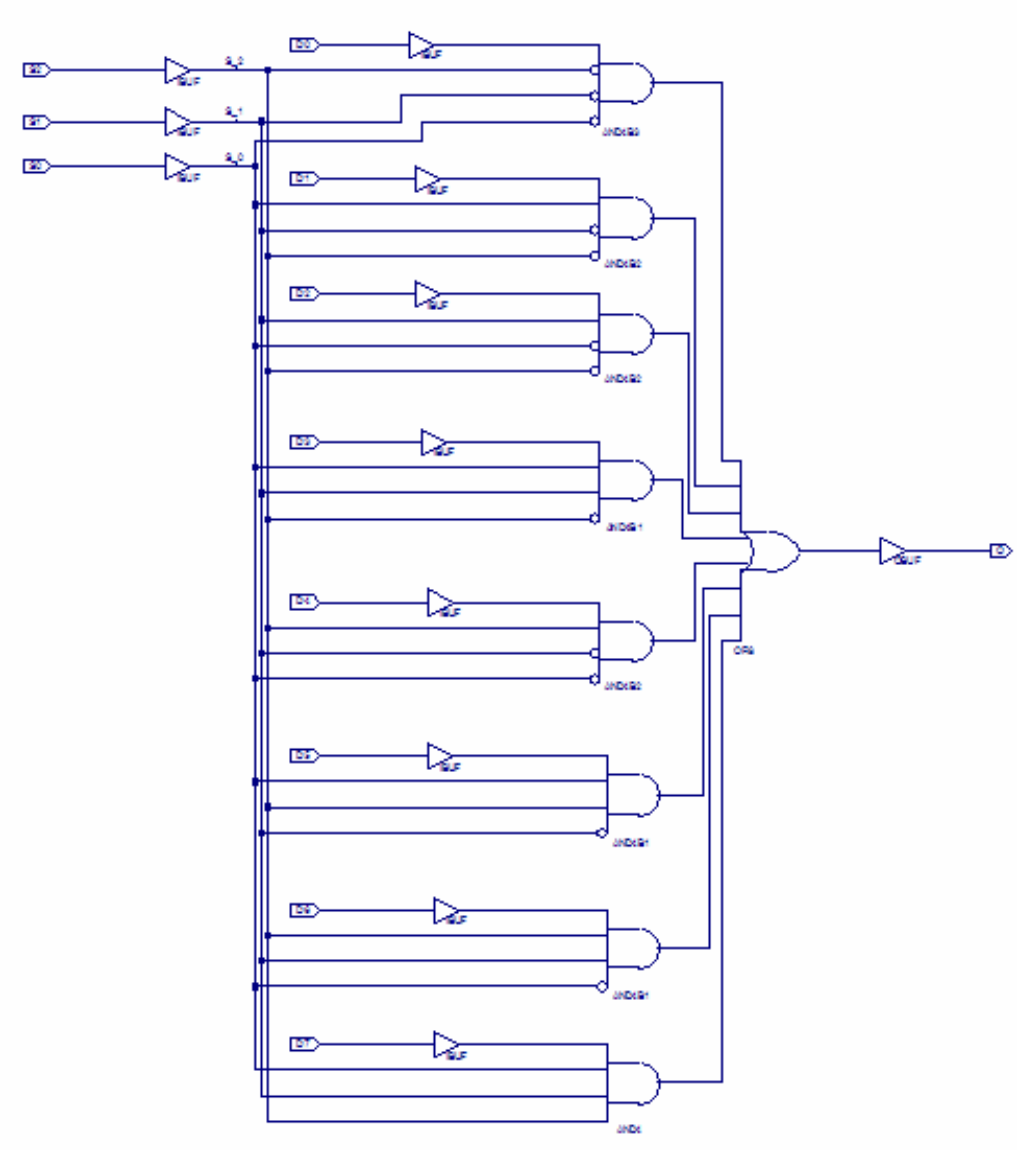
- Απαραίτητη προϋπόθεση για τη διαδικασία του προγραμματισμού αποτελεί η επιλογή του **Synthesis/Implementation** στην ετικέτα **Sources** (**εικόνα Α.60γ**). Σε διαφορετική περίπτωση εκκινούμε τη διαδικασία όπως φαίνεται στην εικόνα **Α.60α** και τελειώνει στην εικόνα **Α.60γ**.
- Κατά την ασφαλή αντιστοίχηση των εισόδων/εξόδων (I/O) με τα ποδαράκια ή ακίδες (pins) του ολοκληρωμένου χρειάζεται το αρχείο με κατάληξη ngd που παράγεται κατά την διαδικασία εκτέλεσης της εφαρμογής **Translate**. Για την ενεργοποίηση της παραπάνω διαδικασίας επιλέγουμε και «ανοίγουμε» το **Implement Design** στην ετικέτα **Processes**. Κατόπιν επιλέγουμε το **Translate** και με διπλό κλικ του ποντικιού ενεργοποιούμε την εφαρμογή όπως φαίνεται στις **εικόνες Α.61α** ως **Α.61γ**. Μετά την επιτυχημένη εκτέλεση (**εικόνα Α.62**) παράγεται το αρχείο με κατάληξη ngd που χρειαζόμαστε.
- «Κλείνουμε» το **Implement Design** όπως φαίνεται στις **εικόνες Α.63α** και **Α.63β** του **πίνακα Α.19**. Από το **User Constraints** επιλέγουμε την εφαρμογή **Assign Package Pins** και με διπλό κλικ του ποντικιού την ενεργοποιούμε (**εικόνες Α.63γ** ως **Α.63δ** του **πίνακα Α.19)**. Έτσι ξεκινά η εφαρμογή της αντιστοίχησης εισόδων/εξόδων και ακίδων με την παραγωγή του ορθού αρχείου κατάληξης ucf. Στην **εικόνα Α.64α** του **πίνακα Α.20** μας ζητείται από το πρόγραμμα η αυτόματη δημιουργία του αρχείου κατάληξης ucf. Μετά την επιλογή του **Yes** (ναι) της **εικόνας Α.64β** του **πίνακα Α.20**, το πρόγραμμα (Project Navigator) κάνει εισαγωγή στο καινούργιο παραθυρικό περιβάλλον (**πίνακας Α.20**, **εικόνα Α.64γ**) που το βλέπουμε μεγενθυμένο στην **εικόνα Α.64δ** του **πίνακα Α.20**. Με μια πιο προσεχτική ματιά στην τελευταία εικόνα παρατηρούμε ότι το παραθυρικό της περιβάλλον δεν αντιστοιχεί στην συνολική εργασία μας (project) επομένως η αντιστοίχηση εισόδων/εξόδων και ακίδων δεν θα είναι ορθή.

- Επιλέγουμε **File→Close** (κλικ ποντικιού) για να κλείσουμε το ucf αρχείο με την λανθασμένη αντιστοίχηση (**εικόνες Α.65α** ως **Α.65β**). Παρομοίως επιλέγουμε το εικονίδιο **Open** (**εικόνα Α.65γ** ή εναλλακτική διαδρομή **File→ Open**) οπότε προκύπτει το παράθυρο της **εικόνας Α.65δ**. Στις **εικόνες Α.65ε** ως **Α.65θ** εισάγουμε το ορθό ucf αρχείο σύμφωνα με τη συνολική εργασία μας (project). Παρομοίως στις **εικόνες Α.66α** ως **Α.66δ** εισάγουμε το ορθό ngd αρχείο όπως προέκυψε από την παραπάνω εκτέλεση της εφαρμογής **Translate** σύμφωνα με τη συνολική εργασία μας (project). Συμπληρώνοντας τα στοιχεία του ολοκληρωμένου που χρησιμοποιούμε (**εικόνες Α.67β** ως **Α.67ε**) και μετά την επιλογή **OK** (**εικόνα Α.68στ**) εισερχόμαστε στο παραθυρικό περιβάλλον της **εικόνας Α.68**. όπου μπορούμε να προχωρήσουμε στην υλοποίηση της ορθής αντιστοίχησης. Οι **εικόνες Α.69α** ως **Α.69γ** αντιστοιχίζουν την είσοδο **S2** στο ποδαράκι ή ακίδα (pin) του ολοκληρωμένου με αριθμό **43** ή **P43**, επίσης δείχνουν τον τρόπο που γίνεται η αντιστοιχία αυτή σε ό,τι αναφορά τη χρήση του ποντικιού. Οι **εικόνες Α.71α** ως **Α.71δ** μας δείχνουν το παραθυρικό περιβάλλον μετά την ολοκλήρωση της ορθής αντιστοίχησης εισόδων/εξόδων και ακίδων όπως και τον τρόπο που αποχωρούμε από το περιβάλλον αυτό.

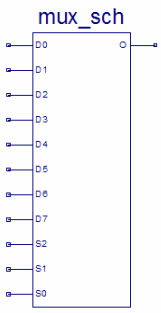
1.1.2.5.2 Γρήγορος προγραμματισμός

- Στις **εικόνες Α.72α** ως **Α.72γ** φαίνεται η ενεργοποίηση όλων των εφαρμογών του **Implement Design** με το επιτυχημένο τέλος αυτών. Στις **εικόνες Α.73α** ως **Α.73β** φαίνεται η ενεργοποίηση της εφαρμογής **Configure Device (iMPACT).** Ομοίως στις **εικόνες Α.73δ** ως **Α.73στ** φαίνεται το άνοιγμα του κατάλληλου **jed** αρχείου που παράχθηκε κατά την υποεφαρμογή **Fit** του **Implement Design**. Στις **εικόνες Α.74α** ως **Α.74ε** ακολουθεί ο γρήγορος προγραμματισμός. Η επιτυχία του φαίνεται στις **εικόνες Α.75α** ως **Α.75β**. (Συνολικό τέλος ενεργειών.)

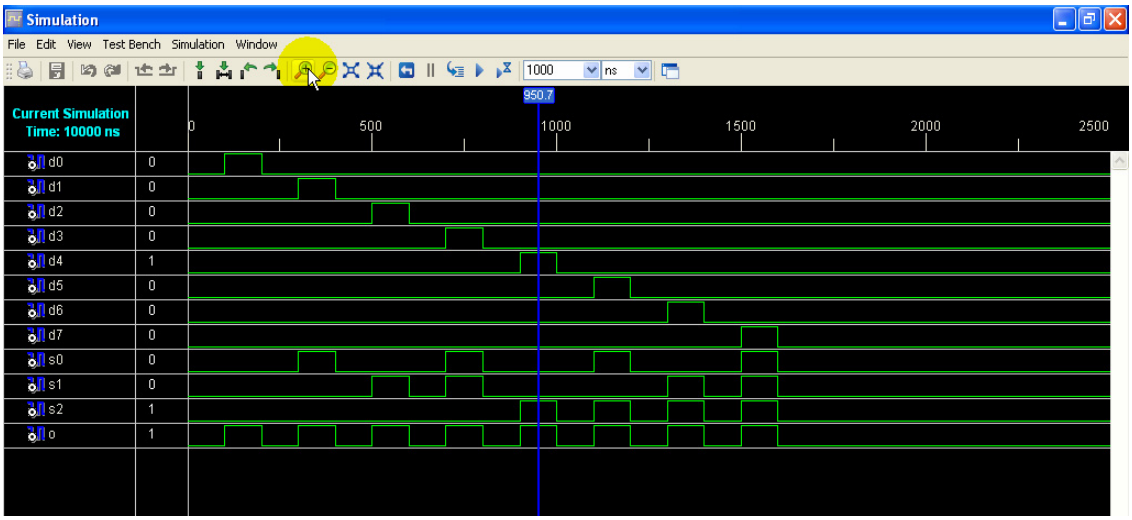
1.1.3 ΑΠΟΤΕΛΕΣΜΑΤΑ ΑΝΑΠΤΥΞΗΣ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ



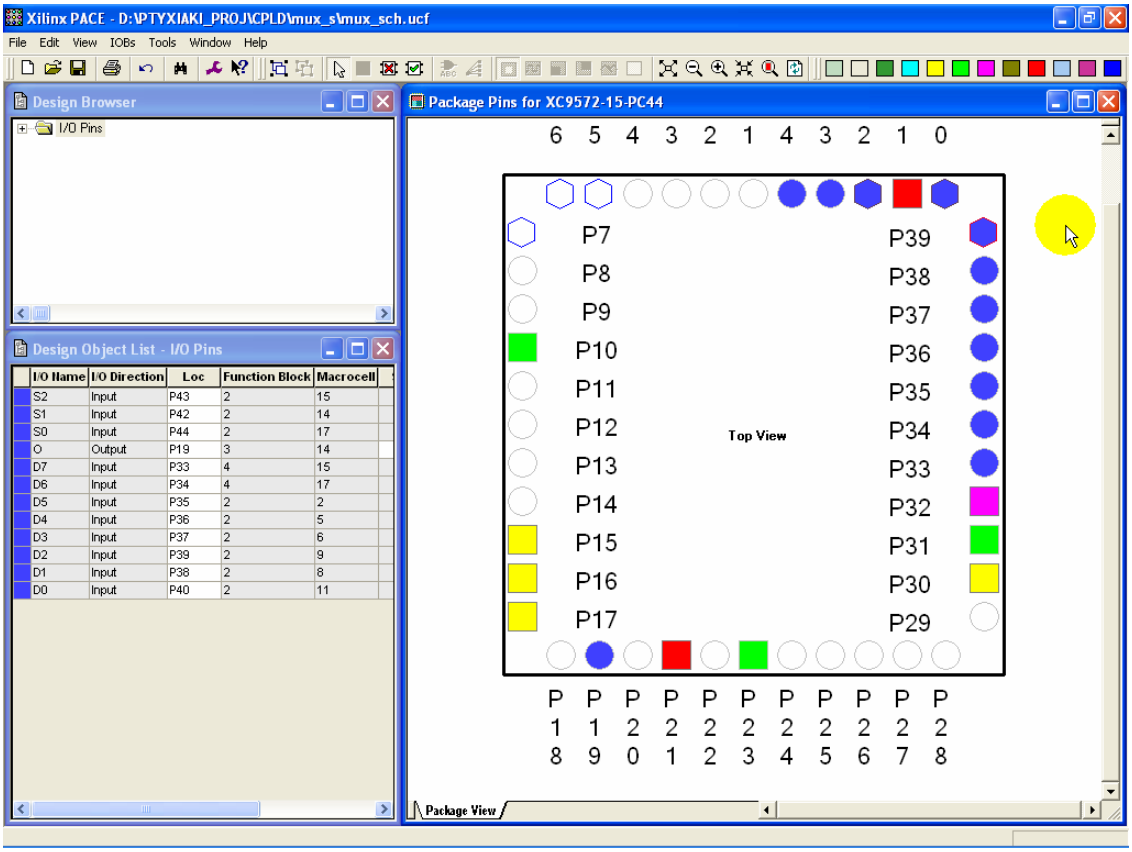
Εικόνα 1.4 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 1.5 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.6 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.7 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

1.1.4 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Πίνακας 1.2 Περιοχή δήλωσης βιβλιοθηκών (library)

01	----------	Σχόλια: οτιδήποτε έπεται μετά από το σύμβολο «--», έχουν χαρακτηριστικό πράσινο χρώμα.
02	library IEEE;	Δήλωση βιβλιοθήκης με όνομα **ieee.**
03	use IEEE.STD_LOGIC_1164.ALL;	Δήλωση χρήσης όλων των στοιχείων του πακέτου **std_logic_1164** της βιβλιοθήκης **ieee**, απαραίτητο πακέτο για χρήση δεδομένων **std_logic** και **std_logic_vector**.
04	use IEEE.STD_LOGIC_ARITH.ALL;	Δήλωση χρήσης όλων των στοιχείων του πακέτου **std_logic_arith** της βιβλιοθήκης **ieee**. **Δεν χρειάζεται** για τη συγκεκριμένη εργασία.
05	use IEEE.STD_LOGIC_UNSIGNED.ALL;	Δήλωση χρήσης όλων των στοιχείων του πακέτου **std_logic_unsigned** της βιβλιοθήκης **ieee**. **Δεν χρειάζεται** για τη συγκεκριμένη εργασία.
06	----------	Σχόλια: οτιδήποτε έπεται μετά από το σύμβολο «--», έχουν χαρακτηριστικό πράσινο χρώμα.

Σημείωση. Οι εντολές των γραμμών 02, 03, 04, 05 βρίσκονται εξ' ορισμού από το πρόγραμμα σε όλες τις συνολικές εργασίες (projects).
Το ερωτηματικό δηλώνει το τέλος της εντολής.
Τα σχόλια στις γραμμές 01, 06 και 12 χρησιμοποιούνται για να ξεχωρίζουν τις περιοχές δήλωσης βιβλιοθηκών, οντότητας και αρχιτεκτονικής μεταξύ τους.

Πίνακας 1.3 Περιοχή δήλωσης οντότητας (entity)

07	entity mux_vhf1 is	Αρχή δήλωσης οντότητας με όνομα **mux_vhf1**.
08	Port (D : in STD_LOGIC_VECTOR (7 downto 0);	Αρχή δήλωσης θυρών (**ports**) με την πρώτη θύρα να δηλώνεται με όνομα **D** σαν είσοδος (**in**) σε μορφή **std_logic_vector** 8 δυαδικών ψηφίων.
09	S : in STD_LOGIC_VECTOR (2 downto 0);	Άλλη θύρα να δηλώνεται με όνομα **S** σαν είσοδο (**in**) διανύσματος (**vector**) σε μορφή **std_logic_vector**

```
10              O : out  STD_LOGIC);
11      end mux_vhf1;
```

3 δυαδικών ψηφίων. Η τελευταία θύρα δηλώνεται με όνομα **O** σαν έξοδος (**out**) σε μορφή **std_logic**. Τέλος δήλωσης εισόδων/εξόδων. Τέλος δήλωσης οντότητας με όνομα **mux_vhf1**.

Σημείωση. Στις γραμμές 07 και 11 το όνομα οντότητας δεν μπορεί να ταυτίζεται με δεσμευμένες λέξεις από την VHDL όπως οι εντολές της ή οι χρωματισμένες λέξεις των παραπάνω γραμμών. Παραπάνω με χρήση της λέξης θύρα (**port**) περιγράφονται οι αρύθμιστες είσοδοι/έξοδοι. Αργότερα ρυθμίζονται ανάλογα σε εισόδους (in) ή εξόδους (out).

Πίνακας 1.4 Καταστάσεις τύπου δεδομένων STD_LOGIC

ΤΙΜΕΣ	ΕΞΗΓΗΣΗ	ΤΙΜΕΣ ΣΥΝΘΕΣΗΣ (XST)
'U'	αρχικά μη ανιχνεύσιμη	μη αποδεκτή
'X'	άγνωστη	αδιάφορη
'0'	ισχυρό λογικό 0	ισχυρό λογικό 0
'1'	ισχυρό λογικό 1	ισχυρό λογικό 1
'Z'	high-Z ή υψηλής εμπέδησης	high-Z ή υψηλής εμπέδησης
'W'	ασθενής άγνωστη	μη αποδεκτή
'L'	ασθενές λογικό 0	ισχυρό λογικό 0
'H'	ασθενές λογικό 1	ισχυρό λογικό 1
'-'	αδιάφορη	αδιάφορη

Σημείωση. Όλες οι παραπάνω τιμές είναι προσομοιώσιμες δηλαδή μπορούν να φανούν όπως αναγράφονται παραπάνω στο παράθυρο προσομοίωσης.

Πίνακας 1.5 Παράδειγμα τύπου δεδομένων STD_LOGIC_VECTOR

Ανάθεση δυαδικής τιμής 10001110 στο διάνυσμα **D**								
Δήλωση διανύσματος **D**	Κατάσταση διανύσματος **D**							
STD_LOGIC_VECTOR (7 downto 0)	**M.S.B.**							**L.S.B.**
	D(7)	**D**(6)	**D**(5)	**D**(4)	**D**(3)	**D**(2)	**D**(1)	**D**(0)
	1	0	0	0	1	1	1	0
STD_LOGIC_VECTOR (0 to 7)	**M.S.B.**							**L.S.B.**
	D(0)	**D**(1)	**D**(2)	**D**(3)	**D**(4)	**D**(5)	**D**(6)	**D**(7)
	1	0	0	0	1	1	1	0

Σημείωση. M.S.B.: Περισσότερο δυαδικό σημαντικό ψηφίο.
L.S.B.: Λιγότερο δυαδικό σημαντικό ψηφίο.
Στην VHDL οποιοσδήποτε συνδυασμός γραμμάτων μικρών ή κεφαλαίων σε λέξη είναι αποδεκτός. ((0 TO 7) ή (0 to 7) ή (0 To 7) ή (0 tO 7) για την VHDL είναι το ίδιο ακριβώς).

Πίνακας 1.6 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

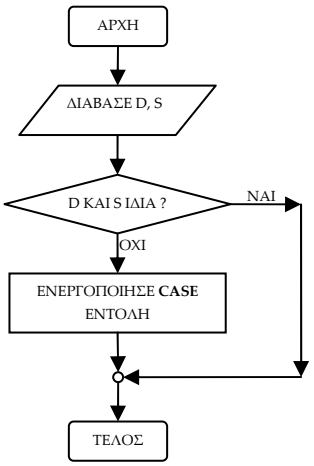
13	architecture BHV of mux_vhf1 is	Αρχή δήλωσης **αρχιτεκτονικής** με όνομα **BHV** για την οντότητα με όνομα **mux_vhf1** όπως παραπάνω.
14	begin	Αρχή κώδικα αρχιτεκτονικής με το begin.
15	process (D, S)	Αρχή της διεργασίας με λίστα ευαισθησίας τις εισόδους **D** και **S** εντός της παρενθέσεως.
16	begin	Αρχή ενεργοποίησης της διεργασίας δηλώνεται με τη λέξη begin και πραγματοποιείται αν αλλάξει η λογική κατάσταση ενός των **D** και **S** της λίστας ευαισθησίας. Με την ενεργοποίηση της διεργασίας εκτελούνται όλες οι εντολές που βρίσκονται εντός της ακολουθιακά.
17	case S is	Ακολουθιακή εντολή **case** εντός της διεργασίας για το **S**.
18	when "000" =>	Κατά την εκτέλεση της **case** γίνεται έλεγχος για το αν η είσοδος **S** έχει πάρει την δυαδική τιμή 000. Το αποτέλεσμα του ελέγχου παίρνει λογική τιμή true (αληθής) ή false (ψευδής). Αν ισχύει η λογική τιμή true για την γραμμή 18 εκτελείται η επόμενη γραμμή 19 μετά το σύμβολο "=>", διαφορετικά το πρόγραμμα μεταβαίνει στην γραμμή 20.
19	O <= D(0);	Εκτελείται με το σύμβολο "<=" η ανάθεση της δυαδικής τιμής του bit 0 του διανύσματος **D** ή αλλιώς **D**(0) στην έξοδο O. Αμέσως μετά το πρόγραμμα μεταβαίνει στην γραμμή 34.

20	when "001" =>	Ισχύει ότι στην γραμμή 18.
21	O <= D(1);	Ισχύει ότι στην γραμμή 19 αντίστοιχα για το bit 1 του διανύσματος **D** στην έξοδο O.
22	when "010" =>	Ισχύει ότι στην γραμμή 18.
23	O <= D(2);	Ισχύει ότι στην γραμμή 19 αντίστοιχα για το bit 2 του διανύσματος **D** στην έξοδο O.
24	when "011" =>	Ισχύει ότι στην γραμμή 18.
25	O <= D(3);	Ισχύει ότι στην γραμμή 19 αντίστοιχα για το bit 3 του διανύσματος **D** στην έξοδο O.
26	when "100" =>	Ισχύει ότι στην γραμμή 18.
27	O <= D(4);	Ισχύει ότι στην γραμμή 19 αντίστοιχα για το bit 4 του διανύσματος **D** στην έξοδο O.
28	when "101" =>	Ισχύει ότι στην γραμμή 18.
29	O <= D(5);	Ισχύει ότι στην γραμμή 19 αντίστοιχα για το bit 5 του διανύσματος **D** στην έξοδο O.
30	when "110" =>	Ισχύει ότι στην γραμμή 18.
31	O <= D(6);	Ισχύει ότι στην γραμμή 19 αντίστοιχα για το bit 6 του διανύσματος **D** στην έξοδο O.
32	when others =>	Ισχύει ότι στην γραμμή 18 με τη διαφορά της χρήσης της λέξης-κλειδί others με το σύμβολο "=>" για να περιληφθούν όλοι οι υπόλοιποι 722 συνδυασμοί της εισόδου **S**.
33	O <= D(7);	Ισχύει ότι στην γραμμή 19 αντίστοιχα για το bit 7 του διανύσματος **D** στην έξοδο O.
34	end case;	Τέλος εκτέλεσης case.
35	end process;	Τέλος εκτέλεσης process.
36	end BHV;	Τέλος κώδικα αρχιτεκτονικής.

Σημείωση. Η εκτέλεση των εντολών της VHDL μέσα σε μια αρχιτεκτονική γίνεται με συντρέχοντα (concurrent) κώδικα οπότε μπορούν να εκτελούνται παράλληλα. Εξαίρεση αποτελεί η εκτέλεση των εντολών μέσα σε μια διεργασία (**process**). Εκεί η εκτέλεση είναι ακολουθιακή δηλαδή μπορεί να εκτελείται μόνο μια εντολή κάθε φορά. Όταν σταματά μια εντολή αρχίζει η εκτέλεση της επομένης των εντολών στη σειρά. Η διαδικασία εκτέλεσης των εντολών συνεχίζει μέχρι να εκτελεστούν όλες.

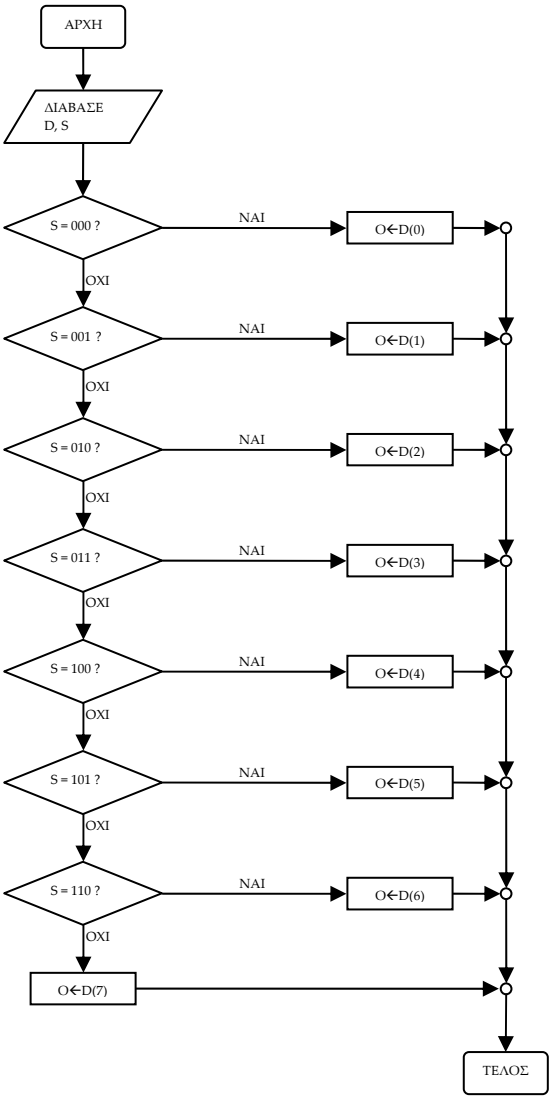
Πίνακας 1.7 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (D,S)
        begin

                case … end case;

end process;
```



Πίνακας 1.8 Η εντολή **case** και το **διάγραμμα ροής**

```
case S is
        when "000" =>
                        O <= D(0);
        when "001" =>
                        O <= D(1);
        when "010" =>
                        O <= D(2);
        when "011" =>
                        O <= D(3);
        when "100" =>
                        O <= D(4);
        when "101" =>
                        O <= D(5);
        when "110" =>
                        O <= D(6);
        when others =>
                        O <= D(7);
end case;
```



Όλα τα παραπάνω που αφορούν τον ακολουθιακό τρόπο λειτουργίας της διεργασίας (**process**) με την περιεχομένη σ' αυτήν εντολή **case** εκφράζονται ποιοτικά από τα παραπάνω διαγράμματα ροής (**flow charts**). Στα διαγράμματα αυτά η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση και του συμβόλου «←» εκφράζει μεταφορά τιμής.

1.1.5 ΑΝΑΠΤΥΞΗ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Οι **εικόνες** (**Α.76** ως **Α.100β**) βρίσκονται στο παράρτημα Α.

1.1.5.1 Έναρξη εφαρμογής

- Με **διπλό** χτύπημα (κλικ) του ποντικιού πάνω στην συντόμευση της επιφάνειας εργασίας (**εικόνα Π.76)** ξεκινά η εφαρμογή **Project Navigator** (**εικόνα Α.77**). Άλλη μία εναλλακτική διαδρομή έναρξης της εφαρμογής για τα Windows XP είναι: **Start→Programs→Xilinx ISE 9.1i→Project Navigator (αγγλική έκδοση) ή Έναρξη→Όλα τα προγράμματα→Xilinx ISE 9.1i→Project Navigator (ελληνική έκδοση).**

Η εφαρμογή **Project Navigator** μας εισάγει στο αναπτυξιακό παραθυρικό περιβάλλον του προγράμματος για συνολικές εργασίες (projects). (**Εικόνα Α.78**)

1.1.5.2 Δημιουργία νέας συνολικής εργασίας

- Στο νέο περιβάλλον επιλέγουμε την διαδρομή: **File→New Project.** (**Εικόνα Α.79**)
- Με το άνοιγμα του παραθύρου **New Project Wizard – Create New Project**

στην περιοχή **Project Name:** πληκτρολογούμε **mux_v1**,

στην περιοχή **Project Location:** έχουμε αυτόματη δημιουργία του φάκελου **mux_v1** στην προεπιλεγμένη διαδρομή της επιλογής μας,

στην περιοχή **Top-Level Source Type:** επιλέγουμε **HDL** σαν την γλώσσα κορυφαίου επίπεδου ιεραρχίας όπου θα εργαστούμε.

Στη συνέχεια επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα Α.80**)

- Με το άνοιγμα του παράθυρου **New Project Wizard – Device Properties**

στην περιοχή **Device Family**: επιλέγουμε **XC9500 CPLDs**

στην περιοχή **Device:** επιλέγουμε **XC9572 (**ή **XC9536)**

στην περιοχή **Package:** επιλέγουμε **PC44**

στην περιοχή **Speed:** επιλέγουμε **-15**

στην περιοχή **Synthesis Tool:** επιλέγουμε **XST (VHDL/Verilog)**

στην περιοχή **Simulator:** επιλέγουμε **ISE Simulator (VHDL/Verilog).**

Στη συνέχεια επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα A.81**)

- Με το άνοιγμα του παράθυρου **New Project Wizard – Create New Source**

επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **New Source**. (**Εικόνα A.82**)

- Με το άνοιγμα του παράθυρου **New Project Wizard – Select Source Type**

με απλό κλικ του ποντικιού υπογραμμίζουμε το κουμπί **Add to Project**,

με απλό κλικ του ποντικιού επιλέγουμε **VHDL Module**,

στην περιοχή **File Name:** πληκτρολογούμε **mux_vhf1**.

Επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα A.83**)

- Με το άνοιγμα του παράθυρου **New Project Wizard – Define Module**

στις περιοχές:

Entity Name προϋπήρχε **mux_vhf1** (όπως είχε ονομαστεί),

Architecture Name πληκτρολογούμε **BHV** και

ειδικότερα για την είσοδο **D:**

Port Name πληκτρολογούμε **D** (ονομασία επιλογής μας),

Direction επιλέγουμε **in** (είσοδος),

Bus επιλέγουμε (τετράγωνο) **Bus**

MSB πληκτρολογούμε **7** (το bit 7 είναι το πιο σημαντικό).

Η παραπάνω διαδικασία για κάθε μία από τις εισόδους/εξόδους (ports) φαίνεται στις **εικόνες A.84α** ως **A.84ε**. Το παράθυρο **New Project Wizard – Define Module** είναι συμπληρωμένο με όλες τις εισόδους/εξόδους, οπότε επιλέγουμε με κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα A.85**) (Να σημειωθεί ότι τα όσα φαίνονται στις **εικόνες A.84α** ως **A.84ε** και **A.85** είναι προαιρετικά. Χωρίς την συμπλήρωση των πεδίων αυτών έχουμε το άνοιγμα του αρχείου VHDL χωρίς εισόδους/εξόδους (ports). Οι είσοδοι/έξοδοι (ports) μπορούν λοιπόν να δηλωθούν με κατοπινή πληκτρολόγησή τους στο αρχείο VHDL.)

- Με το άνοιγμα του παράθυρου **New Project Wizard – Summary** επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Finish**. (**Εικόνα Α.86**)

Στην ερώτηση για δημιουργία του ανύπαρκτου (προς το παρόν) καταλόγου με το όνομα που δώσαμε προηγουμένως, επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Yes**. (**Εικόνα Α.87**)

- Με το άνοιγμα του παράθυρου **New Project Wizard – Create New Source** επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα Α.88**)
- Με το άνοιγμα του παράθυρου **New Project Wizard – Add Existing Sources** επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Next**. (**Εικόνα Α.89**)
- Με το άνοιγμα του παράθυρου **New Project Wizard – Project Summary** επιλέγουμε με απλό κλικ του ποντικιού το κουμπί **Finish**. (**Εικόνα Α.90**) Τέλος της εισαγωγής στοιχείων της νέας συνολικής εργασίας (project).

1.1.5.3 Εργασία με τη VHDL

- Στη συνέχεια εισερχόμαστε στο περιβάλλον ανάπτυξης εργασιών (project) στην VHDL (περιγραφική γλώσσα). (**Εικόνα Α.91**)

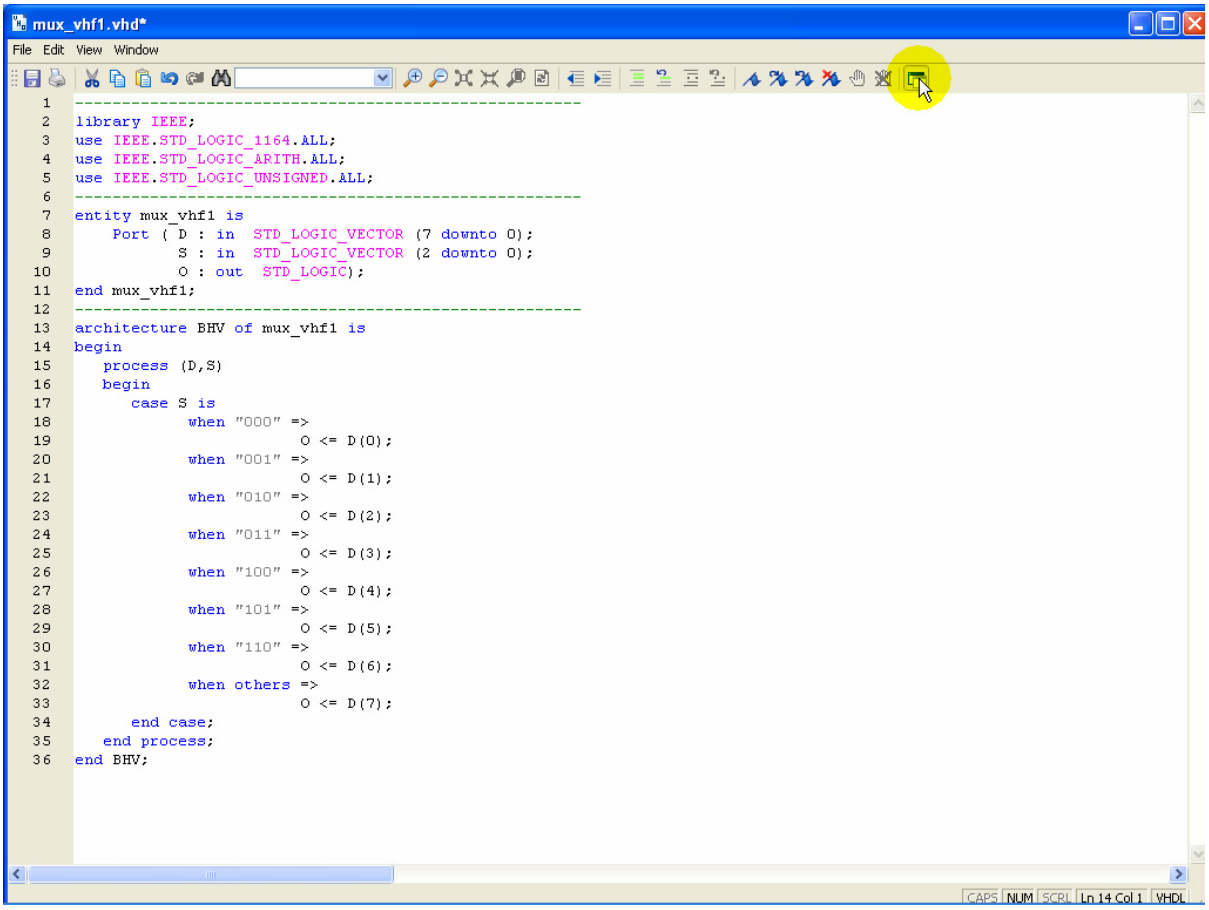
Για τη μεγιστοποίηση της επιφάνειας του περιβάλλοντος επιλέγουμε με απλό κλικ του ποντικιού το εικονίδιο **Float this window** από τη μπάρα εργαλείων. (**Εικόνα Α.92**)

Προκύπτοντας η μεγιστοποιημένη επιφάνεια εργασίας της VHDL με το δικό της παράθυρο. (**Εικόνα Α.93**)

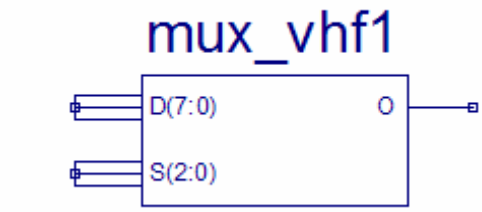
- Στην συνέχεια στο παράθυρο της μεγιστοποιημένης επιφάνειας πληκτρολογούμε τις απαραίτητες εντολές όπως φαίνεται στην **εικόνα Α.94**.
- Με απλό κλικ του ποντικιού επιλέγουμε το εργαλείο **Dock this window** (**εικόνα Α.95**) όπου έχουμε μετάβαση της VHDL στο ίδιο παραθυρικό περιβάλλον (**εικόνα Α.96**) με την υπόλοιπη συνολική εργασία (project). Μεγιστοποιούμε το ελαχιστοποιημένο παράθυρο της VHDL. (**Εικόνα Α.97**)

- Με απλό κλικ του ποντικιού (μετά τη μεγιστοποίηση του παραθύρου) επιλέγουμε το εργαλείο **Save** αποθηκεύοντας έτσι όλη την προηγούμενη δουλειά στην επιφάνεια εργασίας της VHDL. (**Εικόνα Α.98**)
- Με απλό κλικ του ποντικιού επιλέγουμε γραφικά τη διαδρομή **Implement Design→Synthesize-XST→Check Syntax** οπότε με διπλό κλικ στο εργαλείο **Check Syntax** ενεργοποιείται η εφαρμογή για έλεγχο των πιθανών συντακτικών λαθών της γλώσσας. (**Πίνακας Α.21**, **εικόνες Α.99α ως Α.99γ**) Το VHDL αρχείο λοιπόν δεν έχει λάθη. (**Εικόνες Α.100α ως Α.100β**)
- Από εδώ και κάτω όλες οι διαδικασίες στην ανάπτυξη της συνολικής εργασίας (project) με την περιγραφική γλώσσα VHDL ταυτίζονται με τις αντίστοιχες της ενότητας για την ανάπτυξη της συνολικής εργασίας (project) με την σχηματική γλώσσα. Καλείται λοιπόν ο αναγνώστης να επαναλάβει τις διαδικασίες (από τη δημιουργία σχηματικού συμβόλου μέχρι την τελική διαδικασία του γρήγορου προγραμματισμού) όπως γράφτηκαν στην ενότητα για την ανάπτυξη της συνολικής εργασίας (project) με την σχηματική γλώσσα στο παρόν σύγγραμμα.

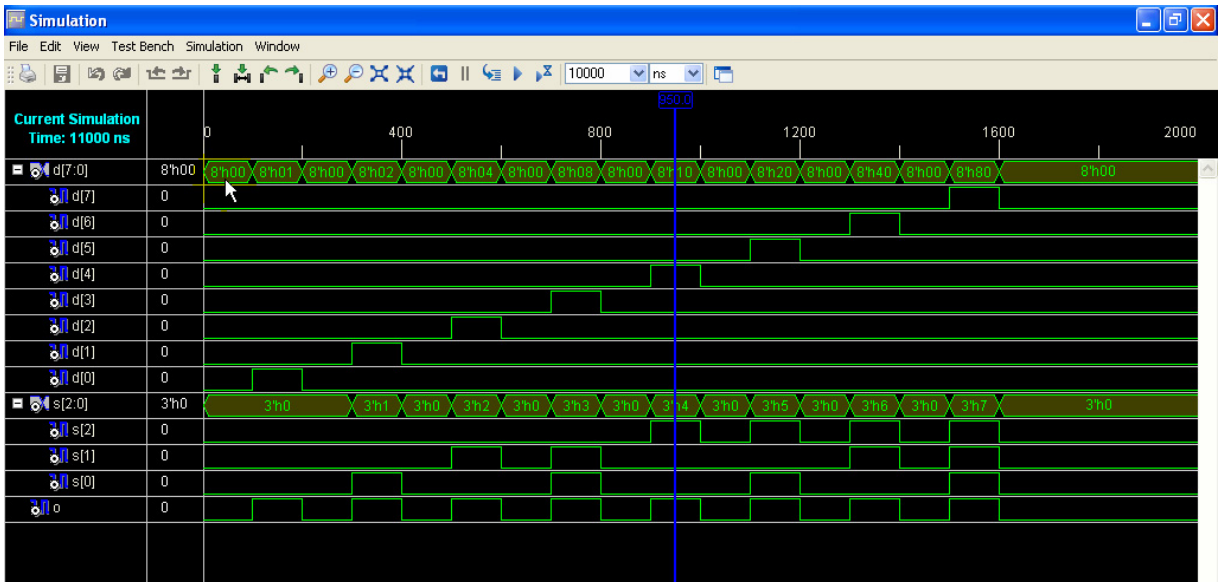
1.1.6 ΑΠΟΤΕΛΕΣΜΑΤΑ ΑΝΑΠΤΥΞΗΣ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)



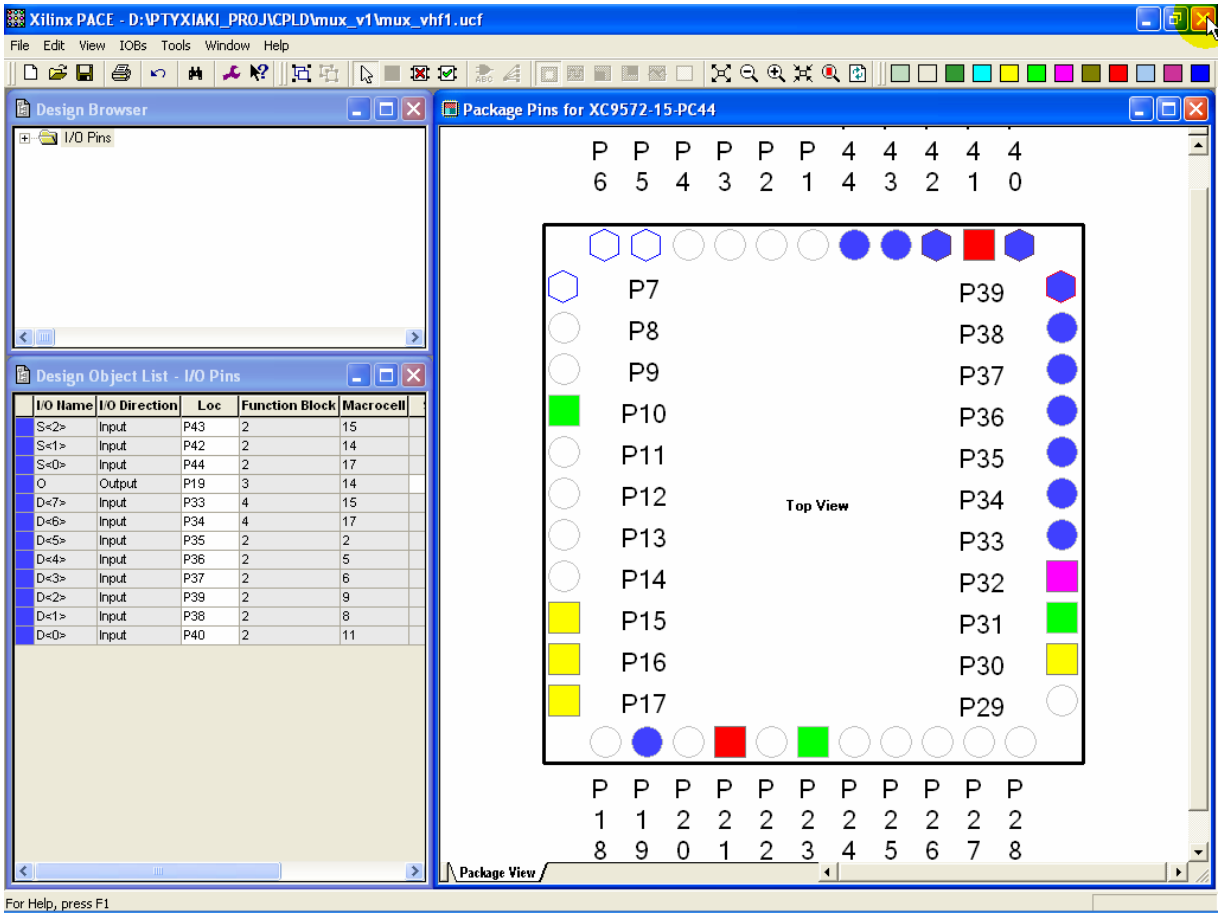
Εικόνα 1.8 VHDL αρχείο του κυκλώματος



Εικόνα 1.9 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.10 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.11 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

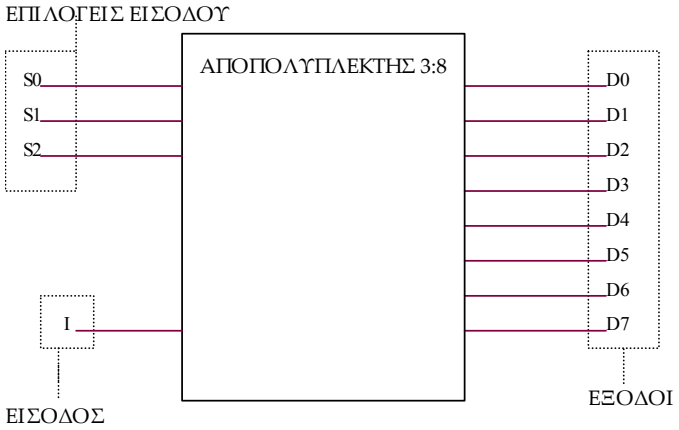
1.2 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ: ΑΠΟΠΟΛΥΠΛΕΚΤΗΣ 3:8

1.2.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

1.2.1.1 Σύντομη θεωρία

Ο αποπολυπλέκτης (*demu*ltiplexer, *demux*) ή διανομέας δεδομένων κάνει το αντίστροφο της λειτουργίας της πολύπλεξης. Αποτελεί εκείνο το συνδυαστικό ψηφιακό κύκλωμα που λαμβάνει τα δεδομένα από μία είσοδο και τα διανέμει σε μία από πεπερασμένο αριθμό εξόδων ενώ οι υπόλοιπες μηδενίζονται. Για τον γενικό αποπολυπλέκτη το σύμβολο $n:2^n$ διαβάζεται «n σε 2^n» ή «2^n από n».

Συγκεκριμένα για τον αποπολυπλέκτη 3:8 μία λεωφόρος n=3 bits δεδομένων επιλογής εισόδου μπορεί να επιλέγει σε ποιο από τα 2^3=8 bits της εξόδου μπορεί να μεταφερθεί το bit της εισόδου. Τα δεδομένα επιλογής εισόδου στην έξοδο συμβολίζονται με S0, S1 και S2 (MSB=S2). Τα δεδομένα εξόδου συμβολίζονται με D0, D1,…,D7 (MSB=D7). Η μοναδική είσοδος συμβολίζεται με I. Στον **πίνακα 1.9** απεικονίζεται ο συντετμημένος πίνακας αλήθειας ενός αποπολυπλέκτη 3:8 με γραφικό σύμβολο αυτό της **εικόνας 1.12** και λογική εξίσωση υλοποίησης σε άθροισμα ελαχίστων όρων (minterm) αυτή της **εικόνας 1.13**.



Εικόνα 1.12 Γραφικό σύμβολο αποπολυπλέκτη 3:8

Πίνακας 1.9 Συντετμημένος πίνακας αλήθειας αποπολυπλέκτη 3:8

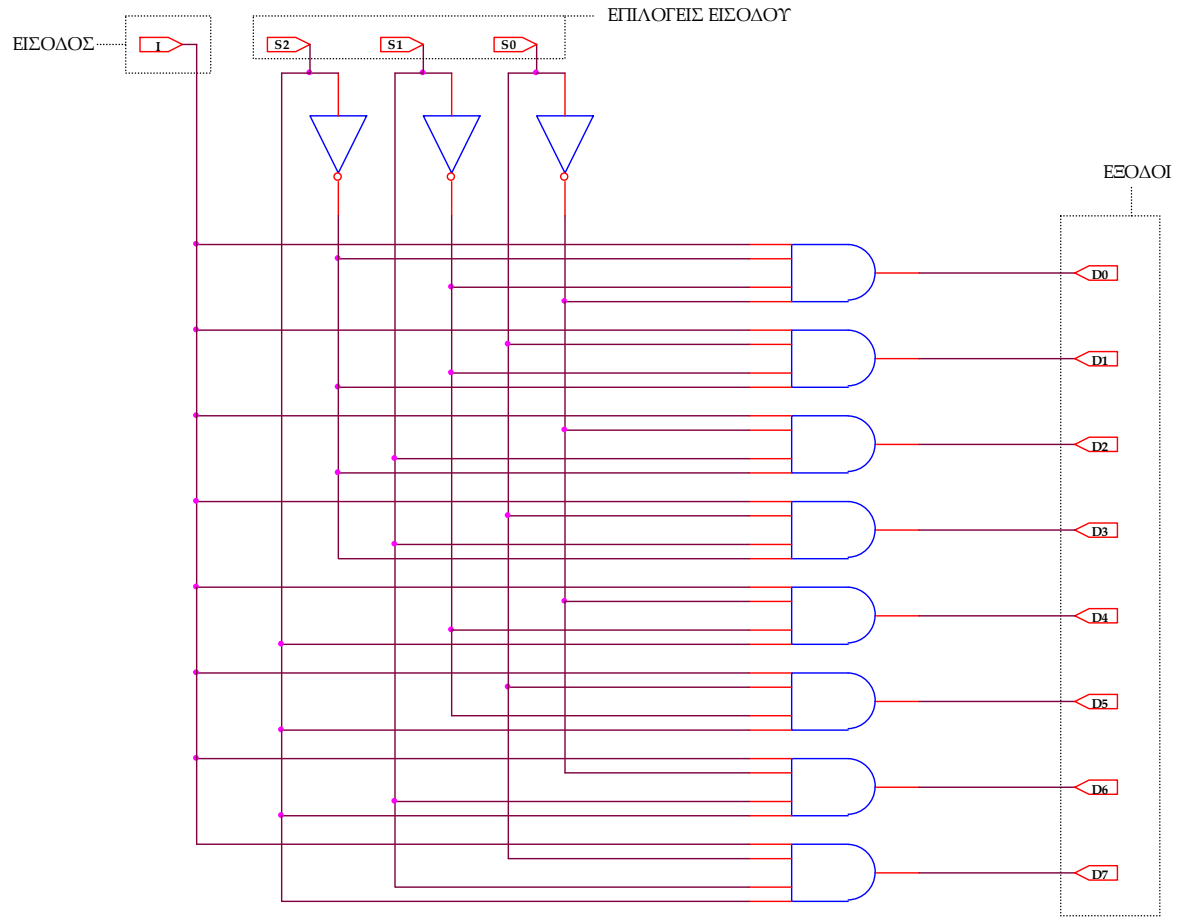
Επιλογή Εισόδων			**Επιλεγόμενη Έξοδος = Εισόδος (I)**							
MSB		**LSB**	**MSB**							**LSB**
S2	**S1**	**S0**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

$$D0 = I \cdot \overline{S2} \cdot \overline{S1} \cdot \overline{S0},\ D1 = I \cdot \overline{S2} \cdot \overline{S1} \cdot S0,\ D2 = I \cdot \overline{S2} \cdot S1 \cdot \overline{S0},\ D3 = I \cdot \overline{S2} \cdot S1 \cdot S0$$
$$D4 = I \cdot S2 \cdot \overline{S1} \cdot \overline{S0},\ D5 = I \cdot S2 \cdot \overline{S1} \cdot S0,\ D6 = I \cdot S2 \cdot S1 \cdot \overline{S0},\ D7 = I \cdot S2 \cdot S1 \cdot S0$$

Εικόνα 1.13 Λογική εξίσωση αποπολυπλέκτη 3:8 σε άθροισμα ελαχίστων όρων (minterm)

1.2.1.2 Συμβατική ψηφιακή υλοποίηση

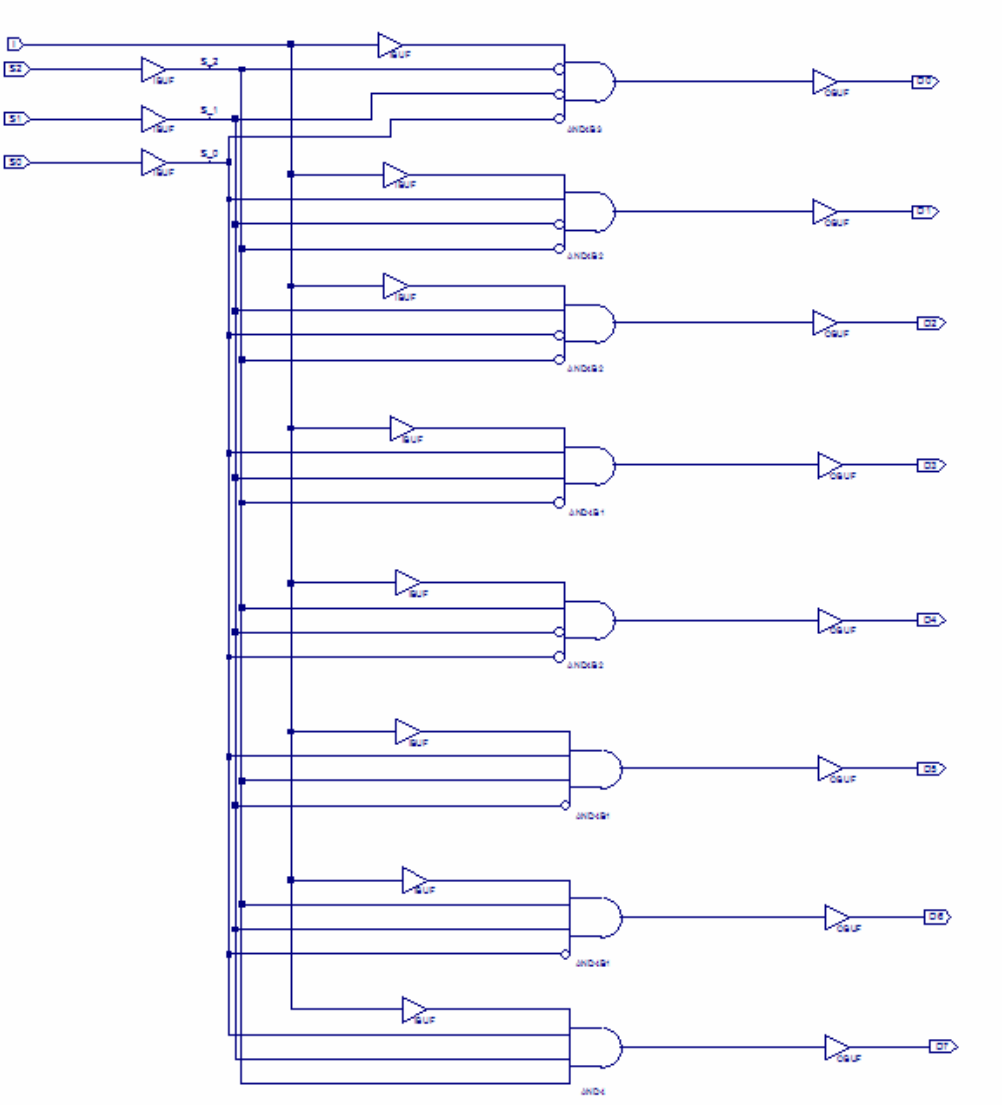
Η λογική εξίσωση της **εικόνας 1.13** παραπέμπει στο λογικό κύκλωμα υλοποίησης με συμβατικές λογικές πύλες της παρακάτω **εικόνας 1.14**.



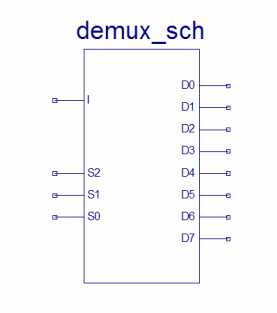
Εικόνα 1.14 Λογικό κύκλωμα υλοποίησης για αποπολυπλέκτη 3:8

1.2.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

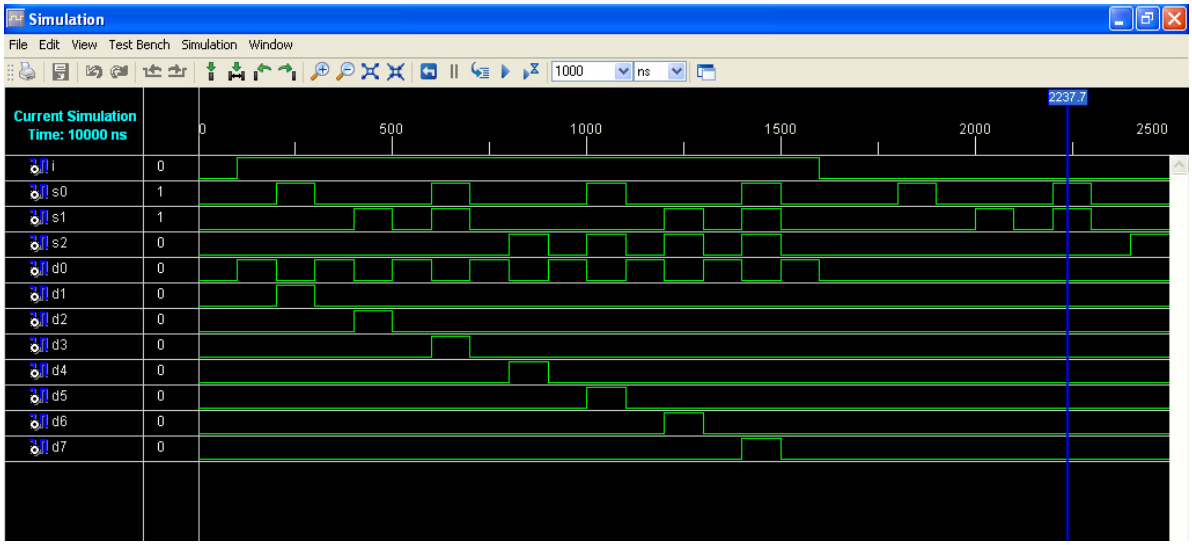
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα. Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



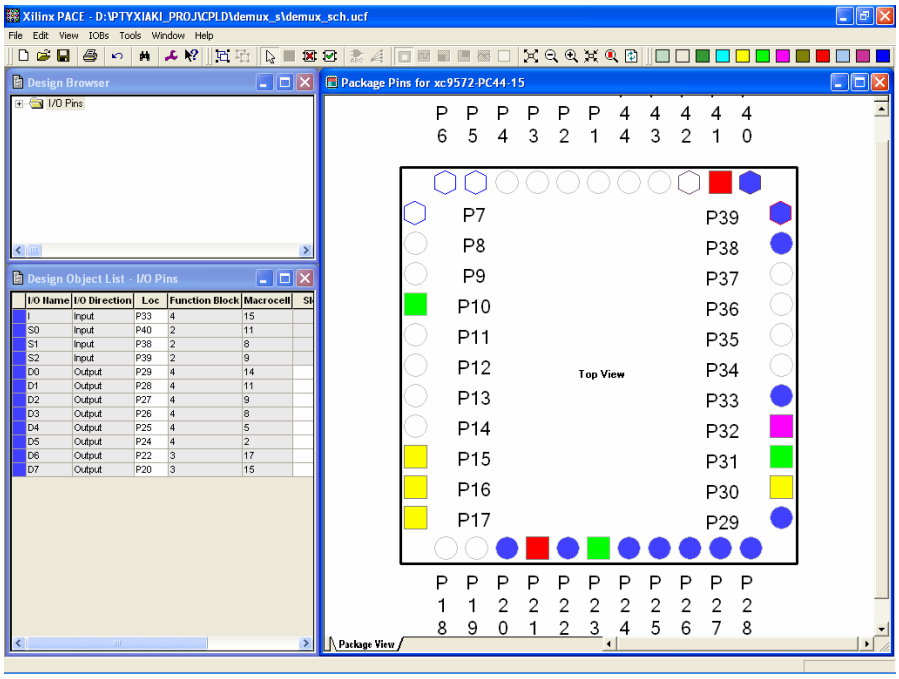
Εικόνα 1.15 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 1.16 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.17 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.18 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

1.2.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 και αναφέρονται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 1.10 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

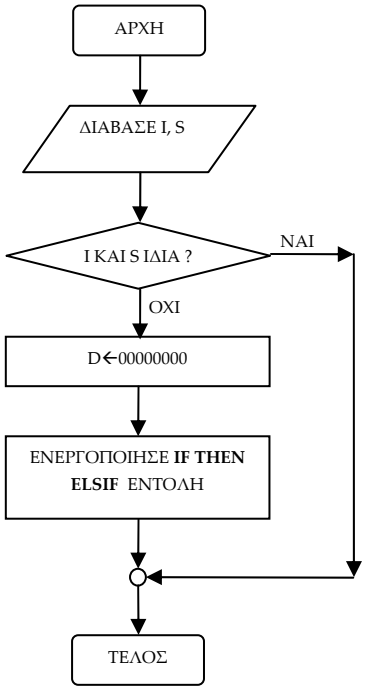
13	architecture BHV of demux_vhd is	Αρχή δήλωσης αρχιτεκτονικής.
14	begin	Αρχή κώδικα αρχιτεκτονικής.
15	process (I, S)	Αρχή της διεργασίας.
16	begin	Αρχή ενεργοποίησης της διεργασίας.
17	D<= ”00000000” ;	Ανάθεση της δυαδικής τιμής 00000000 (8 bit) στην έξοδο **D**.
18	if (s=”000”) then	Κατά την εκτέλεση της **if-elsif-else** γίνεται έλεγχος για το αν η είσοδος **S** έχει πάρει την δυαδική τιμή 000. Το αποτέλεσμα του ελέγχου παίρνει λογική τιμή true (αληθής) ή false (ψευδής). Αν ισχύει η λογική τιμή true για την γραμμή 18 εκτελείται η επόμενη γραμμή 19, διαφορετικά το πρόγραμμα μεταβαίνει στην γραμμή 20.
19	D(0) <=I;	Εκτελείται με το σύμβολο “<=” η ανάθεση της δυαδικής τιμής της εισόδου **I** στο bit 0 του διανύσματος της εξόδου **D** ή **D**(0). Αμέσως μετά το πρόγραμμα μεταβαίνει στην γραμμή 34.
20	elsif (s="001") then	Ισχύει αντίστοιχα ότι στην γραμμή 18 για τη δυαδική τιμή εντός της παρενθέσεως.
21	D(1) <=I;	Ισχύει αντίστοιχα ότι στην γραμμή 19 για το **D**(1).
22	elsif (s="010") then	Ισχύει αντίστοιχα ότι στην γραμμή 18 για τη δυαδική τιμή εντός της παρενθέσεως.
23	D(2) <=I;	Ισχύει αντίστοιχα ότι στην γραμμή 19 για το **D**(2).
24	elsif (s="011") then	Ισχύει αντίστοιχα ότι στην γραμμή 18 για τη δυαδική τιμή εντός της παρενθέσεως.
25	D(3) <=I;	Ισχύει αντίστοιχα ότι στην γραμμή 19 για το **D**(3).
26	elsif (s="100") then	Ισχύει αντίστοιχα ότι στην γραμμή 18 για τη δυαδική τιμή εντός της παρενθέσεως.
27	D(4) <=I;	Ισχύει αντίστοιχα ότι στην γραμμή 19 για το **D**(4).
28	elsif (s="101") then	Ισχύει αντίστοιχα ότι στην γραμμή 18 για τη δυαδική τιμή εντός της παρενθέσεως.

29	D(5) <=I;	Ισχύει αντίστοιχα ότι στην γραμμή 19 για το **D**(5) .
30	elsif (s="110") then	Ισχύει αντίστοιχα ότι στην γραμμή 18 για τη δυαδική τιμή εντός της παρενθέσεως.
31	D(6) <=I;	Ισχύει αντίστοιχα ότι στην γραμμή 19 για το **D**(6).
32	else	Ισχύει αντίστοιχα ότι στην γραμμή 18 για τη δυαδική τιμή εντός της παρενθέσεως.
33	D(7) <=I;	Ισχύει αντίστοιχα ότι στην γραμμή 19 για το **D**(7).
34	end if;	Τέλος εκτέλεσης **if-elsif-else.**
35	end process;	Τέλος εκτέλεσης process.
36	end BHV;	Τέλος κώδικα αρχιτεκτονικής.

Όλα τα παραπάνω που αφορούν ακολουθιακό τρόπο λειτουργίας της διεργασίας (**process**) και της περιεχομένης σ' αυτήν εντολής **if-elsif-else** εκφράζονται ποιοτικά από τα παρακάτω διαγράμματα ροής (flow charts).

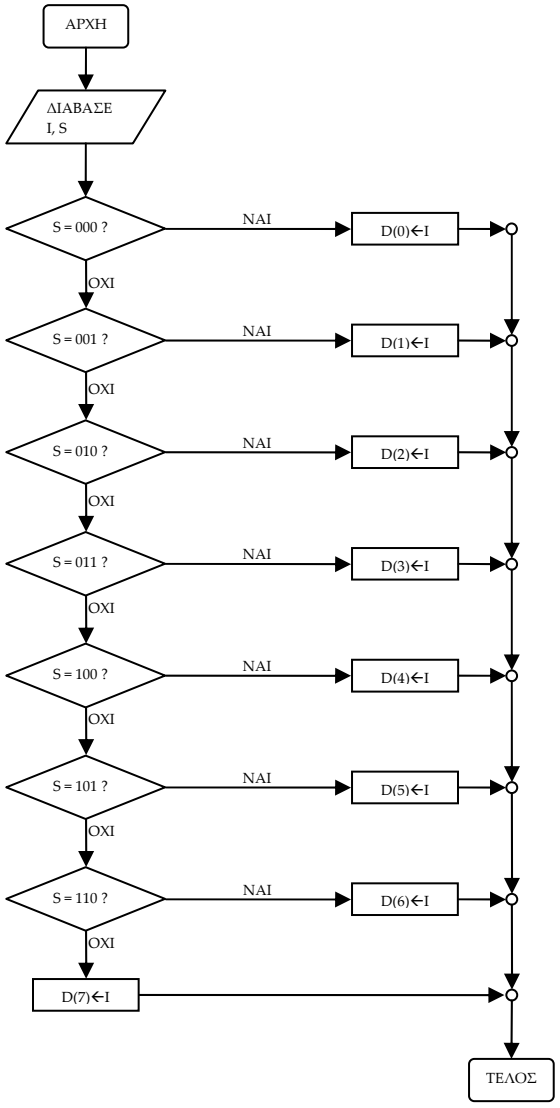
Πίνακας 1.11 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (I, S)

begin

        D<=”00000000”;

        if then … elsif … end if;


end process;
```



Πίνακας 1.12 Η ακολουθιακή εντολή **if-elsif-else** και το **διάγραμμα ροής**

```
if (s=”000”) then
                D(0) <=I;
elsif (s="001") then
                O <= D(1);
elsif (s="010") then
                O <= D(2);
elsif (s="011") then
                O <= D(3);
elsif (s="100") then
                O <= D(4);
elsif (s="101") then
                O <= D(5);
elsif (s="110") then
                O <= D(6);
else
                O <= D(7);
end if;
```



Σημείωση. Στα παραπάνω διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

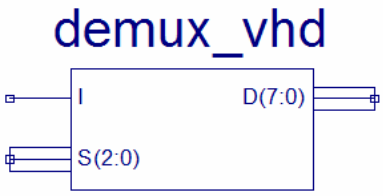
1.2.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

demux_vhd.vhd

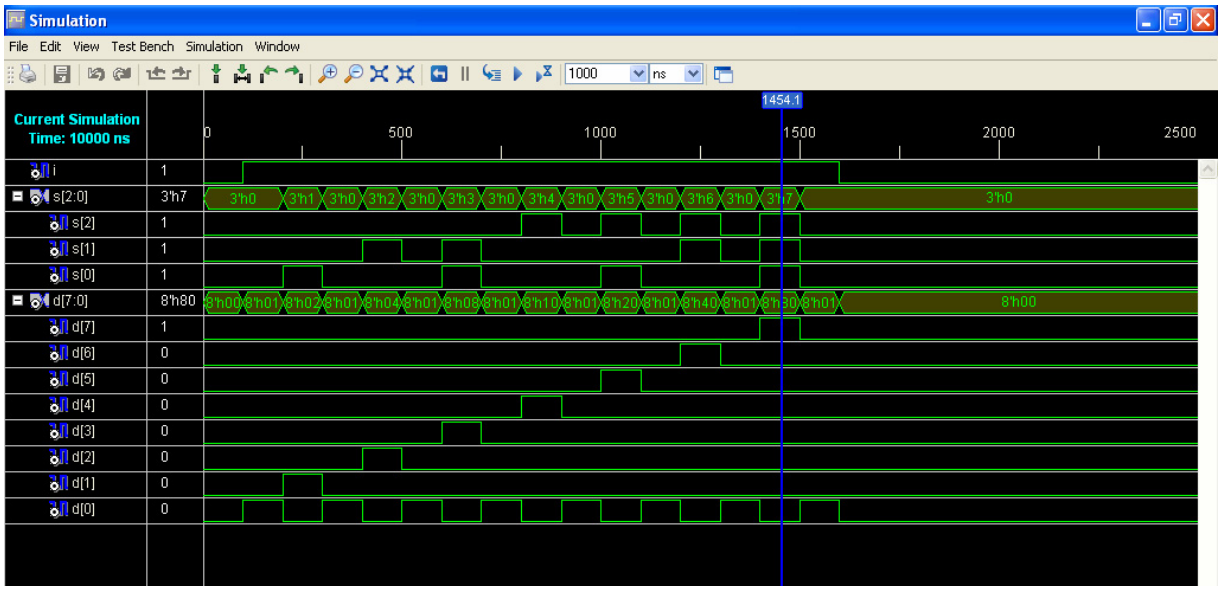
File Edit View Window

```
1  ----------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;
6  ----------------------------------------------------
7  entity demux_vhd is
8      Port ( I : in  STD_LOGIC;
9             S : in  STD_LOGIC_VECTOR (2 downto 0);
10            D : out  STD_LOGIC_VECTOR (7 downto 0));
11 end demux_vhd;
12 ----------------------------------------------------
13 architecture BHV of demux_vhd is
14 begin
15    process (I, S)
16    begin
17    D<="00000000";
18          if (s="000")   then
19                                D(0) <= I;
20          elsif (s="001") then
21                                D(1) <= I;
22          elsif (s="010") then
23                                D(2) <= I;
24          elsif (s="011") then
25                                D(3) <= I;
26          elsif (s="100") then
27                                D(4) <= I;
28          elsif (s="101") then
29                                D(5) <= I;
30          elsif (s="110") then
31                                D(6) <= I;
32          else
33                D(7) <= I;
34          end if;
35    end process;
36 end BHV;
```

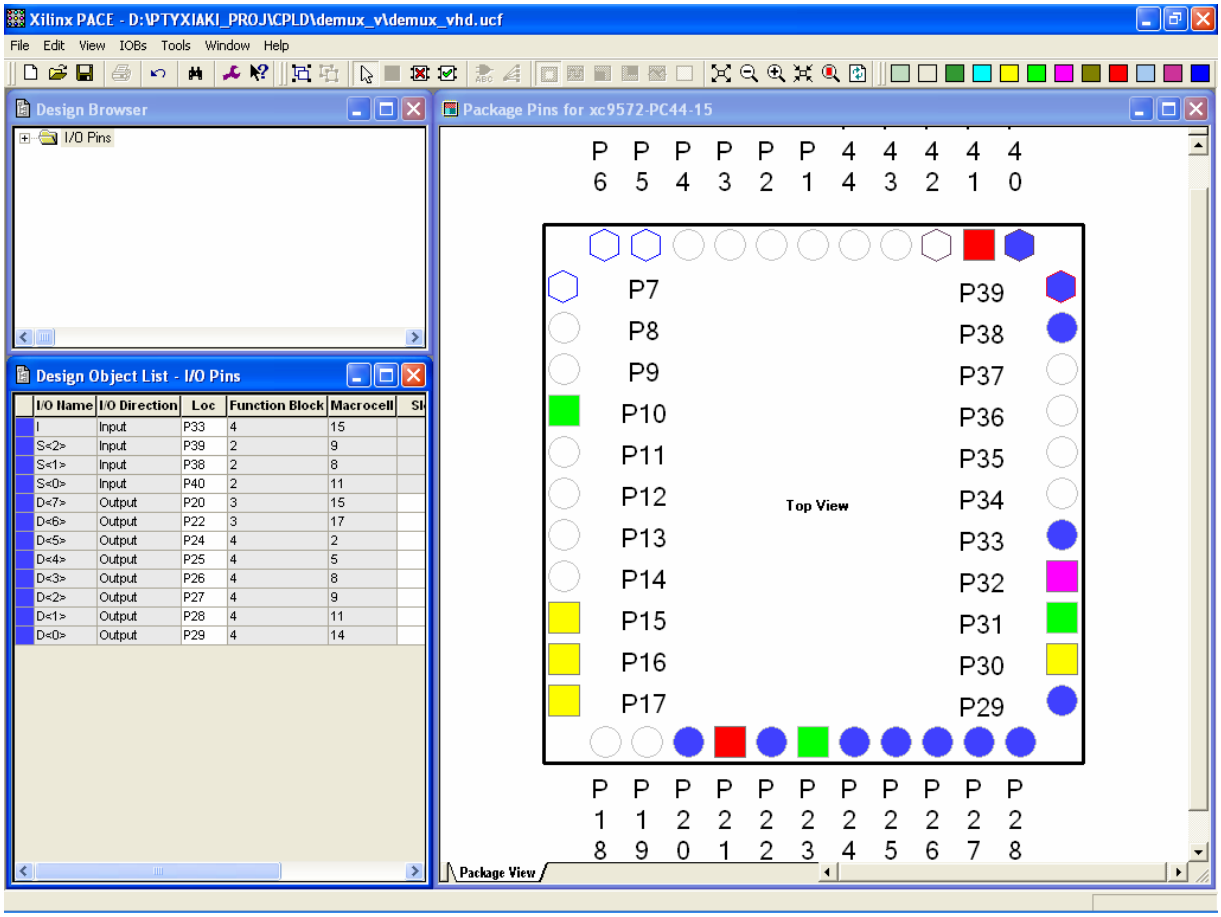
Εικόνα 1.19 VHDL αρχείο του κυκλώματος



Εικόνα 1.20 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.21 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.22 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

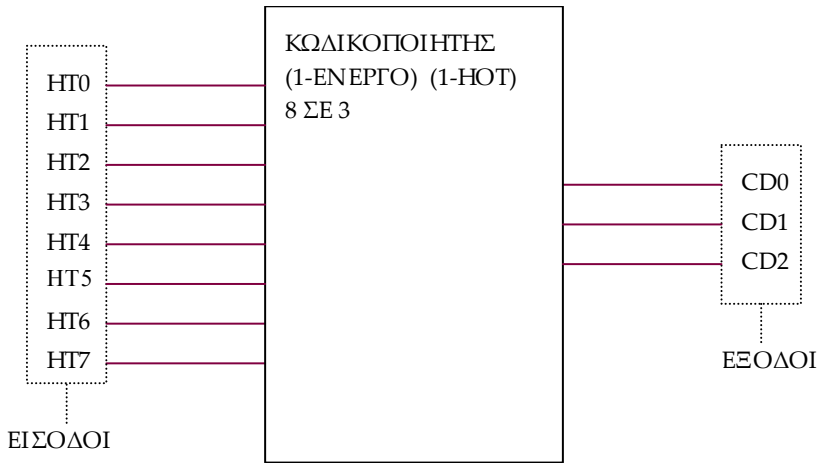
1.3 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ: ΚΩΔΙΚΟΠΟΙΗΤΗΣ 8 ΣΕ 3

1.3.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

1.3.1.1 Σύντομη θεωρία

Ο κωδικοποιητής (*encoder*) αποτελεί το συνδυαστικό εκείνο ψηφιακό κύκλωμα κατά το οποίο η έξοδός του είναι η κωδικοποιημένη είσοδος. Υπάρχουν διάφορα είδη κωδικοποίησης. Εδώ θα εξεταστεί η περίπτωση της κωδικοποίησης του 1-ενεργό (1-hot). Κατά την κωδικοποίηση αυτή έχουμε το πολύ 2^n γραμμές εισόδου να κατευθύνονται σε n γραμμές εξόδου. Να σημειωθεί επίσης ότι κάθε φορά μόνο μία από τις εισόδους μπορεί να είναι ενεργή με λογικό 1 (κωδικοποίηση 1-ενεργό) ενώ όλες οι υπόλοιπες θα βρίσκονται σε λογικό 0. Στην κωδικοποιημένη έξοδο απεικονίζεται το bit εκείνο της εισόδου που είναι ενεργό.

Ειδικότερα στον κωδικοποιητή 8 σε 3 έχουμε 8 γραμμές εισόδου να κατευθύνονται κωδικοποιημένες σε 3 γραμμές εξόδου. Τα δεδομένα εισόδου συμβολίζονται με HT0, HT1,…, HT7 (MSB= HT7). Οι έξοδοι συμβολίζονται με CD0, CD1, CD2 (MSB= CD2). Στον **πίνακα 1.13** απεικονίζεται ο πίνακας αλήθειας ενός κωδικοποιητή 8 σε 3 με γραφικό σύμβολο αυτό της **εικόνας 1.23** και λογική εξίσωση υλοποίησης σε άθροισμα ελαχίστων όρων (minterm) αυτή της **εικόνας 1.24**.



Εικόνα 1.23 Γραφικό σύμβολο του κωδικοποιητή 8 σε 3

Πίνακας 1.13 Πίνακας αλήθειας κωδικοποιητή 8 σε 3

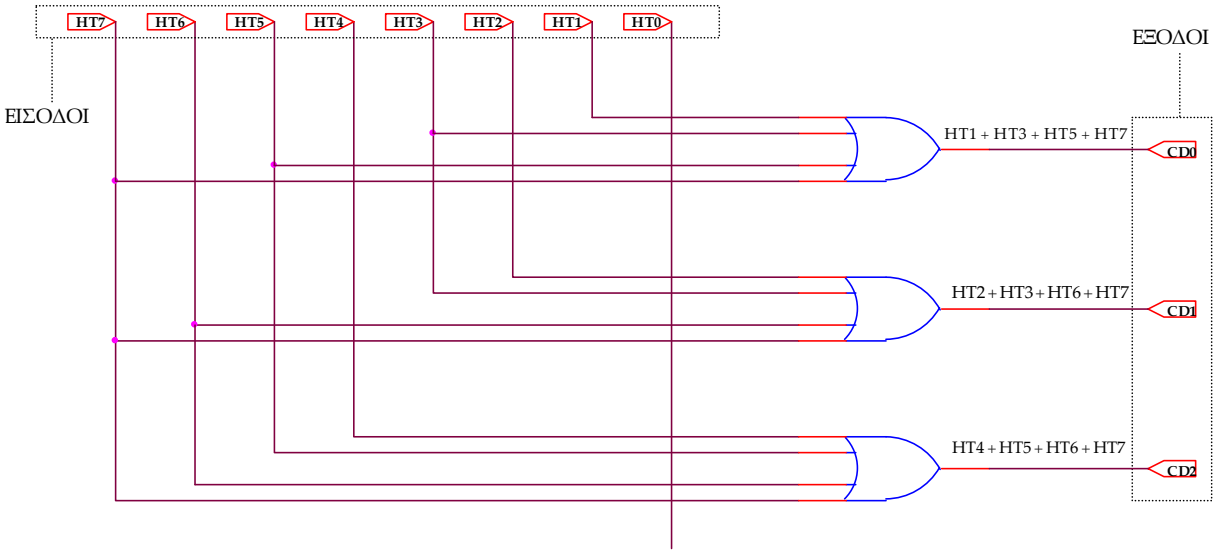
ΕΙΣΟΔΟΙ (1-ΕΝΕΡΓΟ ή 1-HOT) 8 ΣΕ 3								ΕΞΟΔΟΙ		
MSB							**LSB**	**MSB**		**LSB**
HT7	HT6	HT5	HT4	HT3	HT2	HT1	HT0	CD2	CD1	CD0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$CD0 = HT1 + HT3 + HT5 + HT7,\ CD1 = HT2 + HT3 + HT6 + HT7$$
$$CD2 = HT4 + HT5 + HT6 + HT7$$

Εικόνα 1.24 Λογική εξίσωση κωδικοποιητή 8 σε 3 σε άθροισμα ελαχίστων όρων (minterm)

1.3.1.2 Συμβατική ψηφιακή υλοποίηση

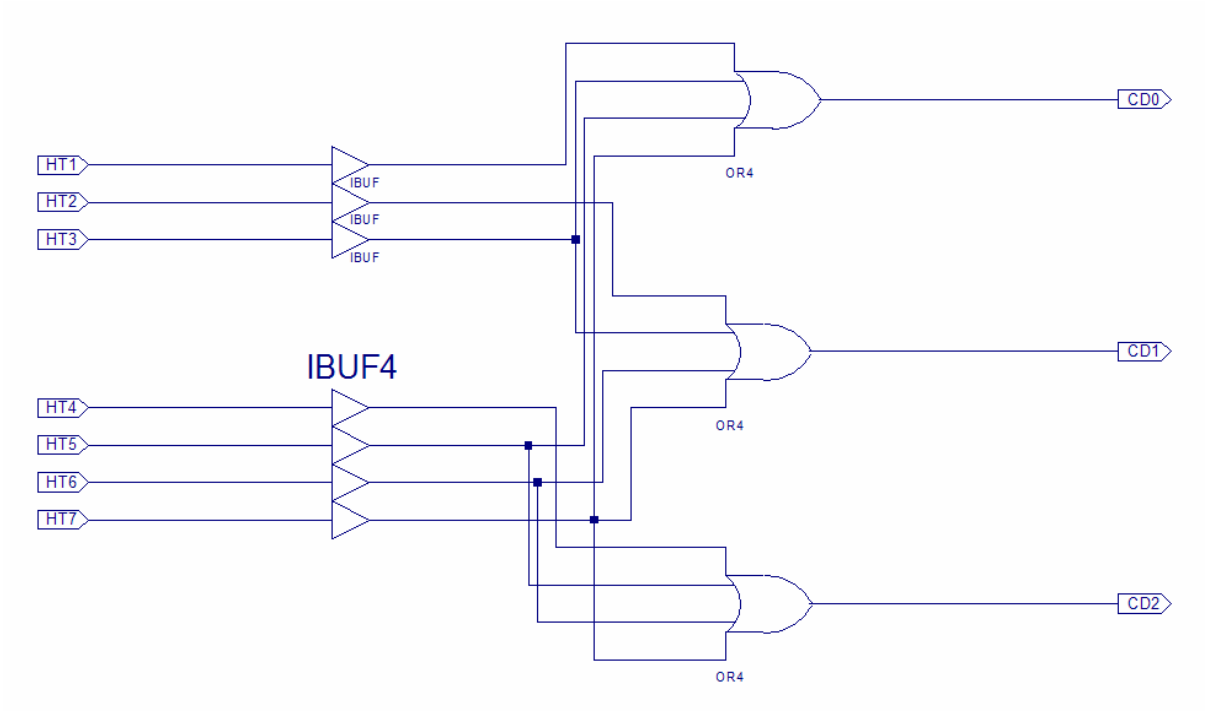
Η παραπάνω λογική εξίσωση παραπέμπει στο παρακάτω κύκλωμα το υλοποιήσιμο με συμβατικές πύλες.



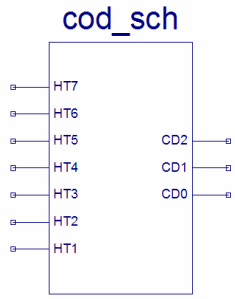
Εικόνα 1.25 Λογικό κύκλωμα υλοποίησης του κωδικοποιητή 8 σε 3

1.3.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

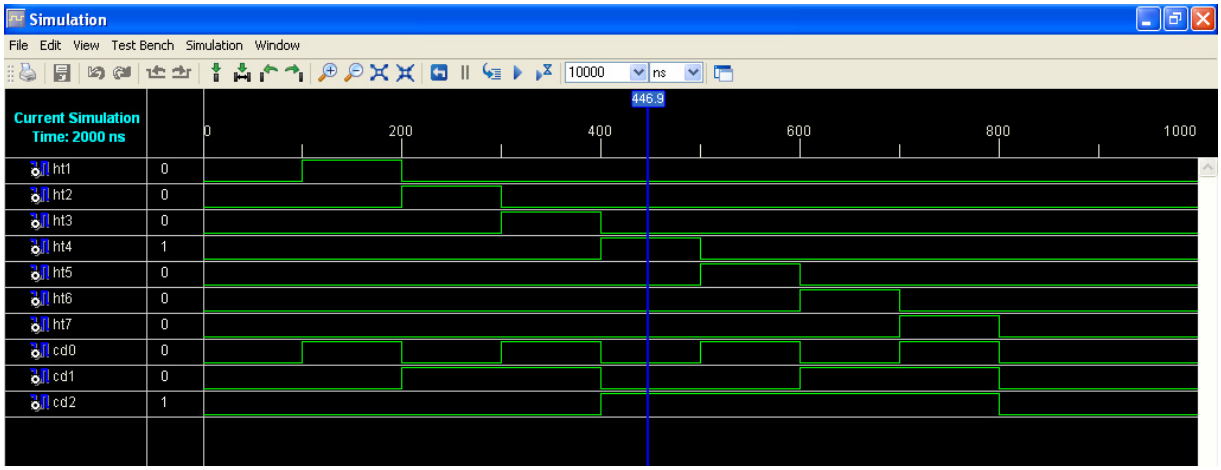
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα. Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



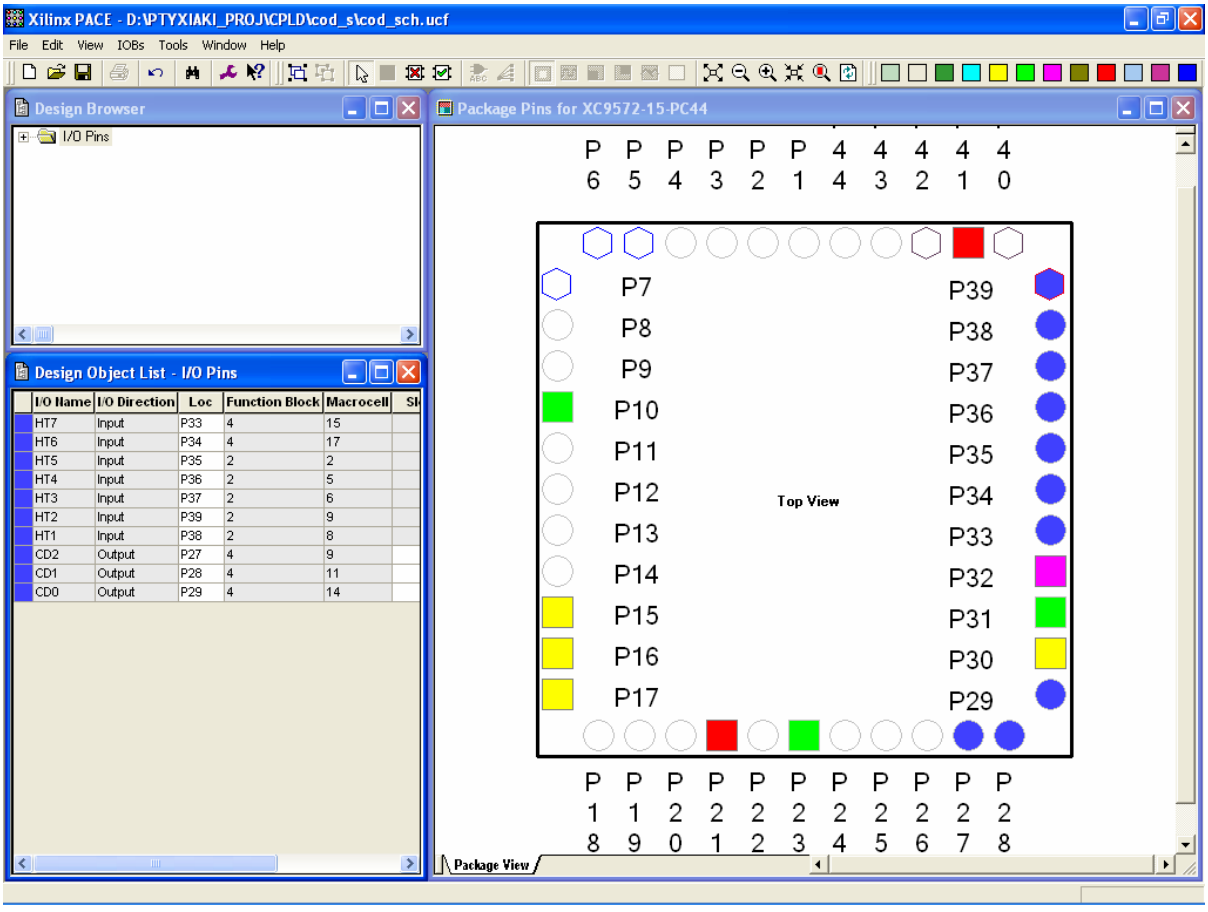
Εικόνα 1.26 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 1.27 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.28 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.29 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

Παρατηρήσεις: Μία από τις εισόδους (η HT0) παραμένει ασύνδετη διότι η οποιαδήποτε λογική της κατάσταση δεν επηρεάζει το τελικό αποτέλεσμα.

