ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και προσομοίωση λειτουργίας του VSM επεξεργαστή σε γλώσσα VHDL

Ευθύμιος Ι. Μάσσιος
Α.Μ. 45585

Εισηγητής: Δρ Παναγιώτης Γιαννακόπουλος, Καθηγητής

(Κενό φύλλο)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος/η ...Μάσσιος Ευθύμιος..... του...Ιωάννη., με αριθμό μητρώου ...45585..... φοιτητής του ..Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών ..της Σχολής...Μηχανικών..... του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι ..... και έπειτα από αίτηση μου στη Βιβλιοθήκη και έγκριση του επιβλέποντα καθηγητή.

Ο/Η Δηλών/ούσα

Η εξεταστική επιτροπή

Panagiotis Giannakopoulos

Digitally signed by Panagiotis Giannakopoulos
Date: 2023.03.23 11:28:19 +02'00'

Π. ΓΙΑΝΝΑΚΟΠΟΥΛΟΣ
ΚΑΘΗΓΗΤΗΣ

Γ. ΠΡΕΖΕΡΑΚΟΣ
ΚΑΘΗΓΗΤΗΣ

Georgios Prezerakos

Digitally signed by Georgios Prezerakos
Date: 2023.03.27 13:06:49 +03'00'

Δ. ΝΙΚΟΠΟΛΟΥΛΟΣ
ΚΑΘΗΓΗΤΗΣ

Dimitrios Nikolopoulos

Digitally signed by Dimitrios Nikolopoulos
Date: 2023.04.27 13:27:04 +03'00'

Η έγκριση της διπλωματικής εργασίας δεν υποδηλοί την αποδοχή των γνωμών του συγγραφέα. Κατά τη συγγραφή τηρήθηκαν οι αρχές της ακαδημαϊκής δεοντολογίας.

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της αρχιτεκτονικής των υπολογιστών. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, τον οποίο θα ήθελα να ευχαριστήσω.

Ακόμα θα ήθελα να ευχαριστήσω τους γονείς μου για την αμέριστη ηθική και υλική υποστήριξη κα'θόλη τη διάρκεια των σπουδών μου

(Κενό φύλλο)

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ασχολείται με το σχεδιασμό και την προσομοίωση λειτουργίας επεξεργαστή VSM σε γλώσσα VHDL. Κύριο χαρακτηριστικό των επεξεργαστών αυτής της οικογένειας είναι οι μικρές τους διαστάσεις αλλά και η απλότητα του σχεδιασμού τους.

ABSTRACT

The present thesis concerns the design and simulation of VSM processor operation in VHDL language. The main feature of the processors of this family is their small dimensions but also the simplicity of their design.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Αρχιτεκτονική Υπολογιστών
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: επεξεργαστής, VHDL, VSM

Περιεχόμενα

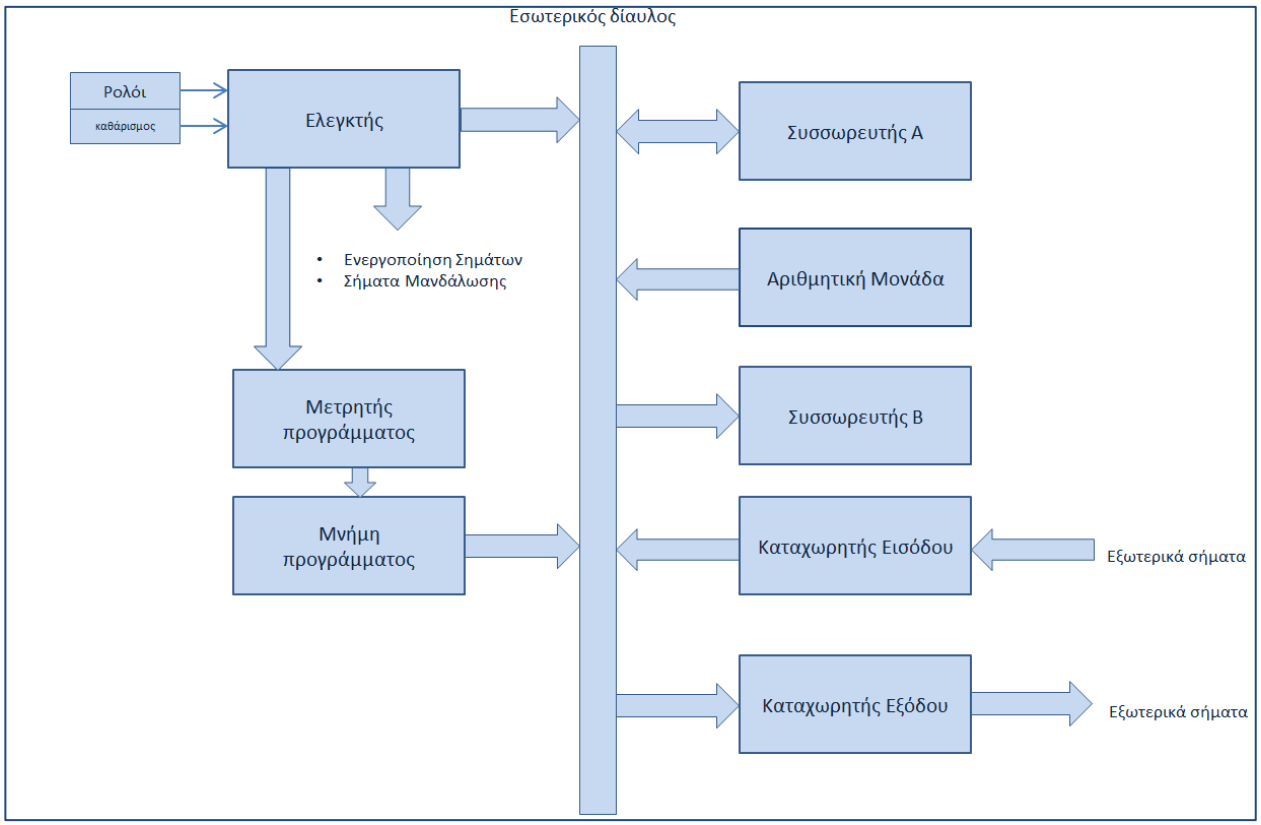
ΚΕΦΑΛΑΙΟ 1- Ο ΕΠΕΞΕΡΓΑΣΤΗΣ VSM

1.1 Εισαγωγή στον επεξεργαστή

Σε αυτή την εργασία θα κατασκευαστεί ένας πολύ απλός μικροεπεξεραστής 4-bit που έχει υλοποιηθεί σε 5 εντολές .Αυτό δίνει τα θεμέλια για τηΣ κατασκευή πιο πολύπλοκων επεξεργαστών με εκτεταμένο σύνολο εντολών, πιο εξελιγμένες ανταλλαγές μεταξύ της κύριας μνήμης και των συσσωρευτών και πιο ισχυρή αριθμητική μονάδα προκειμένου να κατασκευαστεί ένας πιο προσιτός μικροεπεξεργαστής

1.2 Περιγραφή επεξεργαστή

Ο πολύ απλός μικροεπεξεργαστής είνια μια αναβαθμισμένη έκδοση της πολυ δημοφιλούς αρχιτεκτονικής SAP(Simple As Possible) και προτάθηκε απο τον Albert P. Malvino [RefBook] το 1993 στο διάσημο βιβλίο του "Digital Electronics Electronics". Ο υπολογιστής VSM εισάγει τις βασικές έννοιες της αρχιτεκτονικής μικροεπεξεργαστών με τον απλούστερο τρόπο.Ο VSM είναι αρκετό απλός αλλά παράλληλα αρκετά περίπλοκος.



Εικόνα 1.1 : Βασική αρχιτεκτονική VSM

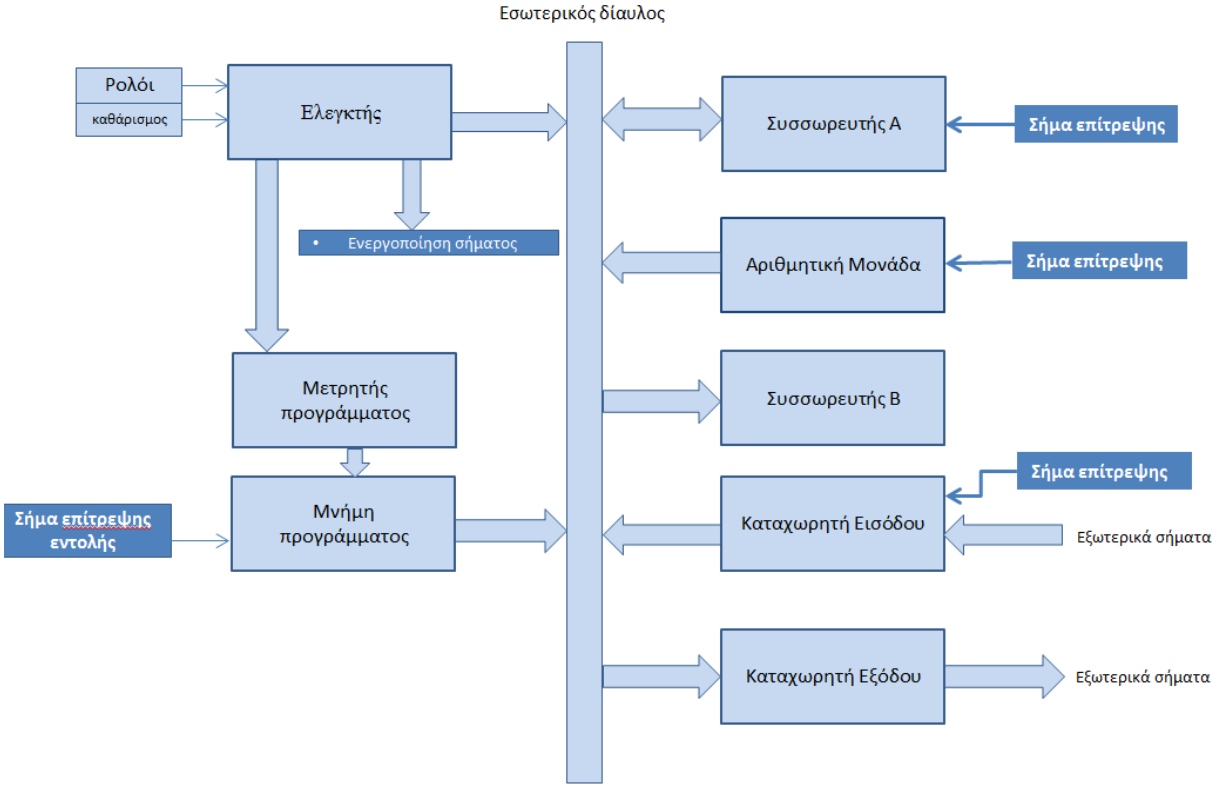
1.3 Ρεπερτόριο εντολών

Ο ρόλος της κάθε δομικής μονάδας φαίνεται στον πίνακα 1.

Πίνακας 1

Block	Block	Μέγεθος (bits)
Μετρητής Προγράμματος	Ο μετρητής προγράμματος μετράει από το 0000 ως το 1111. Κάθε χρονική στιγμή περιέχει την τιμή της ενεργούς διεύθυνσης. Αρχικά, ο μετρητής προγράμματος περιέχει την τιμή 0000, ώστε ο μικροεπεξεργαστής να εκκινεί από την πρώτη εντολή που βρίσκεται στη μνήμης.	**4**
Μνήμη Προγράμματος	Το πρόγραμμα αποθηκεύεται στη μνήμη προγράμματος. Κάθε γραμμή προγράμματος αποτελείται από 8 bits: τα 4 πιο σημαντικά bits περιέχουν στην εντολή, ενώ τα λιγότερα σημαντικά bits περιέχουν τα δεδομένα της εντολής για τις εντολές που περιέχουν τελεστέο.	**8x8**
Συσσωρευτής Α	Ο συσσωρευτής Α αποθηκεύει τα άμεσα αποτελέσματα που υπολογίζονται από τον μικροεπεξεργαστή. Ο συσσωρευτής είναι ένας καταχωρητής 4 bits.	**4**
Συσσωρευτής Β	Ο συσσωρευτής Β χρησιμοποιείται κατα κόρον για να παράγει τον προστιθέμενο ή αφαιρούμενο από τον συσσωρεύτη Β αριθμό με σκοπό να εκτελεστεί η πρόσθεση ή η αφαίρεση.	**4**
Αριθμητική Μονάδα	Η αριθμητική μονάδα εκτελεί τις λειτουργίες S=A+B (Πρόσθεση) ή S=A+~B+1 (Αφαίρεση)	**4**
Καταχωρητής Εισόδου	Με το καταχωρητή εισόδου μεταφέρουμε στον μικροεπεξεργαστή εξωτερικά δεδομένα.	**4**
Καταχωρητής Εξόδου	Ο καταχωρητής εξόδου μεταφέρει περιεχόμενα του εσωτερικού διαύλου στο εξωτερικό περιβάλλον. Αυτή η εντολή εκτελείται με σκοπό τη προβολή του τελικού αποτελέσματος. Στο καταχωρητή εξόδου αποθηκεύονται τα δεδομένα στην αιχμή πτώσης του ρολογιού. Ο καταχωρητής συνδέεται με ένα κύκλωμα το οποίο μεταφέρει ή προβάλλει το αποτέλεσμα στον χρήστη.	**4**

Το κύκλωμα VSM βασίζεται σε ένα δίαυλο που ονομάζεται «εσωτερικός δίαυλος» (IB).Κάθε δομική μονάδα που παρατίθεται στο σχήμα 1.2 μπορεί να πάρει τον έλεγχο του διαύλου, χρησιμοποιώντας ένα συγκεκριμένο σήμα «Ενεργοποίησης». Για παράδειγμα , ο συσσωρευτής Α έχει σήμα ενεργοποίησης EnableA. Όταν το EnableA είναι ενεργοποιημένο , τα 4 bits του συσσωρευτή Α πηγαίνουν στον εσωτερικό δίαυλο.



Εικόνα 1.2: Ο ελεγκτής δημιουργεί σήματα "Ενεργοποίησης" που επιτρέπουν σε μια δομική μονάδα να πάρει τον έλεγχο του διαύλου

Στο Πίνακα 2 περιγράφεται ο έλεγχος του εσωτερικού διαύλου των δομικών μονάδων μέσω του σήματος ¨Ενεργοποίησης¨.

Σήμα Enable	Περιγραφή
Enable A	Δίνεται στον Α η δυνατότητα να πάρει τον έλεγχο του διαύλου.
Enable Alu	Μεταφέρει το αποτέλεσμα της αριθμητικής πράξης (πρόσθεση ή αφαίρεση) στον εσωτερικό δίαυλο.
EnableInstr	Μεταφέρει τα δεδομένα της εντολής (4 λιγότερο σημαντικά ψηφία) στον εσωτερικό δίαυλο.
EnableIn	Μεταφέρει τα περιεχόμενα του εξωτερικού καταχωρητή στον εσωτερικό δίαυλο.

Πίνακας 2

Οι βασικές εντολές παρατίθενται παρακάτω. Καθώς η εντολή κωδικοποιείται σε 4-bit, μπορούν να ληφθούν μόνο 16 διαφορετικές εντολές.

1.3.1 No Operation (NOP=0000)

Η συγκεκριμένη εντολή δεν έχει καμία επίδραση στον εσωτερικό καταχωρητή. Ωστόσο, αυτή η εντολή είναι πολύ σημαντική για τη κατανόηση του τρόπου με τον οποίο λειτουργούν οι βασικοί έλεγχοι ρολογιού.

1.3.2 Addition (ADD=0001)

Το περιεχόμενο του συσσωρευτή Α προστίθεται στα δεδομένα που παρέχονται ως παράμετρο και το αποτέλεσμα ενημερώνει τον συσσωρευτή Α.Η προσθήκη πραγματοποιείται στα τέσσερα ψηφία . Για παράδειγμα λαμβάνοντας ότι Α=2 , η εντολή “ADD 3” αντιστοιχεί στο Α=Α+3, δηλαδή Α=2+3. Η τελική τιμή του Α είναι 5.

1.3.3 Subtraction (SUB=0010)

Το περιεχόμενο του συσσωρευτή A αφαιρείται από δεδομένα που δίνονται ως παράμετρος και το αποτέλεσμα ενημερώνει τον συσσωρευτή A .Η αφαίρεση πραγματοποιείται στα 4 ψηφία.

1.3.4 Get Input (IN=0011)

Στην περίπτωση κατά την οποία θέλουμε να μεταφέρουμε τα περιεχόμενα της πύλης στον συσσωρευτή A τότε εκτελούμε την εντολή με κωδικό 0011.

1.3.5 Give Output (OUT=0100)

Ο συσσωρευτής A αποθηκεύεται στη πύλη εξόδου. Η πύλη εξόδου είναι ένας καταχωρητής τεσσάρων δυαδικών ψηφίων που απομνημονεύει τη τιμή εξόδου και διατηρεί τη μόνιμη διαθεσιμότητα έως ότου ανανεωθεί το περιεχόμενο του με μια νέα εντολή «Δώστε έξοδο»

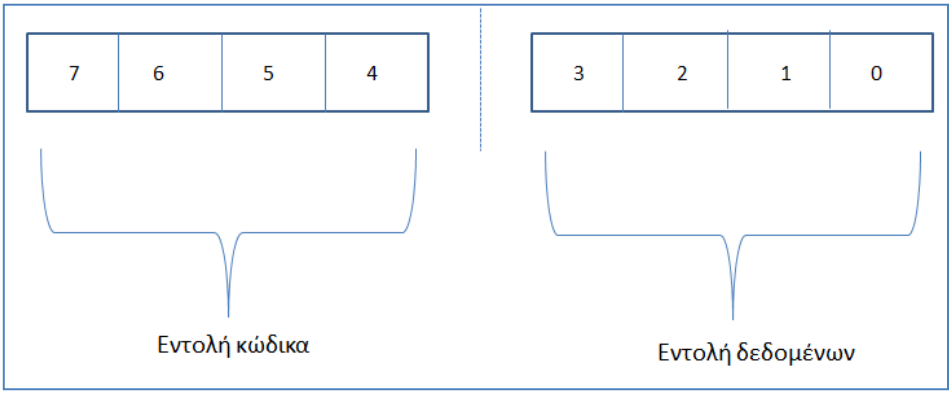
1.3.6 Load Instruction (LDA=0101)

Το LDA σημαίνει «φορτώστε τον συσσωρευτή A με μια τιμή». Για παράδειγμα , η εντολή LDA 9 μεταφέρει τη τιμή 9(1001 σε δυαδική μορφή) στον συσσωρευτή A.

1.4 Μνήμη προγράμματος

Η μνήμη του προγράμματος περιέχει μέχρι 8 byte, όπου αποθηκεύουμε τις εντολές που πρέπει να εκτελεστούν. Κάθε εντολή κωδικοποιείται σε 8 bits.Τα πιο σημαντικά bits αντιστοιχούν στην ίδια την εντολή, ενώ τα λιγότερα σημιτικά bits είναι τα δεδομένα. Το ακόλουθο πρόγραμμα φορτώνει τον συσσωρευτή A με τη τιμή «2»,στη συνέχεια προσθέτει «1» και τοποθετεί το αποτέλεσμα στον καταχωρητή εξόδου.

Το σύμβολο μνήμης παρουσιάζεται στο εικόνα 1.4. Η μνήμη έχει 8 καταχωρτητές, κάθε καταχωρητής έχει 8 στοιχειώδη στοιχεία μνήμης. Μπορούν να αλλαχτούν το περιεχόμενα της εσωτερικής μνήμης με ένα κλικ στο επιθυμητό κελί λογικής. Όταν αποθηκευτεί το σχηματικό διάγραμμα ,αποθηκεύεται επίσης τα περιεχόμενα της μνήμης. Το σύμβολο μνήμης μπορεί να βρεθεί στη βασική παλέτα συμβόλων.



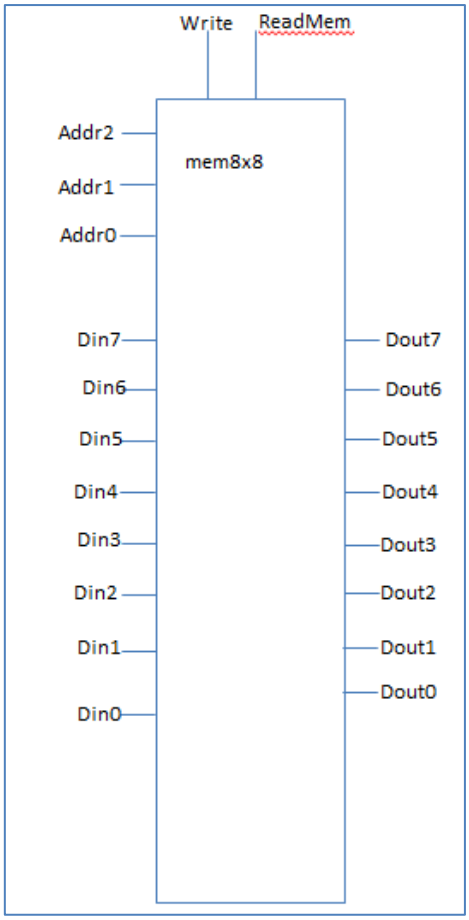
Εικόνα 1.3: Κωδικοποίηση εντολής

Πίνακας 3

Πίνακας 3

Mnemonic	OpCode(δυαδικός)	OpCode(δεκαεξαδικός)
LDA 2	0101|0010	0x52
ADD 1	0010|0001	0x21
OUT	0100|0000	0x40
NOP	0000|0000	0x00

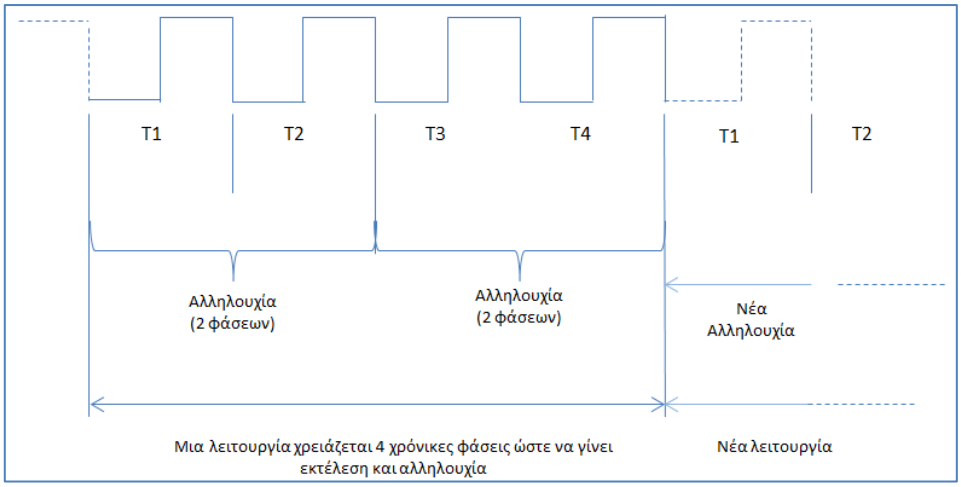
Στο πίνακα 3 περιλαμβάνεται η μνήμη που περιέχει 8bit πληροφορίας και χωρίζεται σε δύο μέρη: τα 4 πιο σημαντικά δυαδικά ψηφία για το κώδικα εντολής και τα λιγότερα σημαντική ψηφία για τα δεδομένα.



Εικόνα 1.4: Αποθήκευση του προγράμματος στη μνήμη

1.5 Εκτέλεση εντολών

Οι μηχανισμοί για τη εκτελέση των εντολών βασλιζονται σε εσωτερικές μικροεπεξεργασίες.Η εκτέλεση κάθε εντολής βασίζεται σε τέσσερις διαφορετικές χρονικές φάσεις,όπως απεικονίζεται στην εικόνα 1.5.Ο χρήστης θα πρέπει να κάνει τη διάκριση μεταξύ της ίδιας της εντολής του μικροεπεξεργαστή όπως ‘’LDA 2’’ και των τεσσάρων εσωτερικών εργαλείων που απαιτούνται για την ολοκλήρωση της εντολής ‘’LDA 2’’ ,που ονομάζεται φάση 0,1,2 και 4.