1.3.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 και αναφέρονται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 1.14 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

```
12 architecture BHV_DFL of cod_vhd is
13 begin
14     cd <= "000" when ht = "00000001" else
15           "001" when ht = "00000010" else
16           "010" when ht = "00000100" else
17           "011" when ht = "00001000" else
18           "100" when ht = "00010000" else
19           "101" when ht = "00100000" else
20           "110" when ht = "01000000" else
21           "111" when ht = "10000000" else
22           "---";
23 end BHV_DFL;
```

Αρχή δήλωσης αρχιτεκτονικής.

Αρχή κώδικα αρχιτεκτονικής.

Εκτέλεση εντολής **when-else**. Για συγκεκριμένη τιμή της εισόδου **ht** έχουμε συγκεκριμένη απόδοση τιμής στην έξοδο **cd** δηλωμένη στην ίδια γραμμή εντολής. Στην πρώτη συνθήκη που θα βρεθεί αληθής (true) ο έλεγχος για την τιμή της εισόδου **ht** θα αποδοθεί στην έξοδο **cd** η αντίστοιχη δηλωμένη τιμή στην ίδια γραμμή εντολής.

Στην περίπτωση εισαγωγής στην είσοδο **cd** μη προβλεπόμενου συνδυασμού τιμής από τις παραπάνω, δηλώνεται (με τη λέξη else της γραμμής 21) ό,τι η έξοδος **cd** παίρνει την αδιάφορη τιμή της γραμμής 22.

Τέλος κώδικα αρχιτεκτονικής.

Σημείωση. Η αρχιτεκτονική περιέχει μόνο συντρέχοντα κώδικα (data flow code) που υλοποιείται μέσω της εντολής **when-else**. Ο συντρέχοντας κώδικας δηλώνεται με τη συντμημένη λέξη **DFL** μέσα στο όνομα της αρχιτεκτονικής BHV_**DFL**.
Παράδειγμα **when-else** για γραμμή εντολών 20: έστω ο Boolean έλεγχος είναι αληθής (true) για την δυαδική τιμή 01000000 της εισόδου **ht**, έχουμε λοιπόν μεταφορά στην έξοδο **cd** της τιμής 110 που επίσης αναγράφεται στην ίδια γραμμή εντολής 20. (Το ίδιο ισχύει για όλους τους αριθμούς γραμμών εντολών.)
Στην **γραμμή 22** εναλλακτικά μπορεί να εισαχθεί η τιμή "XXX".

Όλα τα παραπάνω αφορούν συντρέχοντα κώδικα που δεν περιγράφεται ποιοτικά από διαγράμματα ροής (flow charts) όπως ο ακολουθιακός κώδικας. Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

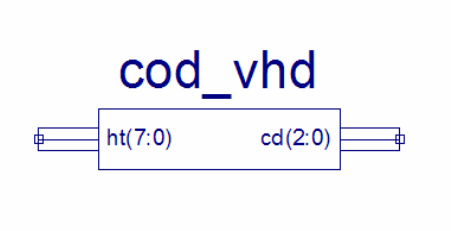
1.3.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

cod_vhd.vhd

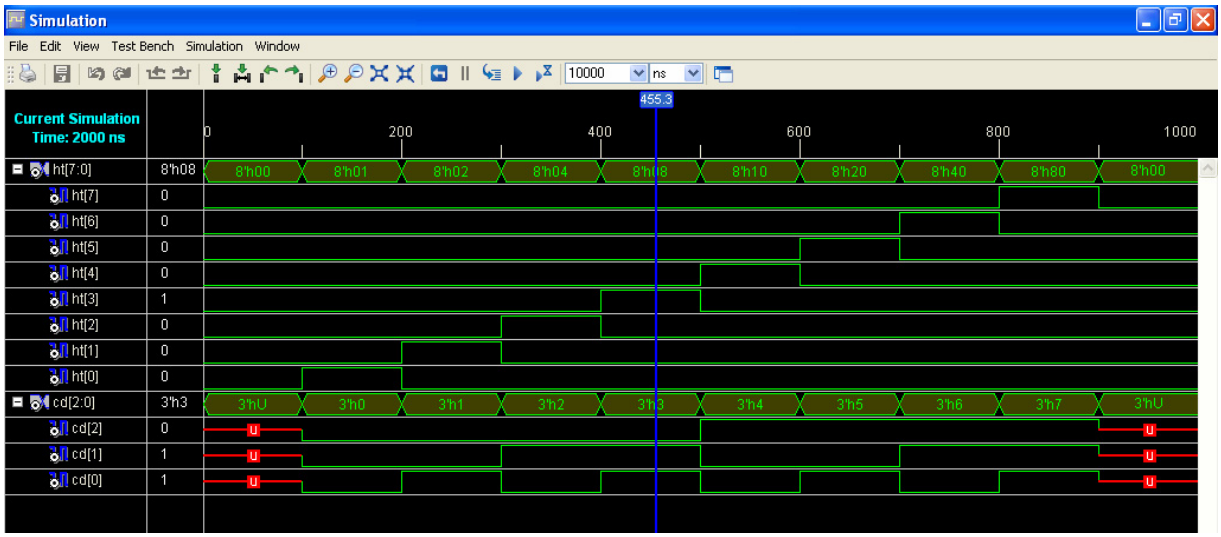
File Edit View Window

```
1  ----------------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;
6  ----------------------------------------------------------
7  entity cod_vhd is
8      Port ( ht : in  STD_LOGIC_VECTOR (7 downto 0);
9             cd : out  STD_LOGIC_VECTOR (2 downto 0));
10 end cod_vhd;
11 ----------------------------------------------------------
12 architecture BHV_DFL of cod_vhd is
13 begin
14       cd <= "000" when ht = "00000001" else
15             "001" when ht = "00000010" else
16             "010" when ht = "00000100" else
17             "011" when ht = "00001000" else
18             "100" when ht = "00010000" else
19             "101" when ht = "00100000" else
20             "110" when ht = "01000000" else
21             "111" when ht = "10000000" else
22             "---";
23 end BHV_DFL;
```

Εικόνα 1.30 VHDL αρχείο του κυκλώματος

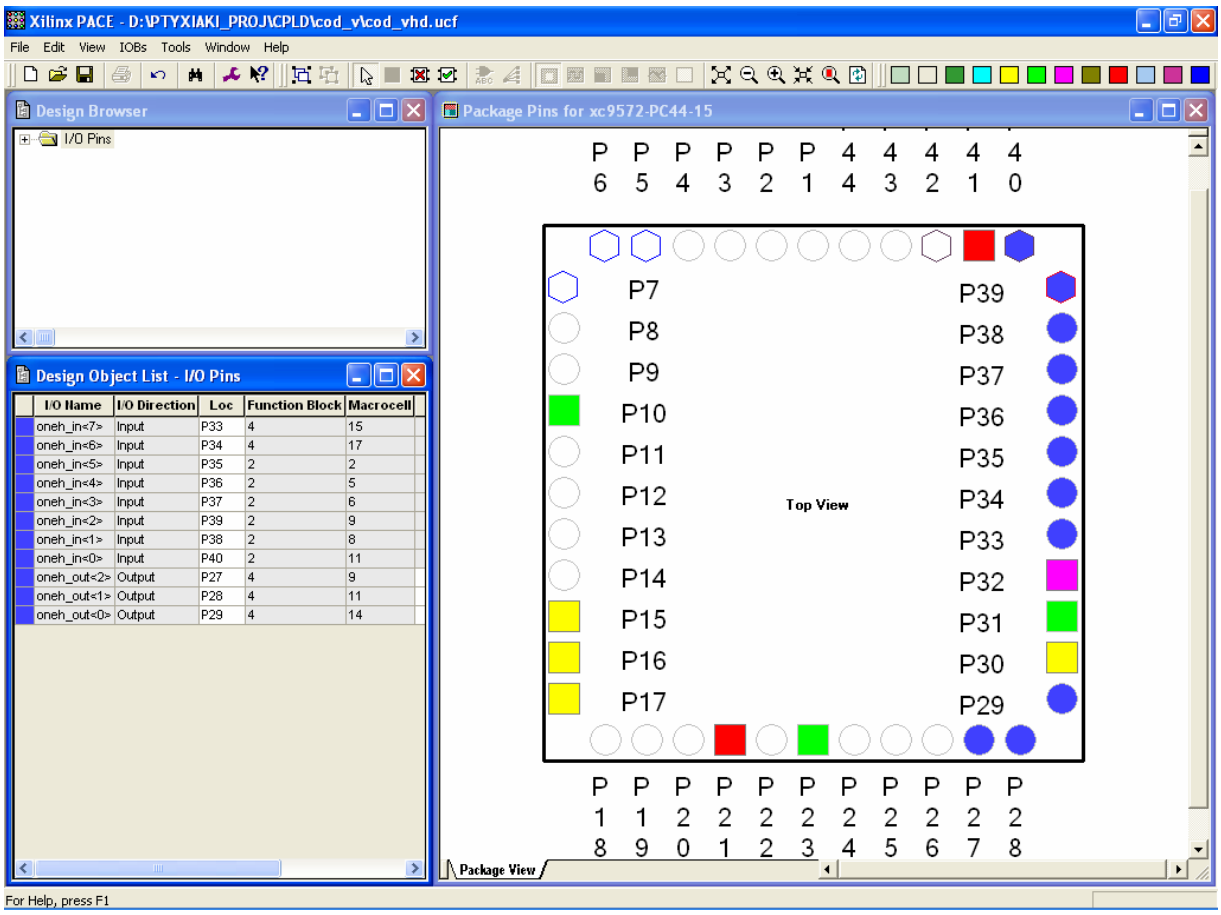


Εικόνα 1.31 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.32 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος

◆ **Παρατηρήσεις:** Από την ανάθεση της τιμής "**---**" (αδιάφορη κατάσταση) στην έξοδο **cd** (γραμμή 22 στο αρχείο VHDL) προκύπτει η προσομοιώσιμη τιμή "**U**" (παραπάνω παράθυρο) που μας «κατευθύνει» για ενδεχόμενο λάθος στην τελική λειτουργία του κυκλώματος μετά τον προγραμματισμό. Αυτό δεν υφίσταται διότι η ανάθεση της τιμής "**---**" είναι συνθέσιμη σαν αδιάφορη κατάσταση με αποτέλεσμα την άψογη μετέπειτα λειτουργία του κυκλώματος για όλους τους δυνατούς συνδυασμούς τιμών της εισόδου. Όλα τα προηγούμενα χωρίς την ασύνδετη είσοδο και την λειτουργία του κυκλώματος μόνο για τους προβλεπόμενους συνδυασμούς τιμών της εισόδου που παρατηρούμε στον αντίστοιχο σχεδιασμό με τη σχηματική γλώσσα.



Εικόνα 1.33 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

1.4 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ: ΑΠΟΚΩΔΙΚΟΠΟΙΗΤΗΣ 3 ΣΕ 8

1.4.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

1.4.1.1 Σύντομη θεωρία

Ο αποκωδικοποιητής (*decoder*) είναι το συνδυαστικό ψηφιακό κύκλωμα το οποίο λειτουργεί αντίστροφα από τον κωδικοποιητή. Κατά την αποκωδικοποίηση 1-ενεργό (1-hot) μετατρέπεται η δυαδική πληροφορία n γραμμών εισόδου σε έως το πολύ 2^n γραμμές εξόδου. Να σημειωθεί επίσης ότι κάθε φορά μόνο μία από τις εξόδους μπορεί να είναι ενεργή με λογικό 1 (κωδικοποίηση 1-ενεργό) ενώ όλες οι υπόλοιπες θα βρίσκονται σε λογικό 0.

Ειδικότερα στην αποκωδικοποίηση 1-ενεργό (1-hot) ή 3 σε 8 έχουμε 3 γραμμές εισόδου να κατευθύνονται αποκωδικοποιημένες σε 8 γραμμές εξόδου. Τα δεδομένα εισόδου συμβολίζονται με DC0, DC1, DC2 (MSB= DC2). Οι έξοδοι συμβολίζονται με HT0, HT1,…, HT7 (MSB= HT7). Στον **πίνακα 1.15** λοιπόν απεικονίζεται ο πίνακας αλήθειας ενός αποκωδικοποιητή 3 σε 8 με γραφικό σύμβολο αυτό της **εικόνας 1.34** και λογική εξίσωση υλοποίησης σε άθροισμα ελαχίστων όρων (minterm) αυτή της **εικόνας 1.35**.

Εικόνα 1.34 Γραφικό σύμβολο αποκωδικοποιητή 3 σε 8

Πίνακας 1.15 Πίνακας αλήθειας αποκωδικοποιητή 3 σε 8

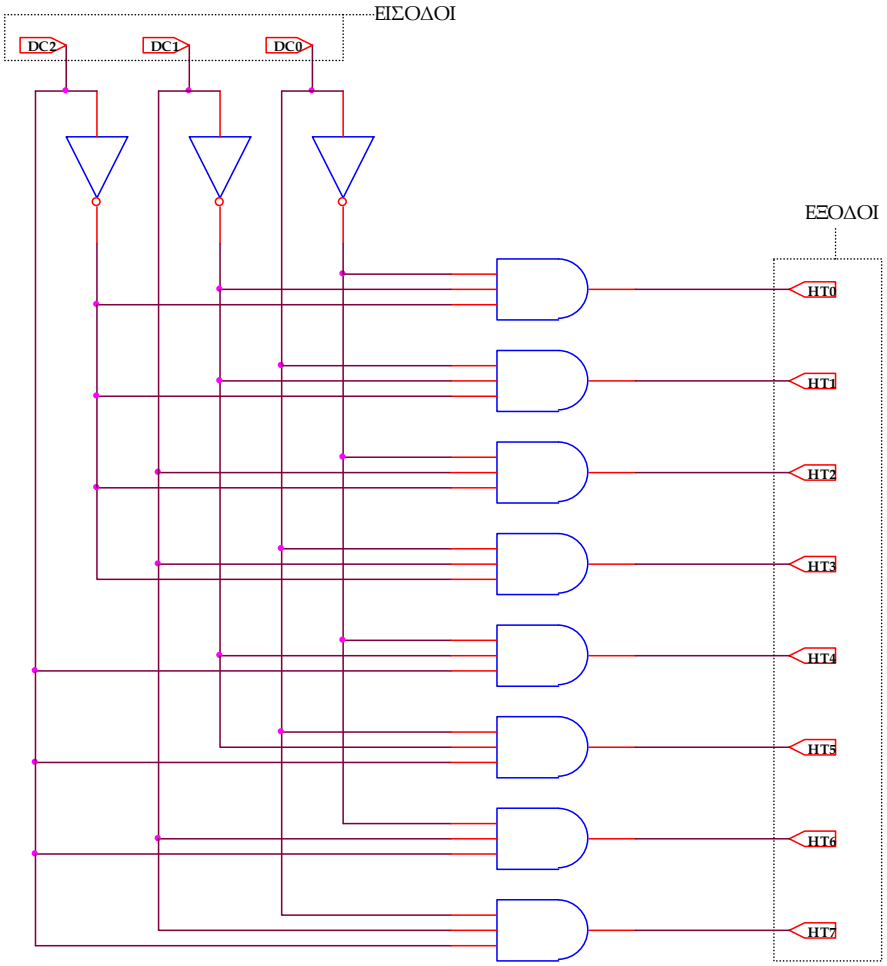
ΕΙΣΟΔΟΙ			ΕΞΟΔΟΙ (1-ΕΝΕΡΓΟ ή 1-HOT) 3 ΣΕ 8							
MSB		**LSB**	**MSB**							**LSB**
DC2	**DC1**	**DC0**	**HT7**	**HT6**	**HT5**	**HT4**	**HT3**	**HT2**	**HT1**	**HT0**
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

$$HT0 = \overline{DC0} \cdot \overline{DC1} \cdot \overline{DC2}, HT1 = DC0 \cdot \overline{DC1} \cdot \overline{DC2}, HT2 = \overline{DC0} \cdot DC1 \cdot \overline{DC2},$$
$$HT3 = DC0 \cdot DC1 \cdot \overline{DC2}, HT4 = \overline{DC0} \cdot \overline{DC1} \cdot DC2, HT5 = DC0 \cdot \overline{DC1} \cdot DC2,$$
$$HT6 = \overline{DC0} \cdot DC1 \cdot DC2, HT7 = DC0 \cdot DC1 \cdot DC2.$$

Εικόνα 1.35 Λογική εξίσωση αποκωδικοποιητή 3 σε 8 σε άθροισμα ελαχίστων όρων (minterm)

1.4.1.2 Συμβατική ψηφιακή υλοποίηση

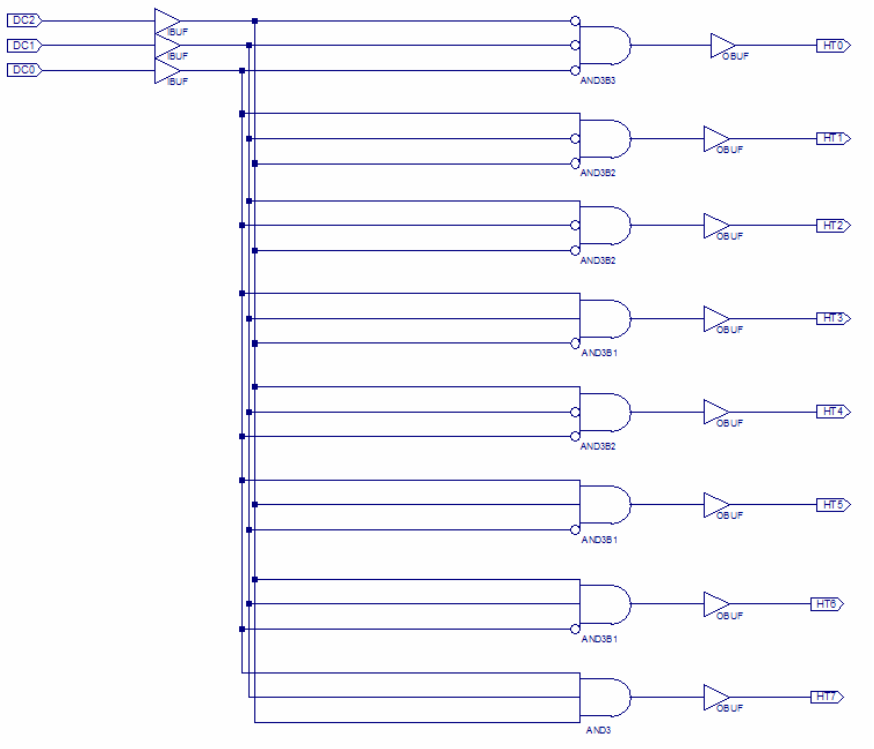
Η παραπάνω λογική εξίσωση παραπέμπει στο παρακάτω κύκλωμα το υλοποιήσιμο με συμβατικές πύλες.



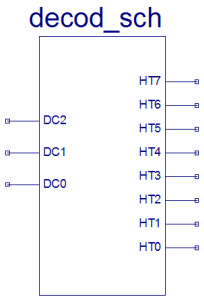
Εικόνα 1.36 Λογικό κύκλωμα υλοποίησης του αποκωδικοποιητή 3 σε 8

1.4.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

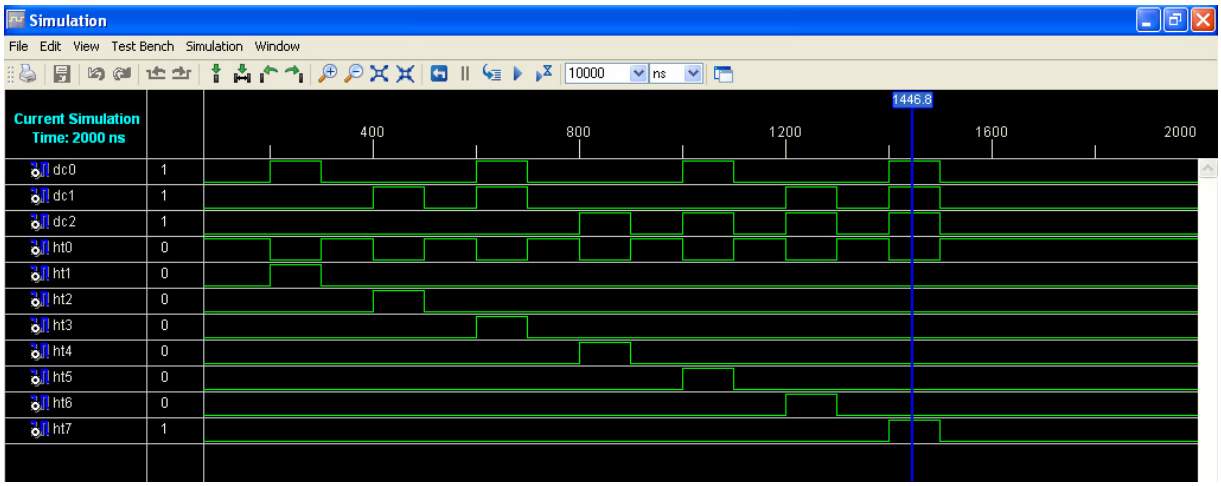
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα. Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



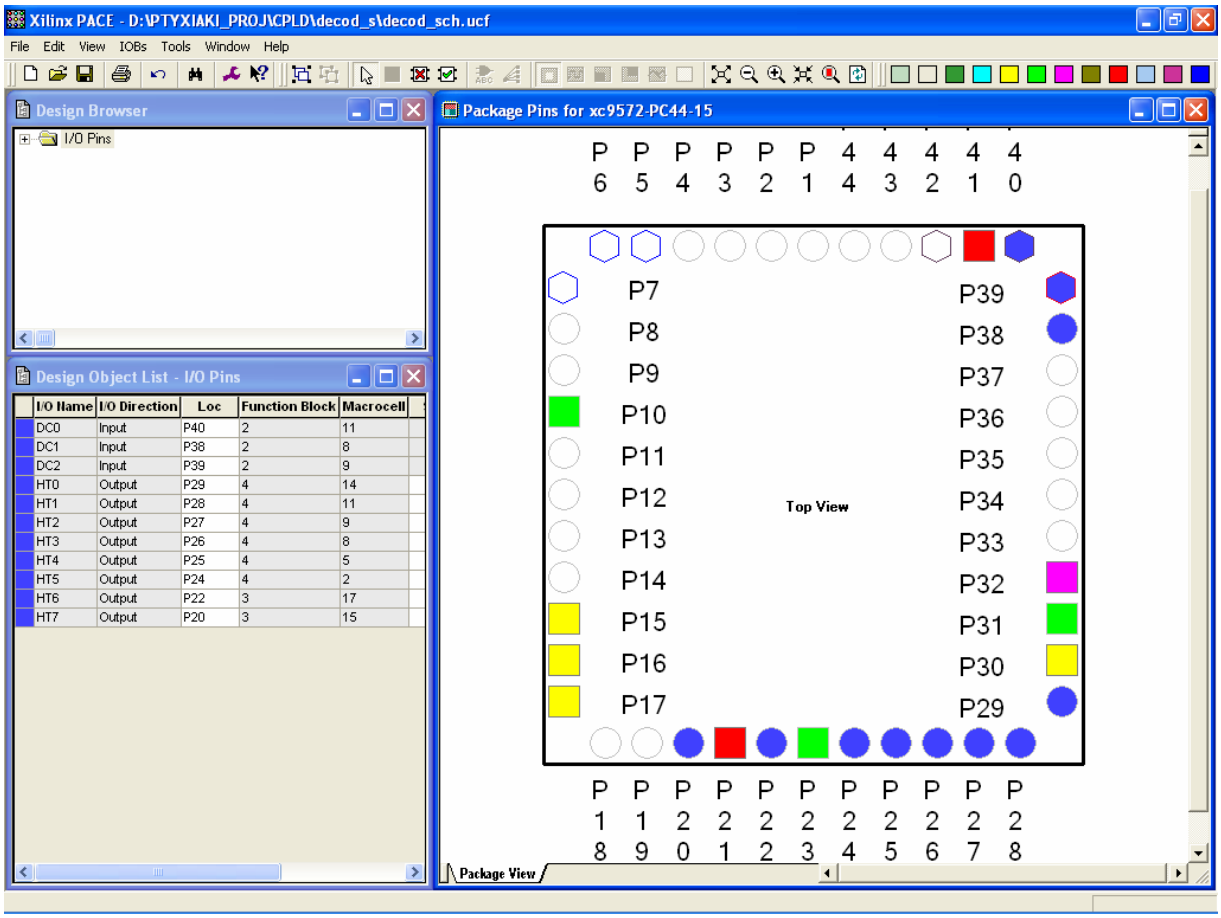
Εικόνα 1.37 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 1.38 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.39 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.40 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

1.4.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 και αναφέρονται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

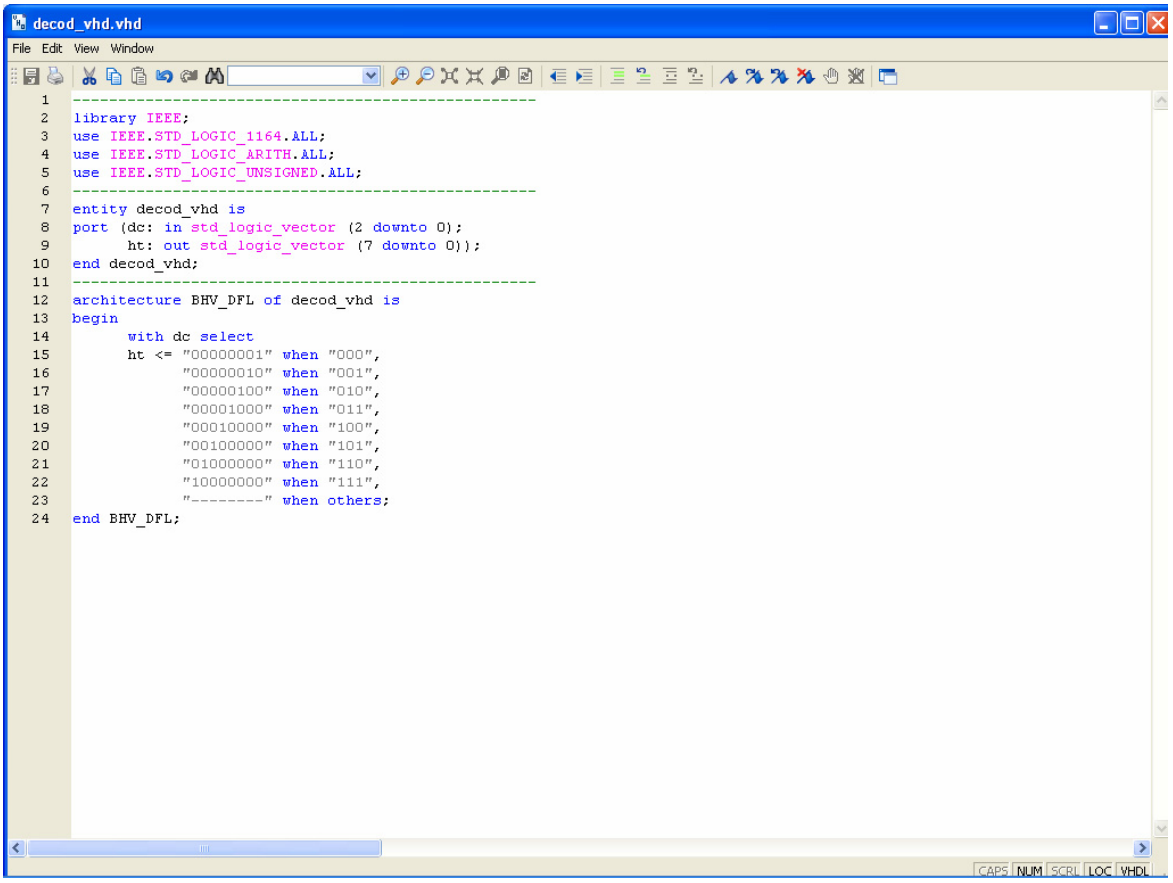
Πίνακας 1.16 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

12	architecture BHV_DFL of decod_vhd is	Αρχή δήλωσης αρχιτεκτονικής.
13	begin	Αρχή κώδικα αρχιτεκτονικής.
14	with dc select	Εκτέλεση εντολής **with-select-when**. Μηχανισμός εντολής ανάλογος της εντολής **when-else** που είδαμε στον κωδικοποιητή 8 σε 3. Στην εντολή **with-select-when** γράφουμε όλους τους δυνατούς συνδυασμούς των τιμών της εισόδου. Το τελευταίο δεν είναι υποχρεωτικό για την λειτουργία της εντολής **when-else**.
15	ht <= "00000001" when = "000",	
16	"00000010" when = "001",	
17	"00000100" when = "010",	
18	"00001000" when = "011",	
19	"00010000" when = "100",	
20	"00100000" when = "101",	
21	"01000000" when = "110",	
22	"10000000" when = "111",	
23	"--------" when others;	Στην περίπτωση εισαγωγής στην είσοδο **dc** μη προβλεπόμενου συνδυασμού τιμής από τις παραπάνω, δηλώνεται (με τη λέξη when others της γραμμής 23) ό,τι η έξοδος **ht** παίρνει την αδιάφορη τιμή της γραμμής 23.
24	end BHV_DFL;	Τέλος κώδικα αρχιτεκτονικής.

Σημείωση. Η αρχιτεκτονική περιέχει μόνο συντρέχοντα κώδικα (data flow code) που υλοποιείται μέσω της εντολής **with-select-when**. Ο συντρέχοντας κώδικας δηλώνεται με τη συντμημένη λέξη **DFL** μέσα στο όνομα της αρχιτεκτονικής BHV_**DFL**.
Στην **γραμμή 23** εναλλακτικά μπορεί να εισαχθεί η τιμή "XXXXXXXX".

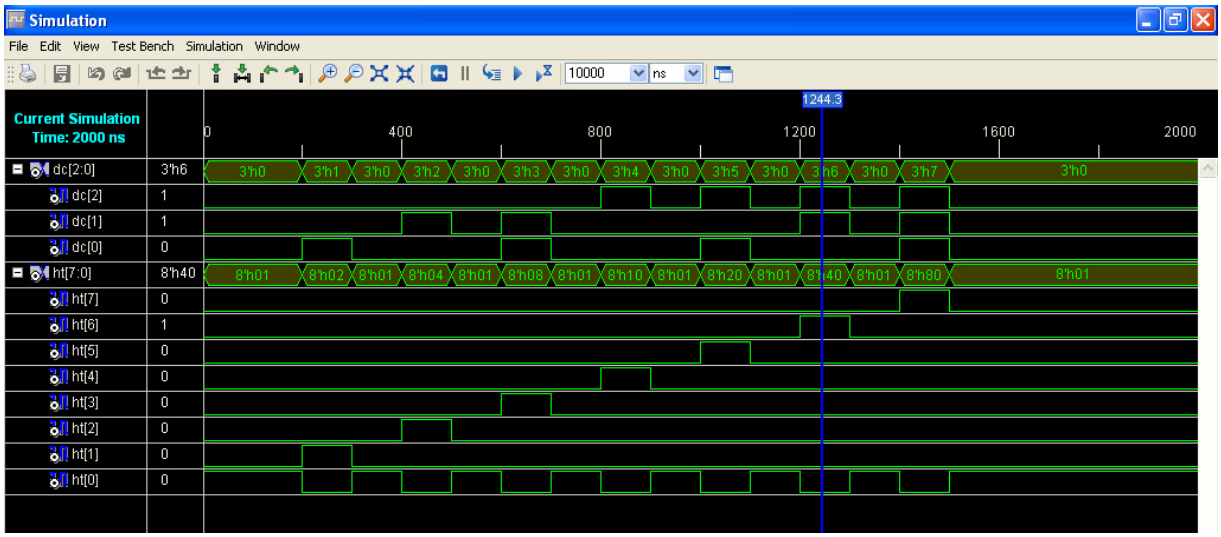
Όλα τα παραπάνω αφορούν συντρέχοντα κώδικα που δεν περιγράφεται ποιοτικά από διαγράμματα ροής (flow charts) όπως ο ακολουθιακός κώδικας. Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

1.4.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

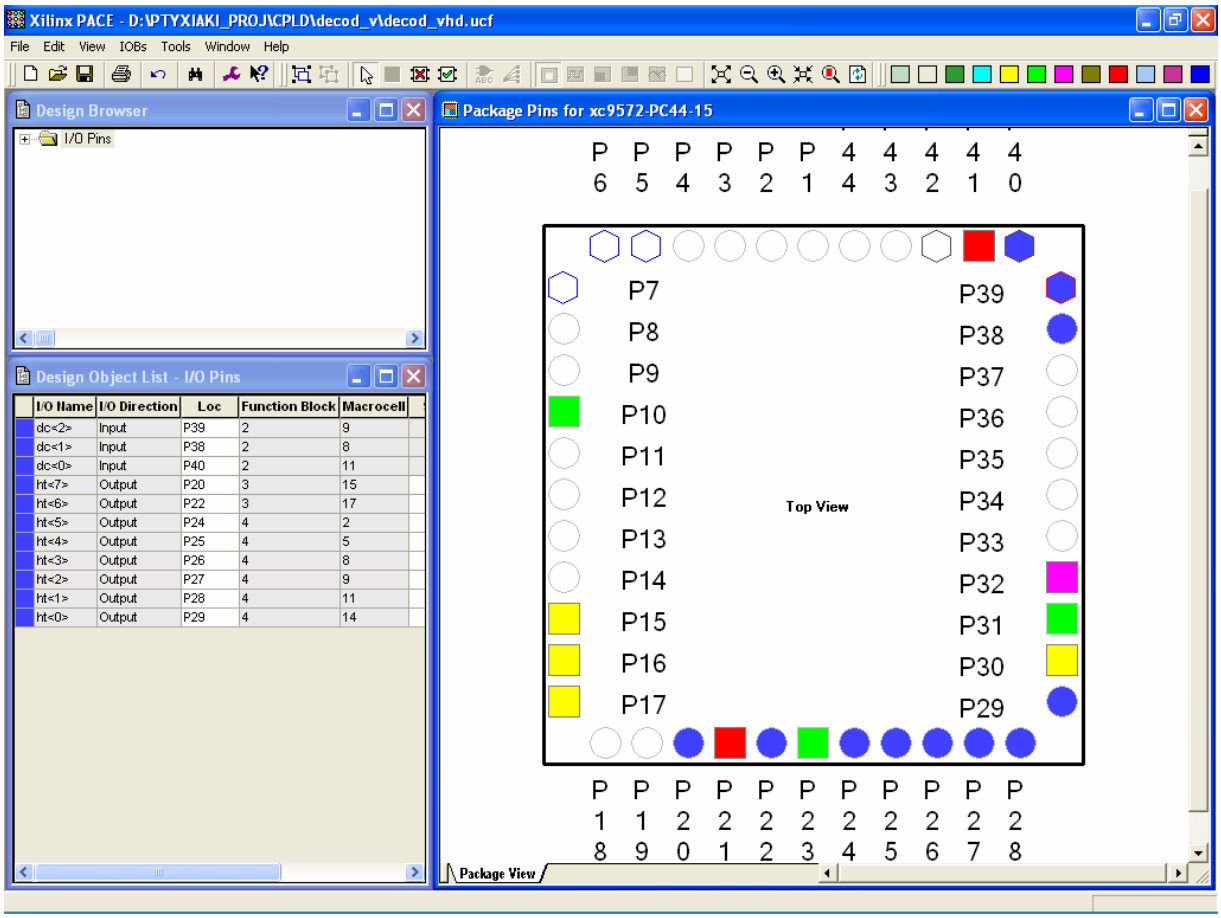


Εικόνα 1.41 VHDL αρχείο του κυκλώματος

Εικόνα 1.42 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.43 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.44 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

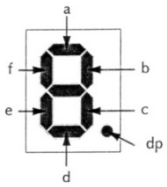
1.5 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ: ΜΕΤΑΤΡΟΠΕΑΣ ΚΩΔΙΚΑ (ΔΕΚΑΕΞΑΔΙΚΟΥ ΣΕ ΚΩΔΙΚΑ ΓΙΑ ΟΔΗΓΗΣΗ ΕΝΔΕΙΚΤΗ 7 ΤΟΜΕΩΝ)

1.5.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

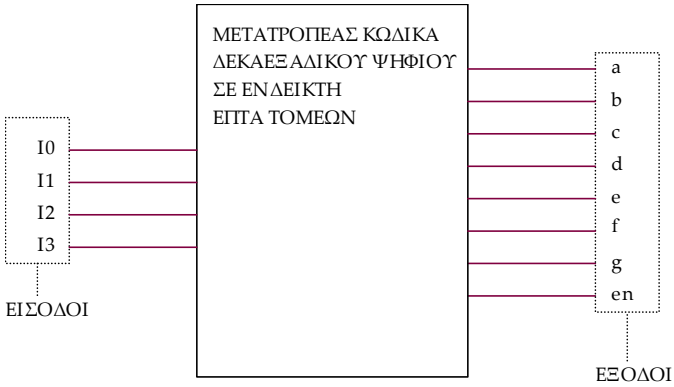
1.5.1.1 Σύντομη θεωρία

Ο μετατροπέας κώδικα (*code converter*) είναι το συνδυαστικό ψηφιακό κύκλωμα το οποίο μετατρέπει ένα κώδικα από μια μορφή σε μία άλλη. Στην συγκεκριμένη περίπτωση γίνεται μετατροπή δυαδικού κώδικα δεκαεξαδικού ψηφίου σε δυαδικό κώδικα για οδήγηση ενδείκτη επτά τομέων (*seven-segment code*). Η οδήγηση ενδείκτη επτά τομέων λοιπόν γίνεται με την χρήση σήματος για τον κάθε τομέα του ενδείκτη. Ο ενδείκτης LED επτά τομέων που υπάρχει στην πλακέτα του εργαστηρίου όπου υλοποιείται η συνολική εργασία μας (project) είναι κοινής ανόδου. Αυτό σημαίνει ό,τι για να ενεργοποιηθούν οι επτά τομείς φωτοδιόδου (LED) του ενδείκτη της πλακέτας χρειάζονται επτά σήματα σε δυναμικό λογικής στάθμης 0. Ο σχεδιαστής της πλακέτας επίσης μας πληροφορεί ό,τι για την ενεργοποίηση της ένδειξης απαιτείται η χρήση ενός επιπλέον σήματος σε δυναμικό λογικής στάθμης 0. Έχουμε λοιπόν τις τέσσερις εισόδους που συμβολίζονται με I0, I1, I2, I3 (MSB= I3). Οι έξοδοι είναι οκτώ και συμβολίζονται με a, b, c, d, e, f, g, en (MSB= a) όπως και οι τομείς του ενδείκτη με τους οποίους συνδέονται. Στην **εικόνα 1.45** του **πίνακα 1.17** υπάρχει ο ενδείκτης με τους επτά τομείς του και το γραφικό σύμβολο του μετατροπέα κώδικα. Στον **πίνακα 1.18** αντιστοιχίζονται οι δυαδικοί κώδικες εισόδου και εξόδου. Στον **πίνακα 1.19** φαίνεται ο πίνακας αλήθειας. Στους **πίνακες 1.20** και **1.21** προβάλονται οι πίνακες Karnaugh όπου μετά την απλοποίησή τους προκύπτει η λογική εξίσωση υλοποιημένη σε άθροισμα ελαχίστων όρων (minterm). (**Εικόνα 1.47**)

Πίνακας 1.17 Παράθεση εικόνων



Εικόνα 1.45 Ενδείκτης επτά τομέων με τον τομέα dp να μην χρησιμοποιείται



Εικόνα 1.46 Γραφικό σύμβολο του μετατροπέα κώδικα

Πίνακας 1.18 Αντιστοίχιση δυαδικών κωδικών εισόδου και εξόδου

ΔΥΑΔΙΚΟΣ ΚΩΔΙΚΑΣ ΕΙΣΟΔΟΥ ΔΕΚΑΕΞΑΔΙΚΟΥ ΨΗΦΙΟΥ				ΔΕΚΑΕΞΑΔΙΚΟ ΨΗΦΙΟ	ΚΩΔΙΚΑΣ ΕΞΟΔΟΥ ΟΔΗΓΗΣΗΣ ΕΝΔΕΙΚΤΗ ΕΠΤΑ ΤΟΜΕΩΝ ΚΟΙΝΗΣ ΑΝΟΔΟΥ								ΕΜΦΑΝΙΣΗ ΕΝΔΕΙΚΤΗ ΕΠΤΑ ΤΟΜΕΩΝ
MSB			**LSB**		**MSB**							**LSB**	
I3	**I2**	**I1**	**I0**		**a**	**b**	**c**	**d**	**e**	**f**	**g**	**en**	
0	0	0	0	0	0	0	0	0	0	0	1	0	
0	0	0	1	1	1	0	0	1	1	1	1	0	
0	0	1	0	2	0	0	1	0	0	1	0	0	
0	0	1	1	3	0	0	0	0	1	1	0	0	
0	1	0	0	4	1	0	0	1	1	0	0	0	
0	1	0	1	5	0	1	0	0	1	0	0	0	
0	1	1	0	6	0	1	0	0	0	0	0	0	
0	1	1	1	7	0	0	0	1	1	1	1	0	
1	0	0	0	8	0	0	0	0	0	0	0	0	
1	0	0	1	9	0	0	0	0	1	0	0	0	
1	0	1	0	A	0	0	0	1	0	0	0	0	
1	0	1	1	B	1	1	0	0	0	0	0	0	
1	1	0	0	C	0	1	1	0	0	0	1	0	
1	1	0	1	D	1	0	0	0	0	1	0	0	
1	1	1	0	E	0	0	1	0	0	0	0	0	
1	1	1	1	F	0	1	1	1	0	0	0	0	

Πίνακας 1.19 Πίνακας αλήθειας του μετατροπέα κώδικα

ΕΙΣΟΔΟΙ				ΕΞΟΔΟΙ							
MSB			**LSB**	**MSB**							**LSB**
I3	**I2**	**I1**	**I0**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**en**
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	1	1	1	0
0	0	1	0	0	0	1	0	0	1	0	0
0	0	1	1	0	0	0	0	1	1	0	0
0	1	0	0	1	0	0	1	1	0	0	0
0	1	0	1	0	1	0	0	1	0	0	0
0	1	1	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	1	0	0	0	0
1	0	1	1	1	1	0	0	0	0	0	0
1	1	0	0	0	1	1	0	0	0	1	0
1	1	0	1	1	0	0	0	0	1	0	0
1	1	1	0	0	0	1	0	0	0	0	0
1	1	1	1	0	1	1	1	0	0	0	0

Πίνακας 1.20 Πίνακες Karnaugh

Έξοδος: a

I2I0 \ I3I2	00	01	11	10
00	0	1	0	0
01	1	0	1	0
11	0	0	0	1
10	0	0	0	0

Έξοδος: b

I2I0 \ I3I2	00	01	11	10
00	0	0	1	0
01	0	1	0	0
11	0	0	1	1
10	0	1	0	0

Έξοδος: c

I2I0 \ I3I2	00	01	11	10
00	0	0	1	0
01	0	0	0	0
11	0	0	1	0
10	1	0	1	0

Έξοδος: d

I2I0 \ I3I2	00	01	11	10
00	0	1	0	0
01	1	0	0	0
11	0	1	1	0
10	0	0	0	1

Έξοδος: e

I2I0 \ I3I2	00	01	11	10
00	0	1	0	0
01	1	1	0	1
11	1	1	0	0
10	0	0	0	0

Έξοδος: f

I2I0 \ I3I2	00	01	11	10
00	0	0	0	0
01	1	0	1	0
11	1	1	0	0
10	1	0	0	0

Πίνακας 1.21 Πίνακες Karnaugh (συνέχεια)

Έξοδος: g

I2I0 \ I3I2	00	01	11	10
00	1	0	1	0
01	1	0	0	0
11	0	1	0	0
10	0	0	0	0

Έξοδος: en

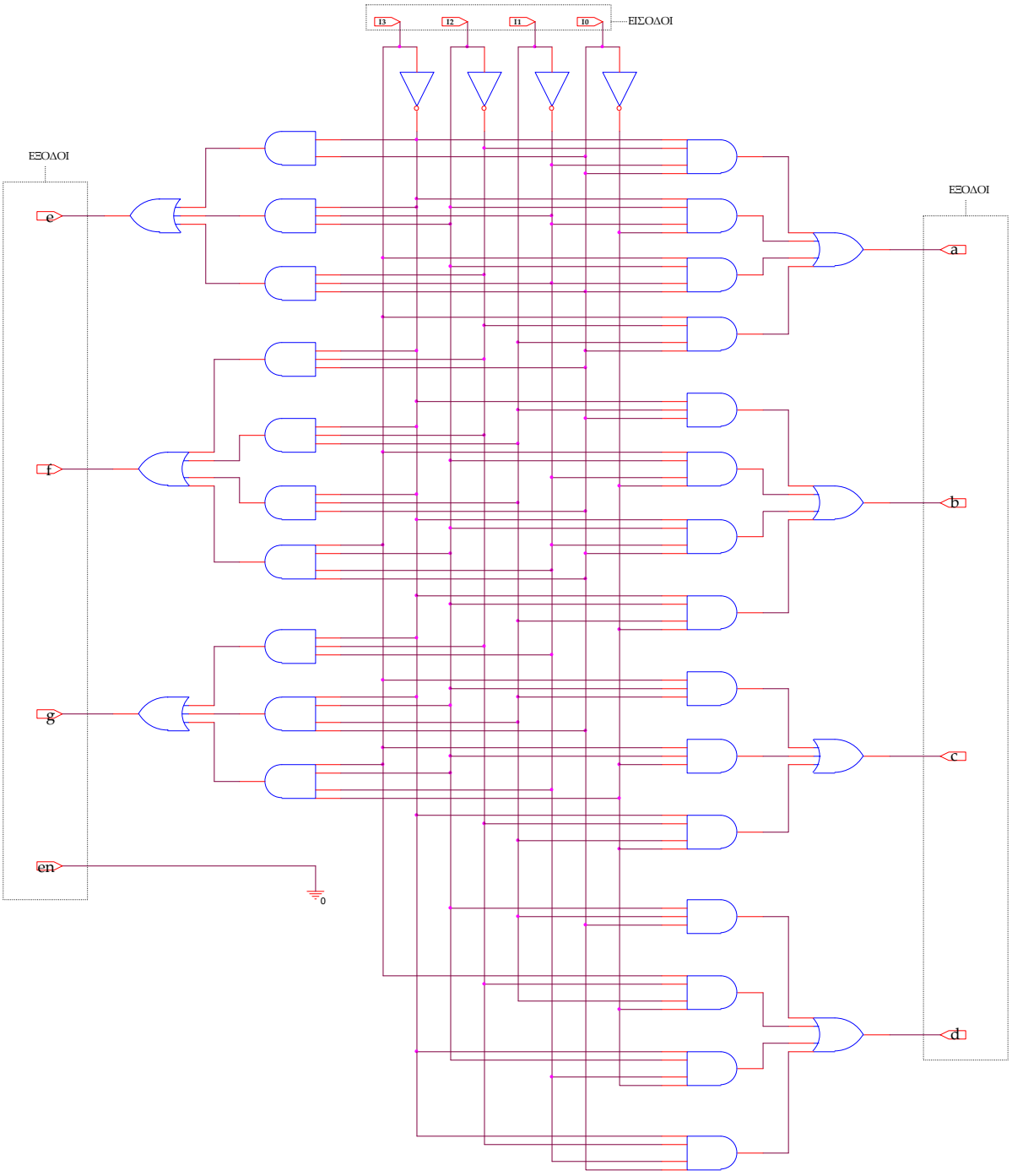
I2I0 \ I3I2	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

$$a = \overline{I3}\cdot\overline{I2}\cdot\overline{I1}\cdot I0 + \overline{I3}\cdot I2\cdot\overline{I1}\cdot\overline{I0} + I3\cdot I2\cdot\overline{I1}\cdot I0 + I3\cdot\overline{I2}\cdot I1\cdot I0$$
$$b = I3\cdot I1\cdot I0 + I3\cdot I2\cdot\overline{I1}\cdot\overline{I0} + \overline{I3}\cdot I2\cdot\overline{I1}\cdot I0 + \overline{I3}\cdot I2\cdot I1\cdot\overline{I0}$$
$$c = I3\cdot I2\cdot I1 + I3\cdot I2\cdot\overline{I0} + \overline{I3}\cdot\overline{I2}\cdot I1\cdot\overline{I0}$$
$$d = I2\cdot I1\cdot I0 + I3\cdot\overline{I2}\cdot I1\cdot\overline{I0} + \overline{I3}\cdot I2\cdot\overline{I1}\cdot\overline{I0} + \overline{I3}\cdot\overline{I2}\cdot\overline{I1}\cdot I0$$
$$e = \overline{I3}\cdot I0 + \overline{I3}\cdot I2\cdot\overline{I1} + \overline{I2}\cdot\overline{I1}\cdot I0$$
$$f = \overline{I3}\cdot\overline{I2}\cdot I0 + \overline{I3}\cdot\overline{I2}\cdot I1 + \overline{I3}\cdot I1\cdot I0 + I3\cdot I2\cdot\overline{I1}\cdot I0$$
$$g = \overline{I3}\cdot\overline{I2}\cdot\overline{I1} + \overline{I3}\cdot I2\cdot I1\cdot I0 + I3\cdot I2\cdot\overline{I1}\cdot\overline{I0},\ en = 0.$$

Εικόνα 1.47 Λογική εξίσωση συγκεκριμένου μετατροπέα κώδικα σε ανάπτυξη ελαχίστων όρων (minterm)

1.5.1.2 Συμβατική ψηφιακή υλοποίηση

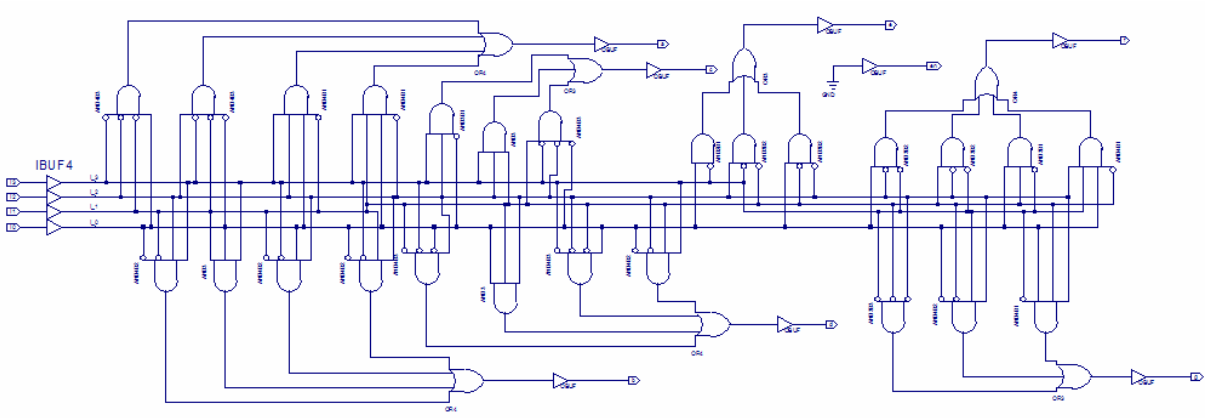
Η παραπάνω λογική εξίσωση παραπέμπει στο παρακάτω κύκλωμα το υλοποιήσιμο με συμβατικές πύλες.



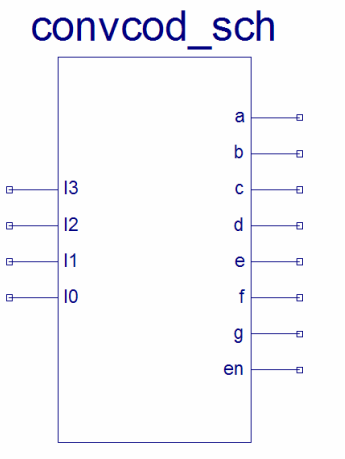
Εικόνα 1.48 Λογικό κύκλωμα μετατροπέα κώδικα δεκαεξαδικού σε ενδείκτη 7 τομέων**.**

1.5.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

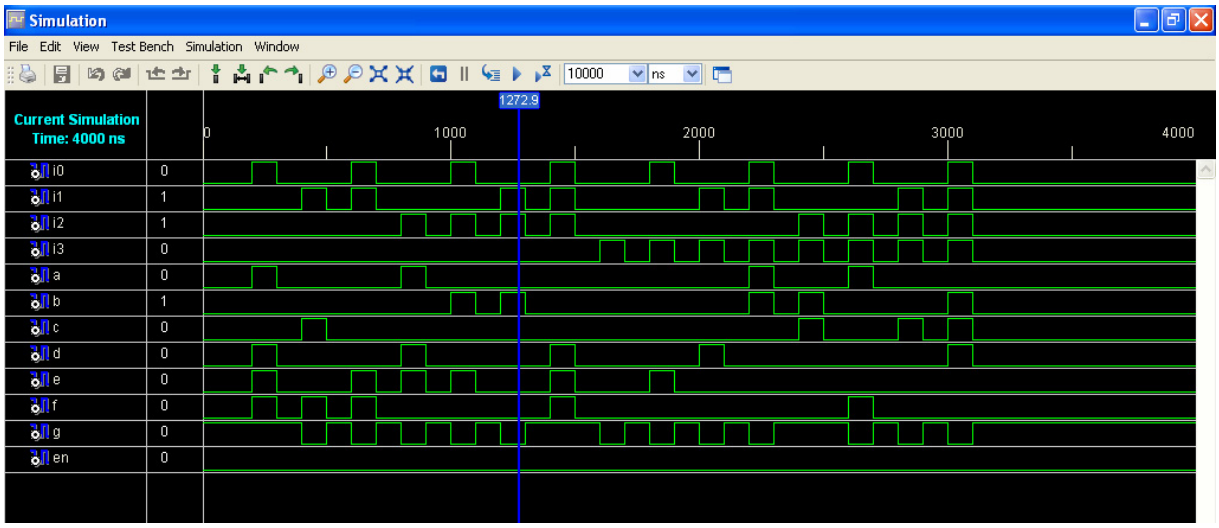
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα. Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



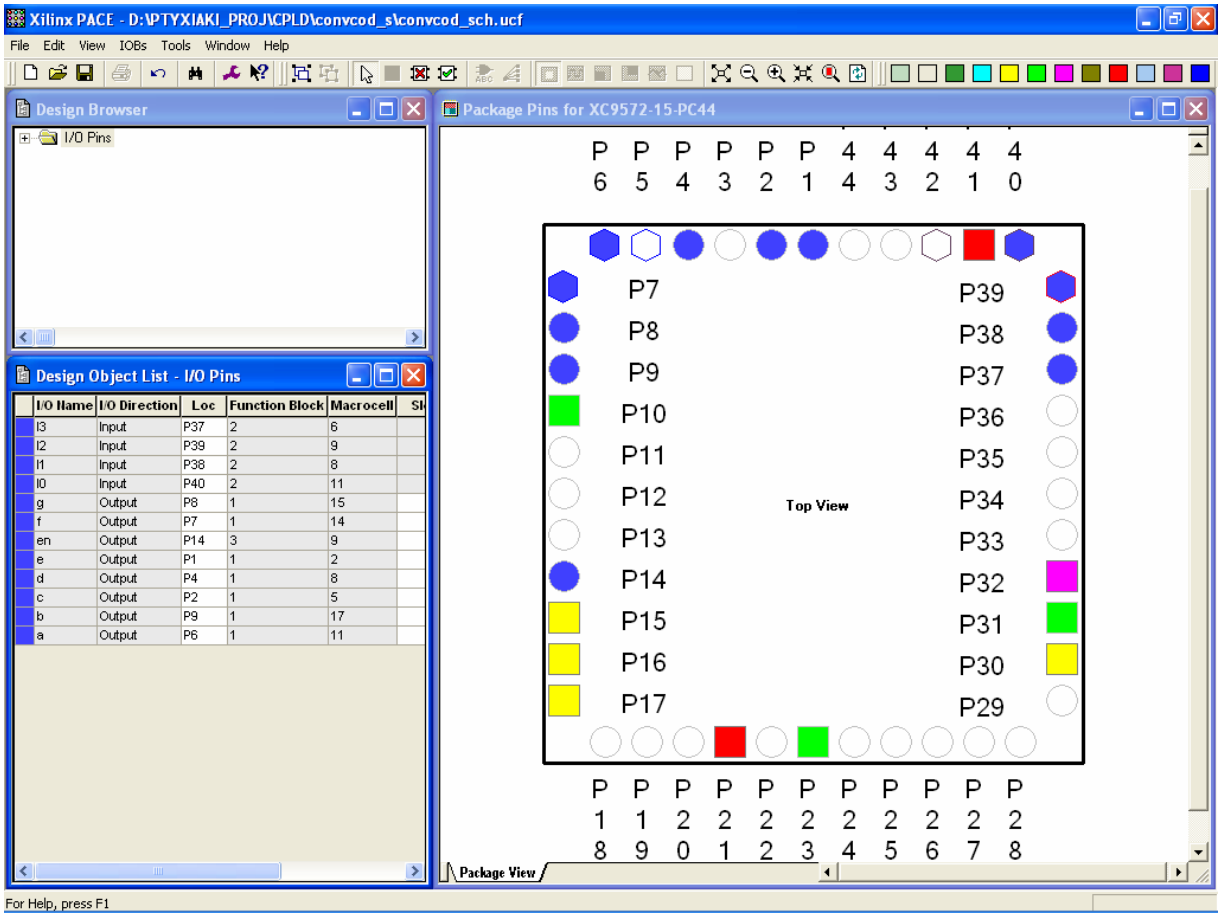
Εικόνα 1.49 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 1.50 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.51 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.52 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

1.5.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 και αναφέρονται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 1.21 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

```
12 architecture BHV_DFL of convcod _vhd is
13 begin
14        with i select
15        LED <= "11110010" when  = "0001",
16                "01001000" when  = "0010",
17                "01100000" when  = "0011",
18                "00110010" when  = "0100",
19                "00100100" when  = "0101",
20                "00000100" when  = "0110",
21                "11110000" when  = "0111",
22                "00000000" when  = "1000",
23                "00100000" when  = "1001",
24                "00010000" when  = "1010",
25                "00000110" when  = "1011",
26                "10001100" when  = "1100",
27                "01000010" when  = "1101",
28                "00001100" when  = "1110",
29                "00011100" when  = "1111",
30                "10000000" when  others;
31  end BHV_DFL;
```

Αρχή δήλωσης αρχιτεκτονικής.
Αρχή κώδικα αρχιτεκτονικής.
Εκτέλεση εντολής **with-select-when**.
Επανάληψη μηχανισμού που είδαμε στη συνολική εργασία του αποκωδικοποιητή 3 σε 8.

Στην περίπτωση εισαγωγής στην είσοδο **i** μη προβλεπόμενου συνδυασμού τιμής από τις παραπάνω, δηλώνεται (με τη λέξη when others της γραμμής 30) ό,τι η έξοδος **LED** παίρνει την δυαδική τιμή 10000000 όπως ορίζει επίσης η γραμμή 30.

Τέλος κώδικα αρχιτεκτονικής.

Σημείωση. Η αρχιτεκτονική περιέχει μόνο συντρέχοντα κώδικα (data flow code) που υλοποιείται μέσω της εντολής **with-select-when**. Ο συντρέχοντας κώδικας δηλώνεται με τη συντμημένη λέξη **DFL** μέσα στο όνομα της αρχιτεκτονικής BHV_**DFL**.

Πίνακας 1.22 Περιοχή δήλωσης σχολίων (comments)

14 -- g f e d c b a en, αρνητική λογική
Γρήγορος συμβολισμός από την διατεταγμένη αντιστοίχιση:
g στο **bit 7** της εξόδου **LED** ή **LED(7)**, **f** στο **bit 6** της εξόδου **LED** ή **LED(6)**,
e στο **bit 5** της εξόδου **LED** ή **LED(5)**, **d** στο **bit 4** της εξόδου **LED** ή **LED(4)**,
c στο **bit 3** της εξόδου **LED** ή **LED(3)**, **b** στο **bit 2** της εξόδου **LED** ή **LED(2)**,
a στο **bit 1** της εξόδου **LED** ή **LED(1)**, **en** στο **bit 0** της εξόδου **LED** ή **LED(0)**.
Υποστηρίζεται η αρνητική λογική, δηλαδή η ενεργοποίηση του κάθε σήματος γίνεται με δυναμικό λογικής στάθμης 0.
15 -- 1
Ένδειξη του ψηφίου 1 στον ενδείκτη όταν η έξοδος **LED** πάρει την δυαδική τιμή 11110010 κατά την εκτέλεση της εντολής στην γραμμή 15.
Σημείωση. Η εξήγηση των σχολίων των γραμμών 16 ως 30 είναι ανάλογη με αυτή της γραμμής 15.

Όλα τα παραπάνω αφορούν συντρέχοντα κώδικα που δεν περιγράφεται ποιοτικά από διαγράμματα ροής (flow charts) όπως ο ακολουθιακός κώδικας. Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

1.5.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

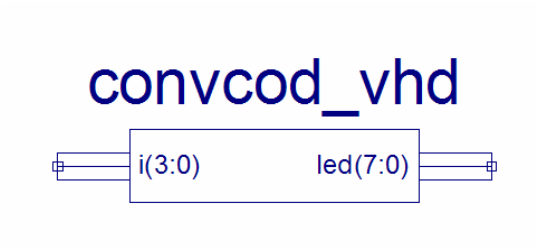
convcod_vhd.vhd

File Edit View Window

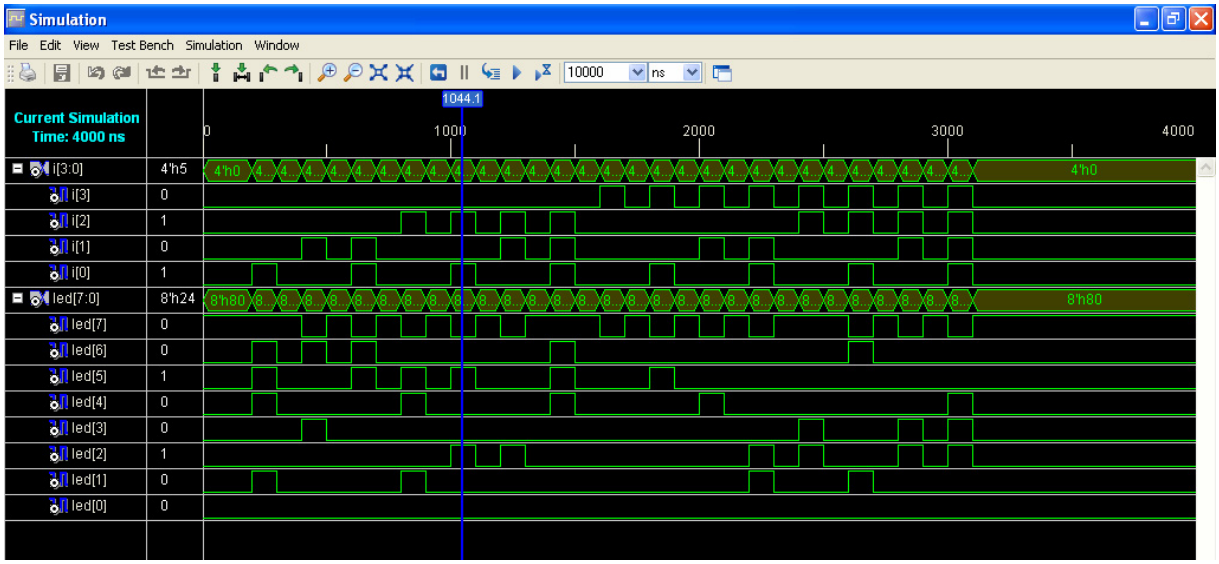
```
1  ----------------------------------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;
6  ----------------------------------------------------------------------------
7  entity convcod_vhd is
8      Port ( i : in  STD_LOGIC_VECTOR (3 downto 0);
9             led : out  STD_LOGIC_VECTOR (7 downto 0));
10 end convcod_vhd;
11 ----------------------------------------------------------------------------
12 architecture BHV_DFL of convcod_vhd is
13 begin
14    with i select                    --g f e d c b a en, αρνητική λογική
15       LED<= "11110010" when "0001",   --1
16             "01001000" when "0010",   --2
17             "01100000" when "0011",   --3
18             "00110010" when "0100",   --4
19             "00100100" when "0101",   --5
20             "00000100" when "0110",   --6
21             "11110000" when "0111",   --7
22             "00000000" when "1000",   --8
23             "00100000" when "1001",   --9
24             "00010000" when "1010",   --A
25             "00000110" when "1011",   --b
26             "10001100" when "1100",   --C
27             "01000010" when "1101",   --d
28             "00001100" when "1110",   --E
29             "00011100" when "1111",   --F
30             "10000000" when others;   --0
31 end BHV_DFL;
```

CAPS NUM SCRL Ln 6 Col 74 VHDL

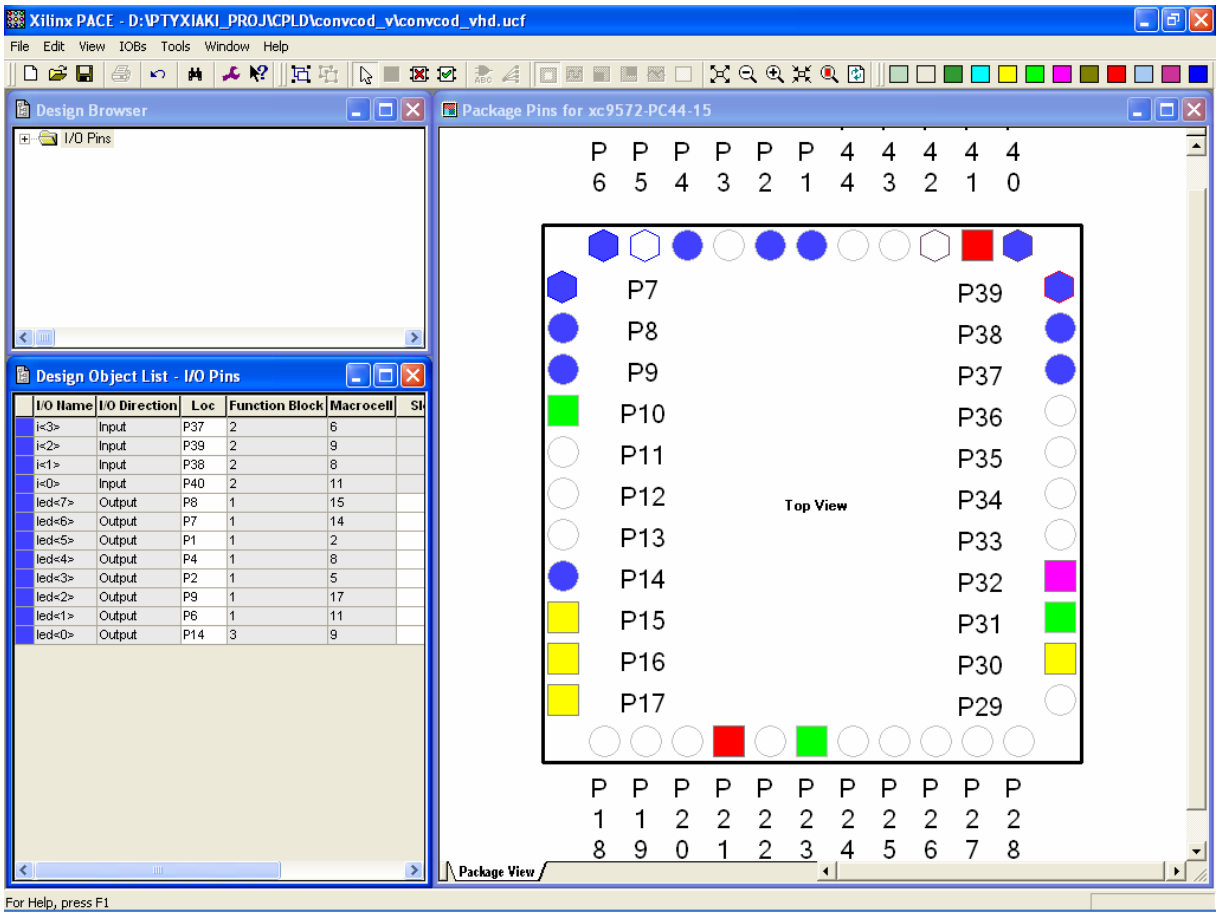
Εικόνα 1.53 VHDL αρχείο του κυκλώματος



Εικόνα 1.54 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.55 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.56 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

1.6 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ: ΣΥΓΚΡΙΤΗΣ ΜΕΓΕΘΟΥΣ ΠΡΟΣΗΜΑΣΜΕΝΩΝ ΑΡΙΘΜΩΝ (ΤΕΣΣΑΡΩΝ ΨΗΦΙΩΝ)

1.6.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

1.6.1.1 Σύντομη θεωρία

Ο συγκριτής μεγέθους (*magnitude comparator*) είναι το συνδυαστικό ψηφιακό κύκλωμα το οποίο συγκρίνει δύο δυαδικές ψηφιολέξεις ***K***=K(n-1)…K1K0 και ***M***=M(n-1)…M1M0 (*n* ψηφίων) που εφαρμόζονται στις δύο εισόδους του και ενεργοποιούν μία από τις τρεις εξόδους του (***K>M***, ***K=M*** και ***K<M***). Θα μελετήσουμε λοιπόν ένα συγκριτή μεγέθους που συγκρίνει δύο δυαδικές ψηφιολέξεις 4 ψηφίων που αντιστοιχούν σε προσημασμένους αριθμούς κατά το **συμπλήρωμα του 2** (**2's**). Στην πρώτη είσοδο εισάγεται ο τετραψήφιος προσημασμένος αριθμός ***A***= A3A2A1A0 (MSB=A3, αν A3=0 ο ***A*** είναι θετικός αριθμός διαφορετικά αρνητικός). Στην δεύτερη είσοδο εισάγεται ο τετραψήφιος προσημασμένος αριθμός ***B***= B3B2B1B0 (MSB=B3, αν B3=0 ο ***B*** είναι θετικός αριθμός διαφορετικά αρνητικός). Την έξοδο ***A>B*** (που σημαίνει ό,τι η είσοδος ***A*** είναι μεγαλύτερη από την είσοδο ***B***) την ονομάζουμε **MEG**. Παρομοίως την έξοδο ***A=B*** (η είσοδος ***A*** είναι ίση με την είσοδο ***B***) την ονομάζουμε **ISO** και την έξοδο ***A<B*** (η είσοδος ***A*** είναι μικρότερη από την είσοδο ***B***) την ονομάζουμε **MIK.** Στην **εικόνα 1.57** φαίνεται το γραφικό σύμβολο του συγκριτή μεγέθους. Στον **πίνακα 1.23** υπάρχει η αντιστοίχιση των γεγονότων σύγκρισης σε συμβατική λογική. Στον **πίνακα 1.24** υπάρχει ο πίνακας αλήθειας του συγκεκριμένου συγκριτή. Στην **εικόνα 1.57** βρίσκεται η λογική εξίσωση υλοποιημένη σε άθροισμα ελαχίστων όρων (minterm).

Εικόνα 1.57 Γραφικό σύμβολο του συγκριτή μεγέθους τεσσάρων ψηφίων

Πίνακας 1.23 Αντιστοίχιση γεγονότων σύγκρισης σε συμβατική λογική

AN	**BN**	**(AN>BN)**	**(AN<BN)**	**(AN=BN)**
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Οπότε ισχύει:

$$(AN = BN) = AN \cdot BN + \overline{AN} \cdot \overline{BN} = \overline{AN \oplus BN},$$

$$(AN < BN) = \overline{AN} \cdot BN,\ (AN > BN) = AN \cdot \overline{BN}$$

AN: Ν δυαδικό ψηφίο αριθμού ***A*** ,
BN: Ν δυαδικό ψηφίο αριθμού ***B***
(AN>BN): Γεγονός «AN μεγαλύτερο από BN»
(AN<BN): Γεγονός «AN μικρότερο από BN»
(AN=BN): Γεγονός «AN ίσο με BN»

Πίνακας 1.24 Πίνακας αλήθειας για τη σύγκριση προσημασμένων αριθμών τεσσάρων ψηφίων

	ΕΙΣΟΔΟΙ										
	A3	**A2**	**A1**	**A0**	**B3**	**B2**	**B1**	**B0**			
	MSB			**LSB**	**MSB**			**LSB**	ΕΞΟΔΟΙ		
ΔΕΚΑΔΙΚΟΣ ΑΡΙΘΜΟΣ	A3, B3		A2, B2		A1, B1		A0, B0		**MEG**	**ISO**	**MIK**
0	(A3<B3)		X		X		X		1	0	0
1	(A3>B3)		X		X		X		0	0	1
2	(A3=B3)		(A2>B2)		X		X		1	0	0
3	(A3=B3)		(A2<B2)		X		X		0	0	1
4	(A3=B3)		(A2=B2)		(A1>B1)		X		1	0	0
5	(A3=B3)		(A2=B2)		(A1<B1)		X		0	0	1
6	(A3=B3)		(A2=B2)		(A1=B1)		(A0>B0)		1	0	0
7	(A3=B3)		(A2=B2)		(A1=B1)		(A0<B0)		0	0	1
8	(A3=B3)		(A2=B2)		(A1=B1)		(A0=B0)		0	1	0

X: δηλώνει αδιάφορη λογική κατάσταση.

Οι αριθμοί είναι προσημασμένοι με το MSB να είναι και ψηφίο προσήμου (0 για θετικό, 1 για αρνητικό προσημασμένο αριθμό).

Οι ανισότητες και ισότητες στην παρένθεση παραπέμπουν σε γεγονότα και λογικές σχέσεις στον **πίνακα 1.24**. Το Ν παίρνει τις συγκεκριμένες τιμές ανά περίπτωση όπως στις παρενθέσεις εντός του παρόντος πίνακα.

$$\mathrm{MEG} = \sum(0,2,4,6) = (\mathrm{A3} < \mathrm{B3}) + ((\mathrm{A3} = \mathrm{B3}) \cdot (\mathrm{A2} > \mathrm{B2})) + ((\mathrm{A3} = \mathrm{B3}) \cdot (\mathrm{A2} = \mathrm{B2}) \cdot (\mathrm{A1} > \mathrm{B1})) +$$
$$+((\mathrm{A3} = \mathrm{B3}) \cdot (\mathrm{A2} = \mathrm{B2}) \cdot (\mathrm{A1} = \mathrm{B1}) \cdot (\mathrm{A0} > \mathrm{B0})) = (\overline{\mathrm{A3}} \cdot \mathrm{B3}) + ((\overline{\mathrm{A3} \oplus \mathrm{B3}}) \cdot (\mathrm{A2} \cdot \overline{\mathrm{B2}})) + ((\overline{\mathrm{A3} \oplus \mathrm{B3}}) \cdot$$
$$\cdot(\overline{\mathrm{A2} \oplus \mathrm{B2}}) \cdot (\mathrm{A1} \cdot \overline{\mathrm{B1}})) + ((\overline{\mathrm{A3} \oplus \mathrm{B3}}) \cdot (\overline{\mathrm{A2} \oplus \mathrm{B2}}) \cdot (\overline{\mathrm{A1} \oplus \mathrm{B1}}) \cdot (\mathrm{A0} \cdot \overline{\mathrm{B0}}))$$

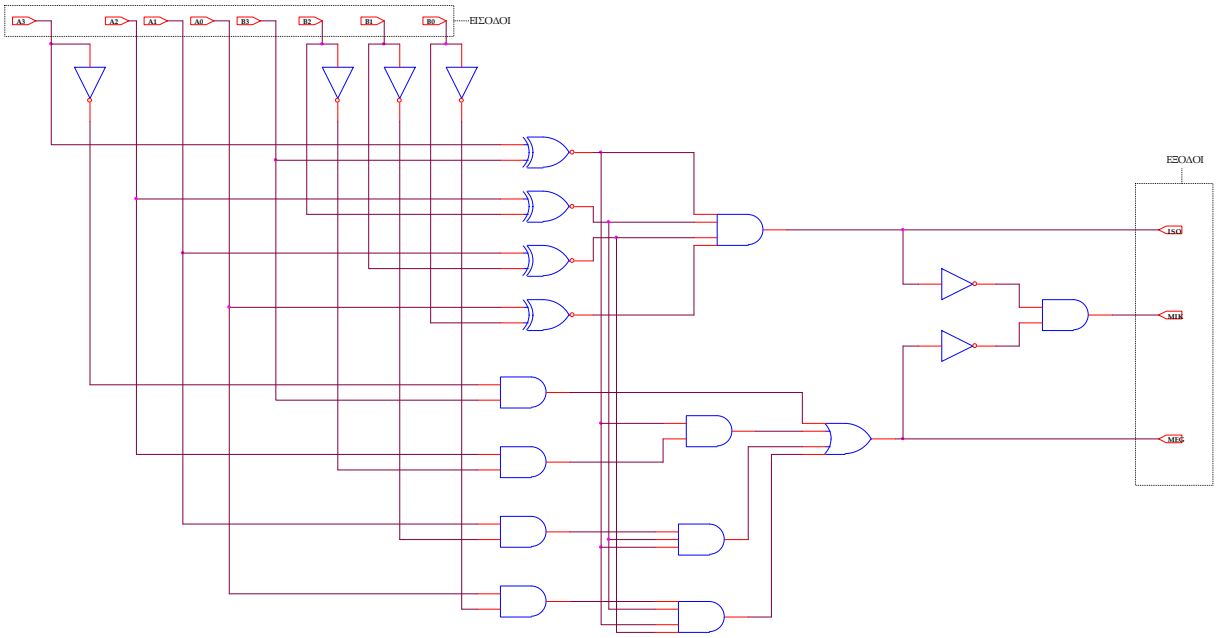
$$\mathrm{ISO} = \sum(8) = ((\mathrm{A3} = \mathrm{B3}) \cdot (\mathrm{A2} = \mathrm{B2}) \cdot (\mathrm{A1} = \mathrm{B1}) \cdot (\mathrm{A0} = \mathrm{B0})) = ((\overline{\mathrm{A3} \oplus \mathrm{B3}}) \cdot (\overline{\mathrm{A2} \oplus \mathrm{B2}}) \cdot (\overline{\mathrm{A1} \oplus \mathrm{B1}}) \cdot$$
$$\cdot(\overline{\mathrm{A0} \oplus \mathrm{B0}}))$$

Για λόγους συντομίας: $\mathrm{MIK} = \overline{\mathrm{MEG} + \mathrm{ISO}} = \overline{\mathrm{MEG}} \cdot \overline{\mathrm{ISO}}$

Εικόνα 1.58 Λογική εξίσωση του συγκριτή μεγέθους σε ανάπτυξη ελαχίστων όρων (minterm)

1.6.1.2 Συμβατική ψηφιακή υλοποίηση

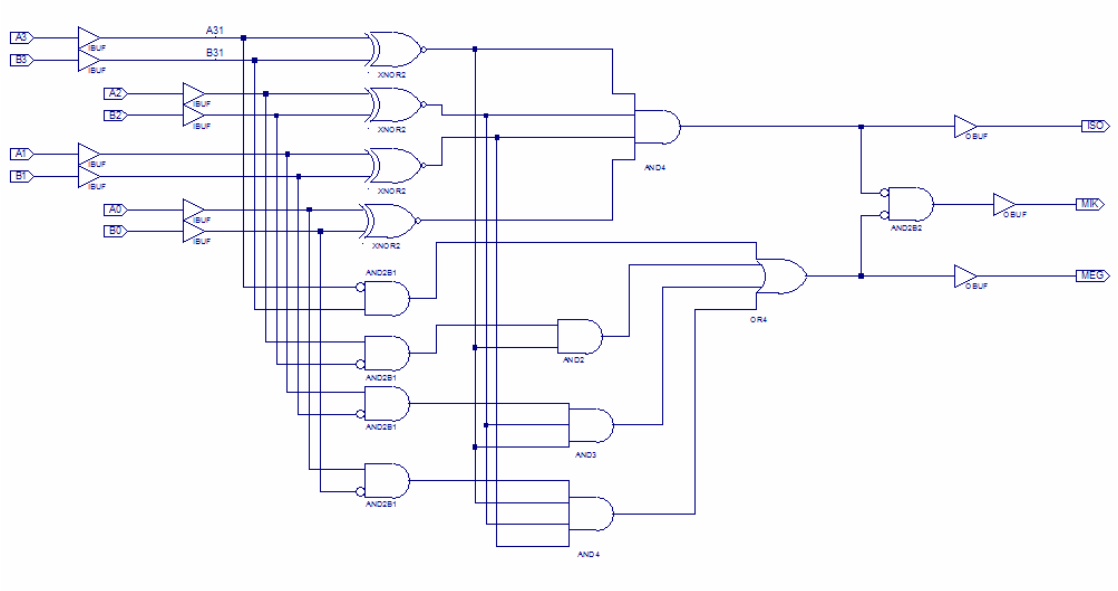
Η παραπάνω λογική εξίσωση παραπέμπει στο παρακάτω κύκλωμα το υλοποιήσιμο με συμβατικές πύλες.



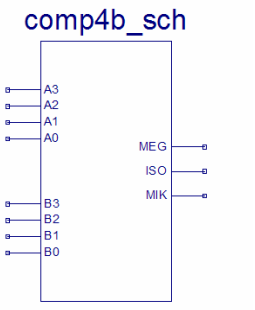
Εικόνα 1.59 Λογικό κύκλωμα υλοποίησης του συγκριτή μεγέθους

1.6.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

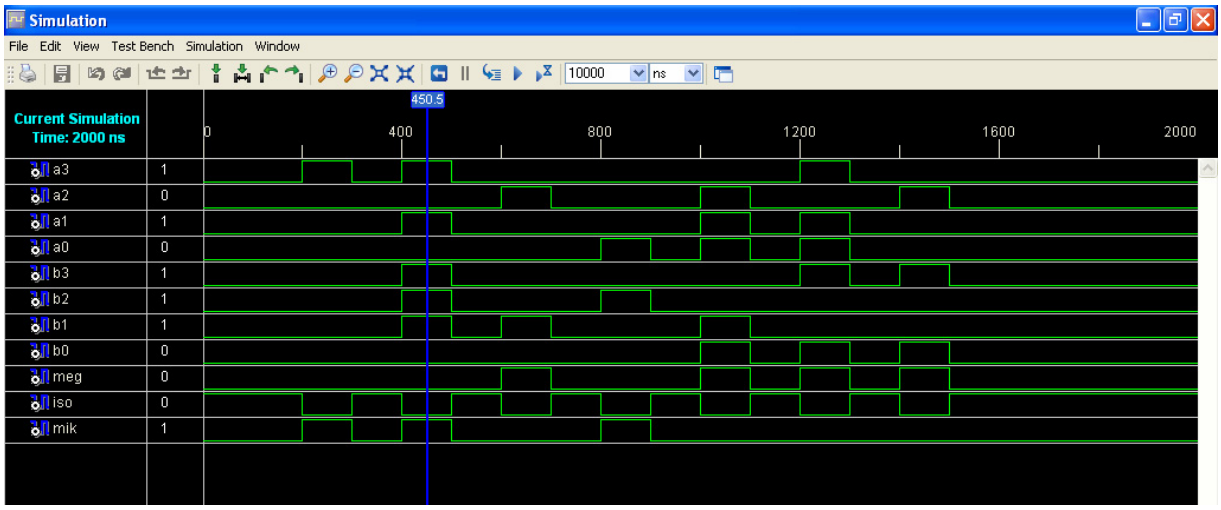
Όλες οι διαδικασίες που αφορούν από τη δημιουργία νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα. Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



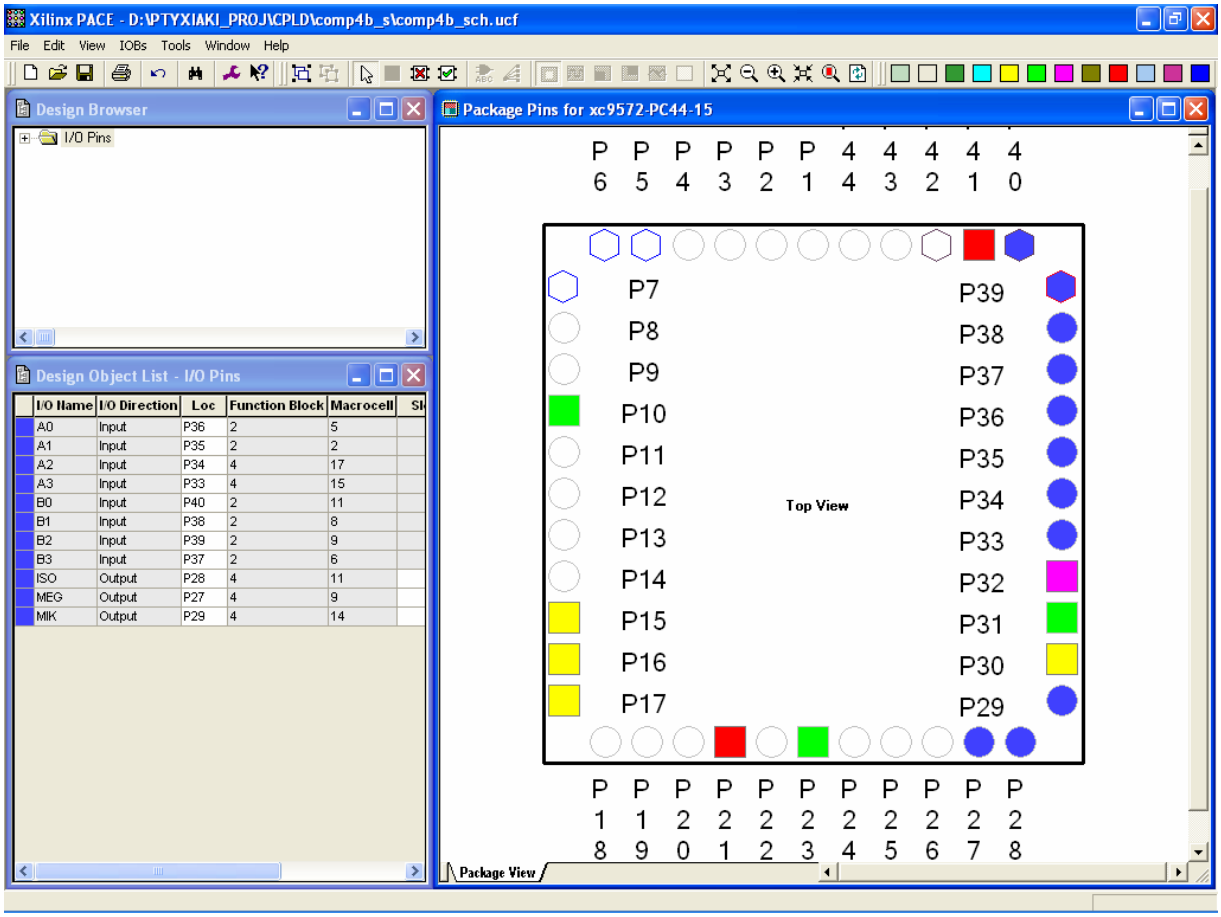
Εικόνα 1.60 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 1.61 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.62 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.63 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

1.6.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Πίνακας 1.25 Περιοχή δήλωσης βιβλιοθηκών (library)

1	----------	Σχόλια.
2	library IEEE;	Δήλωση βιβλιοθήκης με όνομα **ieee**.
3	use IEEE.STD_LOGIC_1164.ALL;	Δήλωση χρήσης όλων των στοιχείων του πακέτου **std_logic_1164** της βιβλιοθήκης **ieee**.
4	use IEEE.STD_LOGIC_ARITH.ALL;	Δήλωση χρήσης όλων των στοιχείων του πακέτου **std_logic_arith** της βιβλιοθήκης **ieee**.
5	use IEEE.STD_LOGIC_SIGNED.ALL;	Δήλωση χρήσης όλων των στοιχείων του πακέτου **std_logic_signed** της βιβλιοθήκης **ieee**.
6	----------	Σχόλια.

Σημείωση. Γραμμή 3: η δήλωση του πακέτου **std_logic_1164** είναι άκρως απαραίτητη για χρήση δεδομένων **std_logic** και **std_logic_vector**.
Γραμμή 4: η δήλωση του πακέτου **std_logic_arith** είναι άκρως απαραίτητη για χρήση δεδομένων τύπου **signed** που θα χρησιμοποιήσουμε στη συνέχεια. (Το πακέτο **std_logic_arith** υποστηρίζεται από την **Xilinx®** στην βιβλιοθήκη **ieee** αλλά πρόκειται για εμπορικό πακέτο της εταιρείας **Synopsys®**.)
Γραμμή 5: η δήλωση του πακέτου **std_logic_signed** είναι άκρως απαραίτητη για χρήση δεδομένων τύπου **std_logic_vector** για να μπορούν να διαχειριστούν σαν **signed**. Είναι συμπλήρωμα του πακέτου **std_logic_arith** για δεδομένα **std_logic_vector** και παρέχεται από την εταιρεία **Synopsys®**. Δεν είναι απαραίτητη η χρήση του στην συγκεκριμένη συνολική εργασία (prolect).
Οι εντολές των γραμμών 02, 03, 04, 05 βρίσκονται εξ' ορισμού από το πρόγραμμα σε όλες τις συνολικές εργασίες (projects).
Το ερωτηματικό δηλώνει το τέλος της εντολής.
Τα σχόλια στις γραμμές 01, 06 και 12 χρησιμοποιούνται για να ξεχωρίζουν τις περιοχές δήλωσης βιβλιοθηκών, οντότητας και αρχιτεκτονικής μεταξύ τους.

Πίνακας 1.26 Περιοχή δήλωσης οντότητας (entity)

7	entity comp4b_vhd is	Αρχή δήλωσης οντότητας με όνομα **comp4b_vhd**.
8	Port (a : in SIGNED (3 downto 0);	Αρχή δήλωσης θυρών (**ports**) με την πρώτη θύρα να δηλώνεται με όνομα **a** σαν είσοδος (**in**) σε μορφή **signed** (προσημασμένοι αριθμοί) τεσσάρων δυαδικών ψηφίων.
9	b : in SIGNED (3 downto 0);	Δήλωση δεύτερης θύρας με όνομα **b** σαν είσοδος (**in**) σε μορφή **signed** (προσημασμένοι αριθμοί) τεσσάρων δυαδικών ψηφίων.
10	mik, meg, iso : out STD_LOGIC);	Δήλωση άλλων τριών θυρών με ονόματα αντίστοιχα **mik**, **meg**, **iso** σαν έξοδος (**out**) σε μορφή **std_logic** ενός δυαδικού ψηφίου.
11	end comp4b_vhd;	Τέλος δήλωσης οντότητας με όνομα **comp4b_vhd**.
12	----------	Σχόλια.

Σημείωση. Τα δεδομένα τύπου **signed** αντιμετωπίζονται σαν δεδομένα τύπου **std_logic_vector** ως προς την σύνταξή τους. Χρησιμοποιούνται για παράσταση προσημασμένων αριθμών σε μορφή συμπλήρωματος του **2** (**2's**). Στα δεδομένα τύπου **signed** επιτρέπονται μόνο αριθμητικές πράξεις ενώ στα δεδομένα τύπου **std_logic_vector** επιτρέπονται μόνο λογικές πράξεις.

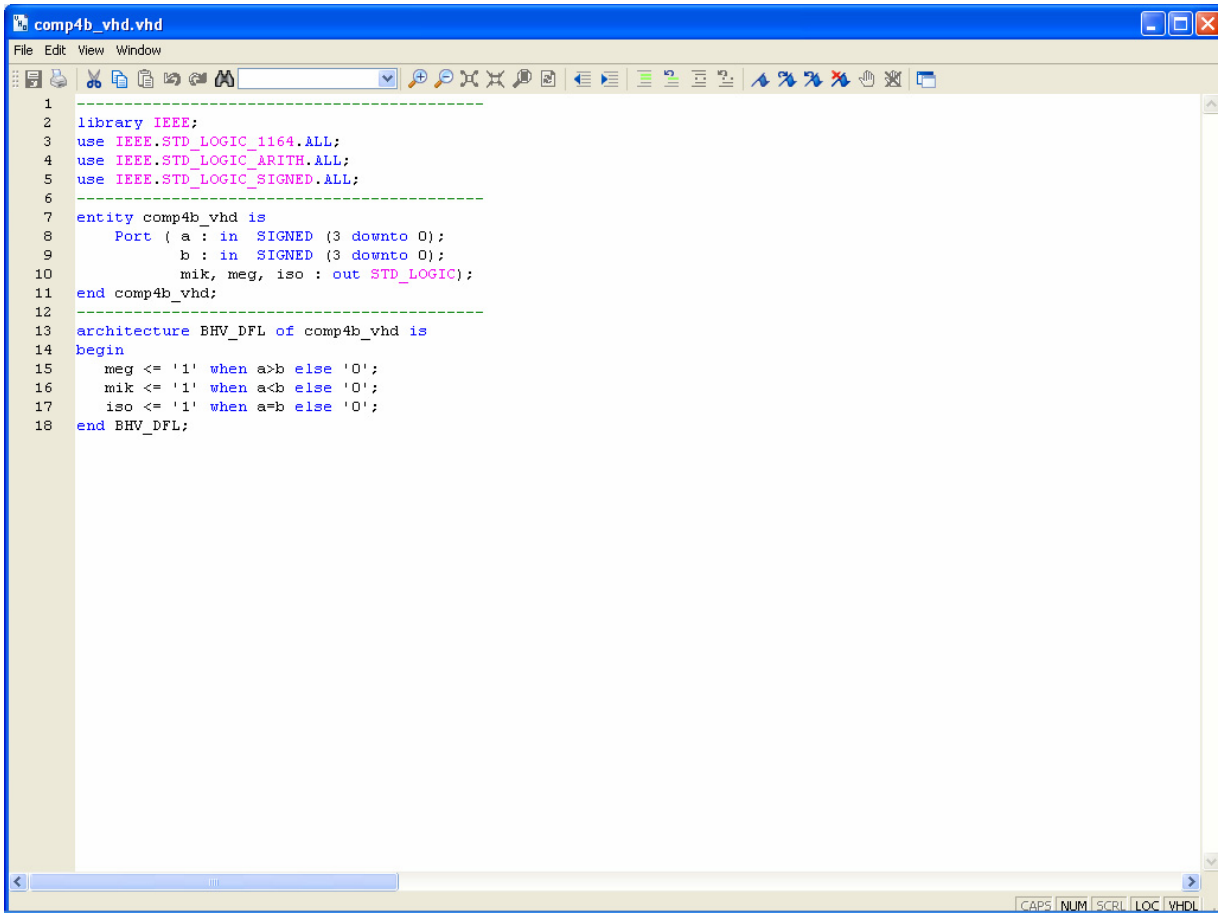
Πίνακας 1.27 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

13	architecture BHV_DFL of comp4b_vhd is	Αρχή δήλωσης αρχιτεκτονικής.
14	begin	Αρχή κώδικα αρχιτεκτονικής.
15	meg <= '1' when a>b else '0';	Εκτέλεση εντολής **when-else**. Αν το αποτέλεσμα του Boolean ελέγχου για την συνθήκη (a>b) είναι **true** μεταφέρεται η δυαδική τιμή 1 στην έξοδο **meg**, διαφορετικά μεταφέρεται η τιμή 0.
16	mik <= '1' when a<b else '0';	Αν το αποτέλεσμα του Boolean ελέγχου για την συνθήκη (a<b) είναι **true** μεταφέρεται η δυαδική τιμή 1 στην έξοδο **mik**, διαφορετικά μεταφέρεται η τιμή 0.
17	iso <= '1' when a=b else '0';	Αν το αποτέλεσμα του Boolean ελέγχου για την συνθήκη (a=b) είναι **true** μεταφέρεται η δυαδική τιμή 1 στην έξοδο **iso**, διαφορετικά μεταφέρεται η τιμή 0.
18	end BHV_DFL;	Τέλος κώδικα αρχιτεκτονικής.

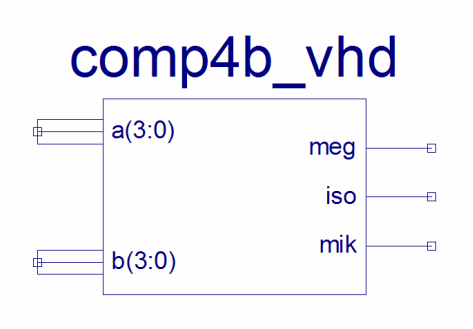
Σημείωση. Η αρχιτεκτονική περιέχει μόνο συντρέχοντα κώδικα (data flow code) που υλοποιείται μέσω της εντολής **when-else**. Ο συντρέχοντας κώδικας δηλώνεται με τη συντμημένη λέξη **DFL** μέσα στο όνομα της αρχιτεκτονικής BHV_**DFL**.

Όλα τα παραπάνω αφορούν συντρέχοντα κώδικα που δεν περιγράφεται ποιοτικά από διαγράμματα ροής (flow charts) όπως ο ακολουθιακός κώδικας. Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

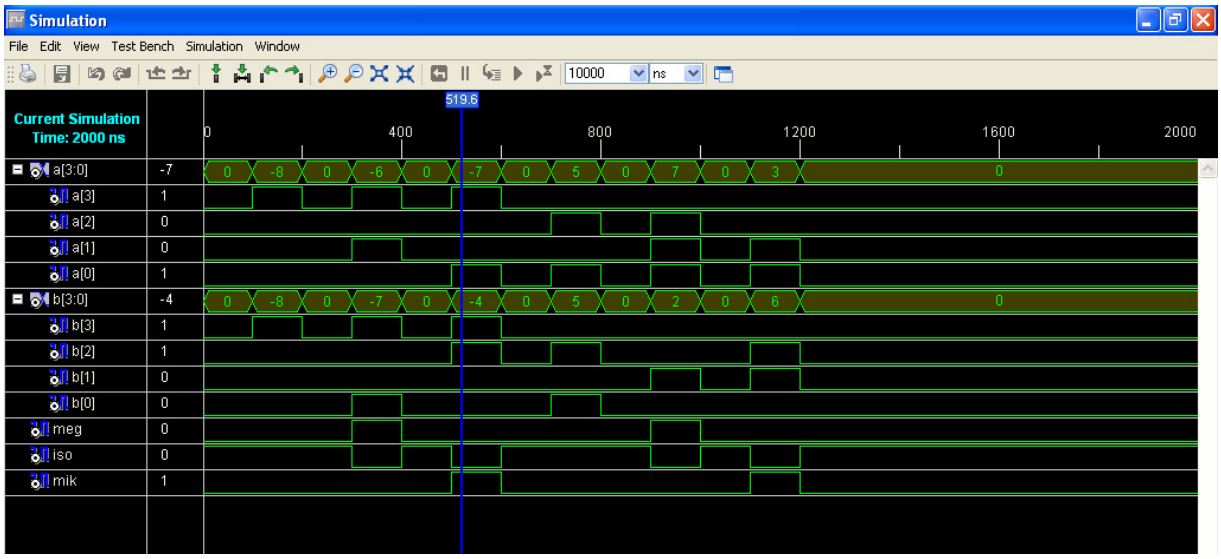
1.6.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)



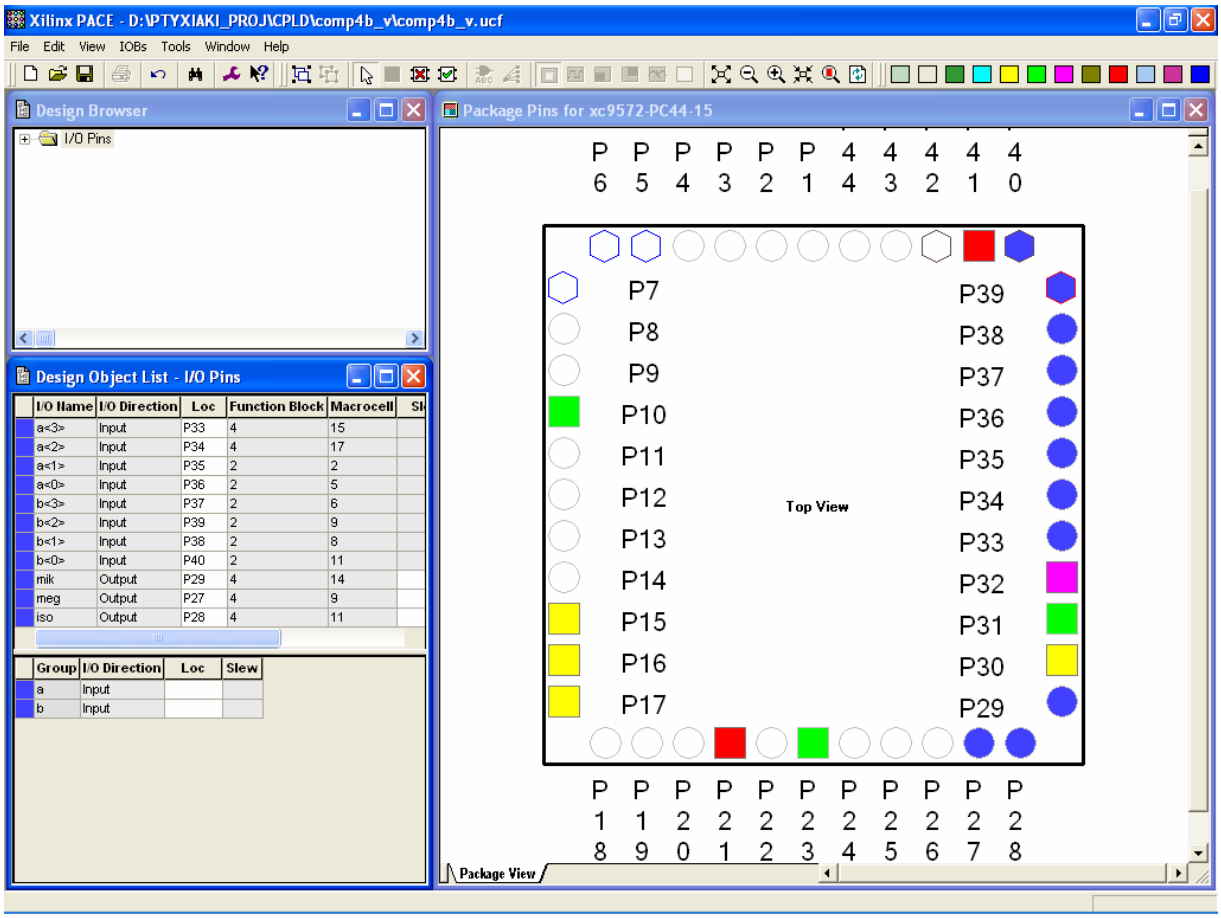
Εικόνα 1.63 VHDL αρχείο του κυκλώματος



Εικόνα 1.64 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.65 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.66 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

1.7 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ: ΠΡΟΣΘΑΦΑΙΡΕΤΗΣ ΠΡΟΣΗΜΑΣΜΕΝΩΝ ΑΡΙΘΜΩΝ 4 ΨΗΦΙΩΝ ΣΥΜΠΛΗΡΩΜΑΤΟΣ ΤΟΥ 2 (2's)

1.7.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

1.7.1.1 Σύντομη θεωρία

Οι αθροιστές n δυαδικών ψηφίων μη προσημασμένων αριθμών προκύπτουν από την διασύνδεση σε σειρά μεταξύ των n πλήρων αθροιστών του ενός δυαδικού ψηφίου. Η διασύνδεση αυτή σε σειρά υποννοεί ό,τι το κρατούμενο εξόδου του ενός είναι το κρατούμενο εισόδου για τον επόμενο. Η n-άδα των δύο αριθμών που προσθέτονται και του αποτελέσματός τους εξέρχονται παράλληλα.

Στην **εικόνα 1.67** φαίνεται η προαναφερθείσα συνδεσμολογία μεταξύ τεσσάρων πλήρων αθροιστών ενός δυαδικού ψηφίου σχηματίζοντας αθροιστή τεσσάρων δυαδικών ψηφίων μη προσημασμένων αριθμών.

Από την αφαίρεση προσημασμένων δυαδικών αριθμών με το συμπλήρωμα του 2 (2's) προκύπτει:

$A - B = A + (-B) = A + \overline{B} + 1 = A + \boxed{B \oplus 1} + 1$ (αφαίρεση προσημασμένων)

$A + B = A + \boxed{B \oplus 0} + 0$ (πρόσθεση προσημασμένων)

ή $A \pm B = A + \boxed{B \oplus E} + E$ ($E = 0$: πρόσθεση, $E = 1$: αφαίρεση)

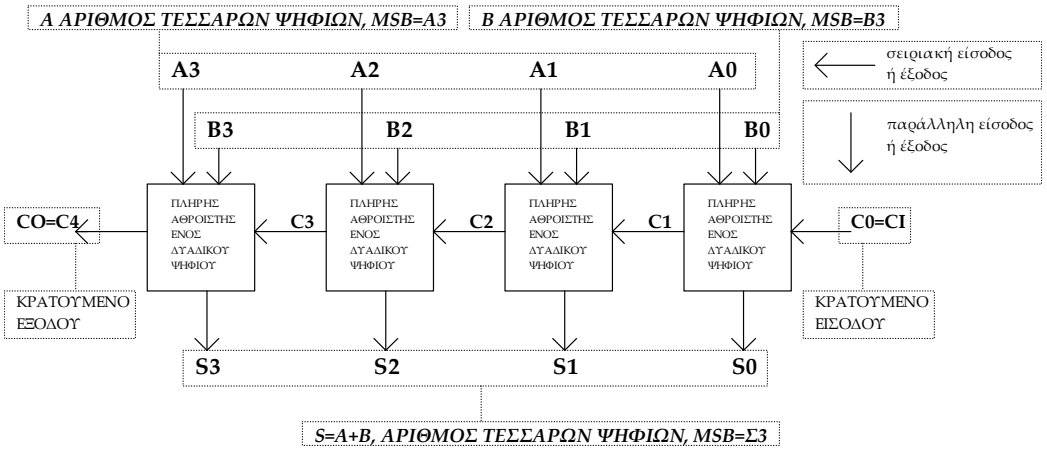
(από τον πίνακα αλήθειας της πύλη XOR: $\boxed{B \oplus 0 = B}$ και $\boxed{B \oplus 1 = \overline{B}}$

σημείωση: στο τετράγωνο πλαίσιο οι πράξεις είναι λογικές)

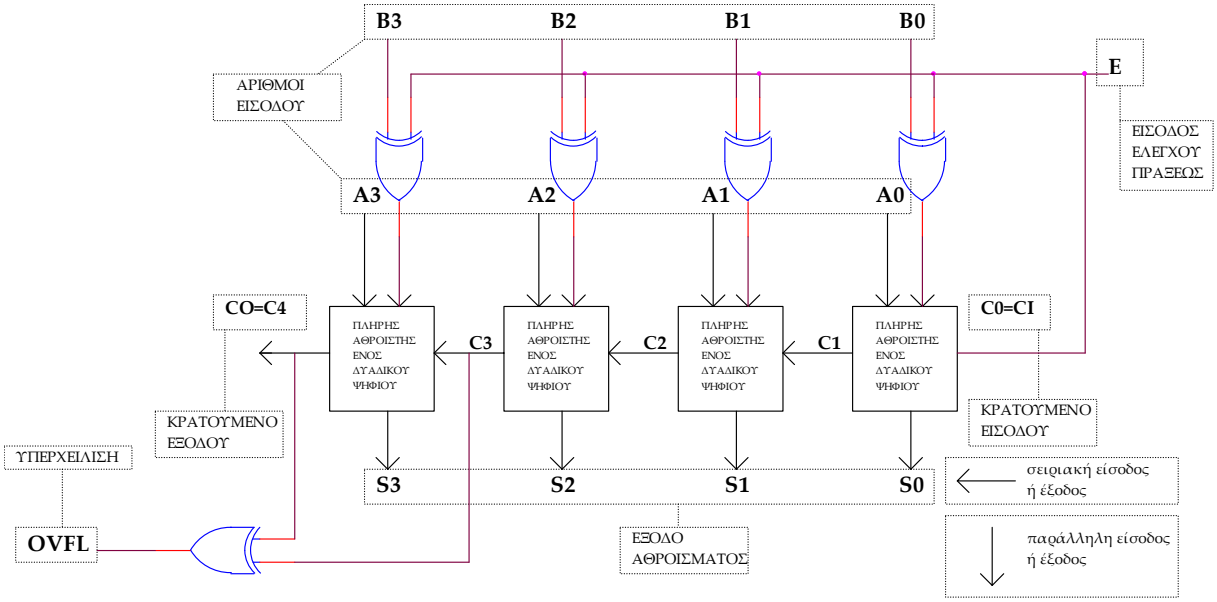
Εκμεταλλευόμαστε λοιπόν την πρόσθεση μη προσημασμένων αριθμών με κατάλληλη συνδεσμολογία πύλης XOR και πραγματοποιούμε την προσθαφαίρεση προσημασμένων αριθμών όπως στην περίπτωσή μας των τεσσάρων δυαδικών ψηφίων. Για λόγους όμως αξιοπιστίας του αποτελέσματος στην περίπτωση της υπερχείλισης επίσης εκμεταλευόμαστε την ύπαρξη της

εξόδου του κρατουμένου της τελευταίας βαθμίδας C4. Υπερχείλιση στην περίπτωση των τεσσάρων δυαδικών ψηφίων έχουμε όταν σε μία μόνο από τις δύο τελευταίες βαθμίδες το κρατούμενο εξόδου είναι σε δυναμικό λογικής κατάστασης 1. Αποτέλεσμα με υπερχείλιση δεν ανταποκρίνεται στην πραγματικότητα οπότε και επιστρατεύεται το επιπλέον δυαδικό ψηφίο του κρατουμένου της τελευταίας βαθμίδας για να δείξει το πραγματικό αποτέλεσμα. Ονομάζουμε λοιπόν OVFL το δυαδικό ψηφίο ένδειξης υπερχείλισης προσημασμένων αριθμών για το οποίο ισχύει $OVFL = C3 \oplus C4$, (όπου C3 και C4 είναι τα κρατούμενα των δύο τελευταίων βαθμίδων).

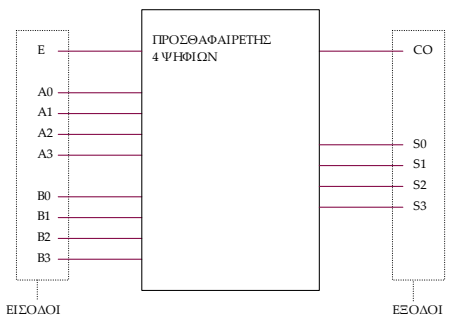
Ο προσθαφαιρέτης λοιπόν τεσσάρων ψηφίων προσημασμένων αριθμών έχει σαν εισόδους του τον αριθμό ***A*** με δυαδικά ψηφία A3, A2, A1, A0 (MSB=A3 και ψηφίο προσήμου), τον αριθμό ***B*** με δυαδικά ψηφία B3, B2, B1, B0 (MSB=B3 και ψηφίο προσήμου) και το δυαδικό ψηφίο E για την επιλογή των πράξεων της πρόσθεσης ή της αφαίρεσης των προσημασμένων αριθμών. Η έξοδός του είναι ο αριθμός ***S*** με τα δυαδικά ψηφία S0, S1,…, S3 (MSB=S3 και ψηφίο προσήμου) σαν το άθροισμα των αριθμών ***A*** και ***B***, το δυαδικό ψηφίο κρατουμένου της εξόδου CO=C4 και OVFL δυαδικό ψηφίο ένδειξης υπερχείλισης. Το δυαδικό ψηφίο κρατουμένου της εξόδου CO=C4 δεν λαμβάνεται υπόψιν παρά μονάχα στην περίπτωση ένδειξης υπερχείλισης. Στην περίπτωση λοιπόν αυτή το αποτέλεσμα είναι ένας προσημασμένος αριθμός πέντε δυαδικών ψηφίων με MSB και ψηφίο προσήμου το δυαδικό ψηφίο κρατουμένου της εξόδου CO=C4. Στην **εικόνα 1.68** δείχνεται η συνδεσμολογία βαθμίδων για τον προσθαφαιρέτη των τεσσάρων δυαδικών ψηφίων προσημασμένων αριθμών**.** Στην **εικόνα 1.69** απεικονίζεται το γραφικό σύμβολο του προσθαφαιρέτη αυτού.



Εικόνα 1.67 Αθροιστής 4 ψηφίων μη προσημασμένων αριθμών



Εικόνα 1.68 Προσθαφαιρέτης 4 ψηφίων προσημασμένων αριθμών



Εικόνα 1.69 Γραφικό σύμβολο προσθαφαιρέτη προσημασμένων αριθμών 4 ψηφίων

Βαθμίδα του πλήρη αθροιστή ενός δυαδικού ψηφίου (αναφέρεται λόγω χρήσης του).

Πίνακας 1.28 Πίνακας αλήθειας πλήρους αθροιστή ενός ψηφίου

ΕΙΣΟΔΟΙ			**ΕΞΟΔΟΙ**	
x	**y**	**ci**	**s**	**co**
0	0	0	0	0
0	0	1	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Η βαθμίδα του πλήρους αθροιστή ενός δυαδικού ψηφίου (προηγούμενος πίνακας αλήθειας) προσθέτει τα δύο δυαδικά ψηφία εισόδου x και y με το κρατούμενο εισόδου ci οπότε η έξοδος είναι το άθροισμα s και το κρατούμενο εξόδου co.

Δηλαδή $x + y + ci = s + co\left(\text{ψηφίο ανώτερης δυαδικής τάξης}\right)$

Σχεδίαση εξαιρετικά βολική για συνδεσμολογία τους σε σειρά σαν δομικά στοιχεία αθροιστών περισσοτέρων δυαδικών ψηφίων (έγινε αναφορά αρχικά).

Πίνακας 1.29 Πίνακας Karnaugh πλήρους αθροιστή ενός ψηφίου

Έξοδος: s

ci \ **xy**	**00**	**01**	**11**	**10**
0	0	1	0	1
1	1	0	1	0

Έξοδος: co

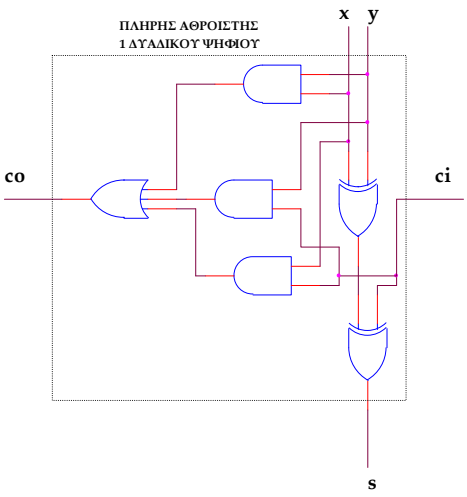
ci \ **xy**	**00**	**01**	**11**	**10**
0	0	0	1	0
1	0	1	1	1

Λογικές εξισώσεις:

$$s = \bar{x}\cdot\bar{y}\cdot ci + \bar{x}\cdot y\cdot\bar{ci} + x\cdot\bar{y}\cdot\bar{ci} + x\cdot y\cdot ci =$$
$$= ci\cdot(\bar{x}\cdot\bar{y} + x\cdot y) + \bar{ci}\cdot(\bar{x}\cdot y + x\cdot\bar{y}) =$$
$$= ci\cdot\overline{(x\oplus y)} + \bar{ci}\cdot(x\oplus y) = ci\oplus(x\oplus y)\Rightarrow$$
$$\Rightarrow \boxed{s = ci\oplus(x\oplus y)}$$
$$\boxed{co = x\cdot y + x\cdot ci + y\cdot ci}$$

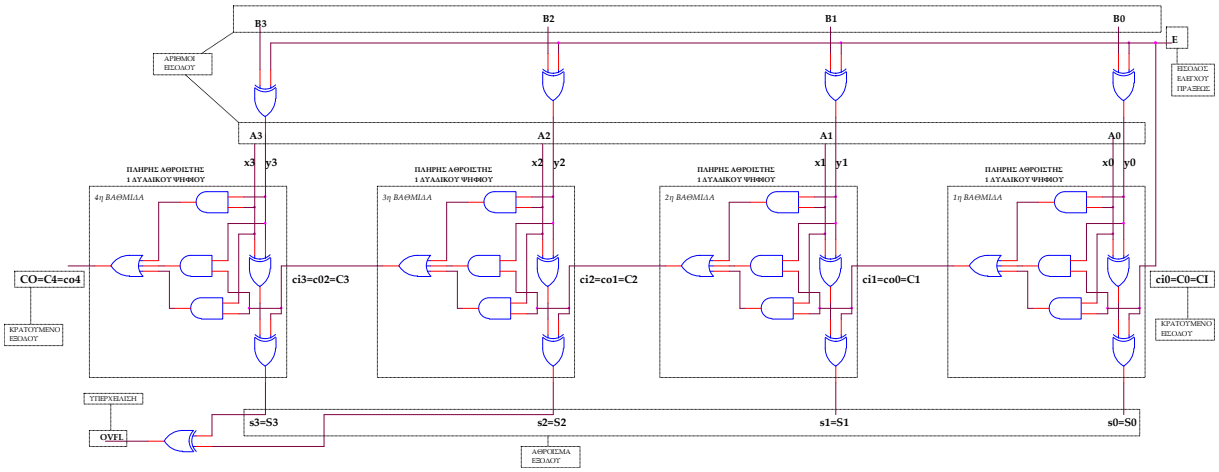
1.7.1.2 Συμβατική ψηφιακή υλοποίηση

Οι παραπάνω λογικές εξισώσεις για τον πλήρη αθροιστή ενός δυαδικού ψηφίου παραπέμπουν στην παρακάτω συμβατική υλοποίηση όπως της **εικόνας 1.70**.



Εικόνα 1.70 Λογικό κύκλωμα πλήρους αθροιστή ενός δυαδικού ψηφίου

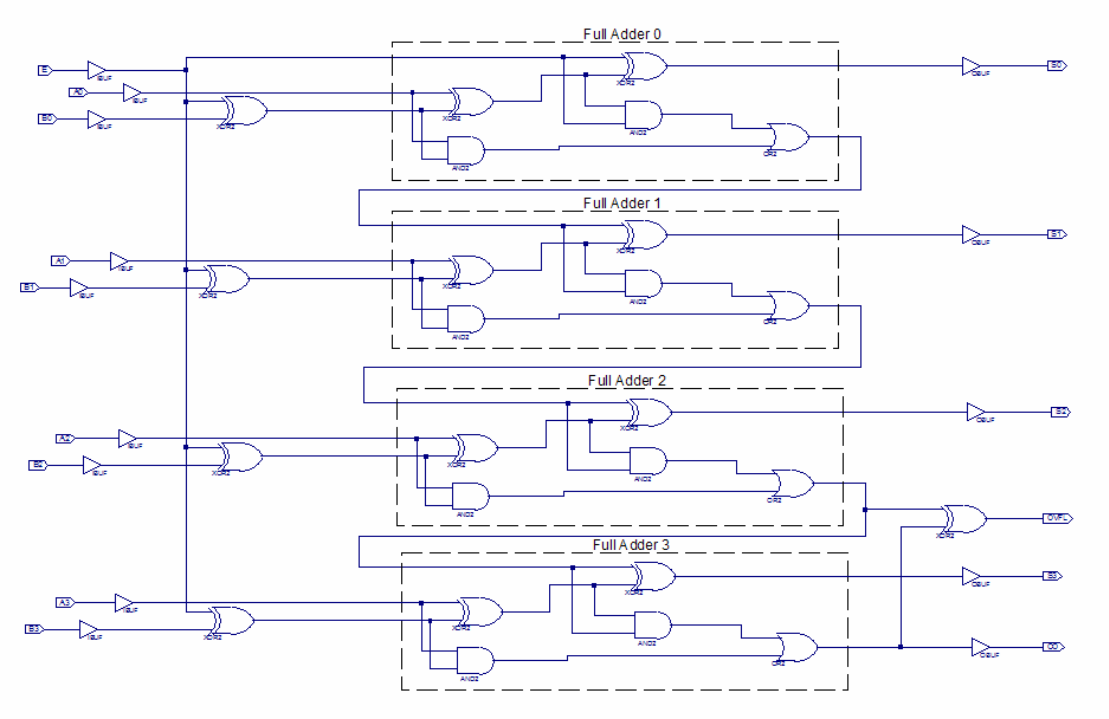
Αντικαθιστώντας την βαθμίδα του πλήρη αθροιστή της **εικόνας 1.70** με το ισοδύναμο σχηματικό της **εικόνας 1.68** προκύπτει η υλοποίηση της **εικόνας 1.71**.



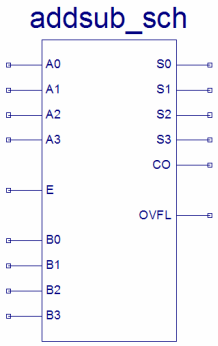
Εικόνα 1.71 Λογικό κύκλωμα υλοποίησης προσθαφαιρέτη 4 ψηφίων προσημασμένων αριθμών

1.7.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

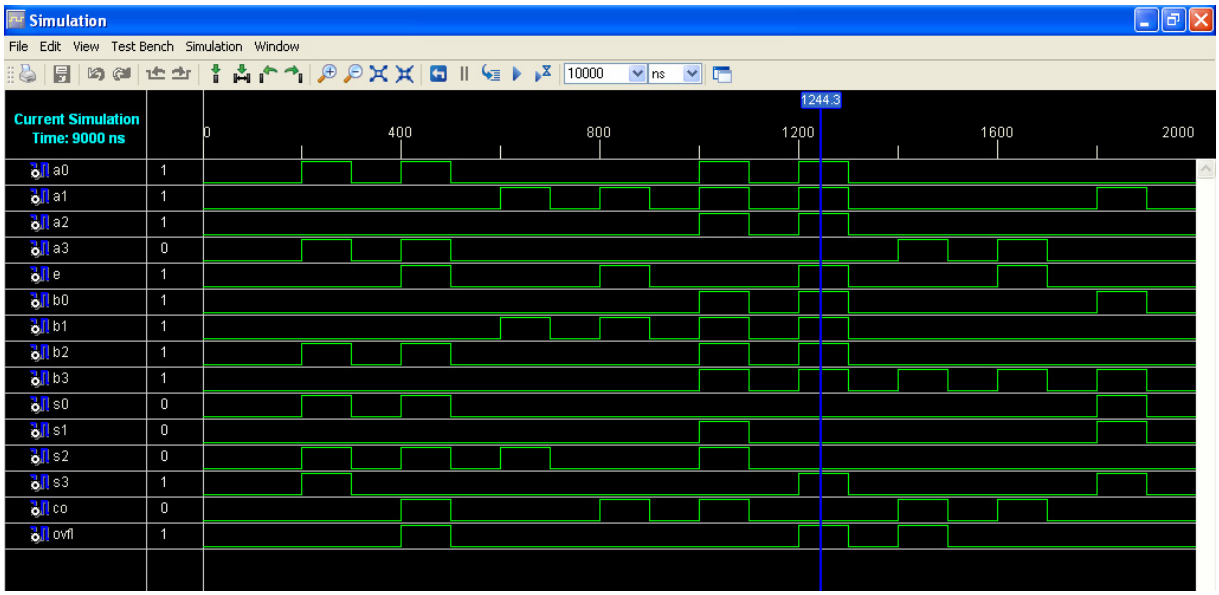
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα. Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



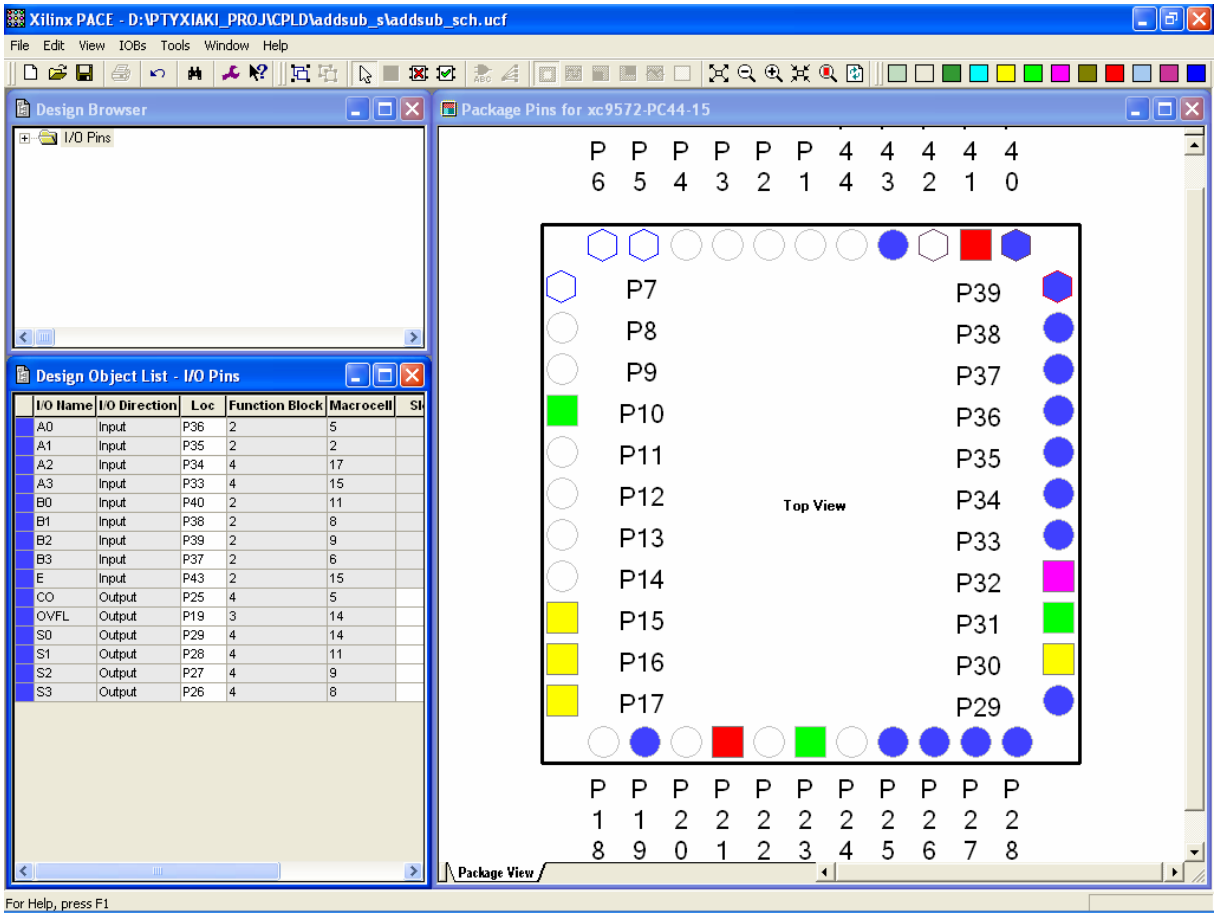
Εικόνα 1.72 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 1.73 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.74 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.75 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

1.7.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 όπως αναφέρεται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 1.30 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

12 architecture BHV of addsub_vhd is	Αρχή δήλωσης αρχιτεκτονικής.
13 begin	Αρχή κώδικα αρχιτεκτονικής.
14 process (a, b, e)	Αρχή της διεργασίας με λίστα ευαισθησίας τις εισόδους **a**, **b**, **e**.
15 variable temp : SIGNED (4 downto 0) ;	Δήλωση της μεταβλητής **temp** πέντε δυαδικών ψηφίων τύπου **signed**.
16 begin	Αρχή ενεργοποίησης της διεργασίας.
17 if e = '0' then	Αρχή εκτέλεσης **if-then-else**.
18 temp :=(CONV_SIGNED(a,5)+ +CONV_SIGNED(b,5));	Μετατροπή δεδομένων τύπου **signed** από τέσσερα δυαδικά ψηφία σε πέντε, άθροιση και μεταφορά τους στην μεταβλητή **temp**.
19 else	Διακλάδωση εντολής **if-then-else**.
20 temp := (CONV_SIGNED(a,5)- -CONV_SIGNED(b,5));	Μετατροπή δεδομένων τύπου **signed** από τέσσερα δυαδικά ψηφία σε πέντε, διαφορά και μεταφορά τους στην μεταβλητή **temp**.
21 end if;	Τέλος εκτέλεσης **if-then-else**.
22 s<=CONV_STD_LOGIC_VECTOR(temp,5);	Μετατροπή τύπου **signed** σε **std_logic_vector** πέντε δυαδικών ψηφίων και μεταφορά τους στην έξοδο **s**.
23 end process;	Τέλος εκτέλεσης process.
24 end BHV;	Τέλος κώδικα αρχιτεκτονικής.

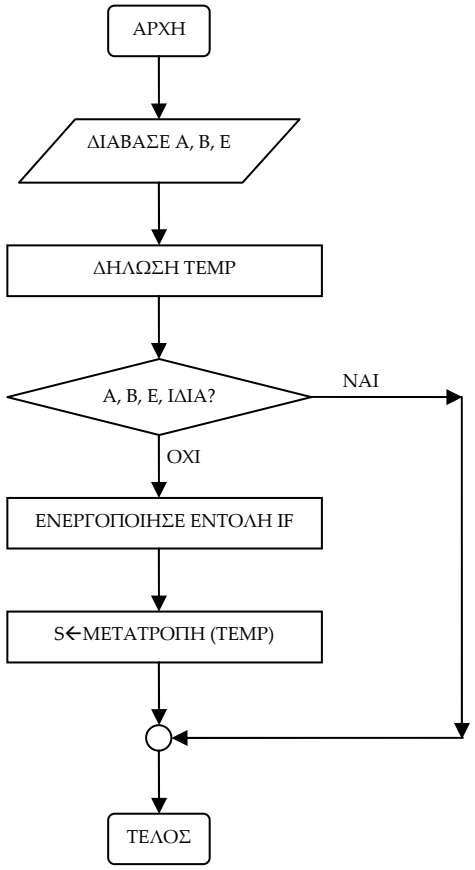
Σημειώσεις.

Γραμμή 16. Η ενεργοποίηση της διεργασίας δηλώνεται με τη λέξη begin. Η διεργασία ενεργοποιείται με την αλλαγή της λογικής κατάστασης μιας από τις εισόδους **a**, **b**, **e** της λίστας ευαισθησίας.

Γραμμή 17. Κατά την εκτέλεση της εντολής **if-then-else** έχουμε το λογικό έλεγχο για την κατάσταση της εισόδου **e** σε λογικό 0. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 18. Σε διαφορετική περίπτωση (γραμμή 19) εκτελείται η γραμμή 20.

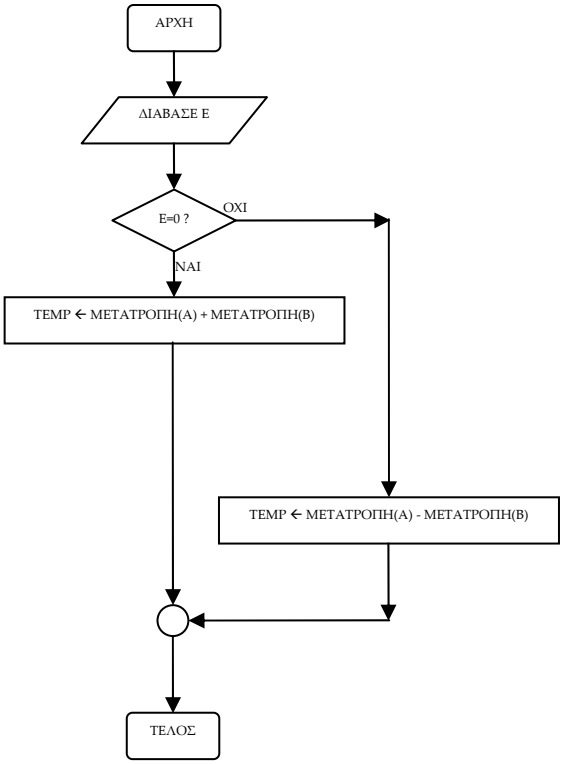
Πίνακας 1.31 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (a, b, e)

  variable temp …;

begin

  if …else …end if;

  s<=CONV_STD_LOGIC_VECTOR(temp,5);

end process;
```



Πίνακας 1.32 Η εντολή **if-then-else** και το **διάγραμμα ροής**

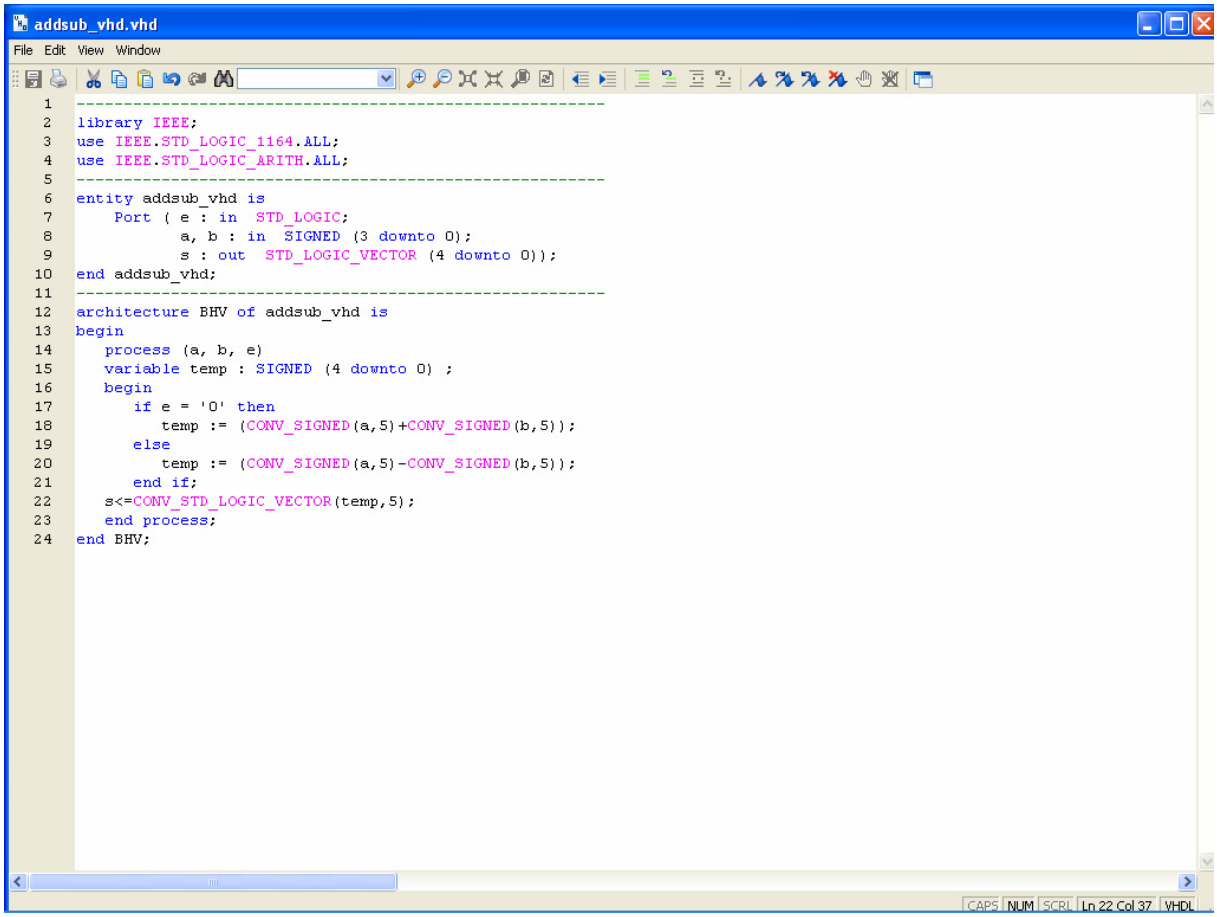
```
if e='0' then
temp :=(conv(…)+conv(…));
   else … temp :=(conv(…)-conv(…));
end if;
```



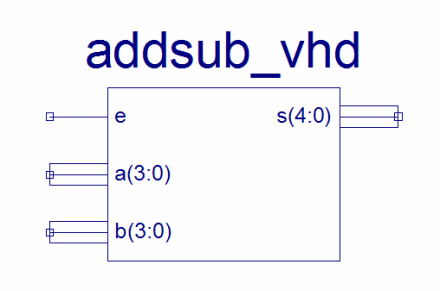
Σημείωση. Στα προηγούμενα διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

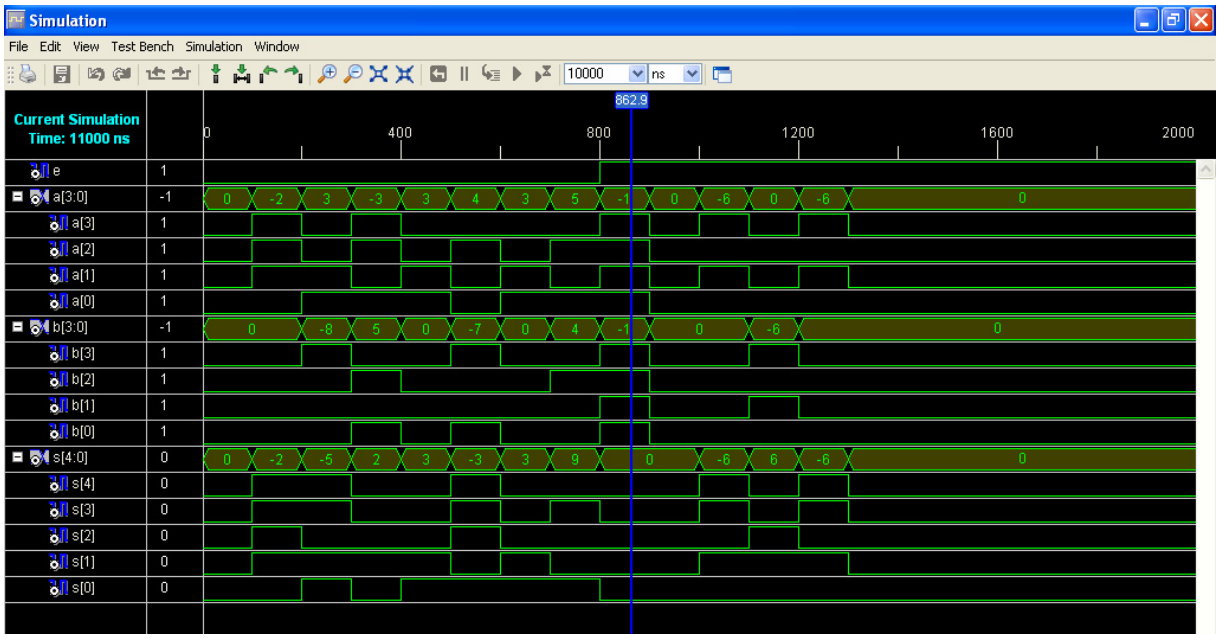
1.7.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)



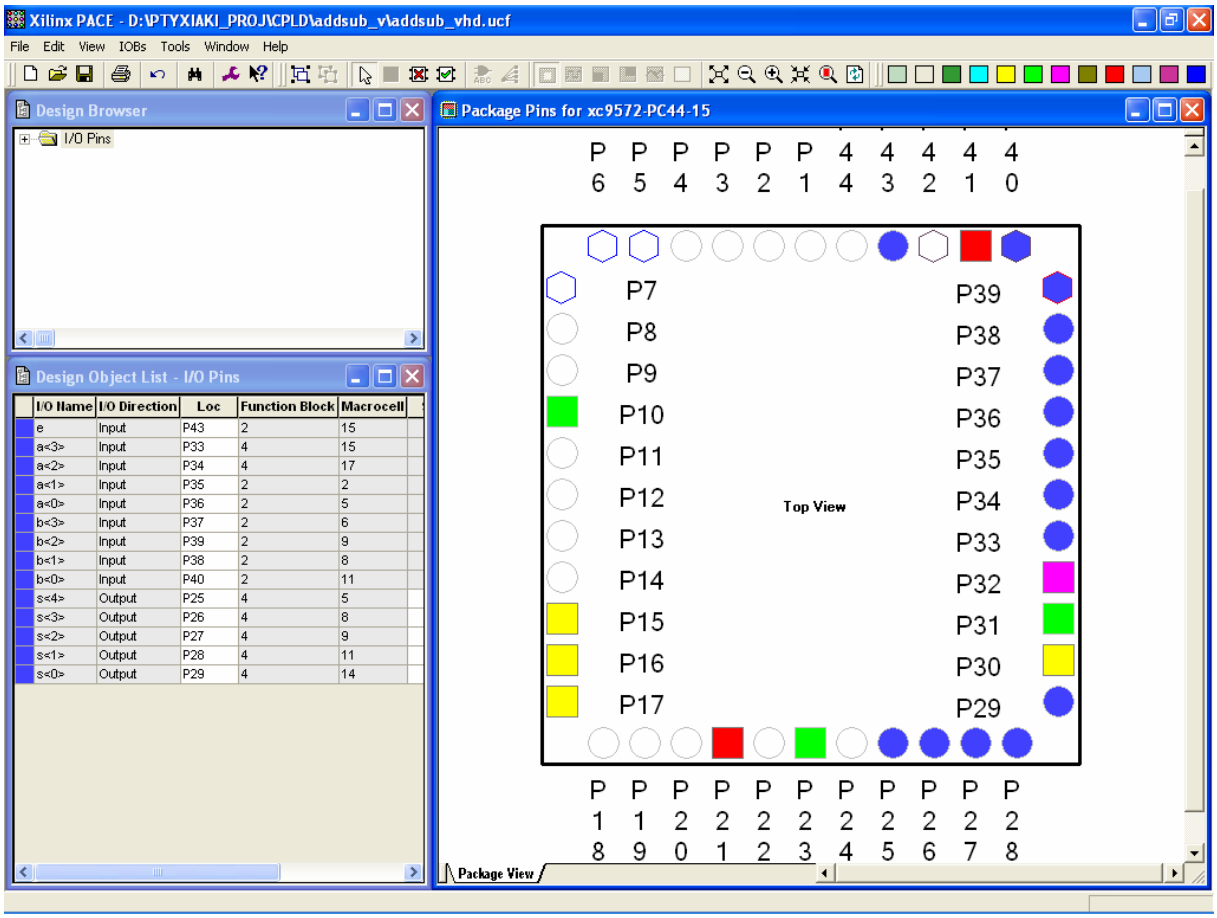
Εικόνα 1.76 VHDL αρχείο του κυκλώματος



Εικόνα 1.77 Σχηματικό σύμβολο κυκλώματος



Εικόνα 1.78 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 1.79 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

2. ΑΚΟΛΟΥΘΙΑΚΑ ΚΥΚΛΩΜΑΤΑ

Ακολουθεί η ανάπτυξη σε σχηματική και περιγραφική γλώσσα (VHDL) των παρακάτω συνολικών εργασιών (project):

- μετρητής_1: δυαδικός σύγχρονος αύξοντας (χωρητικότητας 16)
- μετρητής_2: δυαδικός σύγχρονος φθίνοντας (χωρητικότητας 16)
- μετρητής_3: σύγχρονος αύξοντας και φθίνοντας μετρητής (χωρητικότητας 16)
- καταχωρητής_1: δεξιάς ολίσθησης σειριακής εισόδου – σειριακής εξόδου (οχτώ ψηφίων)
- καταχωρητής_2: δεξιάς ολίσθησης σειριακής εισόδου – παράλληλης εξόδου (οχτώ ψηφίων)
- καταχωρητής_3: δεξιάς ολίσθησης παράλληλης εισόδου – σειριακής εξόδου (οχτώ ψηφίων)
- καταχωρητής_4: δεξιάς ολίσθησης παράλληλης εισόδου – παράλληλης εξόδου (οχτώ ψηφίων)
- κυκλικός μετρητής (οχτώ ψηφίων)
- μετρητής Johnson (οχτώ ψηφίων).

2.1 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ ΜΕΤΡΗΤΗ_1: ΔΥΑΔΙΚΟΣ ΣΥΓΧΡΟΝΟΣ ΑΥΞΟΝΤΑΣ ΧΩΡΗΤΙΚΟΤΗΤΑΣ 16

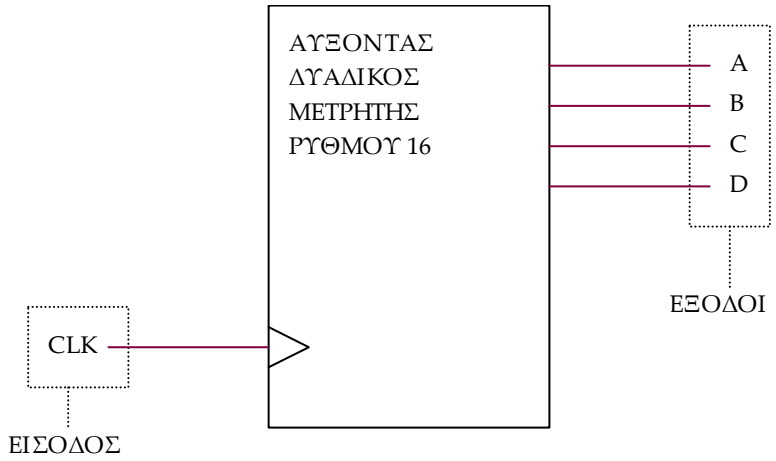
2.1.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

2.1.1.1 Σύντομη θεωρία

Οι δυαδικοί μετρητές (*binary counters*) είναι ακολουθιακά κυκλώματα μέτρησης του αριθμού παλμών του ρολογιού (*clock*) της εισόδου τους. Η ένδειξη της μέτρησης αυτής στην έξοδο γίνεται στο δυαδικό σύστημα αρίθμησης. Ονομάζονται αύξοντες μετρητές στην περίπτωση που η ακολουθία των δυαδικών αριθμών ένδειξης της μέτρησης είναι αύξουσα. Δομικό στοιχείο της κατασκευής τους αποτελούν οι δισταθείς πολυδονητές (*flip-flop*). Αποκαλούνται λοιπόν σύγχρονοι στην περίπτωση που τα ρολόγια όλων των δισταθών πολυδονητών που τους αποτελούν δέχονται παλμούς από το ρολόγι της εισόδου τους. Ο αριθμός δισταθών πολυδονητών που τους αποτελούν επίσης καθορίζει την χωρητικότητά τους (*module* ή *mod*). Συγκεκριμένα ένας μετρητής αποτελούμενος από n δισταθείς πολυδονητές ή εξόδους μετράει (στο δυαδικό σύστημα) μέχρι και την τιμή 2^n (mod-2^n) παλμών ρολογιού.

Ο αύξοντας δυαδικός μετρητής χωρητικότητας 16 (mod-16) έχει σαν εισοδό του το ρολόγι CLK και τις τέσσερις εξόδους A, B, C, D (D=MSB). Είναι κατασκευασμένος από τέσσερις θετικής διέγερσης δισταθείς πολυδονητές (*flip-flop*) συνδεσμολογίας J-K για να μπορούν να μετρούν ως την χωρητικότητα ή το ρυθμό του (16 παλμούς ή *mod-16*). Στην **εικόνα 2.1** απεικονίζεται το γραφικό σύμβολο του παραπάνω μετρητή, στον **πίνακα 2.1** υπάρχει ο πίνακας διέγερσης του J-K δισταθή πολυδονητή. Στον **πίνακα 2.2** υπάρχει ο πίνακας διέγερσης του

μετρητή και στους κατοπινούς **πίνακες 2.3** ως και **2.6** υπάρχουν ενημερωμένοι οι αντίστοιχοι Karnaugh για τον κάθε πολυδονητή.



Εικόνα 2.1 Γραφικό σύμβολο αύξοντα δυαδικού μετρητή mod-16

Πίνακας 2.1 Πίνακας διέγερσης δισταθή πολυδονητή J-K (flip-flop)

Q	**Q_+**	**J**	**K**
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Πίνακας 2.2 Πίνακας διέγερσης σύγχρονου αύξοντα δυαδικού μετρητή χωρητικότητας 16

Παρούσα κατάσταση Q_n				Επόμενη κατάσταση Q_{n+1}											
MSB			**LSB**	**MSB**			**LSB**								
D	**C**	**B**	**A**	D_+	C_+	B_+	A_+	J_D	K_D	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
1	1	1	0	1	1	1	1	X	0	X	0	X	0	1	X
1	1	1	1	0	0	0	0	X	1	X	1	X	1	X	1

Έξοδος είναι η παρούσα κατάσταση Q_n.

X: αδιάφορη κατάσταση.

Η κωδικοποίηση των καταστάσεων έγινε με το δυαδικό κωδικό 8421.

Πίνακας 2.3 Πίνακας Karnaugh για τον πολυδονητή Α

Έξοδος: K_A

BA \ **DC**	**00**	**01**	**11**	**10**
00	X	X	X	X
01	1	1	1	1
11	1	1	1	1
10	X	X	X	X

Έξοδος: J_A

BA \ **DC**	**00**	**01**	**11**	**10**
00	1	1	1	1
01	X	X	X	X
11	X	X	X	X
10	1	1	1	1

Πίνακας 2.4 Πίνακας Karnaugh για τον πολυδονητή B

Έξοδος: J_B

BA \ DC	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	X	X	X	X
10	X	X	X	X

Έξοδος: K_B

BA \ DC	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	1	1	1	1
10	0	0	0	0

Πίνακας 2.5 Πίνακας Karnaugh για τον πολυδονητή C

Έξοδος: K_C

BA \ DC	00	01	11	10
00	X	0	0	X
01	X	0	0	X
11	X	1	1	X
10	X	0	0	X

Έξοδος: J_C

BA \ DC	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	1	X	X	1
10	0	X	X	X

Πίνακας 2.6 Πίνακας Karnaugh για τον πολυδονητή D

Έξοδος: K_D

BA \ DC	00	01	11	10
00	X	X	0	0
01	X	X	0	0
11	X	X	1	0
10	X	X	0	0

Έξοδος: J_D

BA \ DC	00	01	11	10
00	0	0	X	X
01	0	0	X	X
11	0	1	X	X
10	0	0	X	X

Συνολικό συμπέρασμα μετά από την απλοποίηση των πινάκων Karnaugh:

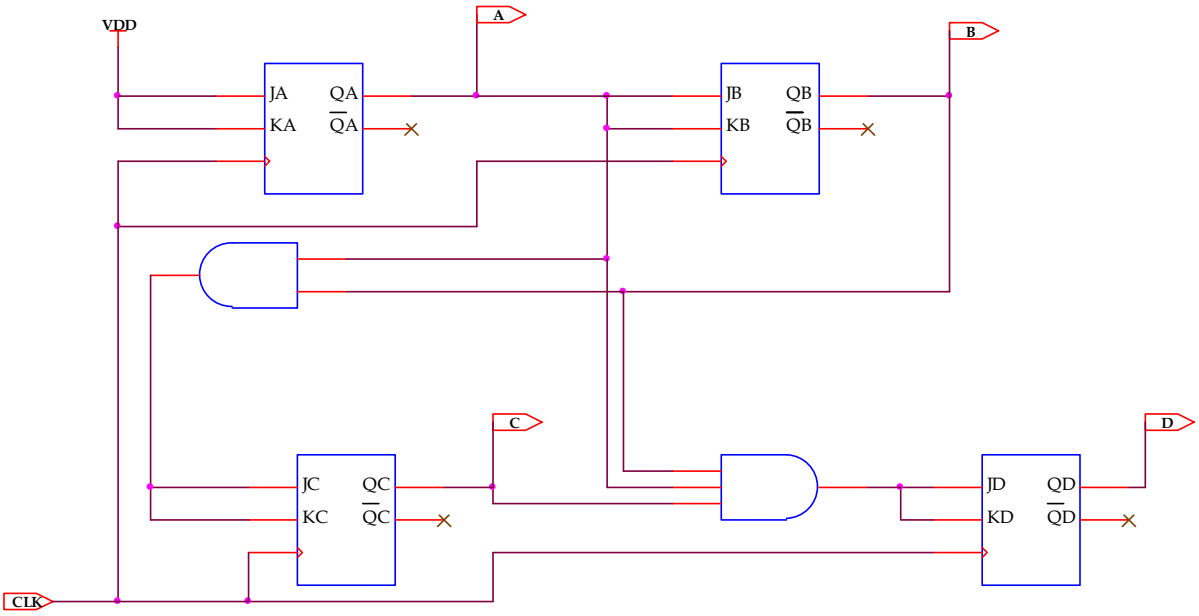
$$\boxed{K_A = 1} \quad \boxed{J_A = 1} \qquad \boxed{K_B = A} \quad \boxed{J_B = A}$$

$$\boxed{K_C = A \cdot B} \quad \boxed{J_C = A \cdot B} \qquad \boxed{K_D = A \cdot B \cdot C} \quad \boxed{J_D = A \cdot B \cdot C}$$

Εικόνα 2.2 Συνολική απλοποίηση πινάκων Karnaugh αύξοντα δυαδικού μετρητή mod-16

2.1.1.2 Συμβατική ψηφιακή υλοποίηση

Η λογική εξίσωση της **εικόνας 2.2** παραπέμπει στο λογικό κύκλωμα υλοποίησης με συμβατικές λογικές πύλες και δισταθείς πολυδονητές J-K της παρακάτω **εικόνας 2.3**.

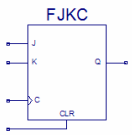


Εικόνα 2.3 Λογικό κύκλωμα υλοποίησης αύξοντα δυαδικού μετρητή mod-16

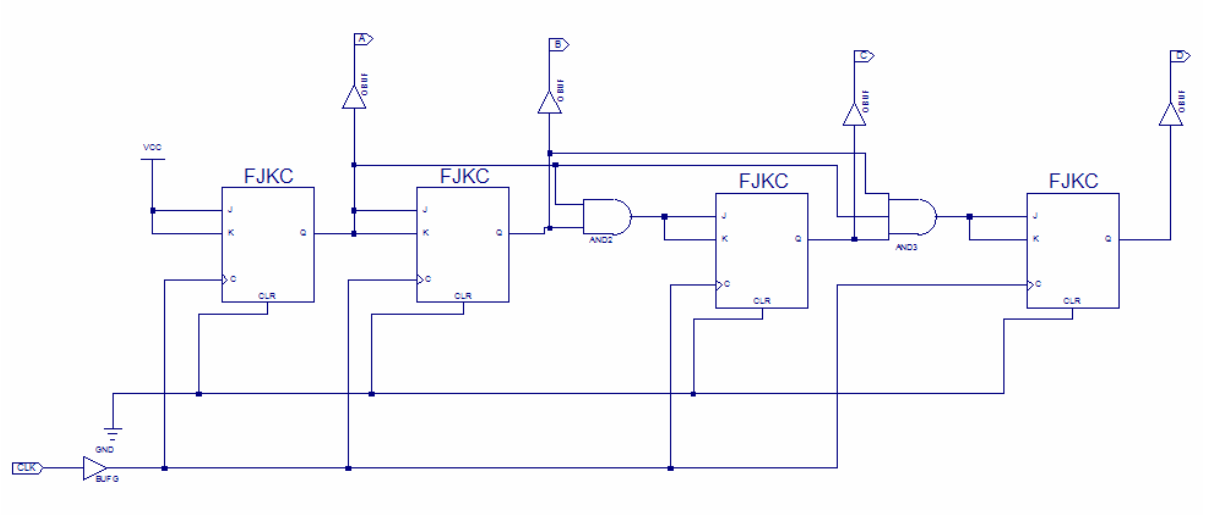
2.1.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα. Στον **πίνακα 2.7** φαίνεται το καινούργιο υλικό. Επιλέχτηκε για την ασύγχρονη λειτουργία μηδενισμού που προσφέρει μέσω της εισόδου του CLR όταν ενεργοποιείται με λογικό 1. Η λειτουργία αυτή είναι απενεργοποιημένη γιατί η είσοδος CLR είναι γειωμένη, είναι διαθέσιμη όμως στην γενικότερη περίπτωση ασύγχρονου μηδενισμού του καταχωρητή μέσω της εισόδου του CLR.

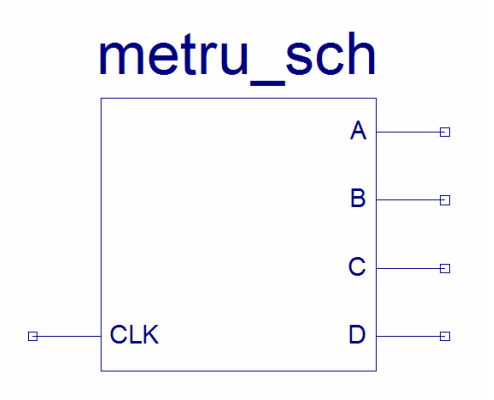
Πίνακας 2.7 Καινούργια υλικά σχηματικού

ΚΑΙΝΟΥΡΓΙΑ ΥΛΙΚΑ ΣΧΕΔΙΑΣΗΣ PROJECT NAVIGATOR			
ΣΧΗΜΑ	ΟΝΟΜΑΣΙΑ	ΙΔΙΟΤΗΤΑ	ΚΑΤΑΛΟΓΟΣ ΒΙΒΛΙΟΘΗΚΗΣ SYMBOL
	FJKC	JK FLIP-FLOP ΜΕ ΑΣΥΓΧΡΟΝΟ CLEAR (CLR ΕΙΣΟΔΟΣ)	FLIP_FLOP

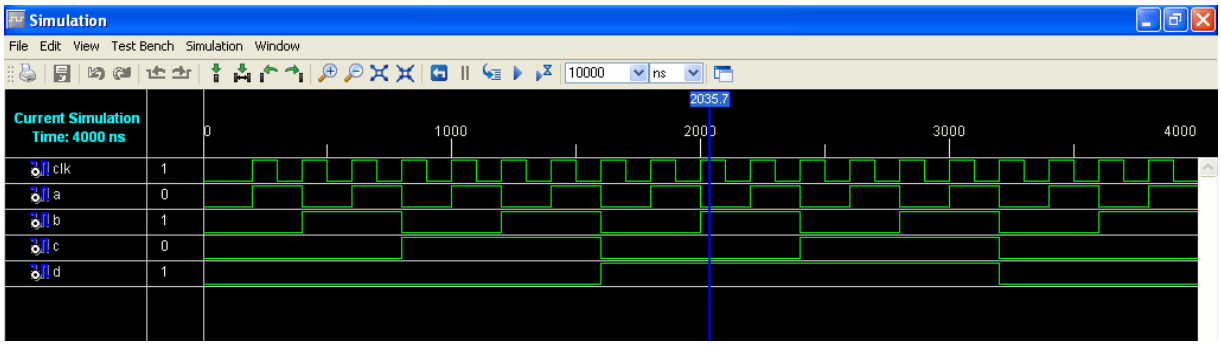
Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



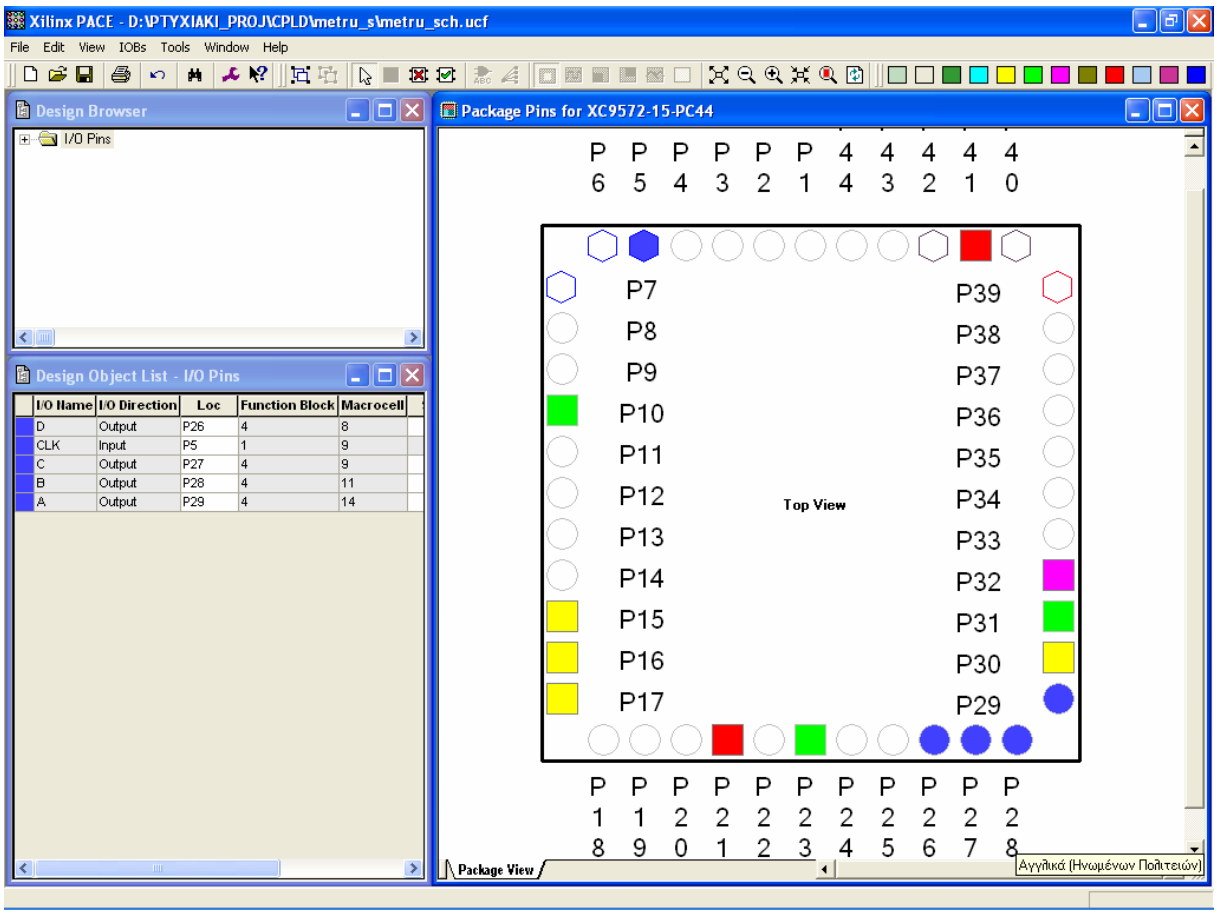
Εικόνα 2.4 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 2.5 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.6 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.7 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.1.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 και τον συγκριτή προσημασμένων αριθμών όπως αναφέρονται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 2.8 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

12	architecture BHV of metru is	Αρχή δήλωσης αρχιτεκτονικής.
13	begin	Αρχή κώδικα αρχιτεκτονικής.
14	process (clk)	Αρχή της διεργασίας με λίστα ευαισθησίας την είσοδο **clk**.
15	variable temp: integer range 0 to 15;	Δήλωση της ακέραιας μεταβλητής **temp** με τιμές από το 0 ως το 15 του δεκαδικού.
16	begin	Αρχή ενεργοποίησης της διεργασίας.
17	if clk='1' and clk'event then	Αρχή εκτέλεσης **if-then.**
18	temp := temp + 1;	Αύξηση της προηγούμενης τιμής του **temp** κατά 1.
19	end if;	Τέλος εκτέλεσης **if-then.**
20	cnt<=CONV_STD_LOGIC_VECTOR(temp, 4);	Μετατροπή δεδομένων μεταβλητής **temp** και μεταφορά τους στην έξοδο **cnt**.
21	end process;	Τέλος εκτέλεσης process.
22	end BHV;	Τέλος κώδικα αρχιτεκτονικής.

Σημειώσεις.

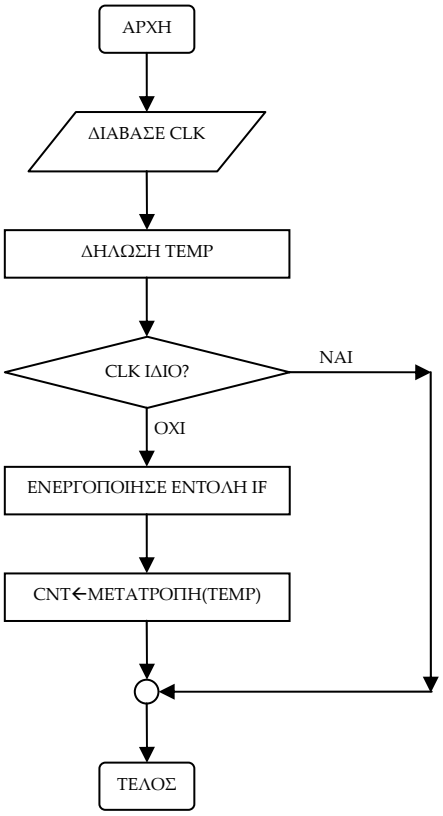
Γραμμή 16. Η ενεργοποίηση της διεργασίας δηλώνεται με τη λέξη begin. Η διεργασία ενεργοποιείται με την αλλαγή της λογικής κατάστασης της εισόδου **clk** της λίστας ευαισθησίας.

Γραμμή 17. Κατά την εκτέλεση της εντολής **if-then** έχουμε το λογικό έλεγχο για αλλαγή της εισόδου **clk** από λογικό 0 σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 18. Σε διαφορετική περίπτωση εκτελείται η γραμμή 19.

Γραμμή 20. Η συνάρτηση **conv_std_logic_vector** βρίσκεται στο πακέτο **std_logic_arith**. Η συνάρτηση αυτή μετατρέπει την ακέραια μεταβλητή σε μια τιμή τεσσάρων δυαδικών ψηφίων τύπου std_logic και μεταφέρει το αποτέλεσμα της μετατροπής στην έξοδο.

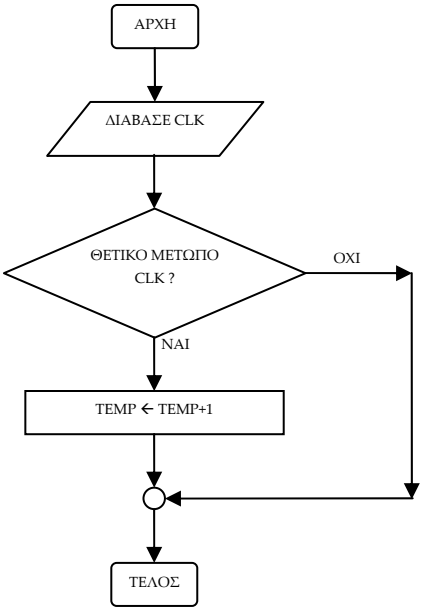
Πίνακας 2.9 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (clk)

variable temp: integer range 0 to 15;

begin

        if …end if;

cnt<=CONV_STD_LOGIC_VECTOR(temp, 4);

end process;
```



Πίνακας 2.10 Η εντολή **if-then** και το **διάγραμμα ροής**

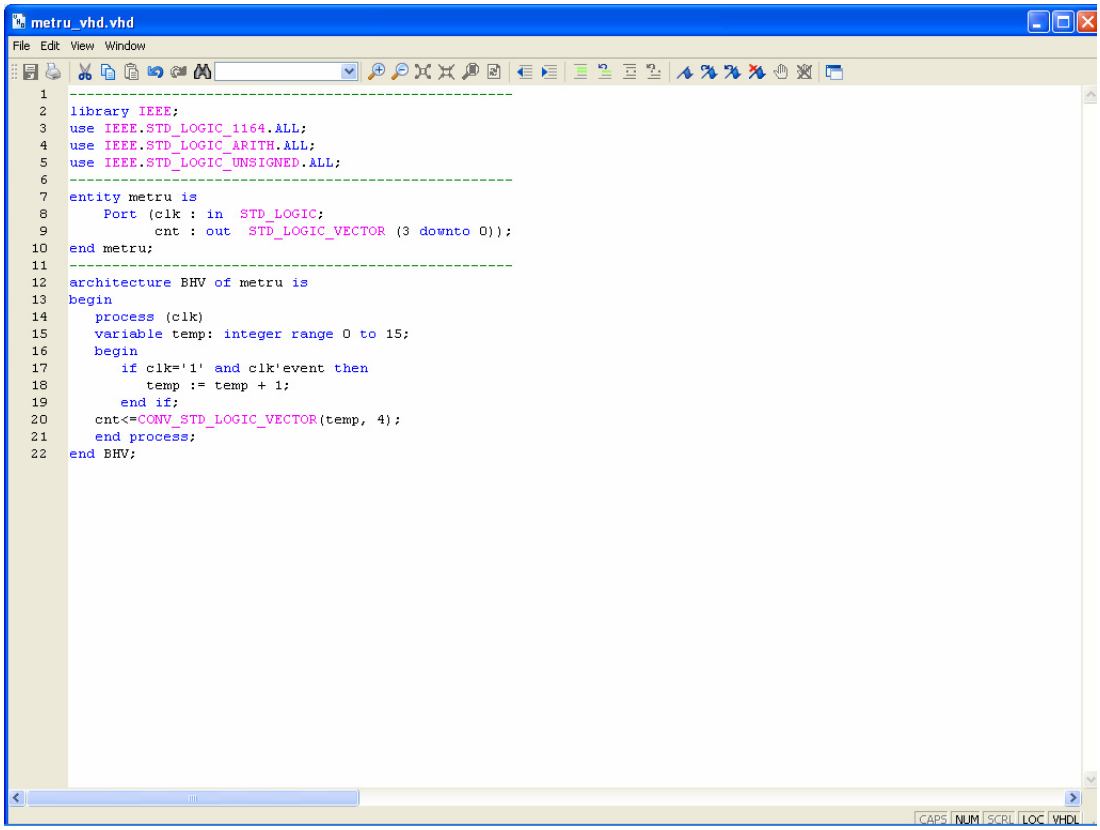
```
if clk='1' and clk'event then

        temp := temp + 1;

end if;
```



Σημείωση. Στα παραπάνω διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

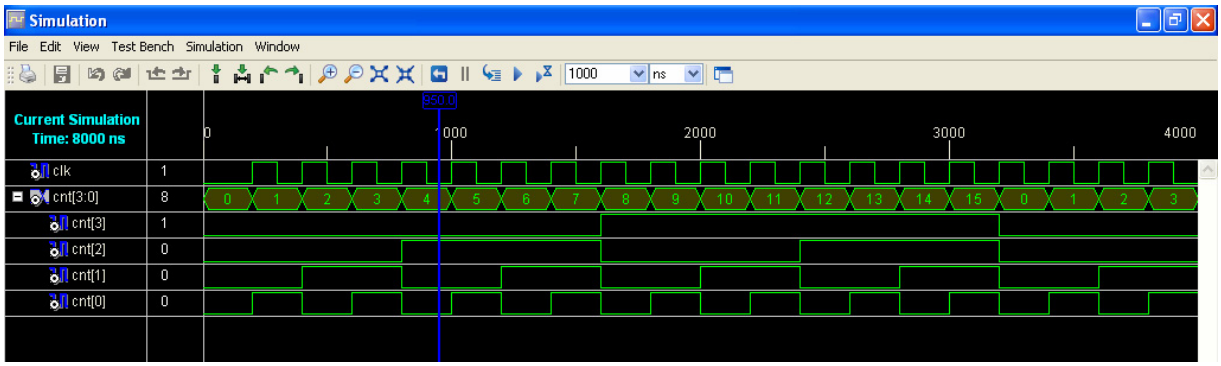
Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

2.1.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

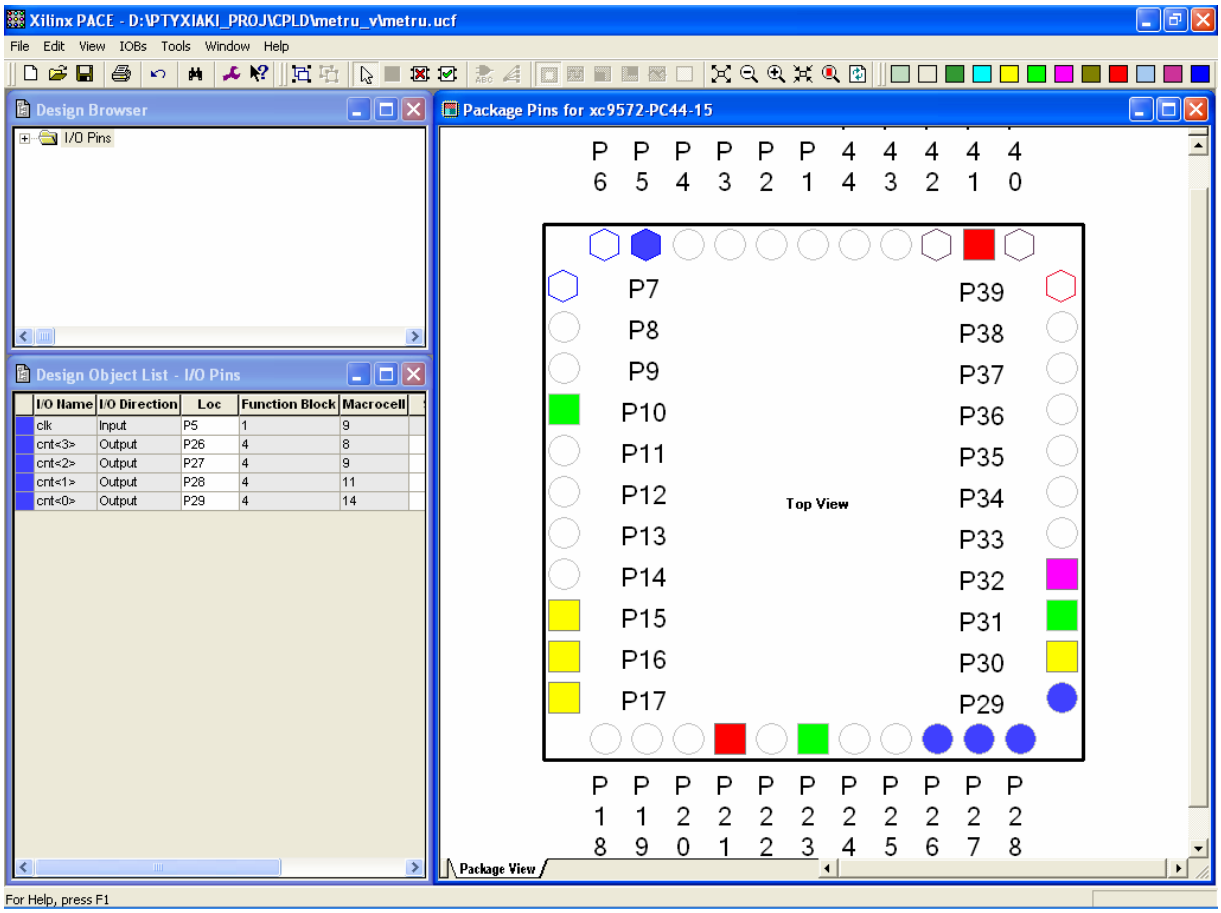


Εικόνα 2.8 VHDL αρχείο του κυκλώματος

Εικόνα 2.9 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.10 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.11 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.2 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ ΜΕΤΡΗΤΗ_2: ΔΥΑΔΙΚΟΣ ΣΥΓΧΡΟΝΟΣ ΦΘΙΝΟΝΤΑΣ ΧΩΡΗΤΙΚΟΤΗΤΑΣ 16

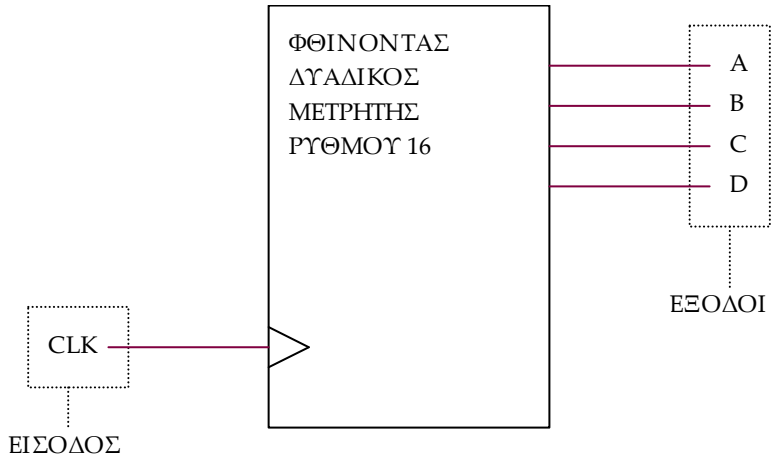
2.2.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

2.2.1.1 Σύντομη θεωρία

Οι δυαδικοί μετρητές (*binary counters*) είναι ακολουθιακά κυκλώματα μέτρησης του αριθμού παλμών του ρολογιού (*clock*) της εισόδου τους. Η ένδειξη της μέτρησης αυτής στην έξοδο γίνεται στο δυαδικό σύστημα αρίθμησης. Ονομάζονται φθίνοντες μετρητές στην περίπτωση που η ακολουθία των δυαδικών αριθμών ένδειξης της μέτρησης είναι φθίνουσα. Δομικό στοιχείο της κατασκευής τους αποτελούν οι δισταθείς πολυδονητές (*flip-flop*). Αποκαλούνται λοιπόν σύγχρονοι στην περίπτωση που τα ρολόγια όλων των δισταθών πολυδονητών που τους αποτελούν δέχονται παλμούς από το ρολόγι της εισόδου τους. Ο αριθμός δισταθών πολυδονητών που τους αποτελούν επίσης καθορίζει την χωρητικότητά τους (*module* ή *mod*). Συγκεκριμένα ένας μετρητής αποτελούμενος από n δισταθείς πολυδονητές ή εξόδους μετράει (στο δυαδικό σύστημα) μέχρι και την τιμή 2^n (mod-2^n) παλμών ρολογιού.

Ο φθίνοντας δυαδικός μετρητής χωρητικότητας 16 (mod-16) έχει σαν εισοδό του το ρολόγι CLK και τις τέσσερις εξόδους A, B, C, D (D=MSB). Είναι κατασκευασμένος από τέσσερις θετικής διέγερσης δισταθείς πολυδονητές (*flip-flop*) συνδεσμολογίας J-K για να μπορούν να μετρούν ως την χωρητικότητα ή το ρυθμό του (16 παλμούς ή *mod-16*). Στην **εικόνα 2.12** απεικονίζεται το γραφικό σύμβολο του παραπάνω μετρητή, στον **πίνακα 2.11** υπάρχει ο πίνακας διέγερσης του J-K δισταθή πολυδονητή. Στον **πίνακα 2.12** υπάρχει ο πίνακας διέγερσης του

μετρητή και στους κατοπινούς **πίνακες 2.13** ως και **2.16** υπάρχουν ενημερωμένοι οι αντίστοιχοι Karnaugh για τον κάθε πολυδονητή.



Εικόνα 2.12 Γραφικό σύμβολο φθίνοντα δυαδικού μετρητή mod-16

Πίνακας 2.11 Πίνακας διέγερσης δισταθή πολυδονητή J-K (flip-flop)

Q	**Q_+**	**J**	**K**
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Πίνακας 2.12 Πίνακας διέγερσης σύγχρονου φθίνοντα δυαδικού μετρητή χωρητικότητας 16

Παρούσα κατάσταση Q_n				Επόμενη κατάσταση Q_{n+1}											
MSB			**LSB**	**MSB**			**LSB**								
D	**C**	**B**	**A**	D_+	C_+	B_+	A_+	J_D	K_D	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	1	1	1	1	1	X	1	X	1	X	1	X
1	1	1	1	1	1	1	0	X	0	X	0	X	0	X	1
1	1	1	0	1	1	0	1	X	0	X	0	X	1	1	X
1	1	0	1	1	1	0	0	X	0	X	0	0	X	X	1
1	1	0	0	1	0	1	1	X	0	X	1	1	X	1	X
1	0	1	1	1	0	1	0	X	0	0	X	X	0	X	1
1	0	1	0	1	0	0	1	X	0	0	X	X	1	1	X
1	0	0	1	1	0	0	0	X	0	0	X	0	X	X	1
1	0	0	0	0	1	1	1	X	1	1	X	1	X	1	X
0	1	1	1	0	1	1	0	0	X	X	0	X	0	X	1
0	1	1	0	0	1	0	1	0	X	X	0	X	1	1	X
0	1	0	1	0	1	0	0	0	X	X	0	0	X	X	1
0	1	0	0	0	0	1	1	0	X	X	1	1	X	1	X
0	0	1	1	0	0	1	0	0	X	0	X	X	0	X	1
0	0	1	0	0	0	0	1	0	X	0	X	X	1	1	X
0	0	0	1	0	0	0	0	0	X	0	X	0	X	X	1

Έξοδος είναι η παρούσα κατάσταση Q_n.
X: αδιάφορη κατάσταση.
Η κωδικοποίηση των καταστάσεων έγινε με το δυαδικό κωδικό 8421.