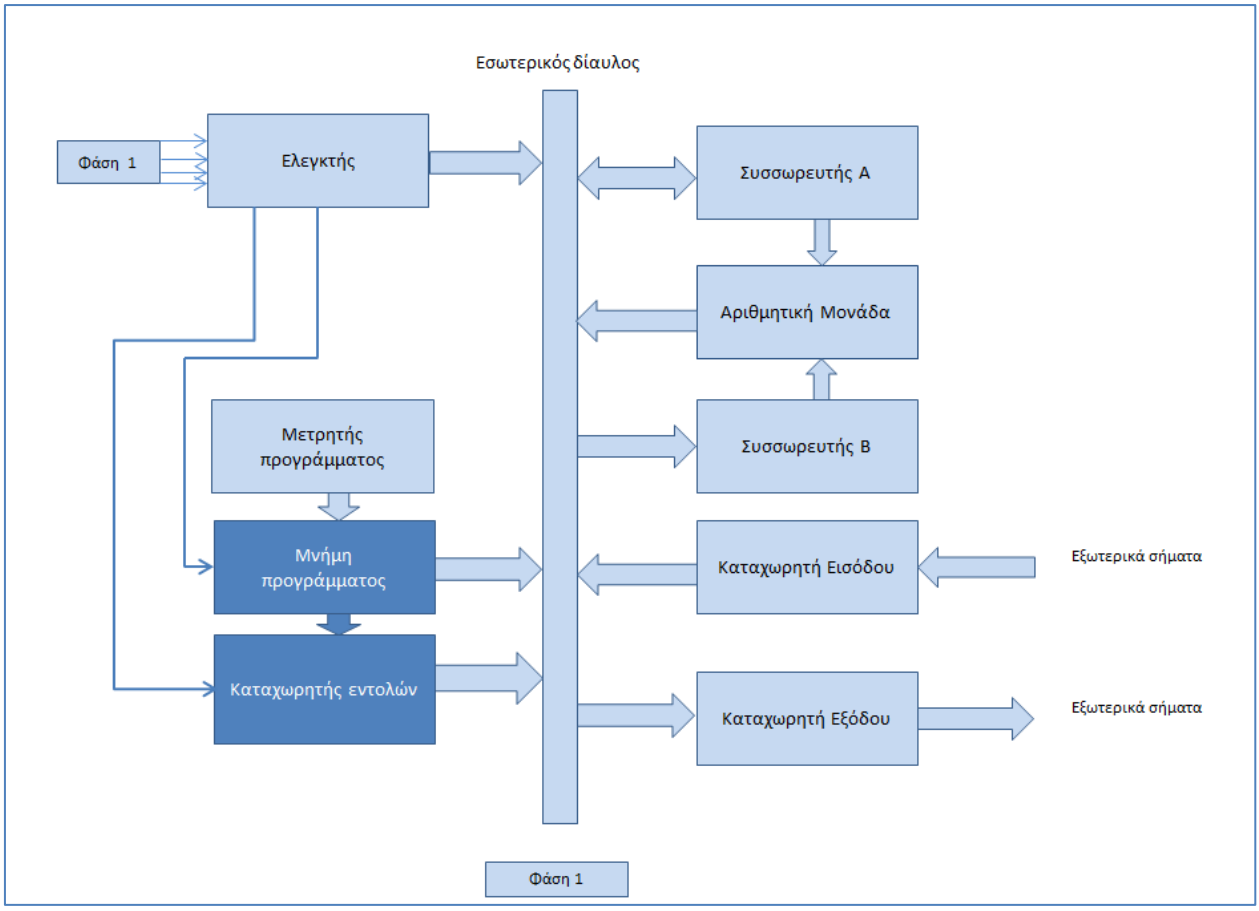
Εικόνα 1.5 : Η εκτέλεση μιας εντολής βασίζεται σε έξι χρονικές φάσεις που περιλαμβάνουν 6 μικροεντολές

Κατά την εκτέλεση μιας εντολής, διακρίνουμε τέσσερις φάσεις, που περιγράφονται ως εξής. Οι δύο πρώτες φάσεις ονομάζονται ακολουθία ανάκτησης. Οι αντίστοιχες μικροεπεξεργασίες είναι ανεξάρτητες από τις οδηγίες του χρήστη.

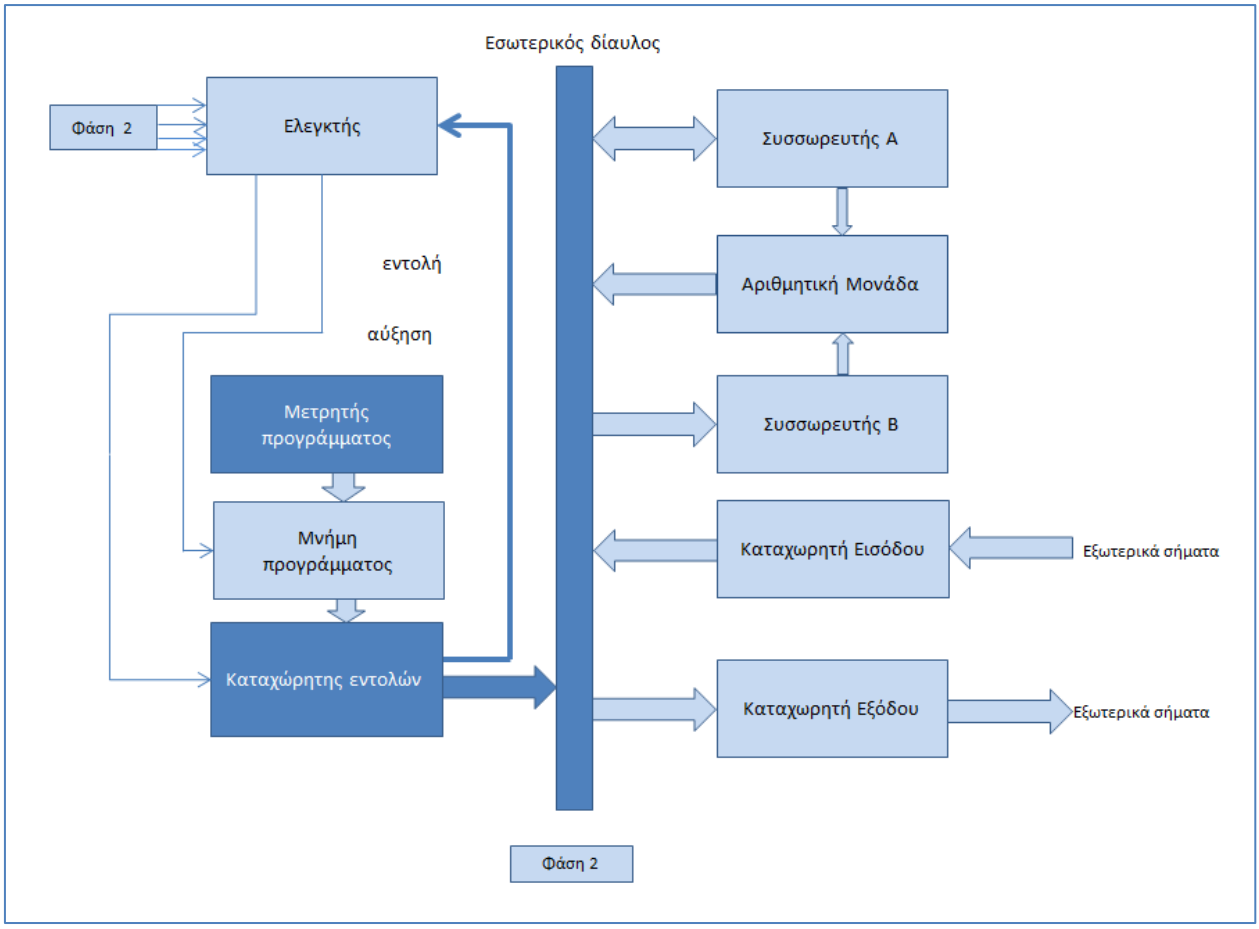
Πίνακας 4

Φάση	**Όνομα**	**Περιγραφή**
Φάση 1	Κατάσταση ανάγνωσης μνήμης	Τα περιεχόμενα της μνήμης μεταφέρονται από τον καταχωρητή εντολών.
Φάση 2	Αύξηση μετρητή προγράμματος	Η διεύθυνση του μετρητή προγράμματος αυξάνεται. Ο καταχωρητής παρέχει τον αποκωδικοποιητή της μικροεντολής μαζί με την την εντολή.
Φάση 3	$1^{η}$ φάση εκτέλεσης	Ανάλογα με την εντολή, ο μικροεπεξεργαστής εκτελεί την πρώτη φάση της εντολής.
Φάση 4	$2^{η}$φάση εκτέλεσης	Ο μικροεπεξεργαστής εκτελεί τη δεύτερη φάση της εντολής.

Όπως απεικονίζεται στον Πίνακα 4 , μία εντολή βασίζεται σε τέσσερις χρονικές στιγμές.



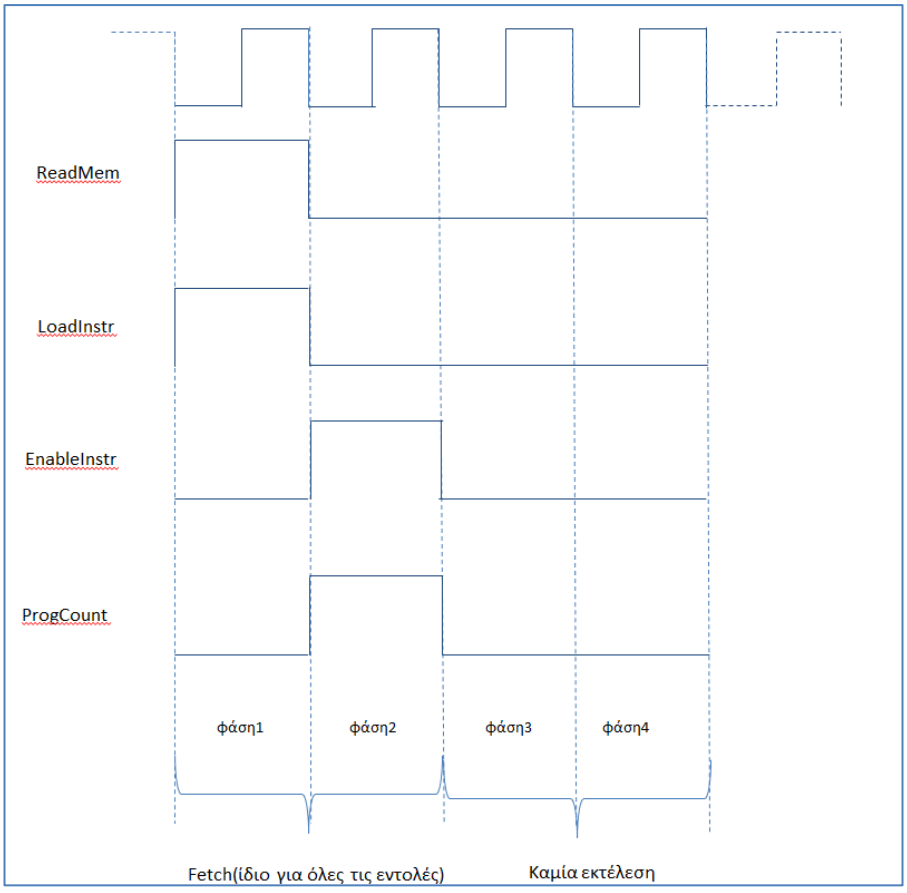
Εικόνα 1.6: Η εκτέλεση των δύο πρώτων μικροεντολών που αντιπροσωπεύουν την ακολουθία "fetch"



Εικόνα 1.7: Η εκτέλεση των δύο πρώτων μικροεντολών που αντιπροσωπεύουν την ακολουθία "fetch"

1.5.1 No Operation (NOP=0000)

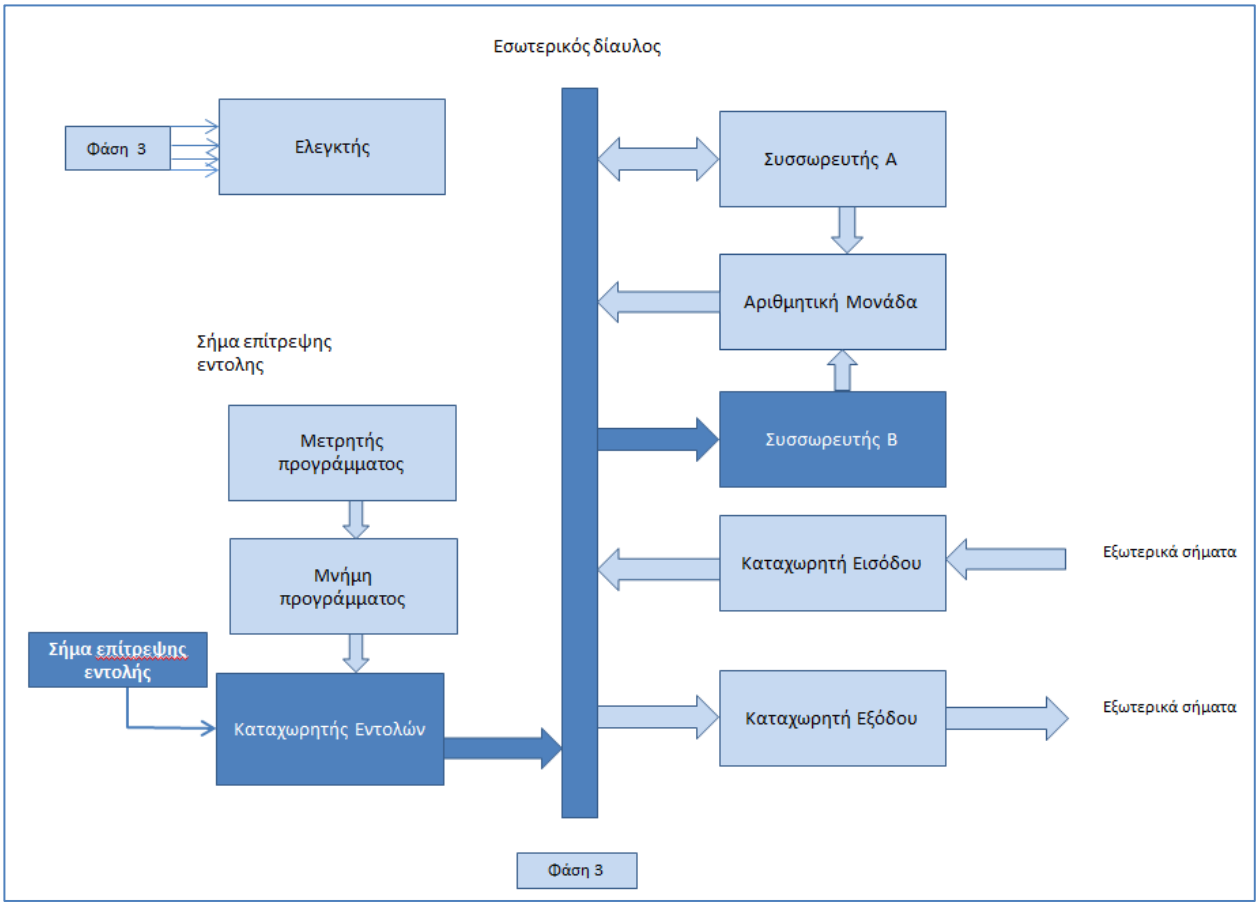
Η ακολουθία Fetch αντιστοιχεί στην πρόσβαση μνήμης (RedMem = 1), τη φόρτωση της αντίστοιχης εντολής (LoadInstr = 1) κατά τη φάση 1 (Εικόνα 1.8). Κατά τη διάρκεια της φάσης 2, η αποθηκευμένη εντολή αποστέλλεται στον ελεγκτή μικροεπεξεργασίας, ενώ ο μετρητής αυξάνεται. Καθώς η εντολή «No Operation» δεν επηρεάζει κανένα εσωτερικό καταχωρητή , οι φάσεις εκτέλεσης (φάση 3 και φάση 4) δεν αντιστοιχούν σε καμία συγκεκριμένη δραστηριότητα

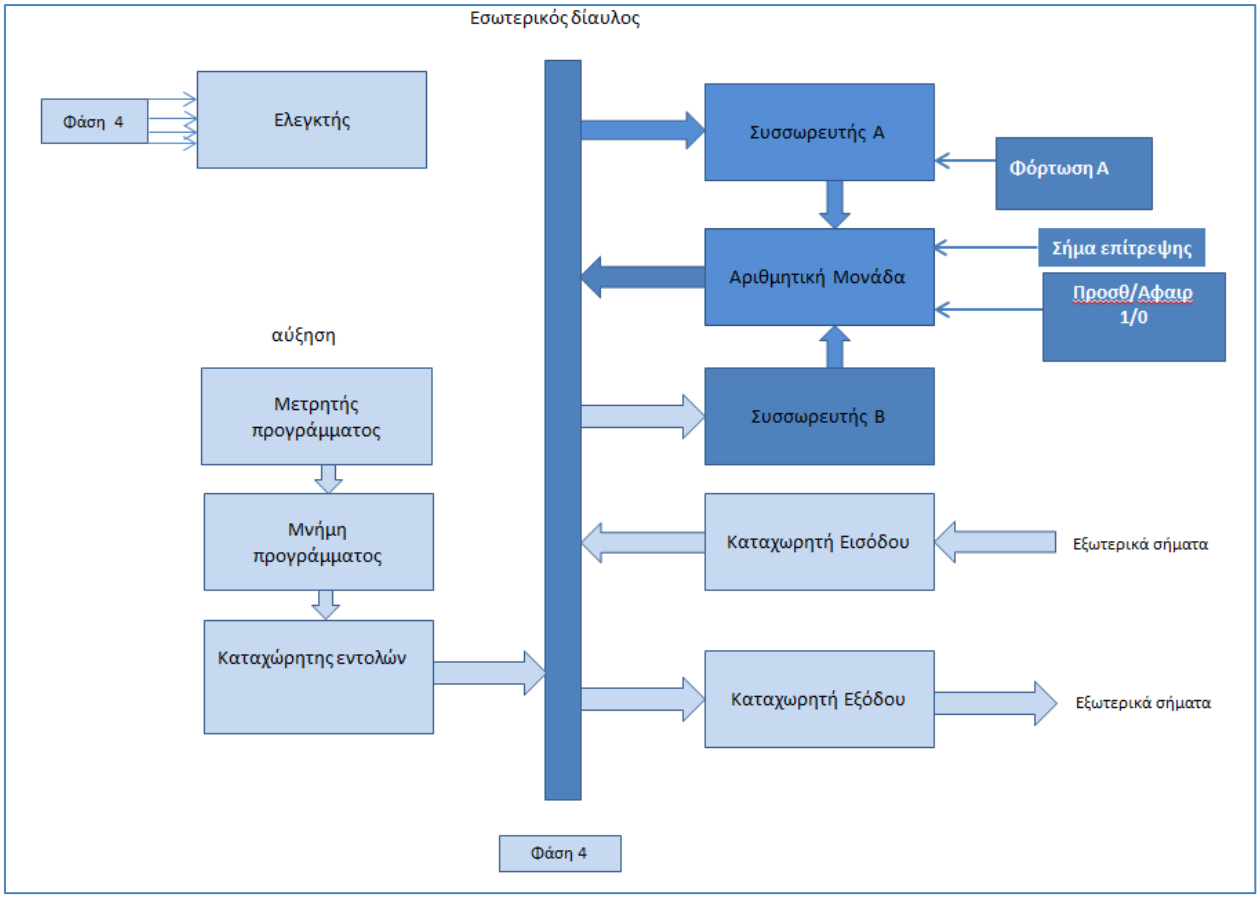


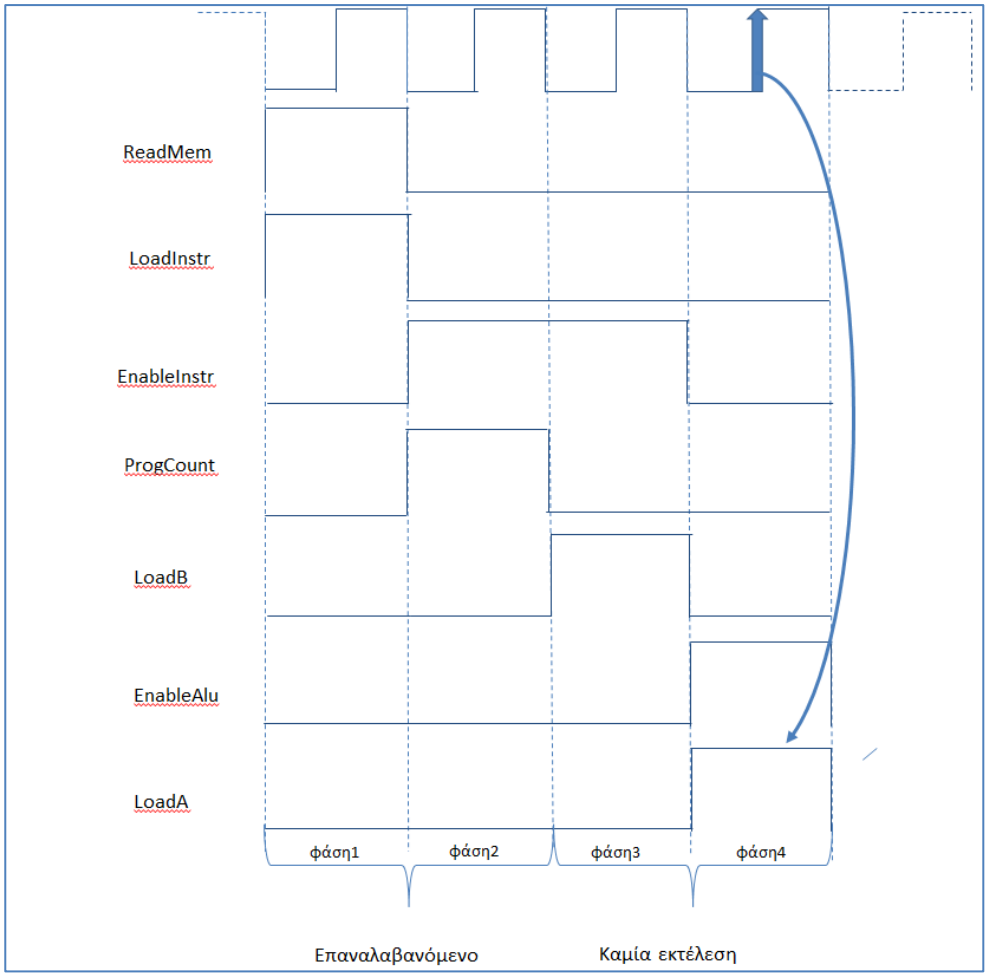
Εικόνα 1.8: Επαναλαμβανόμενη και εκτελέσιμη ακολουθία NOT

1.5.1 Addition (ADD=0001)

Η πρόσθεση πραγματοποιείται μεταξύ του συσσωρευτή A και των δεδομένων των 4 δυαδικών ψηφίων που δίδονται ως παράμετρος της εντολής ADD. Συνεπώς, η πρόσθεση εκτελείται αποθηκεύοντας τα δεδομένα στον συσσωρευτή B και στη συνέχεια ζητώντας από την αριθμητική μονάδα να παράγει την προσθήκη μεταξύ του συσσωρευτή A και του συσσωρευτή B (Φάση 4) και τελικά μεταφέροντας το αποτέλεσμα πίσω στον συσσωρευτή A στην άκρη άνοδο το ρολόι κατά τη φάση 4, όπως απεικονίζεται στην εικόνα 1.9.