Πίνακας 2.13 Πίνακας Karnaugh για τον πολυδονητή Α

Έξοδος: K_A

BA \ DC	00	01	11	10
00	X	X	X	X
01	1	1	1	1
11	1	1	1	1
10	X	X	X	X

Έξοδος: J_A

BA \ DC	00	01	11	10
00	1	1	1	1
01	X	X	X	X
11	X	X	X	X
10	1	1	1	1

Πίνακας 2.14 Πίνακας Karnaugh για τον πολυδονητή B

Έξοδος: J_B

BA \ DC	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	X	X	X	X
10	X	X	X	X

Έξοδος: K_B

BA \ DC	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	0	0
10	1	1	1	1

Πίνακας 2.15 Πίνακας Karnaugh για τον πολυδονητή C

Έξοδος: K_C

BA \ DC	00	01	11	10
00	X	1	1	X
01	X	0	0	X
11	X	0	0	X
10	X	0	0	X

Έξοδος: J_C

BA \ DC	00	01	11	10
00	1	X	X	1
01	0	X	X	0
11	0	X	X	0
10	0	X	X	0

Πίνακας 2.16 Πίνακας Karnaugh για τον πολυδονητή D

Έξοδος: K_D

BA \ DC	00	01	11	10
00	X	X	0	1
01	X	X	0	0
11	X	X	0	0
10	X	X	0	0

Έξοδος: J_D

BA \ DC	00	01	11	10
00	1	0	X	X
01	0	0	X	X
11	0	0	X	X
10	0	0	X	X

Συνολικό συμπέρασμα μετά την απλοποίηση των πινάκων Karnaugh:

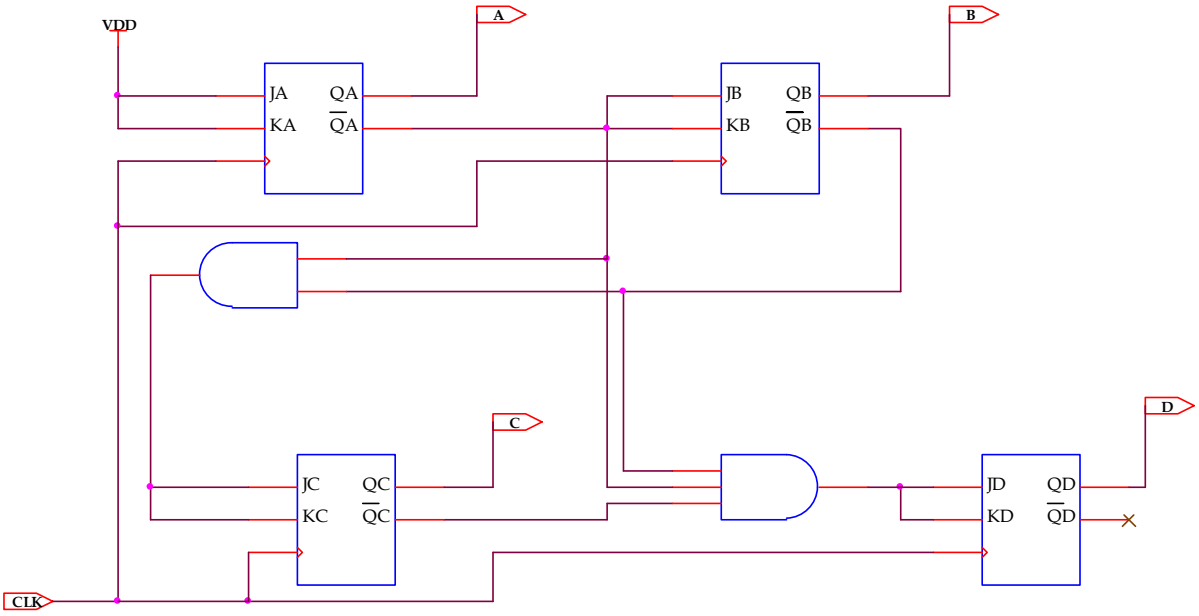
$$\boxed{K_A = 1}\ \boxed{J_A = 1}\qquad \boxed{K_B = \overline{A}}\ \boxed{J_B = \overline{A}}$$

$$\boxed{K_C = \overline{A}\cdot\overline{B}}\ \boxed{J_C = \overline{A}\cdot\overline{B}}\qquad \boxed{K_D = \overline{A}\cdot\overline{B}\cdot\overline{C}}\ \boxed{J_D = \overline{A}\cdot\overline{B}\cdot\overline{C}}$$

Εικόνα 2.13 Συνολική απλοποίηση πινάκων Karnaugh φθίνοντα δυαδικού μετρητή mod-16

2.2.1.2 Συμβατική ψηφιακή υλοποίηση

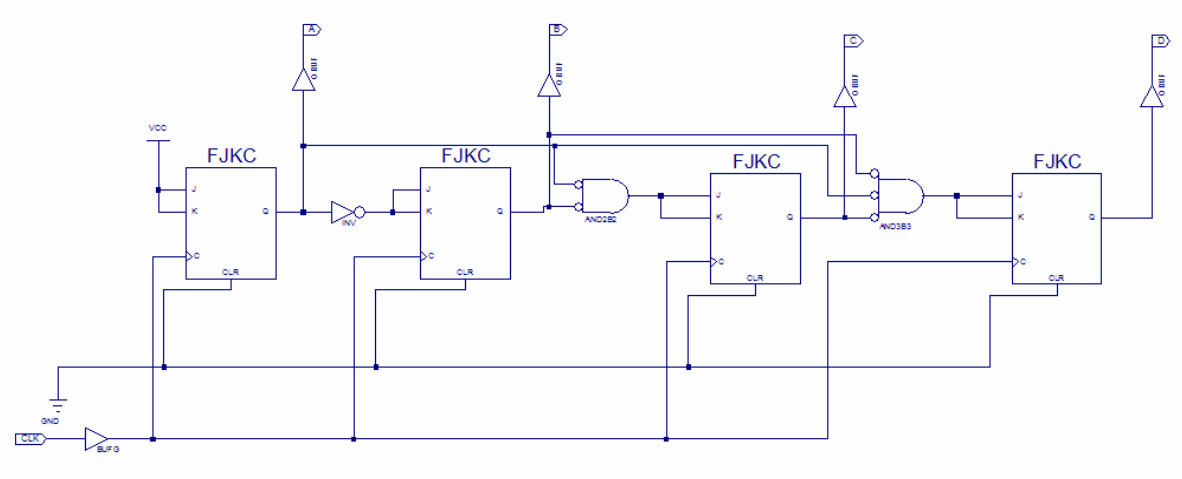
Η λογική εξίσωση της **εικόνας 2.13** παραπέμπει στο λογικό κύκλωμα υλοποίησης με συμβατικές λογικές πύλες και δισταθείς πολυδονητές J-K της παρακάτω **εικόνας 2.14**.



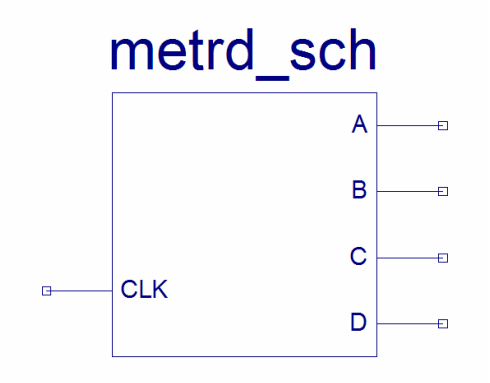
Εικόνα 2.14 Λογικό κύκλωμα υλοποίησης για φθίνοντα δυαδικό μετρητή mod-16

2.2.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

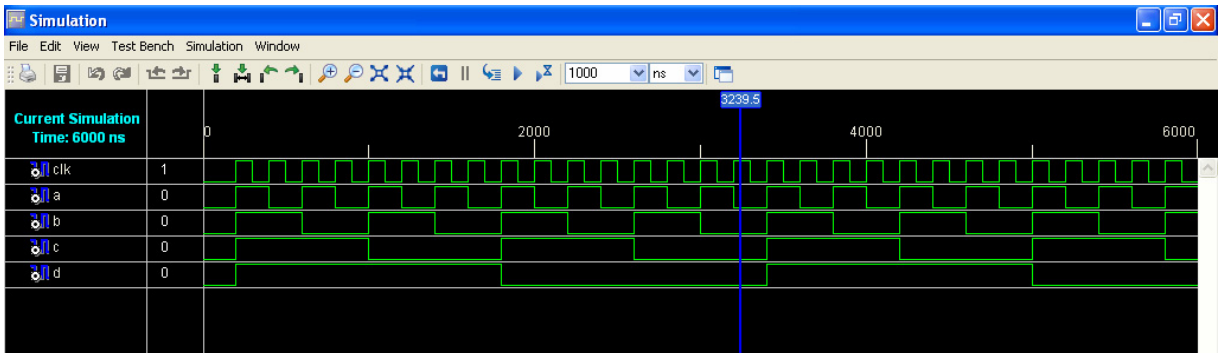
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 και του σύγχρονα αύξοντα μετρητή στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα ή χρήση καινούργιων υλικών. Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



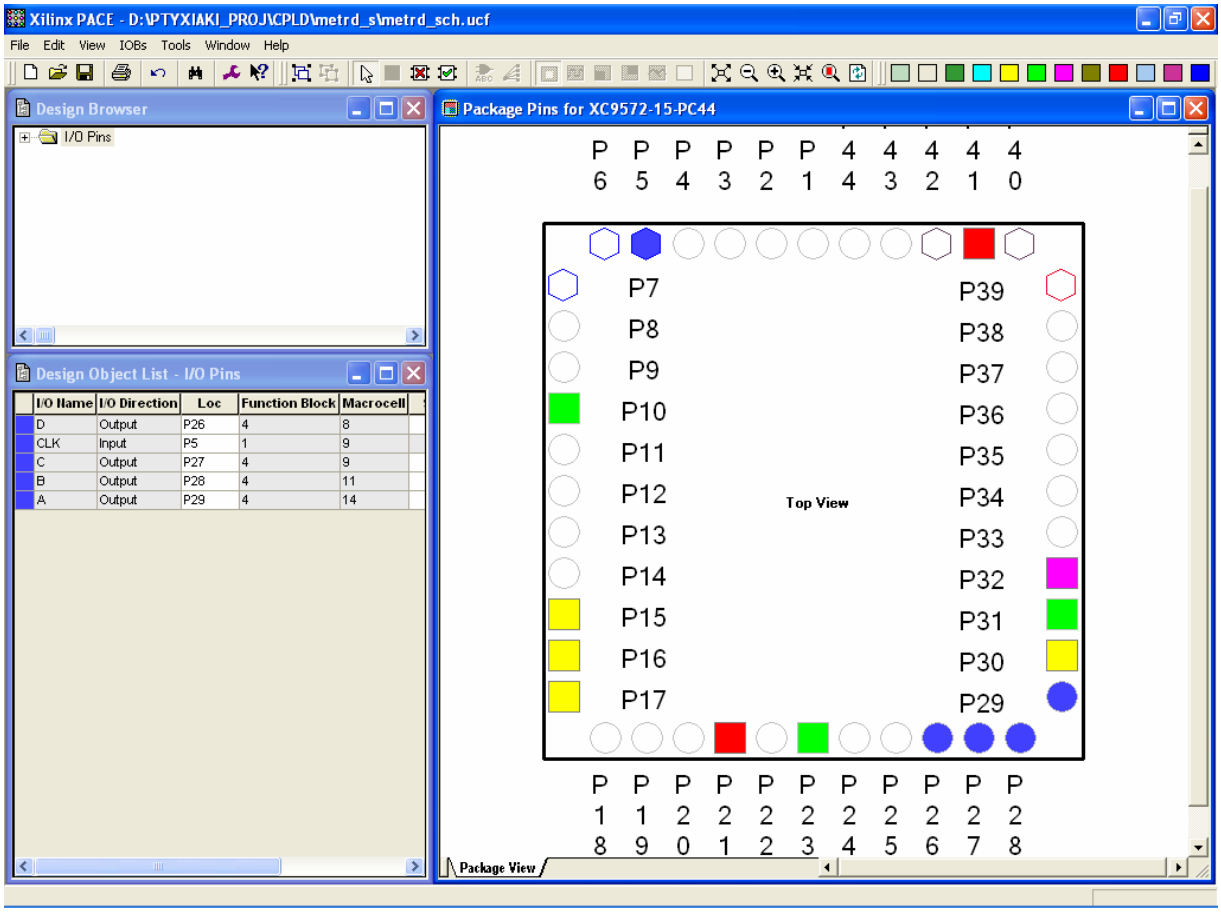
Εικόνα 2.15 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 2.16 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.17 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.18 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.2.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 και τον συγκριτή προσημασμένων αριθμών όπως αναφέρονται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 2.17 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

12	architecture BHV of metrd is	Αρχή δήλωσης αρχιτεκτονικής.
13	begin	Αρχή κώδικα αρχιτεκτονικής.
14	process (clk)	Αρχή της διεργασίας με λίστα ευαισθησίας την είσοδο **clk**.
15	variable temp: integer range 0 to 15;	Δήλωση της ακέραιας μεταβλητής **temp** με τιμές από το 0 ως το 15 του δεκαδικού.
16	begin	Αρχή ενεργοποίησης της διεργασίας.
17	if clk='1' and clk'event then	Αρχή εκτέλεσης **if-then**.
18	temp := temp - 1;	Μείωση της προηγούμενης τιμής του **temp** κατά 1.
19	end if;	Τέλος εκτέλεσης **if-then.**
20	cnt<=CONV_STD_LOGIC_VECTOR(temp, 4);	Μετατροπή δεδομένων μεταβλητής **temp** και μεταφορά τους στην έξοδο **cnt**.
21	end process;	Τέλος εκτέλεσης process.
22	end BHV;	Τέλος κώδικα αρχιτεκτονικής.

Σημειώσεις.

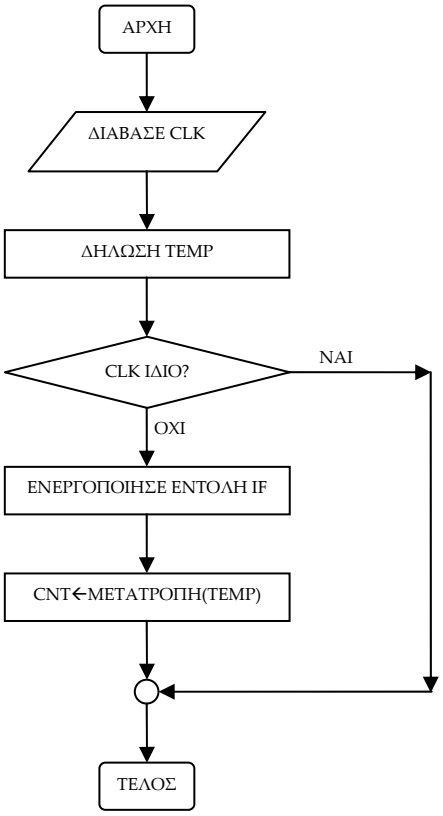
Γραμμή 16. Η ενεργοποίηση της διεργασίας δηλώνεται με τη λέξη begin. Η διεργασία ενεργοποιείται με την αλλαγή της λογικής κατάστασης της εισόδου **clk** της λίστας ευαισθησίας.

Γραμμή 17. Κατά την εκτέλεση της εντολής **if-then** έχουμε το λογικό έλεγχο για αλλαγή της εισόδου **clk** από λογικό 0 σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 18. Σε διαφορετική περίπτωση εκτελείται η γραμμή 19.

Γραμμή 20. Η συνάρτηση **conv_std_logic_vector** βρίσκεται στο πακέτο **std_logic_arith**. Η συνάρτηση αυτή μετατρέπει την ακέραια μεταβλητή σε μια τιμή τεσσάρων δυαδικών ψηφίων τύπου std_logic και μεταφέρει το αποτέλεσμα της μετατροπής στην έξοδο.

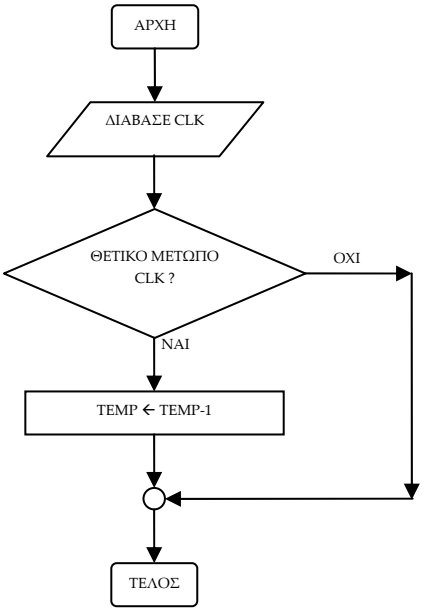
Πίνακας 2.18 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (clk)

variable temp: integer range 0 to 15;

begin

        if …end if;

cnt<=CONV_STD_LOGIC_VECTOR(temp, 4);

end process;
```



Πίνακας 2.19 Η εντολή **if-then** και το **διάγραμμα ροής**

```
if clk='1' and clk'event then

        temp := temp - 1;

end if;
```



Σημείωση. Στα παραπάνω διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

2.2.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

metrd_vhd.vhd

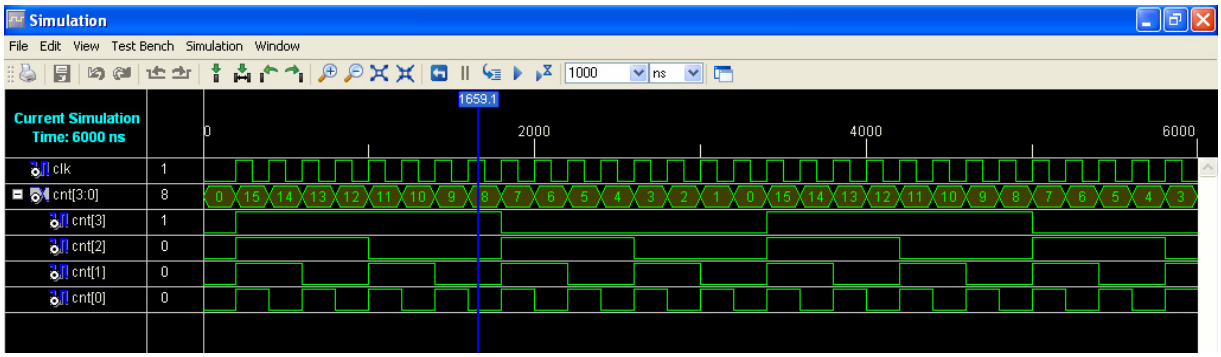
File Edit View Window

```
1  ----------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;
6  ----------------------------------------------------
7  entity metrd_vhd is
8      Port ( clk : in  STD_LOGIC;
9             cnt : out  STD_LOGIC_VECTOR (3 downto 0));
10 end metrd_vhd;
11 ----------------------------------------------------
12 architecture BHV of metrd_vhd is
13 begin
14    process (clk)
15    variable temp: integer range 0 to 15;
16    begin
17       if clk='1' and clk'event then
18          temp := temp - 1;
19       end if;
20    cnt<=CONV_STD_LOGIC_VECTOR(temp, 4);
21    end process;
22 end BHV;
```

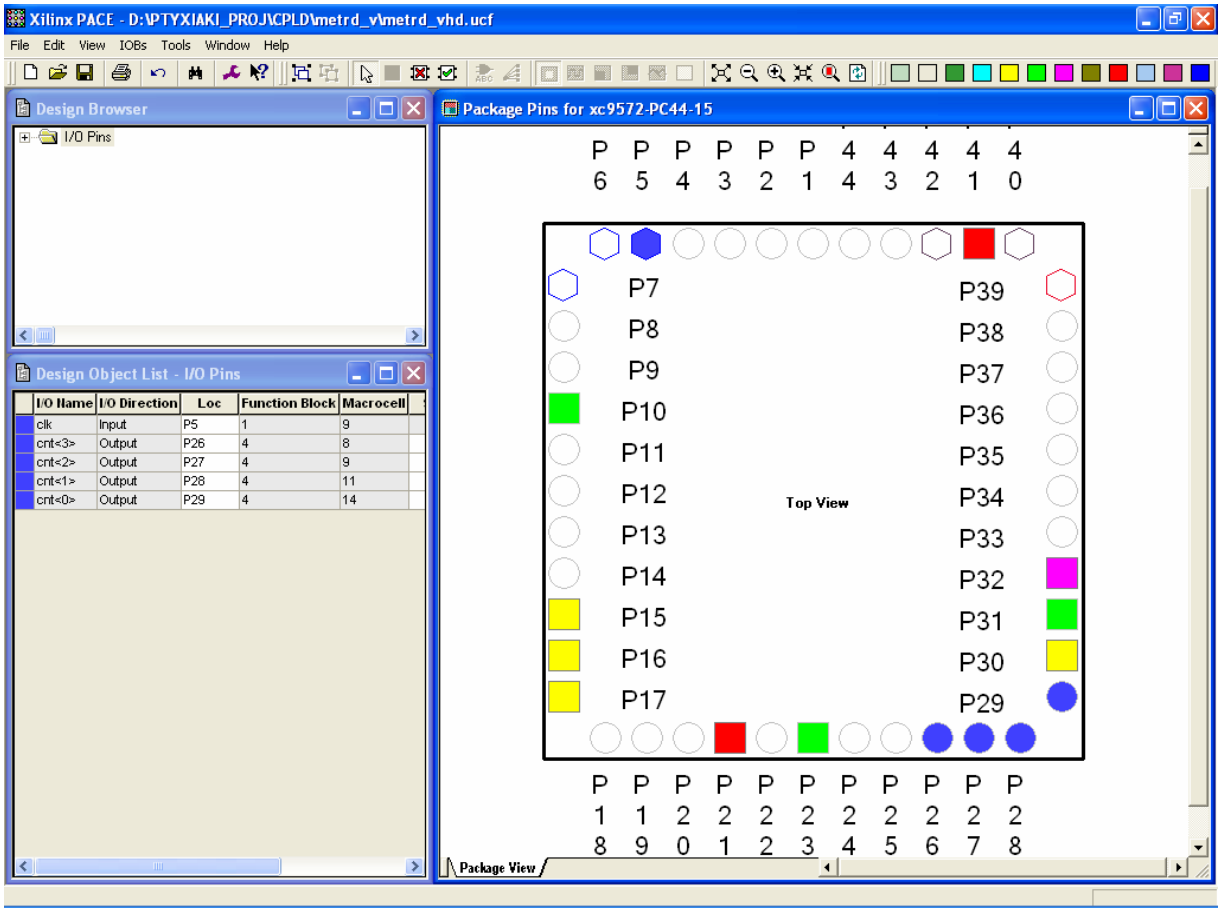
CAPS NUM SCRL LOC VHDL

Εικόνα 2.19 VHDL αρχείο του κυκλώματος

Εικόνα 2.20 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.21 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.22 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.3 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ ΜΕΤΡΗΤΗ_3: ΔΥΑΔΙΚΟΣ ΣΥΓΧΡΟΝΟΣ ΑΥΞΟΝΤΑΣ ΚΑΙ ΦΘΙΝΟΝΤΑΣ ΧΩΡΗΤΙΚΟΤΗΤΑΣ 16

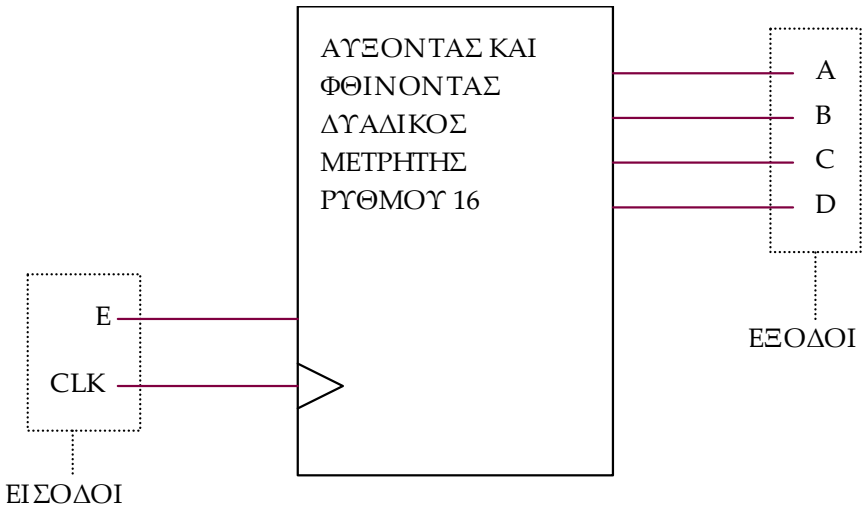
2.3.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

2.3.1.1 Σύντομη θεωρία

Οι δυαδικοί μετρητές (*binary counters*) είναι ακολουθιακά κυκλώματα μέτρησης του αριθμού παλμών του ρολογιού (*clock*) της εισόδου τους. Η ένδειξη της μέτρησης αυτής στην έξοδο γίνεται στο δυαδικό σύστημα αρίθμησης. Ονομάζονται αύξοντες και φθίνοντες μετρητές στην περίπτωση που η ακολουθία των δυαδικών αριθμών ένδειξης της μέτρησης είναι αύξουσα και φθίνουσα. (Λειτουργικά μπορούν να είναι μόνο αύξοντες ή μόνο φθίνοντες για την ίδια χρονική στιγμή μέσω ελέγχου. Ο έλεγχος αυτός επιτρέπει την αλλαγή λειτουργίας τους από αύξουσα σε φθίνουσα και το αντίθετο για την επόμενη χρονική στιγμή.) Δομικό στοιχείο της κατασκευής τους αποτελούν οι δισταθείς πολυδονητές (*flip-flop*). Αποκαλούνται λοιπόν σύγχρονοι στην περίπτωση που τα ρολόγια όλων των δισταθών πολυδονητών που τους αποτελούν δέχονται παλμούς από το ρολόγι της εισόδου τους. Ο αριθμός δισταθών πολυδονητών που τους αποτελούν επίσης καθορίζει την χωρητικότητά τους (*module* ή *mod*). Συγκεκριμένα ένας μετρητής αποτελούμενος από n δισταθείς πολυδονητές ή εξόδους μετράει (στο δυαδικό σύστημα) μέχρι και την τιμή 2^n (mod-2^n) παλμών ρολογιού.

Ο αύξοντας και φθίνοντας δυαδικός μετρητής χωρητικότητας 16 (mod-16) έχει σαν εισόδους του το ρολόγι CLK και το δυαδικό ψηφίο ελέγχου E για αύξουσα ή φθίνουσα λειτουργία. Επίσης διαθέτει τις τέσσερις εξόδους A, B, C, D (D= MSB).

Είναι κατασκευασμένος από τέσσερις θετικής διέγερσης δισταθείς πολυδονητές (*flip-flop*) συνδεσμολογίας J-K για να μπορούν να μετρούν ως την χωρητικότητα ή το ρυθμό του (16 παλμούς ή *mod-16*). Στην **εικόνα 2.23** απεικονίζεται το γραφικό σύμβολο του παραπάνω μετρητή, στον **πίνακα 2.20** υπάρχει ο πίνακας διέγερσης του J-K δισταθή πολυδονητή. Στον **πίνακα 2.21** υπάρχει ο πίνακας διέγερσης του μετρητή και στους κατοπινούς **πίνακες 2.22** ως και **2.25** υπάρχουν ενημερωμένοι οι αντίστοιχοι Karnaugh για τον κάθε πολυδονητή.



Εικόνα 2.23 Γραφικό σύμβολο αύξοντα και φθίνοντα δυαδικού μετρητή mod-16

Πίνακας 2.20 Πίνακας διέγερσης δισταθή πολυδονητή J-K (flip-flop)

Q	**Q_+**	**J**	**K**
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Πίνακας 2.21 Πίνακας διέγερσης σύγχρονου αύξοντα και φθίνοντα δυαδικού μετρητή χωρητικότητας 16

	Παρούσα κατάσταση Q_n				Επόμενη κατάσταση Q_{n+1}											
	MSB			**LSB**	**MSB**			**LSB**								
E	D	C	B	A	D_+	C_+	B_+	A_+	J_D	K_D	J_C	K_C	J_B	K_B	J_A	K_A
1	0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
1	0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
1	0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
1	0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
1	0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
1	0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
1	0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
1	0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
1	1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
1	1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
1	1	1	1	0	1	1	1	1	X	0	X	0	X	0	1	X
1	1	1	1	1	0	0	0	0	X	1	X	1	X	1	X	1
0	0	0	0	0	1	1	1	1	1	X	1	X	1	X	1	X
0	1	1	1	1	1	1	1	0	X	0	X	0	X	0	X	1
0	1	1	1	0	1	1	0	1	X	0	X	0	X	1	1	X
0	1	1	0	1	1	1	0	0	X	0	X	0	0	X	X	1
0	1	1	0	0	1	0	1	1	X	0	X	1	1	X	1	X
0	1	0	1	1	1	0	1	0	X	0	0	X	X	0	X	1
0	1	0	1	0	1	0	0	1	X	0	0	X	X	1	1	X
0	1	0	0	1	1	0	0	0	X	0	0	X	0	X	X	1
0	1	0	0	0	0	1	1	1	X	1	1	X	1	X	1	X
0	0	1	1	1	0	1	1	0	0	X	X	0	X	0	X	1
0	0	1	1	0	0	1	0	1	0	X	X	0	X	1	1	X
0	0	1	0	1	0	1	0	0	0	X	X	0	0	X	X	1
0	0	1	0	0	0	0	1	1	0	X	X	1	1	X	1	X
0	0	0	1	1	0	0	1	0	0	X	0	X	X	0	X	1
0	0	0	1	0	0	0	0	1	0	X	0	X	X	1	1	X
0	0	0	0	1	0	0	0	0	0	X	0	X	0	X	X	1

Έξοδος είναι η παρούσα κατάσταση Q_n. (X: αδιάφορη κατάσταση.)
Η κωδικοποίηση των καταστάσεων έγινε με το δυαδικό κωδικό 8421.
Ο μετρητής είναι αύξοντας για E=1 και φθίνοντας για E=0

Πίνακας 2.22 Πίνακας Karnaugh πέντε μεταβλητών για τον πολυδονητή Α

Έξοδος: K_A

E=1

BA \ DC	00	01	11	10
00	X	X	X	X
01	1	1	1	1
11	1	1	1	1
10	X	X	X	X

E=0

BA \ DC	00	01	11	10
00	X	X	X	X
01	1	1	1	1
11	1	1	1	1
10	X	X	X	X

Έξοδος: J_A

E=1

BA \ DC	00	01	11	10
00	X	X	X	X
01	1	1	1	1
11	1	1	1	1
10	X	X	X	X

E=0

BA \ DC	00	01	11	10
00	X	X	X	X
01	1	1	1	1
11	1	1	1	1
10	X	X	X	X

Πίνακας 2.23 Πίνακας Karnaugh πέντε μεταβλητών για τον πολυδονητή Β

Έξοδος: K_B

E=1

BA \ DC	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	1	1	1	1
10	0	0	0	0

E=0

BA \ DC	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	0	0
10	1	1	1	1

Έξοδος: J_B

E=1

BA \ DC	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	X	X	X	X
10	X	X	X	X

E=0

BA \ DC	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	X	X	X	X
10	X	X	X	X

Πίνακας 2.24 Πίνακας Karnaugh πέντε μεταβλητών για τον πολυδονητή C

Έξοδος: K_C

E=1

BA \ DC	00	01	11	10
00	X	0	0	X
01	X	0	0	X
11	X	1	1	X
10	X	0	0	X

E=0

BA \ DC	00	01	11	10
00	X	1	1	X
01	X	0	0	X
11	X	0	0	X
10	X	0	0	X

Έξοδος: J_C

E=1

BA \ DC	00	01	11	10
00	0	X	X	X
01	0	X	X	X
11	1	X	X	1
10	0	X	X	X

E=0

BA \ DC	00	01	11	10
00	1	X	X	1
01	0	X	X	0
11	0	X	X	0
10	0	X	X	0

Πίνακας 2.25 Πίνακας Karnaugh πέντε μεταβλητών για τον πολυδονητή D

Έξοδος: K_D

E=1

BA \ DC	00	01	11	10
00	X	X	0	0
01	X	X	0	0
11	X	X	1	0
10	X	X	0	0

E=0

BA \ DC	00	01	11	10
00	X	X	0	1
01	X	X	0	0
11	X	X	0	0
10	X	X	0	0

Έξοδος: J_D

E=1

BA \ DC	00	01	11	10
00	0	0	X	X
01	0	0	X	X
11	0	1	X	X
10	0	0	X	X

E=0

BA \ DC	00	01	11	10
00	1	0	X	X
01	0	0	X	X
11	0	0	X	X
10	0	0	X	X

Συνολικό συμπέρασμα μετά την απλοποίηση των πινάκων Karnaugh:

$$K_A = E\cdot 1+\overline{E}\cdot 1=E+\overline{E}=1 \qquad J_A = E\cdot 1+\overline{E}\cdot 1=E+\overline{E}=1$$

$$K_B = E\cdot A+\overline{E}\cdot\overline{A}=E\oplus A \qquad J_B = E\cdot A+\overline{E}\cdot\overline{A}=E\oplus A$$

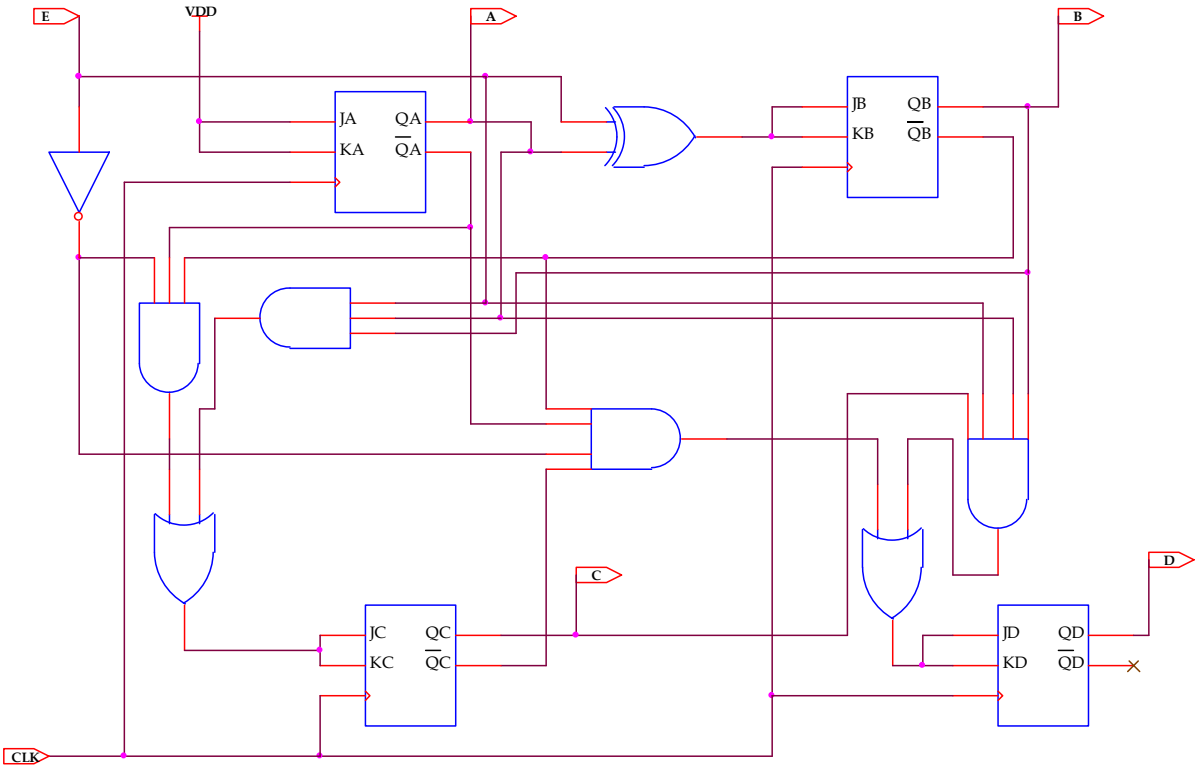
$$K_B = E\cdot A+\overline{E}\cdot\overline{A}=E\oplus A \qquad J_B = E\cdot A+\overline{E}\cdot\overline{A}=E\oplus A$$

$$K_D = E\cdot A\cdot B\cdot C+\overline{E}\cdot\overline{A}\cdot\overline{B}\cdot\overline{C} \qquad J_D = E\cdot A\cdot B\cdot C+\overline{E}\cdot\overline{A}\cdot\overline{B}\cdot\overline{C}$$

Εικόνα 2.24 Συνολική απλοποίηση πινάκων Karnaugh αύξοντα και φθίνοντα δυαδικού μετρητή mod-16

2.3.1.2 Συμβατική ψηφιακή υλοποίηση

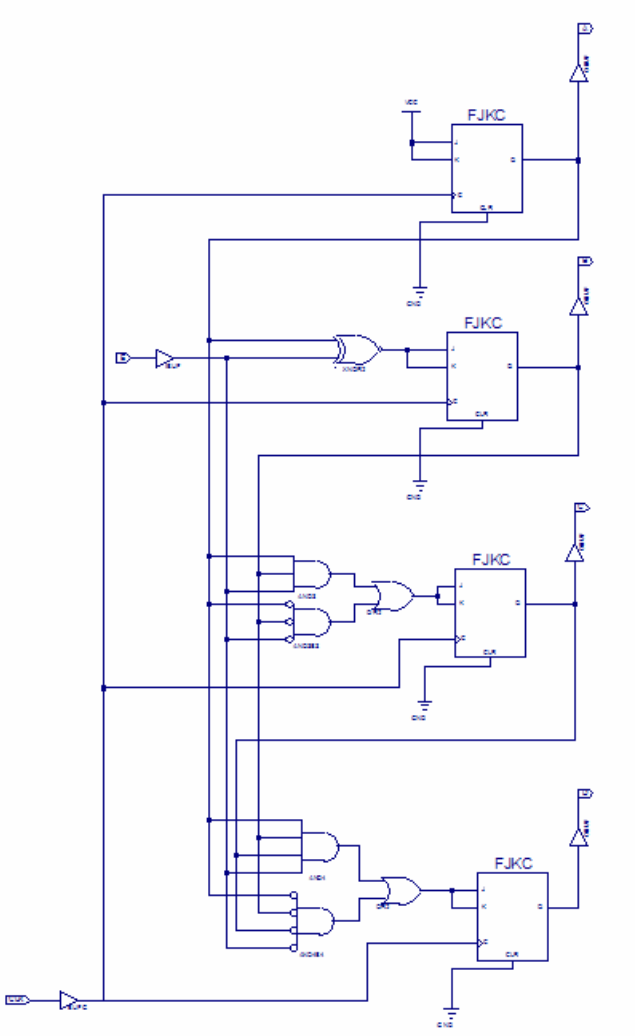
Η λογική εξίσωση της **εικόνας 2.24** παραπέμπει στο λογικό κύκλωμα υλοποίησης με συμβατικές λογικές πύλες και δισταθείς πολυδονητές J-K της παρακάτω **εικόνας 2.25**.



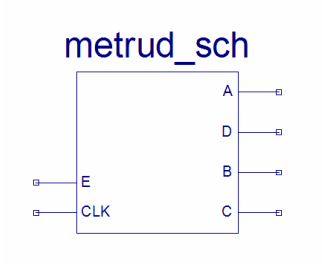
Εικόνα 2.25 Λογικό κύκλωμα υλοποίησης αύξοντα και φθίνοντα δυαδικού μετρητή mod-16

2.3.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

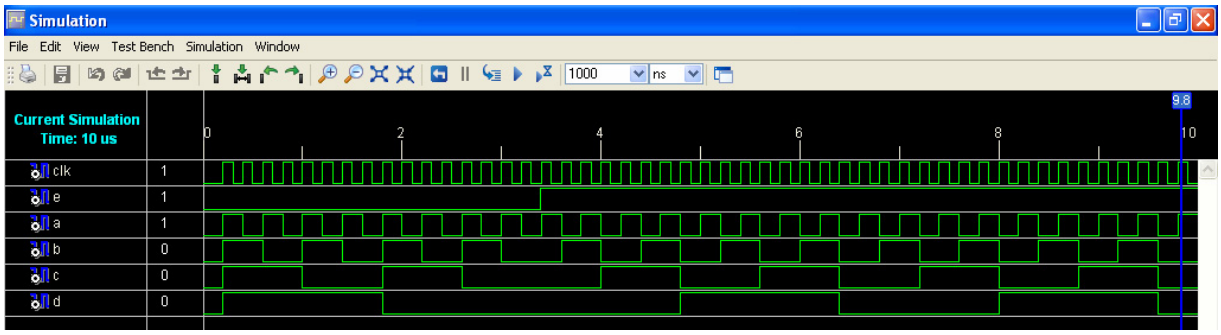
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 και του σύγχρονα αύξοντα μετρητή στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα ή χρήση καινούργιων υλικών. Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



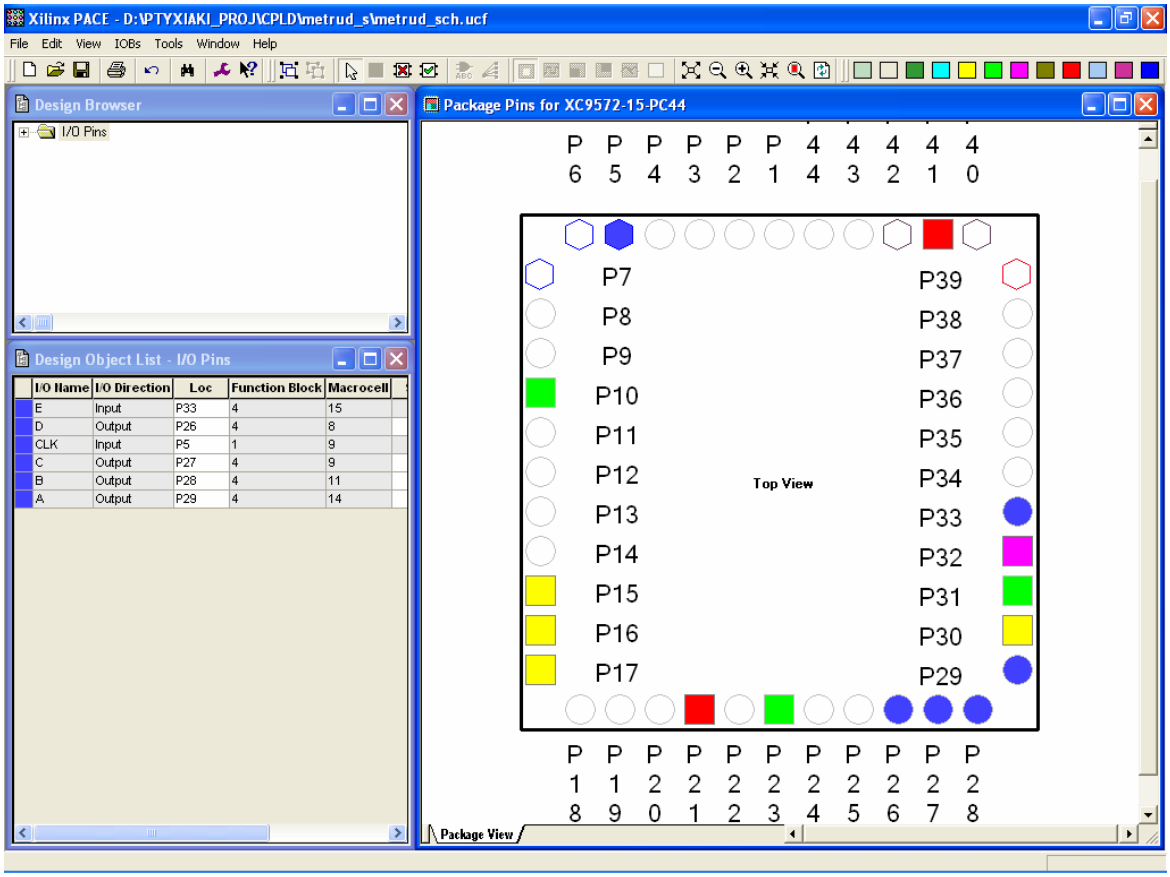
Εικόνα 2.26 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 2.27 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.28 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.29 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.3.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 και τον συγκριτή προσημασμένων αριθμών όπως αναφέρονται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 2.26 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

12	architecture BHV of metrud is	Αρχή δήλωσης αρχιτεκτονικής.
13	begin	Αρχή κώδικα αρχιτεκτονικής.
14	process (clk)	Αρχή της διεργασίας με λίστα ευαισθησίας την είσοδο **clk**.
15	variable temp: integer range 0 to 15;	Δήλωση της ακέραιας μεταβλητής **temp** με τιμές από το 0 ως το 15 του δεκαδικού.
16	begin	Αρχή ενεργοποίησης της διεργασίας.
17	if clk='1' and clk'event then	Αρχή εκτέλεσης **if-then**.
18	if e='1' then	Αρχή εκτέλεσης **if-then-else.**
19	temp := temp + 1;	Αύξηση της προηγούμενης τιμής του **temp** κατά 1.
20	else	Διακλάδωση εντολής **if-then-else.**
21	temp := temp - 1;	Μείωση της προηγούμενης τιμής του **temp** κατά 1.
22	end if;	Τέλος εκτέλεσης **if-then-else.**
23	end if;	Τέλος εκτέλεσης **if-then.**
24	cnt<=CONV_STD_LOGIC_VECTOR(temp, 4);	Μετατροπή δεδομένων μεταβλητής **temp** και μεταφορά τους στην έξοδο **cntud**.
25	end process;	Τέλος εκτέλεσης process.
26	end BHV;	Τέλος κώδικα αρχιτεκτονικής.

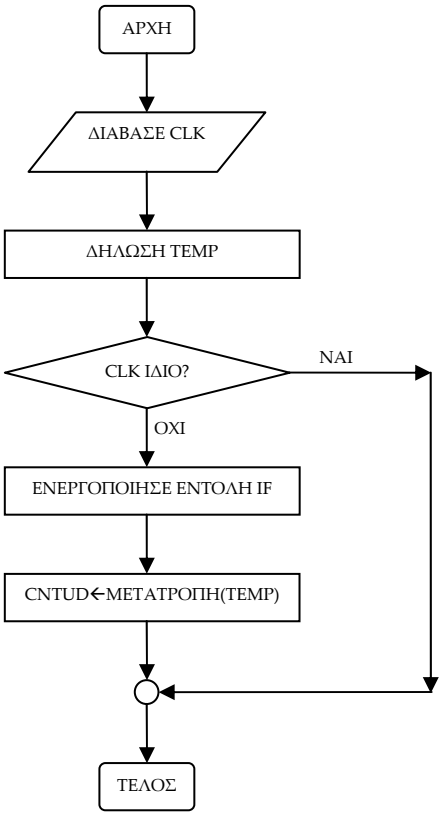
Σημειώσεις.

Γραμμή 16. Η ενεργοποίηση της διεργασίας δηλώνεται με τη λέξη begin. Η διεργασία

ενεργοποιείται με την αλλαγή της λογικής κατάστασης της εισόδου **clk** της λίστας ευαισθησίας.
Γραμμή 17. Κατά την εκτέλεση της εντολής **if-then** έχουμε το λογικό έλεγχο για αλλαγή της εισόδου **clk** από λογικό 0 σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 18. Σε διαφορετική περίπτωση εκτελείται η γραμμή 23.
Γραμμή 18. Κατά την εκτέλεση της εντολής **if-then-else** έχουμε το λογικό έλεγχο για την κατάσταση της εισόδου **e** σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 19. Σε διαφορετική περίπτωση (γραμμή 20) εκτελείται η γραμμή 21.
Γραμμή 24. Η συνάρτηση **conv_std_logic_vector** βρίσκεται στο πακέτο **std_logic_arith**. Η συνάρτηση αυτή μετατρέπει την ακέραια μεταβλητή σε μια τιμή τεσσάρων δυαδικών ψηφίων τύπου std_logic και μεταφέρει το αποτέλεσμα της μετατροπής στην έξοδο.

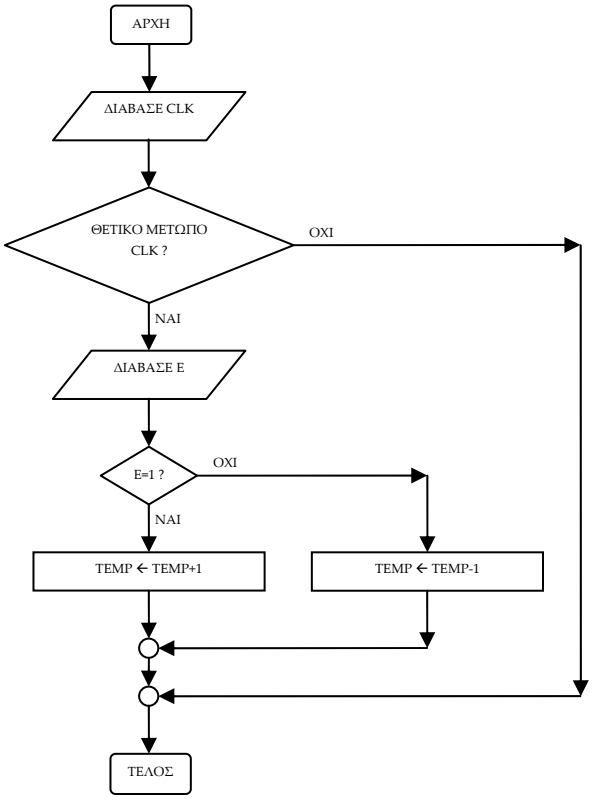
Πίνακας 2.27 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (clk)

variable temp: integer range 0 to 15;

begin

        if … (if …end if )…end if;

cntud<=CONV_STD_LOGIC_VECTOR(temp, 4);

end process;
```



Πίνακας 2.28 Η εντολή **if-then** και το **διάγραμμα ροής**

```
if clk='1' and clk'event then

   if e='1' then

      temp := temp + 1; else  temp := temp - 1;

   end if;
end if;
```



Σημείωση. Στα παραπάνω διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

2.3.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

metrud_vhd.vhd

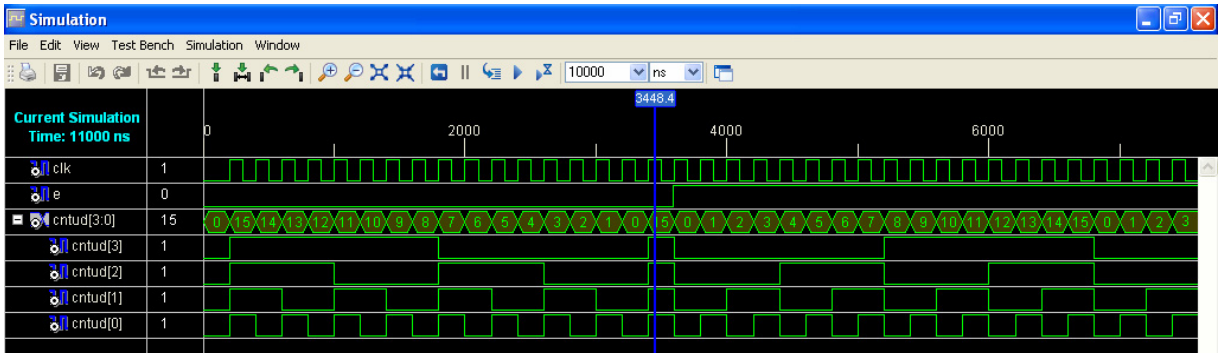
File Edit View Window

```
1  ----------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;
6  ----------------------------------------------------
7  entity metrud_vhd is
8      Port ( clk, e : in  STD_LOGIC;
9             cntud : out  STD_LOGIC_VECTOR (3 downto 0));
10 end metrud_vhd;
11 ----------------------------------------------------
12 architecture BHV of metrud_vhd is
13 begin
14    process (clk)
15    variable temp: integer range 0 to 15;
16    begin
17       if clk='1' and clk'event then
18          if e='1' then
19             temp := temp + 1;
20             else
21             temp := temp - 1;
22          end if;
23       end if;
24    cntud<=CONV_STD_LOGIC_VECTOR(temp, 4);
25    end process;
26 end BHV;
```

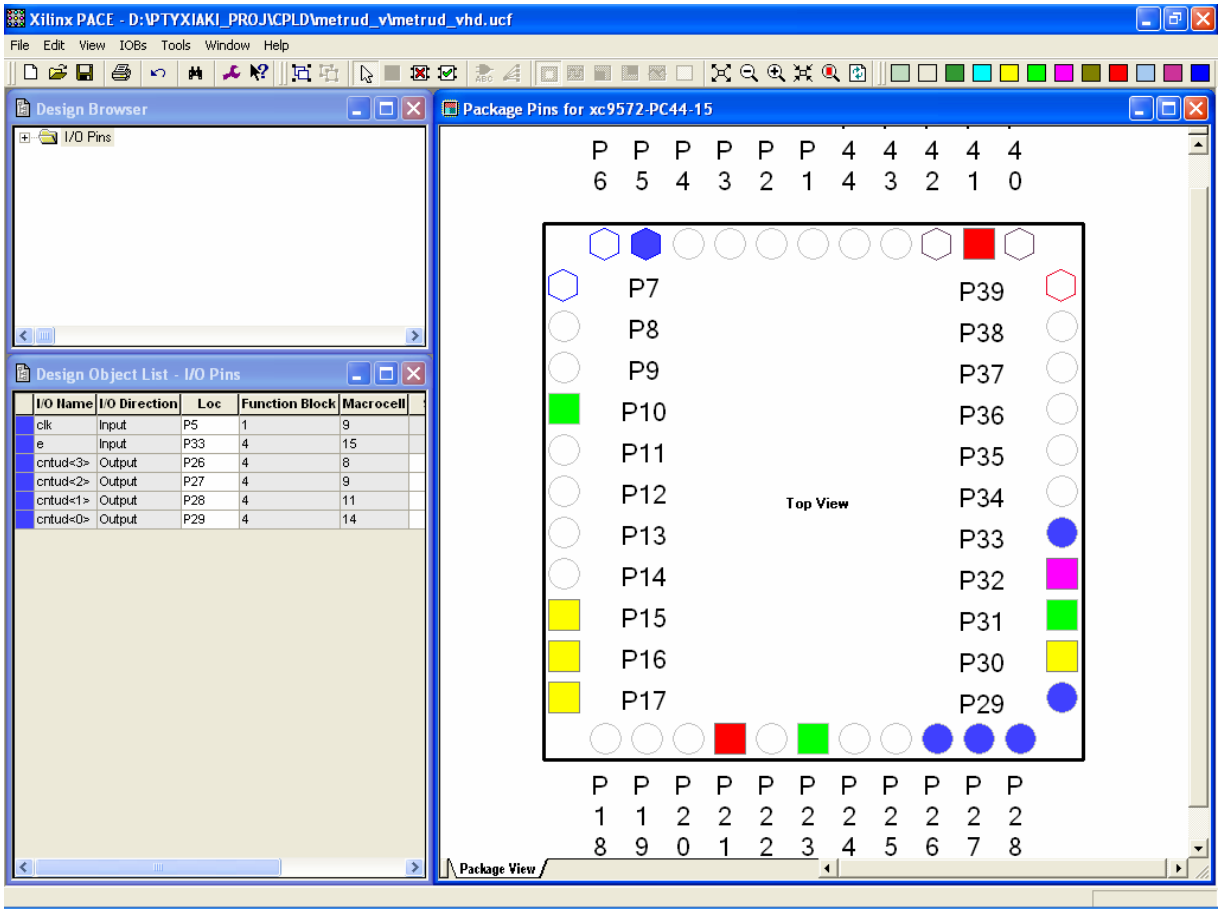
CAPS NUM SCRL LOC VHDL

Εικόνα 2.30 VHDL αρχείο του κυκλώματος

Εικόνα 2.31 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.32 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



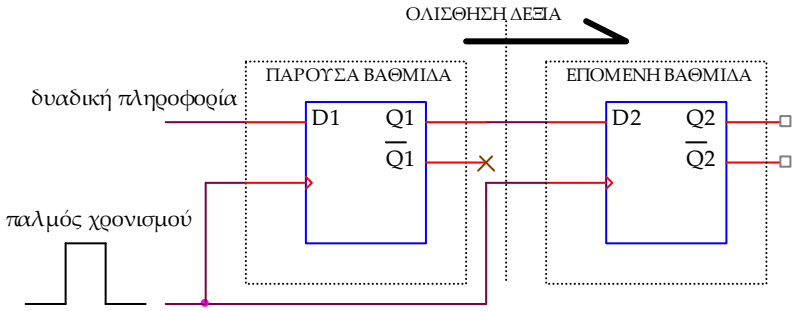
Εικόνα 2.33 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.4 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ ΚΑΤΑΧΩΡΗΤΗ_1: ΔΕΞΙΑΣ ΟΛΙΣΘΗΣΗΣ ΣΕΙΡΙΑΚΗΣ ΕΙΣΟΔΟΥ - ΣΕΙΡΙΑΚΗΣ ΕΞΟΔΟΥ ΧΩΡΗΤΙΚΟΤΗΤΑΣ 8 ΨΗΦΙΩΝ

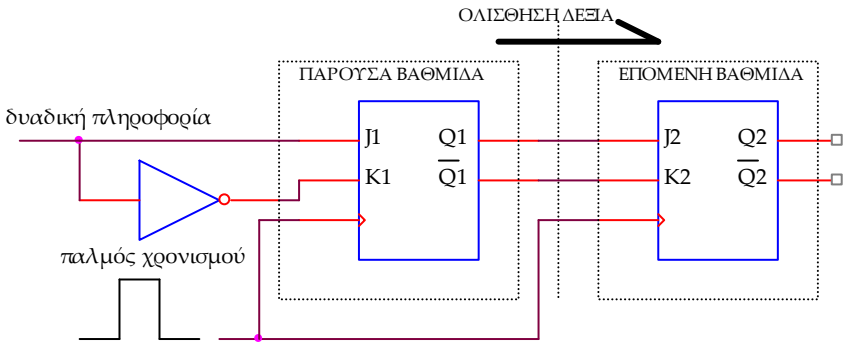
2.4.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

2.4.1.1 Σύντομη θεωρία

Οι καταχωρητές ολίσθησης (*shift registers*) είναι ακολουθιακά κυκλώματα μεταφοράς και προσωρινής αποθήκευσης δυαδικής πληροφορίας. Δομικό στοιχείο της κατασκευής τους αποτελούν οι δισταθείς πολυδονητές (*flip-flop*). Ο καθένας από τους δισταθείς πολυδονητές έχει ικανότητα αποθήκευσης ενός δυαδικού ψηφίου. Ο αριθμός λοιπόν των δισταθών πολυδονητών που δομούν τον καταχωρητή ολίσθησης εκφράζει τον αριθμό των δυαδικών ψηφίων που μπορεί να αποθηκεύσει αυτός δηλαδή τη χωρητικότητα αποθήκευσης του καταχωρητή. Η κατά σειρά διασύνδεση μεταξύ των δισταθών πολυδονητών μέσα σε ένα καταχωρητή ολίσθησης επιτρέπει τόσο την μεταφορά της πληροφορίας (ολίσθηση) προς μια κατεύθυνση όσο και την προσωρινή αποθήκευση της πληροφορίας. Η συνδεσμολογία αυτή επειδή εξυπηρετεί τη λειτουργία όλων των καταχωρητών ολίσθησης έχει γενικευμένη χρήση. Η συγκεκριμένη συνδεσμολογία δείχνεται στην **εικόνα 2.34** με την προς τα δεξιά ολίσθηση της πληροφορίας από τον ένα πολυδονητή στον άλλο με τον παλμό χρονισμού. Οι δύο πολυδονητές τύπου D είναι μέρος ενός ευρύτερου καταχωρητή ολίσθησης. Στην **εικόνα 2.35** δείχνεται το αντίστοιχο για δύο πολυδονητές τύπου J-K. Αναλόγως τώρα την κατεύθυνση ολίσθησης της πληροφορίας (αριστερά ή δεξιά) έχουμε καταχωρητές αριστερής ή δεξιάς αντίστοιχα ολίσθησης.



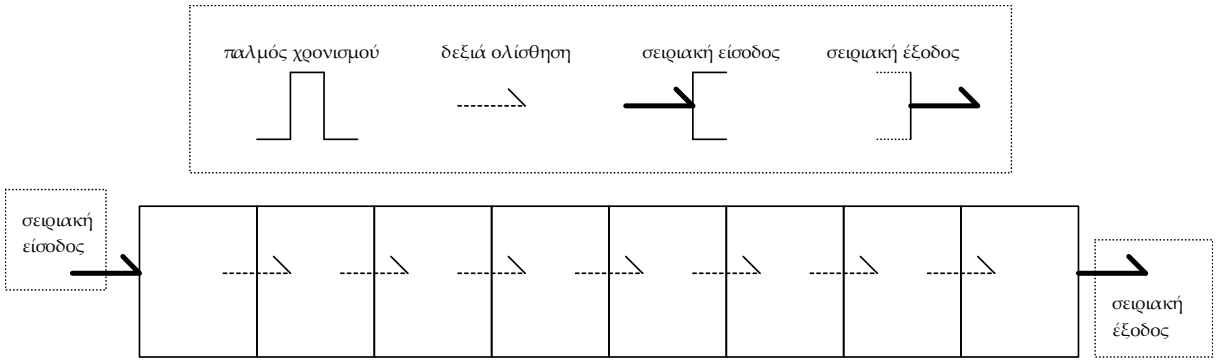
Εικόνα 2.34 Ολίσθηση πληροφορίας μεταξύ πολυδονητών τύπου D



Εικόνα 2.35 Ολίσθηση πληροφορίας μεταξύ πολυδονητών τύπου J-K

Η είσοδος της πληροφορίας γίνεται είτε σειριακά είτε παράλληλα. Οι καταχωρητές ολίσθησης σειριακής εισόδου δέχονται την πληροφορία ενός μόνο δυαδικού ψηφίου σε ένα δισταθή πολυδονητή τους σε κάθε παλμό χρονισμού. Στην περίπτωση των περισσοτέρων του ενός δυαδικών ψηφίων πληροφορίας αυτά ολισθαίνουν σε κάθε παλμό χρονισμού μέχρι και την καταχώρηση του τελευταίου σε δισταθή πολυδονητή του καταχωρητή. Οι καταχωρητές ολίσθησης παράλληλης εισόδου επιτρέπουν την είσοδο όλων των δυαδικών ψηφίων της πληροφορίας στην αντιστοιχία ενός μόνο δυαδικού ψηφίου ανά δισταθή πολυδονητή τους σε ένα παλμό χρονισμού. Οι καταχωρητές ολίσθησης σειριακής εξόδου επιτρέπουν έξοδο ενός μόνο δυαδικού ψηφίου της πληροφορίας από τον τελευταίο δισταθή πολυδονητή τους σε κάθε παλμό χρονισμού. Οι καταχωρητές ολίσθησης παράλληλης εξόδου επιτρέπουν την έξοδο όλων των δυαδικών ψηφίων της πληροφορίας στην αντιστοιχία ενός μόνο δυαδικού ψηφίου ανά δισταθή πολυδονητή τους σε ένα παλμό χρονισμού.

Ο καταχωρητής δεξιάς ολίσθησης σειριακής εισόδου και εξόδου χωρητικότητας οκτώ ψηφίων έχει σαν εισόδους του το δυαδικό ψηφίο χρονισμού CLK, το δυαδικό ψηφίο σύγχρονου μηδενισμού RST και το δυαδικό ψηφίο SI για τη σειριακή είσοδο της πληροφορίας. Η σειριακή έξοδός του είναι το δυαδικό ψηφίο SO. Είναι κατασκευασμένος από οκτώ θετικής διέγερσης δισταθείς πολυδονητές (*flip-flop*) D τύπου σε σειρά συνδεδεμένους μεταξύ τους. Στην **εικόνα 2.36** τα ορθογώνια παριστούν τους οκτώ πολυδονητές που με τον παλμό χρονισμού ένα δυαδικό ψηφίο μεταβαίνει από τον ένα στον άλλο. Στην **εικόνα 2.37** απεικονίζεται το γραφικό σύμβολο του παραπάνω καταχωρητή.

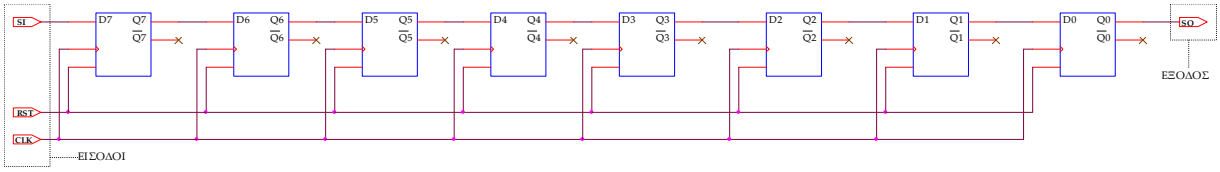


Εικόνα 2.36 Καταχωρητής δεξιάς ολίσθησης σειριακής εισόδου και εξόδου 8 ψηφίων

Εικόνα 2.37 Γραφικό σύμβολο καταχωρητή δεξιάς ολίσθησης σειριακής εισόδου και εξόδου 8 ψηφίων

2.4.1.2 Συμβατική ψηφιακή υλοποίηση

Η γενικευμένη συνδεσμολογία για τους καταχωρητές ολίσθησης παραπέμπει στην υλοποίηση με δισταθείς πολυδονητές D τύπου της παρακάτω **εικόνας 2.38**.

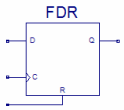


Εικόνα 2.38 Λογικό κύκλωμα υλοποίησης καταχωρητή δεξιάς ολίσθησης σειριακής εισόδου και εξόδου 8 ψηφίων

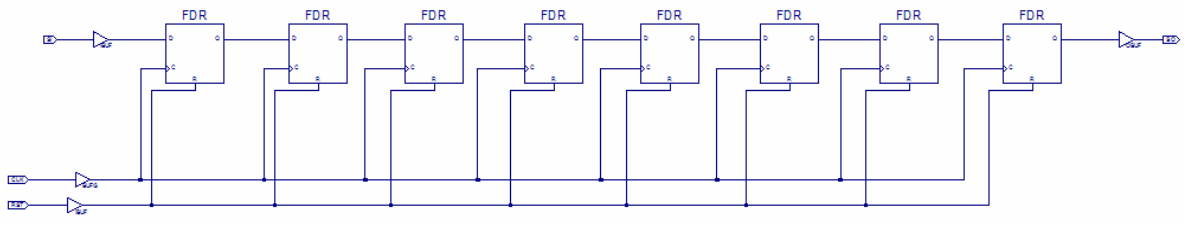
2.4.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα. Στον **πίνακα 2.29** φαίνεται το καινούργιο υλικό. Επιλέχτηκε για την σύγχρονη λειτουργία μηδενισμού που προσφέρει μέσω της εισόδου του R όταν ενεργοποιείται με λογικό 1. Η λειτουργία αυτή εξυπηρετεί την γενικότερη περίπτωση σύγχρονου μηδενισμού του καταχωρητή μέσω της εισόδου του RST.

Πίνακας 2.29 Καινούργια υλικά σχηματικού

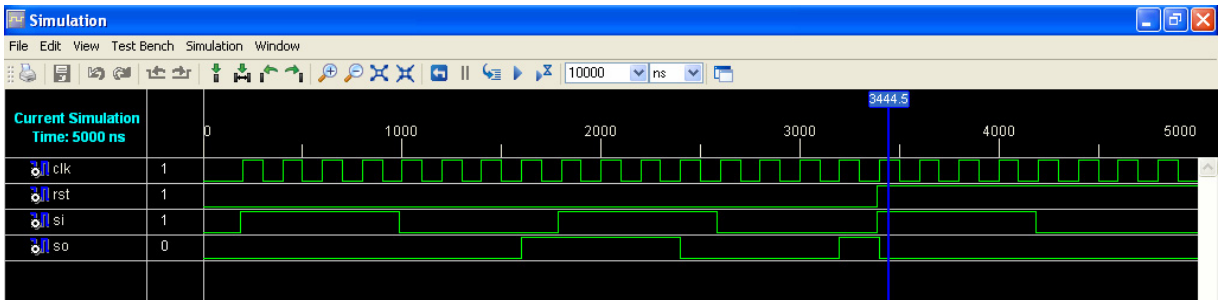
ΚΑΙΝΟΥΡΓΙΑ ΥΛΙΚΑ ΣΧΕΔΙΑΣΗΣ PROJECT NAVIGATOR			
ΣΧΗΜΑ	ΟΝΟΜΑΣΙΑ	ΙΔΙΟΤΗΤΑ	ΚΑΤΑΛΟΓΟΣ ΒΙΒΛΙΟΘΗΚΗΣ SYMBOL
	FDR	D FLIP-FLOP ΜΕ ΣΥΓΧΡΟΝΟ RESET (R ΕΙΣΟΔΟΣ)	FLIP_FLOP

Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.

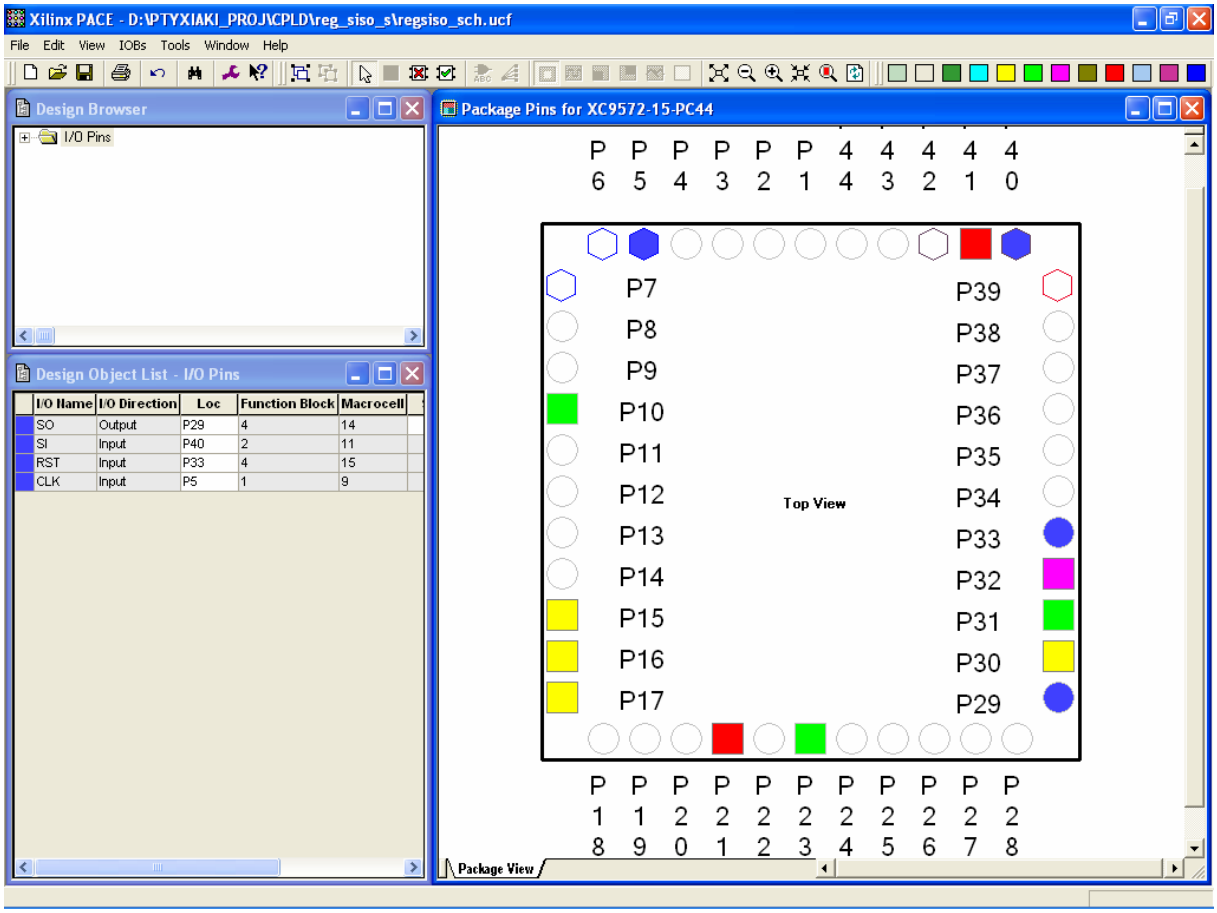


Εικόνα 2.39 Σχηματικό διάγραμμα κυκλώματος

Εικόνα 2.40 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.41 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.42 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.4.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 όπως αναφέρεται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 2.30 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

10 architecture BHV of regsiso_vhd is	Αρχή δήλωσης αρχιτεκτονικής.
11 begin	Αρχή κώδικα αρχιτεκτονικής.
12 process (clk)	Αρχή της διεργασίας με λίστα ευαισθησίας την είσοδο **clk**.
13 variable tmp : STD_LOGIC_VECTOR(7 downto 0):= = "00000000";	Δήλωση της μεταβλητής **tmp** οκτώ δυαδικών ψηφίων τύπου **std_logic** με αρχική δυαδική τιμή προσομοίωσης "00000000".
14 begin	Αρχή ενεργοποίησης της διεργασίας.
15 if (clk'event and clk='1') then	Αρχή εκτέλεσης **if-then**.
16 if rst='1' then	Αρχή εκτέλεσης **if-then-else.**
17 tmp := (others => '0');	Μηδενισμός της προηγούμενης τιμής του **tmp**.
18 else	Διακλάδωση εντολής **if-then-else.**
19 for i in 0 to 6 loop	Αρχή εκτέλεσης βρόχου της εντολής **for-loop** με ακέραια μεταβλητής **i** μεταξύ τιμών από 0 ως και 6.
20 tmp(i) := tmp(i+1);	Μεταφορά τιμής στην μεταβλητή **tmp** που αντιστοιχεί στην δεξιά ολίσθηση κατά ένα δυαδικό ψηφίο.
21 end loop;	Τέλος εκτέλεσης βρόχου.
22 tmp(7) := si;	Μεταφορά τιμής στο έβδομο δυαδικό ψηφίο της μεταβλητής **tmp** που αντιστοιχεί στην καταχώρηση της

πληροφορίας της σειριακής εισόδου **si**.

23 end if; Τέλος εκτέλεσης **if-then-else**.

24 end if; Τέλος εκτέλεσης **if-then**.

25 so <= tmp(0); Μεταφορά της τιμής του δυαδικού ψηφίου 0 της μεταβλητής **tmp** στη σειριακή έξοδο.

26 end process; Τέλος εκτέλεσης process.

27 end BHV; Τέλος κώδικα αρχιτεκτονικής.

Σημειώσεις.

Γραμμή 14. Η ενεργοποίηση της διεργασίας δηλώνεται με τη λέξη begin. Η διεργασία ενεργοποιείται με την αλλαγή της λογικής κατάστασης της εισόδου **clk** της λίστας ευαισθησίας.

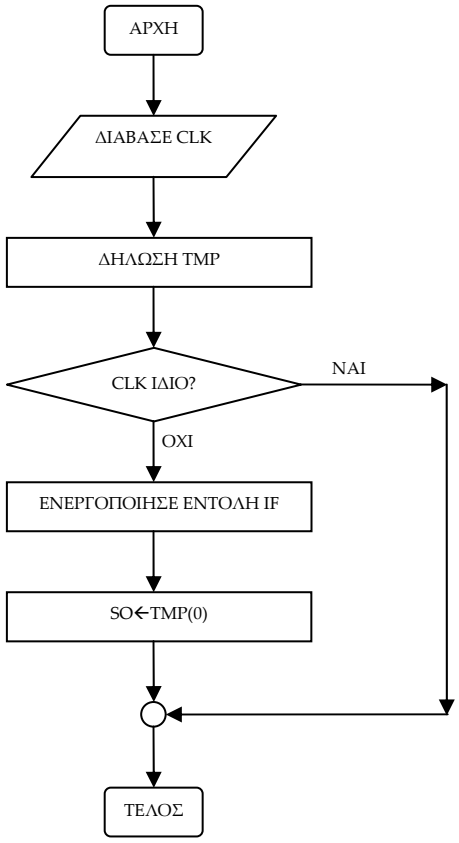
Γραμμή 15. Κατά την εκτέλεση της εντολής **if-then** έχουμε το λογικό έλεγχο για αλλαγή της εισόδου **clk** από λογικό 0 σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 16. Σε διαφορετική περίπτωση εκτελείται η γραμμή 25.

Γραμμή 16. Κατά την εκτέλεση της εντολής **if-then-else** έχουμε το λογικό έλεγχο για την κατάσταση της εισόδου **rst** σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 17. Σε διαφορετική περίπτωση (γραμμή 18) εκτελείται η γραμμή 19.

Γραμμή 19. Στην περίπτωση που η μεταβλητή **i** πάρει την τιμή 7 τότε εκτελείται η εντολή της γραμμής 22 λύνοντας το βρόχο εκτέλεσης της εντολής **for loop**.