Εσωτερικός δίαυλος
Φάση 3
Ελεγκτής
Συσσωρευτής Α
Αριθμητική Μονάδα
Σήμα επίτρεψης εντολης
Μετρητής προγράμματος
Συσσωρευτής Β
Μνήμη προγράμματος
Καταχωρητή Εισόδου
Εξωτερικά σήματα
Σήμα επίτρεψης εντολής
Καταχωρητής Εντολών
Καταχωρητή Εξόδου
Εξωτερικά σήματα
Φάση 3


Εσωτερικός δίαυλος
Φάση 4
Ελεγκτής
Συσσωρευτής Α
Φόρτωση Α
Αριθμητική Μονάδα
Σήμα επίτρεψης
Προσθ/Αφαιρ
1/0
αύξηση
Μετρητής προγράμματος
Συσσωρευτής Β
Μνήμη προγράμματος
Καταχωρητή Εισόδου
Εξωτερικά σήματα
Καταχώρητης εντολών
Καταχωρητή Εξόδου
Εξωτερικά σήματα
Φάση 4



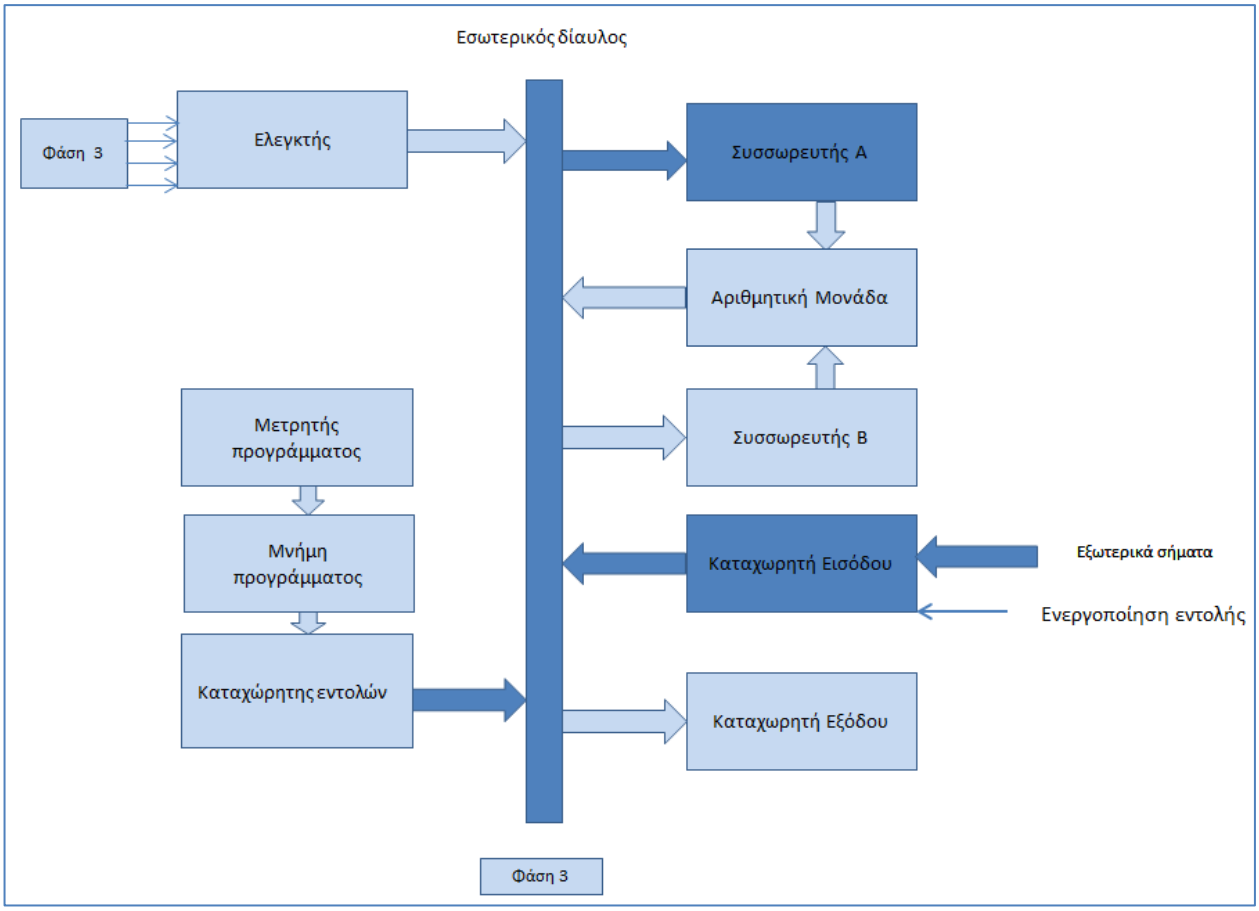
Εικόνα 1.9 : Εκτέλεση των μικροεντολών που αντιστοιχούν στην εντολή ADD

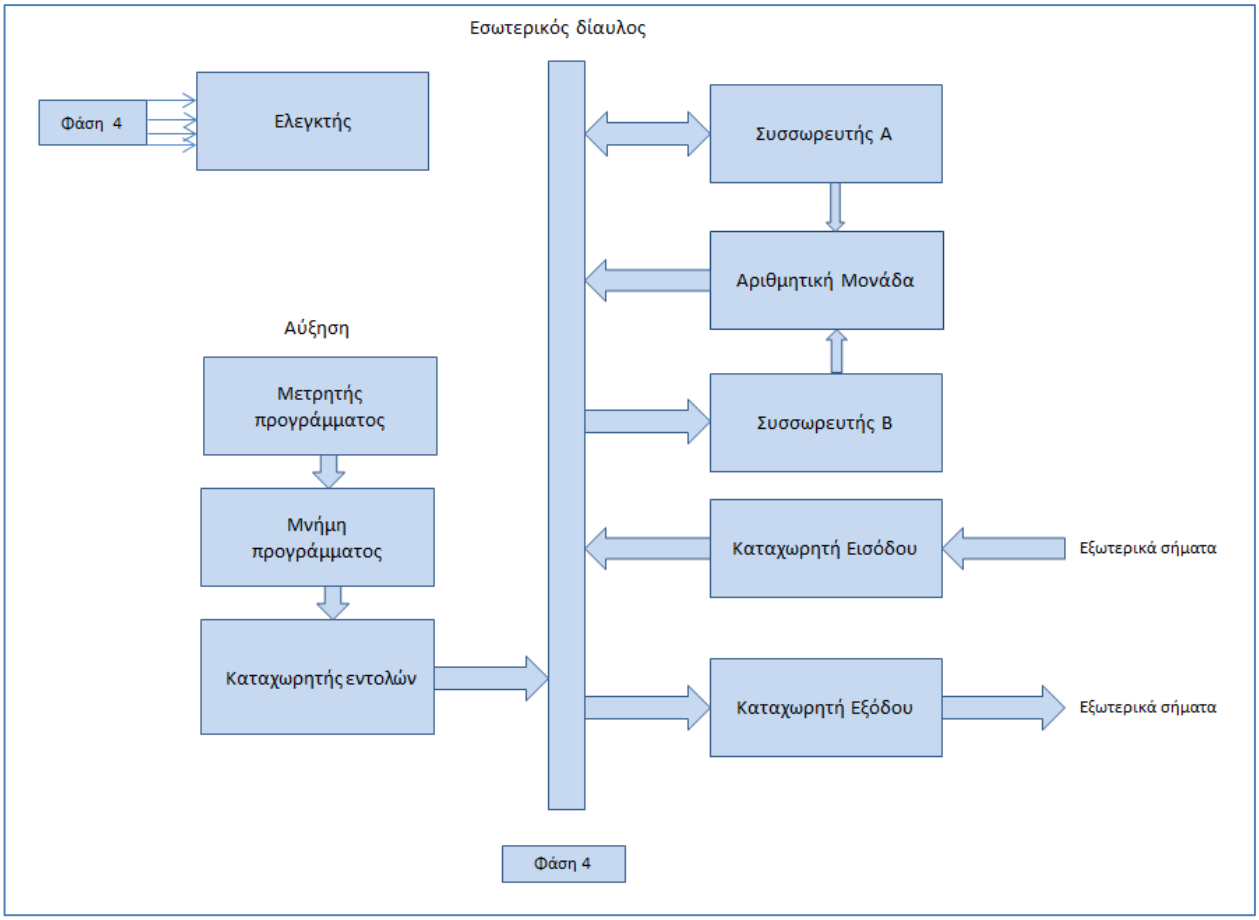
1.5.2 Subtraction (SUB=0010)

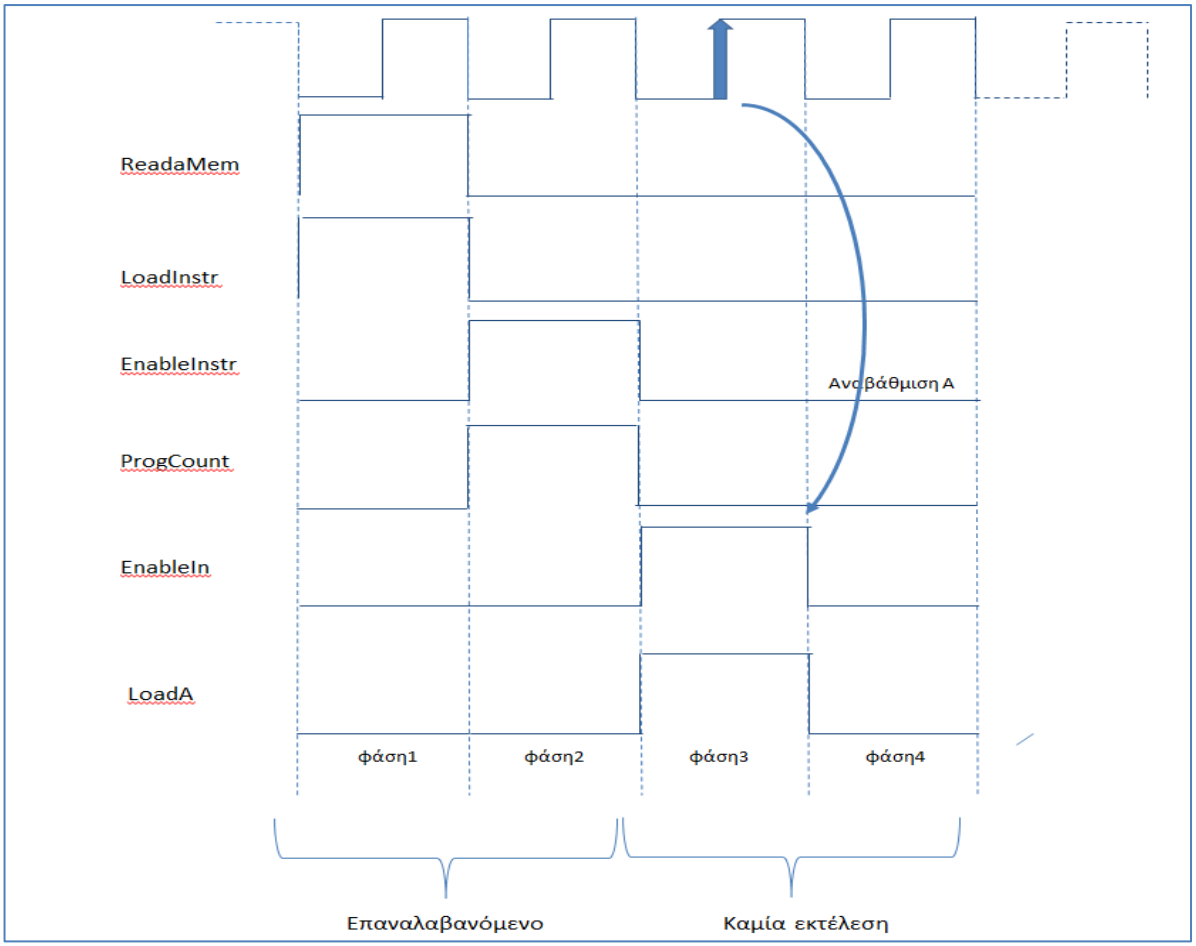
Η φάση εκτέλεσης της εντολής αφαίρεσης είναι ίδια με εκείνη της εντολής πρόσθεσης. Η μόνη διαφορά είναι ότι το "AddSub" έχει οριστεί σε 0, το οποίο σημαίνει "αφαίρεση".

1.5.3 Get Input (In=0011)

Η πύλη εισόδου μεταφέρεται στον συσσωρευτή A κατά τη φάση 3 (εικόνα 1.10).Δεν υπάρχει τίποτα άλλο να πραγματοποιηθεί στιη φάση 4 ,όπου όλοι οι κατανεμητές παραμένουν ανενεργοί.


Εσωτερικός δίαυλος
Φάση 3
Ελεγκτής
Συσσωρευτής Α
Αριθμητική Μονάδα
Μετρητής προγράμματος
Συσσωρευτής Β
Μνήμη προγράμματος
Καταχωρητή Εισόδου
Εξωτερικά σήματα
Ενεργοποίηση εντολής
Καταχώρητης εντολών
Καταχωρητή Εξόδου
Φάση 3

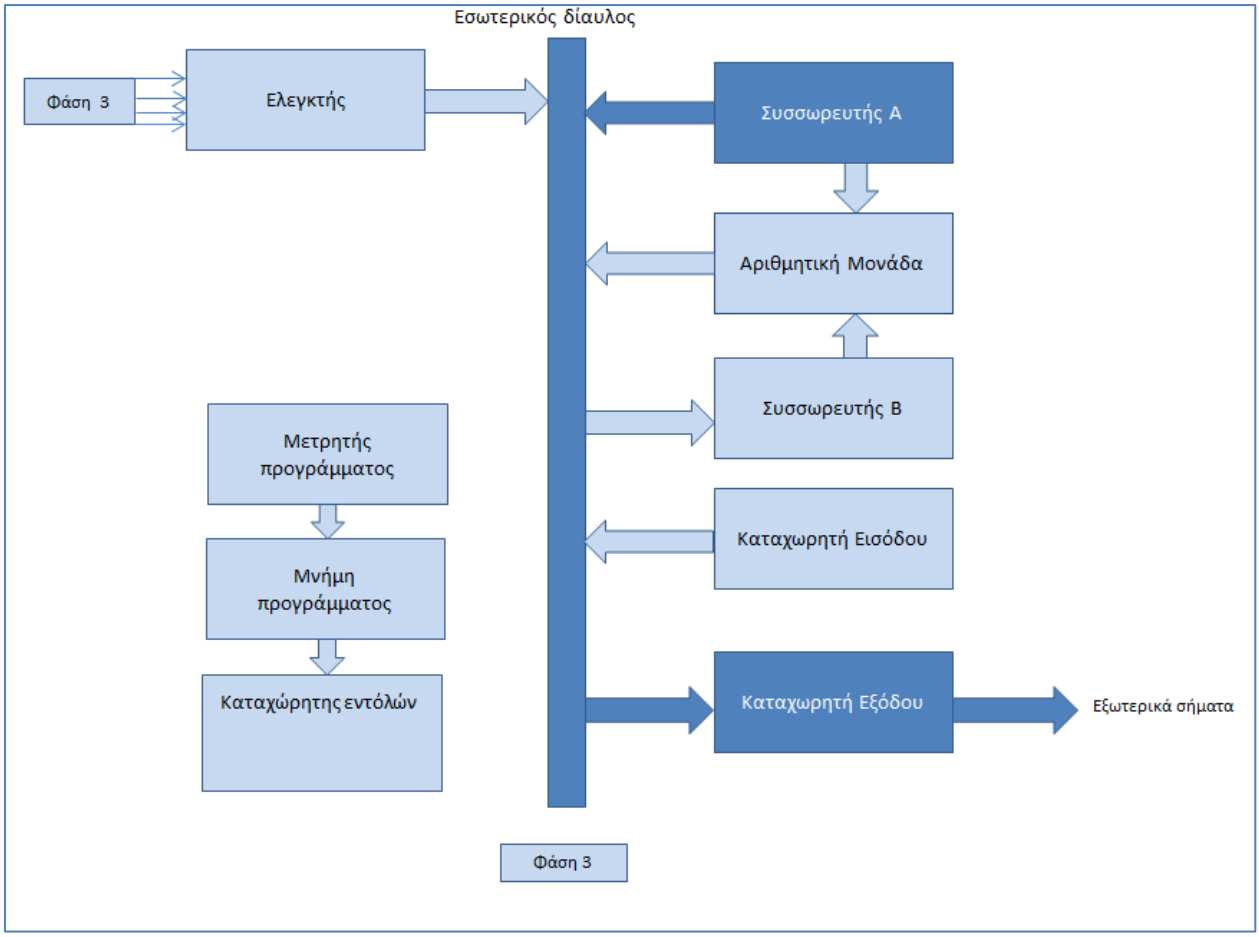

Εσωτερικός δίαυλος
Φάση 4
Ελεγκτής
Συσσωρευτής Α
Αριθμητική Μονάδα
Αύξηση
Μετρητής προγράμματος
Συσσωρευτής Β
Μνήμη προγράμματος
Καταχωρητή Εισόδου
Εξωτερικά σήματα
Καταχωρητής εντολών
Καταχωρητή Εξόδου
Εξωτερικά σήματα
Φάση 4



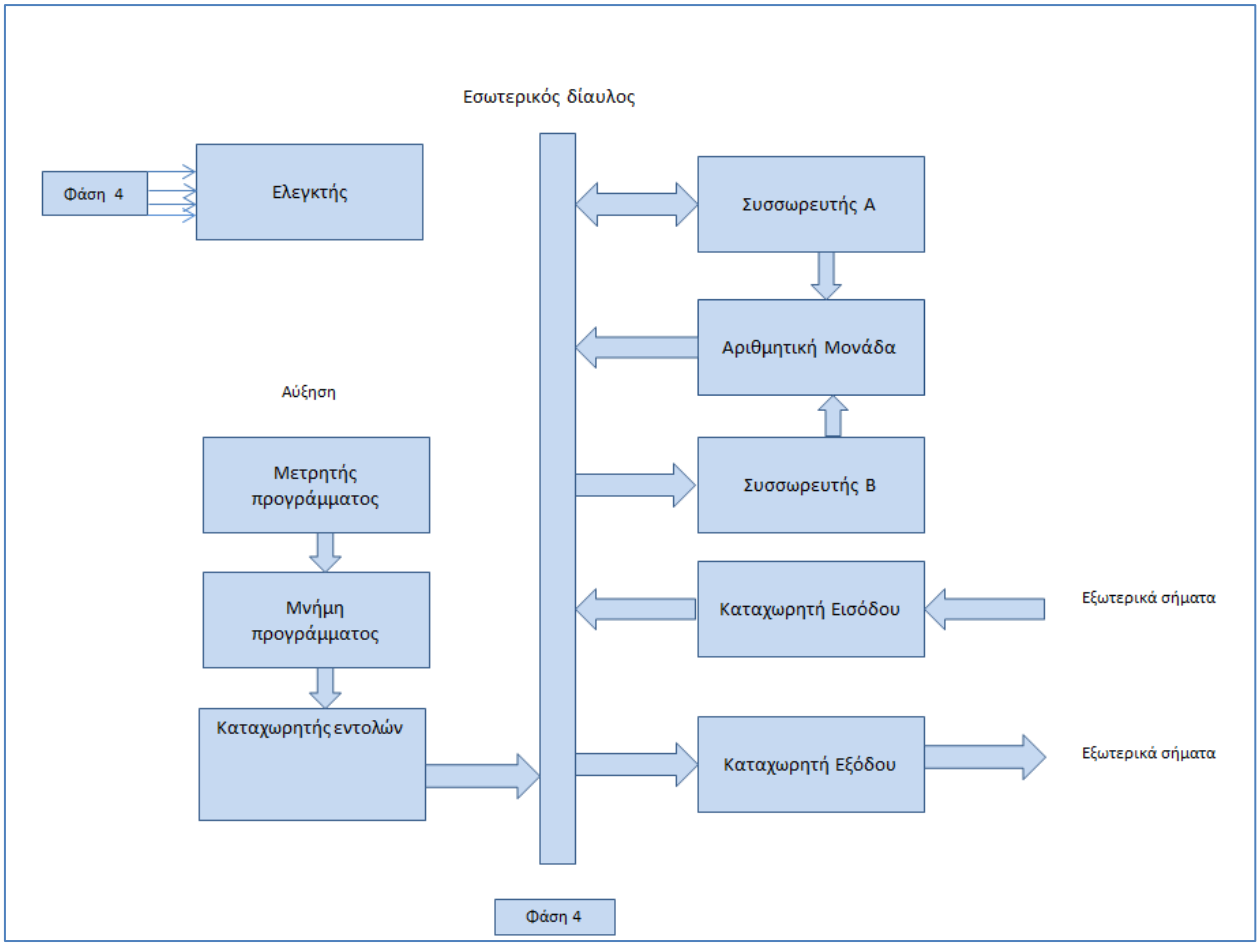
Εικόνα 1.10: Εκτέλεση των μικροεντολών που αντιστοιχούν στην εντολή IN

1.5.4 Give Output (OUT=0100)

Το περιεχόμενο που συσσωρευτή Α μεταφέρεται στη πύλη εξόδου μέσω του εσωτερικού διαύλου κατα τη φάση 3.Η πύλη εξόδου απομνημονεύει την τιμή του συσσωρευτή και την διατηρεί διαρκώς διαθέσιμη λόγω των τεσσάρων καταχωρητών.Ο επεξεργαστής είναι ανενεργός κατά τη διάρκεια της φάσης 4.



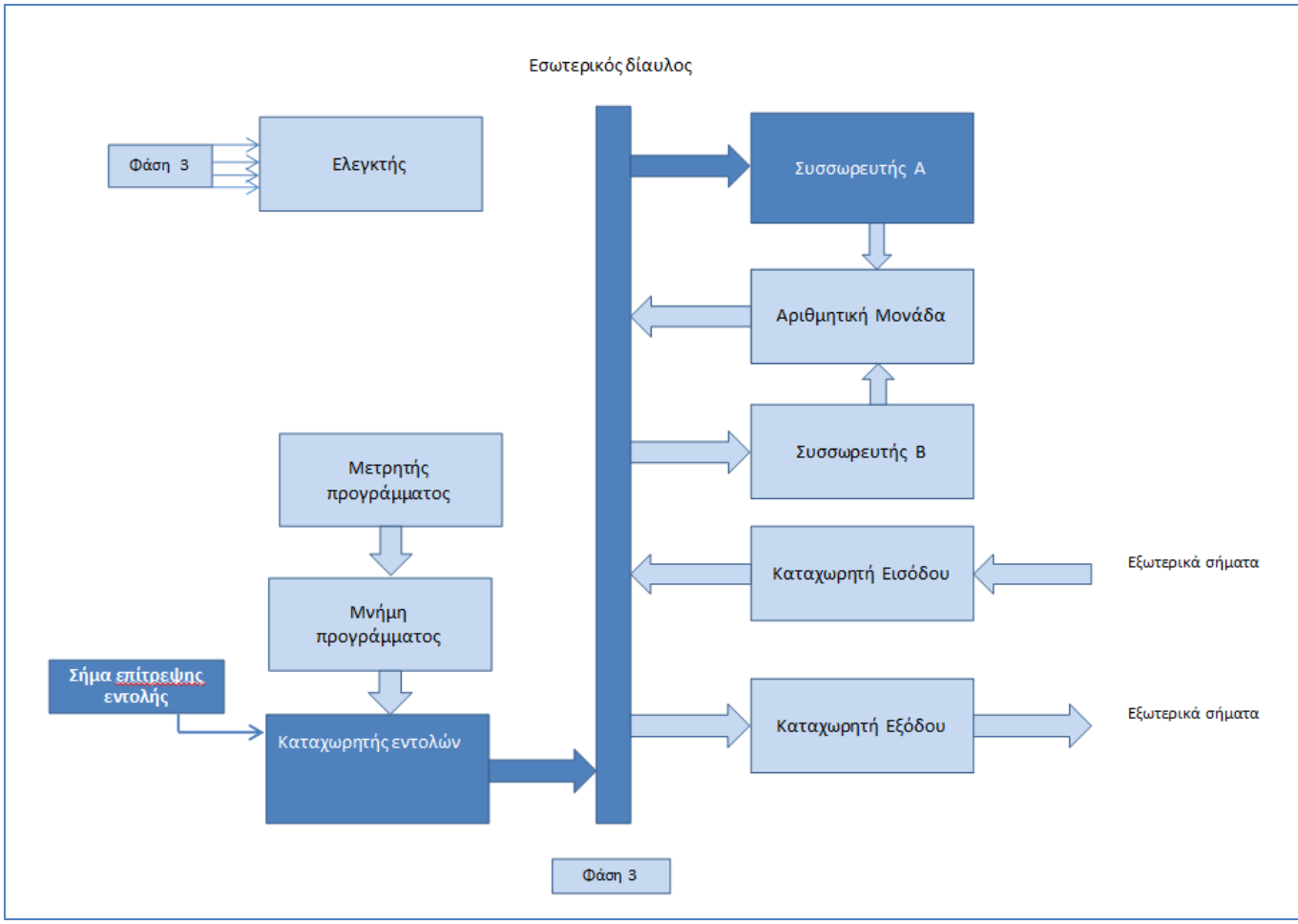
Εικόνα 1.11: Η εκτέλεση των μικροεργαλείων που αντιστοιχούν στην εντολή OUT

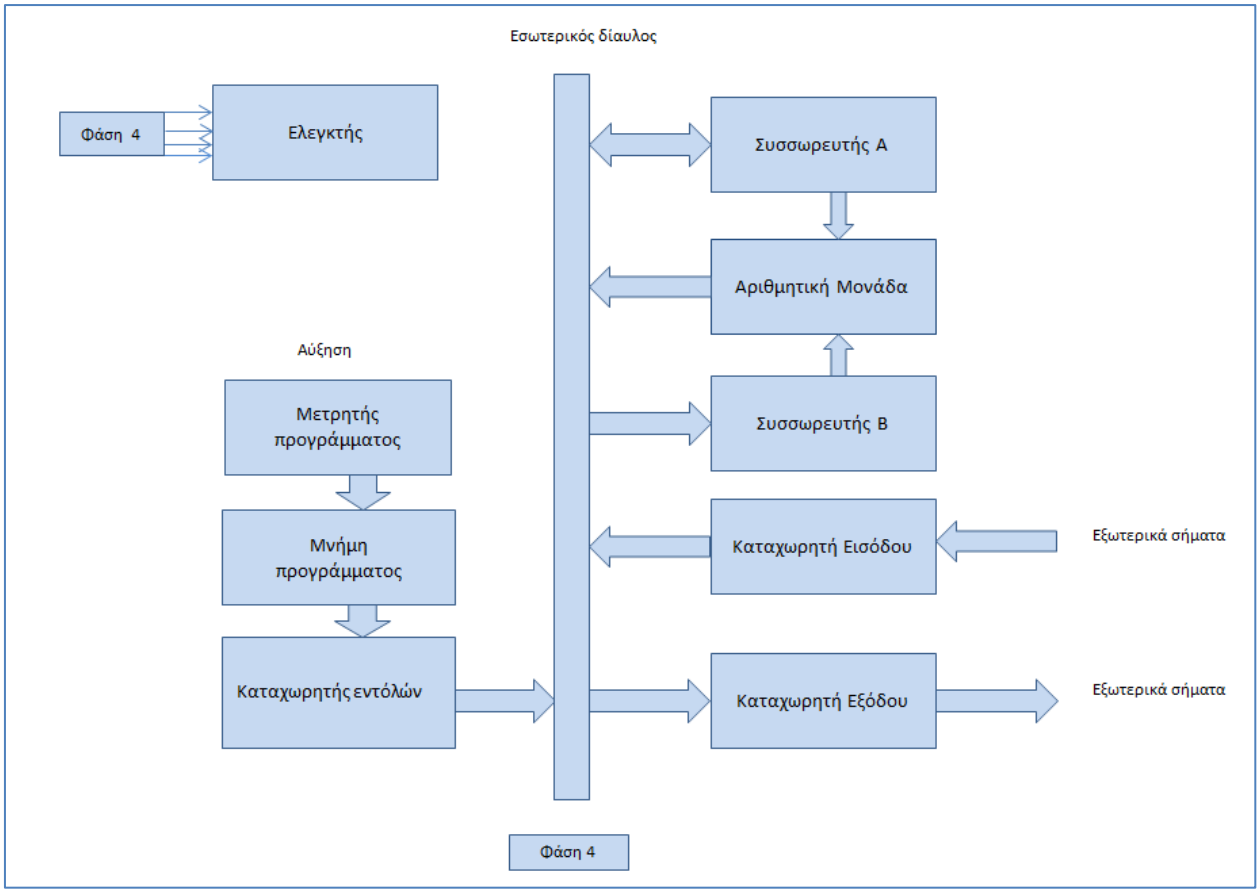


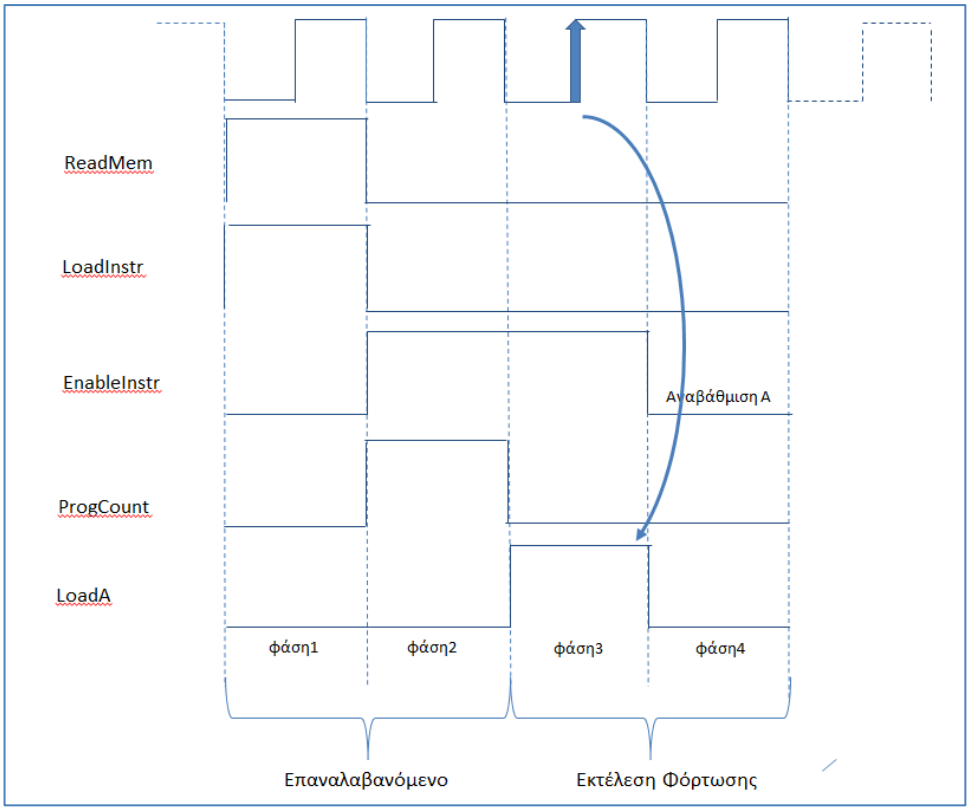
Εικόνα 1.12: Η εκτέλεση των μικροεργαλείων που αντιστοιχούν στην εντολή OUT

1.5.5 Load Instruction (LDA=0101)

Η εντολή φορτίου μεταφέρει τα δεδομένα τεσσάρων δυαδικών ψηφίων που δίδονται ως παράμετρος της εντολής LDA στον συσσωρευτή Α.Για παράδειγμα , η οδηγία LDA 9 μεταφέρει τη τιμή 9 στον συσσωρευτή Α.Στο σχήμα 7,τα σημαντικά κομμάτια του καταχωρητή εντολών τοποθετούνται στον εσωτερικό δίαυλο και στη συνέχεια μεταφέρονται στον συσσωρευτή Α.Ως αποτελέσμα, η αναβαθμισμένη τιμή του Α να είναι 1001.Στη φάση 4 δεν υπάρχει καμία δραστηριότητα.


Εσωτερικός δίαυλος
Φάση 3
Ελεγκτής
Συσσωρευτής Α
Αριθμητική Μονάδα
Συσσωρευτής Β
Μετρητής προγράμματος
Μνήμη προγράμματος
Καταχωρητή Εισόδου
Εξωτερικά σήματα
Σήμα επίτρεψης εντολής
Καταχωρητής εντολών
Καταχωρητή Εξόδου
Εξωτερικά σήματα
Φάση 3


Εσωτερικός δίαυλος
Φάση 4
Ελεγκτής
Συσσωρευτής Α
Αριθμητική Μονάδα
Αύξηση
Μετρητής προγράμματος
Συσσωρευτής Β
Μνήμη προγράμματος
Καταχωρητή Εισόδου
Εξωτερικά σήματα
Καταχωρητής εντολών
Καταχωρητή Εξόδου
Εξωτερικά σήματα
Φάση 4

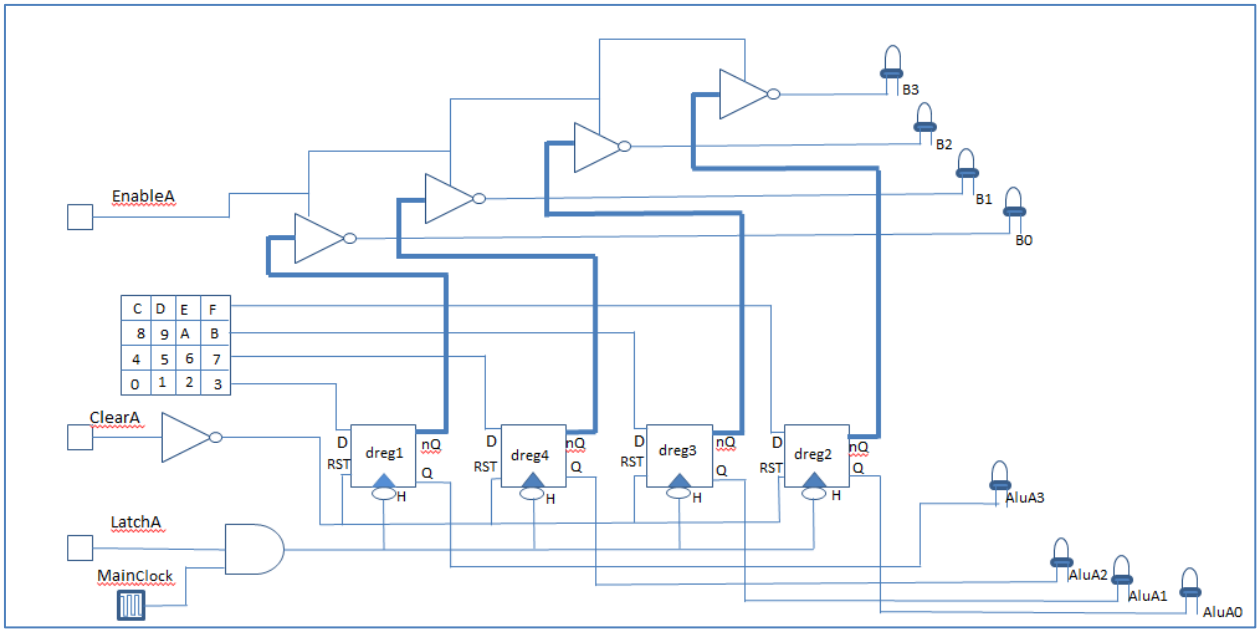


Εικόνα 1.13: Η μικροεπεξεργασία κατά τη φάση 3 εκτελεί τη λειτουργία φορτίου. Κατά τη διάρκεια της φάσης 4, ο επεξεργαστής είναι ανενεργός.

ΚΕΦΑΛΑΙΟ-2 ΤΜΗΜΑΤΑ ΤΟΥ ΕΠΕΞΕΡΓΑΣΤΗ

2.1 Ο συσσωρευτής A

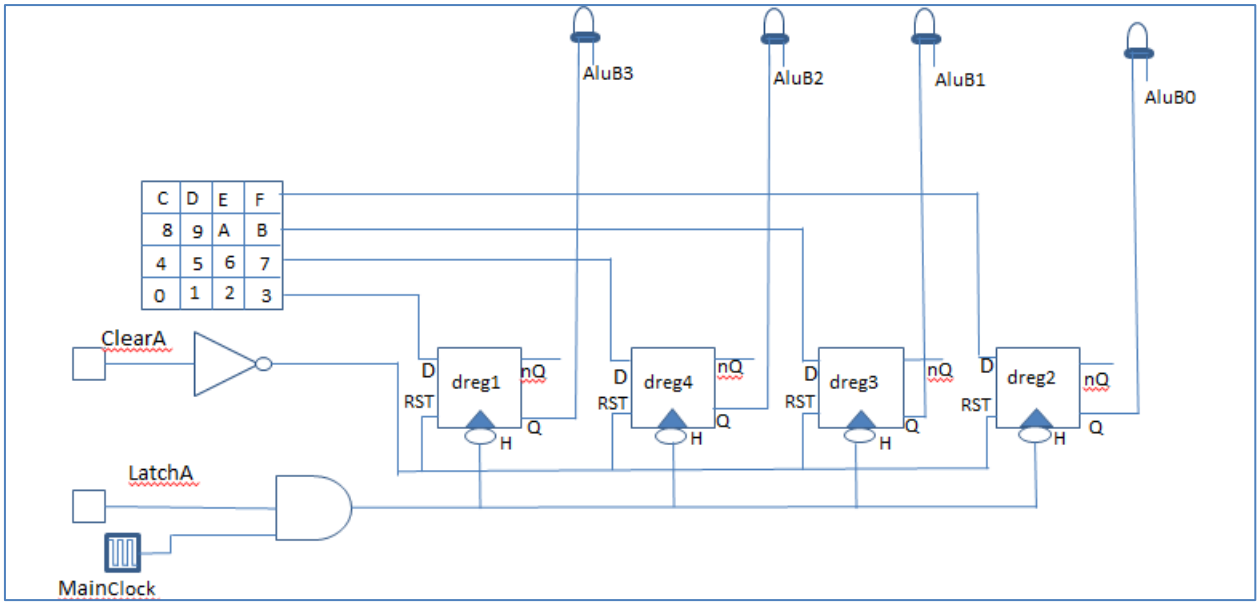
Ο συσσωρευτής βασίζεται σε τέσσερις καταχωρητές D. Η έξοδος του καταχωρητή είναι μόνιμα διαθέσιμη μέσω του AluA0..AluA3 για τις λειτουργίες ΠΡΟΣΘΕΣΗΣ και ΑΦΑΙΡΕΣΗΣ. Τα περιεχόμενα του A μπορούν να μεταφερθούν στον εσωτερικό δίαυλο όταν επιβεβαιωθεί το "EnableA". Χρησιμοποιούμε τρεις μετατροπείς κατάστασης για να δημιουργήσουμε πρόσβαση στον εσωτερικό δίαυλο. Το σήμα "latchA" επιτρέπει τη μεταφορά δεδομένων εισόδου (εδώ πληκτρολογίου) στους καταχωρητές D στην άκρη πτώσης του κύριου ρολογιού.



Εικόνα 2.1: Δομή του συσσωρευτή A που δείχνει τις συνδέσεις του με τον εσωτερικό δίαυλο και την αριθμητική μονάδα

2.2 Ο συσσωρευτής *B*

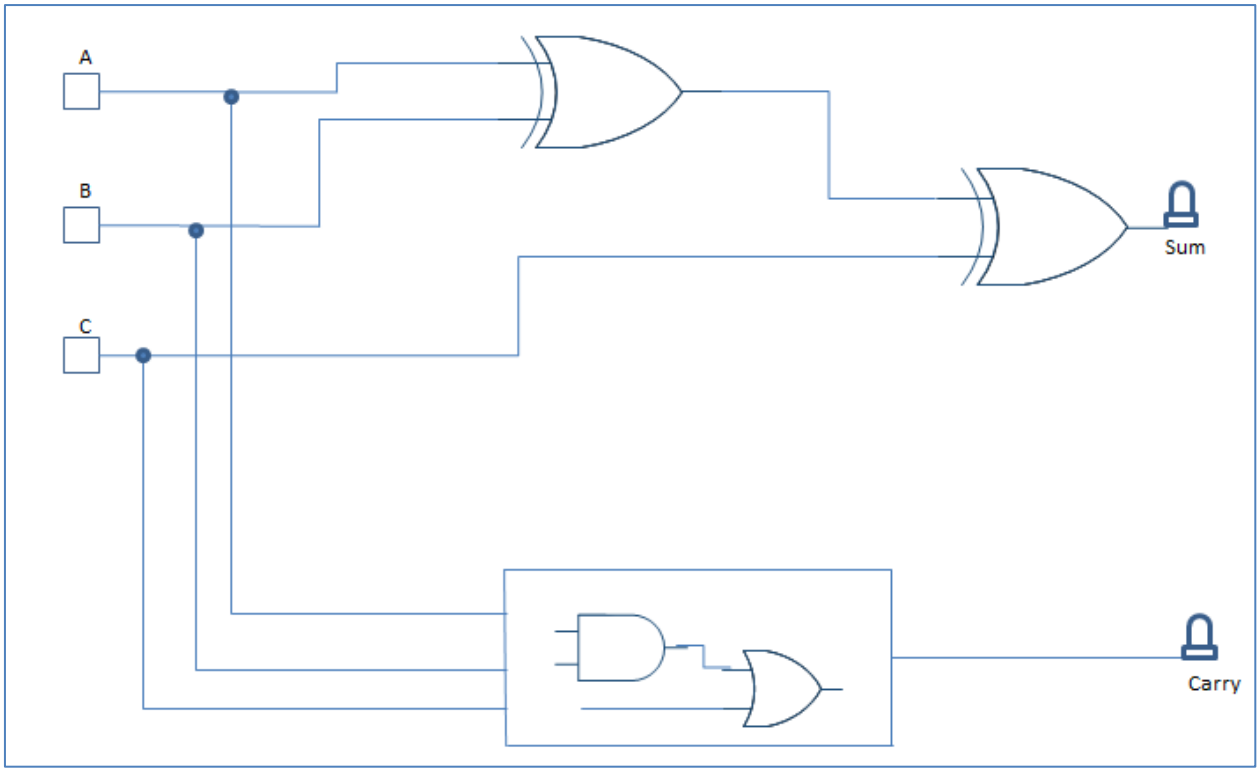
Ο συσσωρευτής B βασίζεται επίσης σε τέσσερις καταχωρητές D ευαίσθητους στην άκρη. Η έξοδος του καταχωρητή είναι μόνιμα διαθέσιμη μέσω του AluB0..AluB3 για τις λειτουργίες ADD και SUB. Το σήμα "latchB" επέτρεψε τη μεταφορά δεδομένων εισόδου (εδώ πληκτρολογίου) στους καταχωρητές D στην άκρη πτώσης του κύριου ρολογιού.



Εικόνα 2.2: Δομή του συσσωρευτή B που δείχνει τις συνδέσεις του με την αριθμητική μονάδα

2.3 Αθροιστής - Αφαιρέτης

Η πρόσθεση βασίζεται σε υποκυκλώματα πλήρους αθροιστή. Η εσωτερική δομή του πλήρους αθροιστή είναι ένα σύνολο θυρών XOR για το σήμα "SUM" και μια σύνθετη πύλη για το σήμα "Carry", όπως φαίνεται στο σχήμα 2.3.



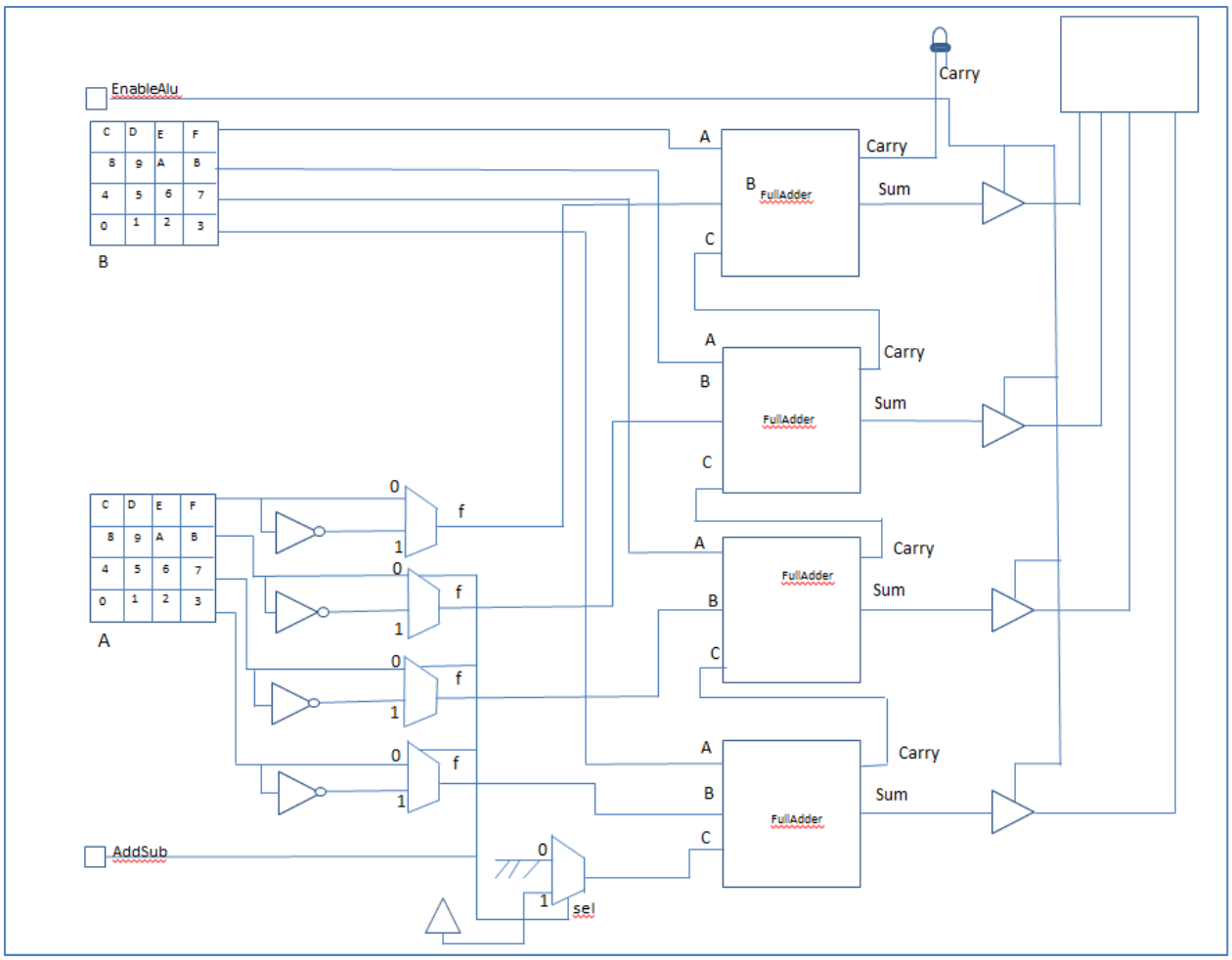
Εικόνα 2.3: Η εσωτερική δομή του Πλήρη αθροιστή

Η πρόσθεση δύο πληροφοριών τεσσάρων δυαδικών ψηφίων απαιτεί τέσσερις πλήρεις αθροιστές όπως απεικονίζεται στο σχήμα 13. Το σήμα μεταφοράς μεταδίδεται από το κάτω στάδιο στο ανώτερο στάδιο για να εκτελέσει την πλήρη λειτουργία της πρόσθεσης.

Για να αφαιρέσουμε δύο αριθμούς (B-A σε αυτή την περίπτωση) και να χρησιμοποιήσουμε ξανά τα ίδια κυκλώματα πλήρους αθροιστή, πρέπει να δημιουργήσουμε δύο συμπληρωματικά πράγματα:

- Ένα κύκλωμα που δημιουργεί το συμπλήρωμα 2 του A
- Ένα μικρό κύκλωμα που θέτει την αρχική μεταφορά στο 1.

Όταν το "Sel" ισούται με 0, η είσοδος i0 μεταφέρεται στην έξοδο, διαφορετικά, το i1 μεταφέρεται στην έξοδο. Συνεπώς, το "AddSub" = 0 αντιστοιχεί στη μεταφορά του A στην πρόσθεση, (Add operation), ενώ το "AddSub" = 1 αντιστοιχεί στη μεταφορά του ~ A στην αφαίρεση (λειτουργία Sub).



Εικόνα 2.4 : Δομή της αριθμητικής μονάδας που μπορεί να εκτελέσει τις λειτουργίες πρόσθεσης και αφαίρεσης

.

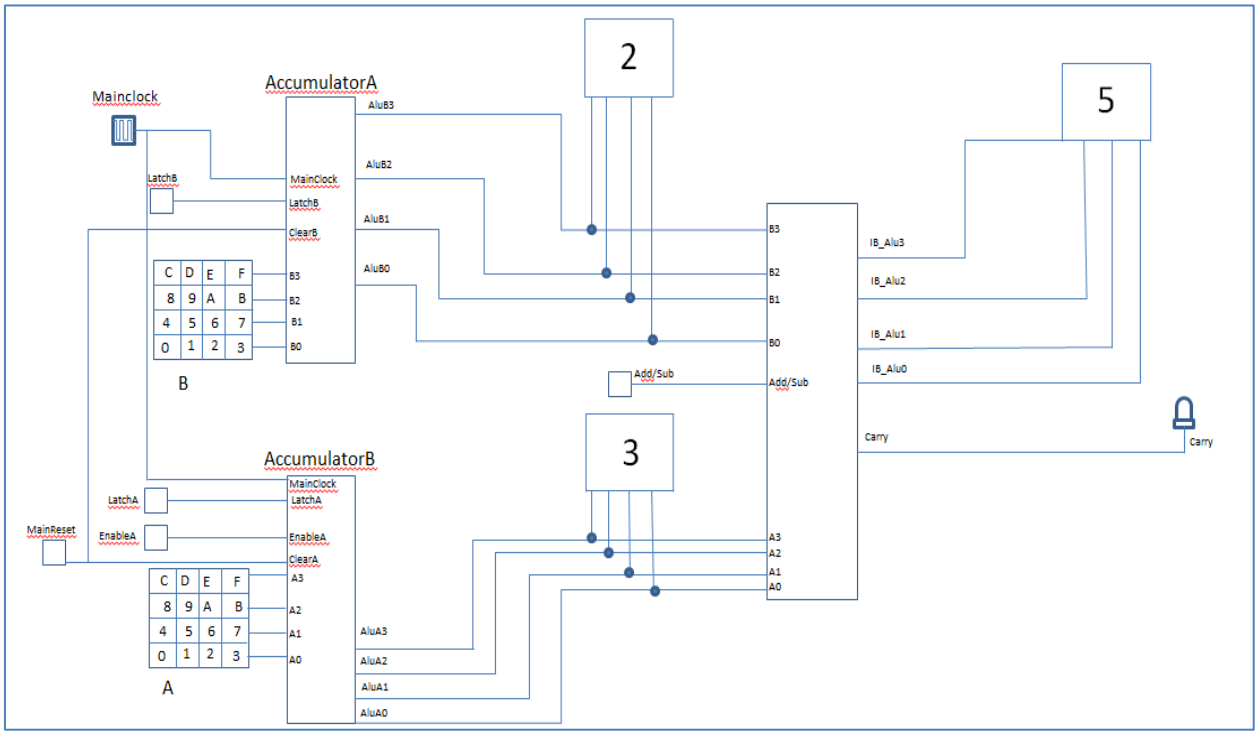
Σε αυτό το σημείο, είναι πολύ ενδιαφέρον η σύνδεση των συσσωρευτών με την αριθμιτική μονάδα για εκτελεστεί χειροκίνητα το αποτέλεσμα του μικροεπεξεργαστή.Το κύκλωμα από τον συσσωρευτή A, B και την αριθμητική μονάδα παρουσιάζεται στο σχήμα 14. Τα δύο πληκτρολόγια χρησιμεύουν ως είσοδοι A και B, οι οθόνες τοποθετούνται στους δίαυλους εξόδου και η σύνδεση αριθμητικής μονάδας στον εσωτερικό δίαυλο.

Η προσπάθεια να λειτουργήσει αυτό το απλό κύκλωμα είναι μια πολύ ενδιαφέρουσα εισαγωγή στη λειτουργία του μικροεπεξεργαστή. Παρακάτω είναι το σύνολο των ενεργειών που πρέπει να εκτελεστούν διαδοχικά για να προσθεθούν δύο αριθμοί:

• Απενεργοποιήται η κύρια επαναφορά. Αρχικά, ο ακροδέκτης επαναφοράς είναι ρυθμισμένος στο 0 (προεπιλεγμένη τιμή στην αρχή), που αντιστοιχεί σε μια ενεργή επαναφορά. Και οι δύο καταχωρητές A και B διαγράφονται (A = 0, B = 0). Τίποτα δεν μπορεί να λειτουργήσει μέχρι να ρυθμιστεί το πλήκτρο "~ MainReset" στο 1.

• Τοποθετείται η επιθυμητή τιμή στο A,με πληκτρολόγιση στο κάτω πληκτρολόγιο με όνομα "A", για παράδειγμα ‘3’. Έπειτα ενεργοποίηση "LatchA"και αναμένεται τουλάχιστον έναν πλήρη κύκλο του κύριου ρολογιού. Ο συσσωρευτής A αποθηκεύει 3 στην άκρη πτώσης του ρολογιού.

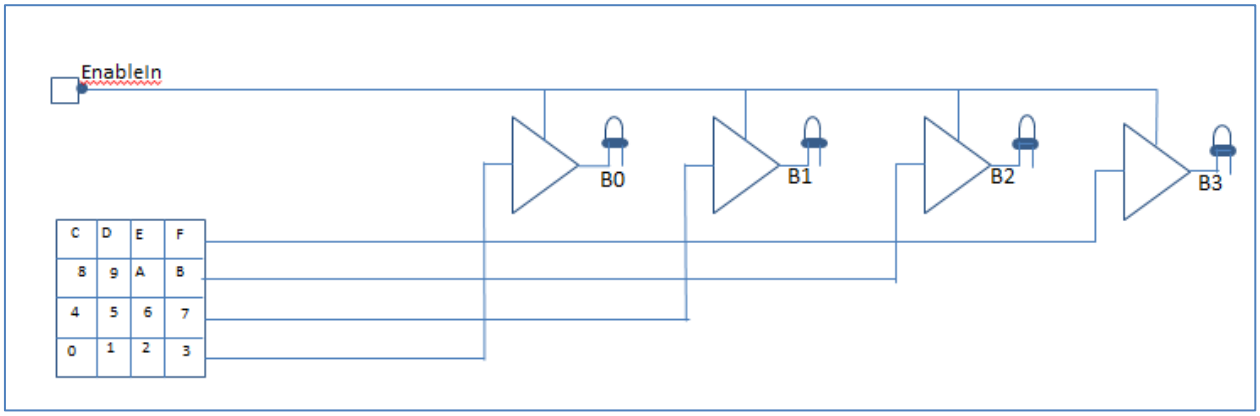
• Τοποθετείται η επιθυμητή τιμή στο B και στο πληκτρολόγιο "B" πληκτρολογείται για παράδειγμα ‘2’. Επαναλαμβάνεται πάλι η ενεργοποίηση του "LatchB"και αναμένεται τουλάχιστον έναν πλήρη κύκλο του κύριου ρολογιού. Ο συσσωρευτής B αποθηκεύει 2 στην άκρη πτώσης του ρολογιού. Η αριθμητική μονάδα υπολογίζει το άθροισμα A + B, καθώς το "AddSub" ορίζεται από προεπιλογή στο 0, το οποίο αντιστοιχεί στην εντολή ADD. Ωστόσο, το αποτέλεσμα δεν εμφανίζεται στο "EnableAlu" όταν είναι 0.
• Ρυθμίζεται το "EnableAlu" στο 1 για να εμφανιστεί το αποτέλεσμα "5.