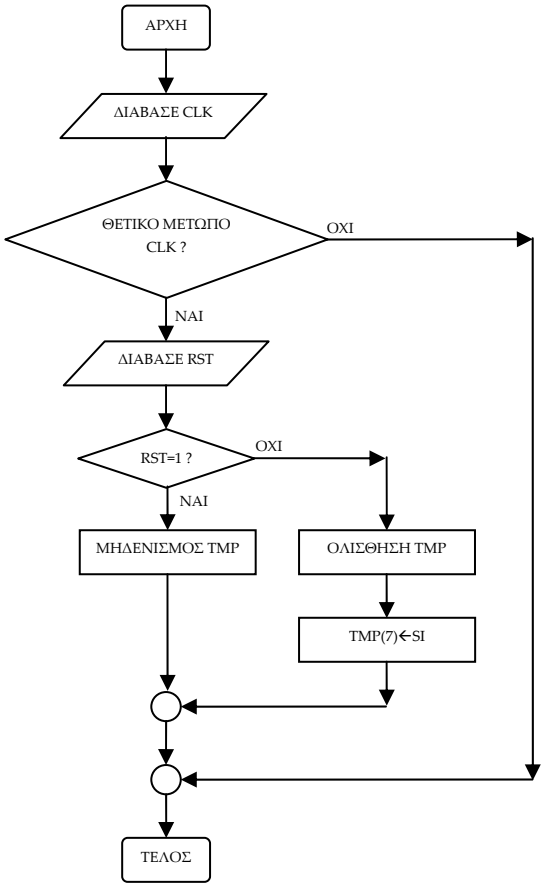
Πίνακας 2.31 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (clk)

variable tmp : STD_LOGIC_VECTOR(7 downto
0):= "00000000";

begin


  if … (if …else(for…end loop)…end if )…end if;

so <= tmp(0);

end process;
```



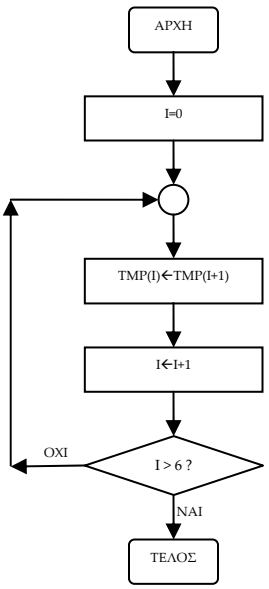
Πίνακας 2.32 Η εντολή **if-then** και το **διάγραμμα ροής**

```
if clk='1' and clk'event then

   if rst='1' then

      tmp := (others => '0'); else for…end loop;

      tmp(7) := si;
   end if;

end if;
```



Πίνακας 2.33 Η ολίσθηση με εντολή **for loop** και το **διάγραμμα ροής**

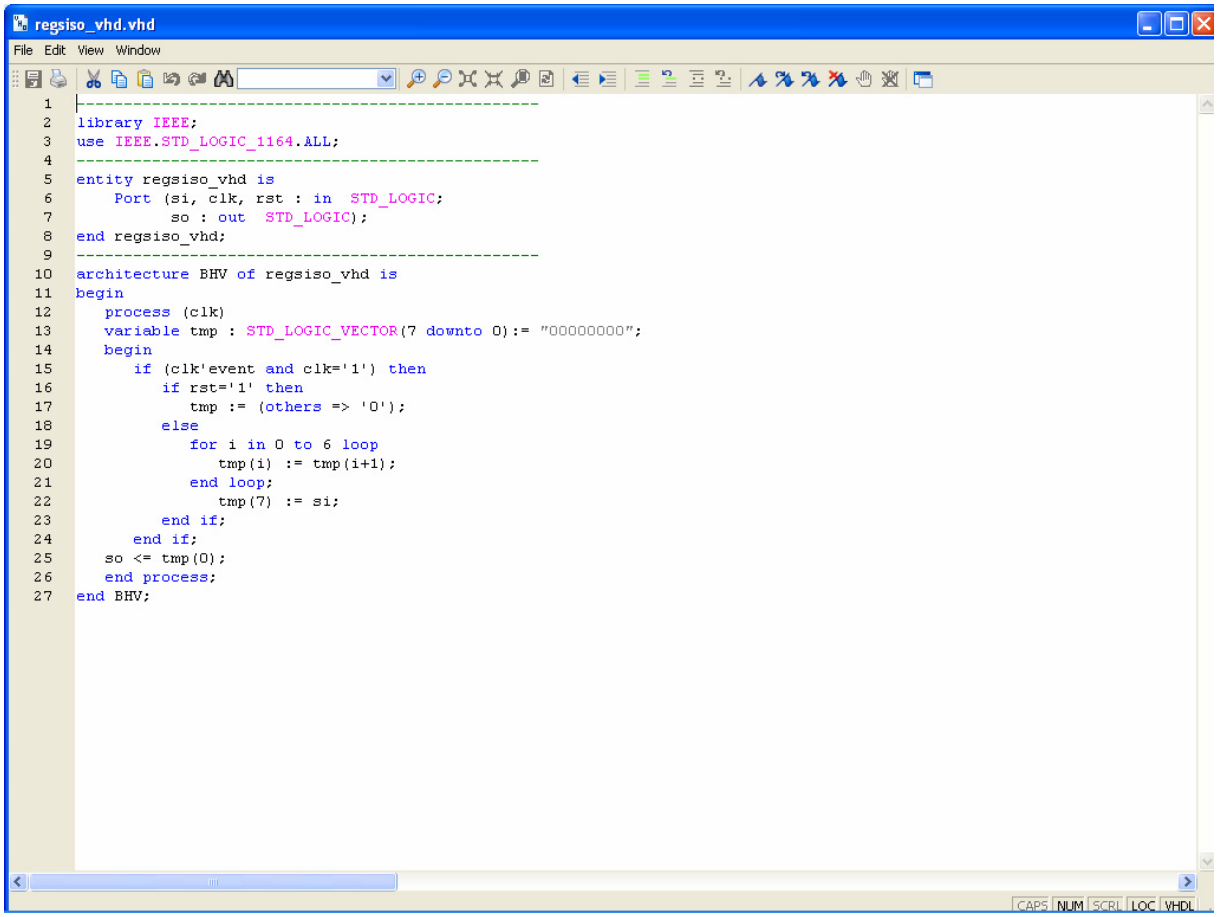
```
for i in 0 to 6 loop

tmp(i) := tmp(i+1);

end loop;
```



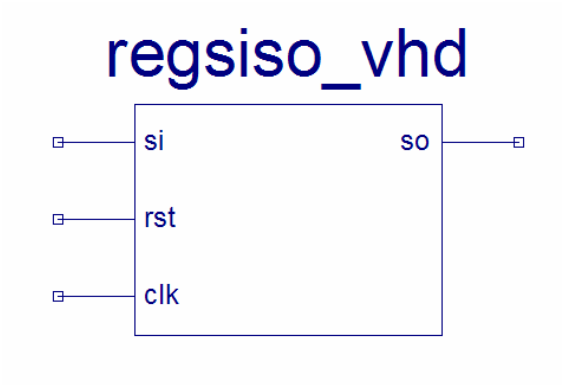
Σημείωση. Στα παραπάνω διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

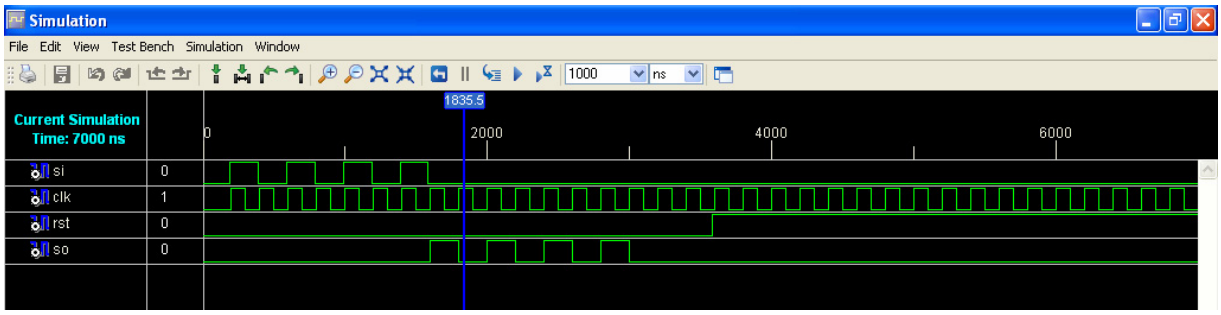
2.4.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)



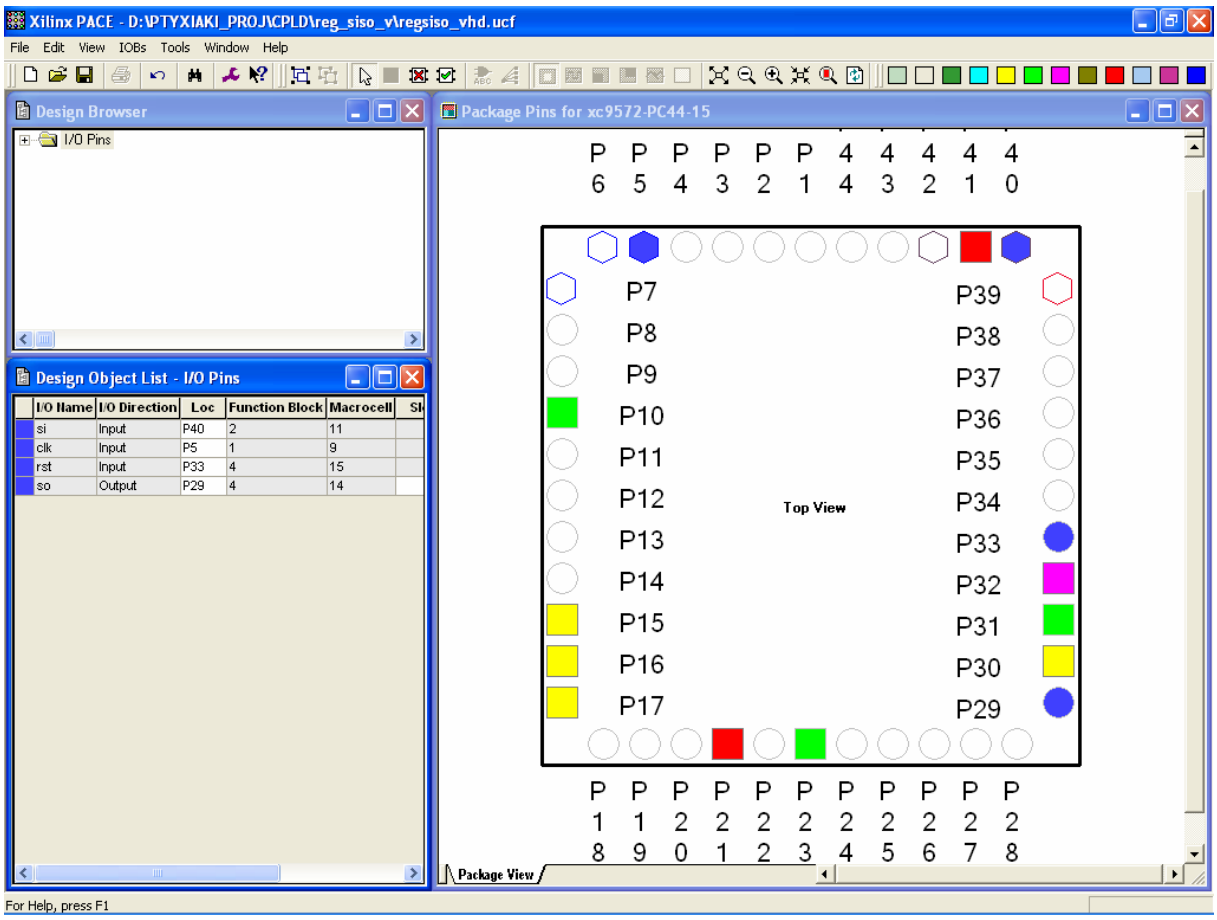
Εικόνα 2.43 VHDL αρχείο του κυκλώματος



Εικόνα 2.44 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.45 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.46 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

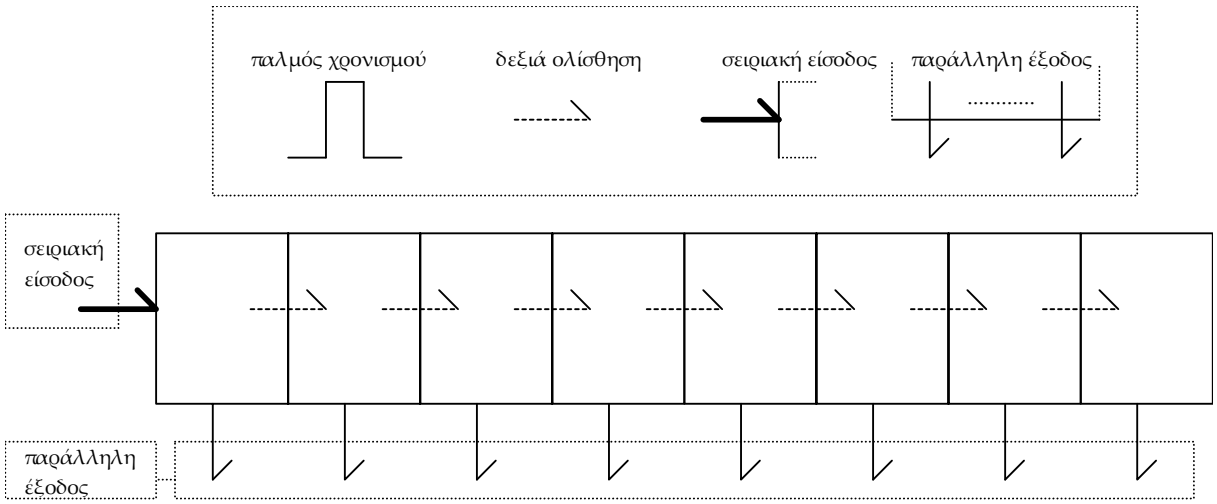
2.5 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ ΚΑΤΑΧΩΡΗΤΗ_2: ΔΕΞΙΑΣ ΟΛΙΣΘΗΣΗΣ ΣΕΙΡΙΑΚΗΣ ΕΙΣΟΔΟΥ-ΠΑΡΑΛΛΗΛΗΣ ΕΞΟΔΟΥ ΧΩΡΗΤΙΚΟΤΗΤΑΣ 8 ΨΗΦΙΩΝ

2.5.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

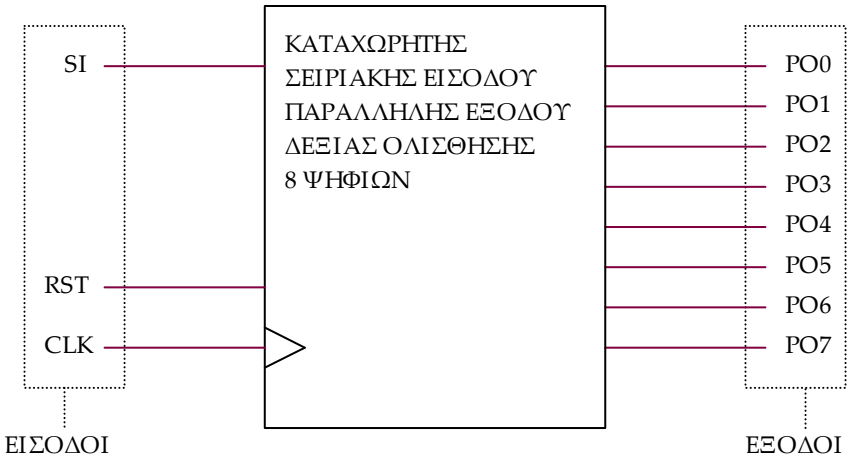
2.5.1.1 Σύντομη θεωρία

Τα βασικά της θεωρίας των καταχωρητών ολίσθησης περιλαμβάνονται στην αντίστοιχη σύντομη θεωρία της συνολικής εργασίας του καταχωρητή δεξιάς ολίσθησης σειριακής εισόδου και εξόδου οκτώ ψηφίων του παρόντος συγγράμματος.

Ο καταχωρητής δεξιάς ολίσθησης σειριακής εισόδου και παράλληλης εξόδου χωρητικότητας οκτώ ψηφίων έχει σαν εισόδους του το δυαδικό ψηφίο χρονισμού CLK, το δυαδικό ψηφίο σύγχρονου μηδενισμού RST και το δυαδικό ψηφίο SI για τη σειριακή είσοδο της πληροφορίας. Η παράλληλη έξοδός του είναι τα δυαδικά ψηφία PO0, PO1,…, PO7 (MSB=PO7). Είναι κατασκευασμένος από οκτώ θετικής διέγερσης δισταθείς πολυδονητές (*flip-flop*) D τύπου σε σειρά συνδεδεμένους μεταξύ τους. Στην **εικόνα 2.47** τα ορθογώνια παριστούν τους οκτώ πολυδονητές που με τον παλμό χρονισμού ένα δυαδικό ψηφίο μεταβαίνει από τον ένα πολυδονητή στον άλλο. Με τον ίδιο παλμό χρονισμού μπορούν να εξέρχονται στην παράλληλη έξοδο και τα οκτώ δυαδικά ψηφία πληροφορίας. Στην **εικόνα 2.48** απεικονίζεται το γραφικό σύμβολο του παραπάνω καταχωρητή.



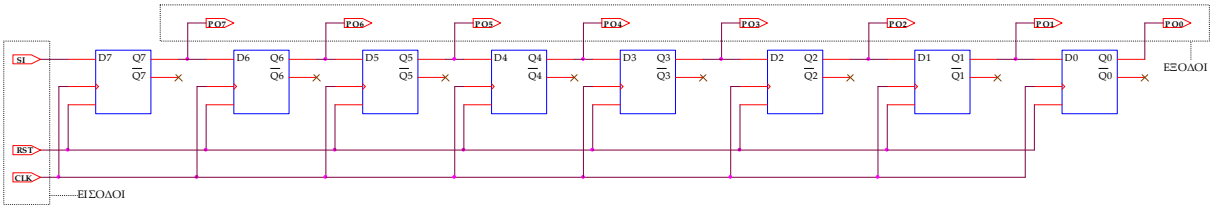
Εικόνα 2.47 Καταχωρητής δεξιάς ολίσθησης σειριακής εισόδου και παράλληλης εξόδου 8 ψηφίων



Εικόνα 2.48 Γραφικό σύμβολο καταχωρητή δεξιάς ολίσθησης σειριακής εισόδου και παράλληλης εξόδου 8 ψηφίων

2.5.1.2 Συμβατική ψηφιακή υλοποίηση

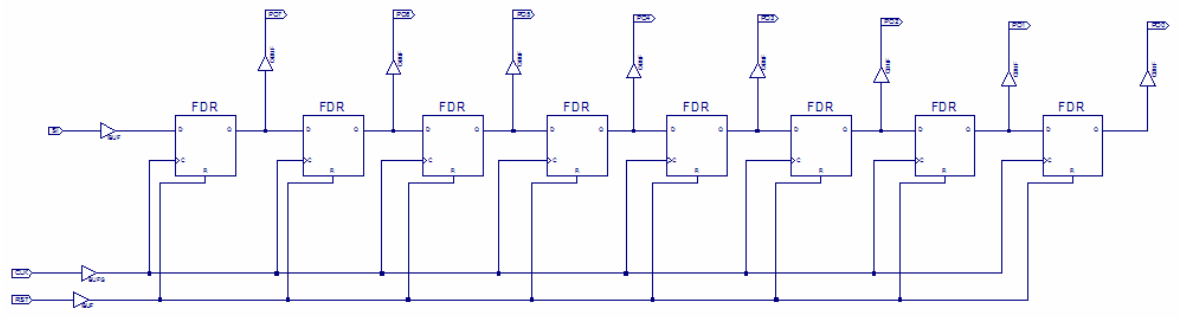
Η γενικευμένη συνδεσμολογία για τους καταχωρητές ολίσθησης παραπέμπει στην υλοποίηση με δισταθείς πολυδονητές D τύπου της παρακάτω **εικόνας 2.49**.



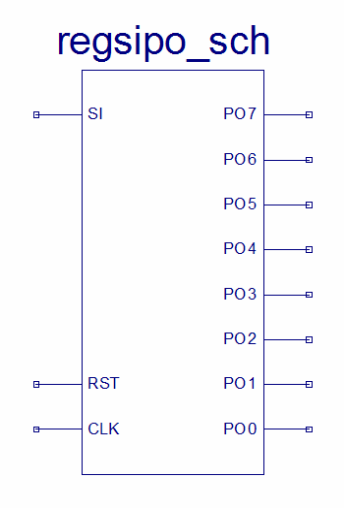
Εικόνα 2.49 Λογικό κύκλωμα υλοποίησης καταχωρητή δεξιάς ολίσθησης σειριακής εισόδου και παράλληλης εξόδου 8 ψηφίων

2.5.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

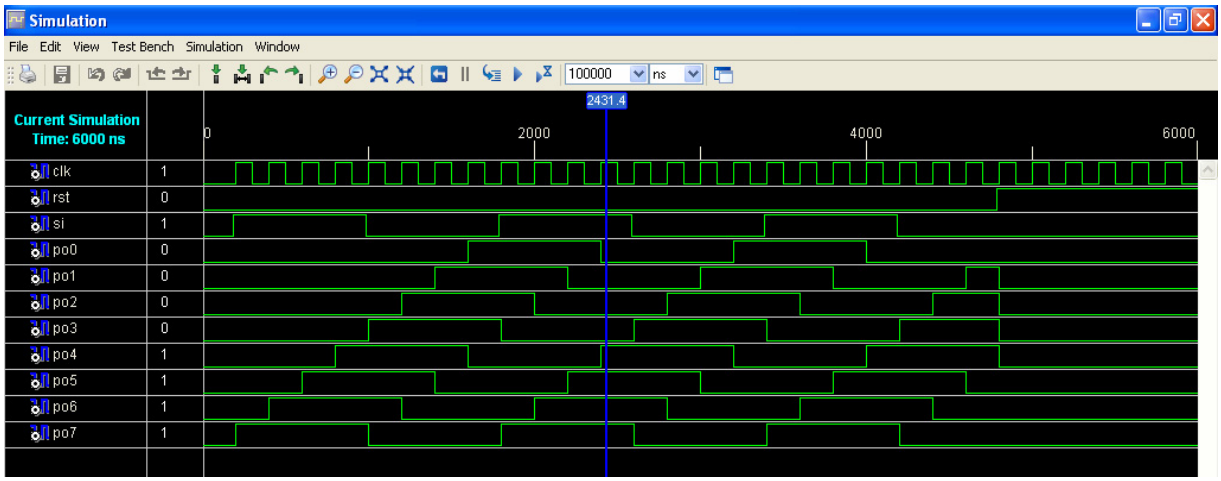
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 και του καταχωρητή δεξιάς ολίσθησης σειριακής εισόδου και εξόδου 8 ψηφίων στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.



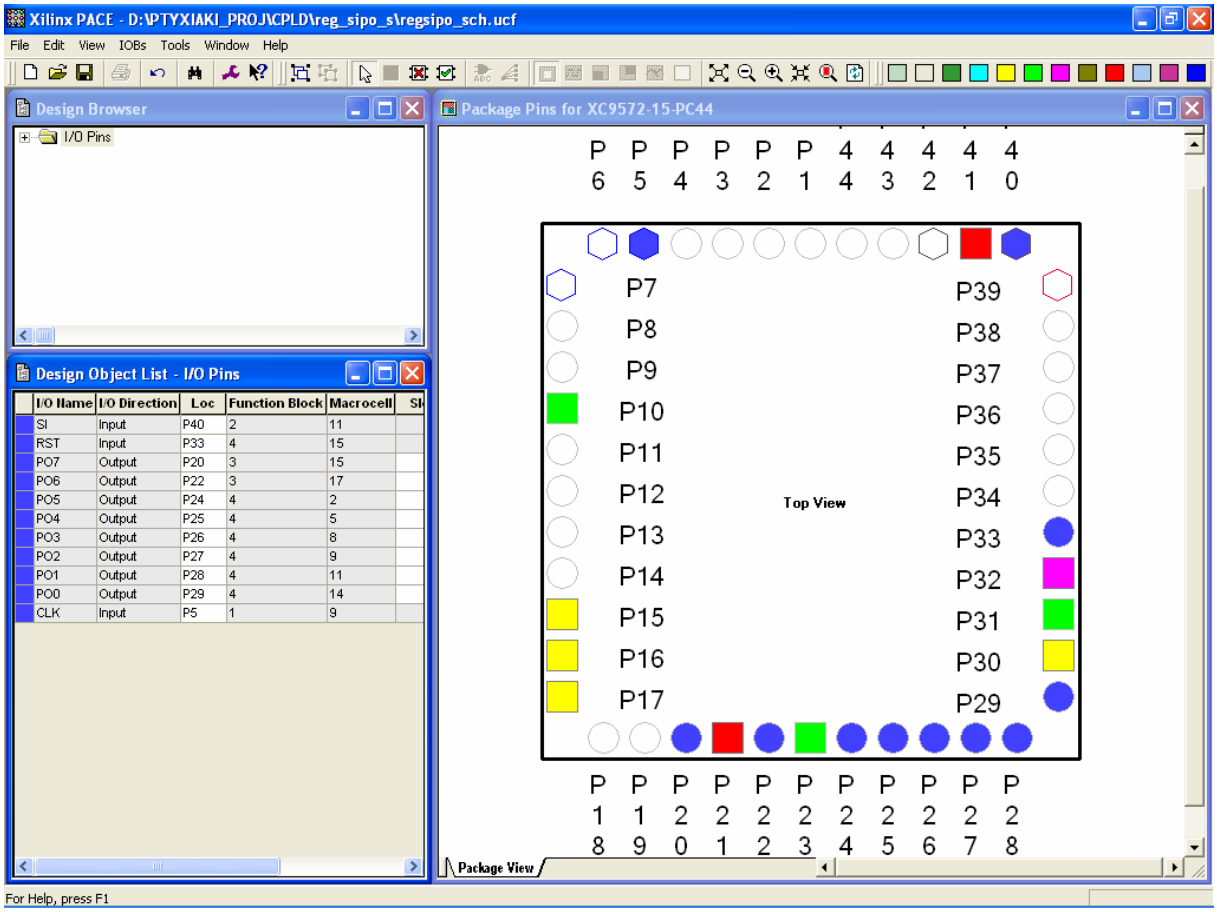
Εικόνα 2.50 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 2.51 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.52 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.53 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.5.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 όπως αναφέρεται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 2.34 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

10 architecture BHV of regsipo_vhd is	Αρχή δήλωσης αρχιτεκτονικής.
11 begin	Αρχή κώδικα αρχιτεκτονικής.
12 process (clk)	Αρχή της διεργασίας με λίστα ευαισθησίας την είσοδο **clk**.
13 variable tmp : STD_LOGIC_VECTOR(7 downto 0):= = "00000000";	Δήλωση της μεταβλητής **tmp** οκτώ δυαδικών ψηφίων τύπου **std_logic** με αρχική δυαδική τιμή προσομοίωσης "00000000".
14 begin	Αρχή ενεργοποίησης της διεργασίας.
15 if (clk'event and clk='1') then	Αρχή εκτέλεσης **if-then**.
16 if rst='1' then	Αρχή εκτέλεσης **if-then-else.**
17 tmp := (others => '0');	Μηδενισμός της προηγούμενης τιμής του **tmp**.
18 else	Διακλάδωση εντολής **if-then-else.**
19 for i in 0 to 6 loop	Αρχή εκτέλεσης βρόχου της εντολής **for-loop** με ακέραια μεταβλητής **i** μεταξύ τιμών από 0 ως και 6.
20 tmp(i) := tmp(i+1);	Μεταφορά τιμής στην μεταβλητή **tmp** που αντιστοιχεί στην δεξιά ολίσθηση κατά ένα δυαδικό ψηφίο.
21 end loop;	Τέλος εκτέλεσης βρόχου.
22 tmp(7) := si;	Μεταφορά τιμής στο έβδομο δυαδικό ψηφίο της μεταβλητής **tmp** που αντιστοιχεί στην καταχώρηση της

	πληροφορίας της σειριακής εισόδου **si**.
23 end if;	Τέλος εκτέλεσης **if-then-else**.
24 end if;	Τέλος εκτέλεσης **if-then**.
25 po <= tmp;	Μεταφορά της τιμής της μεταβλητής **tmp** στην παράλληλη έξοδο **po**.
26 end process;	Τέλος εκτέλεσης process.
27 end BHV;	Τέλος κώδικα αρχιτεκτονικής.

Σημειώσεις.

Γραμμή 14. Η ενεργοποίηση της διεργασίας δηλώνεται με τη λέξη begin. Η διεργασία ενεργοποιείται με την αλλαγή της λογικής κατάστασης της εισόδου **clk** της λίστας ευαισθησίας.

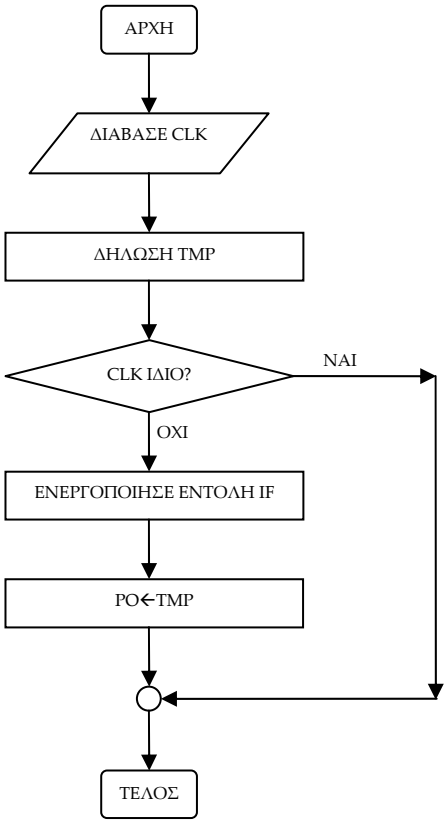
Γραμμή 15. Κατά την εκτέλεση της εντολής **if-then** έχουμε το λογικό έλεγχο για αλλαγή της εισόδου **clk** από λογικό 0 σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 16. Σε διαφορετική περίπτωση εκτελείται η γραμμή 25.

Γραμμή 16. Κατά την εκτέλεση της εντολής **if-then-else** έχουμε το λογικό έλεγχο για την κατάσταση της εισόδου **rst** σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 17. Σε διαφορετική περίπτωση (γραμμή 18) εκτελείται η γραμμή 19.

Γραμμή 19. Στην περίπτωση που η μεταβλητή **i** πάρει την τιμή 7 τότε εκτελείται η εντολή της γραμμής 22 λύνοντας το βρόχο εκτέλεσης της εντολής **for loop**.

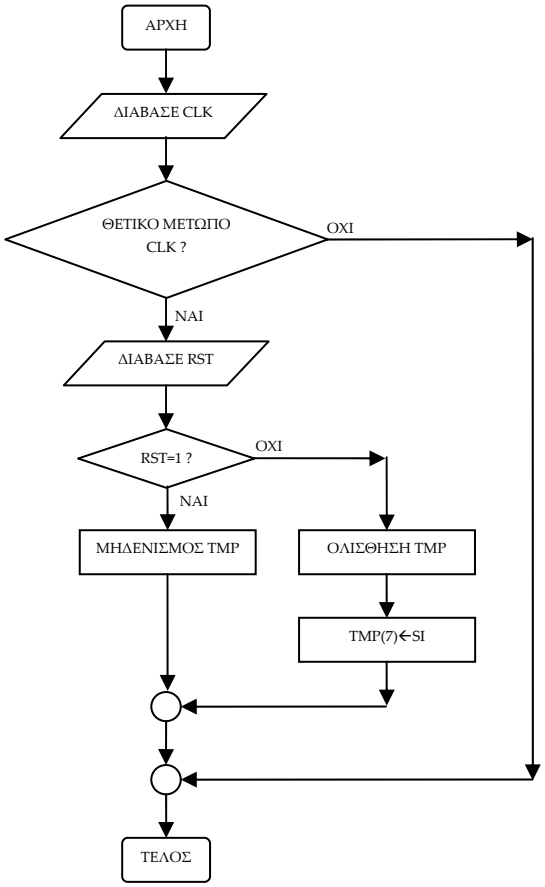
Πίνακας 2.35 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (clk)

variable tmp : STD_LOGIC_VECTOR(7 downto
0):= "00000000";

begin


  if … (if …else(for…end loop)…end if )…end if;

po <= tmp;

end process;
```



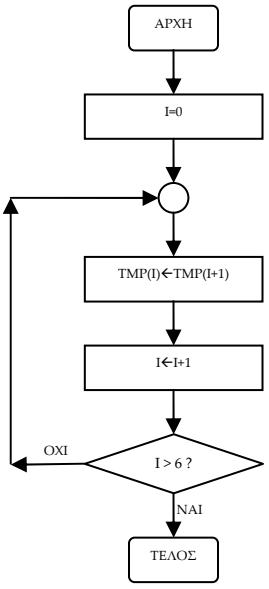
Πίνακας 2.36 Η εντολή **if-then** και το **διάγραμμα ροής**

```
if clk='1' and clk'event then

   if rst='1' then

      tmp := (others => '0'); else for…end loop;

      tmp(7) := si;
   end if;

end if;
```



Πίνακας 2.37 Η ολίσθηση με εντολή **for loop** και το **διάγραμμα ροής**

```
for i in 0 to 6 loop

tmp(i) := tmp(i+1);

end loop;
```



Σημείωση. Στα παραπάνω διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

2.5.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

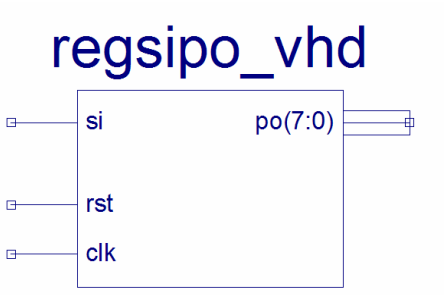
regsipo_vhd.vhd

File Edit View Window

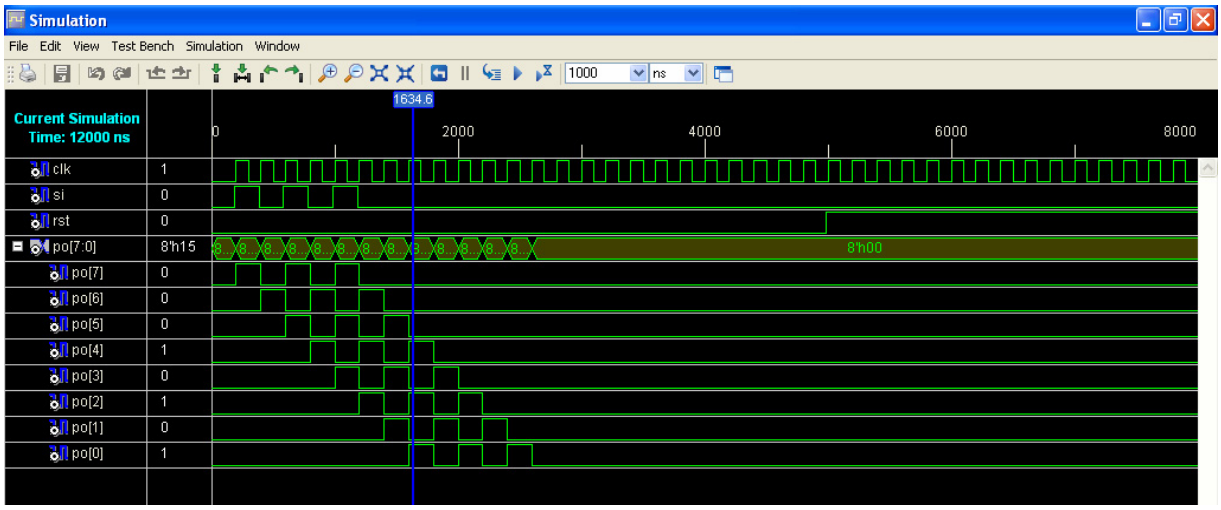
```
1  ----------------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  ----------------------------------------------------------
5  entity regsipo_vhd is
6      Port (clk, si, rst : in  STD_LOGIC;
7            po : out  STD_LOGIC_VECTOR (7 downto 0));
8  end regsipo_vhd;
9  ----------------------------------------------------------
10 architecture BHV of regsipo_vhd is
11 begin
12    process (clk)
13    variable tmp : STD_LOGIC_VECTOR(7 downto 0):= "00000000";
14    begin
15       if (clk'event and clk='1') then
16          if rst='1' then
17             tmp := (others => '0');
18          else
19             for i in 0 to 6 loop
20                tmp(i) := tmp(i+1);
21             end loop;
22             tmp(7) := si;
23          end if;
24       end if;
25    po <= tmp;
26    end process;
27 end BHV;
```

CAPS NUM SCRL LOC VHDL

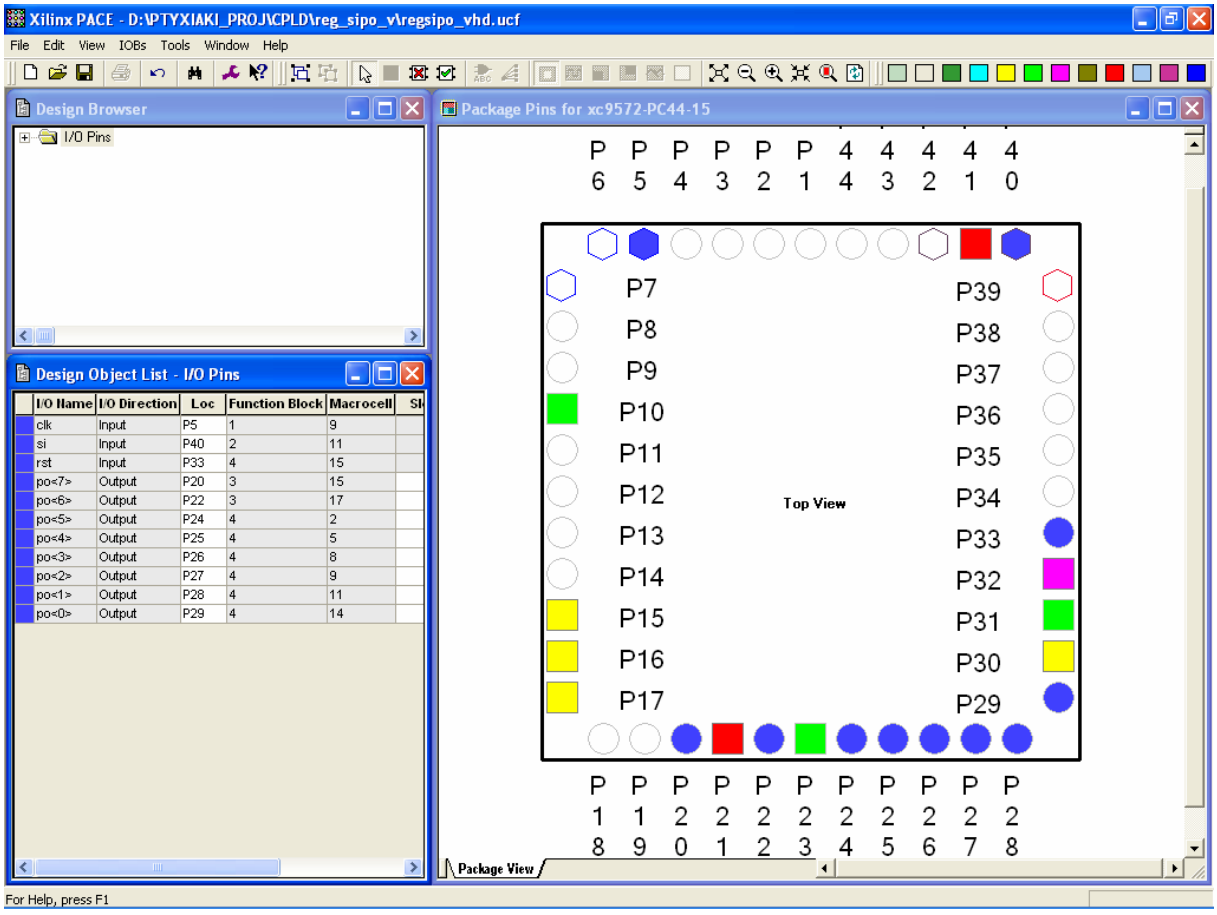
Εικόνα 2.54 VHDL αρχείο του κυκλώματος



Εικόνα 2.55 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.56 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.57 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

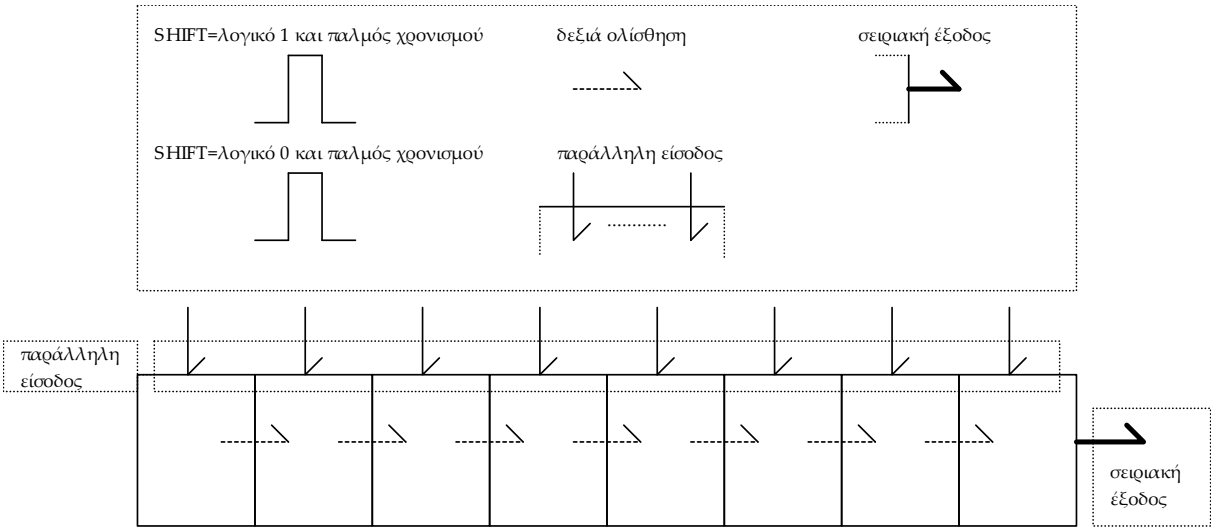
2.6 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ ΚΑΤΑΧΩΡΗΤΗ_3: ΔΕΞΙΑΣ ΟΛΙΣΘΗΣΗΣ ΠΑΡΑΛΛΗΛΗΣ ΕΙΣΟΔΟΥ-ΣΕΙΡΙΑΚΗΣ ΕΞΟΔΟΥ ΧΩΡΗΤΙΚΟΤΗΤΑΣ 8 ΨΗΦΙΩΝ

2.6.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

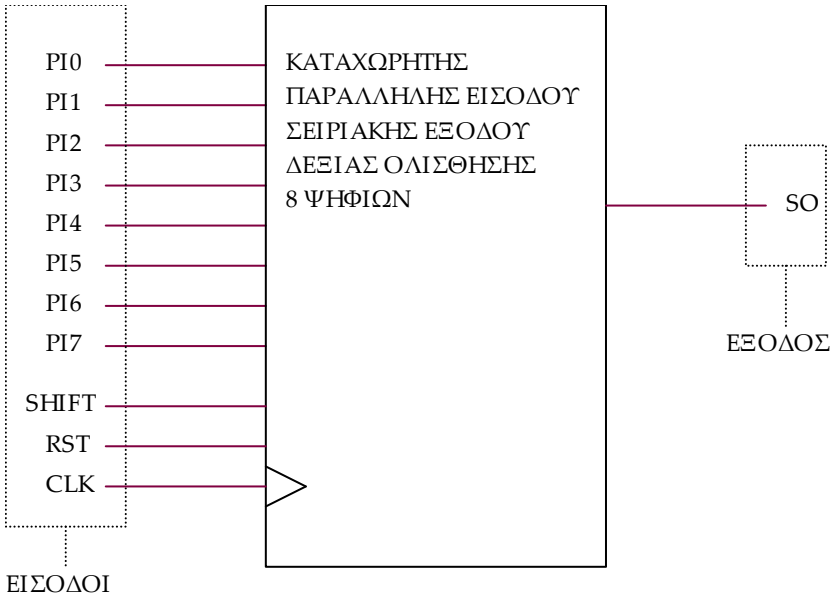
2.6.1.1 Σύντομη θεωρία

Τα βασικά της θεωρίας των καταχωρητών ολίσθησης περιλαμβάνονται στην αντίστοιχη σύντομη θεωρία της συνολικής εργασίας του καταχωρητή δεξιάς ολίσθησης σειριακής εισόδου και εξόδου οκτώ ψηφίων του παρόντος συγγράμματος.

Ο καταχωρητής δεξιάς ολίσθησης παράλληλης εισόδου και σειριακής εξόδου χωρητικότητας οκτώ ψηφίων έχει σαν εισόδους του το δυαδικό ψηφίο χρονισμού CLK, το δυαδικό ψηφίο σύγχρονου μηδενισμού RST, το δυαδικό ψηφίο ολίσθησης SHIFT και τα δυαδικά ψηφία PI0, PI1,…, PI7 (MSB=PI7) για την παράλληλη είσοδο της πληροφορίας. Η σειριακή έξοδός του είναι το δυαδικό ψηφίο SO. Είναι κατασκευασμένος από οκτώ θετικής διέγερσης δισταθείς πολυδονητές (*flip-flop*) D τύπου σε σειρά συνδεδεμένους μεταξύ τους. Στην **εικόνα 2.58** τα ορθογώνια παριστούν τους οκτώ πολυδονητές. Η αποκλειστική εισαγωγή των οκτώ δυαδικών ψηφίων πληροφορίας πραγματοποιείται με τον παλμό χρονισμού και την είσοδο SHIFT ενεργοποιημένη με λογικό 0. Η αποκλειστική εισαγωγή πραγματοποιείται σε αντιστοιχία ενός δυαδικού ψηφίου ανά πολυδονητή. Η αποκλειστική δεξιά ολίσθηση δυαδικού ψηφίου μεταξύ των πολυδονητών αυτών πραγματοποιείται με τον παλμό χρονισμού και την είσοδο SHIFT ενεργοποιημένη με λογικό 1. Στην **εικόνα 2.59** απεικονίζεται το γραφικό σύμβολο του παραπάνω καταχωρητή.



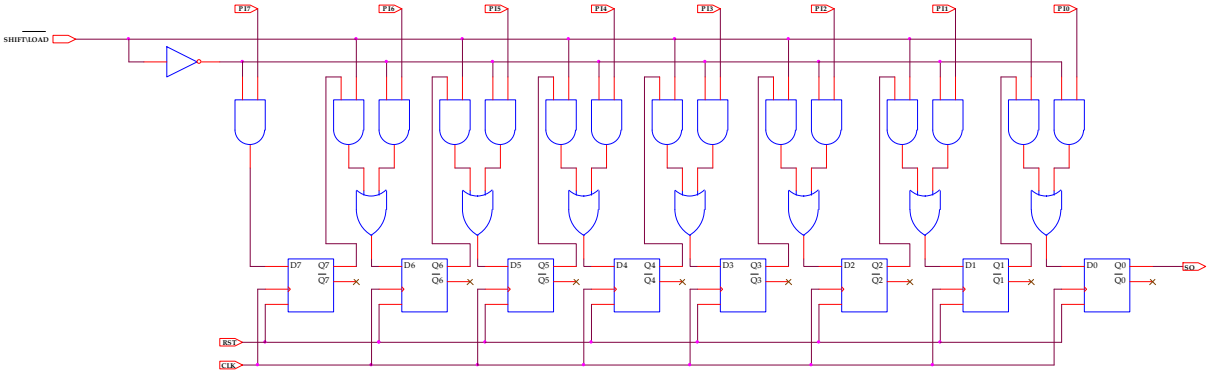
Εικόνα 2.58 Καταχωρητής δεξιάς ολίσθησης παράλληλης εισόδου και σειριακής εξόδου 8 ψηφίων



Εικόνα 2.59 Γραφικό σύμβολο καταχωρητή δεξιάς ολίσθησης παράλληλης εισόδου και σειριακής εξόδου 8 ψηφίων

2.6.1.2 Συμβατική ψηφιακή υλοποίηση

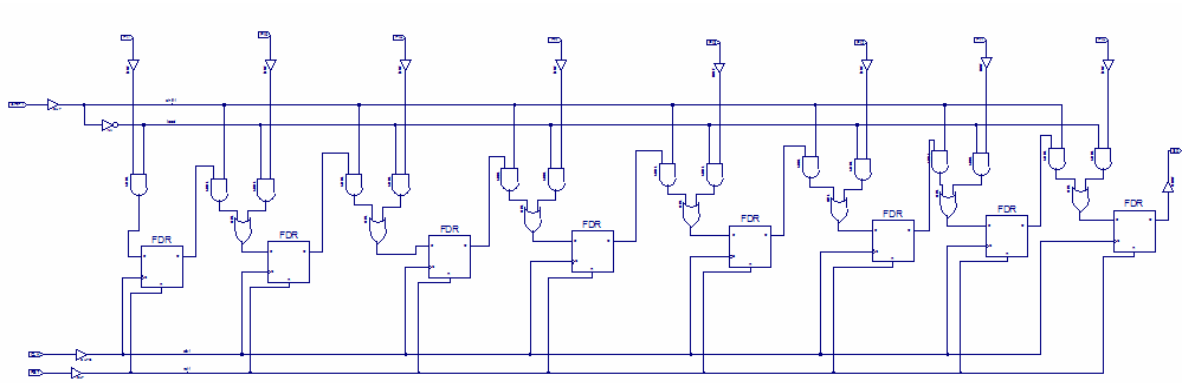
Η γενικευμένη συνδεσμολογία για τους καταχωρητές ολίσθησης παραπέμπει στην υλοποίηση με δισταθείς πολυδονητές D τύπου της παρακάτω **εικόνας 2.60**.



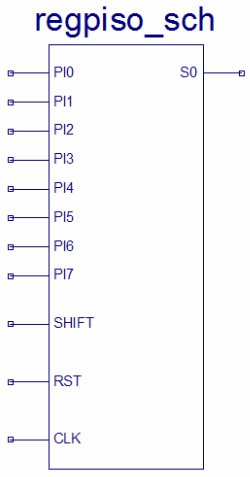
Εικόνα 2.60 Λογικό κύκλωμα υλοποίησης καταχωρητή δεξιάς ολίσθησης παράλληλης εισόδου και σειριακής εξόδου 8 ψηφίων

2.6.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

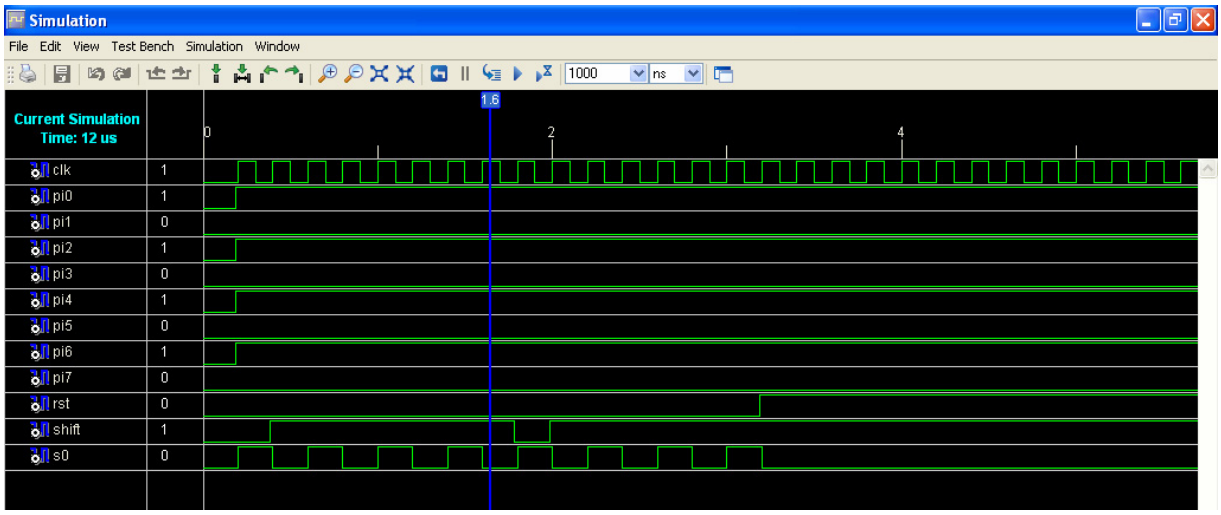
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 και του καταχωρητή δεξιάς ολίσθησης σειριακής εισόδου και εξόδου 8 ψηφίων στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.



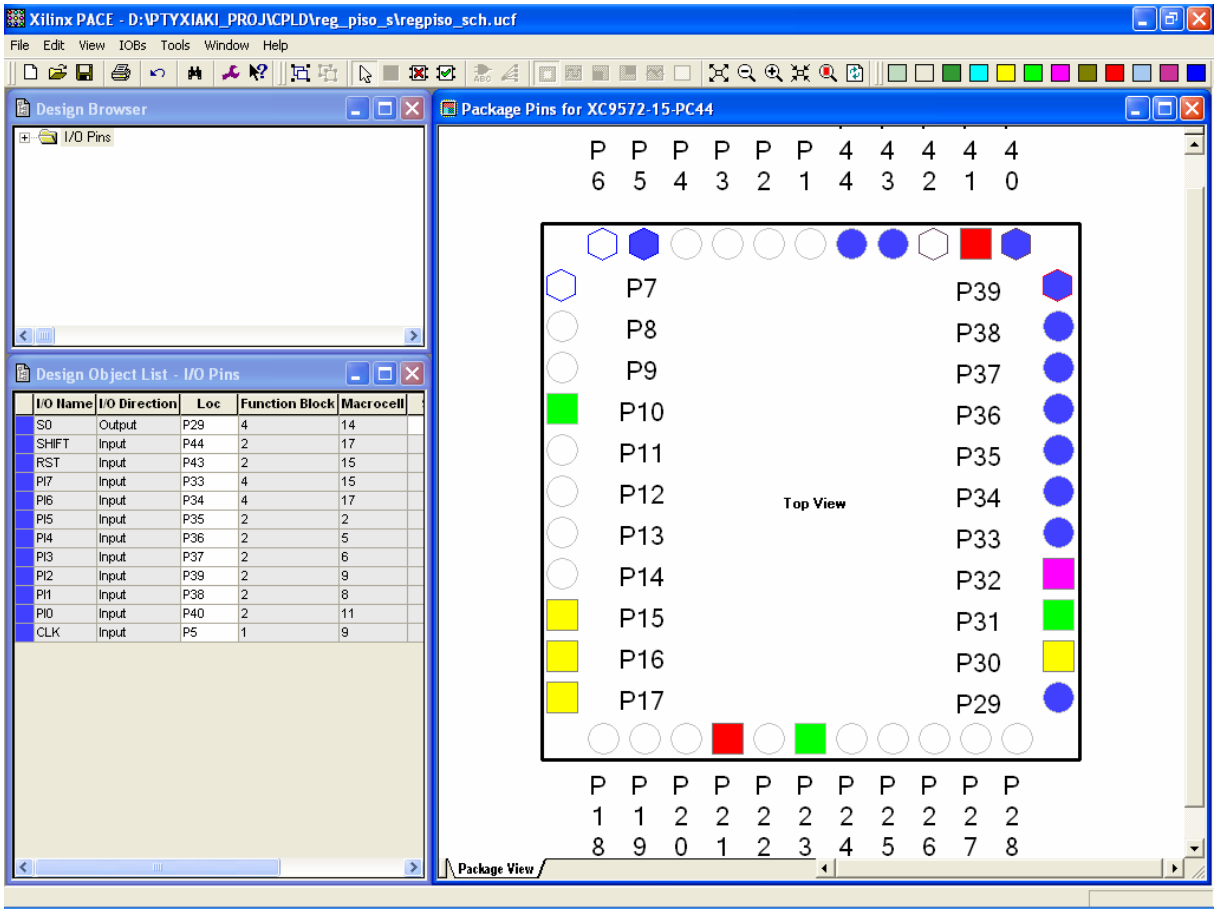
Εικόνα 2.61 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 2.62 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.63 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.64 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.6.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 όπως αναφέρεται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

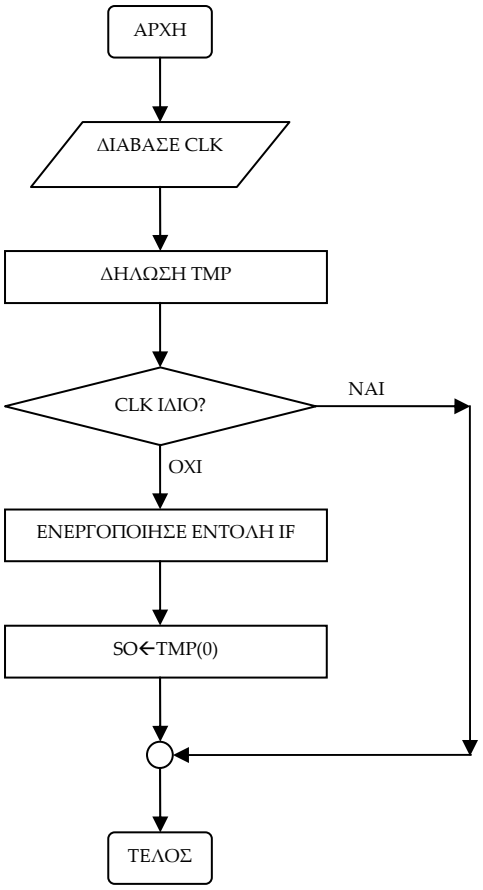
Πίνακας 2.38 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

11 architecture BHV of regpiso_vhd is	Αρχή δήλωσης αρχιτεκτονικής.
12 begin	Αρχή κώδικα αρχιτεκτονικής.
13 process (clk)	Αρχή της διεργασίας με λίστα ευαισθησίας την είσοδο **clk**.
14 variable tmp : STD_LOGIC_VECTOR (7 downto 0):= "00000000";	Δήλωση της μεταβλητής **tmp** οκτώ δυαδικών ψηφίων τύπου **std_logic** με αρχική δυαδική τιμή προσομοίωσης "00000000".
15 begin	Αρχή ενεργοποίησης της διεργασίας.
16 if (clk'event and clk='1') then	Αρχή εκτέλεσης **if-then**.
17 if rst='1' then	Αρχή εκτέλεσης **if-then-else.**
18 tmp := (others => '0');	Μηδενισμός της προηγούμενης τιμής του **temp**.
19 else	Διακλάδωση εντολής **if-then-else.**
20 if shift='1' then	Αρχή εκτέλεσης **if-then**.
21 for i in 0 to 6 loop	Αρχή εκτέλεσης βρόχου της εντολής **for-loop** με ακέραια μεταβλητής **i** μεταξύ τιμών από 0 ως και 6.
22 tmp(i) := tmp(i+1);	Μεταφορά τιμής στην μεταβλητή **tmp** που αντιστοιχεί στην δεξιά ολίσθηση κατά ένα δυαδικό ψηφίο.
23 end loop;	Τέλος εκτέλεσης βρόχου.
24 tmp(7) := '0';	Μεταφορά τιμής στο έβδομο δυαδικό ψηφίο της μεταβλητής **tmp**.
25 else	Διακλάδωση εντολής **if-then-else.**
26 tmp:=pi;	Μεταφορά της τιμής της παράλληλης εισόδου **pi** στη μεταβλητή **tmp**.
27 end if;	Τέλος εκτέλεσης **if-then-else**.
28 end if;	Τέλος εκτέλεσης **if-then-else**.
29 end if;	Τέλος εκτέλεσης **if-then**.
30 so <= tmp(0);	Μεταφορά της τιμής του δυαδικού ψηφίου 0 της μεταβλητής **tmp** στη σειριακή έξοδο.
31 end process;	Τέλος εκτέλεσης process.
32 end BHV;	Τέλος κώδικα αρχιτεκτονικής.

Σημειώσεις. Γραμμή 15. Η ενεργοποίηση της διεργασίας δηλώνεται με τη λέξη begin. Η διεργασία ενεργοποιείται με την αλλαγή της λογικής κατάστασης της εισόδου **clk** της λίστας ευαισθησίας.
Γραμμή 16. Κατά την εκτέλεση της εντολής **if-then** έχουμε το λογικό έλεγχο για αλλαγή της εισόδου **clk** από λογικό 0 σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 17. Σε διαφορετική περίπτωση εκτελείται η γραμμή 30.
Γραμμή 17. Κατά την εκτέλεση της εντολής **if-then-else** έχουμε το λογικό έλεγχο για την κατάσταση της εισόδου **rst** σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 18. Σε διαφορετική περίπτωση (γραμμή 19) εκτελείται η γραμμή 20.
Γραμμή 20. Κατά την εκτέλεση της εντολής **if-then-else** έχουμε το λογικό έλεγχο για την κατάσταση της εισόδου **shift** σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 21. Σε διαφορετική περίπτωση (γραμμή 25) εκτελείται η γραμμή 26.
Γραμμή 21. Στην περίπτωση που η μεταβλητή **i** πάρει την τιμή 7 τότε εκτελείται η εντολή της γραμμής 24 λύνοντας το βρόχο εκτέλεσης της εντολής **for loop**.
Γραμμή 24. Η μεταφορά τιμής εξυπηρετεί μη αλλοίωση πληροφορίας κατά την δεξιά ολίσθηση.

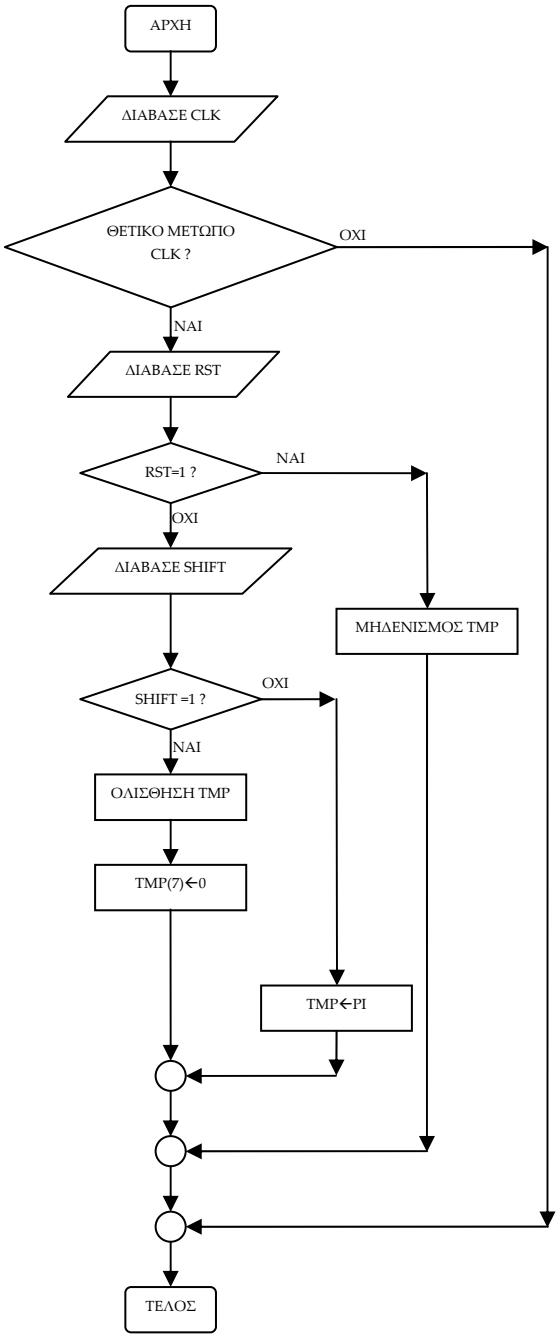
Πίνακας 2.39 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (clk)

variable tmp : STD_LOGIC_VECTOR(7 downto
0):= "00000000";

begin

  if … (if …else (if …else(for…end loop)…end
if)…end if )…end if;

so <= tmp(0);

end process;
```



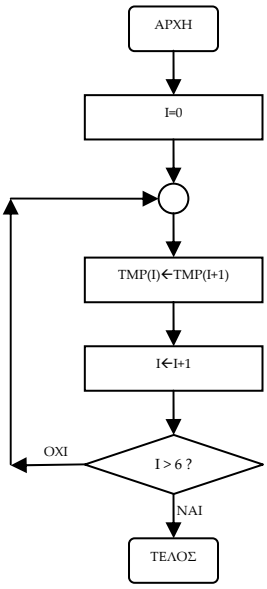
Πίνακας 2.40 Η εντολή **if-then** και το **διάγραμμα ροής**

```
if (clk'event and clk='1') then

 if rst='1' then

   tmp := (others => '0');
 else if shift='1' then
    for i in… 0 to 6 … end loop;
   tmp(7) := '0';
   else  tmp:=pi;
   end if;
 end if;
end if;
```



Πίνακας 2.41 Η ολίσθηση με εντολή **for loop** και το **διάγραμμα ροής**

```
for i in 0 to 6 loop

tmp(i) := tmp(i+1);

end loop;
```



Σημείωση. Στα παραπάνω διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

2.6.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

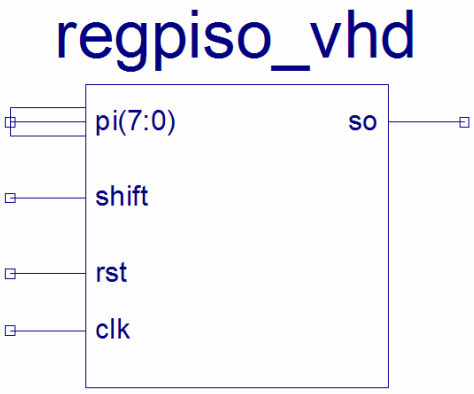
regpiso_vhd.vhd

File Edit View Window

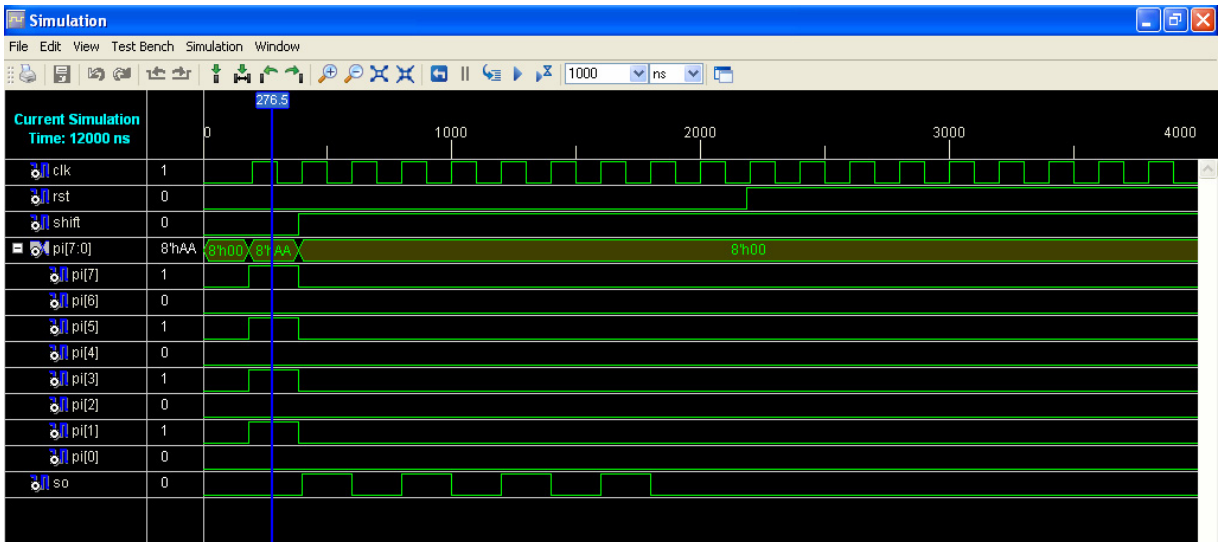
```
1  ----------------------------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  ----------------------------------------------------------------------
5  entity regpiso_vhd is
6      Port ( clk, rst, shift : in  STD_LOGIC;
7             pi : in  STD_LOGIC_VECTOR (7 downto 0);
8             so : out  STD_LOGIC);
9  end regpiso_vhd;
10 ----------------------------------------------------------------------
11 architecture BHV of regpiso_vhd is
12 begin
13    process (clk)
14    variable tmp : STD_LOGIC_VECTOR(7 downto 0):= "00000000";
15    begin
16       if (clk'event and clk='1') then
17          if rst='1' then
18             tmp := (others => '0');
19          else
20             if shift='1' then
21                for i in 0 to 6 loop
22                tmp(i) := tmp(i+1);
23                end loop;
24             tmp(7) := '0';
25             else
26                tmp:=pi;
27             end if;
28          end if;
29       end if;
30    so <= tmp(0);
31    end process;
32 end BHV;
```

CAPS NUM SCRL LOC VHDL

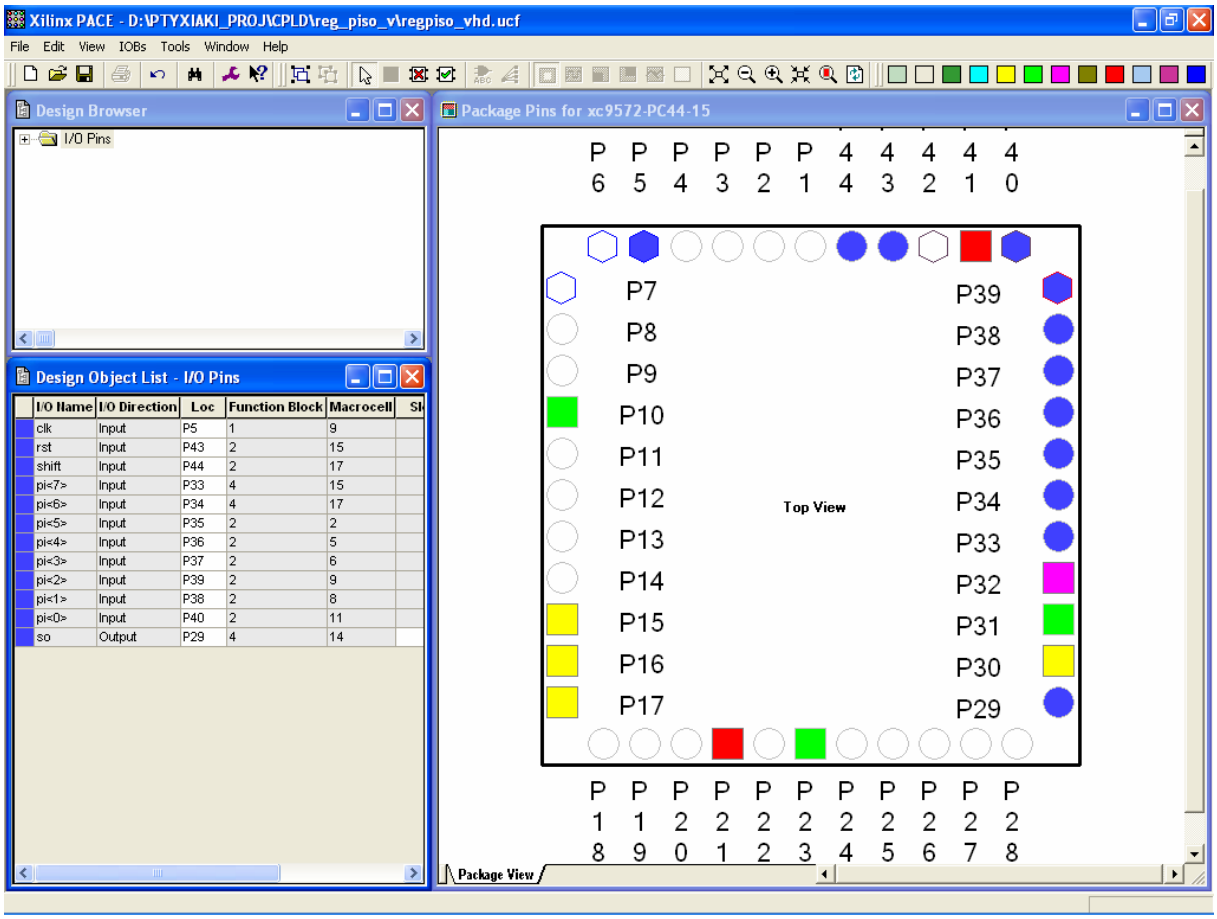
Εικόνα 2.65 VHDL αρχείο του κυκλώματος



Εικόνα 2.66 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.67 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.68 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

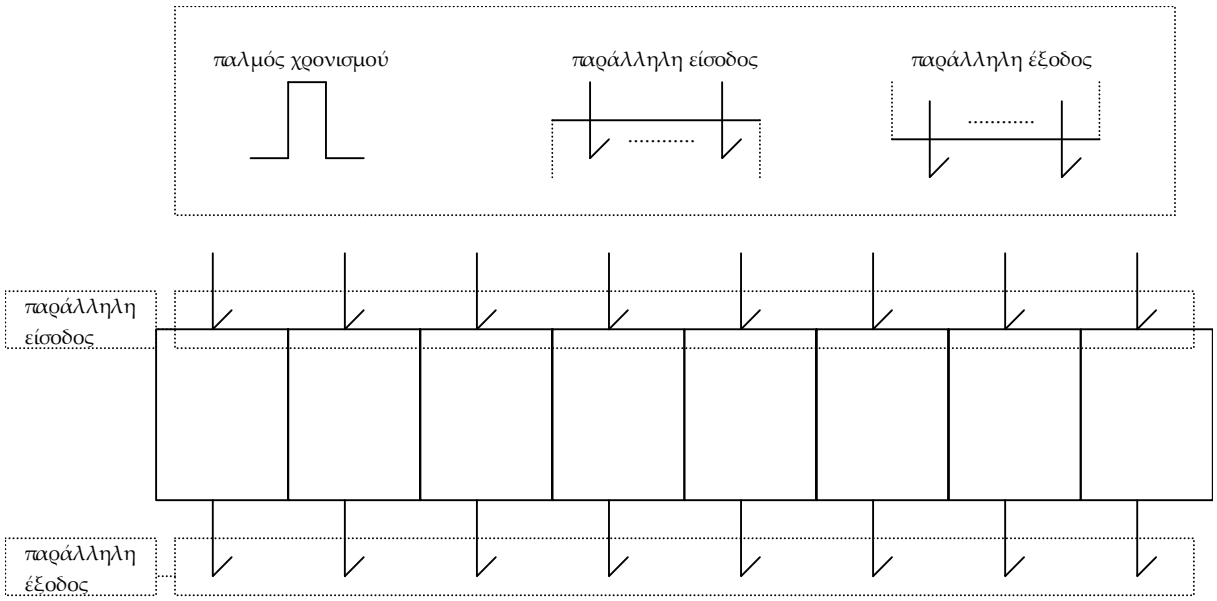
2.7 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ ΚΑΤΑΧΩΡΗΤΗ_4: ΔΕΞΙΑΣ ΟΛΙΣΘΗΣΗΣ ΠΑΡΑΛΛΗΛΗΣ ΕΙΣΟΔΟΥ-ΠΑΡΑΛΛΗΛΗΣ ΕΞΟΔΟΥ ΧΩΡΗΤΙΚΟΤΗΤΑΣ 8 ΨΗΦΙΩΝ

2.7.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

2.7.1.1 Σύντομη θεωρία

Τα βασικά της θεωρίας των καταχωρητών ολίσθησης περιλαμβάνονται στην αντίστοιχη σύντομη θεωρία της συνολικής εργασίας του καταχωρητή δεξιάς ολίσθησης σειριακής εισόδου και εξόδου οκτώ ψηφίων του παρόντος συγγράμματος.

Ο καταχωρητής δεξιάς ολίσθησης παράλληλης εισόδου και εξόδου χωρητικότητας οκτώ ψηφίων έχει σαν εισόδους του το δυαδικό ψηφίο χρονισμού CLK, το δυαδικό ψηφίο σύγχρονου μηδενισμού RST και τα δυαδικά ψηφία PI0, PI1,…, PI7 (MSB=PI7) για την παράλληλη είσοδο της πληροφορίας. Η παράλληλη έξοδός του είναι τα δυαδικά ψηφία PO0, PO1,…, PO7 (MSB=PO7). Είναι κατασκευασμένος από οκτώ θετικής διέγερσης δισταθείς πολυδονητές (*flip-flop*) D τύπου σε σειρά συνδεδεμένους μεταξύ τους. Στην **εικόνα 2.69** τα ορθογώνια παριστούν τους οκτώ πολυδονητές. Τα οκτώ δυαδικά ψηφία πληροφορίας με τον παλμό χρονισμού εισέρχονται στην παράλληλη είσοδο και εξέρχονται στην παράλληλη έξοδο. Στην παρούσα λειτουργία δεν έχουμε ολίσθηση της πληροφορίας εντός του καταχωρητή. Στην **εικόνα 2.70** απεικονίζεται το γραφικό σύμβολο του παραπάνω καταχωρητή.

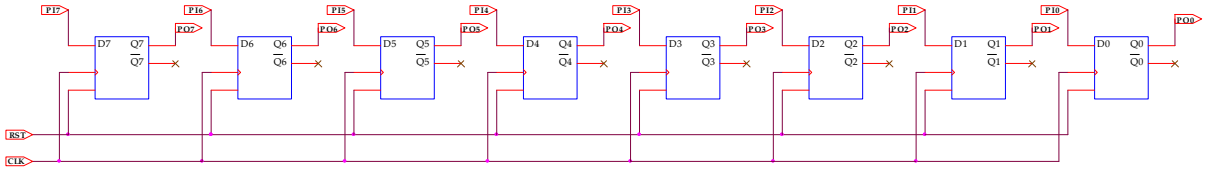


Εικόνα 2.69 Καταχωρητής δεξιάς ολίσθησης παράλληλης εισόδου και εξόδου 8 ψηφίων

Εικόνα 2.70 Γραφικό σύμβολο καταχωρητή δεξιάς ολίσθησης παράλληλης εισόδου και εξόδου 8 ψηφίων

2.7.1.2 Συμβατική ψηφιακή υλοποίηση

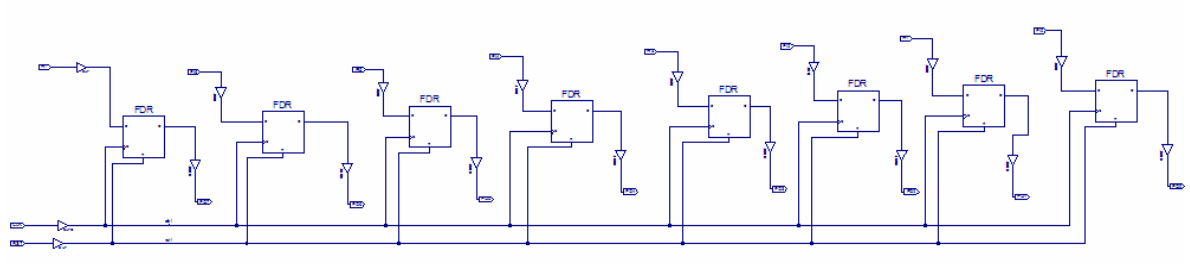
Η γενικευμένη συνδεσμολογία για τους καταχωρητές ολίσθησης παραπέμπει στην υλοποίηση με δισταθείς πολυδονητές D τύπου της παρακάτω **εικόνας 2.71**.



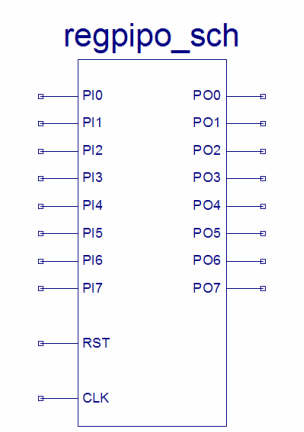
Εικόνα 2.71 Λογικό κύκλωμα υλοποίησης καταχωρητή δεξιάς ολίσθησης παράλληλης εισόδου και εξόδου 8 ψηφίων

2.7.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

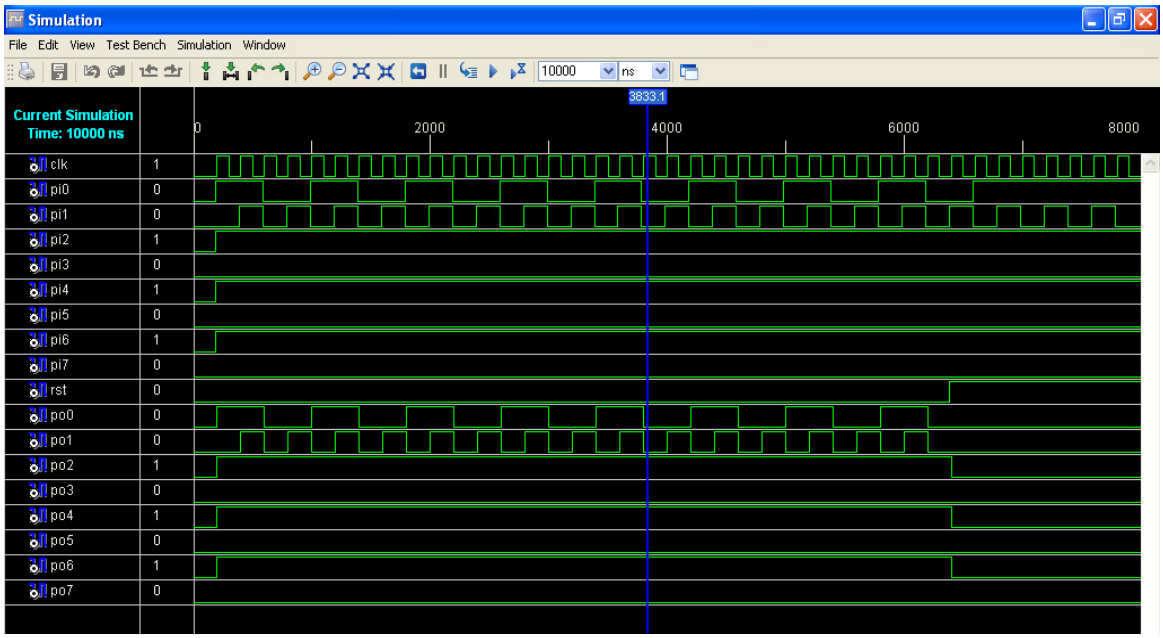
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 και του καταχωρητή δεξιάς ολίσθησης σειριακής εισόδου και εξόδου 8 ψηφίων στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.



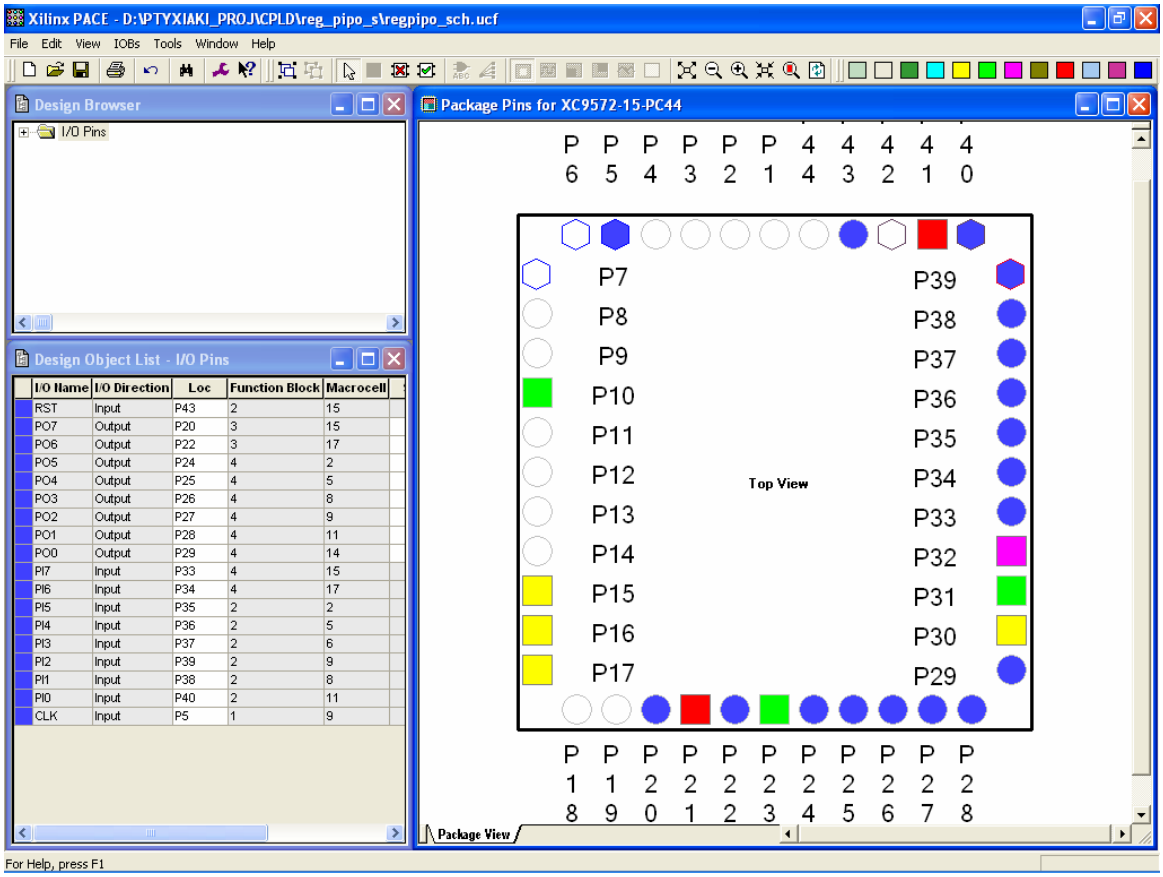
Εικόνα 2.72 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 2.73 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.74 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.75 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.7.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 όπως αναφέρεται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 2.42 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

11 architecture BHV of regpipo_vhd is	Αρχή δήλωσης αρχιτεκτονικής.
12 begin	Αρχή κώδικα αρχιτεκτονικής.
13 process (clk)	Αρχή της διεργασίας με λίστα ευαισθησίας την είσοδο **clk**.
14 variable tmp : STD_LOGIC_VECTOR (7 downto 0):= "00000000";	Δήλωση της μεταβλητής **tmp** οκτώ δυαδικών ψηφίων τύπου **std_logic** με αρχική δυαδική τιμή προσομοίωσης "00000000".
15 begin	Αρχή ενεργοποίησης της διεργασίας.
16 if (clk'event and clk='1') then	Αρχή εκτέλεσης **if-then**.
17 if rst='1' then	Αρχή εκτέλεσης **if-then-else.**
18 tmp := (others => '0');	Μηδενισμός της προηγούμενης τιμής του **tmp**.
19 else	Διακλάδωση εντολής **if-then-else.**
20 tmp := pi;	Μεταφορά τιμής της παράλληλης εισόδου **pi** στην μεταβλητή **tmp**.
21 end if;	Τέλος εκτέλεσης **if-then-else**.
22 end if;	Τέλος εκτέλεσης **if-then**.
23 po <= tmp;	Μεταφορά της τιμής της μεταβλητής **tmp** στη παράλληλη έξοδο.
24 end process;	Τέλος εκτέλεσης process.
25 end BHV;	Τέλος κώδικα αρχιτεκτονικής.

Σημειώσεις.

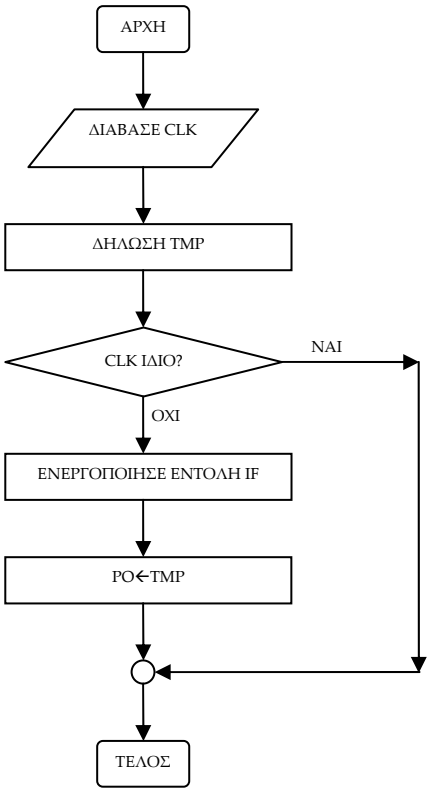
Γραμμή 15. Η ενεργοποίηση της διεργασίας δηλώνεται με τη λέξη begin. Η διεργασία ενεργοποιείται με την αλλαγή της λογικής κατάστασης της εισόδου **clk** της λίστας ευαισθησίας.

Γραμμή 16. Κατά την εκτέλεση της εντολής **if-then** έχουμε το λογικό έλεγχο για αλλαγή της εισόδου **clk** από λογικό 0 σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 17. Σε διαφορετική περίπτωση εκτελείται η γραμμή 23.

Γραμμή 17. Κατά την εκτέλεση της εντολής **if-then-else** έχουμε το λογικό έλεγχο για την κατάσταση της εισόδου **rst** σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 18. Σε διαφορετική περίπτωση (γραμμή 19) εκτελείται η γραμμή 20.

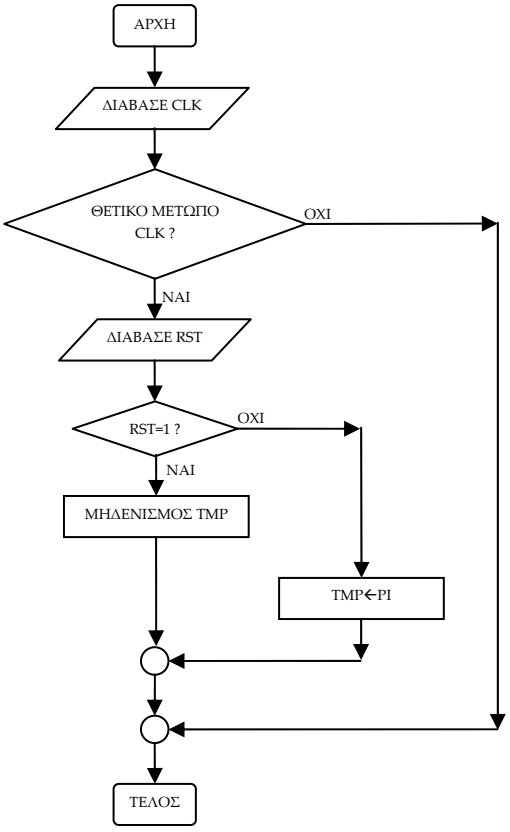
Πίνακας 2.43 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (clk)

variable tmp…= "00000000";


begin

  if … (if …else…end if )…end if;

po <= tmp;

end process;
```



Πίνακας 2.44 Η εντολή **if-then** και το **διάγραμμα ροής**

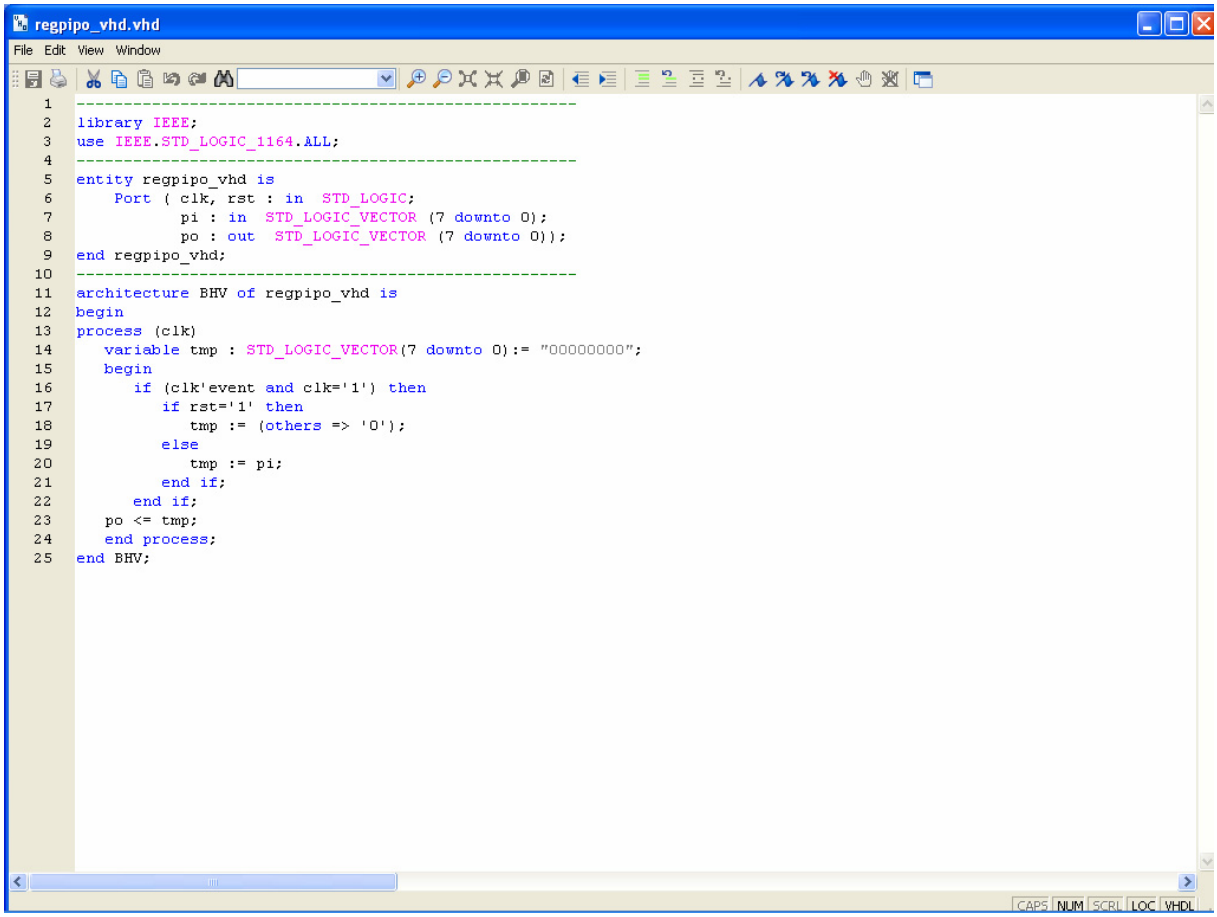
```
if clk='1' and clk'event then


   if rst='1' then

      tmp := (others => '0');

   else   tmp := pi;

   end if;

end if;
```



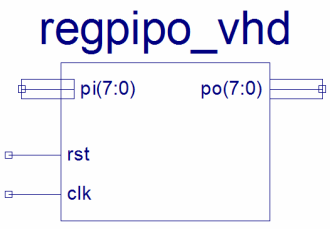
Σημείωση. Στα προηγούμενα διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

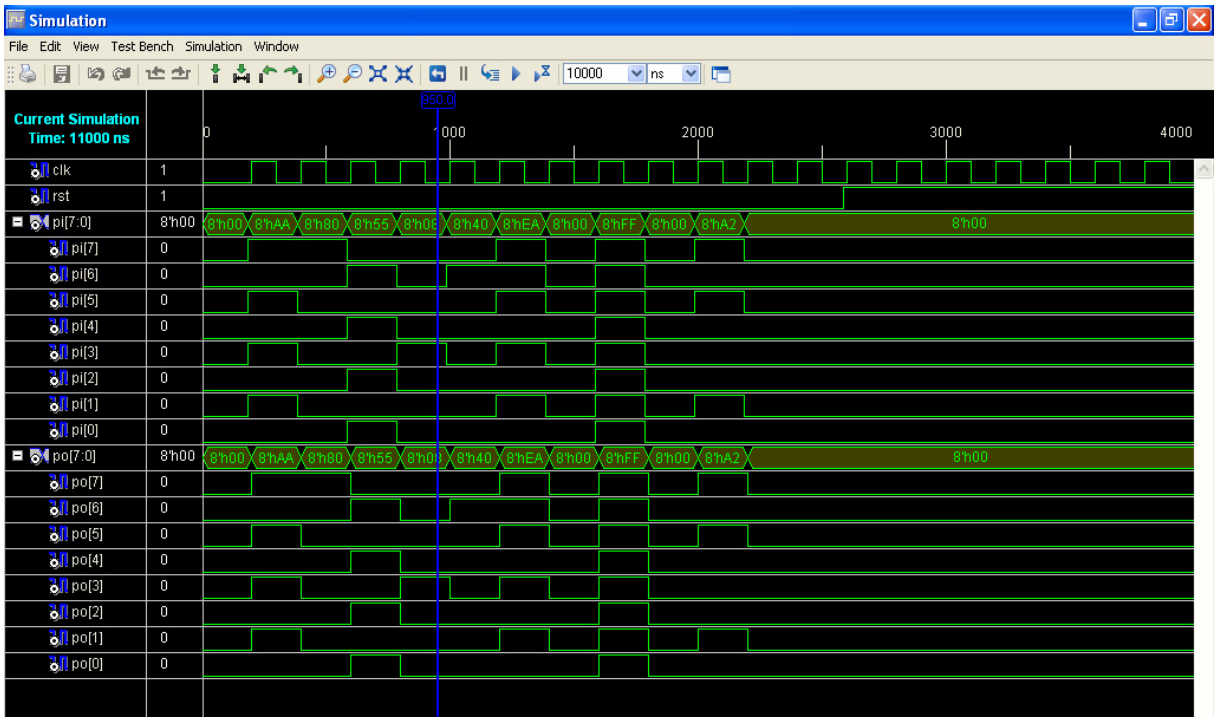
2.7.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)



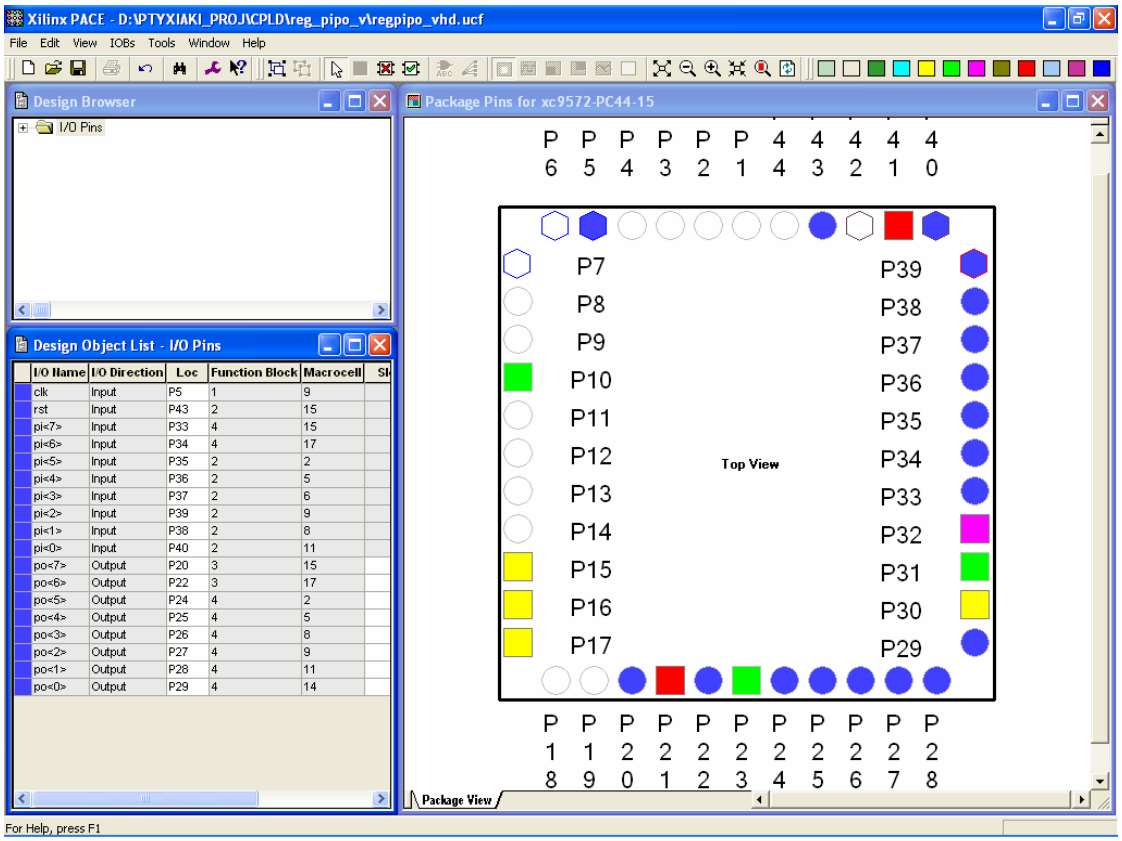
Εικόνα 2.76 VHDL αρχείο του κυκλώματος



Εικόνα 2.77 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.78 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.79 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.8 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ: ΚΥΚΛΙΚΟΣ ΜΕΤΡΗΤΗΣ 8 ΨΗΦΙΩΝ

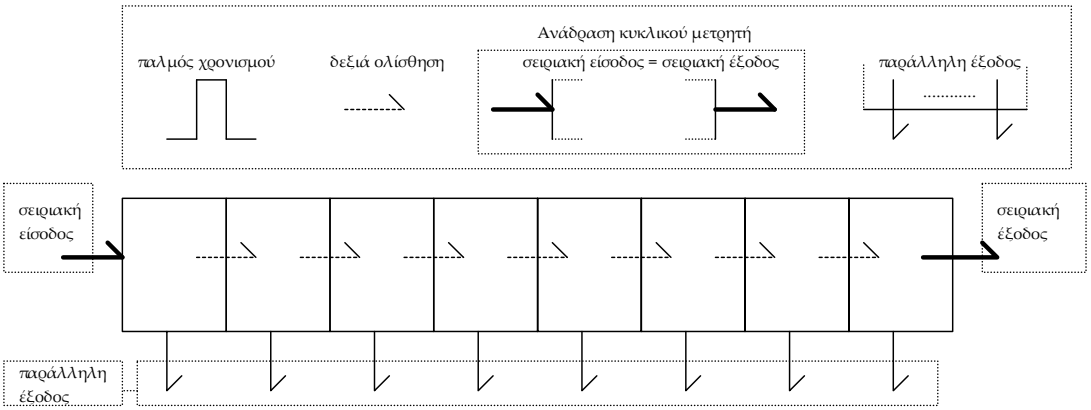
2.8.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

2.8.1.1 Σύντομη θεωρία

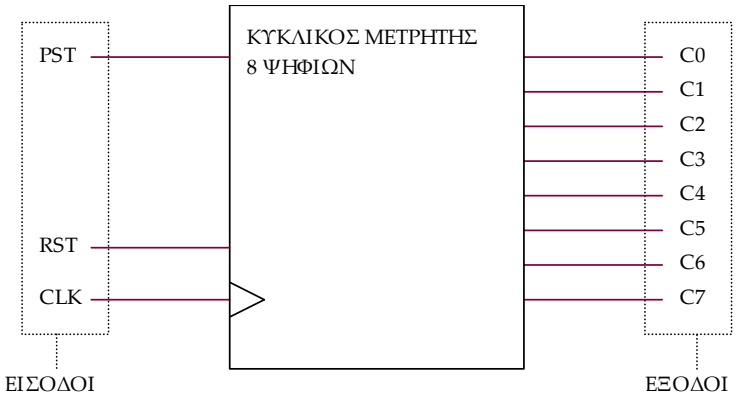
Οι κυκλικοί μετρητές (*ring counters*) είναι καταχωρητές ολίσθησης με ανατροφοδότηση της εισόδου τους από την έξοδο. Έχουν όλους τους πολυδονητές τους σε δυναμικό λογικής στάθμης 0 εκτός ενός πολυδονητή όπου βρίσκεται σε δυναμικό λογικής κατάστασης 1. Με την εφαρμογή των παλμών χρονισμού έχουμε την συνεχή διάδοση του λογικού 1 από τον ένα πολυδονητή στον άλλο. Θεωρούνται μετρητές γιατί δίνουν συγκεκριμένη ακολουθία καταστάσεων και συγκεκριμένα τόσες καταστάσεις όσοι και οι πολυδονητές που τους δομούν.