Εικόνα 2.5: Η σύνδεση του συσσωρευτή A,B με την αριθμητική μονάδα για τη δοκιμή των εντολών πρόσθεσης/αφαίρεσης

2.4 Καταχωρητής για εισόδου

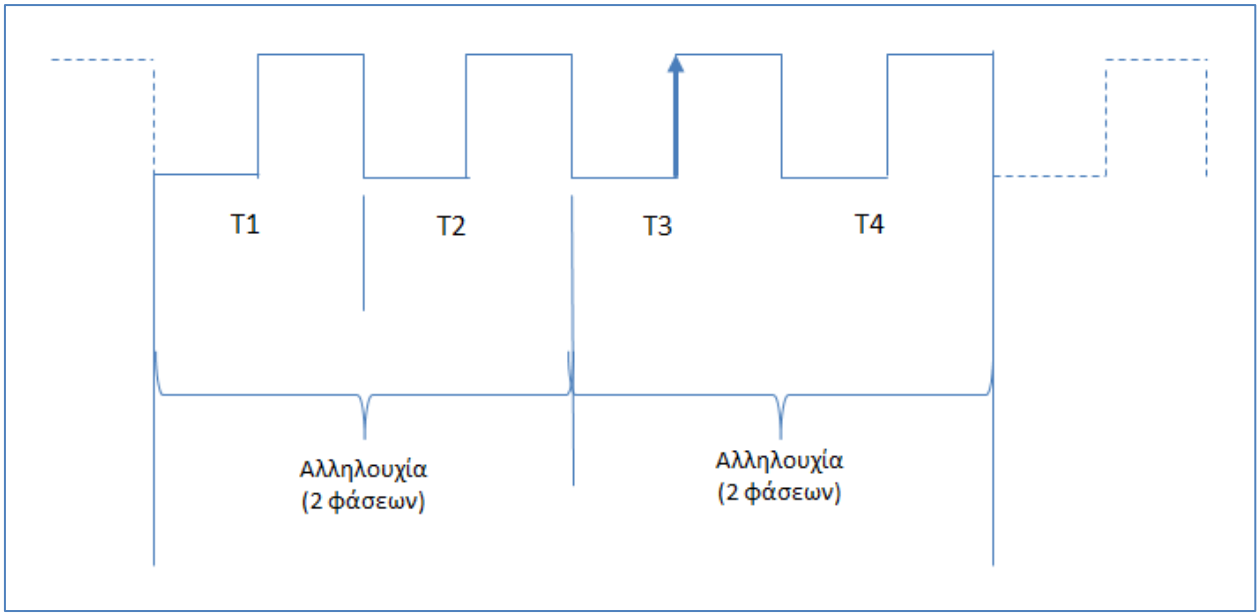
Ο καταχωρητής εισόδου είναι ένα απλό σύνολο ρυθμιστικών 3 καταστάσεων, όπως φαίνεται στο σχήμα 2.5. Δεν υπάρχει ανάγκη για καταχωρητές D καθώς η είσοδος θα μεταφερθεί απευθείας στον συσσωρευτή A



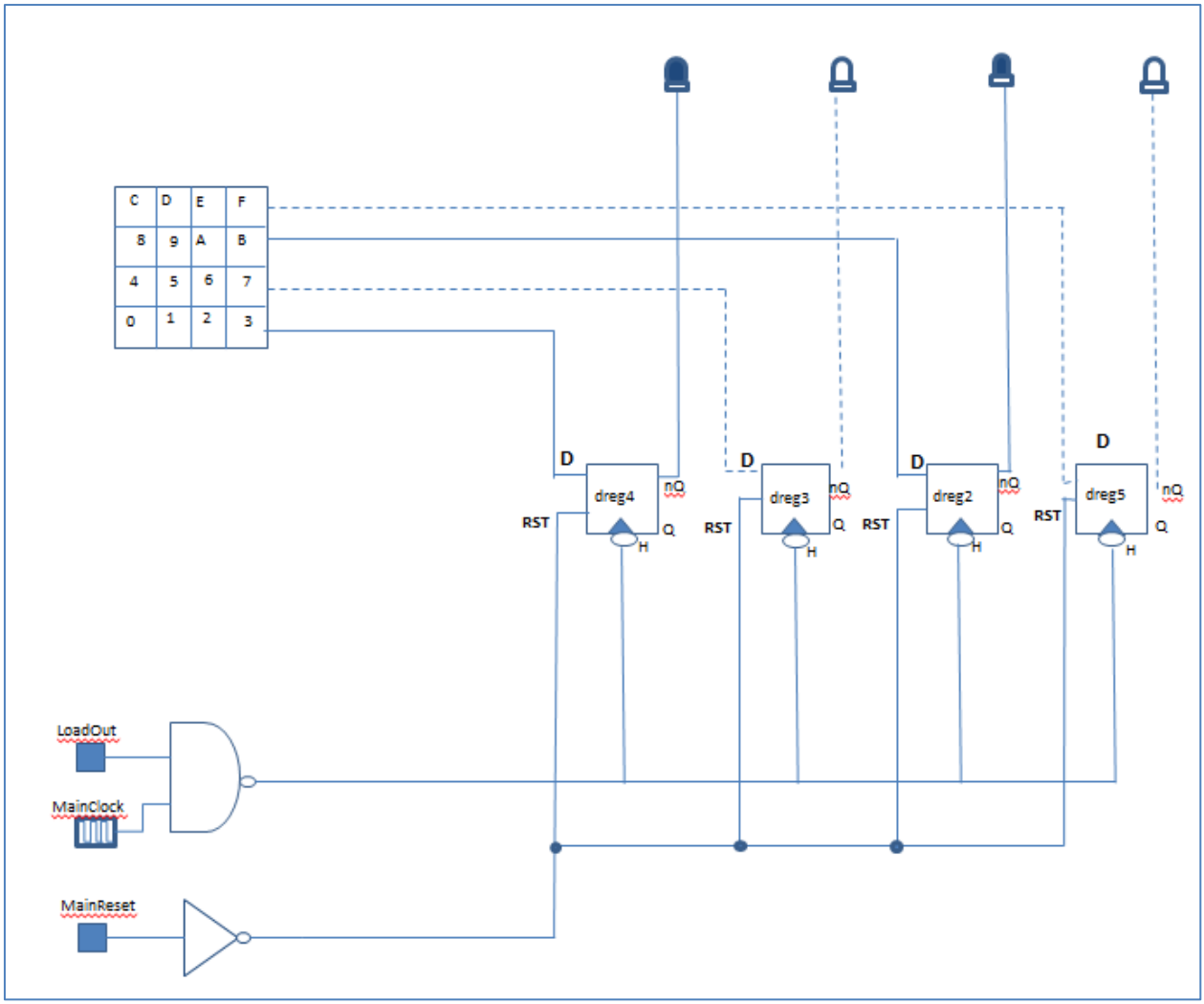
Εικόνα 2.6: Ο καταχωρητής εισόδου

2.5 Καταχωρητής για έξόδου

Ο καταχωρητής εξόδου βασίζεται σε καταχωρητές D όπως φαίνεται παρακάτω. Στη θετική άκρη του ρολογιού, τα δεδομένα αποθηκεύονται στους καταχωρητές. Είναι πολύ σημαντικό τα δεδομένα να αποθηκεύονται στη θετική άκρη του ρολογιού κατά τη διάρκεια της φάσης 3 και όχι στην αρνητική άκρη που θα προκαλούσαν συγκρούσεις συγχρονισμού. Επομένως, μια πύλη NAND χρησιμοποιείται για να κάνει το κύκλωμα ευαίσθητο στην άκρη ανύψωσης του κύριου ρολογιού, όπως φαίνεται στο σχήμα 16.



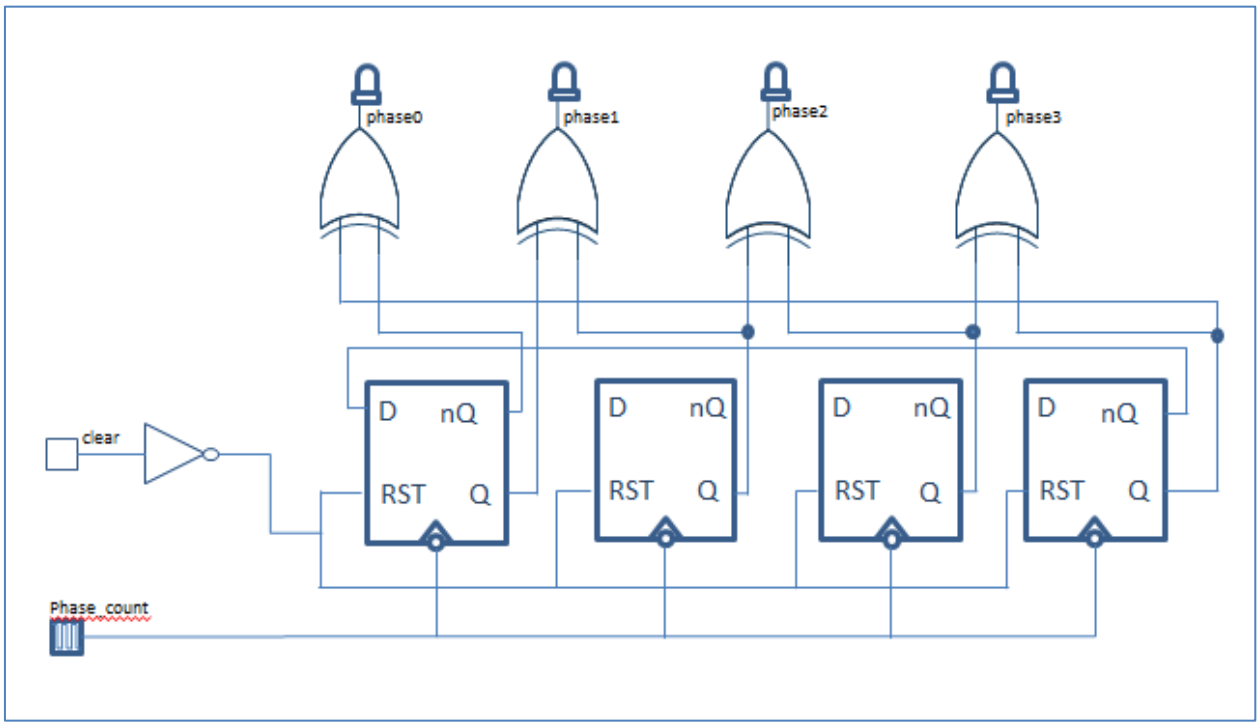
Εικόνα 2.7: Ο καταχωρητής εξόδου πρέπει να αποθηκεύει τα δεδομένα στην άνοδος της φάσης 3



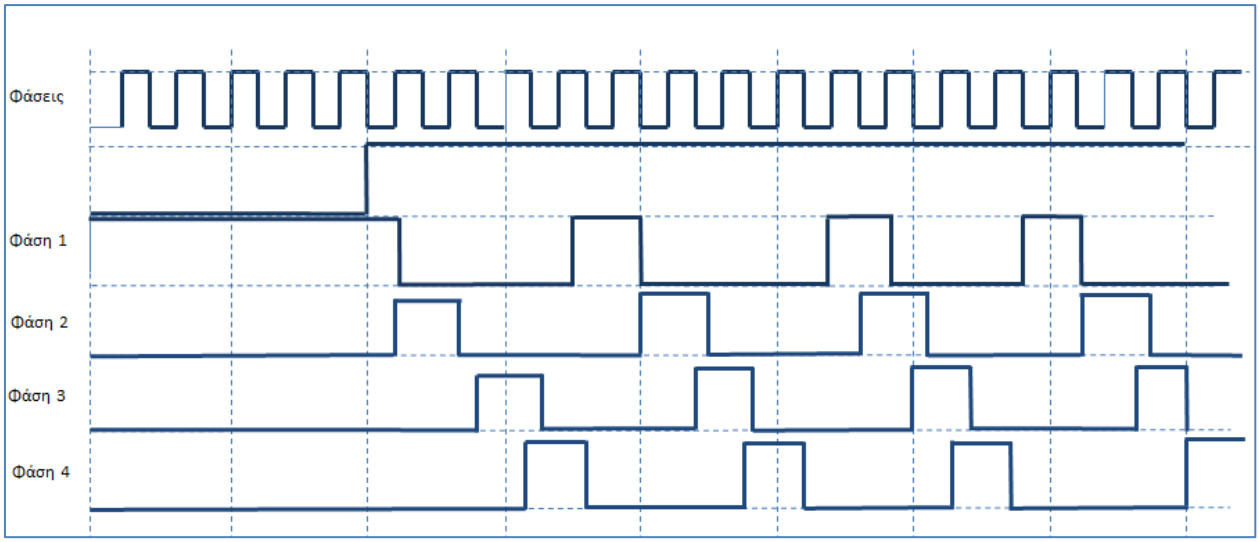
Εικόνα 2.8: Εσωτερική δομή του καταχωρητή εξόδου

2.6 Ο μετρητής των φάσεων

Προκειμένου να μετατραπεί ο προηγούμενος «χειροκίνητος» μικροεπεξεργαστής σε πλήρως προγραμματιζόμενο μικροεπεξεργαστή, πρέπει να δημιουργηθούν αρκετά κυκλώματα για να υπάρχουν τα κατάλληλα σήματα ελέγχου. Πρώτον, ο μετρητής φάσης πρέπει να παράγει τα σήματα τεσσάρων φάσεων Phase0 έως Phase3, στο αρνητικό άκρο του ρολογιού. Ο μετρητής πρέπει να επαναρυθμιστεί με ένα σήμα "Clear" που είναι ενεργό χαμηλό. Ο σχεδιασμός του μετρητή φάσης βασίζεται στις θύρες μανδάλου και XOR, όπως φαίνεται στο σχήμα 19



Εικόνα 2.9: Η δομή του μετρητή φάσεων

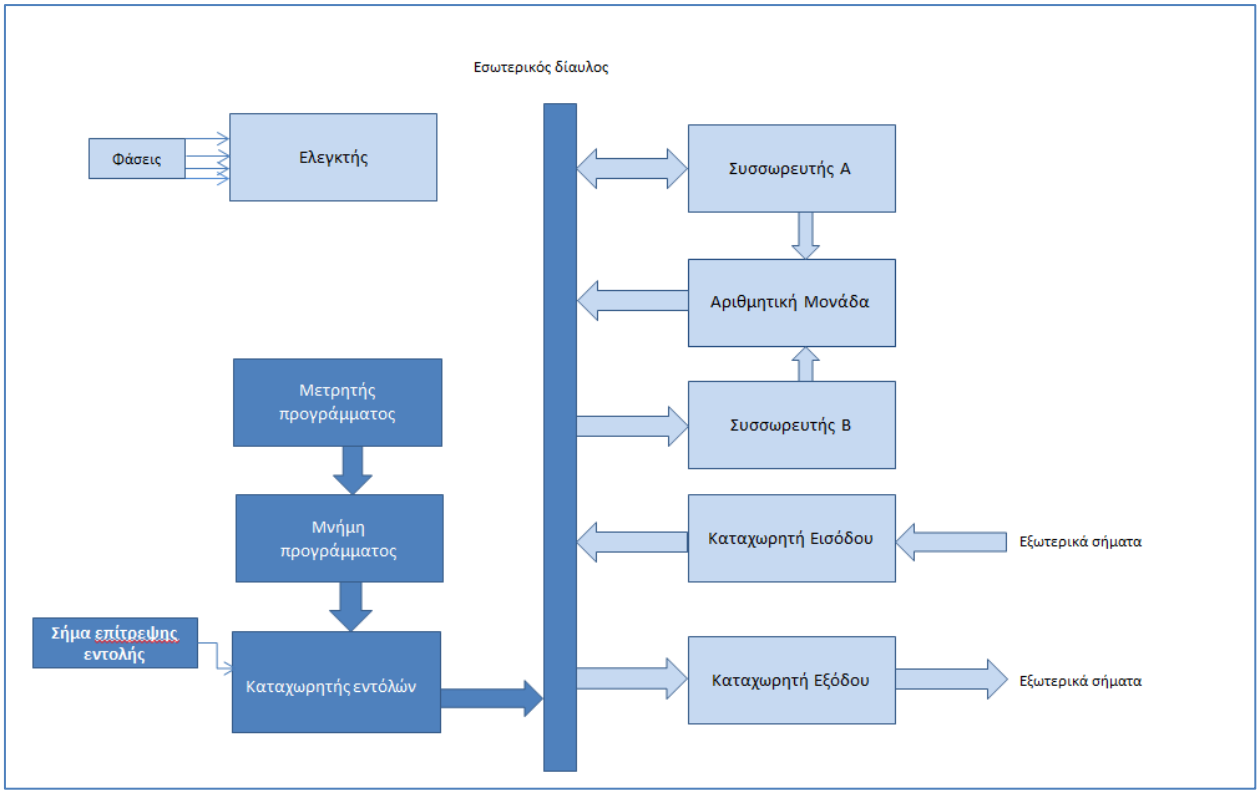


Εικόνα 2.9: Προσομοίωση του μετρητή φάσεων

.

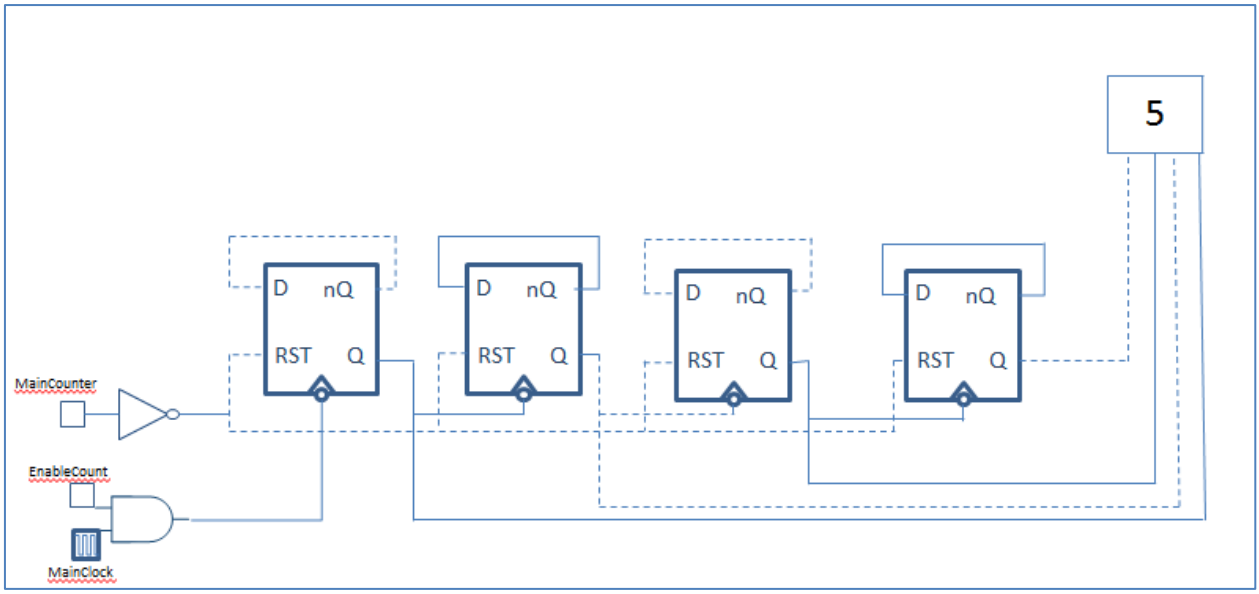
2.7 Ο μετρητής του προγράμματος

Ο μετρητής προγράμματος παίζει πολύ σημαντικό ρόλο στον μικροεπεξεργαστή καθώς τροφοδοτεί την κύρια μνήμη του προγράμματος με τη διεύθυνση της ενεργούς εντολής (Εικόνα 21). Στην αρχή, ο μετρητής προγράμματος είναι 0. Στη συνέχεια, στο τέλος κάθε εντολής, ο μετρητής προγράμματος αυξάνεται, για να επιλεχθεί την επόμενη εντολή.



Εικόνα 2.10: Ο μετρητής προγράμματος τροφοδοτεί τη μνήμη του προγράμματος με τη διεύθυνση της ενεργής εντολής

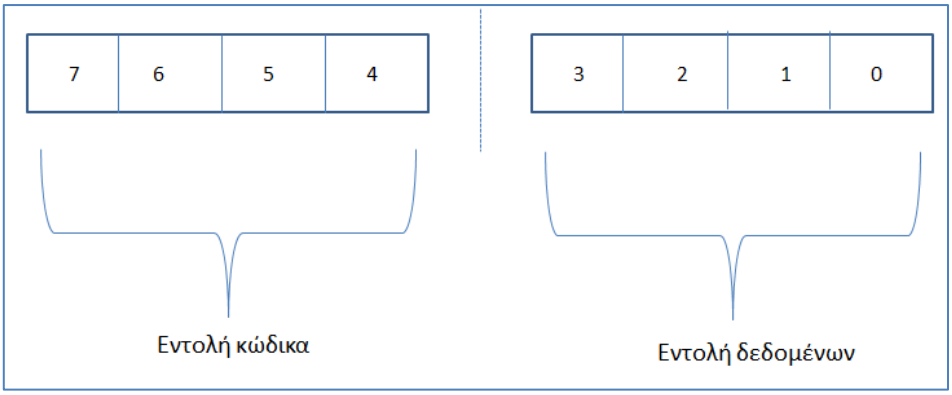
Ένας απλός τρόπος για τη δημιουργία ενός μετρητή από το 0 έως το 15 είναι να συνδεθούν ευαίσθητοι στα άκρα D-καταχωρητές, όπως φαίνεται παρακάτω. Το κύκλωμα είναι πολύ απλό, αλλά λειτουργεί ασύγχρονα. Αυτό σημαίνει ότι λόγω των καθυστερήσεων διάδοσης μεταξύ των σταδίων, κάποια ενδιάμεσα αποτελέσματα εμφανίζονται στην οθόνη σε πολύ σύντομο χρονικό διάστημα. Οι δυσλειτουργίες δεν έχουν αντίκτυπο στον σχεδιασμό του μικροεπεξεργαστή καθώς ο μετρητής κυκλωμάτων αυξάνεται κατά τη διάρκεια της φάσης 2 της ακολουθίας μικροεντολών και χρησιμοποιείται μόνο στην επόμενη εντολή κατά τη διάρκεια της φάσης 1 για τη φόρτωση του καταχωρητή εντολών.



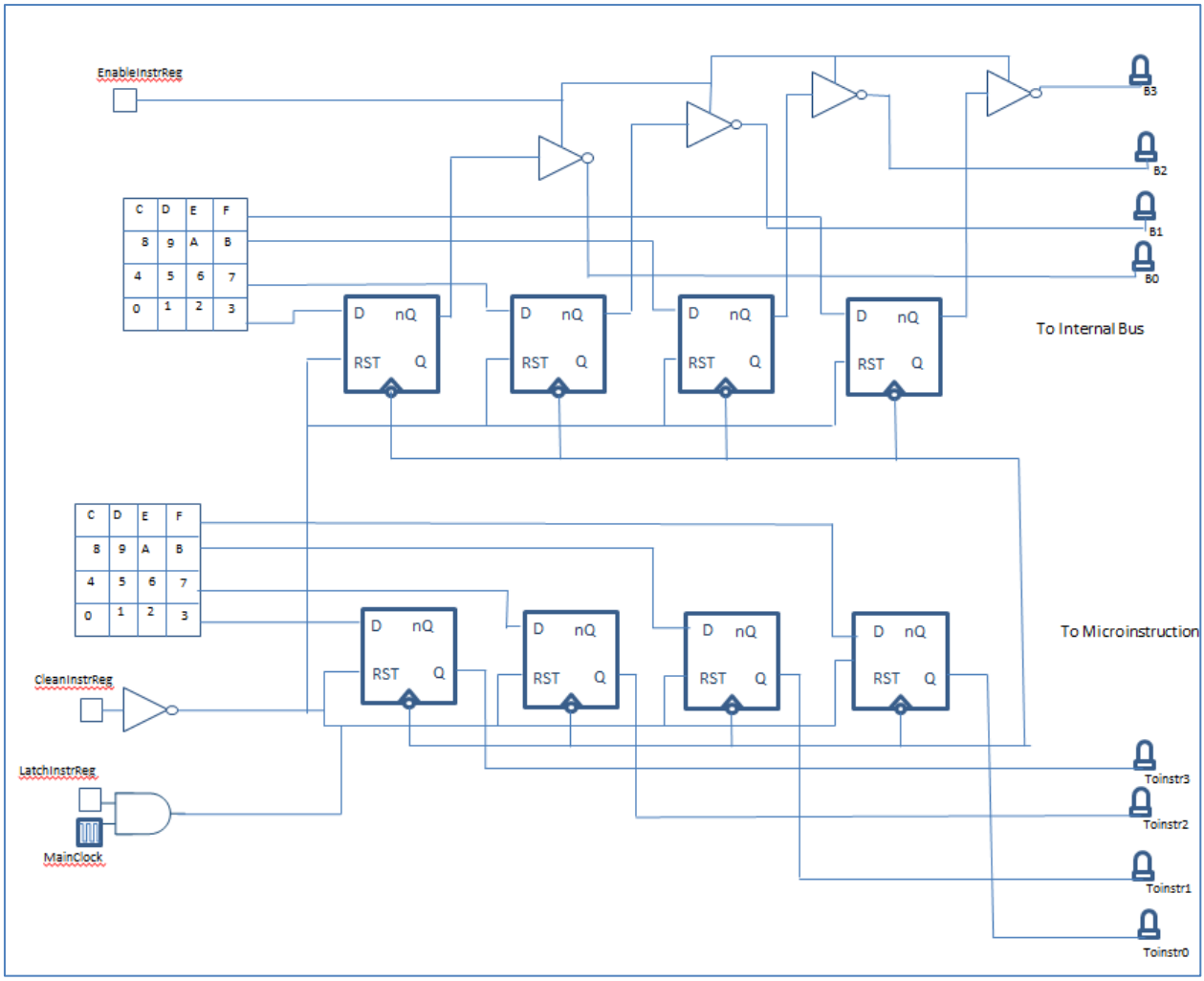
Εικόνα 2.11: Ο μετρητής προγράμματος κατά την λειτουργία. Η καταμέτρηση ενεργοποιείται μόνο κατά τη διάρκεια της φάσης 2, στην πτώση του κύριου ρολογιού.