Ο κυκλικός μετρητής οκτώ ψηφίων έχει σαν εισόδους του το δυαδικό ψηφίο χρονισμού CLK, το δυαδικό ψηφίο σύγχρονου μηδενισμού RST και το δυαδικό ψηφίο PST για την είσοδο δυναμικού λογικής κατάστασης 1 σε έναν από τους πολυδονητές του. Η παράλληλη έξοδός του είναι τα δυαδικά ψηφία C0, C1,…, C7 (MSB=C7). Είναι κατασκευασμένος από οκτώ θετικής διέγερσης δισταθείς πολυδονητές (*flip-flop*) D τύπου σε σειρά συνδεδεμένους μεταξύ τους. Στην **εικόνα 2.80** τα ορθογώνια παριστούν τους οκτώ πολυδονητές που με τον παλμό χρονισμού ένα δυαδικό ψηφίο μεταβαίνει από τον ένα πολυδονητή στον άλλο όπου η έξοδος του τελευταίου είναι η είσοδος του πρώτου. Με τον ίδιο παλμό χρονισμού μπορούν να εξέρχονται στην παράλληλη έξοδο και τα οκτώ δυαδικά ψηφία πληροφορίας. Στην **εικόνα 2.81** απεικονίζεται το γραφικό σύμβολο του κυκλικού μετρητή. Από την περιγραφή και την είσοδο αρχικά του δυναμικού

λογικής στάθμης 1 στο δυαδικό ψηφίο C7 προκύπτει ο πίνακας αλήθειας του κυκλικού μετρητή (**πίνακας 2.45**).



Εικόνα 2.80 Κυκλικός μετρητής 8 ψηφίων



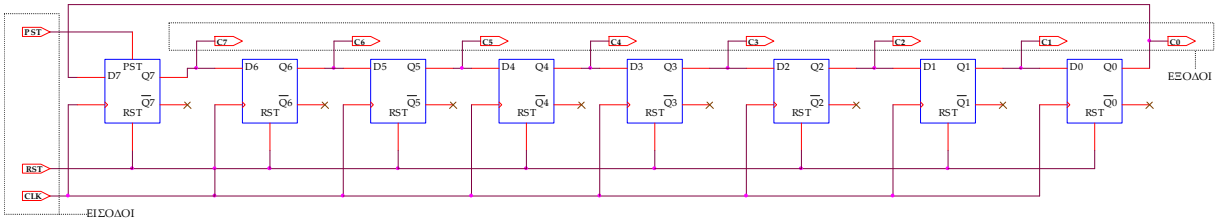
Εικόνα 2.81 Γραφικό σύμβολο κυκλικού μετρητή 8 ψηφίων

Πίνακας 2.45 Πίνακας αλήθειας κυκλικού μετρητή

παλμοί	**MSB**							**LSB**
CLK	**C7**	**C6**	**C5**	**C4**	**C3**	**C2**	**C1**	**C0**
0	1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0
3	0	0	0	1	0	0	0	0
4	0	0	0	0	1	0	0	0
5	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	1

2.8.1.2 Συμβατική ψηφιακή υλοποίηση

Η ειδική περίπτωση καταχωρητή ολίσθησης του κυκλικού μετρητή ακολουθεί τη γενικευμένη συνδεσμολογία των καταχωρητών ολίσθησης με δισταθείς πολυδονητές D τύπου σε σειρά (**εικόνα 2.82**).

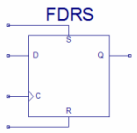


Εικόνα 2.82 Λογικό κύκλωμα υλοποίησης κυκλικού μετρητή 8 ψηφίων

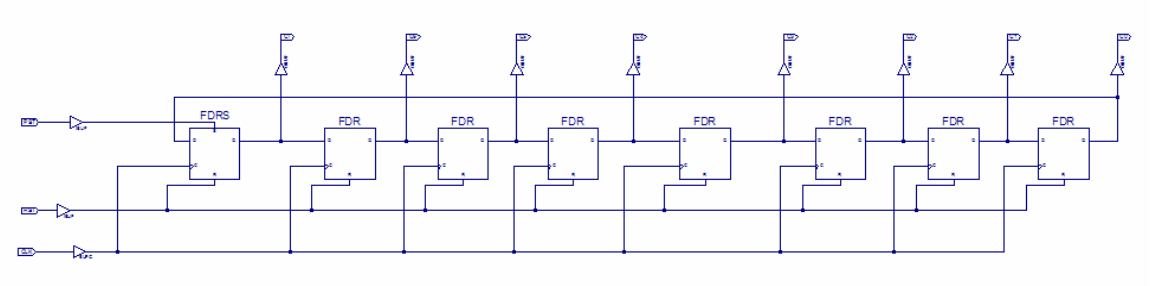
2.8.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα. Στον **πίνακα 2.46** φαίνεται το καινούργιο υλικό. Επιλέχτηκε για την σύγχρονη λειτουργία μηδενισμού και τοποθέτησης λογικής στάθμης 1 που προσφέρει ως πρώτος πολυδονητής του κυκλικού μετρητή. Η σύγχρονη λειτουργία μηδενισμού ενεργοποιείται μέσω της εισόδου R του πολυδονητή με λογικό 1. Η λειτουργία αυτή εξυπηρετεί την γενικότερη περίπτωση σύγχρονου μηδενισμού του κυκλικού μετρητή μέσω της εισόδου του RST. Η σύγχρονη τοποθέτηση λογικής στάθμης 1 ενεργοποιείται επίσης με λογικό 1 μέσω της εισόδου του πολυδονητή S. Η λειτουργία αυτή εξυπηρετεί την τοποθέτηση της λογικής στάθμης 1 σαν είσοδο πληροφορίας στον κυκλικό μετρητή για την ανακύκλωσή της μέσω της εισόδου του PST.

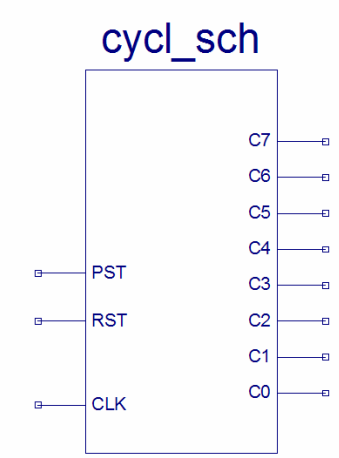
Πίνακας 2.46 Καινούργια υλικά σχηματικού

ΚΑΙΝΟΥΡΓΙΑ ΥΛΙΚΑ ΣΧΕΔΙΑΣΗΣ PROJECT NAVIGATOR			
ΣΧΗΜΑ	ΟΝΟΜΑΣΙΑ	ΙΔΙΟΤΗΤΑ	ΚΑΤΑΛΟΓΟΣ ΒΙΒΛΙΟΘΗΚΗΣ SYMBOL
	FDRS	D FLIP-FLOP ΜΕ ΣΥΓΧΡΟΝΟ RESET (R ΕΙΣΟΔΟΣ) ΚΑΙ ΣΥΓΧΡΟΝΟ PRESET (S ΕΙΣΟΔΟΣ)	FLIP_FLOP

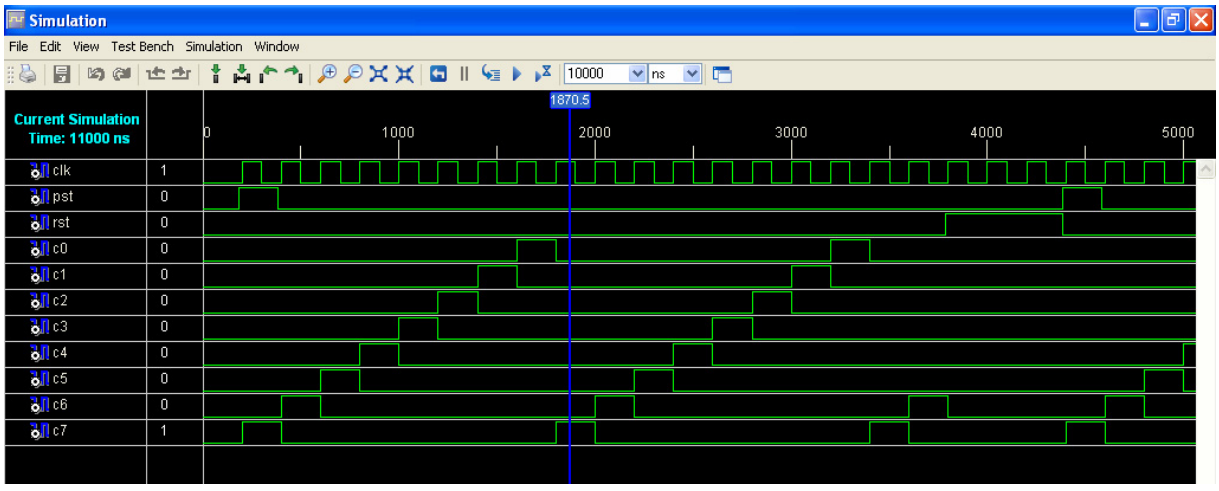
Από τις πληροφορίες του προγράμματος για τον πολυδονητή FDRS δηλώνεται ό,τι η λειτουργία του σύγχρονου μηδενισμού προηγείται της λειτουργίας σύγχρονης τοποθέτησης της λογικής στάθμης 1. Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



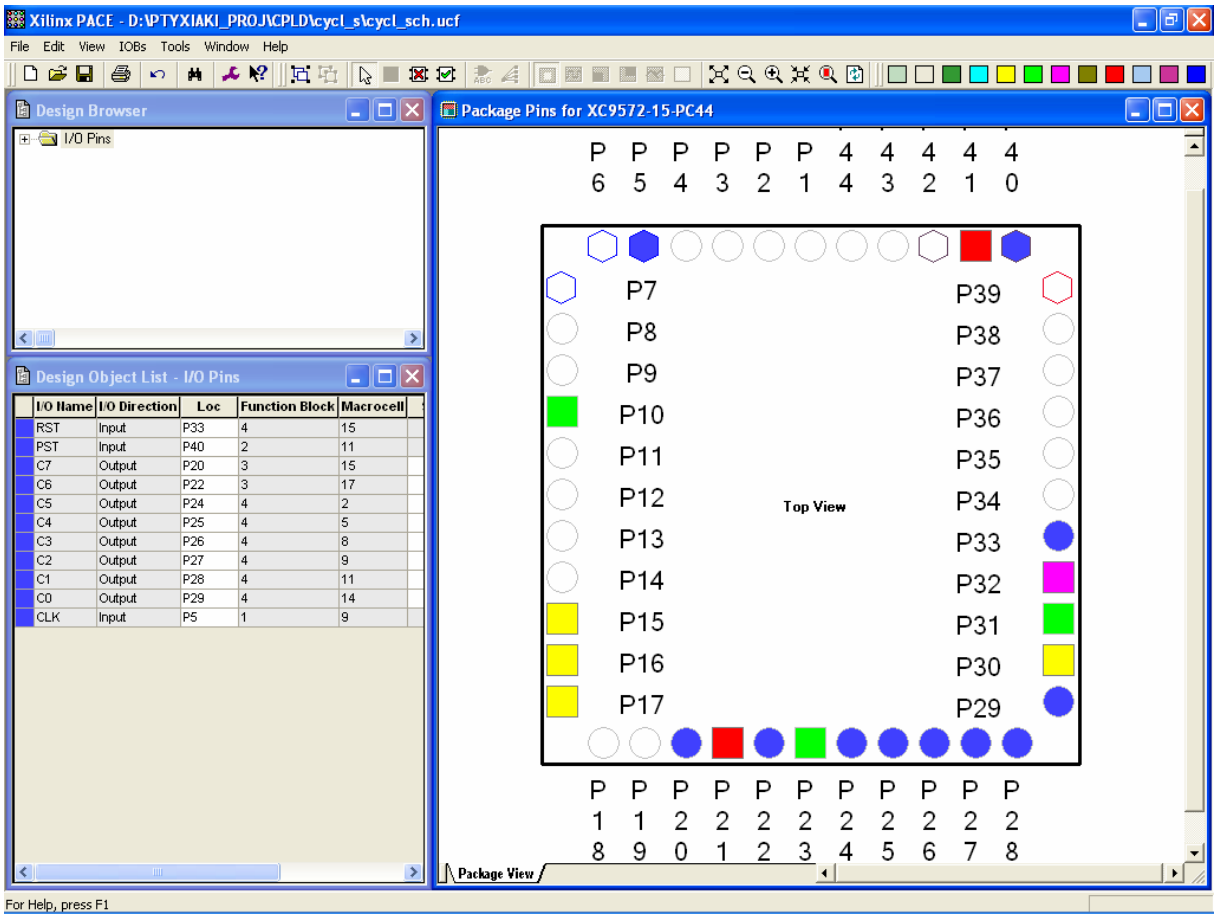
Εικόνα 2.83 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 2.84 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.85 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.86 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.8.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 όπως αναφέρεται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

Πίνακας 2.47 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

10 architecture BHV of cycl_vhd is	Αρχή δήλωσης αρχιτεκτονικής.
11 begin	Αρχή κώδικα αρχιτεκτονικής.
12 process (clk)	Αρχή της διεργασίας με λίστα ευαισθησίας την είσοδο **clk**.
13 variable tmp : STD_LOGIC_VECTOR (7 downto 0):= "00000000";	Δήλωση της μεταβλητής **tmp** οκτώ δυαδικών ψηφίων τύπου **std_logic** με αρχική δυαδική τιμή προσομοίωσης "00000000".
14 variable tmp1 : STD_LOGIC;	Δήλωση της μεταβλητής **tmp1** τύπου **std_logic**.
15 begin	Αρχή ενεργοποίησης της διεργασίας.
16 if (clk'event and clk='1') then	Αρχή εκτέλεσης **if-then**.
17 if rst='1' then	Αρχή εκτέλεσης **if-then-else**.
18 tmp := (others => '0');	Μηδενισμός της προηγούμενης τιμής του **tmp**.
19 else	Διακλάδωση εντολής **if-then-else**.
20 if pst='0' then	Αρχή εκτέλεσης **if-then-else**.
21 tmp1 := tmp (0);	Μεταφορά τιμήστου δυαδικού ψηφίου 0 της μεταβλητής **temp** στην μεταβλητή **tmp1**.
22 for i in 0 to 6 loop	Αρχή εκτέλεσης βρόχου της εντολής **for-loop** με ακέραια μεταβλητής **i** μεταξύ τιμών από 0 ως και 6.
23 tmp(i) := tmp(i+1);	Μεταφορά τιμής στην μεταβλητή **tmp** που αντιστοιχεί στην δεξιά ολίσθηση κατά ένα δυαδικό ψηφίο.
24 end loop;	Τέλος εκτέλεσης βρόχου.
25 tmp(7) := tmp1;	Μεταφορά τιμής της μεταβλητής **tmp1** στο έβδομο δυαδικό ψηφίο της μεταβλητής **tmp** που αντιστοιχεί στην καταχώρηση της πληροφορίας που θα ανακυκλωθεί σειριακά.
26 else	Διακλάδωση εντολής **if-then-else.**
27 tmp:=”10000000”;	Παράλληλη μεταφορά της δυαδικής τιμής 1000000 στην μεταβλητής **tmp** που αντιστοιχεί στην καταχώρηση λογικού 1 στον κυκλικό μετρητή.
28 end if;	Τέλος εκτέλεσης **if-then-else**.

29 end if;	Τέλος εκτέλεσης **if-then-else**.
30 end if;	Τέλος εκτέλεσης **if-then**.
31 c <= tmp;	Μεταφορά της τιμής της μεταβλητής **tmp** στην παράλληλη έξοδο **c**.
32 end process;	Τέλος εκτέλεσης process.
33 end BHV;	Τέλος κώδικα αρχιτεκτονικής.

Σημειώσεις.

Γραμμή 11. Η ενεργοποίηση της διεργασίας δηλώνεται με τη λέξη begin. Η διεργασία ενεργοποιείται με την αλλαγή της λογικής κατάστασης της εισόδου **clk** της λίστας ευαισθησίας.

Γραμμή 16. Κατά την εκτέλεση της εντολής **if-then** έχουμε το λογικό έλεγχο για αλλαγή της εισόδου **clk** από λογικό 0 σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 17. Σε διαφορετική περίπτωση εκτελείται η γραμμή 31.

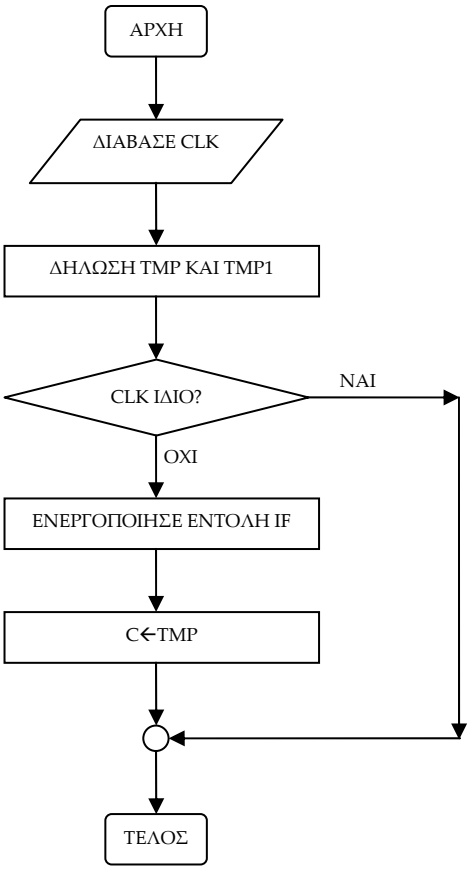
Γραμμή 17. Κατά την εκτέλεση της εντολής **if-then-else** έχουμε το λογικό έλεγχο για την κατάσταση της εισόδου **rst** σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 18. Σε διαφορετική περίπτωση (γραμμή 19) εκτελείται η γραμμή 20.

Γραμμή 20. Κατά την εκτέλεση της εντολής **if-then-else** έχουμε το λογικό έλεγχο για την κατάσταση της εισόδου **pst** σε λογικό 0. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 21. Σε διαφορετική περίπτωση (γραμμή 26) εκτελείται η γραμμή 27.

Γραμμή 22. Στην περίπτωση που η μεταβλητή **i** πάρει την τιμή 7 τότε εκτελείται η εντολή της γραμμής 25 λύνοντας το βρόχο εκτέλεσης της εντολής **for loop**.

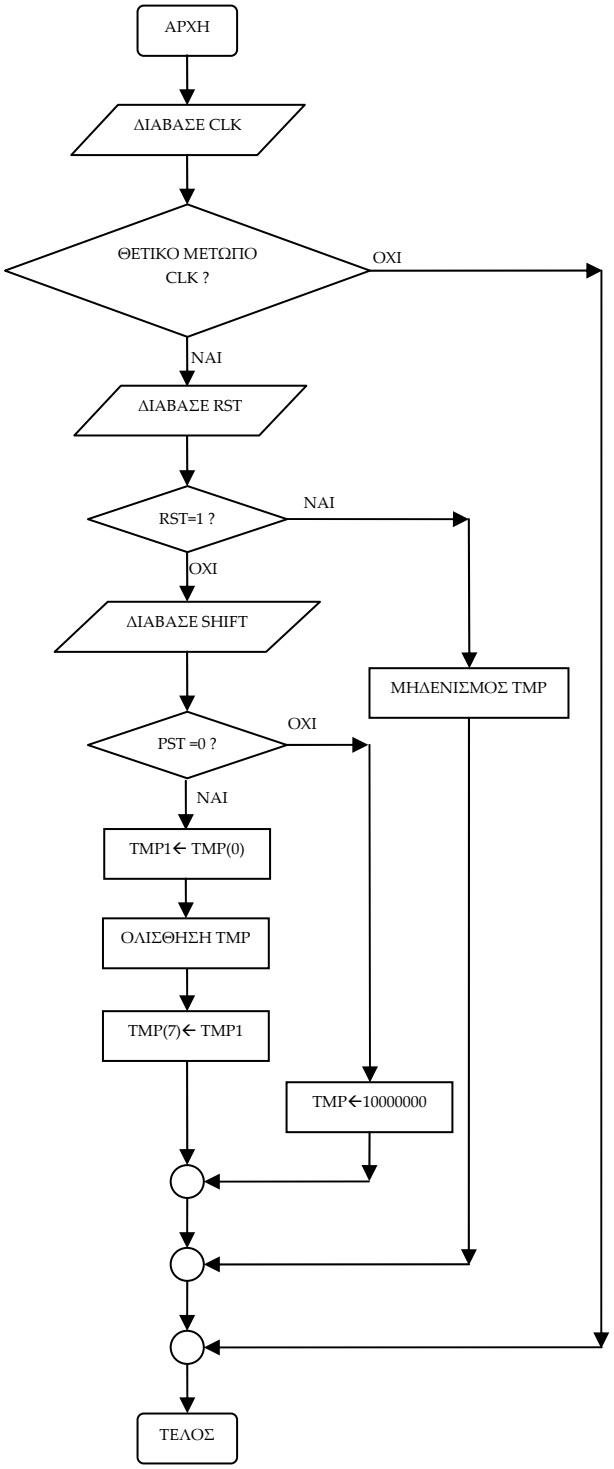
Πίνακας 2.48 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (clk)

variable tmp …; variable tmp1 …;

begin

  if … (if …else (if … (for…end
loop) else …end if)…end if )…end
if;

c <= tmp;

end process;
```



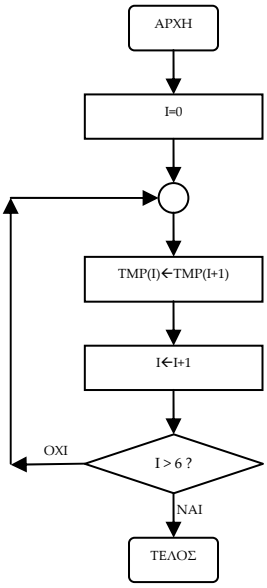
Πίνακας 2.49 Η εντολή **if-then** και το **διάγραμμα ροής**

```
if (clk'event and clk='1') then

 if rst='1' then

   tmp := (others => '0');

 else if pst='0' then

   tmp1 := tmp(0);

    for i in 0 to 6 … end loop;

   tmp(7) := tmp1;

   else  tmp:=”10000000”;

   end if;

 end if;

end if;
```



Πίνακας 2.50 Η ολίσθηση με εντολή **for loop** και το **διάγραμμα ροής**

```
for i in 0 to 6 loop

tmp(i) := tmp(i+1);

end loop;
```



Σημείωση. Στα προηγούμενα διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

2.8.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

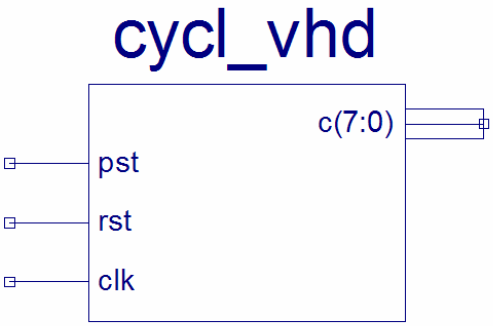
cycl_vhd.vhd

File Edit View Window

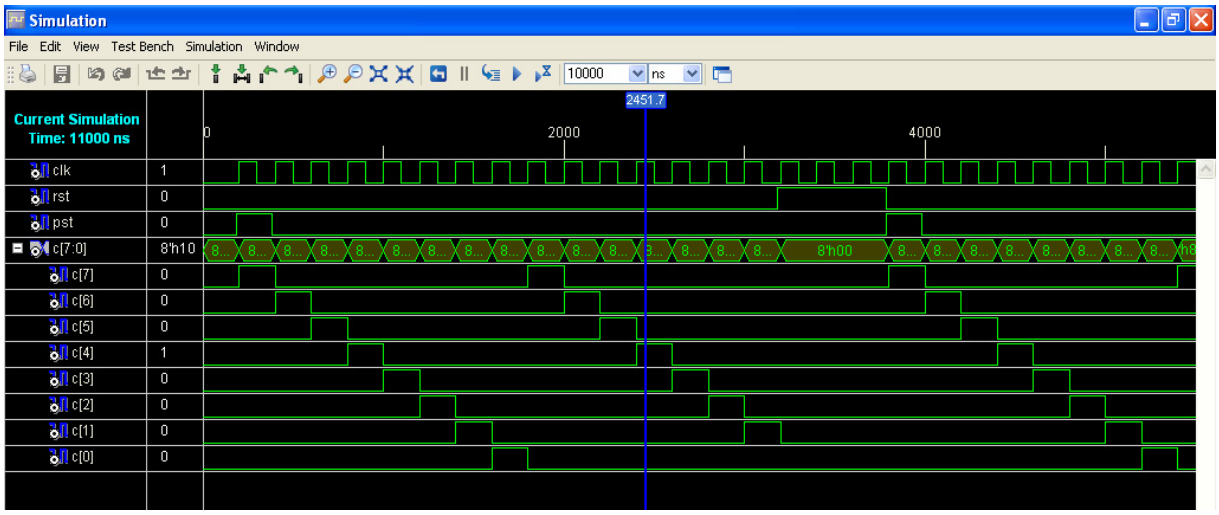
```
1  ----------------------------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  ----------------------------------------------------------------------
5  entity cycl_vhd is
6      Port ( clk, rst, pst : in  STD_LOGIC;
7             c : out  STD_LOGIC_VECTOR (7 downto 0));
8  end cycl_vhd;
9  ----------------------------------------------------------------------
10 architecture BHV of cycl_vhd is
11 begin
12 process (clk)
13    variable tmp : STD_LOGIC_VECTOR(7 downto 0):= "00000000";
14    variable tmp1 : STD_LOGIC;
15    begin
16       if (clk'event and clk='1') then
17          if rst='1' then
18             tmp := (others => '0');
19          else
20             if pst='0' then
21                tmp1 := tmp(0);
22                for i in 0 to 6 loop
23                   tmp(i) := tmp(i+1);
24                end loop;
25                tmp(7) := tmp1;
26             else
27                tmp:="10000000";
28             end if;
29          end if;
30       end if;
31    c <= tmp;
32    end process;
33 end BHV;
```

CAPS NUM SCRL LOC VHDL

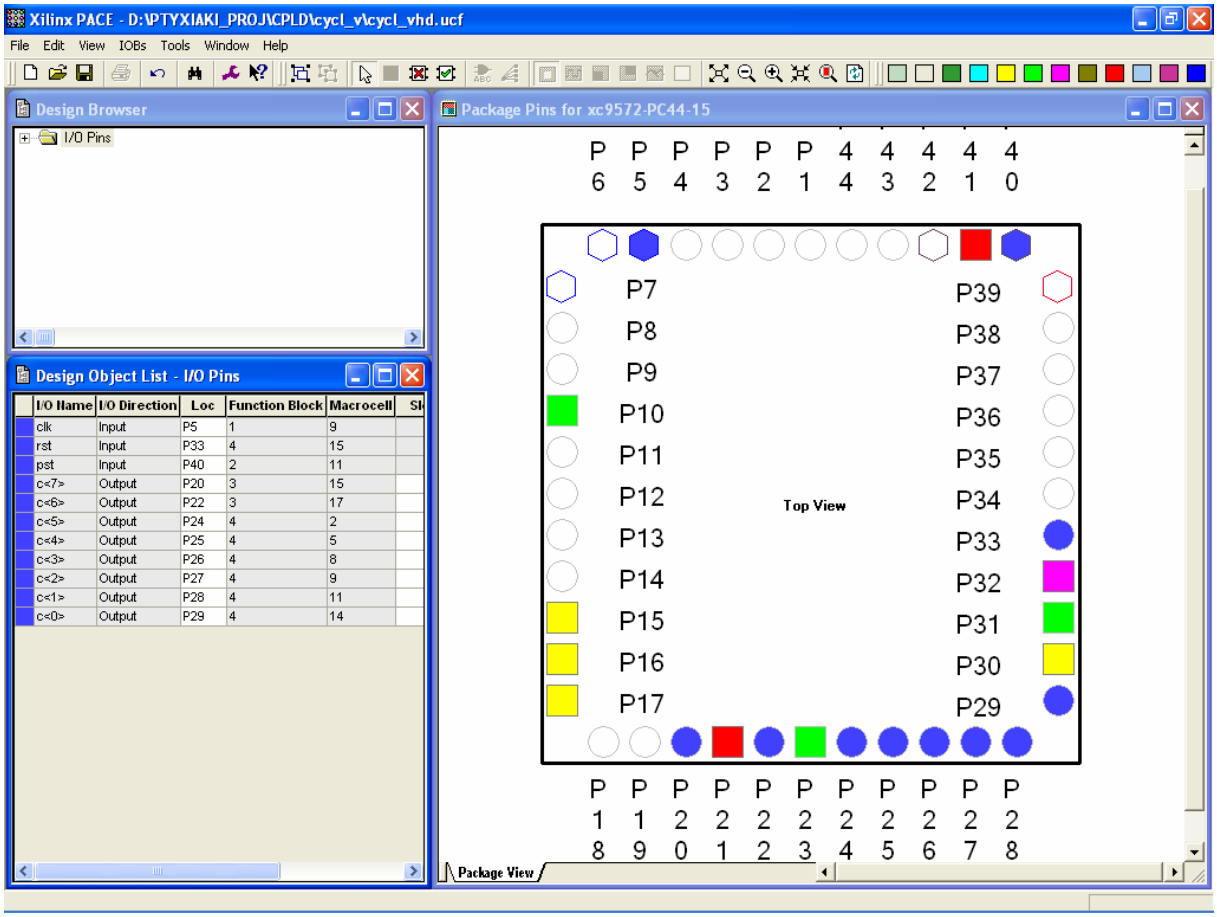
Εικόνα 2.87 VHDL αρχείο του κυκλώματος



Εικόνα 2.88 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.89 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.90 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

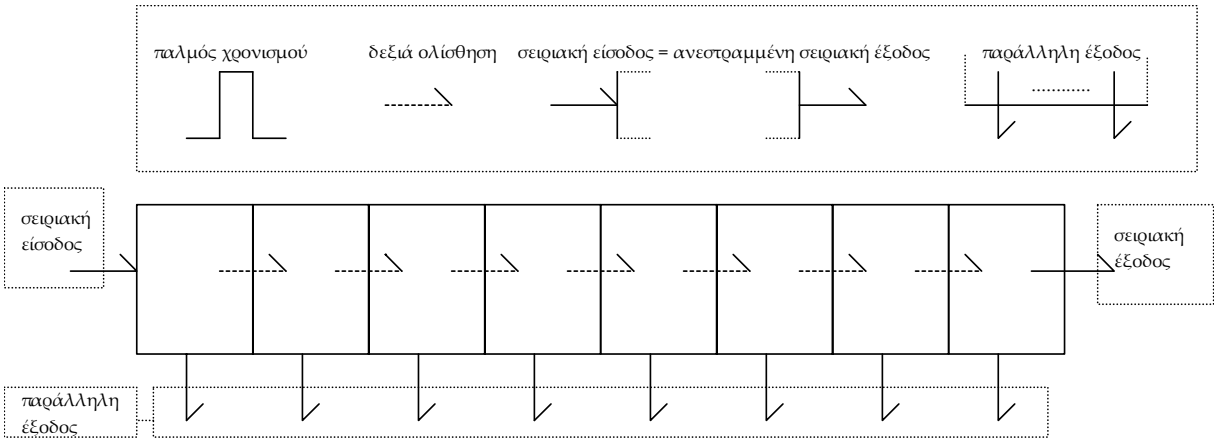
2.9 ΣΥΝΟΛΙΚΗ ΕΡΓΑΣΙΑ: ΜΕΤΡΗΤΗΣ JOHNSON 8 ΨΗΦΙΩΝ

2.9.1 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

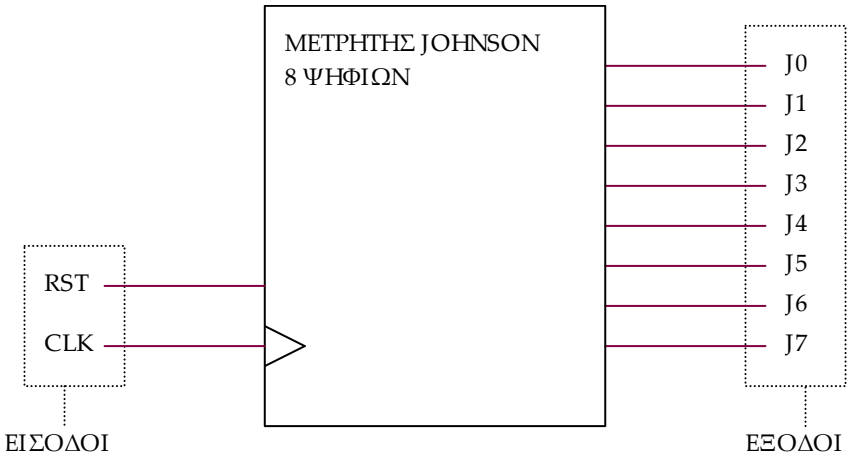
2.9.1.1 Σύντομη θεωρία

Οι μετρητές Johnson (*Johnson counters*) είναι καταχωρητές ολίσθησης με ανατροφοδότηση της εισόδου τους από την ανάστροφη έξοδο. Θεωρούνται μετρητές γιατί δίνουν συγκεκριμένη ακολουθία καταστάσεων και μάλιστα διπλάσιες από τους πολυδονητές που τους δομούν.

Ο μετρητής Johnson οκτώ ψηφίων έχει σαν εισόδους του το δυαδικό ψηφίο χρονισμού CLK και το δυαδικό ψηφίο σύγχρονου μηδενισμού RST. Η παράλληλη έξοδός του είναι τα δυαδικά ψηφία J0, J1,…, J7 (MSB=J7). Είναι κατασκευασμένος από οκτώ θετικής διέγερσης δισταθείς πολυδονητές (*flip-flop*) D τύπου σε σειρά συνδεδεμένους μεταξύ τους. Στην **εικόνα 2.91** τα ορθογώνια παριστούν τους οκτώ πολυδονητές που με τον παλμό χρονισμού ένα δυαδικό ψηφίο μεταβαίνει από τον ένα πολυδονητή στον άλλο όπου η ανάστροφη έξοδος του τελευταίου είναι η είσοδος του πρώτου. Με τον ίδιο παλμό χρονισμού μπορούν να εξέρχονται στην παράλληλη έξοδο και τα οκτώ δυαδικά ψηφία πληροφορίας. Στην **εικόνα 2.92** απεικονίζεται το γραφικό σύμβολο του κυκλικού μετρητή. Από την περιγραφή και την είσοδο αρχικά του δυναμικού λογικής στάθμης 1 στο δυαδικό ψηφίο J7 προκύπτει ο πίνακας αλήθειας του μετρητή Johnson (**πίνακας 2.51**).



Εικόνα 2.91 Μετρητής Johnson 8 ψηφίων



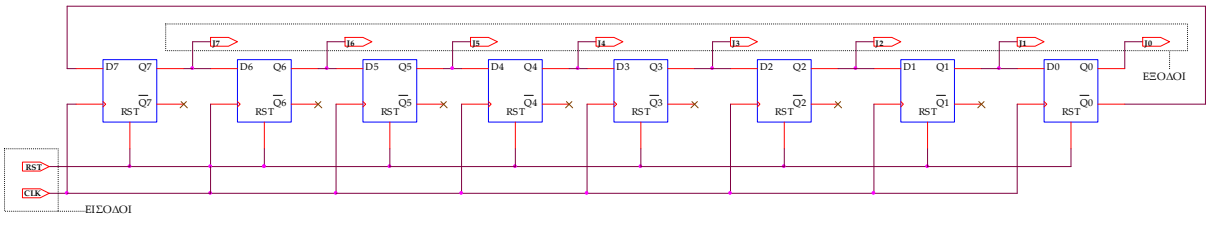
Εικόνα 2.92 Γραφικό σύμβολο μετρητή Johnson 8 ψηφίων

Πίνακας 2.51 Πίνακας αλήθειας μετρητή Johnson

παλμοί	**MSB**							**LSB**
CLK	**J7**	**J6**	**J5**	**J4**	**J3**	**J2**	**J1**	**J0**
0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0
4	1	1	1	1	0	0	0	0
5	1	1	1	1	1	0	0	0
6	1	1	1	1	1	1	0	0
7	1	1	1	1	1	1	1	0
8	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1
10	0	0	1	1	1	1	1	1
11	0	0	0	1	1	1	1	1
12	0	0	0	0	1	1	1	1
13	0	0	0	0	0	1	1	1
14	0	0	0	0	0	0	1	1
15	0	0	0	0	0	0	0	1

2.9.1.2 Συμβατική ψηφιακή υλοποίηση

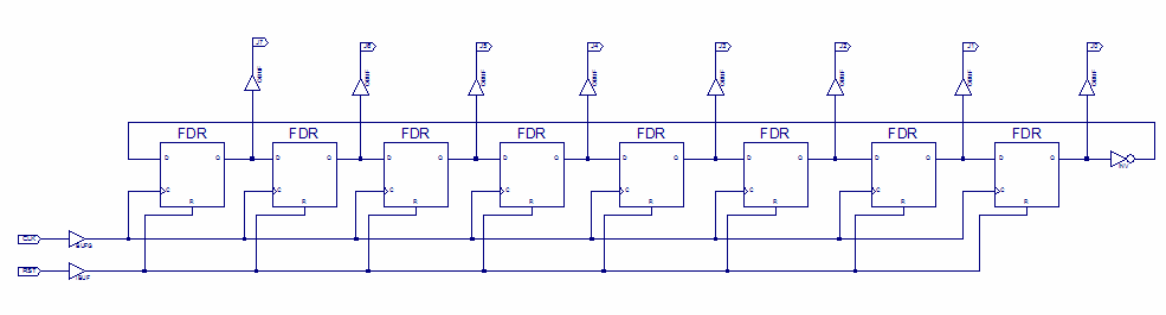
Η ειδική περίπτωση καταχωρητή ολίσθησης του μετρητή Johnson ακολουθεί τη γενικευμένη συνδεσμολογία των καταχωρητών ολίσθησης με δισταθείς πολυδονητές D τύπου σε σειρά (**εικόνα 2.93**).



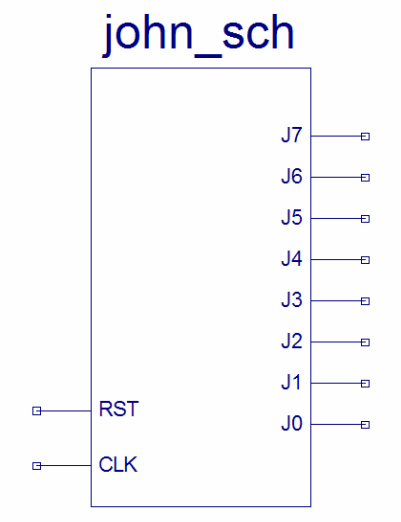
Εικόνα 2.93 Λογικό κύκλωμα υλοποίησης μετρητή Johnson 8 ψηφίων

2.9.2 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

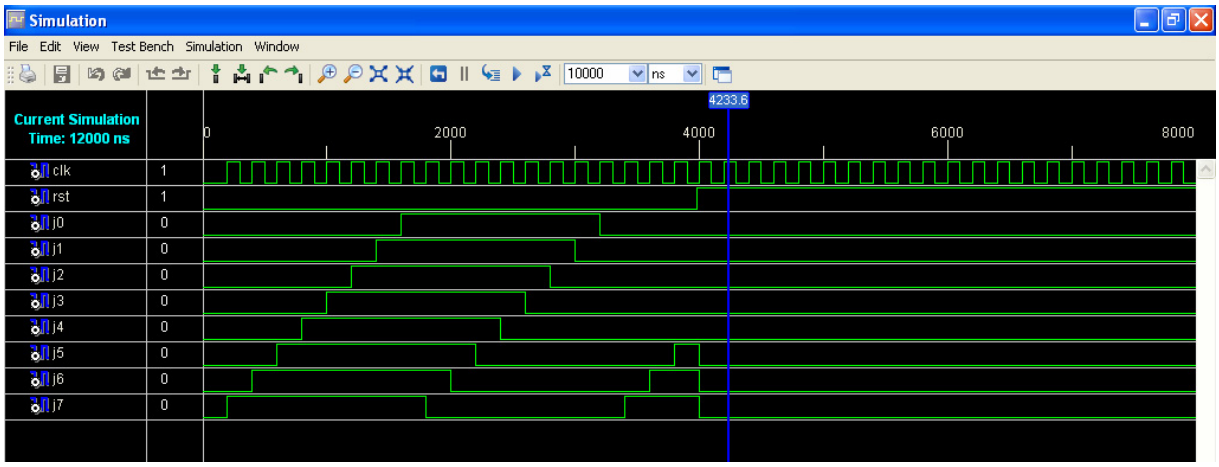
Όλες οι διαδικασίες που αφορούν από τη δημιουργία της νέας συνολικής εργασίας (project) μέχρι το τέλος του γρήγορου προγραμματισμού επαναλαμβάνονται οι ίδιες όπως οι αντίστοιχες της συνολικής εργασίας του πολυπλέκτη 8:1 και του κυκλικού μετρητή οκτώ ψηφίων στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα. Από την τέλεση των παραπάνω διαδικασιών στη σχηματική γλώσσα προκύπτουν όλα τα παρακάτω.



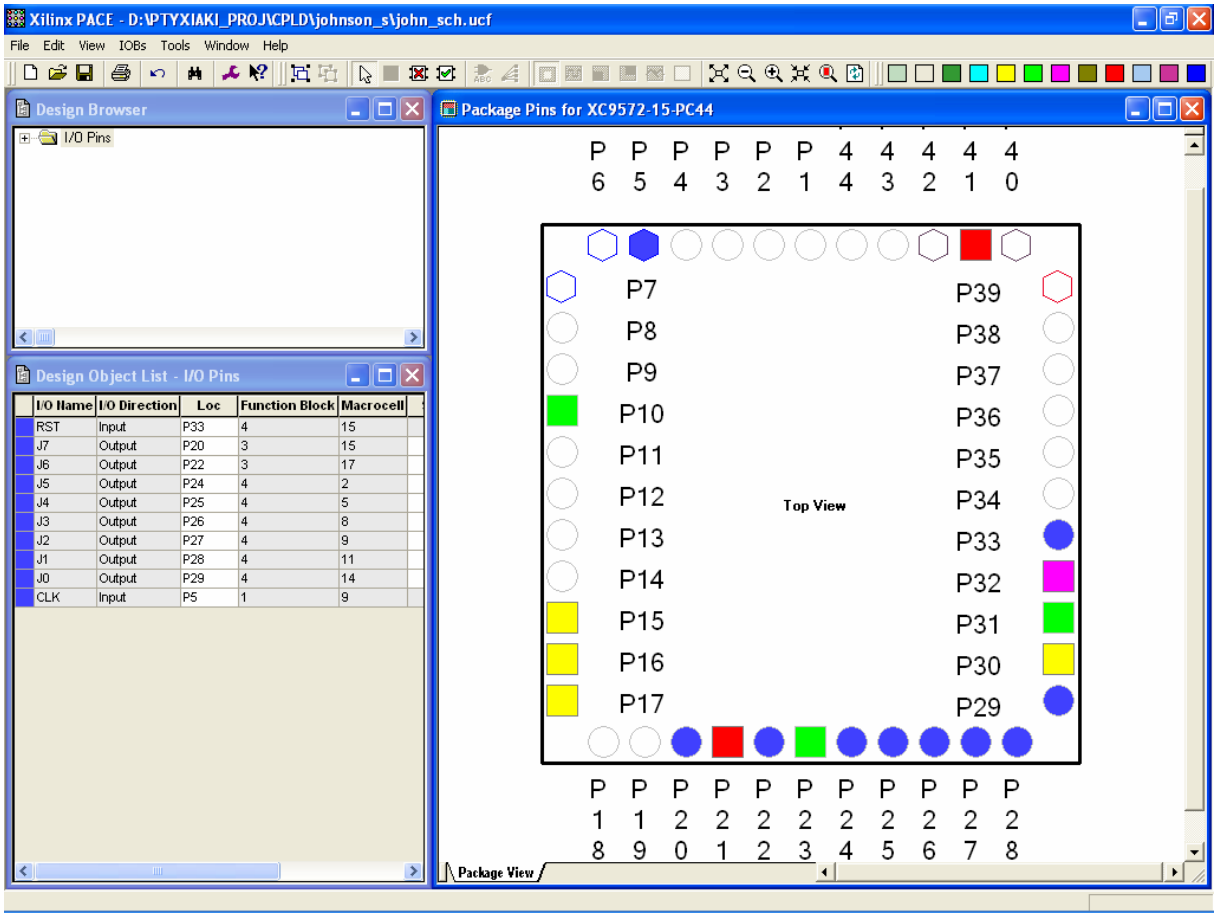
Εικόνα 2.94 Σχηματικό διάγραμμα κυκλώματος



Εικόνα 2.95 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.96 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.97 Παράθυρο αντιστοίχισης εισόδων/εξόδων (I/O) και ακίδων (pins)

2.9.3 ΑΝΑΛΥΣΗ ΓΙΑ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Στην τρέχουσα συνολική εργασία (project) οι δηλώσεις που αφορούν τις περιοχές δήλωσης βιβλιοθηκών και οντότητας επαναλαμβάνονται όπως οι αντίστοιχες της συνολικής εργασίας (project) για τον πολυπλέκτη 8:1 όπως αναφέρεται στις ανάλογες σελίδες του παρόντος συγγράμματος. Αναμενόμενες οι αλλαγές σε ονόματα.

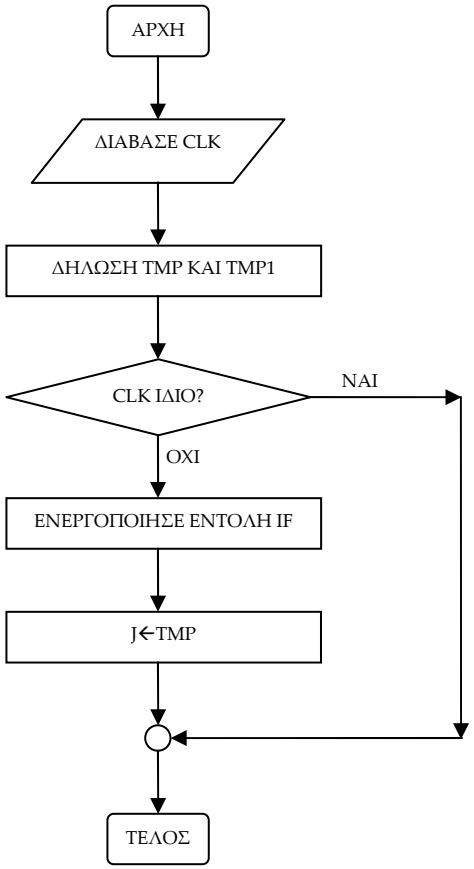
Πίνακας 2.52 Περιοχή δήλωσης αρχιτεκτονικής (architecture)

10 architecture BHV of john_vhd is	Αρχή δήλωσης αρχιτεκτονικής.
11 begin	Αρχή κώδικα αρχιτεκτονικής.
12 process (clk)	Αρχή της διεργασίας με λίστα ευαισθησίας την είσοδο **clk**.
13 variable tmp : STD_LOGIC_VECTOR (7 downto 0):= "00000000";	Δήλωση της μεταβλητής **tmp** οκτώ δυαδικών ψηφίων τύπου **std_logic** με αρχική δυαδική τιμή προσομοίωσης "00000000".
14 variable tmp1 : STD_LOGIC;	Δήλωση της μεταβλητής **tmp1** τύπου **std_logic**.
15 begin	Αρχή ενεργοποίησης της διεργασίας.
16 if (clk'event and clk='1') then	Αρχή εκτέλεσης **if-then**.
17 if rst='1' then	Αρχή εκτέλεσης **if-then-else**.
18 tmp := (others => '0');	Μηδενισμός της προηγούμενης τιμής του **tmp**.
19 else	Διακλάδωση εντολής **if-then-else**.
20 tmp1 := tmp (0);	Μεταφορά τιμήστου δυαδικού ψηφίου 0 της μεταβλητής **temp** στην μεταβλητή **tmp1**.
21 for i in 0 to 6 loop	Αρχή εκτέλεσης βρόχου της εντολής **for-loop** με ακέραια μεταβλητής **i** μεταξύ τιμών από 0 ως και 6.
22 tmp(i) := tmp(i+1);	Μεταφορά τιμής στην μεταβλητή **tmp** που αντιστοιχεί στην δεξιά ολίσθηση κατά ένα δυαδικό ψηφίο.
23 end loop;	Τέλος εκτέλεσης βρόχου.
24 tmp(7) := (not tmp1);	Μεταφορά ανεστραμμένης τιμής της μεταβλητής **tmp1** στο έβδομο δυαδικό ψηφίο της μεταβλητής **tmp** που αντιστοιχεί στην καταχώρηση της πληροφορίας που θα ανακυκλωθεί σειριακά.
25 end if;	Τέλος εκτέλεσης **if-then-else**.
26 end if;	Τέλος εκτέλεσης **if-then**.
27 j <= tmp;	Μεταφορά της τιμής της μεταβλητής **tmp** στην παράλληλη έξοδο **j**.
28 end process;	Τέλος εκτέλεσης process.
29 end BHV;	Τέλος κώδικα αρχιτεκτονικής.

Σημειώσεις.
Γραμμή 11. Η ενεργοποίηση της διεργασίας δηλώνεται με τη λέξη begin. Η διεργασία ενεργοποιείται με την αλλαγή της λογικής κατάστασης της εισόδου **clk** της λίστας ευαισθησίας.
Γραμμή 16. Κατά την εκτέλεση της εντολής **if-then** έχουμε το λογικό έλεγχο για αλλαγή της εισόδου **clk** από λογικό 0 σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 17. Σε διαφορετική περίπτωση εκτελείται η γραμμή 27.
Γραμμή 17. Κατά την εκτέλεση της εντολής **if-then-else** έχουμε το λογικό έλεγχο για την κατάσταση της εισόδου **rst** σε λογικό 1. Αν το αποτέλεσμα του ελέγχου είναι true εκτελείται η επόμενη γραμμή 18. Σε διαφορετική περίπτωση (γραμμή 19) εκτελείται η γραμμή 20.
Γραμμή 21. Στην περίπτωση που η μεταβλητή **i** πάρει την τιμή 7 τότε εκτελείται η εντολή της γραμμής 24 λύνοντας το βρόχο εκτέλεσης της εντολής **for loop**.

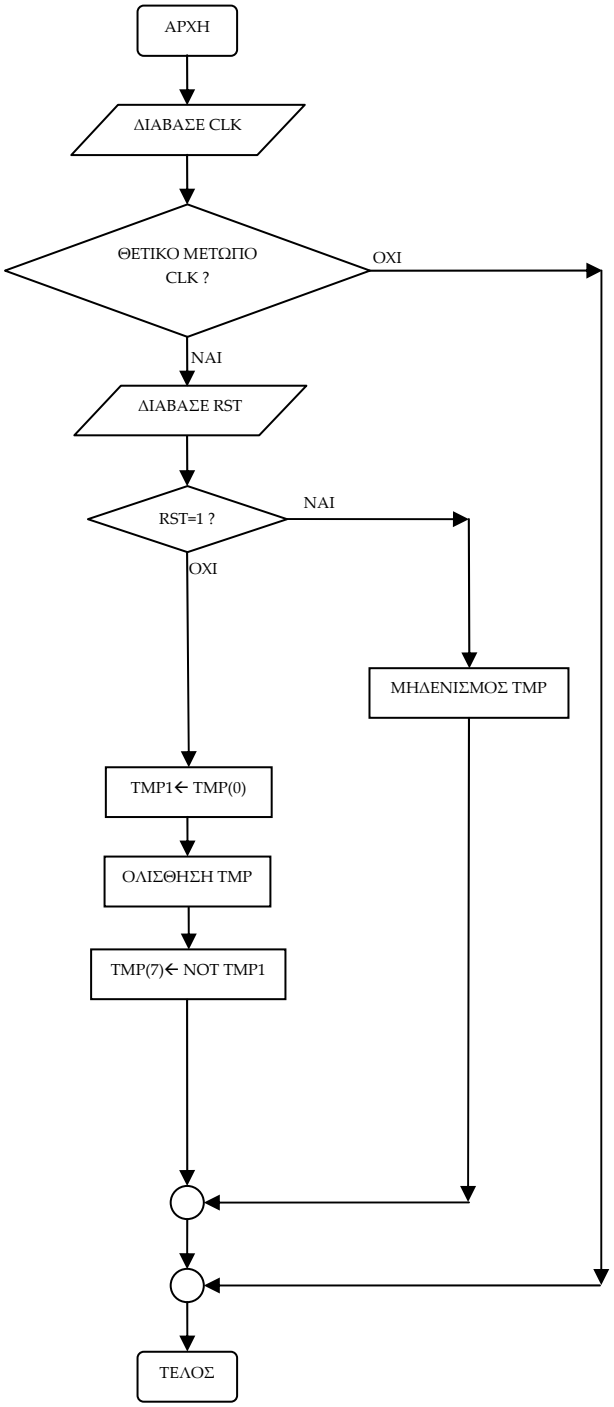
Πίνακας 2.53 Η **διεργασία (process)** και το **διάγραμμα ροής**

```
process (clk)

variable tmp …; variable tmp1 …;

begin

  if … (if …else (for…end loop) …end if)…end
if;

j <= tmp;

end process;
```



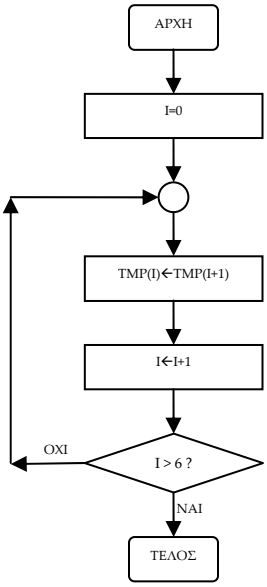
Πίνακας 2.54 Η εντολή **if-then** και το **διάγραμμα ροής**

```
if (clk'event and clk='1') then

 if rst='1' then

   tmp := (others => '0');

 else…tmp1 := tmp(0);

   for i in 0 to 6 … end loop;

   tmp(7) := (not tmp1);

 end if;

end if;
```



Πίνακας 2.55 Η ολίσθηση με εντολή **for loop** και το **διάγραμμα ροής**

```
for i in 0 to 6 loop

tmp(i) := tmp(i+1);

end loop;
```



Σημείωση. Στα προηγούμενα διαγράμματα ροής (**flow charts**) η θέση του αγγλικού ερωτηματικού (?) εκφράζει ερώτηση. Η θέση του συμβόλου «←» εκφράζει ανάθεση τιμής.

Παρομοίως για τις υπόλοιπες διαδικασίες μέχρι τον γρήγορο προγραμματισμό παραπέμπεται ο αναγνώστης στις αντίστοιχες διαδικασίες για τον πολυπλέκτη 8:1.

2.9.4 ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

john_vhd.vhd

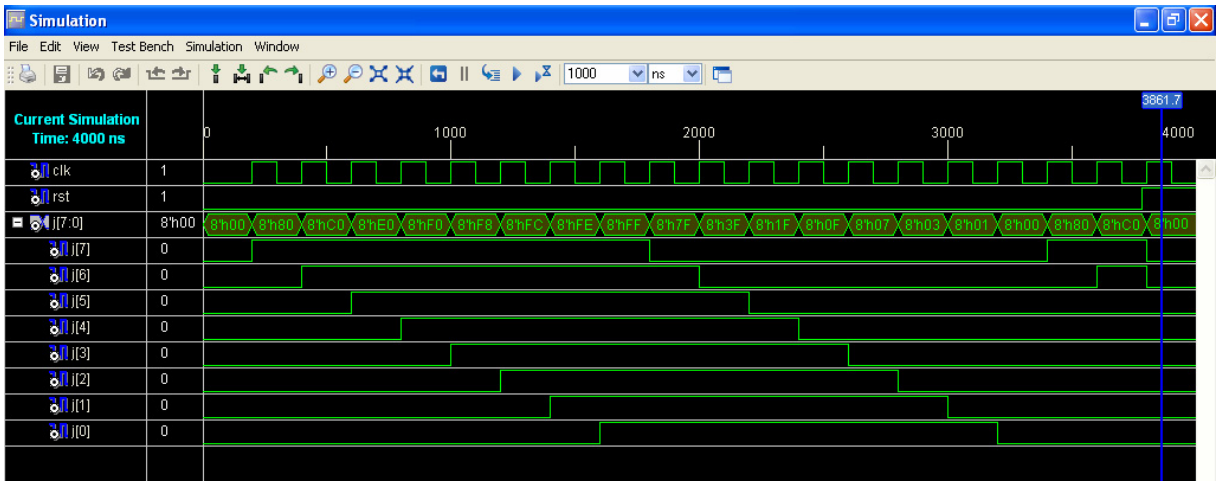
File Edit View Window