2.8 Ο καταχωρητής των εντολών

Ο καταχωρητής εντολών αποθηκεύει τα περιεχόμενα της μνήμης προγράμματος. Οι 8-bit πληροφορίες χωρίζονται σε δύο μέρη: τα πιο σημαντικά bits αντιστοιχούν στην ίδια την εντολή, ενώ τα λιγότερο σημαντικά δυφία είναι τα δεδομένα. Ο κώδικας εντολών αποθηκεύεται από τα τέσσερα κυκλώματα καταχωρητών D που βρίσκονται στο κάτω μέρος του σχήματος 23, προκειμένου να είναι μόνιμα διαθέσιμα για τον αποκωδικοποιητή μικροεντολών, ενώ τα δεδομένα αποθηκεύονται σε τέσσερεις καταχωρητές D και μπορούν να διατίθενται στον εσωτερικό δίαυλο. Ο κατάλογος οδηγιών δίνει ένα αντίγραφο της τρέχουσας εντολής και απελευθερώνει την κύρια μνήμη, η οποία μπορεί να προσπελαστεί αργότερα για λειτουργία ανάγνωσης ή εγγραφής.



Εικόνα 2.12: Η 8-bit πληροφορίας κατηγοριοποιημένη σε 2 βασικές κατηγορίες.



Εικόνα 2.13: Ο καταχωρητής εντολών αποθηκεύει το περιεχόμενο της μνήμης και διαχωρίζει το τμήμα κώδικα (Κάτω καταχωρητές) από το τμήμα δεδομένων (άνω καταχωρητές)

ΚΕΦΑΛΑΙΟ 3Ο – Η ΓΛΩΣΣΑ VHDL

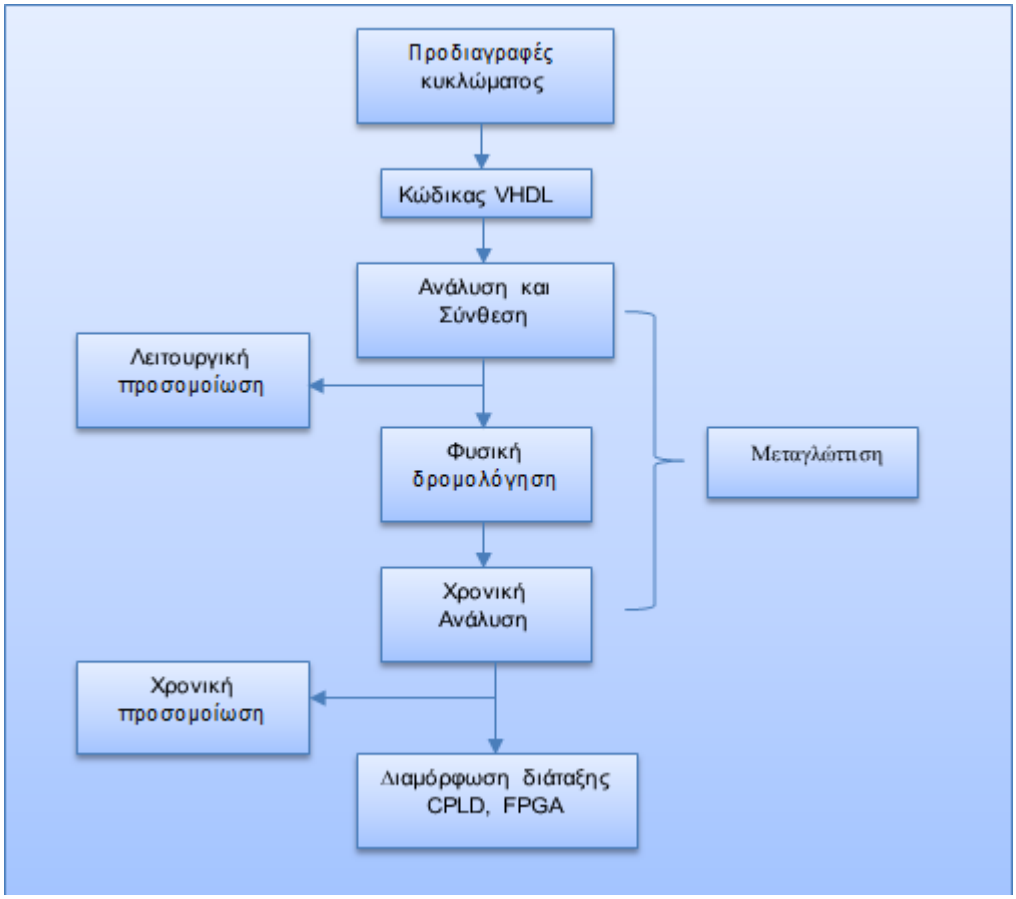
3.1 Εισαγωγή

Η γλώσσα VHDL χρησιμοποιείται για την σχεδίαση ολοκληρωμένων ψηφιακών κυκλωμάτων και συστημάτων. Η δημιουργία της είχε στόχο στη μοντελοποίηση και στη προσομοίωση κυκλωμάτων και συστημάτων με αποτέλεσμα πολλά χαρακτηριστικά της γλώσσας να έχουν αντικειμενικό στόχο την προσομοίωση λειτουργιών. Μετέπειτα, η γλώσσα εμπλουτίστηκε με δυνατότητας εργαλείων σύνθεσης. Κατά τη φάση της σύνθεσης, ο μεταγλωττιστής δημιουργεί ένα κύκλωμα του οποίου η συμπεριφορά περιγράφεται επακριβώς από το πρόγραμμα.

Η γλώσσα αυτή δημιούργήθηκε τη δεκαετία του 1980 με συγχρηματοδότηση του υπουργείου Άμυνας των ΗΠΑ και εφαρμόστηκε το 1987 από το ινστιστούτο IEEE το 1987. Ο όρος VHDL είναι συντομογραφία της VHSIC(Hardware Description Language), που VHSIC σημαίνει Very High Speed Integrated Circuit. Στα συνέχεια κατασκευάστηκα και βελτιωμένες εκδόσεις όπως η *IEEE 1164* (1993), *IEEE 1076-2002* και *IEEE 1076-2008*.

Λογισμικά τύπου ψηφιακής σχεδίασης κυκλωμάτων χρησιμοποιούν ως πρωταρχικό εργαλείο τη γλώσσα VHDL και συγκεκριμένα τα προγράμματα που είναι γραμμένα σε αυτήν. Επίσης χρησιμοποιείται και στον τομέα των FPGAs και CPLDs, αλλά έχει προτυποποιηθεί για σχεδίαση κυκλωμάτων Application Specific Intergrated Ciruits(ASICs).

Τα βασικά βήματα που ακολουθεί ένας σχεδιαστής λογικών κυκλωμάτων παρουσιάζονται στην εικόνα 3.1. Στο πρώτο βήμα θέτονται οι προδιαγραφές του κυκλώματος με ακρίβεια για τον επιθυμητό στόχο. Συνεχίζοντας, το κύκλωμα περιέχει ιεραρχική δομημένη σχεδίαση ακολουθώντας κανόνες ιεραρχικής σχεδίασης. Το μέρη του κυκλώματος περιγράφονται με γλώσσα VHDL, με σκόπο να υπάρχει λειτουργικός συνδιασμός μεταξύ τους.



Εικόνα 3.1 Βασική ροή εργασιών κατά τη σχεδίαση με τη γλώσσα VHDL

Τα αρχεία σχεδίασης που χρησιμοποιούνται για τη δημιουργία ενός συστήματος καθώς και τα αρχεία που εχούν βασικό στόχο τη προσομοίωση και τη διαμόσφωση αποτελούν ένα ολοκληρωμένο project καθώς αποθηκεύονται στον ίδιο φάκελο. Ο εκτελεσμένος κώδικας αποθηκεύεται με επέκταση .vhd. Η περιγραφή του κώδικα αποτελείται από ένα αρχείο όταν η σχεδίαση είναι μικρή , που σε σύγκριση με τις μεγάλες σχεδιάσεις, αποτελούνται απο πολλά αρχεία ή αλλιώς design units. Αναλύοντας διεξοδικά το ιεραρχικό σκέλος ,το πρώτο αρχείο δηλώνεται ως «οντότητα ανώτερου επιπέδου».Τα υπόλοιπα κατώτερα σε ιεραρχεία αρχεία επικοινωνούν με το ανώτερο σκέλος με κατάλληλες δηλώσεις και αναφορές.Το σύνολο των αρχείων που υπάρχουν καθώς και μετέπειτα αρχεία που χρησιμοποιούνται για τη διαμόρφωση και τη προσομοίωση του προγράμματος αποθηκεύονται σε έναν φάκελο εργασίας.

Αμέσως μετή τη συγγραφή του προγράμματος ακολουθεί η μεταγλώττιση. Κατά τη διαδικασία της μεταγλώττισης γνωρίζεται το πρώτο και βασικό στάδιο της που ονομάζεται ανάλυση. Στο στάδιο της ανάλυσης γίνεται επεξεργασία του κώδικα για λάθη συντακτικού υπόβαθρου που μπορούν διορθωθούν με την επιστροφή σχολίων στη χρήστη .

Επόμενο στάδιο μετά τη την ανάλυση αποτελεί η σύνθεση που με τη χρήση του compiler σχεδιάζει και προσομοιώνει το κύκλωμα που δημιουργεί ο κώδικας. Η σύνθεση είναι από τα βασικότερα στάσια στη συνολική ροή εργασιών.

Επόμενο επίπεδο αποτελεί η φυσική τοποθέτηση και δρομολόγηση,κατά τη σύνθεση στην οποία έχει δημιουργηθεί κάθε λογική δομή βρίσκεται η φυσική της αντιστοίχηση μέσα σε μια

λογική βαθμίδα που βρίσκεται στη διάταξη.Αποτέλεσμα αυτής της διαδικασίας είναι ο υπολογισμός χρονικών καθυστερίσεων που δημιουργούνται από τα σήματα.

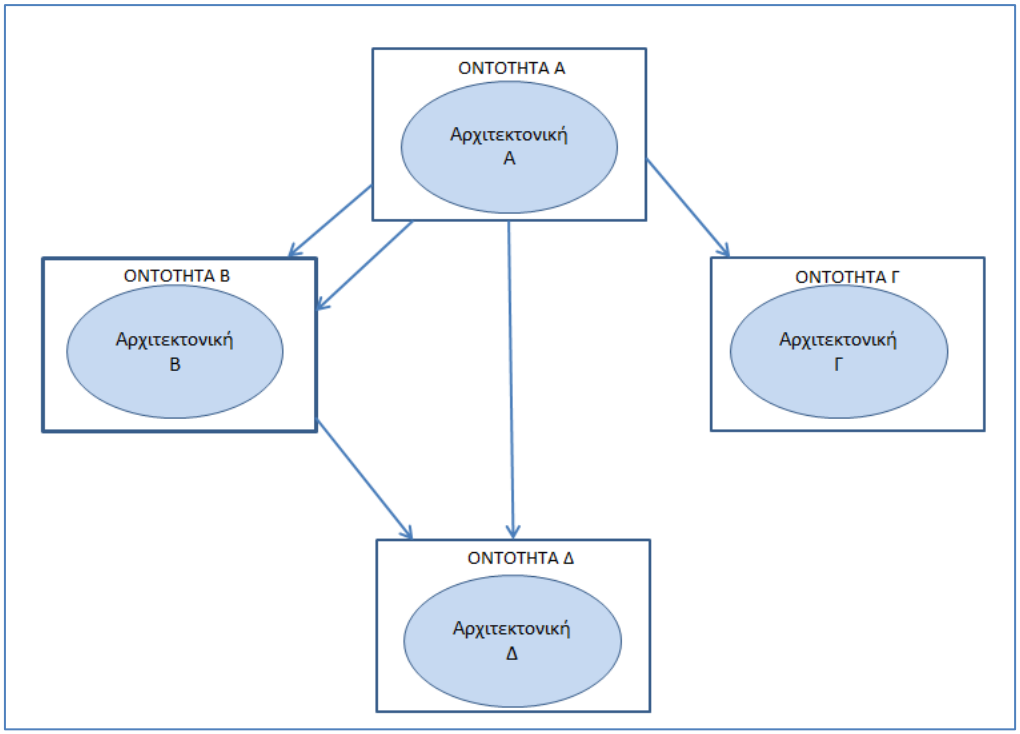
Μετά τη πραγματοποίηση των ανωτέρο σταδίων και παράλληλα τα αποτελέσματα είναι σύμφωνα με τις αντίστοιχες προδιαγραφές , είναι εφικτό να διαμορφωρθεί την αντίστοιχη διάταξη και να φορτωθεί το αρχείο διαμόρφωσης που προέκυψε στα τελικά επίπεδα της μεταγλώττισης.Από το σημείο αυτό το πρόγραμμα είναι σε θέση να αναταπεξέλθει στα σήματα εισόδου μέσω των αντίστοιχων καταχωρητών εισόδου αλλά και να παράγει τα αντίστοιχα σήματα εξόδου μέσω των κατάλληλων καταχωρητών εξόδου.

3.2 Χαρακτηριστικά της γλώσσας

Η VHDL προέρχεται από τη γλώσσα προγραμματισμού Ada και έχει χαρακτηριστικά στοιχεία του παράλληλου προγραμματισμού. Ο σχεδιασμός ενός προγράμματος στη VHDL μπορεί να γίνει ακολουθώντας τις βασικές αρχές του δομημένου προγραμματισμού, επιτρέποντας της top down σχεδίαση προγραμματών.Τέλος η γλώσσα που μελετάται υπακούει στις αρχές συγχρονισμού και χρονισμού, καθώς μια από τις ιδιαιτερότητες της VHDL είναι ότι προσομοιώνει με λεπτομέριες τις λειτουργίες του κυκλώματος και να παράγουν τα αντίστοιχα αποτελέσματα.

Η VHDL διαφέρει από τις υπόλοιπες γλώσσες προγραμματισμού διότι δεν περιγράφει με σειριακό τρόπο τις λειτουργίες της. Χαρακτηριστικό είναι ότι μέρη του κώδικα δημιουργούν αποτελέσματα σε απόλυτο συγχρονισμό με άλλες λειτουργίες του προγράμματος . Τα τελικά αποτελέσματα της μοντελοποιήσης παράγονται σε διάφορα μέρη του κυκλώματος με κυρίαρχο ρόλο τη χρονική ακρίβεια. Με τη διαδικασία αυτή είναι λογικό ενα μέρος του κώδικα μπορεί να υλοποιηθεί σε οποιδήποτε σημείο καθώς το αποτελέσμα μπορεί να εκτελεστεί ανεξαρτήτως της σειράς του.

Οι ενέργειες που εκτελούνται στην VHDL έχουν την ικανότητα να δημιουργούν α)παράλληλα αποτελέσματα με την τοποθέτηση εισόδων ή β) αποτελέσματα που αναπαράγονται με συγχρονισμό με κλασσικό παράδειγμα τους παλμούς ρολογιού.



Εικόνα 3.2 Ιεραρχική σύνδεση οντοτήτων στη δομημένη σχεδίαση με τη γλώσσα VHDL

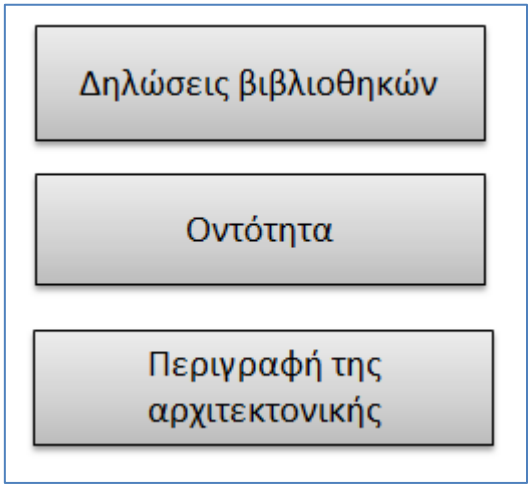
Στην VHDL η ιεραρχική δομή αποτελείται κυρίως από τη βαθμίδα που αποτελεί περιγραφή του κυκλώματος καθώς και από τη περιγραφή της λειτουργικοτητάς του.Σχεδιάζοντας ένα ψηφιακό κύκλωμα σε γλώσσα VHDL,είναι εύκολο να το συμπεριλάβουμε σε πιο σύνθετα κυκλώματα κάνοντας απλά «κλήση» στην οντότητα χωρίς να εμπεριέχεται η αναφορά στην αρχιτεκτονική.

Σημαντικό χαρακτηριστικό στη VHDL είναι η ευκολία στην επανάληψη των στυγμιοτύπων ,γνωστό και ως επαναχρησιμοποίηση για τη περιγραφή του υλικού , που γίνεται με αναφόρα στην αρχιτεκτονική της ανώτερης βαθμίδας της ιεραρχίας. Κάθε κύκλωμα που περιγράφεται μία φορά μπορεί να χρησιμοποιηθεί ξανά σαν βαθμίδα υποκυκλώματος, σε οποιοδήποτε άλλο ψηφιακό σύστημα. Με αυτή τη διαδικασία επιτεύχθηκε τόσο το hardware όσο και το software να είναι προσβιβάσιμο σε όλους τους χρήστες, είτε με τη μορφή λογισμικού είτε με τη μορφή βιβλίοθηκης όπως για παράδειγμα σε εργαλεία για ψηφιακή σχεδίαση(QuartusII).

3.3 Δομή προγράμματος VHDL

Στη VHDL τα δύο βασικά δομικά μέρη του προγράμματος αποτελούνται από την *οντότητα* και την *αρχιτεκτονική*.Δημιουργώντας το κώδικα θα πρέπει να συμπεριλάβουμε ότι πριν από τα δύο βασικά μέρη της γλώσσας ,θα πρέπει να δηλώσουμε τις βιβλιοθήκες που έχουν το σκοπό τη περιγρταφή των τύπων δεδομένων.

Ο σκοπός των δηλώσεων των βιβλιοθηκών είναι η ευκολία στο χρήστη να χρησιμοποιεί έναν ήδη υπάρχων κώδικα που θα το συμπεριλάβει στο κύριο προγραμμά του. Τέτοιες βιβλιοθήκες αποτελούν τμήματα κώδικα που συμπεριλαμβάνουν συναρτήσεις , υποκυκλώματα ή ακόμα και διαδικασίες.



Εικόνα 3.3 Βασικά τμήματα ενός αρχείου VHDL

Παράδειγμα βιβλιοθήκης που εφαρμόζεται συχνά είναι η *std_logic_1164* της βιβλιοθήκης *ieee*, η οποία χρησιμοποιείται για την περιγραφή δεδομένων std_logic. Κατα τη δήλωση της βιβλιοθήκης εισάγεται η κωδική λέξη LIBRARY ενώ στη περίπτωση του πακέτου δηλώνεται η κωδική λέξη USE.

Στη VHDL η *οντότητα* περιγράφει ως βαθμίδα το κύκλωμα με όλες τις κατάλληλες εισόδους και εξόδους.Η ιδιαιτερότητα αυτής της βαθμίδας είναι ότι περιέχει όλες τις διασυνδέσεις του κυκλώματος με άλλες βαθμίδες αλλά δεν περιέχεται η λειτουργία του κυκλώματος. Κατά την ολοκλήρωση του τμήματος της οντότητας ο χρήστης δίνει το όνομα καθώς και τα σήματα εισόδου-εξόδου.

Στο δεύτερο τμήμα της VHDL είναι η *αρχιτεκτονική* που περιέχει όλα τα λεπτομερή μέρη στο τρόπο λειτουργίας του κυκλώματος καθώς περιλαμβάνονται όλες οι εντολές και οι δηλώσεις,

που προσδιορίζουν τις λογικές συναρτήσεις που υλοποιούνται καθώς και τις τιμές που λαμβάνουν τα σήματα σε ακριβής χρονικές στιγμές

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3 ---------------------------------------------------
4 ENTITY mux IS
5    PORT (x, y : IN std_logic;--input channels
6             s : IN std_logic;--selection line
7             f : OUT std_logic);--output channel
8 END mux;
9 ---------------------------------------------------
10 ARCHITECTURE behaviour OF mux IS
11 BEGIN
12    WITH s SELECT
13          f<= x WHEN '0',
14           y WHEN OTHERS;
15 END behaviour;
```

Κώδικας 1: Κώδικας VHDL για τη περιγραφή πολυπλέκτη 2:1.

3.3.1 Η οντότητα

Στη δημιουργία της οντότητας πρέπει να δηλώνεται για αρχή το όνομα της καθώς οι είσοδοι και οι έξοδοι στο κύκλωμα με σκοπό τη διασύνδεση με τις άλλες βαθμίδες. Συντακτικά μετά τη λέξη ENTITY είναι το όνομα που έχει θέσει ο χρήστης με ακόλουθο τη λέξη IS

```
ENTITY όνομα_οντότητας IS;
```

Κατά τη δήλωση των διασυνδέσεων είναι βασικό να υπάρχει η κωδική λέξη PORT ,που εποτελείται απο δύο παρενθέσεις και εντός αυτών συμπεριλαμβάνονται οι δηλώσεις των σημάτων:

```
PORT(όνομα_σήματος_1  :  τρόπος_λειτουργίας  τύπος_σήματος_1;
     όνομα_σήματος_2 : τρόπος_λειτουργίας  τύπος_σήματος_2;
     ……
```

όνομα_σήματος_N : τρόπος_λειτουργίας τύπος_σήματος_N);

Η κατεύθηνση του σήματος διασύνδεσης καθορίζεται από το τρόπο λειτουργίας και ανήκει σε από τις παρακάτω κατηγορίες που αναφέρονται :

- ***IN*** Βασική χρησιμότητα για τα σηματα που αποτελούν είσοδοι.
- ***OUT*** Αποτελεί συμβολισμό για τα σήματα εξόδου.
- ***INOUT*** Είναι σήματα που συνδέονται κυρίως με διαδρόμους που αποτελούνται από δύο κατευθύνσεις γνωστά και ως δικατευθυντήρια.
- ***BUFFER*** Αφορά τα σήματα εξόδου, που υπάρχει η δυνατότητα να διαβαστούν στο εσωτερικό της οντότητας.

Για να κλείσει μια οντότητα ακολουτείται η εξής εντολή :

END [ENTITY] [όνομα_οντότητας];

3.3.2 Η αρχιτεκτονική

Στο τμήμα της αρχιτεκτονικής(architecture) παρατηρούνται όλες οι λεπτομέριες περιγραφής ως προς τη λειτουργία του κυκλώματος.Η μορφή της δήλωσης της αρχιτεκτονικής είναι η εξής:

ARCHITECTURE όνομα_αρχιτεκτονικής OF όνομα_οντότητας IS

με όνομα_αρχιτεκτονικής να είναι το αλφαρτιθμητικό το οποίο ορίζεται από τον χρήστη και όνομα_οντότητας να είναι το ίδιο αλφαριθμητικό που έχει ορίσει ο χρήστης για να ονομάσει την οντότητα.

Η διασικασία της αρχιτεκτονικής ξεκινάει με τη λέξη BEGIN . Στην οντότητα τα σήματα που έχουν οριστεί μεταφέρονται στο τμήμα της αρχιτεκτονικής αντίστοιχα με τους τύπους της. Στο διάστημα μεταξύ της αρχιτεκτονικής και του BEGIN είναι δυνατόν να υπάρχουν και άλλες δηλώσεις σημάτων εισόδου και εξόδου τα όποια δεν έχουν δηλωθεί αρχικά ως σήματα εισόδων/εξόδων και είναι απαραίτητα για τη λειτουργία της αρχιτεκτονικής.

Η αρχιτεκτονική για ένα κύκλωμα ορίζεται ως:

```
ARCHITECTURE όνομα_αρχιτεκτονικής OF όνομα_οντότητας
IS [Δηλώσεις επιπλέον σημάτων]
BEGIN
Εντολές που περιγράφουν λογικές λειτουργίες και αναθέτουν
τιμές σε σήματα
END [ARCHITECTURE] [όνομα_αρχιτεκτονικής];
```

Ο αγκύλες στο σημείο της δήλωσης επιπλέον σημάτων είναι προαιρετικό.

Στο παράδειγμα φένεται ότι ανάμεσα στο BEGIN και το END είναι οι εντολές που περιγράφουν τις λογικές λειτουργίες του κυκλώματος. Στο κώδικα 1 παράτηρείται η δημιουργία ενός πολυπλέκτη 2:1. Πιο αναλυτικά φένεται ότι στον πολυπλέκτη η έξοδος μεταφέρεται στο κανάλι x αν υπάρχει σήμα s το οποίο είναι '0' ή αλλιώς όταν το σήμα s είναι '1'τότε η έξοδος μεταφέρεται στο κανάλι y.Με την εντολή SELECT δίνεται τιμή στο σήμα εξόδου.Η εντολή SELECT καλύπτει όλες τις τιμές του σήματος εισόδου s. Τα σήματα std_logic δε περιέχουν μόνο τις τιμές '0' και '1' αλλά περιλαμβάνουν άλλους εννέα χαρακτήρες (π.χ 'Z','X') ,γι'αυτο και στο κώδικα 2.1 στη σειρά 14 η εντολή WHEN OTHERS κλείνει όλες τι υπόλοιπες περιπτώσεις που θα πρέπει να καλυφθούν.

3.4 Παράσταση αριθμών και χαρακτήρων στη VHDL

Οι αριθμητικές μεταβλητές που υποστηρίζει η VHDL είναι κυρίως ακέραιες και δυαδικές τιμές, όσο για τις πράξεις που μπορούν να πραγματοποιηθούν είναι για προσημασμένους και μη προσημασμένους αριθμούς. Οι ακέραιες τιμές στη VHDL είναι στη μορφή του δεκαδικού συστήματος, ενώ για τις δυαδικές τιμές γράφονται συνήθως με μονά ή διπλά εισαγωγικά(π.χ '0') καθώς και οι τιμές των πολλών ψηφίων μπορούν να αναπαρασταθούν σε δεκαεξαδική μορφή(π.χ b "1010").Η πληροφορία επίσης αναπαράγεται και σε μορφή χαρακτήρων.

Πιο αναλυτικά , οι προσημασμένοι αριθμοί ανήκουν στη περιοχή (-2^{N-1} ως $2^{N-1}-1$). Ως προς τους προσημασμένους αρνητικούς αριθμούς προκύπτουν και παριστάνονται από το συμπλήρωμα του 2 το οποίο βγαίνει από το συμπλήρωμα του αριθμού στο 1 προσθέτοντας τη μονάδα.Το σημαντικότερο ψηφίο δίχνει το πρόσημο του αριθμού (π.χ. αν το MSB είναι 1 τότε ο αριθμός είναι θετικός , αν αντίστοιχα το MSB είναι ίσο με 0 τοτε ο αριθμός θεωρείται αρνητικός).

Ωστόσο στους μη-προσημασμένους αριθμούς ισχύει ότι με N-ψηφία καλύπτουν το πεδίο τιμών απο $0-2^{N}-1$ που καθηστά λογικό ότι οι είσοδοι και οι έξοδοι αποτελούν θετικοί αριθμοί στο πεδίο που καθορίστηκε.

Στη VHDL ο χαρακτήρες της συνδέονται άρρηκτα με τους χαρακτήρες από το κώδικα ASCII, όπως για παράδειγμα το 'A' το οποίο ο μεταγλωτιστής θα το δημιουργήσει αποκτώντας τιμές από τον πίνακα του κώδικα ASCII.

3.5 Τελεστές και χαρακτηριστικά τους

Οι τελεστές αποτελούν κομμάτι κάθε τύπου δεδομένων . Παρακάτω απεικονίζονται οι κατηγορίες τελεστών:

- ***Αριθμητικοί τελεστές***:Οι συγκεκριμένοι τελεστές αποτελούν τις βασικές μαθηματικές πράξεις όπως της πρόσθεσης, αφαίρεσης, πολλαπλασιασμού και διαίρεσης, ενώ υποστηρίζεται η ύψωση σε δύναμη, η ακέραια διαίρεση και το υπόλοιπο της ακέραιας διαίρεσης. (+,–, *, /, **, DIV, MOD)
- ***Σχεσιακοί τελεστές***: : οι σχεσιακοί τελεστές εφαρμόζονται στο ίδιο τύπο δεδομένων (π.χ. χαρακτήρες **>=** χαρακτήρες) με τη διαφορά ότι στους ακέραιους και τους πραγματικούς αριθμούς μπορούν να συγκρίνονται και μεταξύ τους (π.χ. ακέραιος **=** πραγματικός). Το αποτέλεσμα των συγκριτικών τελεστών είναι πάντα λογικού τύπου. (=, /=, >, <, >=, <=).
- ***Τελεστής συνένωσης***: (&).
- ***Τελεστές ολίσθησης***(SLL,SRL,κλπ)
- ***Λογικοί τελεστές (****NOT,AND,NAND,OR,XOR,NOR,XNOR)*

ΚΕΦΑΛΑΙΟ 4 – ΤΟ ΛΟΓΙΣΜΙΚΟ TINA

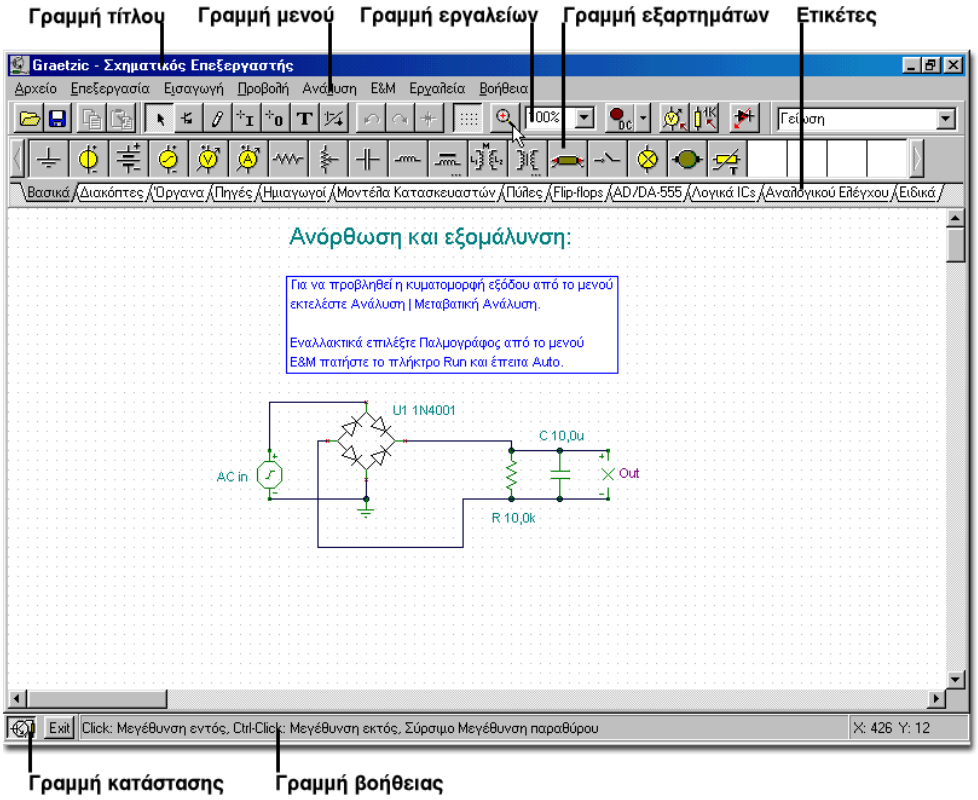
4.1 Εισαγωγή

Το TINA αποτελεί λογισμικό από την εταιρία **TEXAS INSTRUMENTS** με σκοπό την ανάλυση,σχεδίαση και προσομοίωση κυκλωμάτων.Εμπεριέχει μεγάλο μέρος από ολοκληρωμένα κυκλώματα καθώς και ηλεκτρονικών στοιχείων για τη σχεδίαση αναλογικών ή ψηφιακών κυκλωμάτων.Η βιβλιοθήκη εμπεριέχει παθητικά και ενεργητικά εξαρτήματα καθώς υπάρχει η δυνατότητα κατασκευής τροποποιημένων ή προσωπικών κυκλωμάτων.

Κατά το σχεδιασμό ενός κυκλώματος υπάρχει η δυνατότητα να υπάρχουν πλαίσια κειμένου καθώς και διαγράμματα.Υπάρχει η δυνατότητα σχεδιασμού κυκλώματος σε πραγματικό χρόνο και επιτρέπει την ανίχνευση σφαλμάτων και την ενεργοποίηση βλαβών.Επίσης το περιβάλλον εργασίας είναι οικείο και εύκολο στη χρήση, η γραμμή βοηθείας παράλληλα με τα παραδείγματα βοηθούν στην εύκολη και γρήγορη εκμαθησή του.

4.2 Σχηματικός Επεξεργαστής

Το κύριο παράθυρο εργασίας του λογισμικού φαίνεται στην εικόνα 4.1,



Εικόνα 4.1 Σχηματικός επεξεργαστής Tina

Στο TINA υπάρχει η ίδια λογική και φιλοσοφία με όλα τα υπόλοιπα προγράμματα στα Windows.

Στη *Γραμμή τίτλου* είναι το όνομα του κυκλώματος, η *Γραμμή μενού* εντολές όπως *Αρχείο, Εκτύπωση κ.τ.λ.* και τις εντολές καθορίζουν την επεξεργασία των κυκλωμάτων όπως είναι η *Ανάλυση*.

Η *Γραμμή εργαλείων* περιέχει εντολές που εφαρμόζονται συχνά π.χ. Αντιγραφή, Επικόλληση κ.α

Γραμμή εξαρτημάτων και οργάνων περιέχουν βασικά κατηγοριοποιημένα εξαρτήματα καθώς και όργανα μέτρησης. Στα *Βασικά* υπάρχουν εξαρτήματα πολύ συχνής χρήσης όπως αντιστάσεις, πυκνωτές. Στο τομέα των *Ημιαγωγών* υπάρχουν στοιχεία όπως δίοδοι, τρανζίστορ. Η επιλογή της γραμμής εξαρτημάτων κάθε ομάδας γίνεται πατώντας με το δείκτη του ποντικιού στην αντίστοιχη *Ετικέτα*.

Κάτω αριστερά βρίσκεται η *Γραμμή κατάστασης*, η οποία είναι η τρέχουσα κατάσταση του ενεργού παραθύρου. Με το πάτημα του πλήκτρου το οποίο βρίσκεται τέρμα αριστερά κλειδώνει ή ξεκλειδώνει το σχηματικό επεξεργαστή έτσι ώστε να αποκρύπτει τα διάφορα εικονικά όργανα ή παράθυρα.

Στη Γραμμή κατάστασης υπάρχει και η *Γραμμή βοήθειας*, όπου προβάλλονται σύντομες

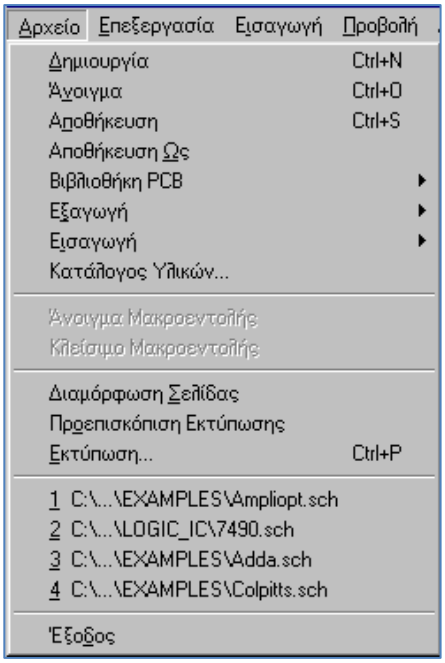
πληροφορίες όταν το ποντίκι βρίσκεται πάνω σε κάποιο εξάρτημα.

4.3 Γραμμή μενού

Με την επιλογή του ποντικιού στο μενού υπάρχει το αντίστοιχο *"Drop down menu"*. Τα μενού παραπέμπουν σε κάποιο άλλο παράθυρο, π.χ. Αρχείο και Εκτύπωση ή εκτελούν μια συγκεκριμένη εντολή π.χ. Επεξεργασία και Αντιγραφή.

4.3.1 Αρχείο

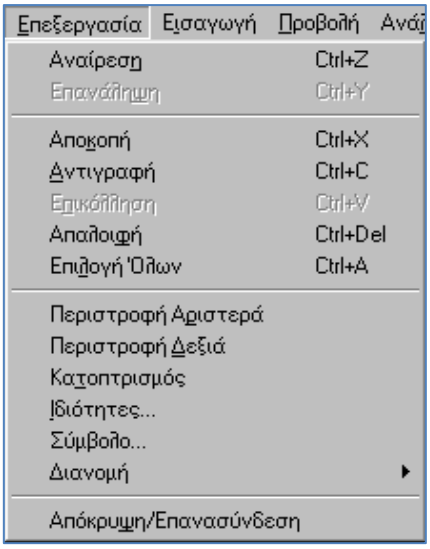
Στην επιλογή μενού περιλαμβάνονται οι κλασικές εντολές των Windows. Η εντολή *Δημιουργία* ανοίγει μια νέα σελίδα στο Σχηματικό Επεξεργαστή με σκοπό το σχεδιασμό και δημιουργία κυκλωμάτων. Η εντολή *Άνοιγμα* ανοίγει ένα υπάρχον κύκλωμα. Με την εντολή *Αποθήκευση* αποθηκεύεται το υπάρχον ή ένα νέο κύκλωμα, ενώ με την εντολή *Αποθήκευση Ως* δημιουργείτε ένα αντίγραφο ή μια νέα έκδοση του κυκλώματός μας. Με την εντολή *Εξαγωγή* εξάγεται ένα αρχείο του κυκλώματός σε μορφή PSpice (*.CIR), Windows Metafile (*.WMF) και σε μορφή PCB (*.NET). Με την εντολή *Εισαγωγή* εισάγεται αρχείο κυκλώματος τύπου TINA 2xx (*.ICE), Palmtop (*.SCH), Pspice(*.CIR). Με το Κατάλογο Υλικών βρίσκεται ο αντίστοιχος κατάλογος υλικών από ένα κύκλωμα. Οι εντολές *Άνοιγμα* και *Κλείσιμο Μακροεντολής* ελέγχουν τη διαδικασία ανοίγματος και κλεισίματος μακροεντολών του TINA . Η *Διαμόρφωση Σελίδας* επιτρέπει την αλλαγή των ρυθμίσεων της σελίδας για εκτύπωση, η *Προεπισκόπηση* δείχνει πως έχει διαμορφωθεί η σελίδα σύμφωνα με τις ρυθμίσεις και η *Εκτύπωση επιτρέπει* τη διαμόρφωση των παραμέτρων της εκτύπωσης.



Εικόνα 4.2 Περιεχόμενα του Αρχείου

4.3.2 Επεξεργασία

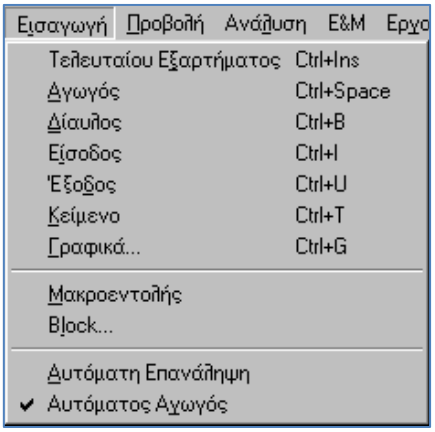