```
1  ----------------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  ----------------------------------------------------------
5  entity john_vhd is
6      Port ( clk, rst: in  STD_LOGIC;
7             j : out  STD_LOGIC_VECTOR (7 downto 0));
8  end john_vhd;
9  ----------------------------------------------------------
10 architecture BHV of john_vhd is
11 begin
12    process (clk)
13    variable tmp : STD_LOGIC_VECTOR(7 downto 0):="00000000";
14    variable tmp1 : STD_LOGIC := '0';
15    begin
16       if (clk'event and clk='1') then
17          if rst='1' then
18             tmp := (others => '0');
19          else
20             tmp1 := tmp(0);
21             for i in 0 to 6 loop
22                tmp(i) := tmp(i+1);
23             end loop;
24             tmp(7) := (not tmp1);
25          end if;
26       end if;
27    j <= tmp;
28    end process;
29 end BHV;
```

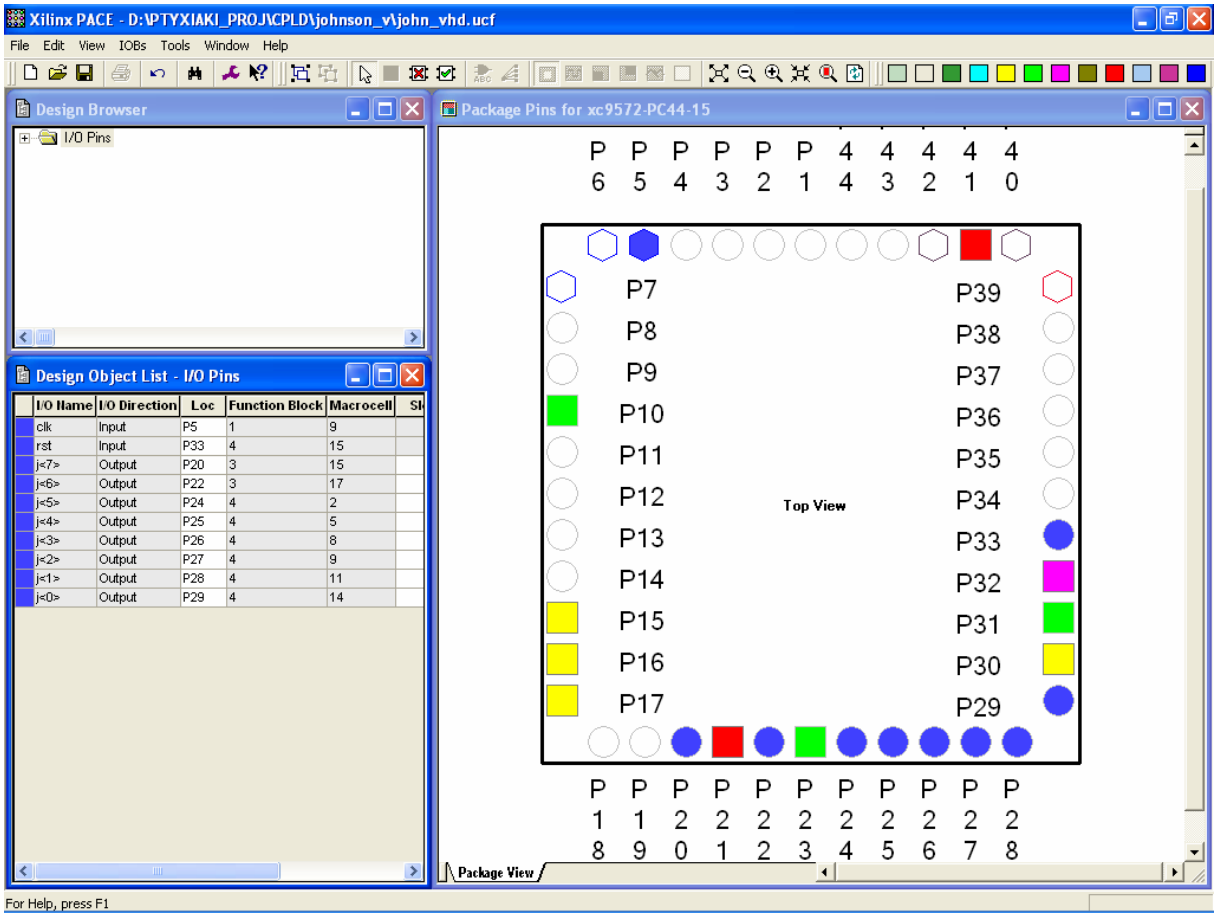
CAPS NUM SCRL LOC VHDL

Εικόνα 2.98 VHDL αρχείο του κυκλώματος

Εικόνα 2.99 Σχηματικό σύμβολο κυκλώματος



Εικόνα 2.100 Παράθυρο διαγράμματος προσομοίωσης (συμπεριφοράς) κυκλώματος



Εικόνα 2.101 Παράθυρο αντιστοίχησης εισόδων/εξόδων (I/O) και ακίδων (pins)

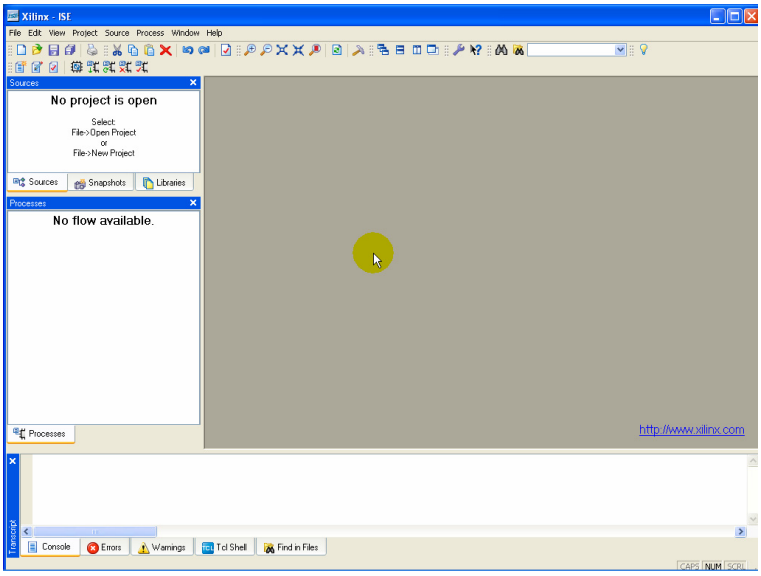
Α. ΕΙΚΟΝΕΣ (ΠΟΛΥΠΛΕΚΤΗΣ 8:1)

Α.1 ΕΙΚΟΝΕΣ ΑΝΑΠΤΥΞΗΣ ΜΕ ΤΗ ΣΧΗΜΑΤΙΚΗ ΓΛΩΣΣΑ

Α.1.1 Έναρξη εφαρμογής

Εικόνα Α.1 Συντόμευση στην επιφάνεια εργασίας του προγράμματος **Xilinx® ISE™ 9.1.03i**

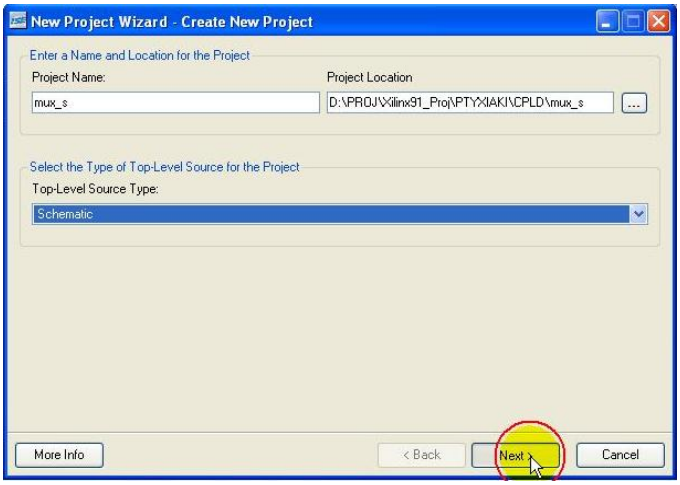
Εικόνα Α.2 Παράθυρο κατά την έναρξη της εφαρμογής **Project Navigator**



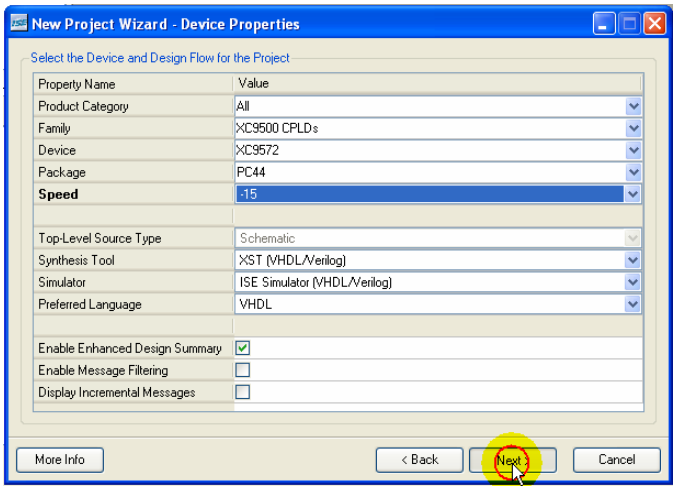
Εικόνα Α.3 Περιβάλλον **Xilinx® ISE™ 9.1.03i** για ανάπτυξη εργασιών (projects)

Α.1.2 Δημιουργία νέας συνολικής εργασίας

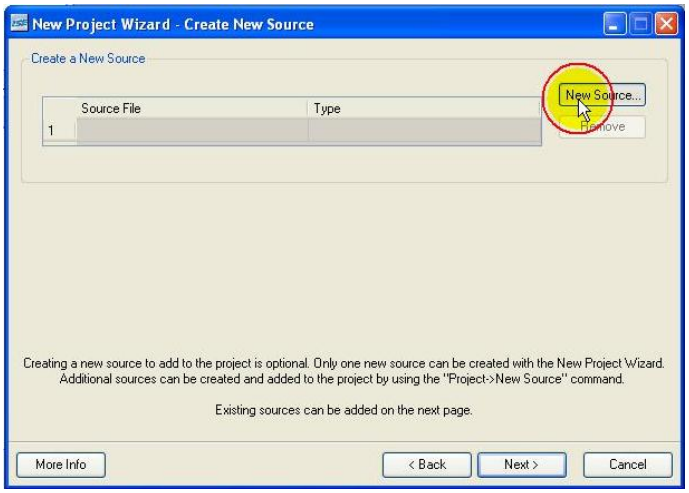
Εικόνα Α.4 Επιλογή διαδρομής **File→New Project**



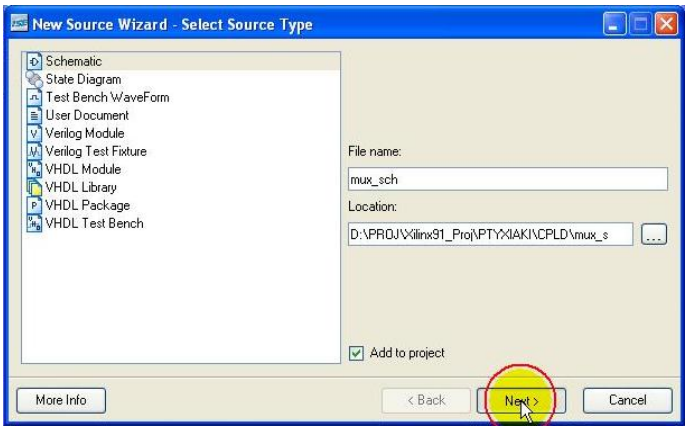
Εικόνα Α.5 Εισαγωγή στοιχείων συνολικής εργασίας (project), επιλογή **Next**



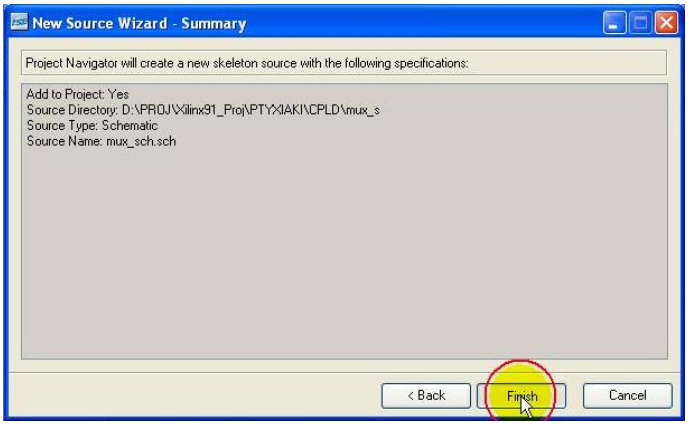
Εικόνα Α.6 Εισαγωγή στοιχείων ολοκληρωμένου και εργαλείων, επιλογή **Next**



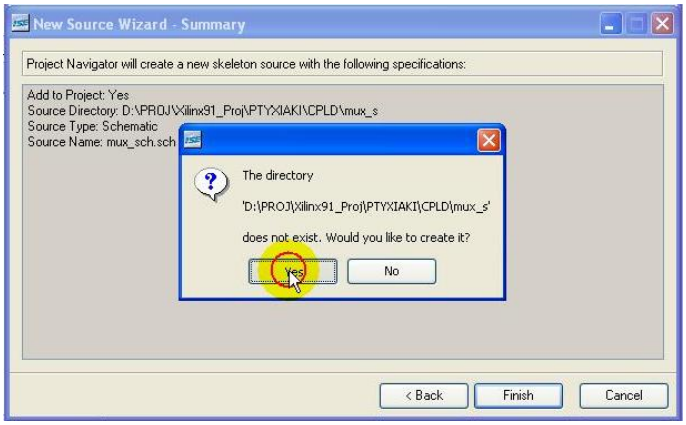
Εικόνα Α.7 Εισαγωγή νέου αρχείου, επιλογή **New Source**



Εικόνα Α.8 Εισαγωγή τύπου και ονόματος του νέου αρχείου, επιλογή **Next**

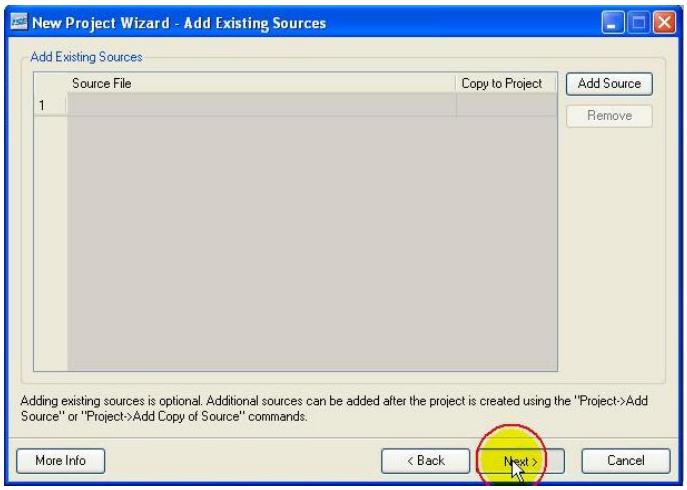


Εικόνα Α.9 Επισκόπηση στοιχείων του νέου αρχείου, επιλογή **Finish**

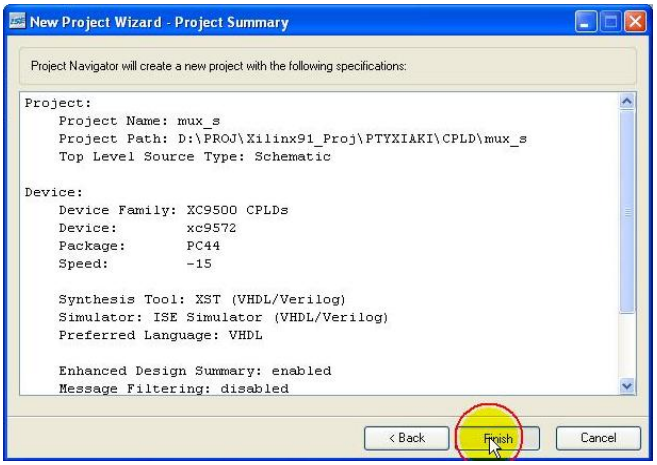


Εικόνα Α.10 Ερώτηση για δημιουργία καταλόγου σε επιλεγμένη διαδρομή, επιλογή **Yes**

Εικόνα Α.11 Εισαγωγή και άλλων αρχείων σχετικών με την εργασία (project), επιλογή **Next**



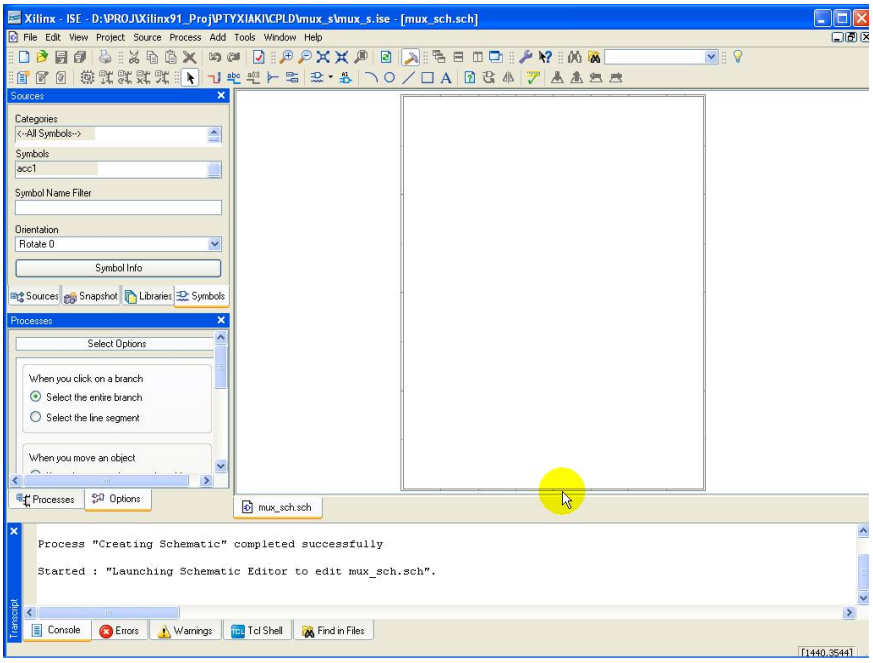
Εικόνα Α.12 Συνέχεια διαδικασίας, επιλογή **Next**



Εικόνα Α.13 Συνολική επισκόπηση της εργασίας (project), επιλογή **Finish**

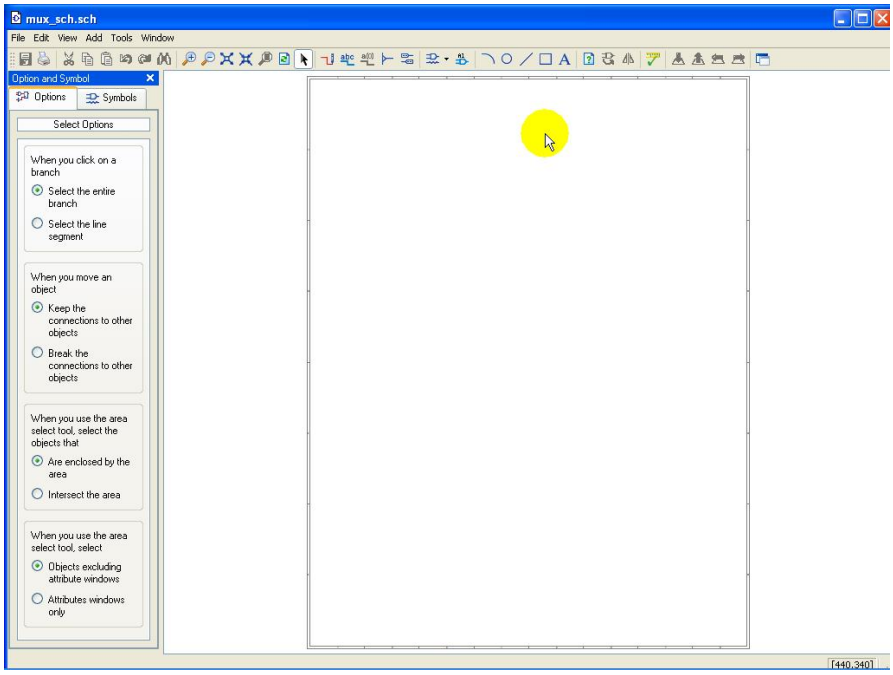
Α.1.3 Νέο σχηματικό

Α.1.3.1 Μεγιστοποίηση επιφάνειας εργασίας



Εικόνα Α.14 Εισαγωγή στο περιβάλλον της εργασίας (project) στη σχηματική γλώσσα

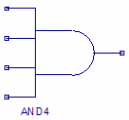
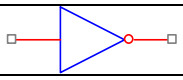
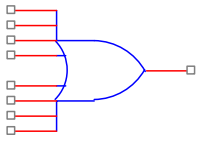
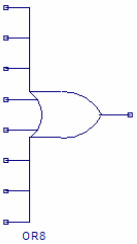
Εικόνα Α.15 Επιλογή εργαλείου **Float this window**



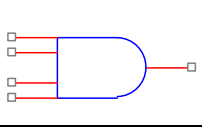
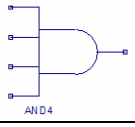
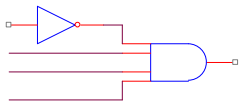
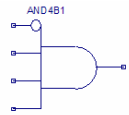
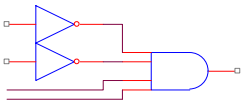
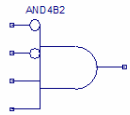
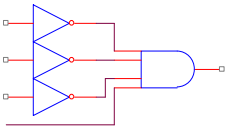
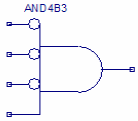
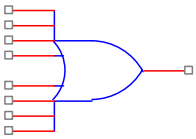
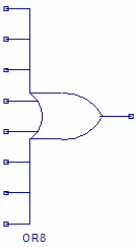
Εικόνα Α.16 Βελτιστοποιημένη επιφάνεια στη σχηματική γλώσσα

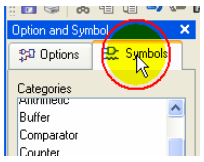
Α.1.3.2. Αντιστοίχηση και τοποθέτηση υλικών

Πίνακας Α.1 Αντιστοίχηση υλικών

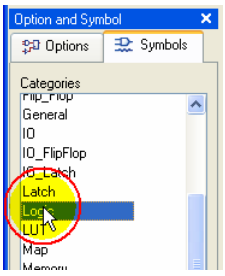
ΣΥΜΒΑΤΙΚΑ ΥΛΙΚΑ ΣΧΕΔΙΑΣΗΣ		ΥΛΙΚΑ ΣΧΕΔΙΑΣΗΣ PROJECT NAVIGATOR		
ΟΝΟΜΑΣΙΑ	ΣΧΗΜΑ	ΣΧΗΜΑ	ΟΝΟΜΑΣΙΑ	ΚΑΤΑΛΟΓΟΣ ΒΙΒΛΙΟΘΗΚΗΣ SYMBOL
ΠΥΛΗ AND 4 ΕΙΣΟΔΩΝ			AND4	LOGIC
ΠΥΛΗ NOT			INV	LOGIC
ΠΥΛΗ OR 8 ΕΙΣΟΔΩΝ			OR8	LOGIC

Πίνακας Α.2 Αντιστοίχηση υλικών για βέλτιστη τοποθέτηση

ΣΥΜΒΑΤΙΚΑ ΥΛΙΚΑ ΣΧΕΔΙΑΣΗΣ		ΥΛΙΚΑ ΣΧΕΔΙΑΣΗΣ PROJECT NAVIGATOR		
ΟΝΟΜΑΣΙΑ	ΣΧΗΜΑ	ΣΧΗΜΑ	ΟΝΟΜΑΣΙΑ	ΚΑΤΑΛΟΓΟΣ ΒΙΒΛΙΟΘΗΚΗΣ SYMBOL
ΠΥΛΗ AND 4 ΕΙΣΟΔΩΝ			AND4	LOGIC
ΠΥΛΗ AND 4 ΕΙΣΟΔΩΝ ΜΕ 1 ΠΥΛΗ NOT ΣΤΗΝ ΕΙΣΟΔΟ			AND4B1	LOGIC
ΠΥΛΗ AND 4 ΕΙΣΟΔΩΝ ΜΕ 2 ΠΥΛΕΣ NOT ΣΤΗΝ ΕΙΣΟΔΟ			AND4B2	LOGIC
ΠΥΛΗ AND 4 ΕΙΣΟΔΩΝ ΜΕ 3 ΠΥΛΕΣ NOT ΣΤΗΝ ΕΙΣΟΔΟ			AND4B3	LOGIC
ΠΥΛΗ OR 8 ΕΙΣΟΔΩΝ			OR8	LOGIC

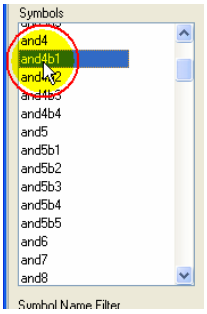


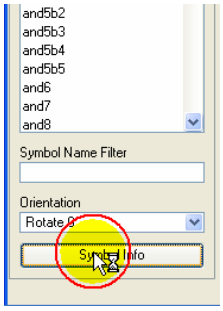
α. Επιλογή **Symbols**.



β. Επιλογή **Logic.**

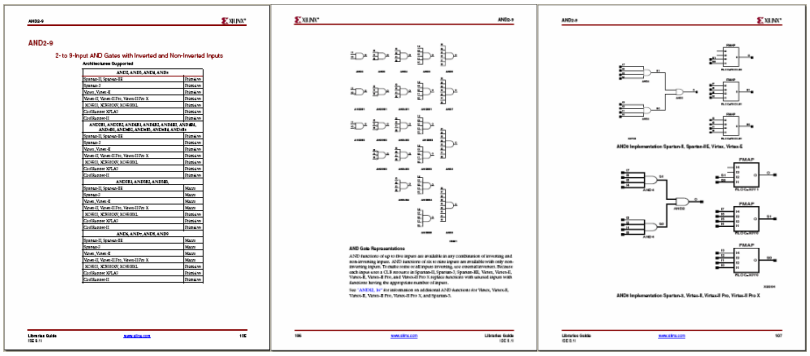
Εικόνα Α.17 Επιλογή διαδρομής **Symbols→Logic,** (**εικόνες Α.17α** ως **Α.17β**)

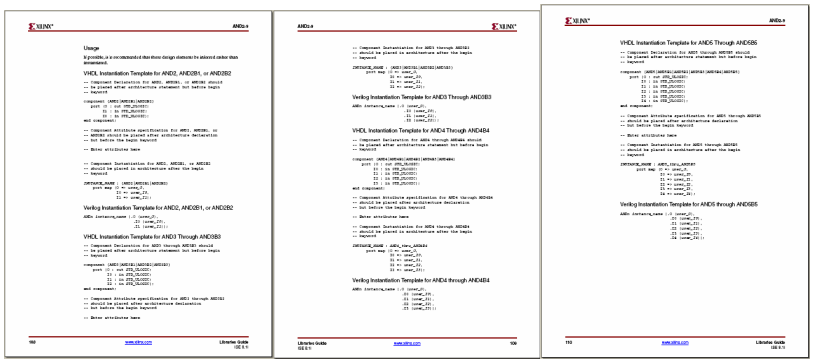




α. Επιλογή υλικού β. Επιλογή **Symbol Info**

Εικόνα Α.18 Παράδειγμα συλλογής πληροφορίας του υλικού **and4b1,** (**εικόνες Α.18α** ως **Α.18β**)

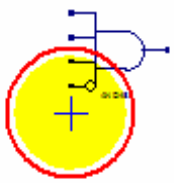




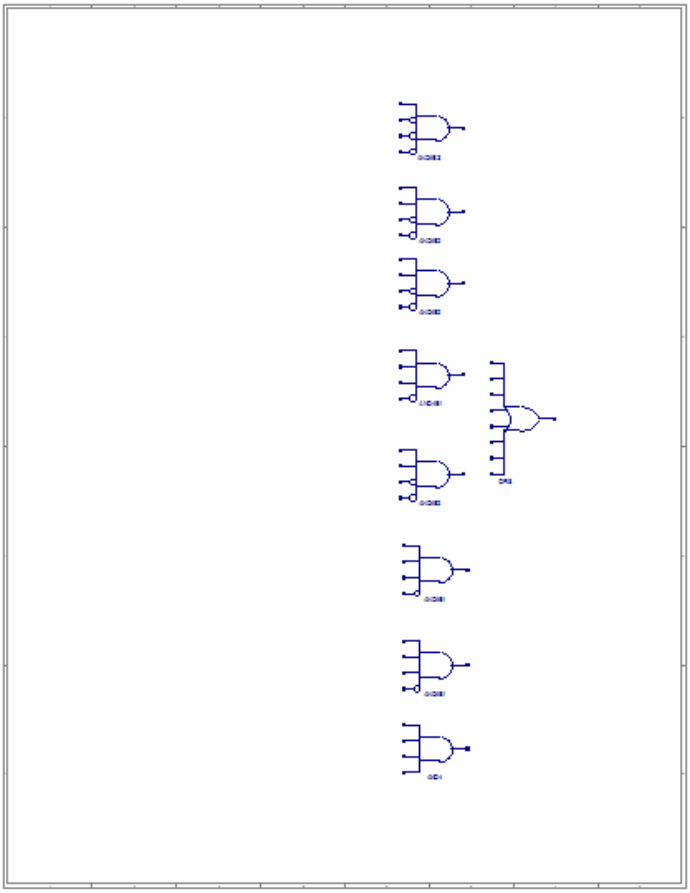
Εικόνα Α.19 Σμίκρυνση σελίδων πληροφορίας του υλικού **and4b1** εντός του **AND2-9**, σε αρχείο μορφής pdf

Πίνακας Α.3 Επιλογή και τοποθέτηση υλικών στην επιφάνεια εργασίας του σχηματικού, (**εικόνες Α.20α** ως **Α.20β**)

α. Επιλεγμένο υλικό προς τοποθέτηση.



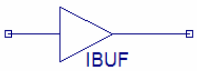
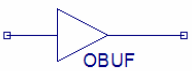
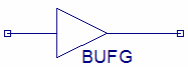
β. Επιλεγμένο υλικό και μόλις τοποθετημένο μετά από απλό κλικ του ποντικιού.

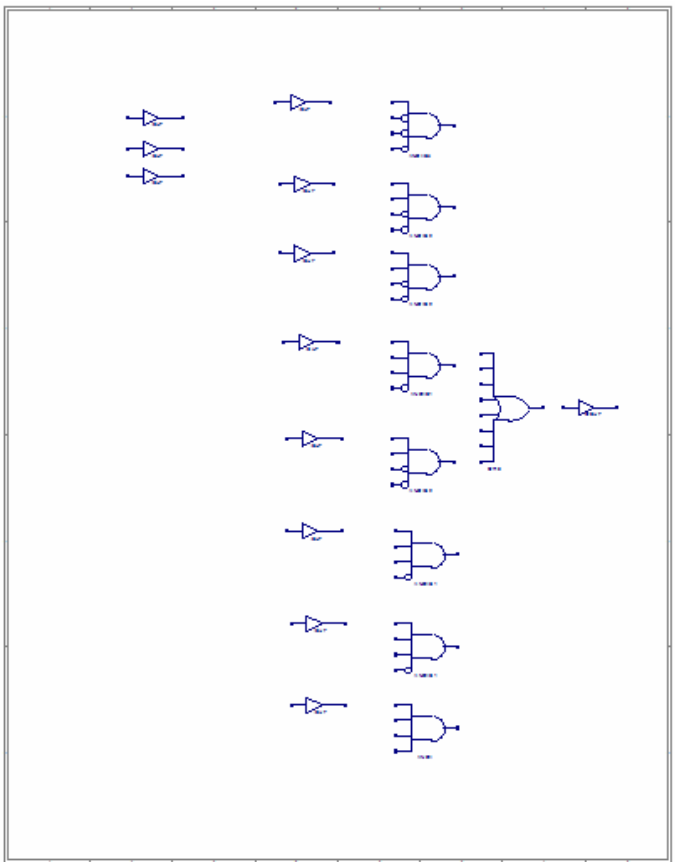


Εικόνα Α.21 Επιφάνεια εργασίας μετά την τοποθέτηση των υλικών

Α.1.3.3 Τοποθέτηση απομονωτών εισόδου/εξόδου (I/O)

Πίνακας Α.4 Απομονωτές εισόδου/εξόδου (I/O)

ΣΧΗΜΑ	ΟΝΟΜΑΣΙΑ	ΚΑΤΑΛΟΓΟΣ ΒΙΒΛΙΟΘΗΚΗΣ SYMBOL	ΧΡΗΣΗ
	IBUF	IO	ΑΠΟΜΟΝΩΤΗΣ ΕΙΣΟΔΟΥ 1 BIT
	OBUF	IO	ΑΠΟΜΟΝΩΤΗΣ ΕΞΟΔΟΥ 1 BIT
	BUFG	BUFFER	ΑΠΟΜΟΝΩΤΗΣ ΕΙΣΟΔΟΥ ΓΙΑ ΤΟ CLOCK

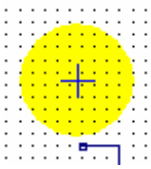


Εικόνα Α.22 Η επιφάνεια εργασίας μετά την τοποθέτηση απομονωτών εισόδου/εξόδου

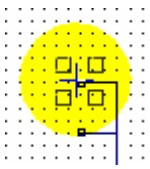
Α.1.3.4 Δημιουργία συνδέσεων

Εικόνα Α.23 Επιλογή εργαλείου **Add Wire** για δημιουργία συνδέσεων

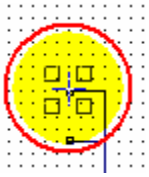
Πίνακας Α.5 Αρχή σύνδεσης από το ένα άκρο, (**εικόνες Α.24α** ως **Α.24γ**)



α. Ο κέρσορας του ποντικιού μετακινείται προς το άκρο της σύνδεσης.

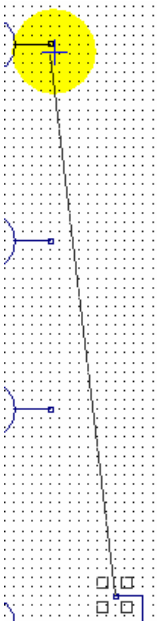


β. Ο κέρσορας του ποντικιού συναντά το άκρο σύνδεσης.

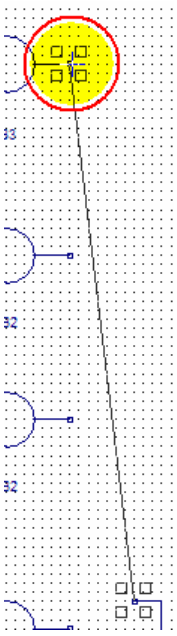


γ. Ο κέρσορας ποντικιού ξεκινά τη σύνδεση με απλό κλικ.

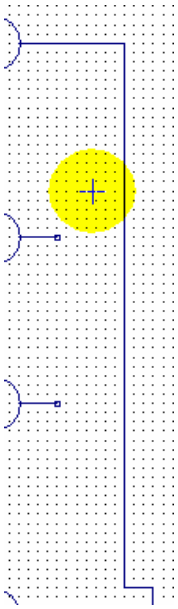
Πίνακας Α.6 Τέλος σύνδεσης στο άλλο άκρο, (**εικόνες Α.25α** ως **Α.25γ**)



α. Κέρσορας ποντικιού προς το άλλο άκρο σύνδεσης.

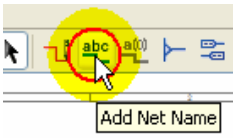


β. Κέρσορας του ποντικιού τελειώνει τη σύνδεση (απλό κλικ).



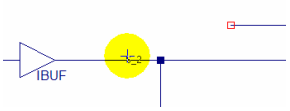
γ. Κέρσορας ποντικιού απομακρύνεται για άλλη σύνδεση.

Α.1.3.4.1 Τοποθέτηση ονομάτων συνδέσεων

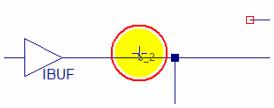


Εικόνα Α.26 Η επιλογή του εργαλείου **Add Net Name** για ονομασία συνδέσεων

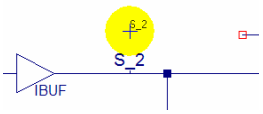
Πίνακας Α.7 Ολοκλήρωση ονομασίας συνδέσεων, (**εικόνες Α.27α** ως **Α.27γ**)



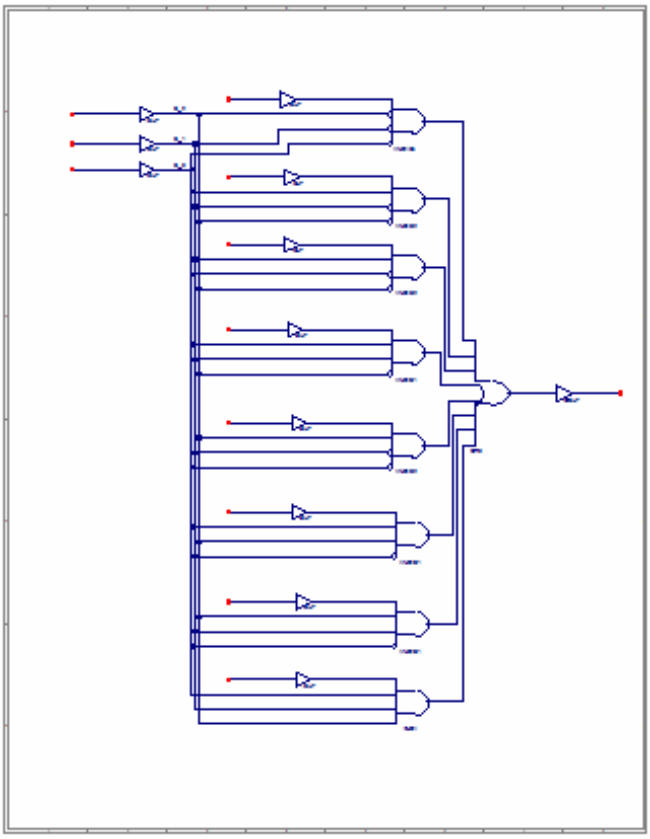
α. Κέρσορας ποντικιού μετά την πληκτρολόγηση «**S_2**» προς το σημείο ονομασίας.



β. Κέρσορας ποντικιού επιλέγει το σημείο ονομασίας (απλό κλικ).

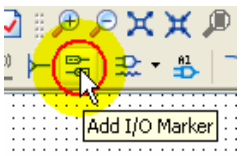


γ. Κέρσορας ποντικιού μετά την εισαγωγή ονόματος απομακρύνεται.



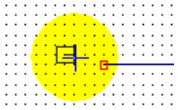
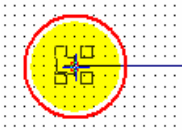
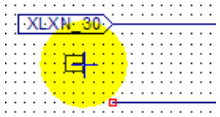
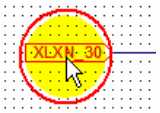
Εικόνα Α.28 Η επιφάνεια εργασίας μετά τη δημιουργία των συνδέσεων και τοποθέτηση των ονομάτων τους

Α.1.3.5 Τοποθέτηση I/O markers

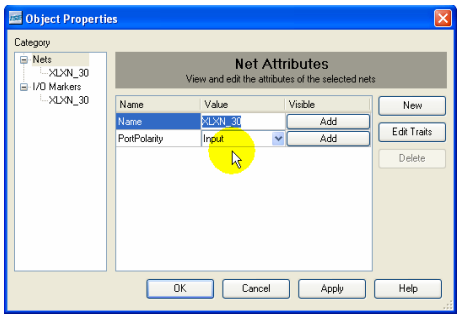


Εικόνα Α.29 Επιλογή εργαλείου τοποθέτησης I/O Marker

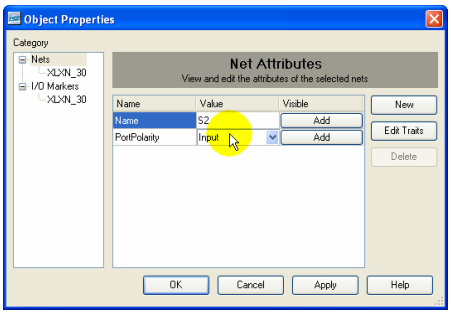
Πίνακας Α.8 Τοποθέτηση I/O Marker σε είσοδο (ή έξοδο), (**εικόνες Α.30α** ως **Α.30δ**)

			
α. Κέρσορας του ποντικιού πλησιάζει την είσοδο.	β. Κέρσορας του ποντικιού επιλέγει την είσοδο (απλό κλικ).	γ. Κέρσορας του ποντικιού απομακρύνεται από την είσοδο μετά την ονομασία ετικέτας από την εφαρμογή.	δ. Κέρσορας του ποντικιού επιλέγει την είσοδο για την αλλαγή σε ονομασία της επιθυμίας μας.

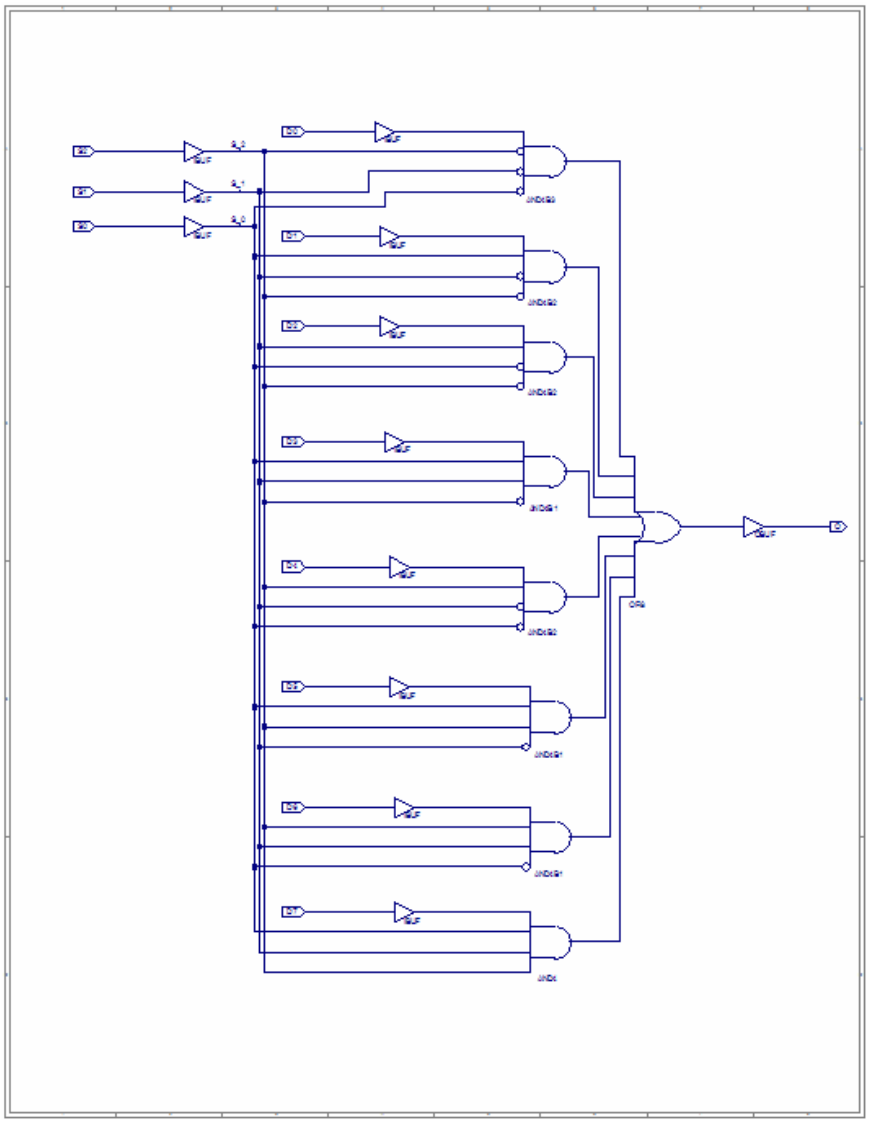
Πίνακας Α.9 Αλλαγή ονομασίας I/O Marker σε είσοδο (ή έξοδο), (**εικόνες Α.31α** ως **Α.31β**)



α. Επιλογή ονόματος που θέλουμε να αλλάξουμε.



β. Πληκτρολόγηση ονόματος της επιλογής μας.

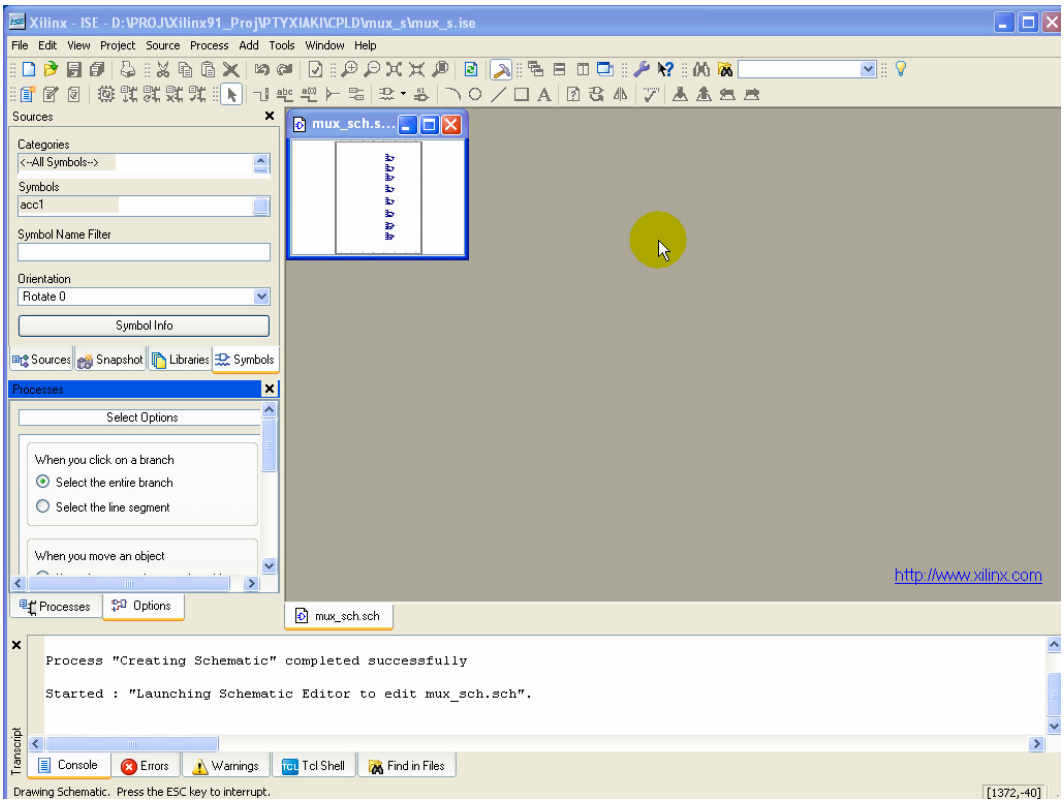


Εικόνα Α.32 Η επιφάνεια εργασίας μετά την τοποθέτηση I/O Marker, (τελικό στάδιο)

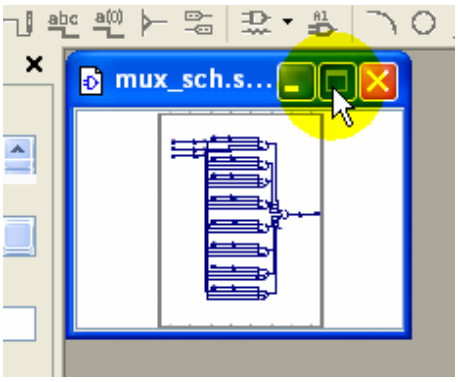
Α.1.3.6 Αποθήκευση και έλεγχος λαθών

Εικόνα Α.33 Η επιλογή του εργαλείου **Save**

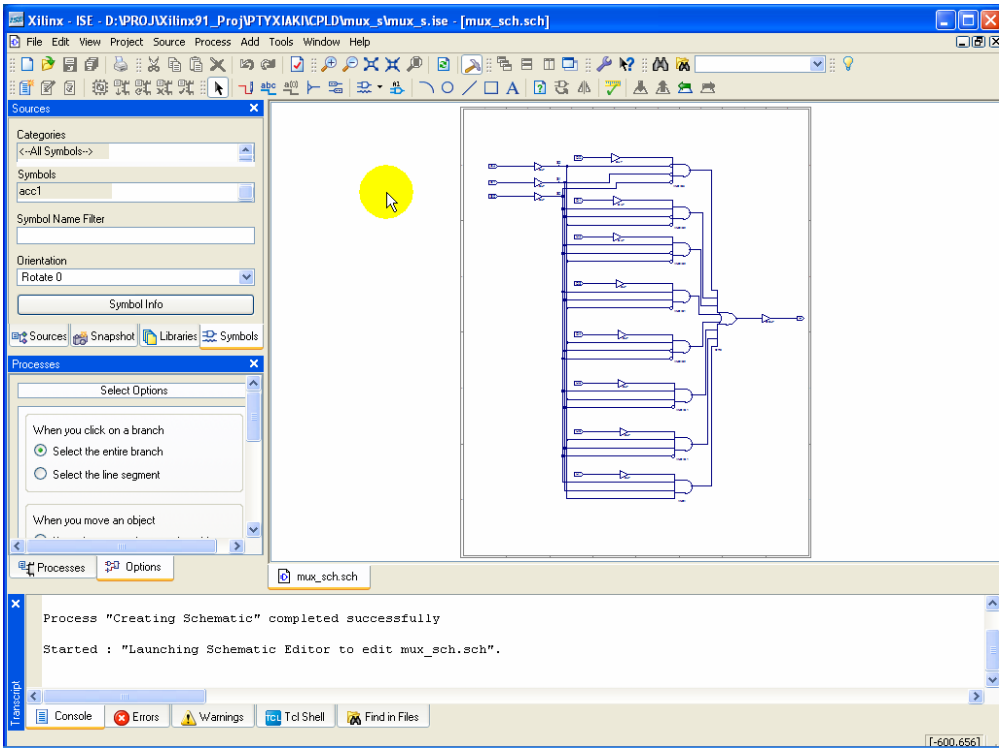
Εικόνα Α.34 Η επιλογή του εργαλείου **Dock this window**



Εικόνα Α.35 Μετάβαση του σχηματικού στο ίδιο παραθυρικό περιβάλλον με την υπόλοιπη εργασία (project)

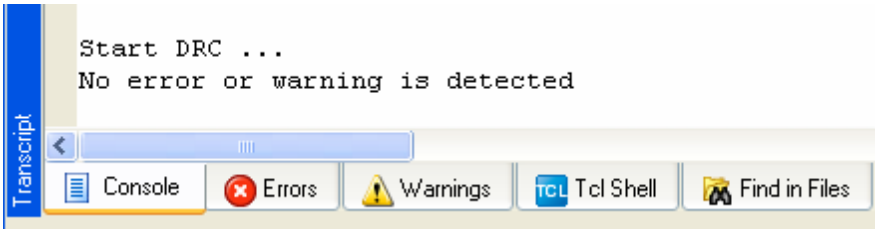


Εικόνα Α.36 Επιλογή μεγιστοποίησης παραθύρου του σχηματικού



Εικόνα Α.37 Η εξέλιξη μετά την τελευταία επιλογή στην επιφάνεια εργασίας

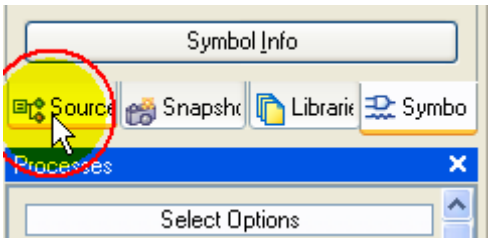
Εικόνα Α.38 Η επιλογή **Check Schematic**



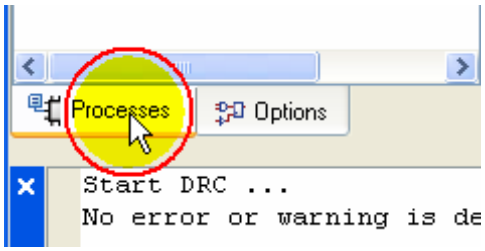
Εικόνα Α.39 Καταγραφή μηνύματος μη εύρεσης λαθών

Α.1.3.7 Δημιουργία σχηματικού συμβόλου

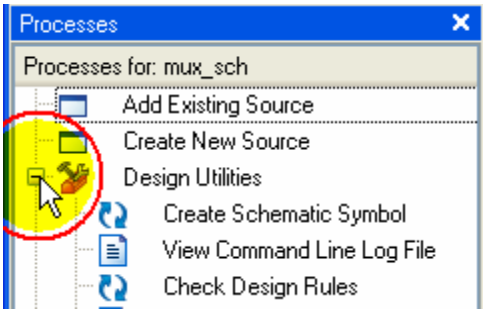
Πίνακας Α.10 Διαδικασία **Create Schematic Symbol**, (**εικόνες Α.40α** ως **Α.40δ**)



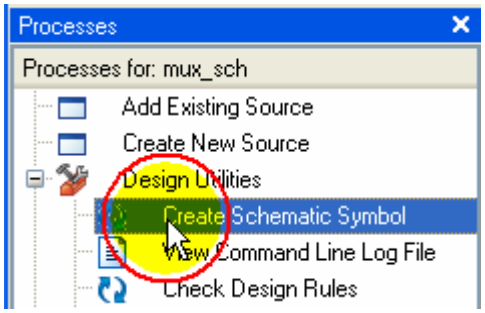
α. Η επιλογή της ετικέτας **Source**.



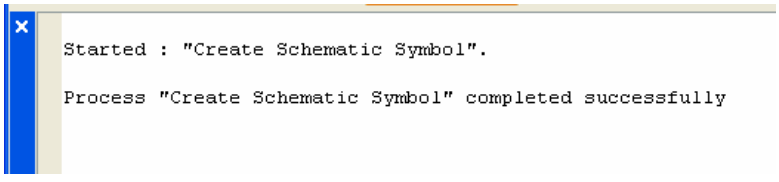
β. Η επιλογή της ετικέτας **Processes**.



γ. Η επιλογή με απλό κλικ του ποντικιού στο «+» δίπλα στο εικονίδιο **Design Utilities**.

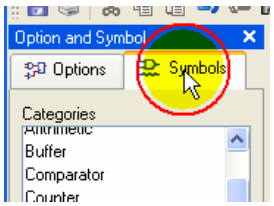


δ. Η επιλογή με διπλό κλικ του ποντικιού για ενεργοποίηση της εφαρμογής **Create Schematic Symbol**.

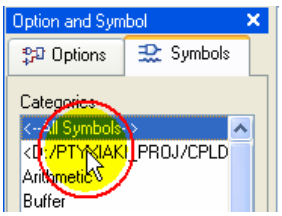


Εικόνα Α.41 Μήνυμα επιτυχούς δημιουργίας σχηματικού συμβόλου

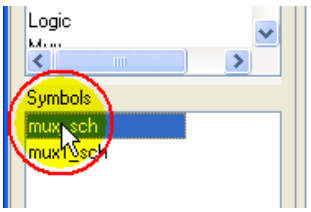
Πίνακας Α.11 Επιλογή και τοποθέτηση σχηματικού συμβόλου, (**εικόνες Α.42α** ως Α.**42στ**)



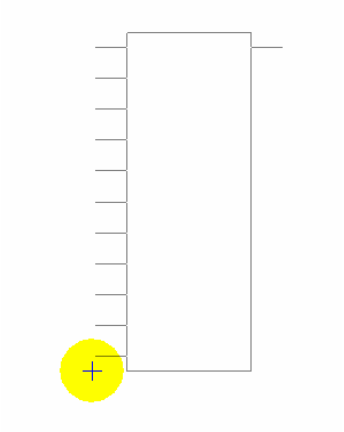
α. Επιλογή **Symbols**.



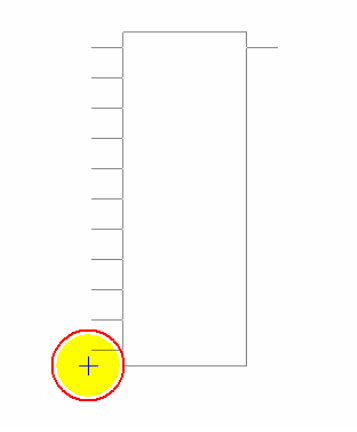
β. Επιλογή **διαδρομής συμβόλου**.



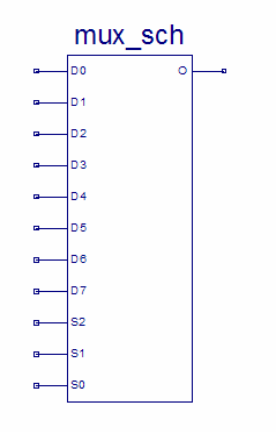
γ. Επιλογή του σχηματικού συμβόλου.



δ. Επιλεγμένο το σχηματικό σύμβολο προς τοποθέτηση.



ε. Επιλεγμένο το σχηματικό σύμβολο και μόλις τοποθετημένο μετά από απλό κλικ του ποντικιού.

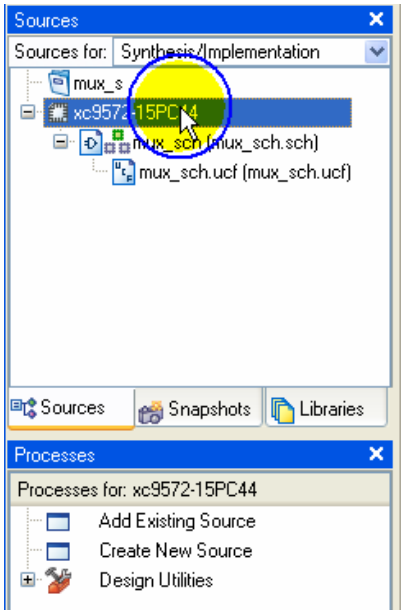


στ. Τέλος τοποθέτησης.

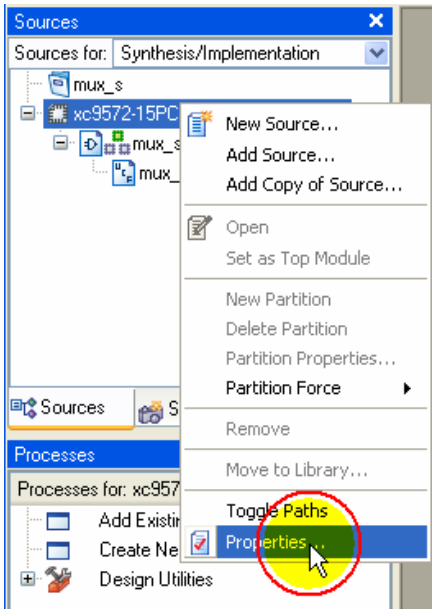
Α.1.4 Προσομοίωση συμπεριφοράς (με ISE Simulator)

Α.1.4.1 Ορισμός προσομοιωτή

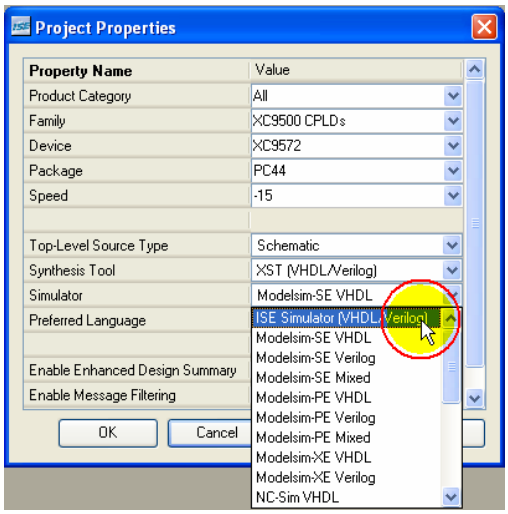
Πίνακας Α.12 Ορισμός **ISE Simulator (VHDL/Verilog)**, (**εικόνες Α.43α** ως **Α.43δ**)



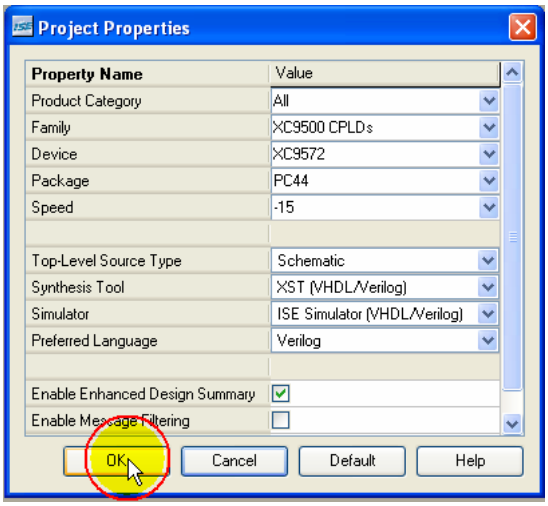
α. Με **δεξί** κλικ του ποντικιού επιλέγουμε στην ετικέτα **Sources** το ολοκληρωμένο (**xc957215PC44**).



β. Με κλικ του ποντικιού επιλέγουμε **Properties**.



γ. Με το άνοιγμα του παράθυρου **Project Properties** στην περιοχή **Simulator:** επιλέγουμε **ISE Simulator (VHDL/Verilog)**.

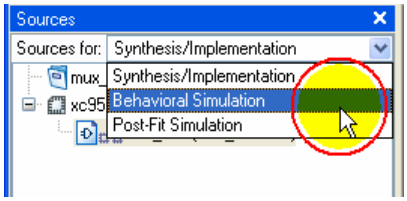


δ. Με κλικ του ποντικιού επιλέγουμε **OK** για επιβεβαίωση αλλαγών.

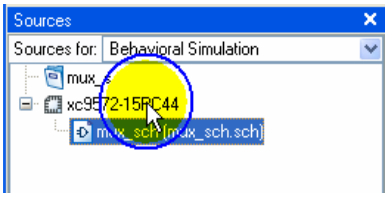
Α.1.4.2 Κυματομορφή ελέγχου

Α.1.4.2.1 Δημιουργία αρχείου κυματομορφής

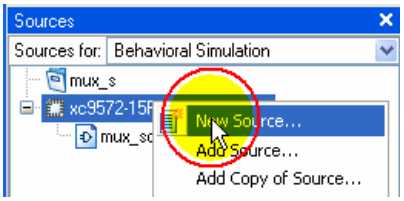
Πίνακας Α.13 Δημιουργία του αρχείου κυματομορφής ελέγχου, (**εικόνες Α.44α** ως **Α.44ζ**)



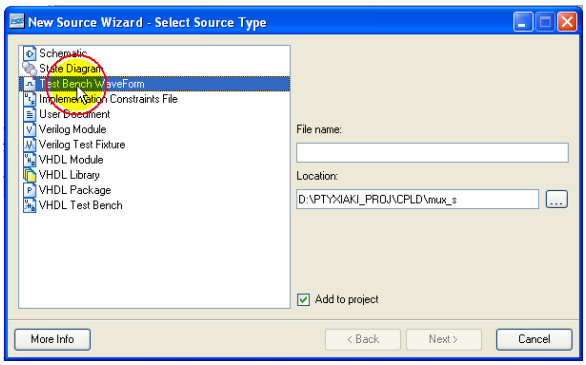
α. Με κλικ του ποντικιού επιλέγουμε **Behavioral Simulation**.



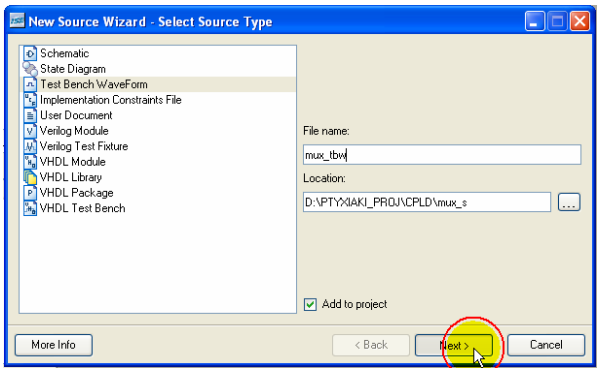
β. Με **δεξί** κλικ του ποντικιού επιλέγουμε το ολοκληρωμένο (**xc9572-15PC44**).



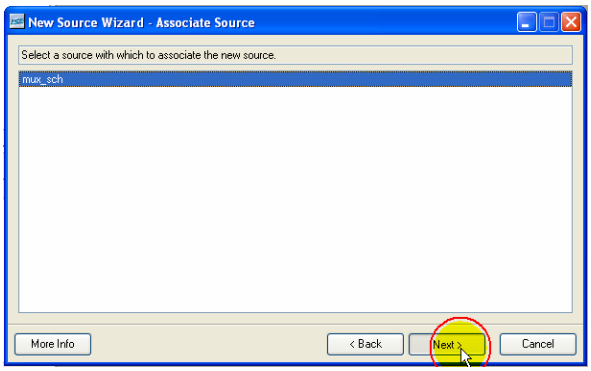
γ. Μετά με κλικ του ποντικιού επιλέγουμε **New Source…**



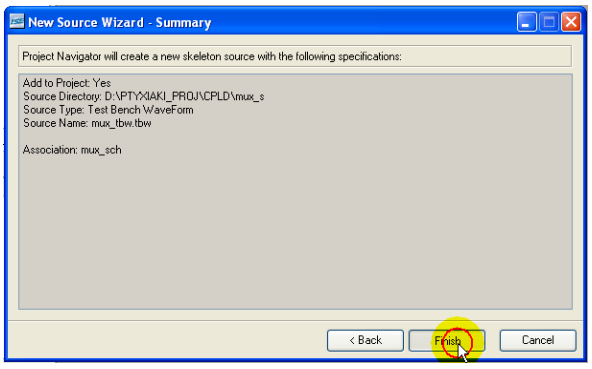
δ. Με κλικ του ποντικιού επιλέγουμε **Test Bench Waveform**.



ε. Πληκτρολογούμε **mux_tbw** και μετά με κλικ του ποντικιού επιλέγουμε **Next**.



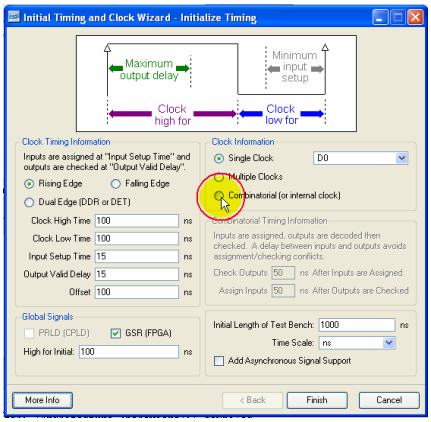
στ. Επανεπιλέγουμε με κλικ του ποντικιού το κουμπί **Next**.



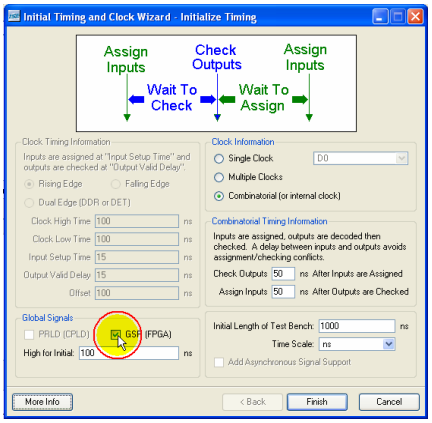
ζ. Τελικά με κλικ του ποντικιού επιλέγουμε το κουμπί **Finish**.

Α.1.4.2.2 Χρονική αρχικοποίηση

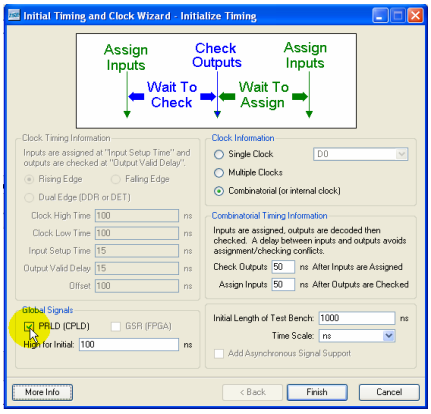
Πίνακας Α.14 Διαδικασία αρχικοποίησης της κυματομορφής ελέγχου, (**εικόνες Α.45α** ως **Α.45ε**)



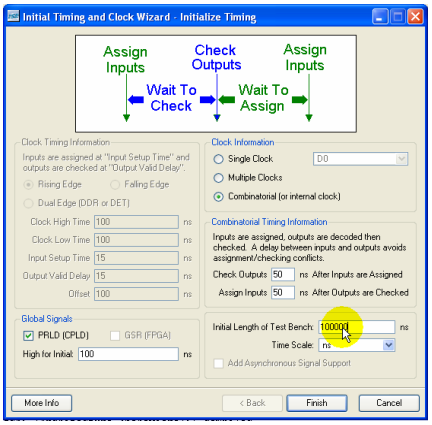
α. Με κλικ του ποντικιού επιλέγουμε **Combinatorial (or internal clock)**.



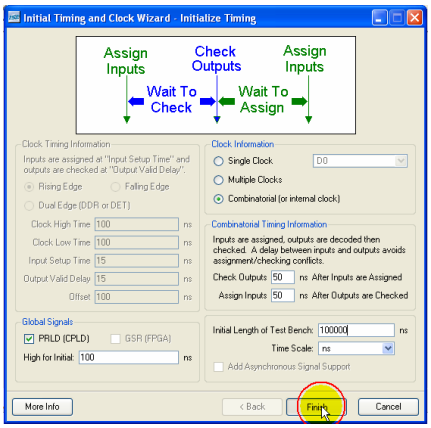
β. Με κλικ του ποντικιού αποεπιλέγουμε **GSR (FPGA)**.



γ. Με κλικ του ποντικιού επιλέγουμε **PRLD (CPLD)**.



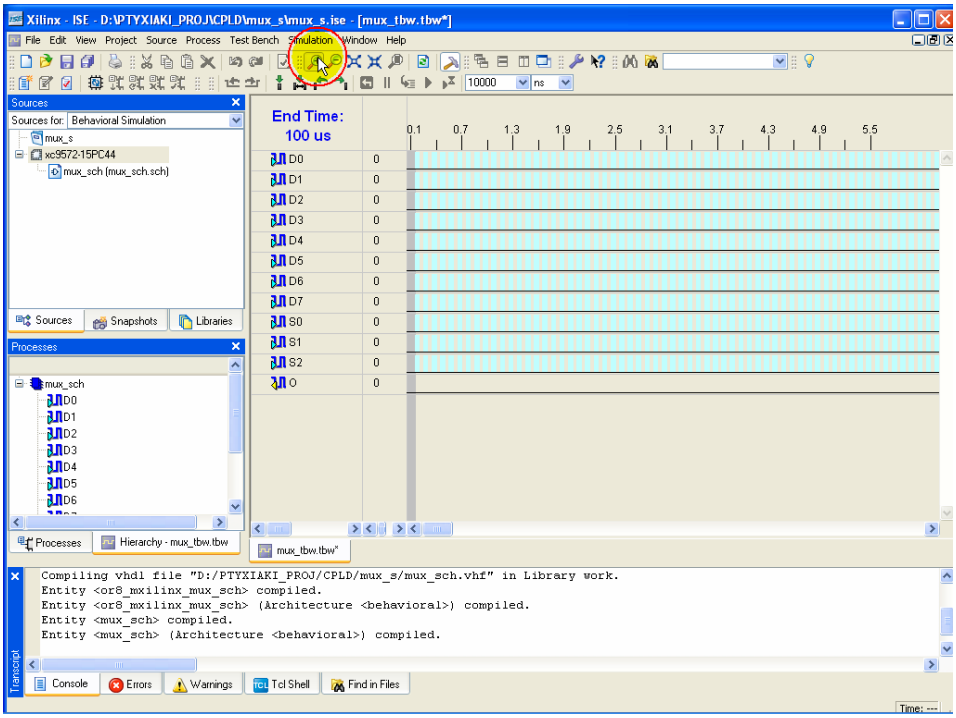
δ. Πληκτρολόγηση χρονικής τιμής στο πλαίσιο της περιοχής **Initial Length of Test Bench**.



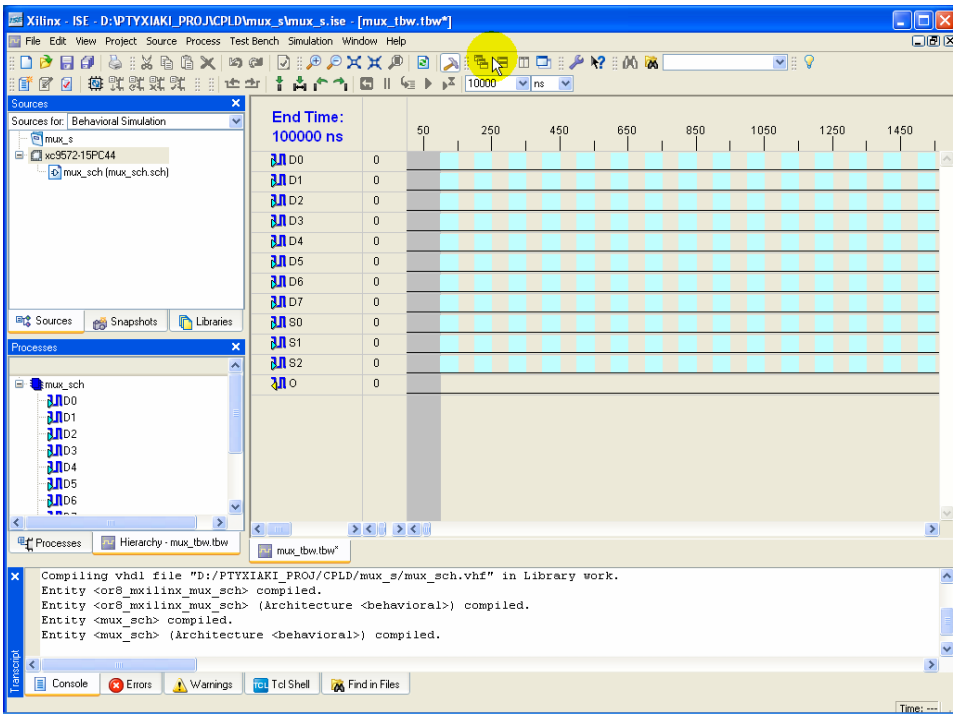
ε. Τελικά με κλικ του ποντικιού επιλέγουμε το κουμπί **Finish**.

Α.1.4.2.3 Ρύθμιση κυματομορφής

Πίνακας Α.15 Ρύθμιση επιφάνειας εργασίας, (**εικόνες Α.46α** ως **Α.46β**)

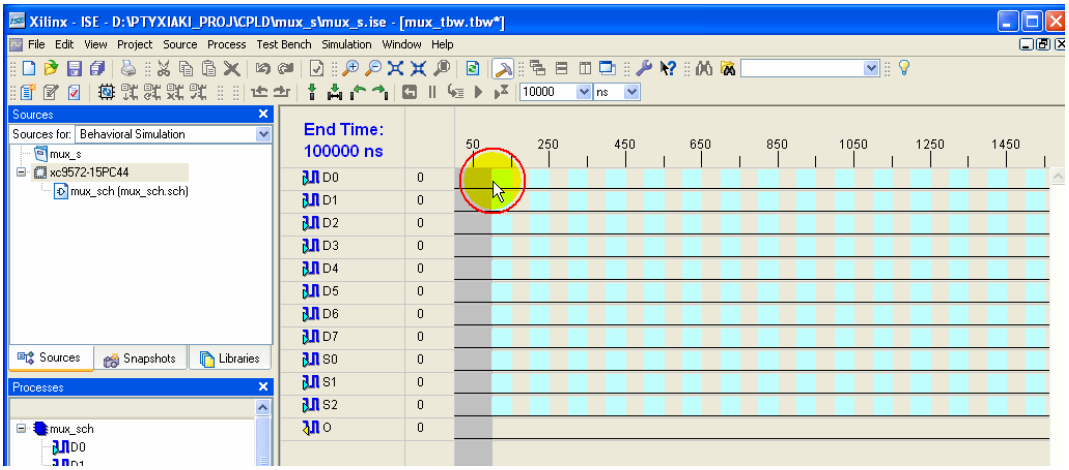


α. Επιλογή εργαλείου **Zoom In**.

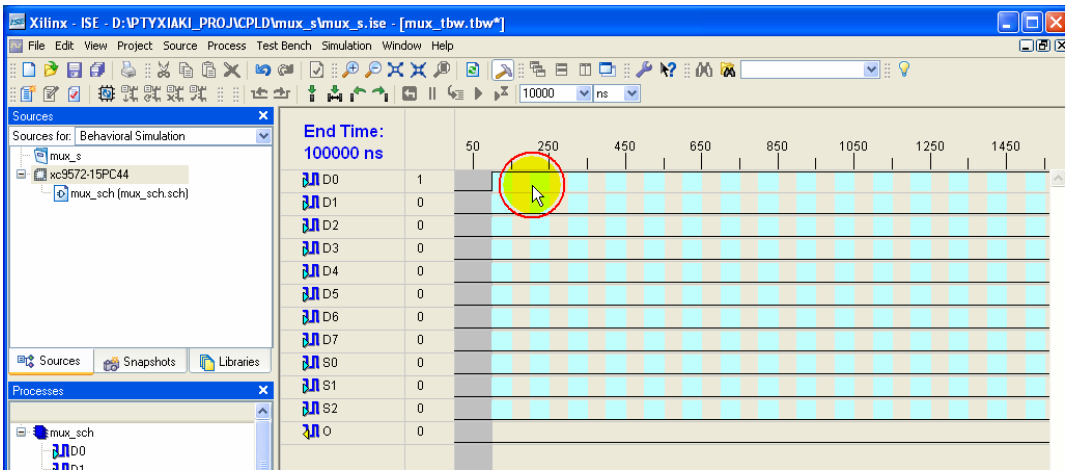


β. Επιλεγμένη μεγέθυνση της επιφάνειας εργασίας.

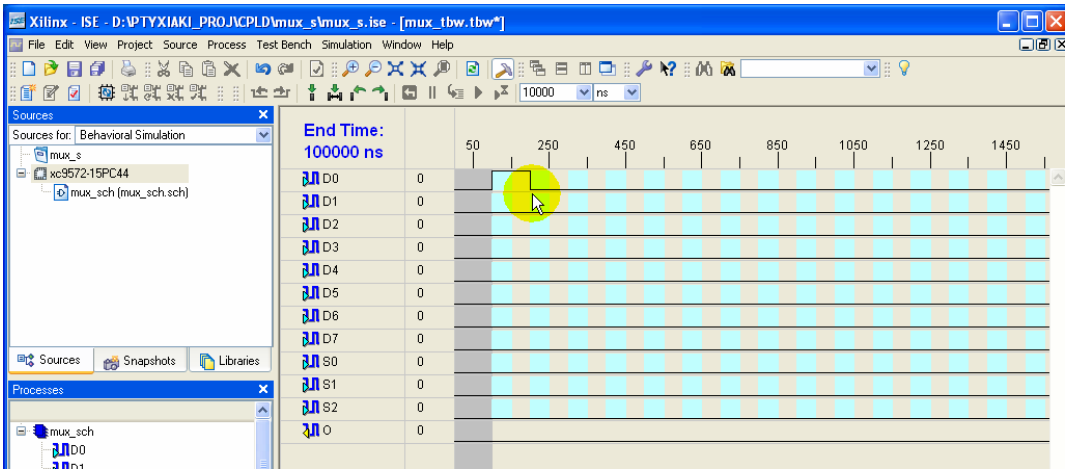
Πίνακας Α.16 Τοποθέτηση αλλαγής λογικής στάθμης μεταξύ επιλεγμένων χρονικών σημείων, (**εικόνες Α.47α** ως **Α.47γ**)



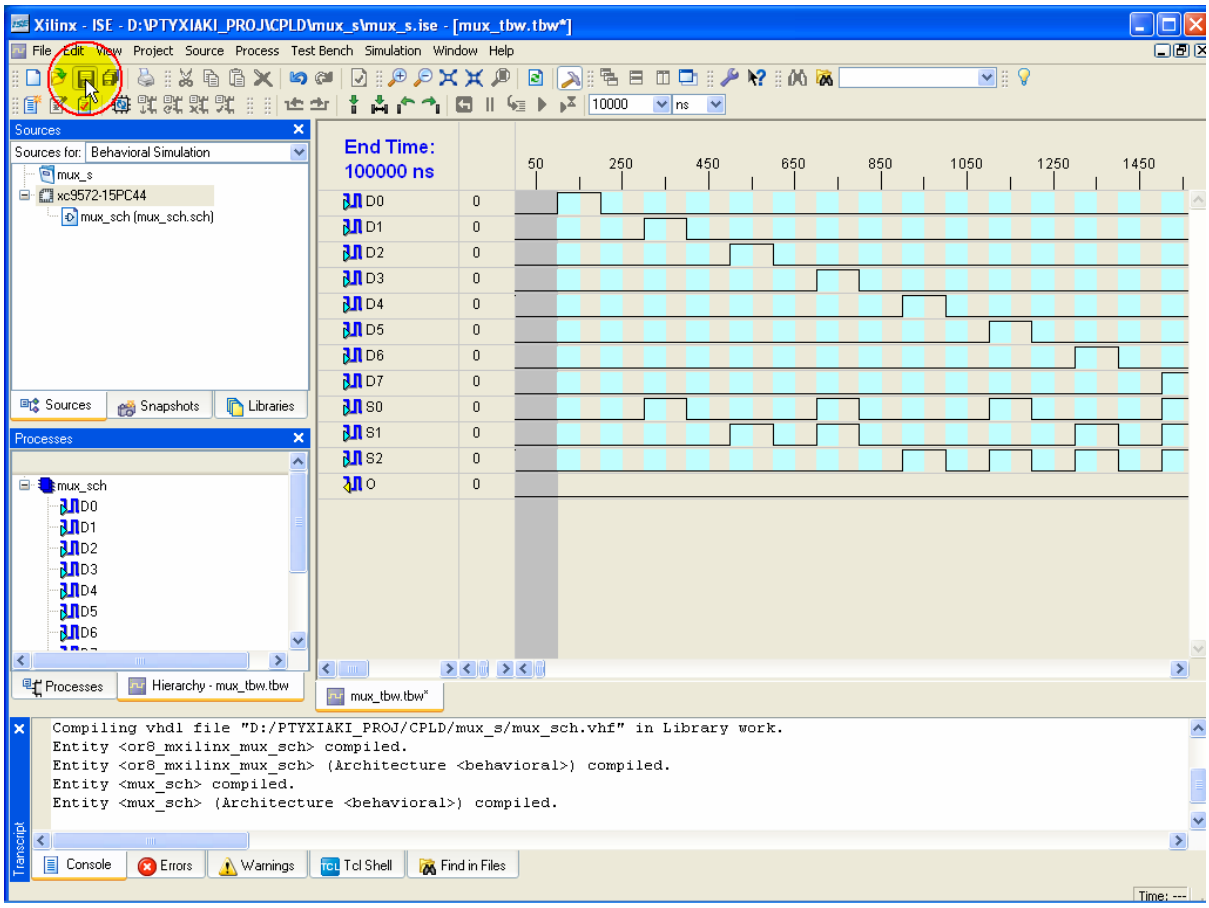
α. Με κλικ του ποντικιού επιλέγουμε το χρονικό σημείο αλλαγής λογικής στάθμης.



β. Επανάληψη προηγούμενης διαδικασίας.



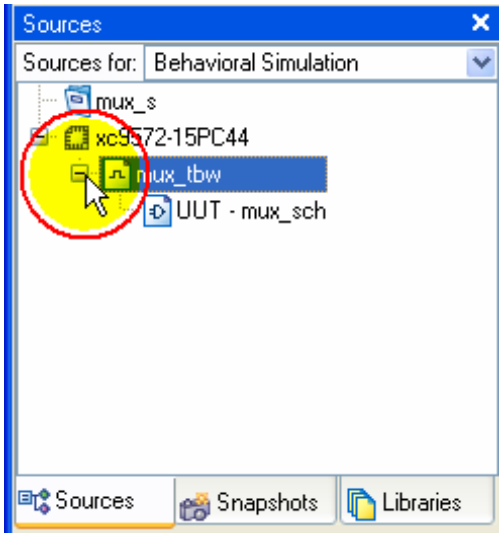
γ. Αποτελέσματα των δύο τελευταίων ενεργειών.



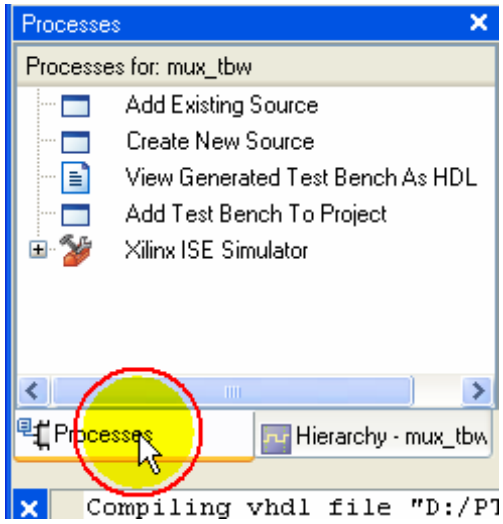
Εικόνα Α.48 Επιλογή του εργαλείου **Save** μετά τις τοποθετήσεις αλλαγών στάθμης

Α.1.4.2.4 Ενεργοποίηση προσομοίωσης

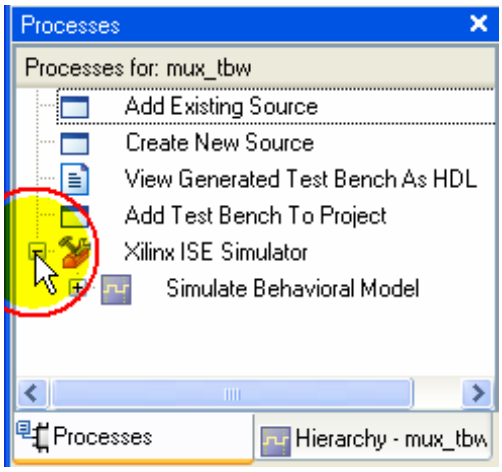
Πίνακας Α.17 Ενεργοποίηση προσομοίωσης, (**εικόνες Α.49α** ως **Α.49δ**)



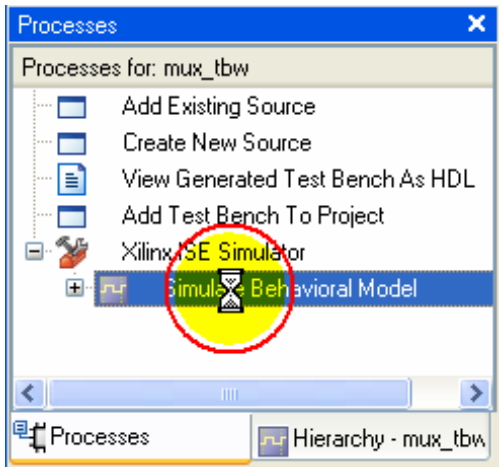
α. Η επιλογή με απλό κλικ του ποντικιού στο «+» δίπλα στο αρχείο κυματομορφής και μετά με κλικ του ποντικιού επιλέγουμε το όνομα αρχείου κυματομορφής **mux_tbw**.



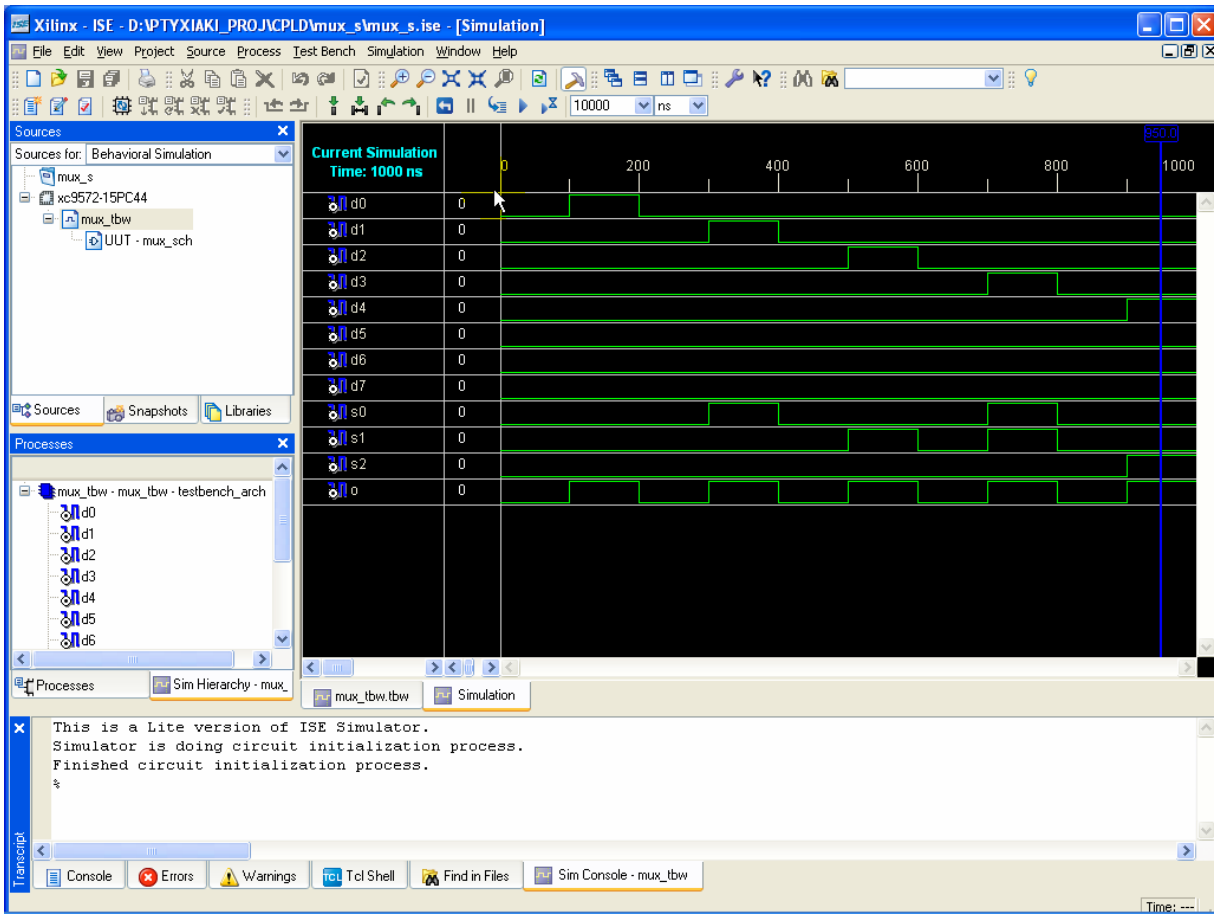
β. Με κλικ του ποντικιού επιλέγουμε την ετικέτα **Processes**.



γ. Η επιλογή με απλό κλικ του ποντικιού στο «+» δίπλα στο εικονίδιο **Xilinx ISE Simulator**.



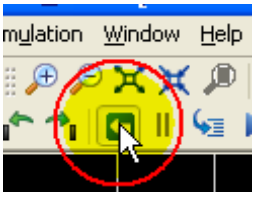
δ. Με **διπλό** κλικ του ποντικιού επιλέγουμε **Simulate Behavioral Model**.



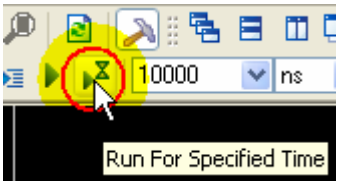
Εικόνα Α.50 Παράθυρο ενεργοποιημένης προσομοίωσης

Α.1.4.2.5 Ενεργοποιημένη προσομοίωση

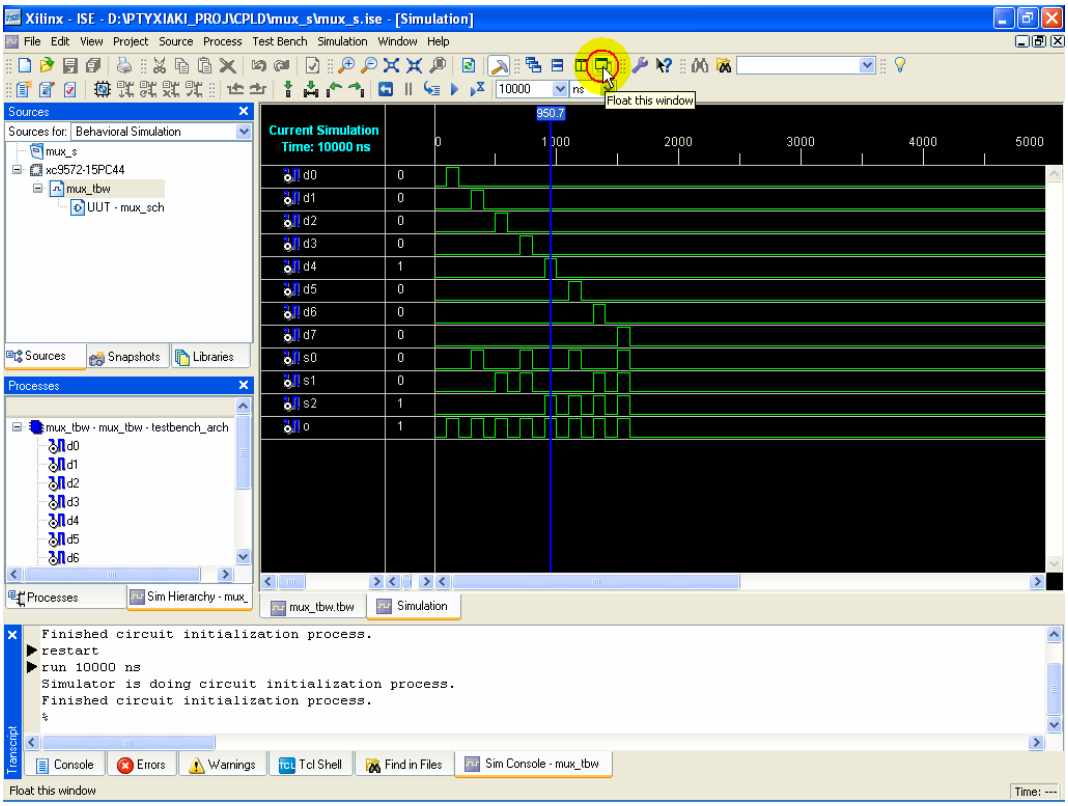
Πίνακας Α.18 Ρύθμιση ενεργοποίησης προσομοίωσης, (**εικόνες Α.51α** ως **Α.51β**)



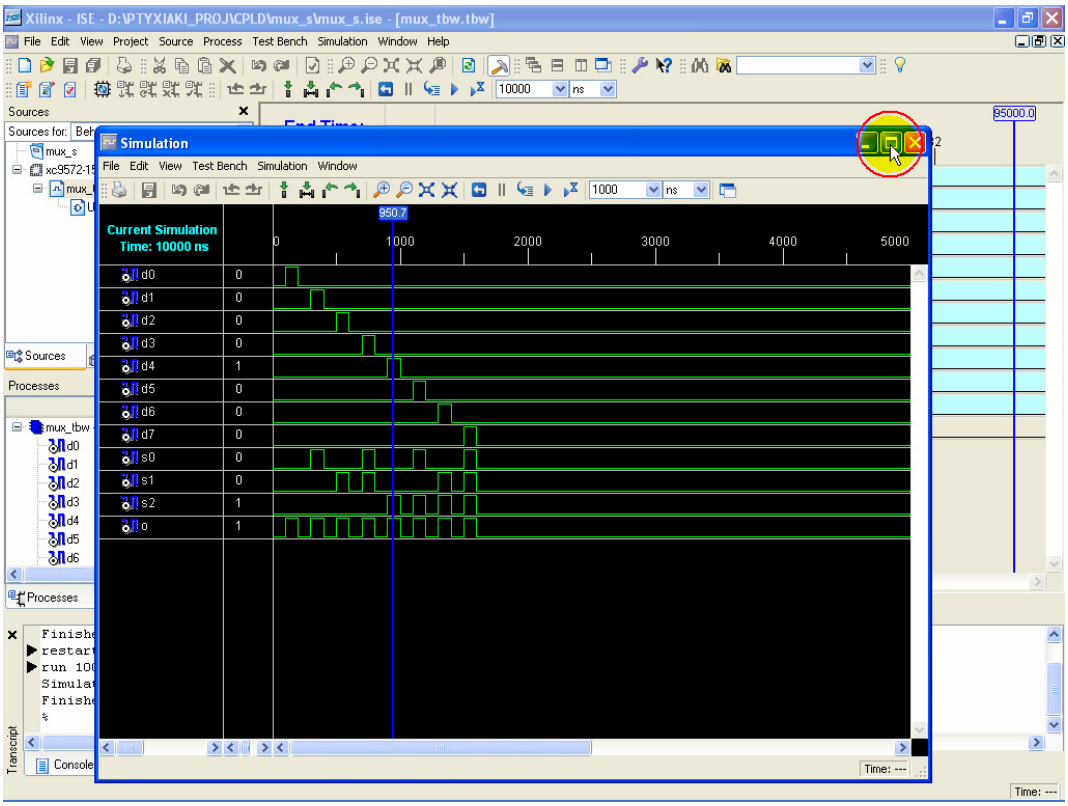
α. Η επιλογή με απλό κλικ του ποντικιού στο εργαλείο για επανεκκίνηση της προσομοίωσης **Restart Simulation**.



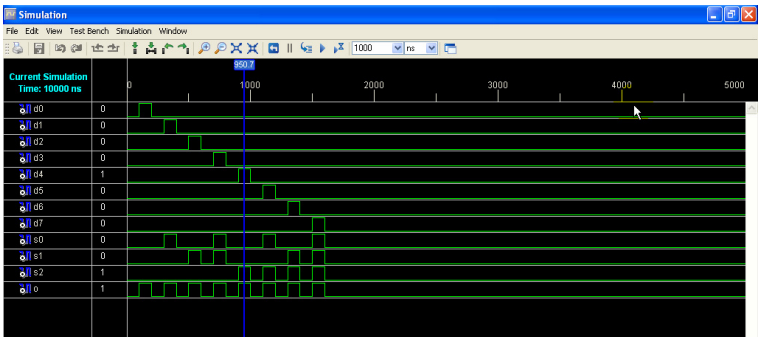
β. Η επιλογή με απλό κλικ του ποντικιού στο εργαλείο για ενεργοποίηση της προσομοίωσης για επιλεγμένο χρονικό διάστημα **Run For Specific Time**.



Εικόνα Α.52 Παράθυρο νέας προσομοίωσης με επιλογή εργαλείου **Float this window**

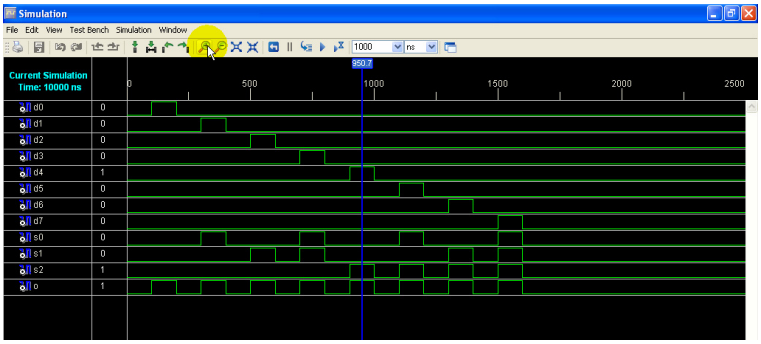


Εικόνα Α.53 Επιλογή μεγέθυνσης απομονωμένου παραθύρου προσομοίωσης

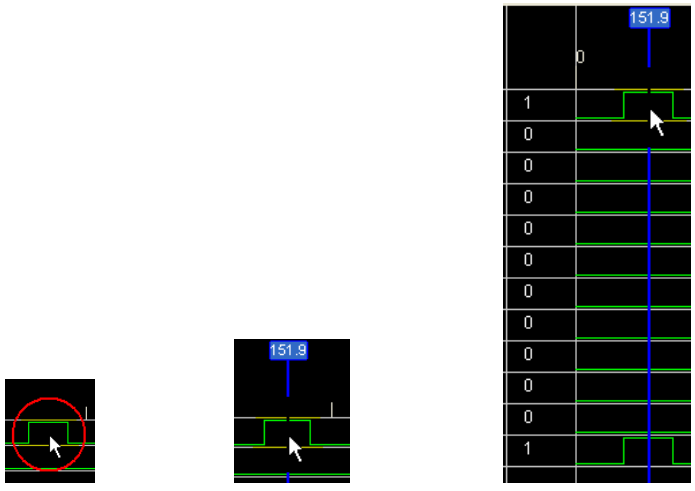


Εικόνα Α.54 Αποτέλεσμα μεγέθυνσης απομονωμένου παραθύρου προσομοίωσης

Εικόνα Α.55 Επιλογή εργαλείου **Zoom In** για παραπάνω μεγέθυνση

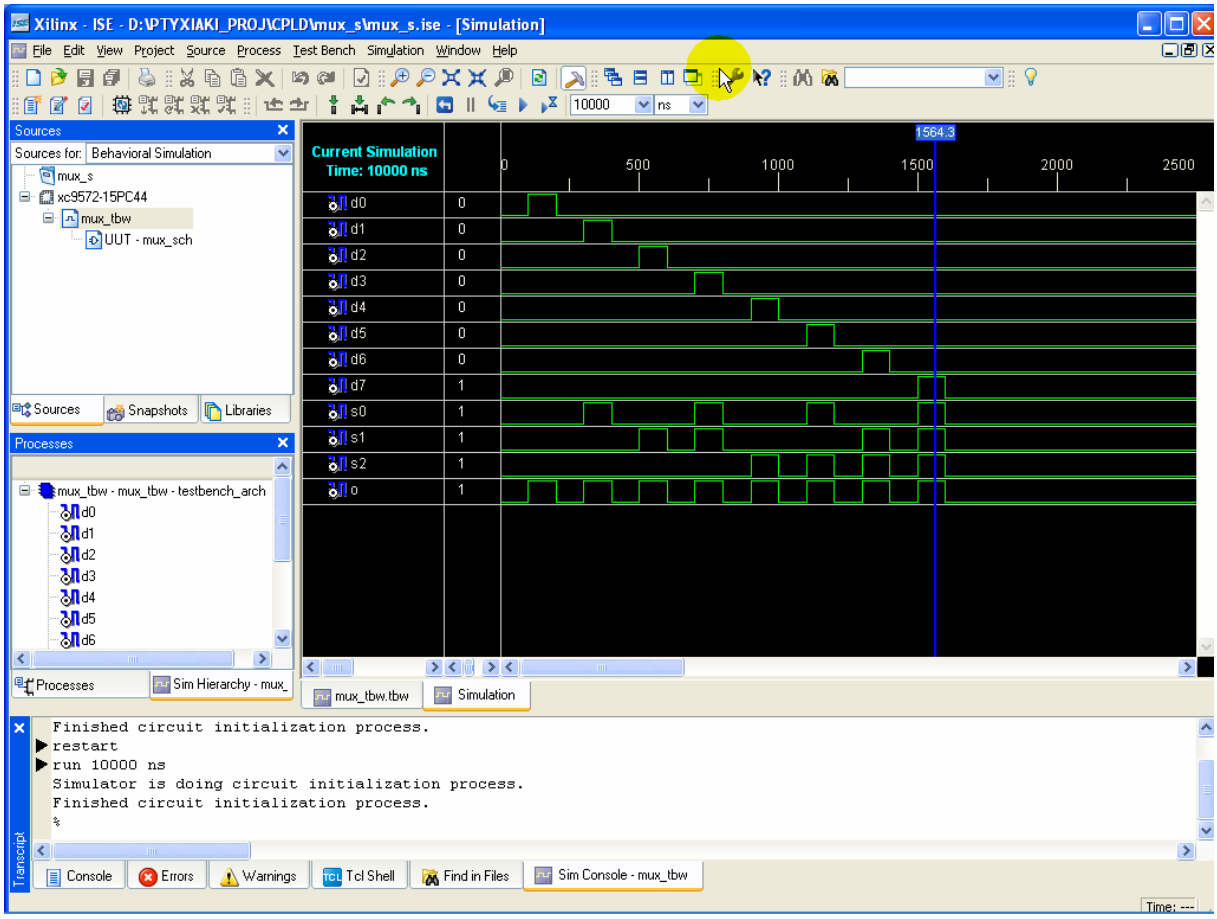


Εικόνα Α.56 Τελευταία μεγέθυνση απομονωμένου παραθύρου προσομοίωσης



Εικόνα Α.57 Επιλεγμένο (με κλικ του ποντικιού) χρονικό σημείο για έλεγχο της ορθότητας λογικών σταθμών σύμφωνα με τη σχεδίαση

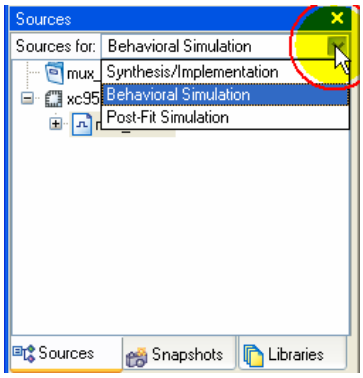
Εικόνα Α.58 Η επιλογή με απλό κλικ του ποντικιού στο εργαλείο **Dock this window**



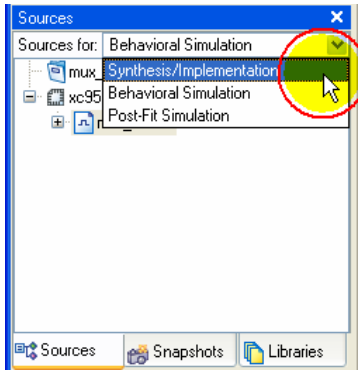
Εικόνα Α.59 Επιστροφή παραθύρου προσομοίωσης στο αντίστοιχο της συνολικής εργασίας (project)

Α.1.5 Προγραμματισμός

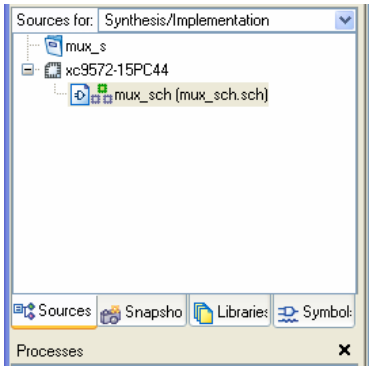
Α.1.5.1 Αντιστοίχιση ακίδων



α. Επιλογή (κλικ του ποντικιού) για άνοιγμα κυλιόμενου παραθύρου.

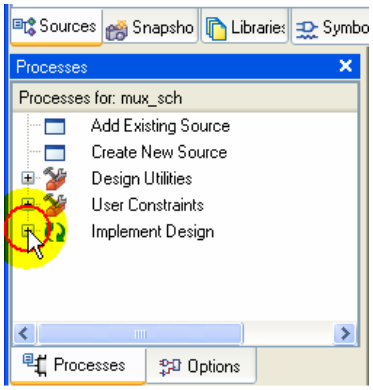


β. Επιλογή (κλικ του ποντικιού) στο **Synthesis/Implementation**.

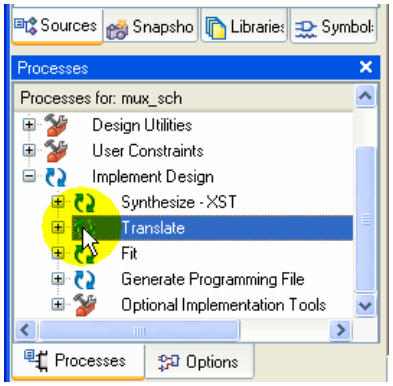


γ. Τελική εμφάνιση μετά την επιλογή.

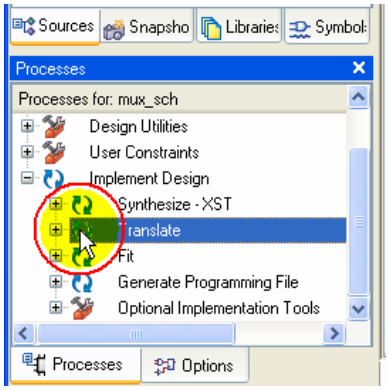
Εικόνα Α.60 Επιλογή **Synthesis/Implementation** στην ετικέτα **Sources**, (**εικόνες Α.60α** ως **Α.60γ**)



α. Επιλογή (κλικ του ποντικιού) στο **Implement Design**.

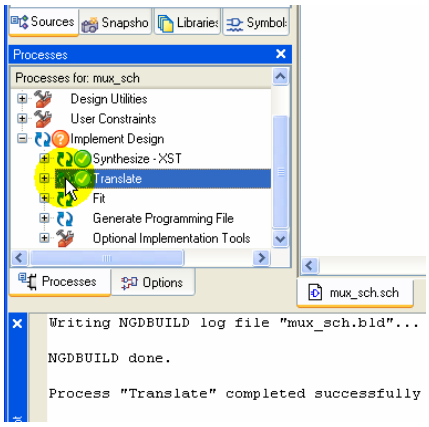


β. Επιλογή (κλικ του ποντικιού) στο **Translate**.



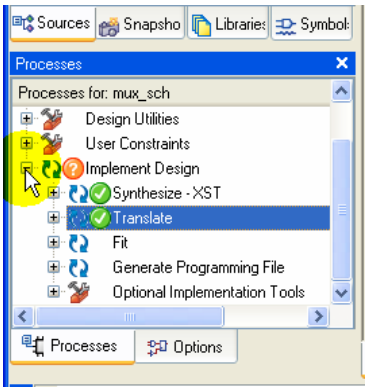
γ. Ενεργοποίηση εφαρμογής (διπλό κλικ του ποντικιού) στο **Translate**.

Εικόνα Α.61 Εκτέλεση εφαρμογής **Translate**, (**εικόνες Α.61α** ως **Α.61γ**)

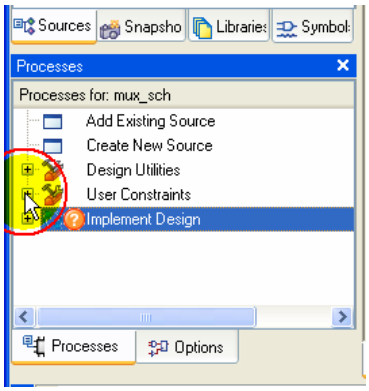


Εικόνα Α.62 Επιτυχημένη εκτέλεση εφαρμογής **Translate**

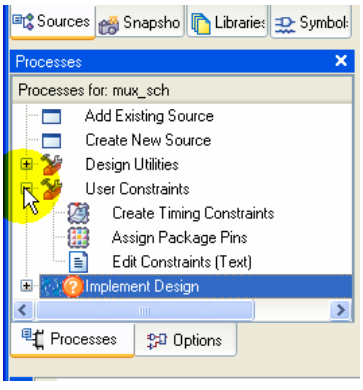
Πίνακας Α.19 Εκτέλεση εφαρμογής **Assign Package Pins** (αντιστοίχησης ακίδων του ολοκληρωμένου), (**εικόνες Α.63α** ως **Α.63δ**)



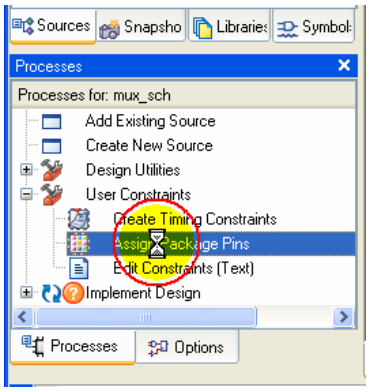
α. Η επιλογή με απλό κλικ του ποντικιού στο «-» δίπλα στο εικονίδιο **Implement Design**.



β. Η επιλογή με απλό κλικ του ποντικιού στο «+» δίπλα στο εικονίδιο **User Constraints**.

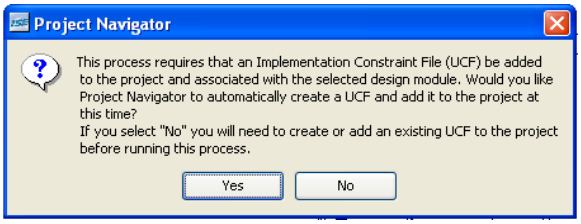


γ. Εμφάνιση μετά την τελευταία επιλογή.

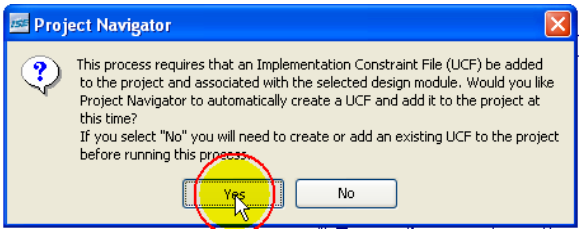


δ. Η επιλογή με διπλό κλικ του ποντικιού στο **Assign Package Pins** για εκτέλεση της εφαρμογής.

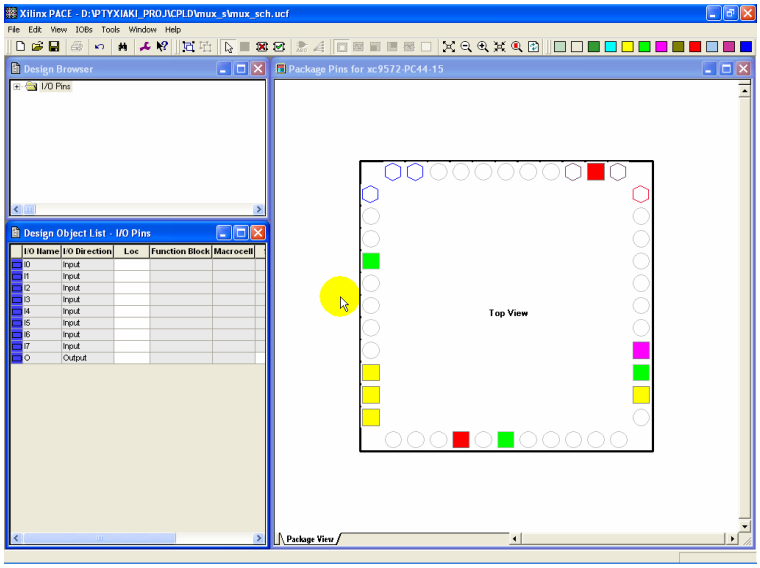
Πίνακας Α.20 Παράθυρα αντιστοίχησης ακίδων του ολοκληρωμένου, (**εικόνες Α.64α** ως **Α.64δ**)



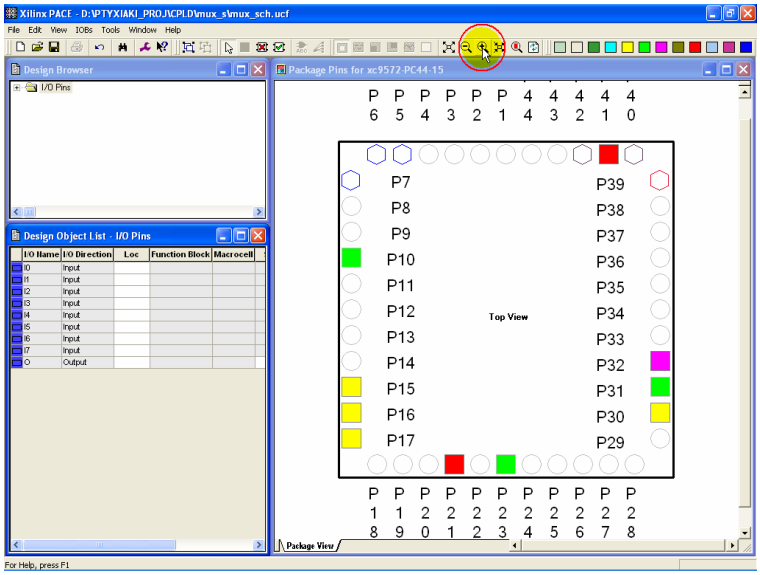
α. Παράθυρο μετά την εκτέλεση της εφαρμογής **Assign Package Pins**.



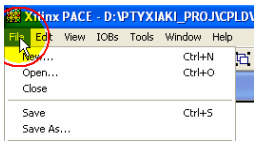
β. Επιλογή (με απλό κλικ του ποντικιού) στο κουμπί **Yes**.



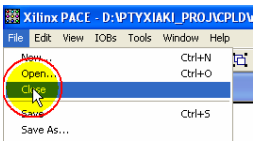
γ. Εισαγωγή σε καινούργιο παραθυρικό περιβάλλον.



δ. Επιλογή (με απλό κλικ του ποντικιού) στο **Zoom In** (εργαλείο μεγέθυνσης).

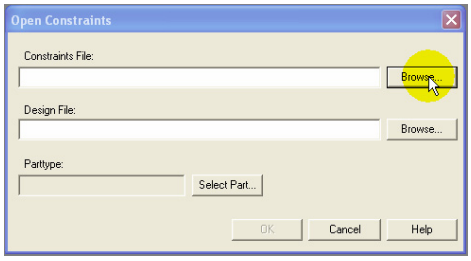


α. Επιλογή **File** με κλικ του ποντικιού.

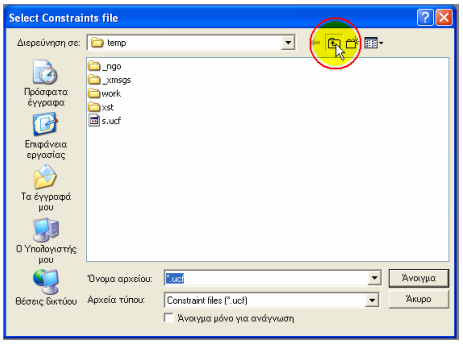


β. Επιλογή **Close** με κλικ του ποντικιού.

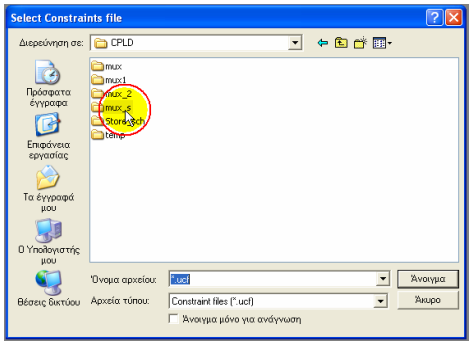
γ. Επιλογή **Open** με κλικ του ποντικιού.



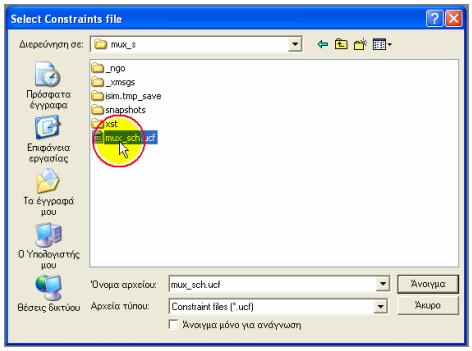
δ. Επιλογή **Browse** με κλικ του ποντικιού.



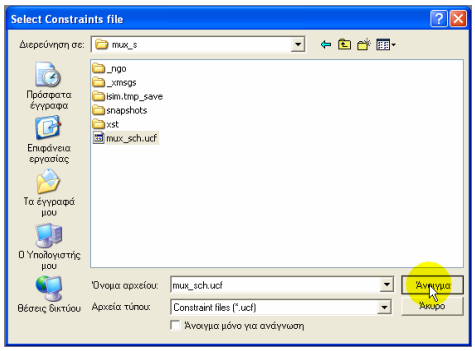
ε. Επιλογή ανοίγματος φακέλου ενός επιπέδου παραπάνω με κλικ του ποντικιού.



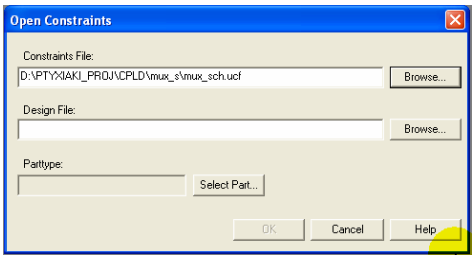
στ. Επιλογή ορθού φακέλου με κλικ του ποντικιού.



ζ. Επιλογή αρχείου ucf με κλικ του ποντικιού.

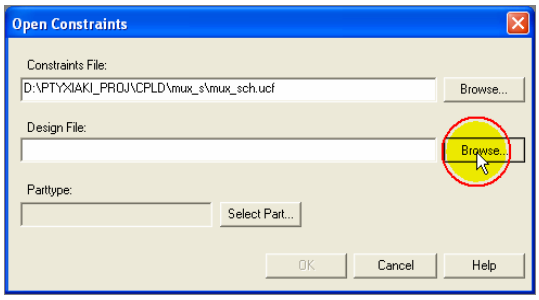


η. Επιλογή ορθού αρχείου με κλικ του ποντικιού.

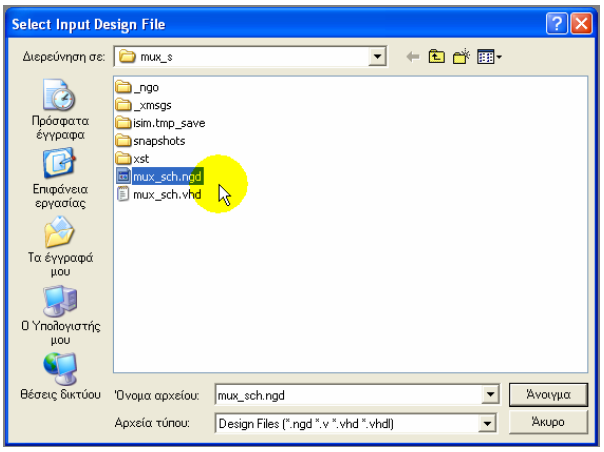


θ. Εισαγωγή αρχείου ucf με κλικ του ποντικιού.

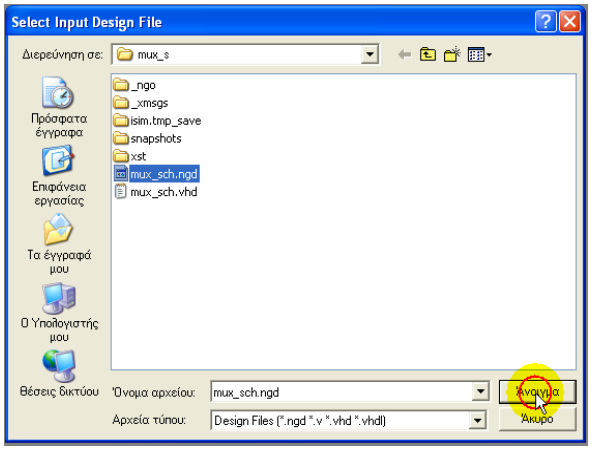
Εικόνα Α.65 Εισαγωγή ορθού ucf αρχείου, (**εικόνες Α.65α** ως **Α.65θ**)



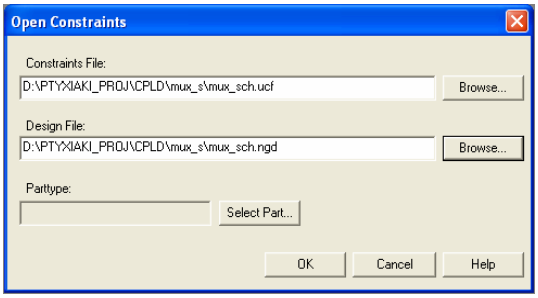
α. Επιλογή **Browse** με κλικ του ποντικιού.



β. Επιλογή αρχείου ngd με κλικ του ποντικιού.

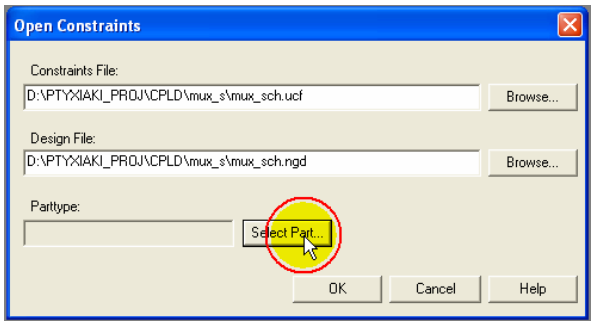


γ. Επιλογή **Άνοιγμα** με κλικ του ποντικιού.

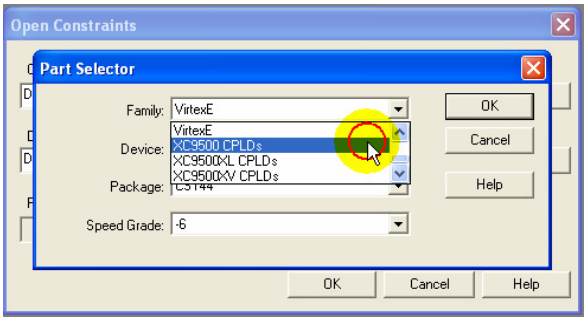


δ. Εισαγωγή αρχείου ngd με κλικ του ποντικιού.

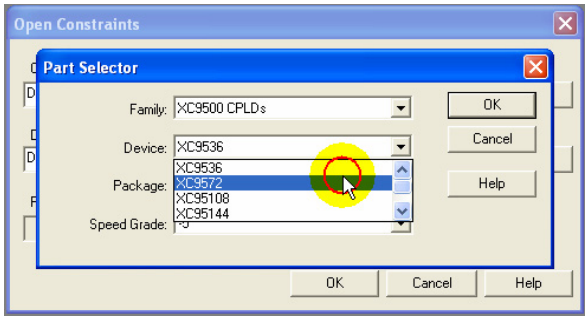
Εικόνα Α.66 Εισαγωγή ορθού ngd αρχείου, (**εικόνες Α.66α** ως **Α.66δ**)



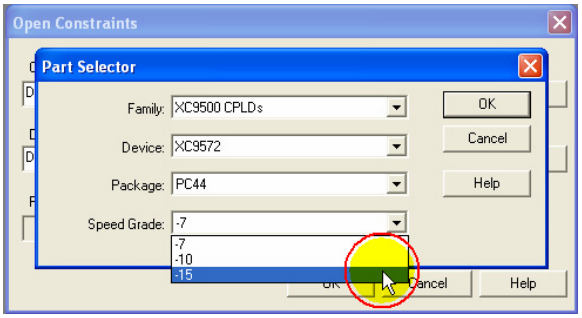
α. Επιλογή **Select Part** με κλικ του ποντικιού.



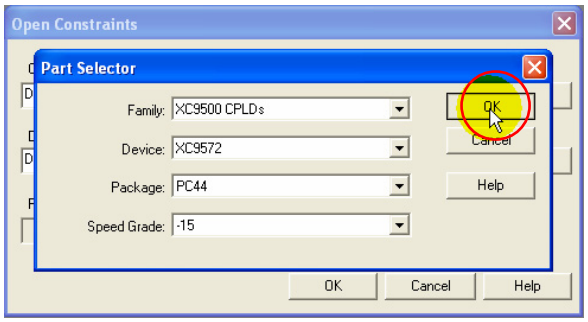
β. Επιλογή οικογένειας ολοκληρωμένου (**Family**) με κλικ του ποντικιού.



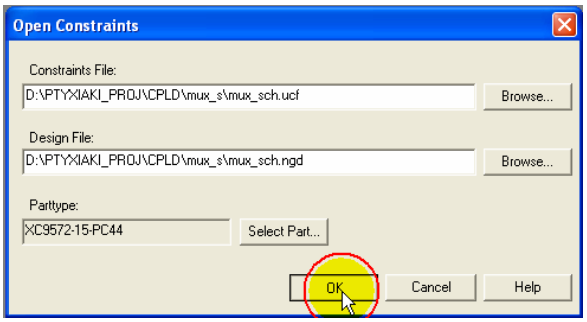
γ. Επιλογή ολοκληρωμένου (**Device**) με κλικ του ποντικιού.



δ. Επιλογή ταχύτητας ολοκληρωμένου (**Speed Grade**) με κλικ του ποντικιού. (Η συσκευασία **Package** είναι επιλεγμένη σωστά, διαφορετικά αλλάζει με τον ίδιο τρόπο στη σωστή της τιμή).

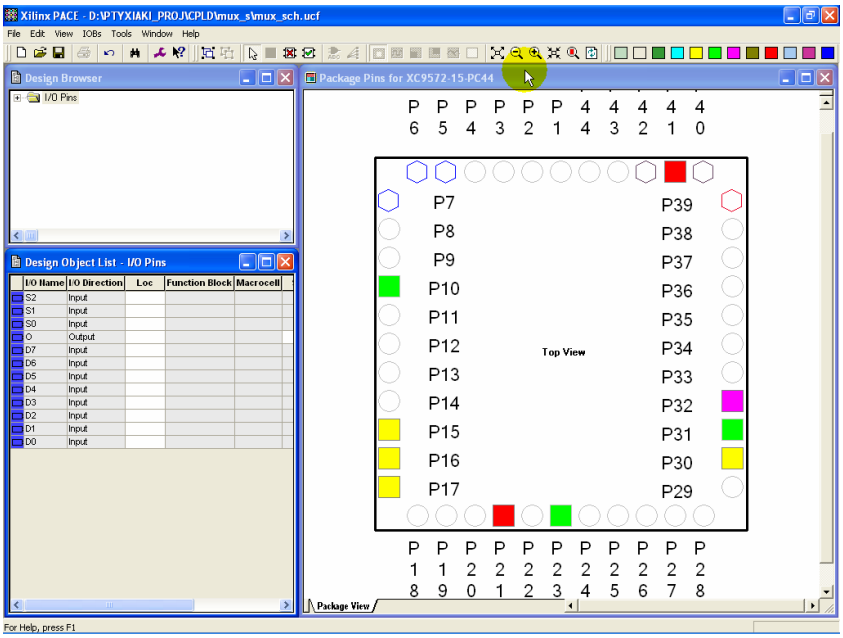


ε. Επιλογή **OK** με κλικ του ποντικιού. (Συμπλήρωση όλων των στοιχείων για το παράθυρο **Part Selector**).

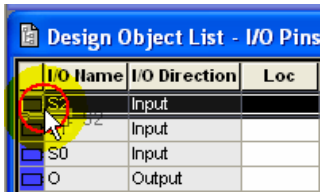


στ. Επιλογή **OK** με κλικ του ποντικιού. (Συμπλήρωση όλων των στοιχείων για το παράθυρο **Open Constraints**).

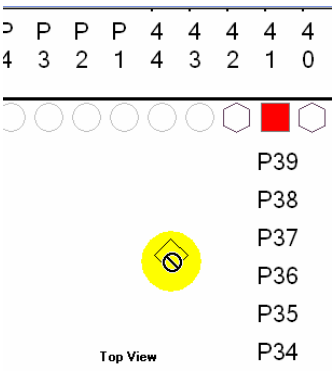
Εικόνα Α.67 Εισαγωγή ορθών στοιχείων του ολοκληρωμένου, (**εικόνες Α.67α** ως **Α.67στ**)



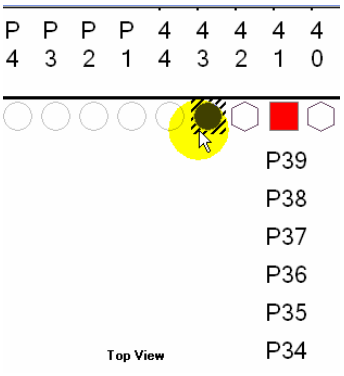
Εικόνα Α.68 Παραθυρικό περιβάλλον ορθής αντιστοίχησης ακίδων του ολοκληρωμένου σύμφωνα με την γενικότερη εργασία μας (project)



α. Επιλογή εισόδου/εξόδου (**I/O**) με πάτημα του ποντικιού συνεχόμενα.

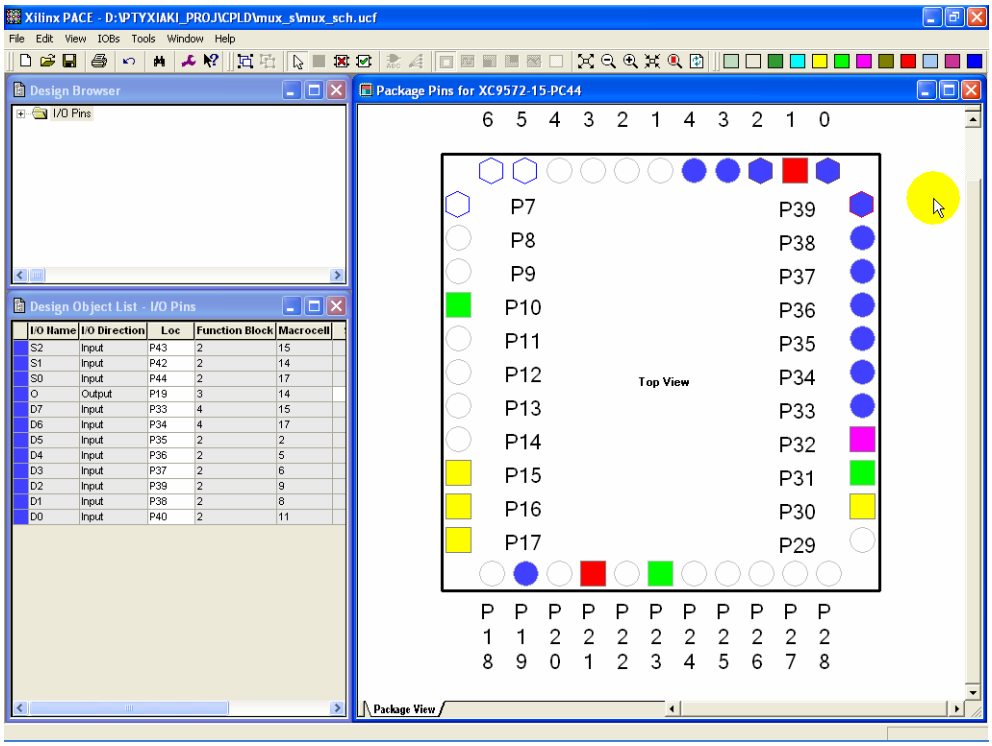


β. Συνεχόμενα πατημένο και συρόμενο ποντίκι.

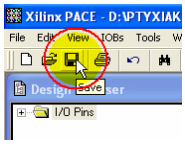


γ. Αφημένο ποντίκι στην επιλεγμένη θέση ακίδας 43 (**Pin 43** ή **P43)**.

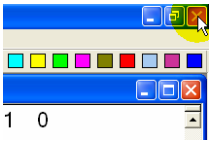
Εικόνα Α.69 Αντιστοίχηση εισόδου/εξόδου (**I/O**) στην ακίδα (**Pin**) της επιλογής μας, (**εικόνες Α.69α** ως **Α.69γ**)



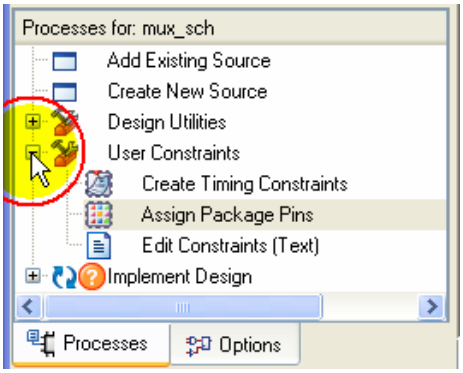
α. Παραθυρικό περιβάλλον μετά την ολοκλήρωση αντιστοίχησης όλων των ακίδων με τις εισόδους/εξόδους



β. Επιλογή **Save** με κλικ του ποντικιού.

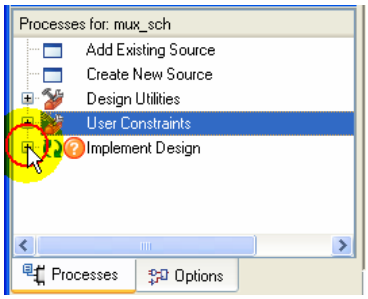


γ. Επιλογή κλεισίματος παραθύρου με κλικ του ποντικιού.

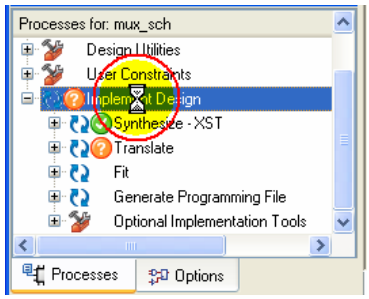


δ. Επιλογή του «-» με κλικ του ποντικιού στα αριστερά του **User Constaints.**

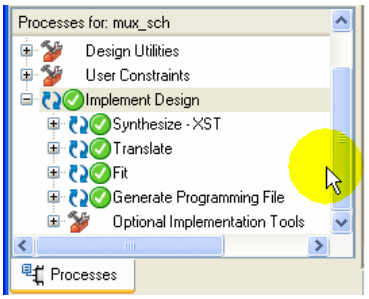
Εικόνα Α.70 Ολοκλήρωση αντιστοίχησης εισόδων/εξόδων (**I/O**) σε ακίδες (**P**ins), (**εικόνες Α.70α** ως **Α.70δ**)



α. Επιλογή του «+» με κλικ του ποντικιού στα αριστερά του **Implement Design.**



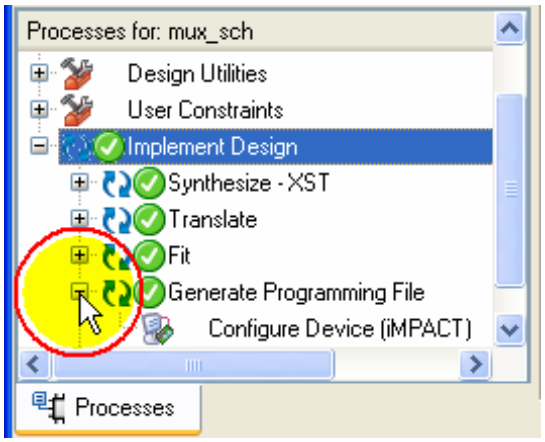
β. Με διπλό αριστερό κλικ η ενεργοποίηση της εφαρμογής **Implement Design.**



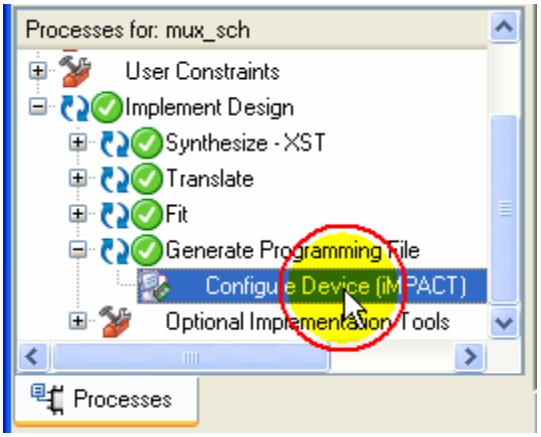
γ. Τέλος επιτυχημένης ενεργοποίησης της εφαρμογής **Implement Design.**

Εικόνα Α.71 Ολοκλήρωση ενεργοποίησης της εφαρμογής **Implement Design,** (**εικόνες Α.71α** ως **Α.71γ**)

Α.1.5.2 Γρήγορος προγραμματισμός

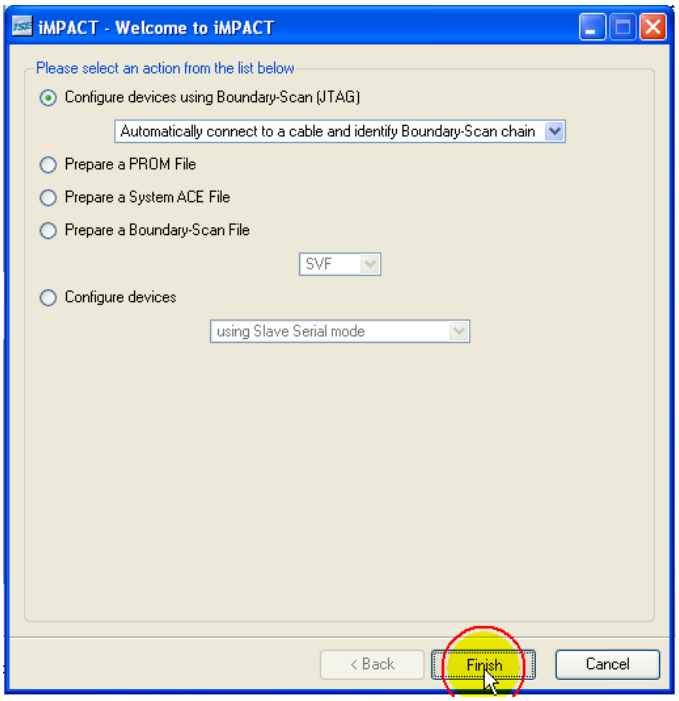


α. Επιλογή του «+» με κλικ του ποντικιού στα αριστερά του **Generate Programming File.** (Εμφανίζεται το **Configure Device**).

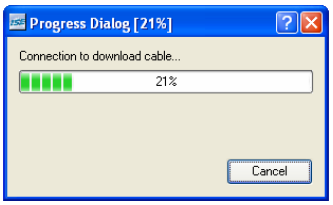


β. Με διπλό αριστερό κλικ η ενεργοποίηση της εφαρμογής **Configure Device.**

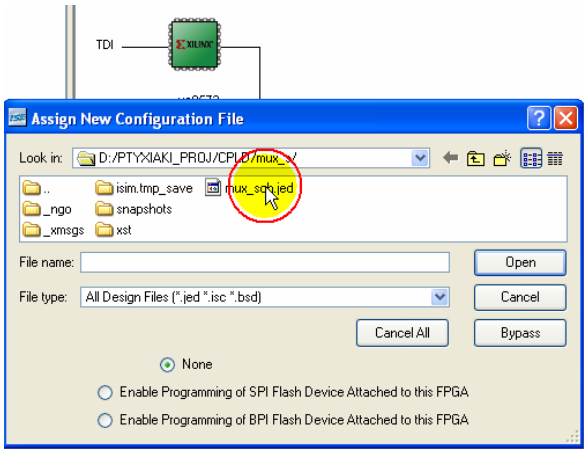
Εικόνα Α.72 Ενεργοποίηση εφαρμογής **Configure Device**, (**εικόνες Α.72α** ως **Α.72β**)



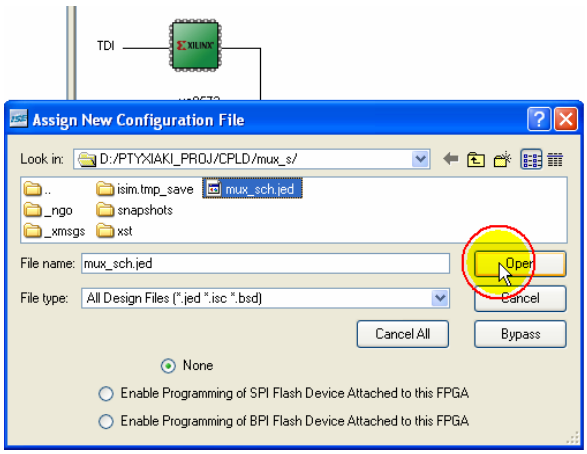
α. Παράθυρο μετά την ενεργοποίηση της εφαρμογής **Configure Device.** Επιλογή (κλικ ποντικιού) **Finish**.



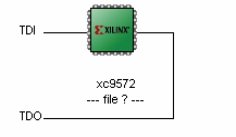
β. Μετάβαση σε παράθυρο.



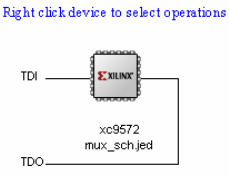
γ. Επιλογή (κλικ ποντικιού) ορθού **jed** αρχείου.



δ. Επιλογή (κλικ ποντικιού) στο κουμπί **Open** για άνοιγμα του ορθού **jed** αρχείου.

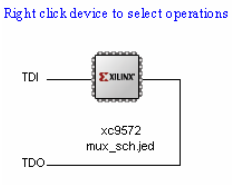


ε. Εμφάνιση ολοκληρωμένου πριν το άνοιγμα του ορθού **jed** αρχείου.

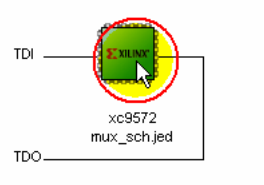


στ. Εμφάνιση ολοκληρωμένου μετά το άνοιγμα του ορθού **jed** αρχείου.

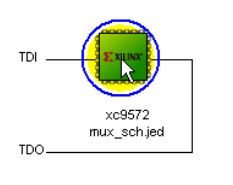
Εικόνα Α.73 Άνοιγμα του κατάλληλου **jed** αρχείου, (**εικόνες Α.73α** ως **Α.73στ**)



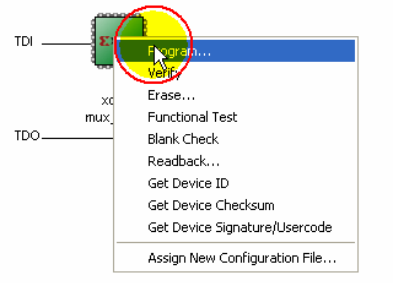
α. Εμφάνιση ολοκληρωμένου μετά το άνοιγμα του ορθού **jed** αρχείου.



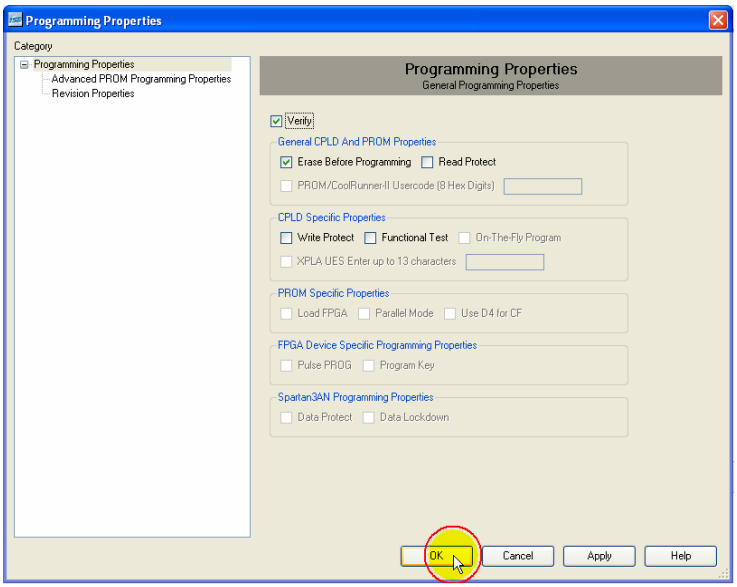
β. Επιλογή (κλικ ποντικιού) στο ολοκληρωμένο (πράσινο χρώμα από γκρί).



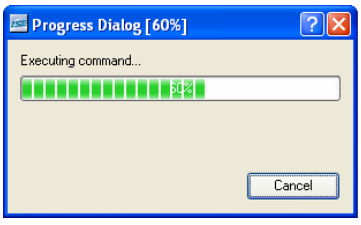
γ. Επιλογή (δεξί κλικ ποντικιού) στο ολοκληρωμένο.



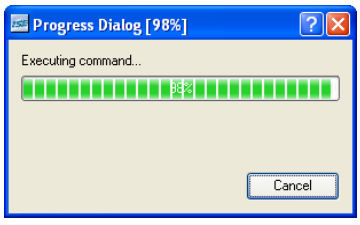
δ. Επιλογή (κλικ ποντικιού) στο κουμπί **Program**.



γ. Επιλογή (κλικ ποντικιού) στο κουμπί **OK**, (Προεπιλεγμένα τα κουμπιά **Verify** και **Erase Before Programming**.)

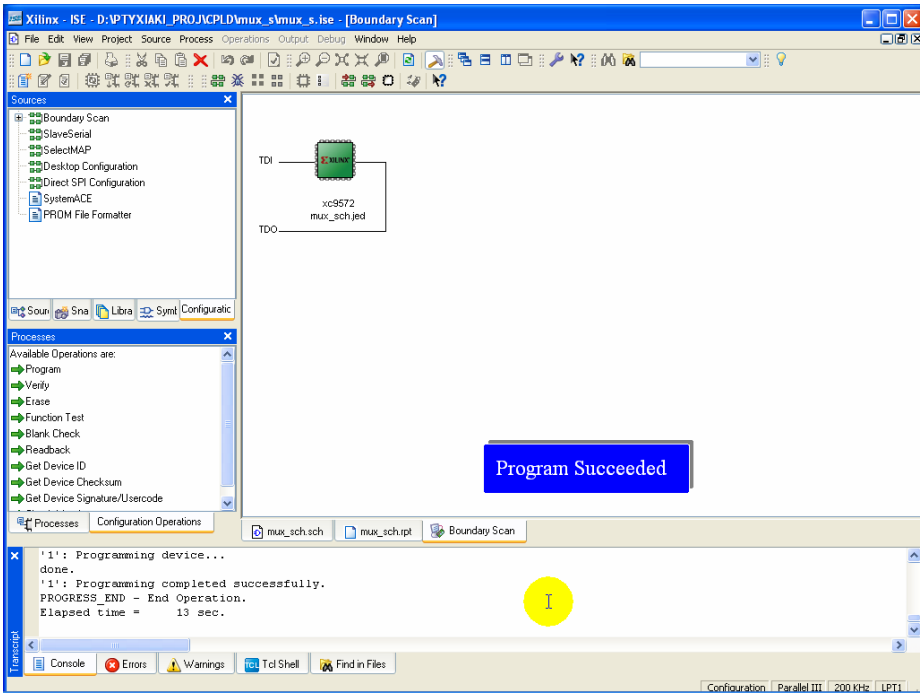


δ. Πρόοδος προγραμματισμού.



ε. Πρόοδος προγραμματισμού.

Εικόνα Α.74 Κυρίως προγραμματισμός, (**εικόνες Α.74α** ως **Α.74ε**)



α. Παράθυρο πέρατος προγραμματισμού.

β. Επιβεβαίωση προγραμματισμού.

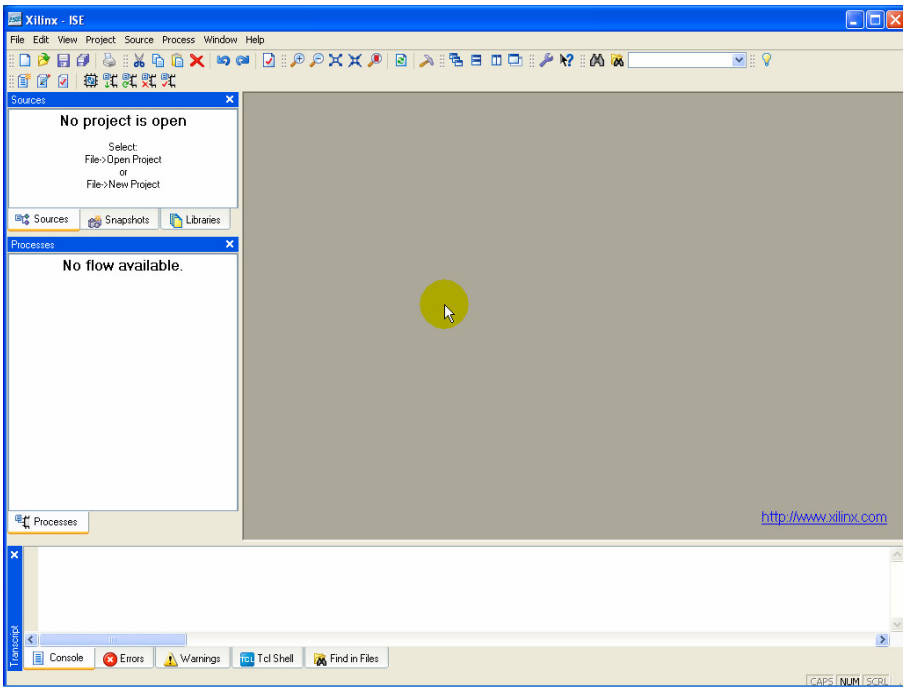
Εικόνα Α.75 Επιτυχημένο πέρας προγραμματισμού, (**εικόνες Α.75α** ως **Α.75β**)

Α.2 ΕΙΚΟΝΕΣ ΑΝΑΠΤΥΞΗΣ ΜΕ ΤΗΝ ΠΕΡΙΓΡΑΦΙΚΗ ΓΛΩΣΣΑ (VHDL)

Α.2.1 Έναρξη εφαρμογής

Εικόνα Α.76 Συντόμευση στην επιφάνεια εργασίας του προγράμματος **Xilinx® ISE™ 9.1.03i**

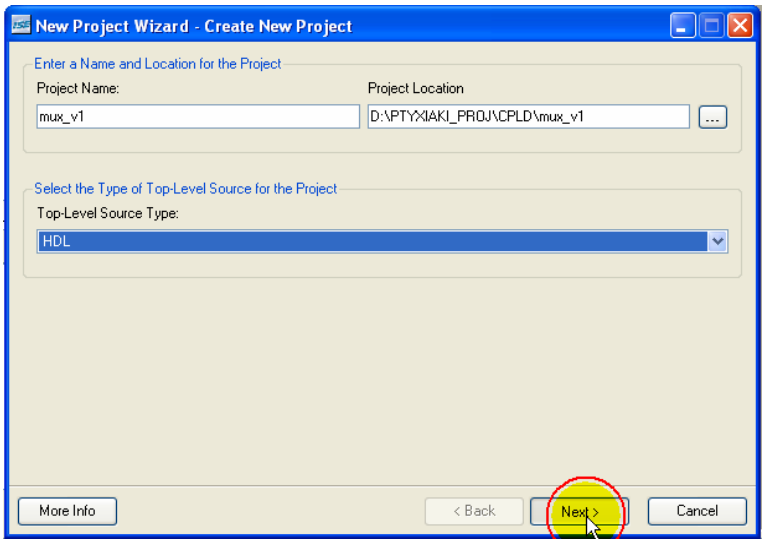
Εικόνα Α.77 Παράθυρο κατά την έναρξη της εφαρμογής **Project Navigator**



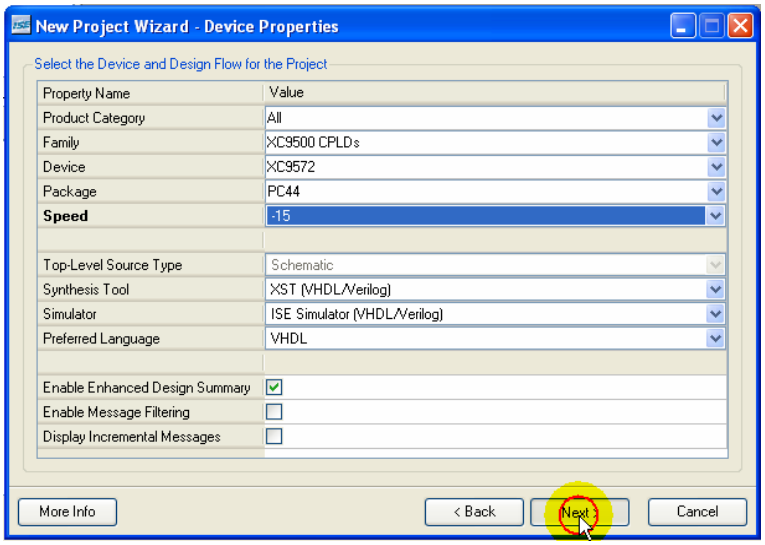
Εικόνα Α.78 Περιβάλλον **Xilinx® ISE™ 9.1.03i** για ανάπτυξη συνολικών εργασιών (projects)

Α.2.2 Δημιουργία νέας συνολικής εργασίας

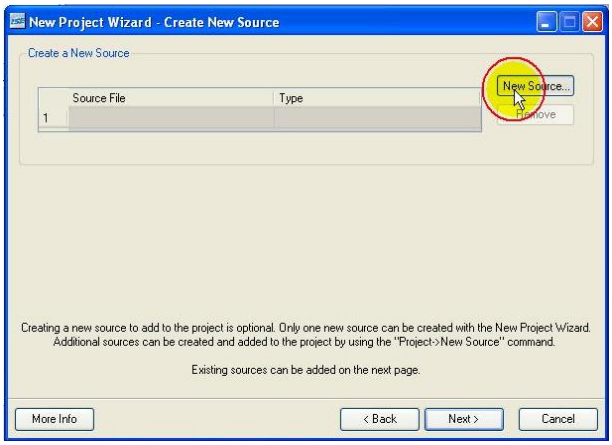
Εικόνα Α.79 Επιλογή διαδρομής **File→New Project**



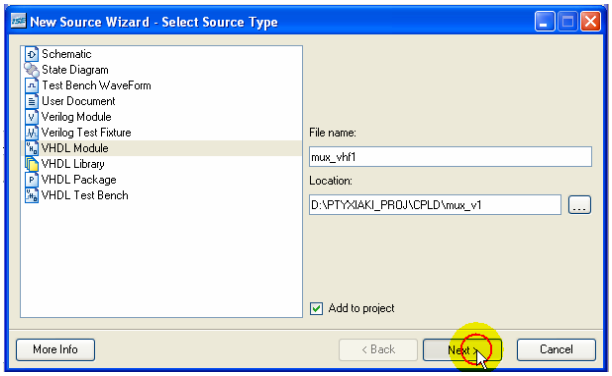
Εικόνα Α.80 Εισαγωγή στοιχείων συνολικής εργασίας (project), επιλογή **Next**



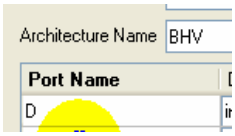
Εικόνα Α.81 Εισαγωγή στοιχείων ολοκληρωμένου και εργαλείων, επιλογή **Next**



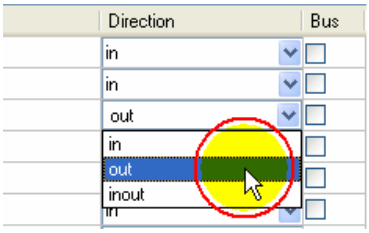
Εικόνα Α.82 Εισαγωγή νέου αρχείου, επιλογή **New Source**



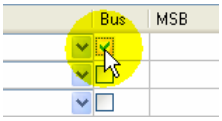
Εικόνα Α.83 Εισαγωγή τύπου και ονόματος του νέου αρχείου, επιλογή **Next**



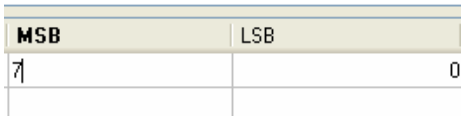
α. Πληκτρολόγηση ονομασίας.



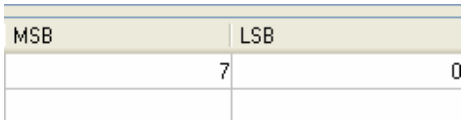
β. Επιλογή κατάστασης.



γ. Επιλογή λεωφόρου (**Bus**).

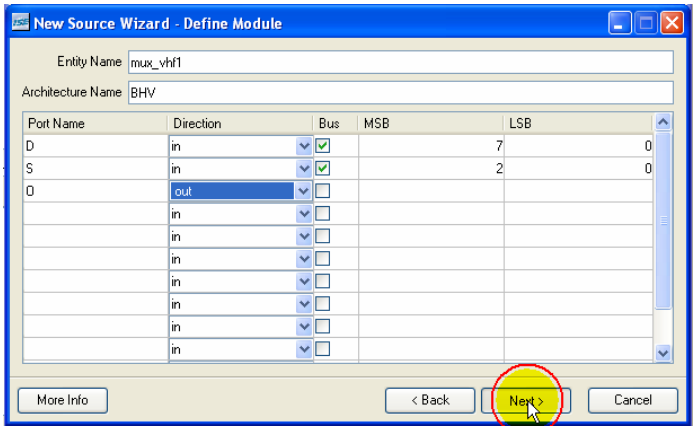


δ. Πληκτρολόγηση αριθμού περισσότερο σημαντικού bit.

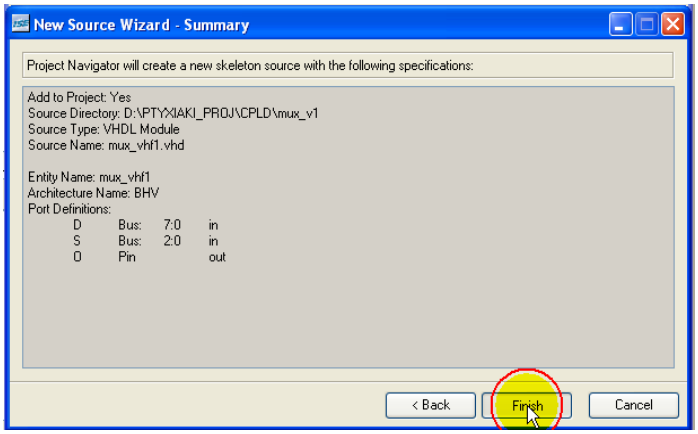


ε. Εισαγωγή προηγούμενης ενέργειας.

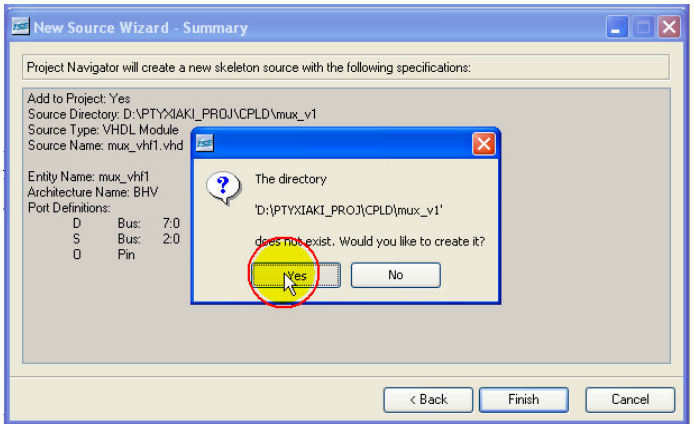
Εικόνα Α.84 Εισαγωγή στοιχείων και ρύθμιση εισόδων/εξόδων (**ports**), (**εικόνες Α.84α** ως **Α.84ε**)



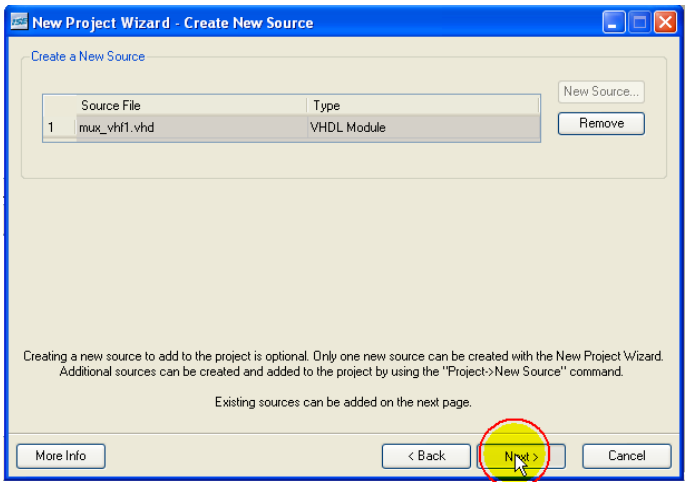
Εικόνα Α.85 Εισαγωγή στοιχείων του αρχείου VHDL, επιλογή **Next**



Εικόνα Α.86 Επισκόπηση στοιχείων του αρχείου VHDL, επιλογή **Finish**

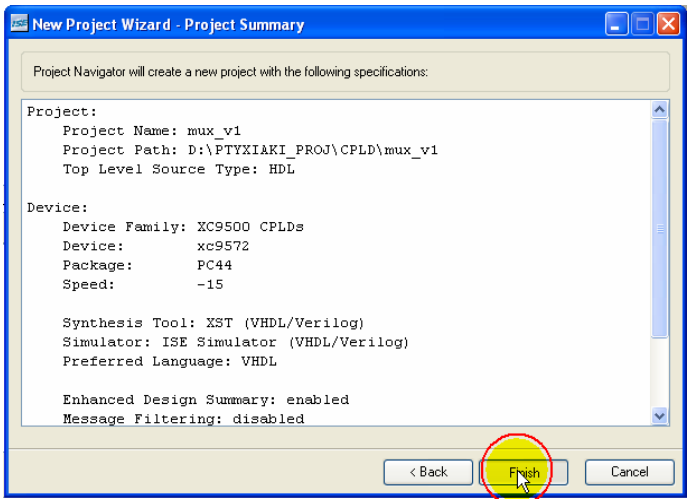


Εικόνα Α.87 Ερώτηση για δημιουργία καταλόγου σε επιλεγμένη διαδρομή, επιλογή **Yes**



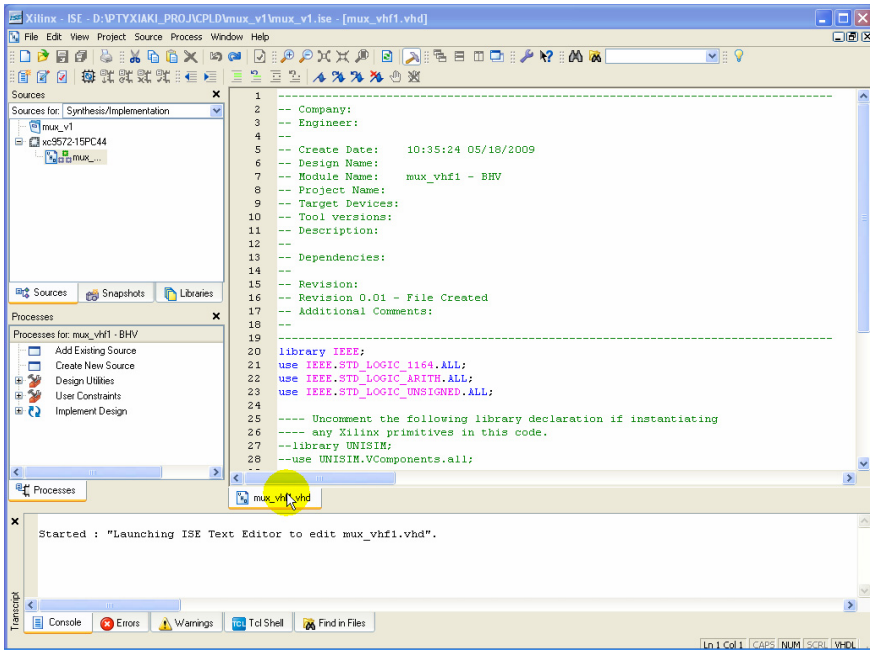
Εικόνα Α.88 Εισαγωγή και άλλων αρχείων σχετικών με την εργασία (project), επιλογή **Next**

Εικόνα Α.89 Συνέχεια διαδικασίας, Επιλογή **Next**



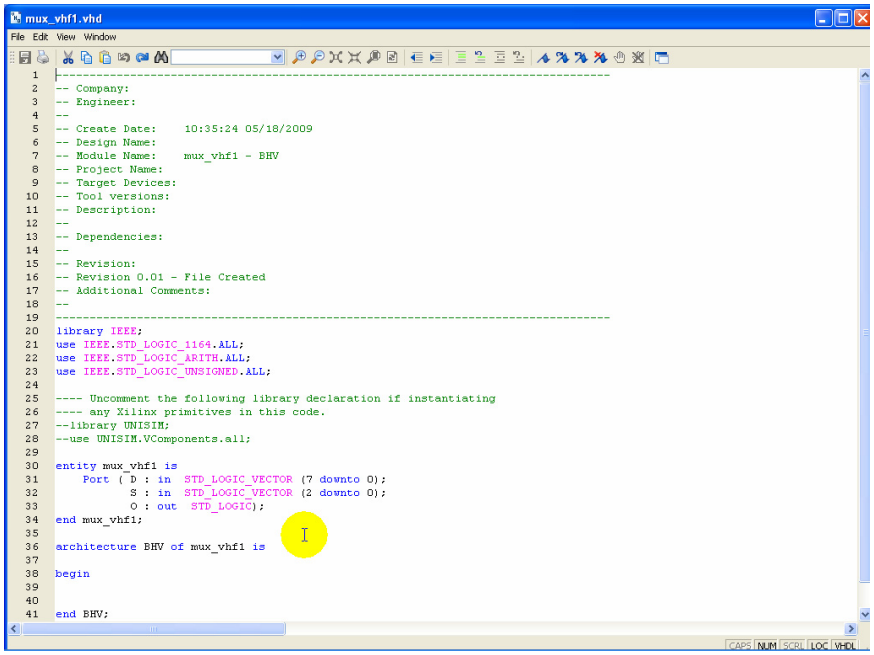
Εικόνα Α.90 Συνολική επισκόπηση της συνολικής εργασίας (project), Επιλογή **Finish**

Α.2.3 Εργασία με τη VHDL

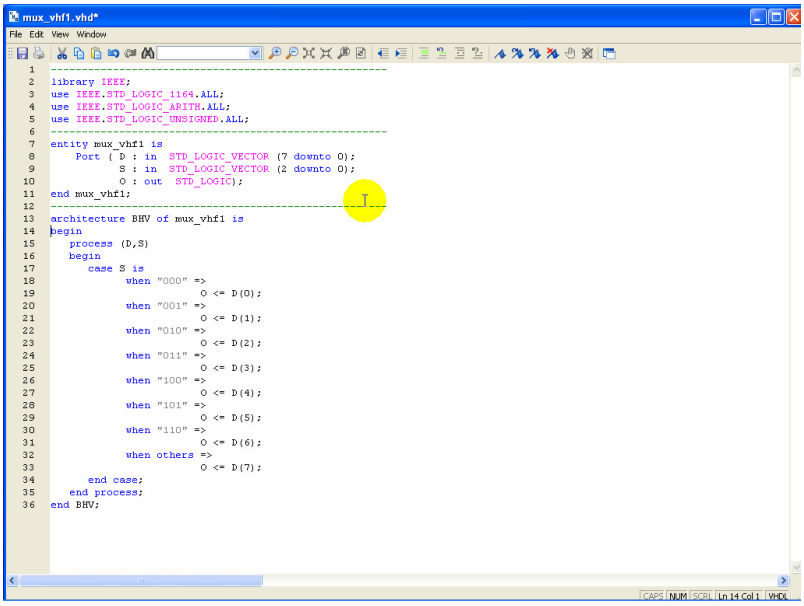


Εικόνα Α.91 Εισαγωγή στο περιβάλλον της συνολικής εργασίας (project) στη VHDL

Εικόνα Α.92 Επιλογή εργαλείου **Float this window**

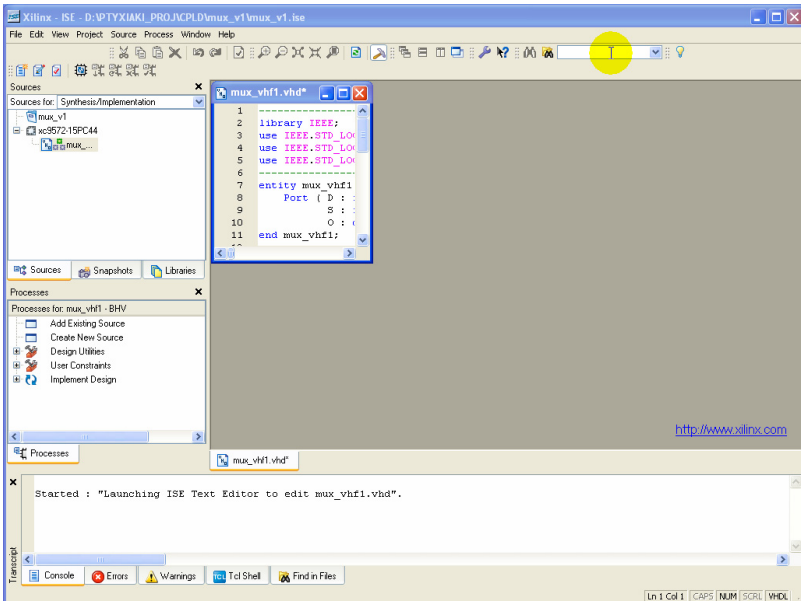


Εικόνα Α.93 Βελτιστοποιημένη επιφάνεια στη VHDL

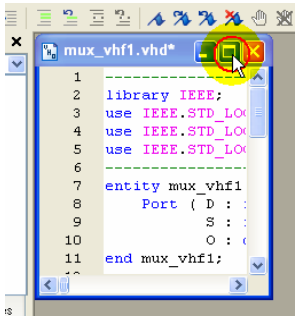


Εικόνα Α.94 Πληκτρολόγηση χρήσιμων εντολών στη βελτιστοποιημένη επιφάνεια της VHDL

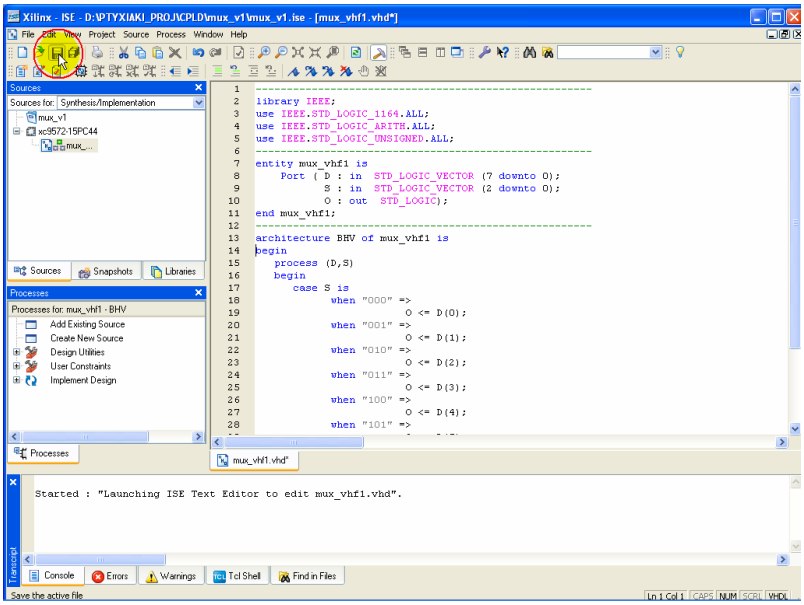
Εικόνα Α.95 Η επιλογή του εργαλείου **Dock this window**



Εικόνα Α.96 Μετάβαση της VHDL στο κοινό παραθυρικό περιβάλλον με την υπόλοιπη συνολική εργασία (project)

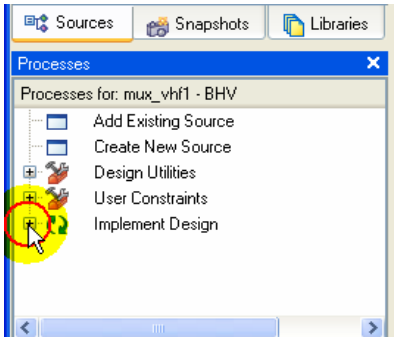
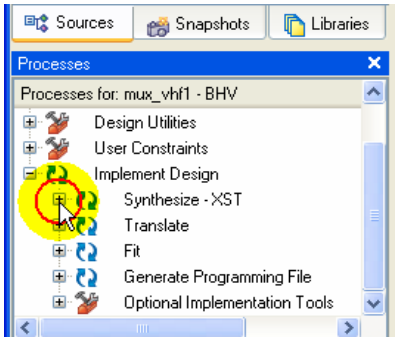
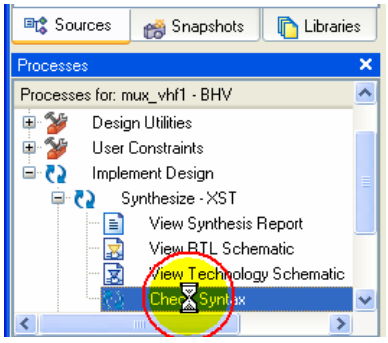


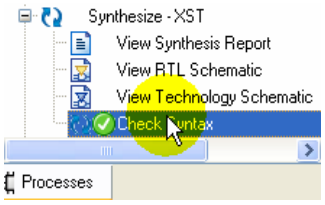
Εικόνα Α.97 Επιλογή μεγιστοποίησης παραθύρου της VHDL

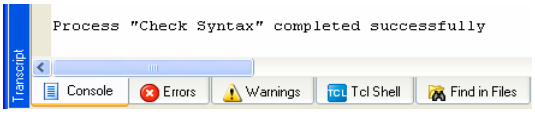


Εικόνα Α.98 Επιλογή **Save** (με κλικ του ποντικιού) για αποθήκευση αλλαγών

Πίνακας Α.21 Διαδικασία ενεργοποίησης εφαρμογής **Check Syntax** (**διπλό** κλικ του ποντικιού) για έλεγχο πιθανών λαθών, (**εικόνες Α.99α** ως **Α.99γ**)

		
α. Επιλογή του «+» με κλικ του ποντικιού στα αριστερά του **Implement Design**.	β. Επιλογή του «+» με κλικ του ποντικιού στα αριστερά του **Synthesize - XST**.	γ. Επιλογή στο **Check Syntax** (με **διπλό** κλικ ποντικιού) για έλεγχο πιθανών λαθών.





α. Επιτυχής εκτέλεση (επιλεγμένος πράσινος κύκλος). β. Επιτυχής εκτέλεση (μήνυμα).

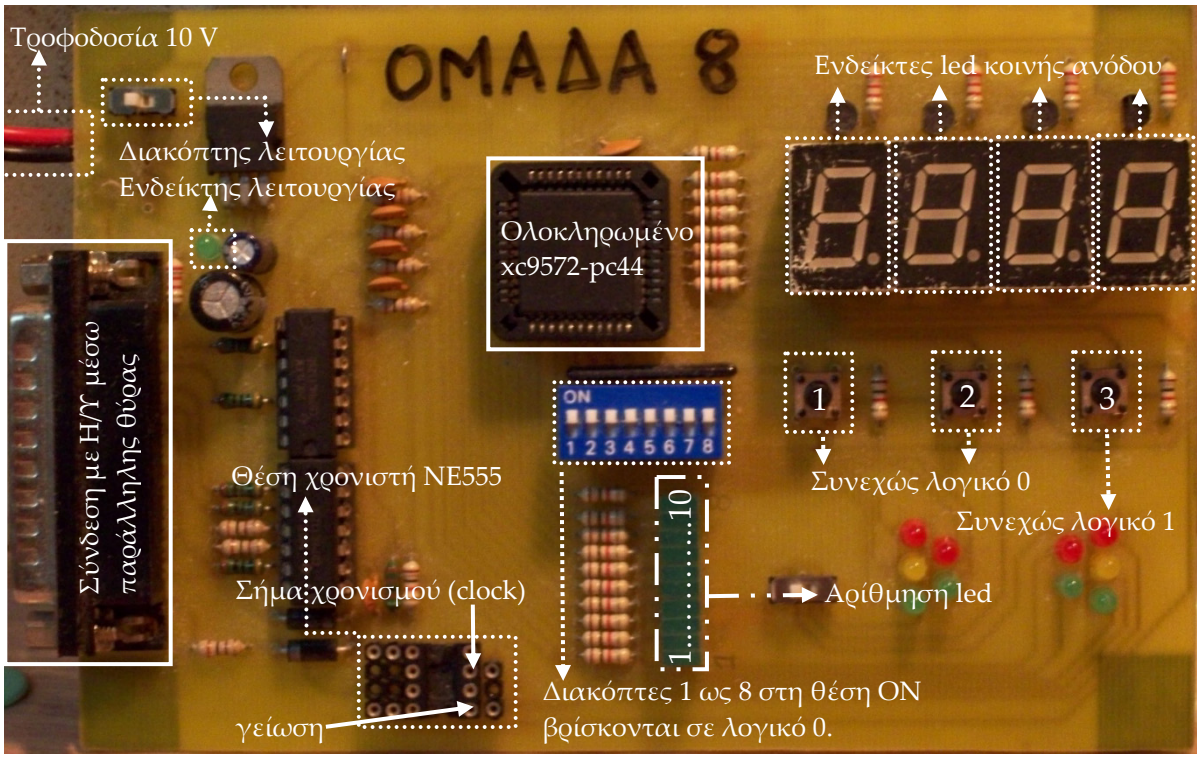
Εικόνα Α.100 Επιτυχής εκτέλεση **Check Syntax (**δεν βρέθηκαν λάθη), (**εικόνες Α.100α** ως **Α.100 β**)

Οδηγία

Στις προηγούμενες εικόνες (και τις περισσότερες των περιπτώσεων) όπου ο κίτρινος κύκλος του ποντικιού εμπεριέχει ή βρίσκεται μέσα σε μεγαλύτερο κόκκινο δακτύλιο σημαίνει αριστερό κλικ του ποντικιού. Στην περίπτωση που το χρώμα του ίδιου δακτυλίου είναι γαλάζιο σημαίνει δεξιό κλικ του ποντικιού.

Β. ΑΝΑΠΤΥΞΙΑΚΗ ΠΛΑΚΕΤΑ ΕΡΓΑΣΤΗΡΙΟΥ

Το πειραματικό μέρος (προγραμματισμός) όλων των συνολικών εργασιών πραγματοποιήθηκε στην αναπτυξιακή πλακέτα της **εικόνας Β.1** κατόπιν παραχώρησης από το εργαστήριο των Ψηφιακών Κυκλωμάτων Ι & ΙΙ του τμήματος μαζί με τις γενικές πληροφορίες για την πλακέτα αυτή που ακολουθούν.



Εικόνα Β.1 Αναπτυξιακή πλακέτα εργαστηρίου

Ο προγραμματισμός του ολοκληρωμένου xc9572-pc44 της εταιρείας Xilinx® τεχνολογίας P.L.D. έγινε με τη βοήθεια υπολογιστή μέσω παράλληλης θύρας και τη χρησιμοποίηση της εφαρμογής Xilinx® ISE™ 9.1.03i. Παρακάτω στον **πίνακα Β.1** φαίνεται η διασύνδεση των ακίδων του ολοκληρωμένου με τα υπόλοιπα υλικά της αναπτυξιακής πλακέτας.

Πίνακας Β.1 Ανάπτυξη των υλικών της πλακέτας

pin1	τομέας e	**pin23**	GND
pin2	τομέας c	**pin24**	green led 5
pin3	D.P.	**pin25**	green led 6
pin4	τομέας d	**pin26**	green led 7
pin5	clock	**pin27**	green led 8
pin6	τομέας a	**pin2S**	green led 9
pin7	τομέας f	**pin29**	green led 10
pin8	τομέας g	**pin30**	programming pin
pin9	τομέας b	**pin31**	GND
pin10	GND	**pin32**	Vcc
pin11	display 4 c.a.	**pin33**	dip-sw 1
pinl2	display 3 c.a.	**pin34**	dip-sw 2
pinl3	display 2 c.a.	**pin35**	dip-sw 3
pini4	display 1 c.a.	**pin36**	dip-sw 4
pin15	programming pin	**pin37**	dip-sw 5
pinl6	programming pin	**pin38**	dip-sw 7
pinl7	programming pin	**pin39**	dip-sw 6
pinl8	green led 2	**pin40**	dip-sw 8
pin19	green led 1	**pin41**	Vcc
pin20	green led 3	**pin42**	Button 2
pin21	Vcc	**pin43**	Button 1
pin22	green led 4	**pin44**	Button 3

Γενικές πληροφορίες

Τα **display** λειτουργούν ως εξής: Αρχικά πρέπει να αναφέρουμε ότι τα **display** είναι κοινής ανόδου. Έτσι, για να λειτουργήσει το **3ο display** πρέπει να ενεργοποιήσουμε την κοινή άνοδο του **3ου display,** δηλαδή, το pin 12 πρέπει να ενεργοποιηθεί με λογικό «Ο». Ανάλογα με τον τομέα που θα ανάψουμε θα ενεργοποιήσουμε και το αντίστοιχο ποδαράκι (pin).

Για τα τρία **button** αναφέρουμε τα εξής: Τα **button** 1 & 2 έχουν αντιστάσεις **pull down,** που σημαίνει ότι δίνουν στο pin που είναι συνδεδεμένα συνεχώς λογικό «Ο», ενώ το **button** 3 έχει αντίσταση **pull up,** δηλαδή δίνει συνεχώς λογικό «1» στο pin44 που έχει συνδεθεί.

Τέλος με το χρονιστή **(timer) NE555** αναπτύσσουμε μια γεννήτρια ψηφιακού σήματος **(clock)** στο pin5 του XC9536. Για να έχουμε την επιθυμητή συχνότητα πρέπει να προσθέσουμε δύο αντιστάσεις και έναν πυκνωτή. Τη συχνότητα μας δίνει ο παρακάτω τύπος:

$$f = \frac{1}{0.693 \cdot C \cdot (R_1 + 2 \cdot R_2)}$$

ΦΥΛΛΑ ΔΕΔΟΜΕΝΩΝ

XILINX®

XC9572 In-System Programmable CPLD

DS065 (v4.3) April 3, 2006 **Product Specification**

Features

- 7.5 ns pin-to-pin logic delays on all pins
- f_{CNT} to 125 MHz
- 72 macrocells with 1,600 usable gates
- Up to 72 user I/O pins
- 5V in-system programmable
  - Endurance of 10,000 program/erase cycles
  - Program/erase over full commercial voltage and temperature range
- Enhanced pin-locking architecture
- Flexible 36V18 Function Block
  - 90 product terms drive any or all of 18 macrocells within Function Block
  - Global and product term clocks, output enables, set and reset signals
- Extensive IEEE Std 1149.1 boundary-scan (JTAG) support
- Programmable power reduction mode in each macrocell
- Slew rate control on individual outputs
- User programmable ground pin capability
- Extended pattern security features for design protection
- High-drive 24 mA outputs
- 3.3V or 5V I/O capability
- Advanced CMOS 5V FastFLASH™ technology
- Supports parallel programming of more than one XC9500 concurrently
- Available in 44-pin PLCC, 84-pin PLCC, 100-pin PQFP, and 100-pin TQFP packages

Description

The XC9572 is a high-performance CPLD providing advanced in-system programming and test capabilities for general purpose logic integration. It is comprised of eight 36V18 Function Blocks, providing 1,600 usable gates with propagation delays of 7.5 ns. See Figure 2 for the architecture overview.

Power Management

Power dissipation can be reduced in the XC9572 by configuring macrocells to standard or low-power modes of operation. Unused macrocells are turned off to minimize power dissipation.

Operating current for each design can be approximated for specific operating conditions using the following equation:

I_{CC} (mA) = MC_{HP} (1.7) + MC_{LP} (0.9) + MC (0.006 mA/MHz) f

Where:

MC_{HP} = Macrocells in high-performance mode

MC_{LP} = Macrocells in low-power mode

MC = Total number of macrocells used

f = Clock frequency (MHz)

Figure 1 shows a typical calculation for the XC9572 device.

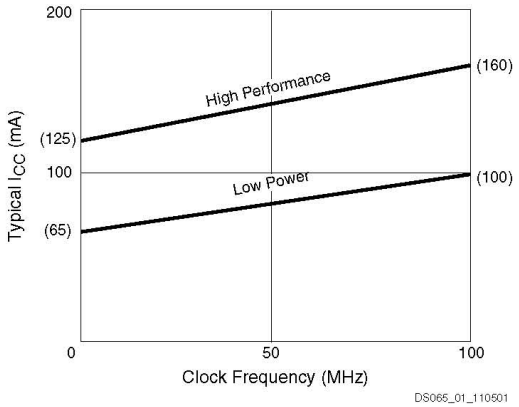


Figure 1: **Typical I_{CC} vs. Frequency for XC9572**

Εικόνα Γ.1 Σελίδα 1 φύλλων δεδομένων για το xc9572

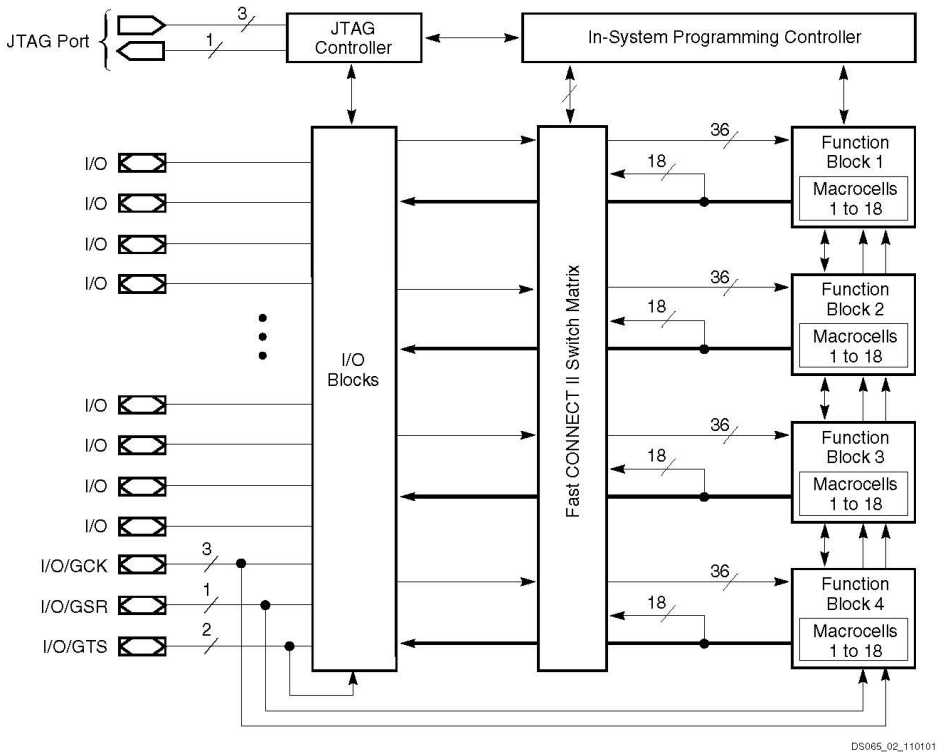


Figure 2: **XC9572 Architecture**
Function block outputs (indicated by the bold line) drive the I/O blocks directly.

Εικόνα Γ.2 Σελίδα 2 φύλλων δεδομένων για το xc9572

Absolute Maximum Ratings

Symbol	Description	Value	Units
V_{CC}	Supply voltage relative to GND	–0.5 to 7.0	V
V_{IN}	Input voltage relative to GND	–0.5 to V_{CC} + 0.5	V
V_{TS}	Voltage applied to 3-state output	–0.5 to V_{CC} + 0.5	V
T_{STG}	Storage temperature (ambient)	–65 to +150	°C
T_J	Junction temperature	+150	°C

Notes:
1. Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those listed under Operating Conditions is not implied. Exposure to Absolute Maximum Ratings conditions for extended periods of time may affect device reliability.

Recommended Operation Conditions

Symbol	Parameter		Min	Max	Units
V_{CCINT}	Supply voltage for internal logic and input buffers	Commercial T_A = 0°C to 70°C	4.75	5.25	V
		Industrial T_A = –40°C to +85°C	4.5	5.5	
V_{CCIO}	Supply voltage for output drivers for 5V operation	Commercial T_A = 0°C to 70°C	4.75	5.25	V
		Industrial T_A = –40°C to +85°C	4.5	5.5	
	Supply voltage for output drivers for 3.3V operation		3.0	3.6	
V_{IL}	Low-level input voltage		0	0.80	V
V_{IH}	High-level input voltage		2.0	V_{CCINT} + 0.5	V
V_O	Output voltage		0	V_{CCIO}	V

Quality and Reliability Characteristics

Symbol	Parameter	Min	Max	Units
T_{DR}	Data Retention	20	-	Years
N_{PE}	Program/Erase Cycles (Endurance)	10,000	-	Cycles

DC Characteristic Over Recommended Operating Conditions

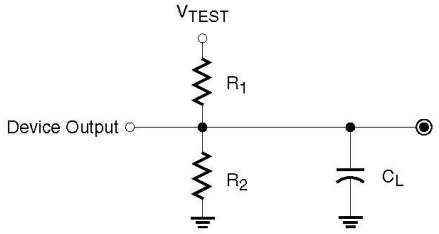
Symbol	Parameter	Test Conditions	Min	Max	Units
V_{OH}	Output high voltage for 5V outputs	I_{OH} = –4.0 mA, V_{CC} = Min	2.4	-	V
	Output high voltage for 3.3V outputs	I_{OH} = –3.2 mA, V_{CC} = Min	2.4	-	V
V_{OL}	Output low voltage for 5V outputs	I_{OL} = 24 mA, V_{CC} = Min	-	0.5	V
	Output low voltage for 3.3V outputs	I_{OL} = 10 mA, V_{CC} = Min	-	0.4	V
I_{IL}	Input leakage current	V_{CC} = Max V_{IN} = GND or V_{CC}	-	±10	µA
I_{IH}	I/O high-Z leakage current	V_{CC} = Max V_{IN} = GND or V_{CC}	-	±10	µA
C_{IN}	I/O capacitance	V_{IN} = GND f = 1.0 MHz	-	10	pF
I_{CC}	Operating supply current (low power mode, active)	V_I = GND, No load f = 1.0 MHz	65 (Typical)		mA

Εικόνα Γ.3 Σελίδα 3 φύλλων δεδομένων για το xc9572

AC Characteristics

Symbol	Parameter	XC9572-7		XC9572-10		XC9572-15		Units
		Min	Max	Min	Max	Min	Max	
T_{PD}	I/O to output valid	-	7.5	-	10.0	-	15.0	ns
T_{SU}	I/O setup time before GCK	4.5	-	6.0	-	8.0	-	ns
T_{H}	I/O hold time after GCK	0	-	0	-	0	-	ns
T_{CO}	GCK to output valid	-	4.5	-	6.0	-	8.0	ns
f_{CNT}(1)	16-bit counter frequency	125.0	-	111.1	-	95.2	-	MHz
f_{SYSTEM}(2)	Multiple FB internal operating frequency	83.3	-	66.7	-	55.6	-	MHz
T_{PSU}	I/O setup time before p-term clock input	0.5	-	2.0	-	4.0	-	ns
T_{PH}	I/O hold time after p-term clock input	4.0	-	4.0	-	4.0	-	ns
T_{PCO}	P-term clock output valid	-	8.5	-	10.0	-	12.0	ns
T_{OE}	GTS to output valid	-	5.5	-	6.0	-	11.0	ns
T_{OD}	GTS to output disable	-	5.5	-	6.0	-	11.0	ns
T_{POE}	Product term OE to output enabled	-	9.5	-	10.0	-	14.0	ns
T_{POD}	Product term OE to output disabled	-	9.5	-	10.0	-	14.0	ns
T_{WLH}	GCK pulse width (High or Low)	4.0	-	4.5	-	5.5	-	ns
T_{APRPW}	Asynchronous preset/reset pulse width (High or Low)	7.0	-	7.5	-	8.0	-	ns

Notes:
1. f_{CNT} is the fastest 16-bit counter frequency available, using the local feedback when applicable. f_{CNT} is also the Export Control Maximum flip-flop toggle rate, f_{TOG}.
2. f_{SYSTEM} is the internal operating frequency for general purpose system designs spanning multiple FBs.



Output Type	V_{CCIO}	V_{TEST}	R_1	R_2	C_L
	5.0V	5.0V	160Ω	120Ω	35 pF
	3.3V	3.3V	260Ω	360Ω	35 pF

DS067_03_110101

Figure 3: **AC Load Circuit**

Εικόνα Γ.4 Σελίδα 4 φύλλων δεδομένων για το xc9572

Internal Timing Parameters

Symbol	Parameter	XC9572-7		XC9572-10		XC9572-15		Units
		Min	Max	Min	Max	Min	Max	
Buffer Delays								
T_{IN}	Input buffer delay	-	2.5	-	3.5	-	4.5	ns
T_{GCK}	GCK buffer delay	-	1.5	-	2.5	-	3.0	ns
T_{GSR}	GSR buffer delay	-	4.5	-	6.0	-	7.5	ns
T_{GTS}	GTS buffer delay	-	5.5	-	6.0	-	11.0	ns
T_{OUT}	Output buffer delay	-	2.5	-	3.0	-	4.5	ns
T_{EN}	Output buffer enable/disable delay	-	0	-	0	-	0	ns
Product Term Control Delays								
T_{PTCK}	Product term clock delay	-	3.0	-	3.0	-	2.5	ns
T_{PTSR}	Product term set/reset delay	-	2.0	-	2.5	-	3.0	ns
T_{PTTS}	Product term 3-state delay	-	4.5	-	3.5	-	5.0	ns
Internal Register and Combinatorial Delays								
T_{PDI}	Combinatorial logic propagation delay	-	0.5	-	1.0	-	3.0	ns
T_{SUI}	Register setup time	1.5	-	2.5	-	3.5	-	ns
T_{HI}	Register hold time	3.0	-	3.5	-	4.5	-	ns
T_{COI}	Register clock to output valid time	-	0.5	-	0.5	-	0.5	ns
T_{AOI}	Register async. S/R to output delay	-	6.5	-	7.0	-	8.0	ns
T_{RAI}	Register async. S/R recover before clock	7.5	-	10.0	-	10.0	-	ns
T_{LOGI}	Internal logic delay	-	2.0	-	2.5	-	3.0	ns
T_{LOGILP}	Internal low power logic delay	-	10.0	-	11.0	-	11.5	ns
Feedback Delays								
T_F	FastCONNECT feedback delay	-	8.0	-	9.5	-	11.0	ns
T_{LF}	Function block local feedback delay	-	4.0	-	3.5	-	3.5	ns
Time Adders								
T_{PTA}(1)	Incremental product term allocator delay	-	1.0	-	1.0	-	1.0	ns
T_{SLEW}	Slew-rate limited delay	-	4.0	-	4.5	-	5.0	ns

Notes:
1. T_{PTA} is multiplied by the span of the function as defined in the XC9500 family data sheet.

Εικόνα Γ.5 Σελίδα 5 φύλλων δεδομένων για το xc9572

XC9572 I/O Pins

Function Block	Macro-cell	PC44	PC84	PQ100	TQ100	BScan Order
1	1	–	4	18	16	213
1	2	1	1	15	13	210
1	3	–	6	20	18	207
1	4	–	7	22	20	204
1	5	2	2	16	14	201
1	6	3	3	17	15	198
1	7	–	11	27	25	195
1	8	4	5	19	17	192
1	9	5[1]	9[1]	24[1]	22[1]	189
1	10	–	13	30	28	186
1	11	6[1]	10[1]	25[1]	23[1]	183
1	12	–	18	35	33	180
1	13	–	20	38	36	177
1	14	7[1]	12[1]	29[1]	27[1]	174
1	15	8	14	31	29	171
1	16	–	23	41	39	168
1	17	9	15	32	30	165
1	18	–	24	42	40	162
2	1	–	63	89	87	159
2	2	35	69	96	94	156
2	3	–	67	93	91	153
2	4	–	68	95	93	150
2	5	36	70	97	95	147
2	6	37	71	98	96	144
2	7	–	76[2]	5[2]	3[2]	141
2	8	38	72	99	97	138
2	9	39[1]	74[1]	1[1]	99[1]	135
2	10	–	75	3	1	132
2	11	40[1]	77[1]	6[1]	4[1]	129
2	12	–	79	8	6	126
2	13	–	80	10	8	123
2	14	42[3]	81[3]	11[3]	9[3]	120
2	15	43	83	13	11	117
2	16	–	82	12	10	114
2	17	44	84	14	12	111
2	18	–	–	94	92	108
3	1	–	25	43	41	105
3	2	11	17	34	32	102
3	3	–	31	51	49	99
3	4	–	32	52	50	96
3	5	12	19	37	35	93
3	6	–	34	55	53	90
3	7	–	35	56	54	87
3	8	13	21	39	37	84
3	9	14	26	44	42	81
3	10	–	40	62	60	78
3	11	18	33	54	52	75
3	12	–	41	63	61	72
3	13	–	43	65	63	69
3	14	19	36	57	55	66
3	15	20	37	58	56	63
3	16	–	45	67	65	60
3	17	22	39	60	58	57
3	18	–	–	61	59	54
4	1	–	46	68	66	51
4	2	24	44	66	64	48
4	3	–	51	73	71	45
4	4	–	52	74	72	42
4	5	25	47	69	67	39
4	6	–	54	78	76	36
4	7	–	55	79	77	33
4	8	26	48	70	68	30
4	9	27	50	72	70	27
4	10	–	57	83	81	24
4	11	28	53	76	74	21
4	12	–	58	84	82	18
4	13	–	61	87	85	15
4	14	29	56	80	78	12
4	15	33	65	91	89	9
4	16	–	62	88	86	6
4	17	34	66	92	90	3
4	18	–	–	81	79	0

Notes:
1. Global control piN.
2. Global control pin GTS1 for PC84, PQ100, and TQ100.
3. Global control pin GTS1 for PC44.

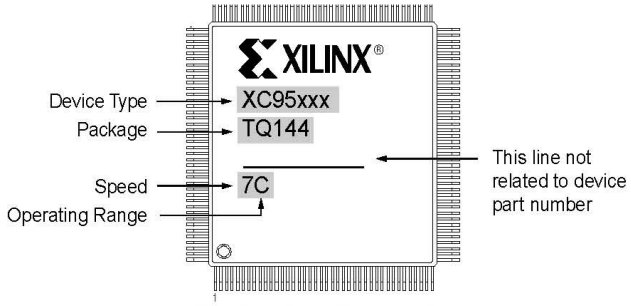
Εικόνα Γ.6 Σελίδα 6 φύλλων δεδομένων για το xc9572

XC9572 Global, JTAG and Power Pins

Pin Type	PC44	PC84	PQ100	TQ100
I/O/GCK1	5	9	24	22
I/O/GCK2	6	10	25	23
I/O/GCK3	7	12	29	27
I/O/GTS1	42	76	5	3
I/O/GTS2	40	77	6	4
I/O/GSR	39	74	1	99
TCK	17	30	50	48
TDI	15	28	47	45
TDO	30	59	85	83
TMS	16	29	49	47
V_{CCINT} 5V	21,41	38,73,78	7,59,100	5,57,98
V_{CCIO} 3.3V/5V	32	22,64	28,40,53,90	26,38,51,88
GND	10,23,31	8,16,27,42, 49,60	2,23,33,46,64,71, 77,86	100,21,31,44,62,69, 75, 84
No Connects	-	-	4,9,21,26,36,45,48, 75, 82	2,7,19,24,34,43,46, 73, 80

Εικόνα Γ.7 Σελίδα 7 φύλλων δεδομένων για το xc9572

Device Part Marking and Ordering Combination Information



Sample package with part marking.

Device Ordering and Part Marking Number	Speed (pin-to-pin delay)	Pkg. Symbol	No. of Pins	Package Type	Operating Range(1)
XC9572-7PC44C	7.5 ns	PC44	44-pin	Plastic Lead Chip Carrier (PLCC)	C
XC9572-7PCG44C	7.5 ns	PCG44	44-pin	Plastic Lead Chip Carrier (PLCC); Pb-Free	C
XC9572-7PC84C	7.5 ns	PC84	84-pin	Plastic Lead Chip Carrier (PLCC)	C
XC9572-7PCG84C	7.5 ns	PCG84	84-pin	Plastic Lead Chip Carrier (PLCC); Pb-Free	C
XC9572-7PQ100C	7.5 ns	PQ100	100-pin	Plastic Quad Flat Pack (PQFP)	C
XC9572-7PQG100C	7.5 ns	PQG100	100-pin	Plastic Quad Flat Pack (PQFP); Pb-Free	C
XC9572-7TQ100C	7.5 ns	TQ100	100-pin	Thin Quad Flat Pack (TQFP)	C
XC9572-7TQG100C	7.5 ns	TQG100	100-pin	Thin Quad Flat Pack (TQFP); Pb-Free	C
XC9572-10PC44C	10 ns	PC44	44-pin	Plastic Lead Chip Carrier (PLCC)	C
XC9572-10PCG44C	10 ns	PCG44	44-pin	Plastic Lead Chip Carrier (PLCC); Pb-Free	C
XC9572-10PC84C	10 ns	PC84	84-pin	Plastic Lead Chip Carrier (PLCC)	C
XC9572-10PCG84C	10 ns	PCG84	84-pin	Plastic Lead Chip Carrier (PLCC); Pb-Free	C
XC9572-10PQ100C	10 ns	PQ100	100-pin	Plastic Quad Flat Pack (PQFP)	C
XC9572-10PQG100C	10 ns	PQG100	100-pin	Plastic Quad Flat Pack (PQFP); Pb-Free	C
XC9572-10TQ100C	10 ns	TQ100	100-pin	Thin Quad Flat Pack (TQFP)	C
XC9572-10TQG100C	10 ns	TQG100	100-pin	Thin Quad Flat Pack (TQFP); Pb-Free	C
XC9572-10PC44I	10 ns	PC44	44-pin	Plastic Lead Chip Carrier (PLCC)	I
XC9572-10PCG44I	10 ns	PCG44	44-pin	Plastic Lead Chip Carrier (PLCC); Pb-Free	I
XC9572-10PC84I	10 ns	PC84	84-pin	Plastic Lead Chip Carrier (PLCC)	I
XC9572-10PCG84I	10 ns	PCG84	84-pin	Plastic Lead Chip Carrier (PLCC); Pb-Free	I
XC9572-10PQ100I	10 ns	PQ100	100-pin	Plastic Quad Flat Pack (PQFP)	I
XC9572-10PQG100I	10 ns	PQG100	100-pin	Plastic Quad Flat Pack (PQFP); Pb-Free	I
XC9572-10TQ100I	10 ns	TQ100	100-pin	Thin Quad Flat Pack (TQFP)	I
XC9572-10TQG100I	10 ns	TQG100	100-pin	Thin Quad Flat Pack (TQFP); Pb-Free	I
XC9572-15PC44C	15 ns	PC44	44-pin	Plastic Lead Chip Carrier (PLCC)	C
XC9572-15PCG44C	15 ns	PCG44	44-pin	Plastic Lead Chip Carrier (PLCC); Pb-Free	C
XC9572-15PC84C	15 ns	PC84	84-pin	Plastic Lead Chip Carrier (PLCC)	C
XC9572-15PCG84C	15 ns	PCG84	84-pin	Plastic Lead Chip Carrier (PLCC); Pb-Free	C
XC9572-15PQ100C	15 ns	PQ100	100-pin	Plastic Quad Flat Pack (PQFP)	C
XC9572-15PQG100C	15 ns	PQG100	100-pin	Plastic Quad Flat Pack (PQFP); Pb-Free	C
XC9572-15TQ100C	15 ns	TQ100	100-pin	Thin Quad Flat Pack (TQFP)	C
XC9572-15TQG100C	15 ns	TQG100	100-pin	Thin Quad Flat Pack (TQFP); Pb-Free	C
XC9572-15PC44I	15 ns	PC44	44-pin	Plastic Lead Chip Carrier (PLCC)	I

Εικόνα Γ.8 Σελίδα 8 φύλλων δεδομένων για το xc9572

Device Ordering and Part Marking Number	Speed (pin-to-pin delay)	Pkg. Symbol	No. of Pins	Package Type	Operating Range(1)
XC9572-15PCG44I	15 ns	PCG44	44-pin	Plastic Lead Chip Carrier (PLCC); Pb-Free	I
XC9572-15PC84I	15 ns	PC84	84-pin	Plastic Lead Chip Carrier (PLCC)	I
XC9572-15PCG84I	15 ns	PCG84	84-pin	Plastic Lead Chip Carrier (PLCC); Pb-Free	I
XC9572-15PQ100I	15 ns	PQ100	100-pin	Plastic Quad Flat Pack (PQFP)	I
XC9572-15PQG100I	15 ns	PQG100	100-pin	Plastic Quad Flat Pack (PQFP); Pb-Free	I
XC9572-15TQ100I	15 ns	TQ100	100-pin	Thin Quad Flat Pack (TQFP)	I
XC9572-15TQG100I	15 ns	TQG100	100-pin	Thin Quad Flat Pack (TQFP); Pb-Free	I

Notes:
1. C = Commercial: T_A = 0° to +70°C; I = Industrial: T_A = –40° to +85°C

Warranty Disclaimer

THESE PRODUCTS ARE SUBJECT TO THE TERMS OF THE XILINX LIMITED WARRANTY WHICH CAN BE VIEWED AT http://www.xilinx.com/warranty.htm. THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY USE OF THE PRODUCTS IN AN APPLICATION OR ENVIRONMENT THAT IS NOT WITHIN THE SPECIFICATIONS STATED ON THE THEN-CURRENT XILINX DATA SHEET FOR THE PRODUCTS. PRODUCTS ARE NOT DESIGNED TO BE FAIL-SAFE AND ARE NOT WARRANTED FOR USE IN APPLICATIONS THAT POSE A RISK OF PHYSICAL HARM OR LOSS OF LIFE. USE OF PRODUCTS IN SUCH APPLICATIONS IS FULLY AT THE RISK OF CUSTOMER SUBJECT TO APPLICABLE LAWS AND REGULATIONS.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/04/98	3.0	Update AC characteristics and internal parameters.
06/18/03	4.0	Updated format.
08/21/03	4.1	Updated Package Device Marking Pin 1 orientation.
04/15/05	4.2	Added asynchronous preset/reset pulse width specification (T_{APRPW})
04/03/06	4.3	Added Warranty Disclaimer. Added Pb-Free package information.

Εικόνα Γ.9 Σελίδα 9 φύλλων δεδομένων για το xc9572

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Α. Θ. Κοσσίδας, Π. Ηρ. Γιαννακόπουλος, *Αριθμητικά συστήματα και ψηφιακά κυκλώματα*, έκδοση 1η, Εκδόσεις Ν. Τεχνολογιών, Αθήνα 2006, ISBN 960-8105-91-9.

[2] Δ. Πογαρίδης, *Σχεδίαση Ψηφιακών Συστημάτων*, έκδοση 3η, Εκδόσεις «ΙΩΝ», Αθήνα 2004, ISBN 960-411-470-0.

[3] Σ. Ι. Σουραβλάς και Μ. Ρουμελιώτης, *Ψηφιακά Συστήματα: Μοντελοποίηση & Προσομοίωση με τη γλώσσα VHDL*, Εκδόσεις Τζιόλα, Θεσσαλονίκη 2008, ISBN 978-960-418-155-1.

[4] Χ. Β. Τζίκας, *Εργαστηριακές Ασκήσεις Ψηφιακών Κυκλωμάτων Ι*, Τμήμα Ηλεκτρονικής Σ.Τ.ΕΦ. Α.Τ.Ε.Ι. Θεσσαλονίκης, Σίνδος 2005.

[5] Χ. Β. Τζίκας, *Εργαστηριακές Ασκήσεις Ψηφιακών Κυκλωμάτων ΙΙ*, Τμήμα Ηλεκτρονικής Σ.Τ.ΕΦ. Α.Τ.Ε.Ι. Θεσσαλονίκης, Σίνδος 2008.

[6] S. Brown, Z. Vranesic, *Σχεδίαση Ψηφιακών Συστημάτων με τη γλώσσα VHDL*, Εκδόσεις Τζιόλα, Θεσσαλονίκη 2001, ISBN 960-8050-50-2.

[7] T. M. Floyd, *Ψηφιακά Ηλεκτρονικά*, έκδοση 1η, Εκδόσεις «ΙΩΝ», Αθήνα 2007, ISBN 978-960-411-646-1.

[8] I. Grout, *Digital System Design with FPGAs and CPLDs,* Elsevier Ltd, 2008, ISBN 978-0-7506-8397-5.

[9] M. M. Mano, *Ψηφιακή Σχεδίαση*, έκδοση 2η, Εκδόσεις Παπασωτηρίου, Αθήνα 1992, ISBN 960-7182-01-4.

[10] V. A. Pedroni, *Σχεδιασμός κυκλωμάτων με τη VHDL*, Εκδόσεις Κλειδάριθμος, Αθήνα 2007, ISBN 978-960-461-118-8.

[11] Xilinx, *ISE 9 In-Depth Tutorial*, Xilinx, Inc. "http://download.xilinx.com/direct/ise9_tutorials/ise9tut.pdf"

[12] Xilinx data sheet xc9572_ds065, Xilinx, Inc. "http://www.xilinx.com/support/documentation/datasheets/ds065.pdf"

ΠΛΗΡΟΦΟΡΙΕΣ

Η επιτυχής πρόσβαση στο αρχείο kmkokidis_ptyxiaki.pdf απαιτεί αριθμό 5.0 ή νεώτερο από όλες τις εκδόσεις Adobe® Acrobat™.

Στον φάκελο Video περιλαμβάνονται αρχεία εκτέλεσης εφαρμογών κινούμενης εικόνας (video). Στον υποφάκελο «Ασύγχρονος» υπάρχουν συγκεκριμένα τα αρχεία της δημιουργίας, της προσομοίωσης και προγραμματισμού στο σχηματικό για λειτουργία του ολοκληρωμένου σαν ασύγχρονος μετρητής mod-8 αρνητικής διέγερσης. Στον φάκελο «Πολυπλέκτης» υπάρχουν συγκεκριμένα τα αρχεία της δημιουργίας, της προσομοίωσης και προγραμματισμού τόσο στο σχηματικό όσο και τη VHDL για λειτουργία του ολοκληρωμένου σαν πολυπλέκτης οχτώ προς ένα. (Στοιχείο διαχωρισμού: η κατάληξη «_SCH» δηλώνει σχηματικό ενώ η κατάληξη «_VHDL» δηλώνει την περιγραφική γλώσσα VHDL.)

Με διπλό κλικ του ποντικιού ενεργοποιείτε τις παραπάνω εφαρμογές. Η χρήση του συνδυασμού του πληκτρολογίου alt+enter ενεργοποιεί ή απενεργοποιεί την ταινία διαχείρισης της κινούμενης εικόνας. Η χρήση του πλήκτρου space βοηθά στην παύση ή τη συνέχιση εξέλιξης της κινούμενης εικόνας.

Στον φάκελο CPLD υπάρχουν υποφάκελοι με όλα τα αρχεία των εργασιών όπως περιγράφονται στην πτυχιακή εργασία.

Κωνσταντίνος Κοκίδης