Το μενού της επεξεργασίας περιλαμβάνει εντολές για την επεξεργασία κυκλωμάτων όπως η *Αναίρεση, η Επανάληψη, η Αποκοπή, η Αντιγραφή, η Επικόλληση, η Απαλοιφή, η Επιλογή Όλων*. οι εντολές *Περιστροφή Αριστερά ή Δεξιά*, και ο *Κατοπτρισμός* είναι ιδιαίτερα χρήσιμες. Με αυτές τις εντολές πραγματοποιείται ο έλεγχος των εξαρτημάτων με σκοπό τη τοποθετησή τους στο σχεδιαστικό χώρο.Στην εντολή *Ιδιότητες* περιέχονται οι ιδιότητες και τα χαρακτηριστικά ενός επιλεγμένου εξαρτήματος ή μοντέλου. Με την εντολή *Σύμβολο* εισάγεται ή δημιουργήται ένα σύμβολο. Με την *Διανομή* δημιουργούνται διαφορετικές εκδόσεις ή κλειδώνεται ένα κύκλωμα. Η *Απόκρυψη/Επανασύνδεση* αφορά στην σύνδεση ή αποσύνδεση δύο αγωγών που δημιουργούν κόμβο.



Εικόνα 4.3 Περιεχόμενα της Επεξεργασίας

4.3.3 Εισαγωγή

Το μενού αυτό περιέχει εντολές όπως *Τελευταίο Εξάρτημα* που εισάγεται το εξάρτημα που χρησιμοποιήθηκε τελευταία. Με τις εντολές *Αγωγός, Δίαυλος, Είσοδος, Έξοδος, Κείμενο, Γραφικά, Μακροεντολή, Block* εισάγονται τα αντίστοιχα εξαρτήματα/εντολές. Η *Είσοδος* και η *Έξοδος* περιέχει τον τρόπο προβολής ορισμένων αναλύσεων. Η *Αυτόματη Επανάληψη* πραγματοποιεί την επανάληψη εισαγωγής εξαρτήματος στυο σχηματικό χώρο, ενώ η εντολή *Αυτόματος Αγωγός* δίνει τη δυνατότητα μετακίνησης εξαρτήματος που έχει σύνδεση αγωγού

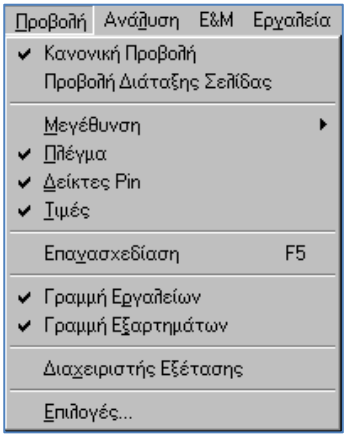


Εικόνα 4.4 Περιεχόμενα της Εισαγωγής

4.3.4 Προβολή

Στο μενού *Προβολή* γίνεται η επιλογή παρουσίασης του σχηματικού επεξεργαστή. Η επιλογή *Κανονική Προβολή* προβάλει την προκαθορισμένη μορφή του Σχηματικού Επεξεργαστή, όπως φαίνεται στην εικόνα 4.1. Η *Προβολή Διάταξης Σελίδας* περιλαμβάνει το Σχηματικό Επεξεργαστή σε μορφή σελίδας. Η εντολή *Μεγέθυνση* εμφανίζει ένα υπομενού για την επιλογή των τρόπων μεγέθυνσης του κυκλώματος. Οι διακόπτες *Πλέγμα, Δείκτες Pin, Τιμές, Γραμμή Εργαλείων και Γραμμή Εξαρτημάτων* έχουν τη κύρια επιλογή της προβολής ή της αποκρυπτογράφησης του περιεχομένου τους .Η εντολή *Επανασχεδίαση* καθαρίζει και επανασχεδιάζει το κύκλωμα. Με την επιλογή του *Διαχειριστή Εξέτασης,* αυτός προβάλλεται μέσα από το σχηματικό επεξεργαστή, και με τις *Επιλογές* εμφανίζεται ένα πλαίσιο διαλόγου που γίνεται δυνατή να επιτρέψει την επιλογή Ομάδων Συμβόλων, Μονάδων Μέτρησης και Βασικής συνάρτησης ημιτόνου στο AC.

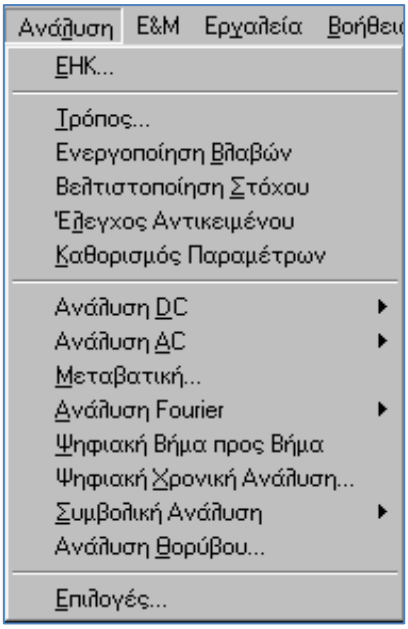
Εικόνα 4.5 Περιεχόμενα της Προβολής



4.3.5 Ανάλυση

Στο μενού *Ανάλυσης* γίνεται η διαμόρφωση και επιλογή ανάλυσης. Με τον ΕΗΚ ελέγχουμε αν έχουν πραγματοποιηθεί οι συνδέσεις σε ένα κύκλωμα. Από τον *Τρόπο* γίνεται η Επιλογή του Τρόπου της ανάλυσης, όπως ο *Απλός τρόπος, το Βήμα Θερμοκρασίας, το Βήμα Παραμέτρου, της Χειρότερης Περίπτωσης, Monte Carlo, της Βελτιστοποίησης.* Στην εντολή *Ενεργοποίησης Βλαβών* ενεργοποιούνται οι βλάβες που έχουν ορισθεί. Η *Βελτιστοποίηση Στόχου* πραγματοποιείται ως προς ένα στόχο, ως προς κάποια Τιμή ή ως προς μια μέγιστη ή ελάχιστη τιμή. Ο *Έλεγχος Αντικειμένου* αναφέρεται στην *Πολλαπλή Ανάλυση*. Ο *Καθορισμός Παραμέτρων* αφορά τις παραμέτρους μιας προσομοίωσης.

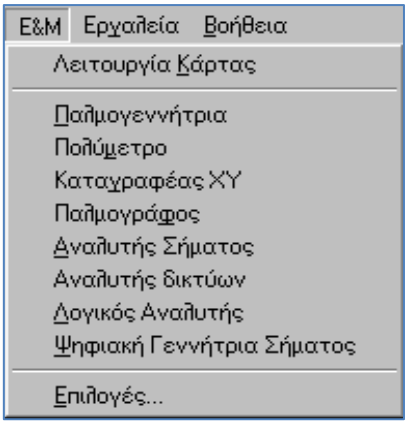
Στη συνέχεια την *Ανάλυση DC, τη Ανάλυση AC, τη Μεταβατική Ανάλυση, τη Ανάλυση Fourier. Επίσης,* οι ψηφιακές αναλύσεις, η *Βήμα προς Βήμα και η Ψηφιακή Χρονική Ανάλυση*. Τέλος η *Συμβολική, και η Ανάλυση Θορύβου.*



Εικόνα 4.6 Περιεχόμενα της Ανάλυσης

4.3.6 Εργαλεία και μετρήσεις

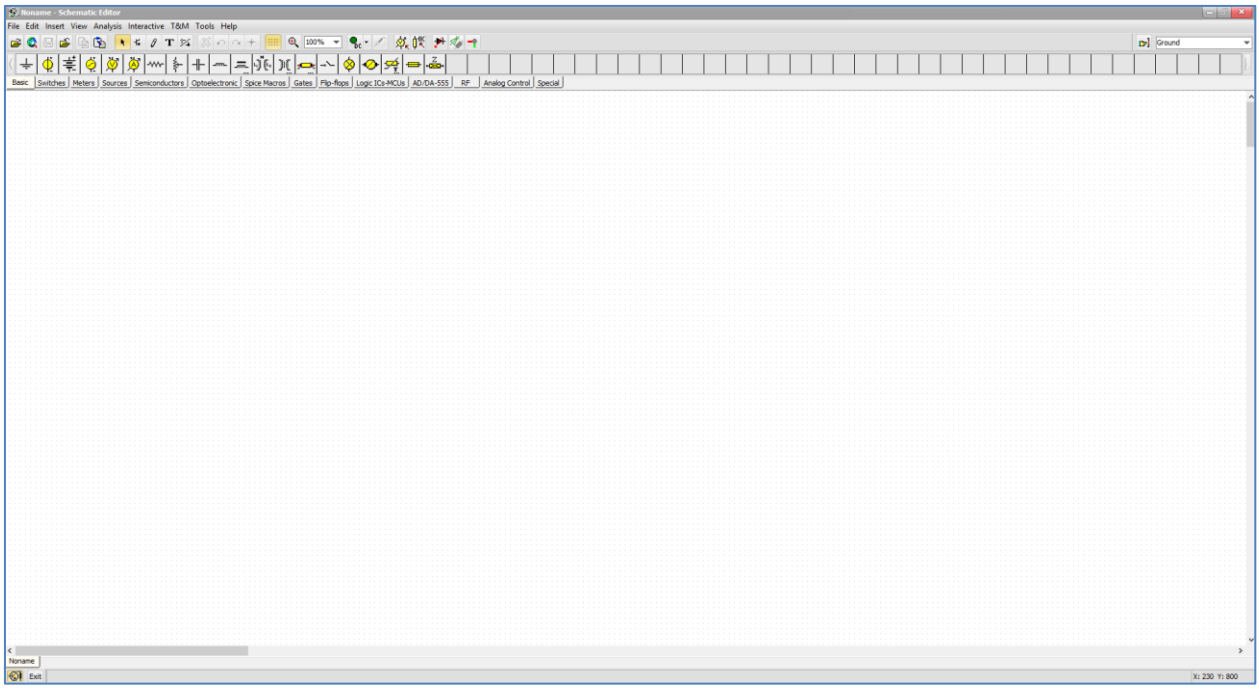
Στο μενού αυτό περιλαμβάνονται όλα τα εικονικά όργανα, καθώς και οι εντολές για τη υλοποίηση πραγματικών μετρήσεων στο περιβάλλον των εικονικών οργάνων. Η επιλογή Λειτουργία Κάρτας ενεργοποιείται εφόσον έχει τοποθετηθεί η ειδική κάρτα ανάκτησης δεδομένων (Tina Lab) στον Η/Υ για τις πραγματικές μετρήσεις, της οποίας η χρησιμότητα είναι ως πολύμετρο, παλμογράφος και παλμογεννήτρια. Στη συνέχεια υπάρχουν τα εικονικά όργανα μετρήσεων. Τέλος στις Επιλογές γίνεται η ενεργοποίηση της γεννήτριας και του εξοπλισμού.



Εικόνα 4.7 Περιεχόμενα των Εργαλείων και Μετρήσεων

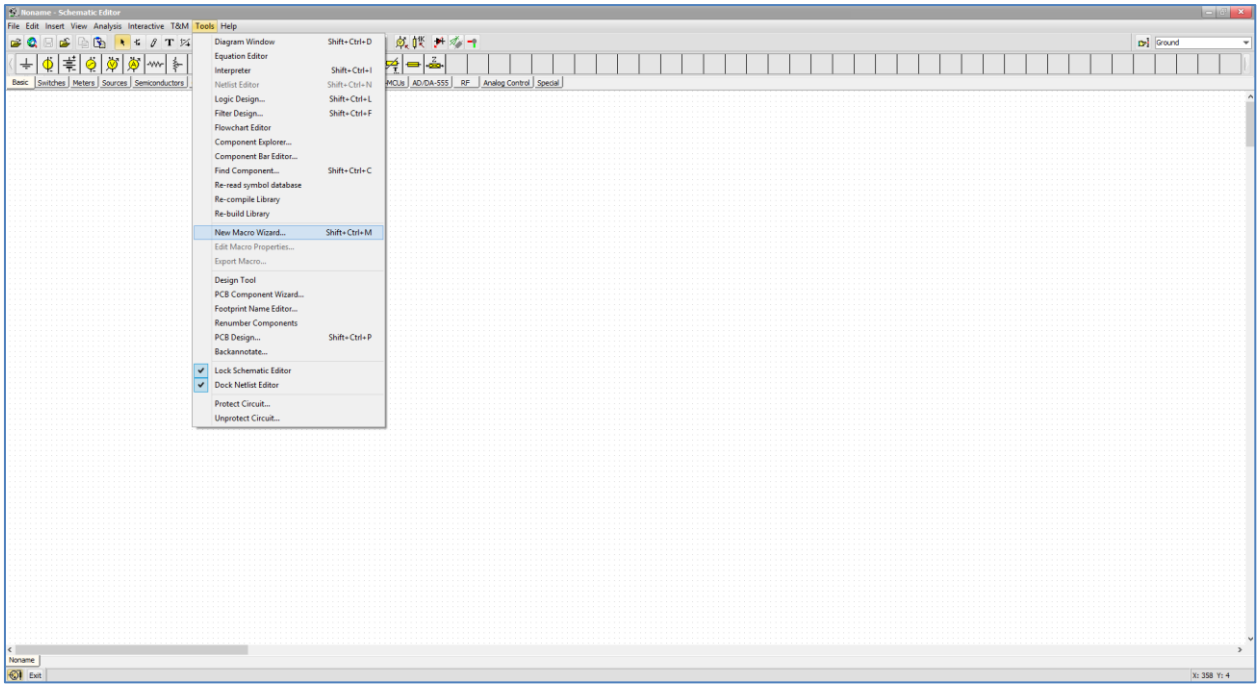
4.4 VHDL και MACROS

Στην παράγραφο θα περιγράψουμε αναλυτικά βήμα βήμα τη δημιουργία του macro από τον πηγαίο κώδικα vhdl της μονάδας πρόσθεσης/αφαίρεσης.



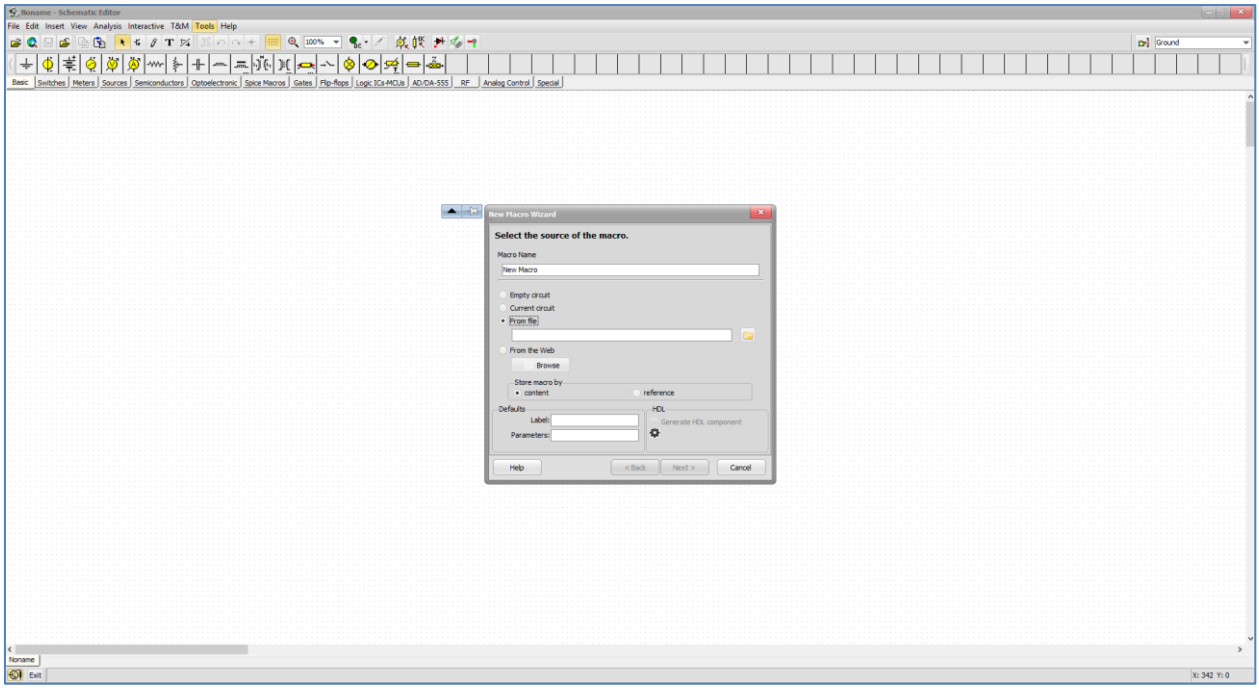
Εικόνα 4.8 Σχηματικός επεξεργαστής

Στην εικόνα 4.8 απεικονίζεται ο σχηματικός επεξεργαστής όπου θα πραγματοποιήσουμε την εισαγωγή της δομικής μονάδας πρόσθεσης/αφαίρεσης.



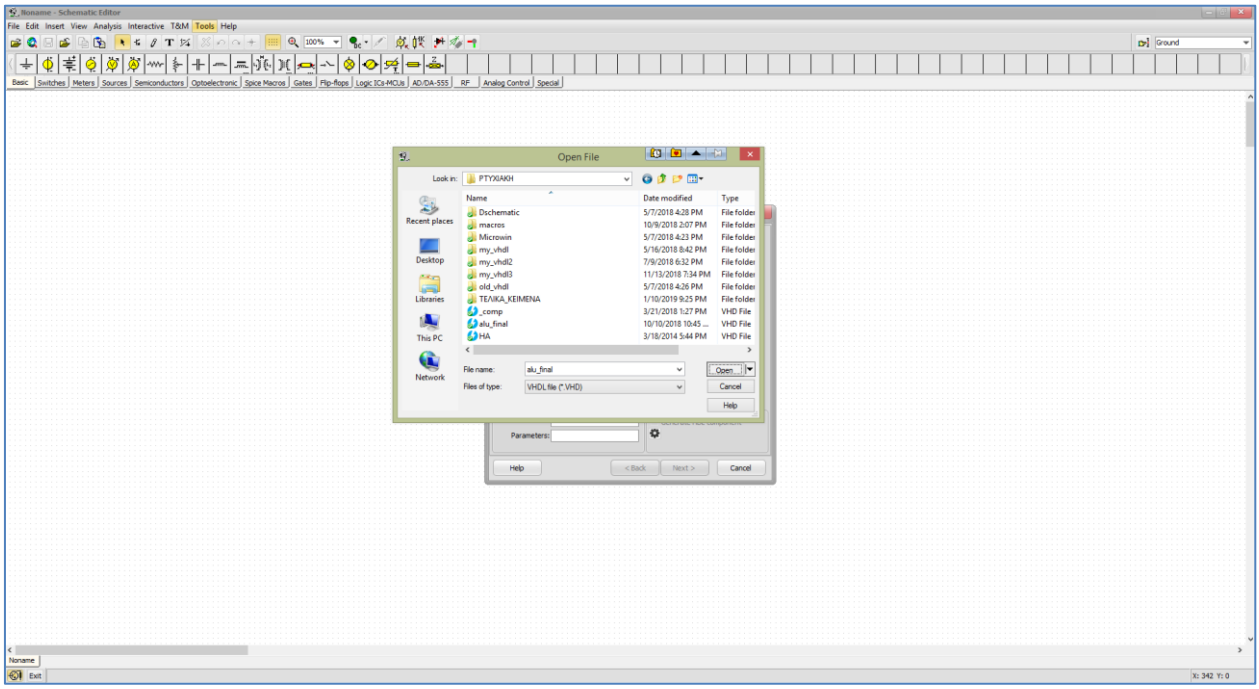
Εικόνα 4.9 Δημιουργία νέου macro

Στη συνέχεια όπως φαίνεται και στην εικόνα 4.9 επιλέγεται από τη γραμμή μενού η επιλογή *εργαλεία* και έπειτα διαλέγεται η υποκατηγορία *New Macro Wizard.*



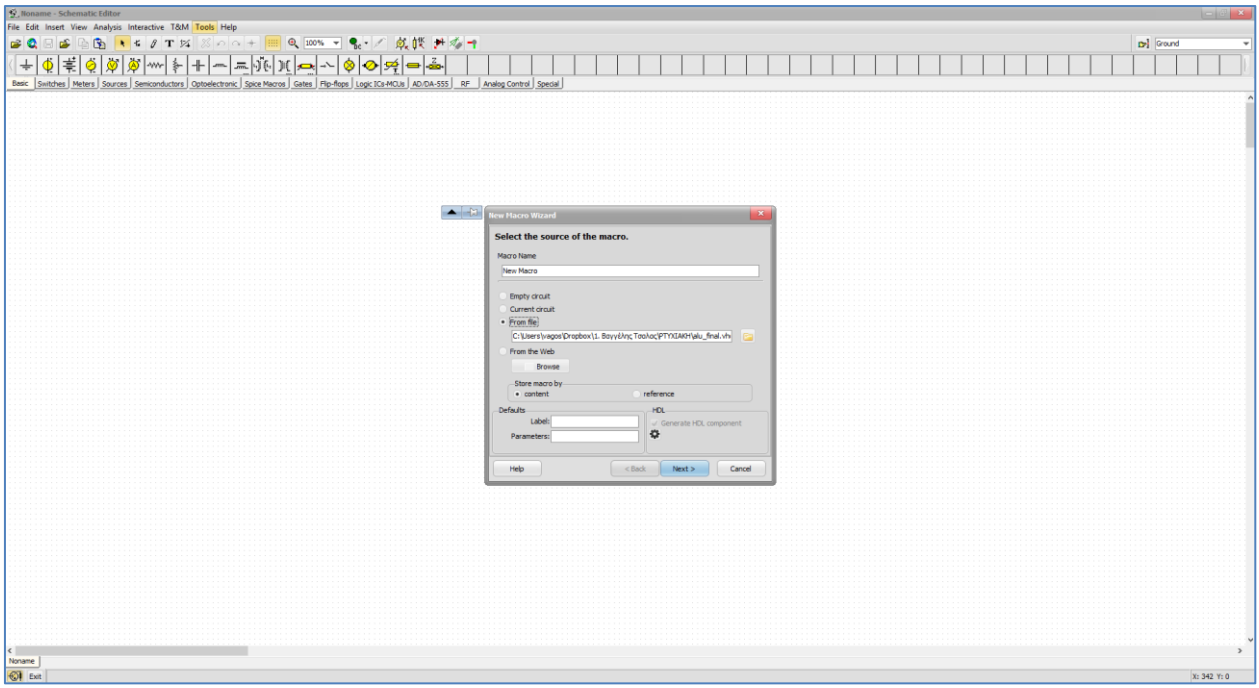
Εικόνα 4.10 Επιλογή αρχείου πηγής vhdl

Με την επιλογή *New Macro Wizard*, ενεργοποιείται παράθυρο στο οποίο διαλέγουμε την επιλογή *From file* και επιλέγουμε το εικονίδιο με το φάκελο για εισάγουμε το macros όπως φαίνεται στην εικόνα 4.10.



Εικόνα 4.11 Επιλογή του αρχείου μέσω File browser

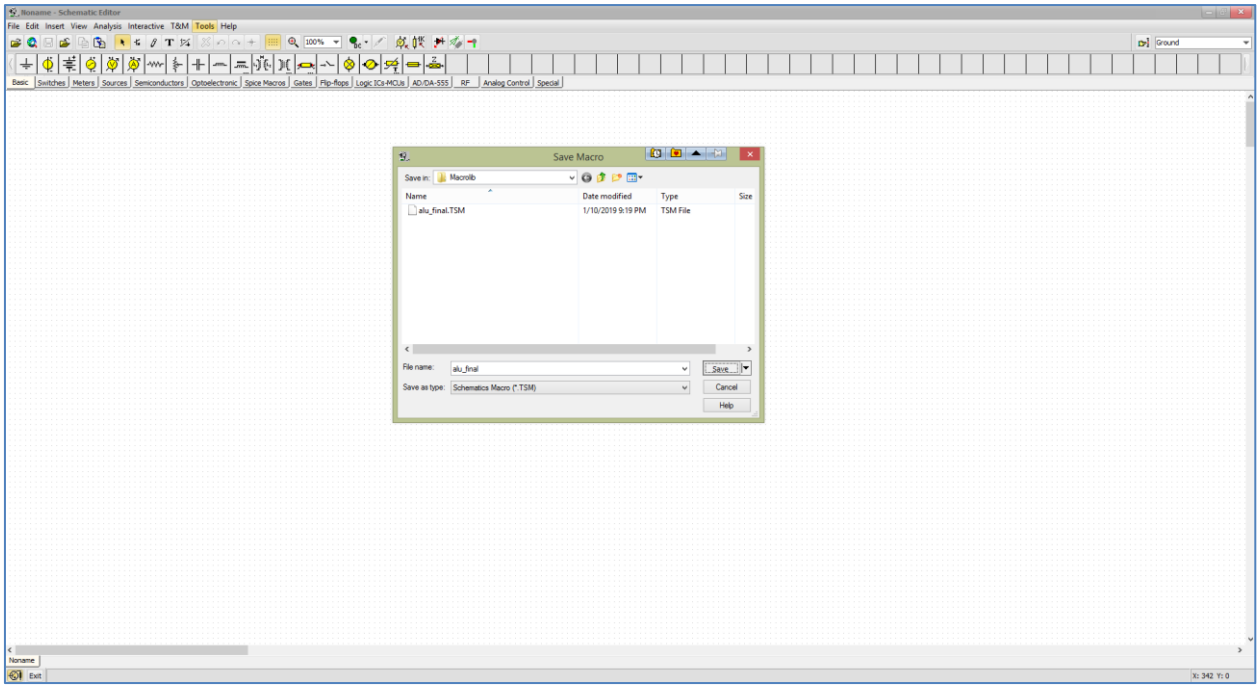
Στη συνέχεια έχοντας ανοίξει τα αρχεία όπως φαίνεται στην εικόνα 4.11 , επιλέγεται για τύπο αρχείου VHDL file(*.VHD) και έπειτα διαλέγεται το αρχείο το οποίο περιέχει το macro (alu_final) της μονάδας πρόσθεσης/αφαίρεσης. Το αρχείο εισάγεται στο σχηματικό επεξεργαστή επιλέγοντας το *Open*



Εικόνα 4.12 Επιλογή θέσης αποθήκευσης τελικού αρχείου

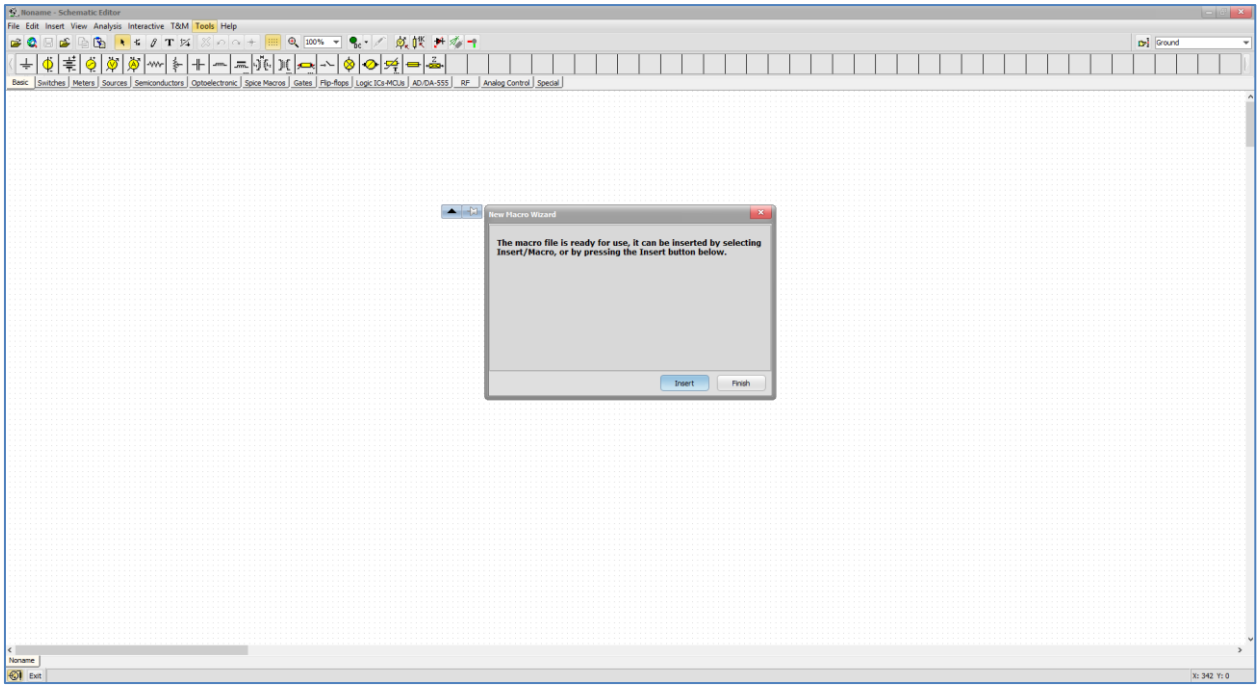
.

Εφόσον επιλέχθηκε το Open, επαναφέρεται στο New Macro Wizard και εφόσον στην επιλογή αρχείου φαίνεται το κατάλληλο αρχείο ,έπειτα επιλέγεται το Next όπως φαίνεται στην εικόνα 4.12.



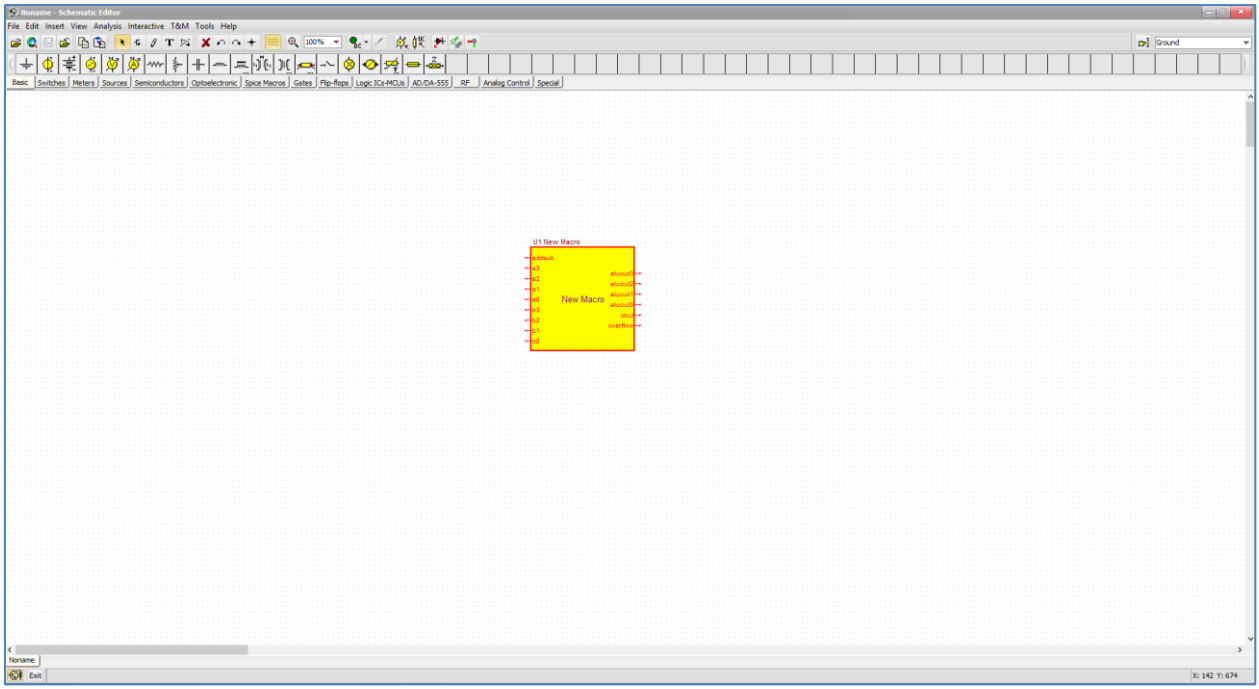
Εικόνα 4.13 Αποθήκευση αρχείου

Στη συνέχεια το αρχείο σε μορφή Schematics Macro(*.TSM) σύμφωνα με την εικόνα 4.13.



Εικόνα 4.14 Εισαγωγή macro box στον εικονικό επεξεργαστή

Έχοντας επιλέξει *Save* εμφανίζεται το παράθυρο στην εικόνα 4.14 και επιλέγουμε *Insert* ώστε να εισάγει το λογισμικό Tina το σχηματικό(schematic) στο τρέχον project .



Εικόνα 4.15 macro schematic μετά την εισαγωγή στον εικονικό επεξεργαστή

Στην εικόνα 4.15 είναι το τελικό αποτέλεσμα έχοντας επιλέξει Insert , και φαίνεται η δομική μονάδα πρόσθεσης/αφαίρεσης.

4.5 Κώδικας VHDL και Macros

Στην παράγραφο αυτή περιγράφουμε αναλυτικά τον κώδικα κάθε μίας από τις δομικές μονάδες που υλοποιούν τον επεξεργαστή που μελετήσαμε στα πλαίσια της παρούσας πτυχιακής εργασίας. Για κάθε μία από τις μονάδες πλέον του κώδικα γίνεται και αναφορά στην μονάδα όπως αυτή υλοποιήθηκε στο λογισμικό Tina.

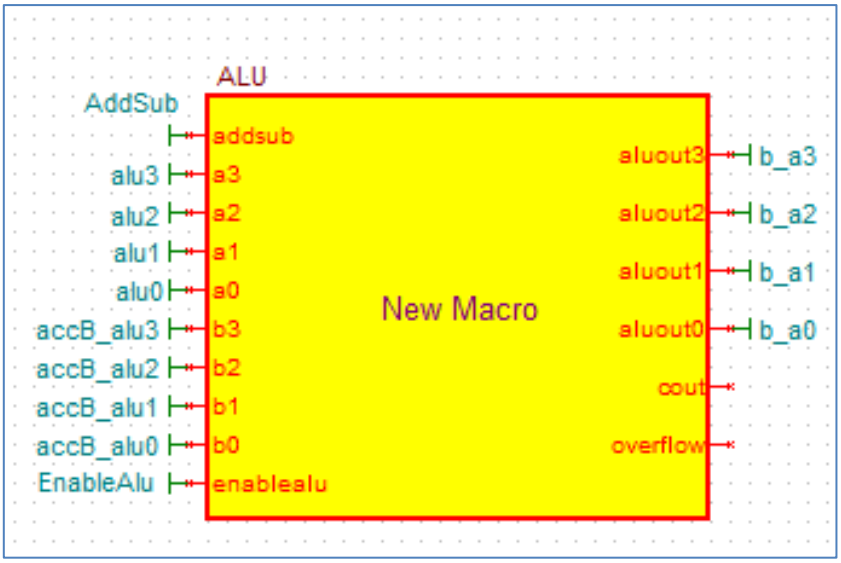
4.5.1 Μονάδα Πρόσθεσης/Αφαίρεσης

Το βασικότερο τμήμα του επεξεργαστή μας είναι η μονάδα πρόσθεσης/αφαίρεσης η οποία τελεί τις πράξεις του ρεπερτορίου εντολών του. Όπως φαίνεται από τον κώδικα που ακολουθεί ο μονάδα αυτή έχει τα εξής σήματα εισόδου: addsub για πρόσθεση ή αφαίρεση και δύο 4-bit είσοδοι A,B, στα σήματα εξόδου διακρίνουμε την 4-bit έξοδο AluOUT καθώς και το κρατούμενο εξόδου Cout και το σήμα υπερχείλισης OverFlow. Στη συνέχεια ακολουθεί ο πηγαίος κώδικας σε γλώσσα VHDL που αναπτύξαμε για τη μονάδα αυτή.

```
library ieee;
use ieee.std_logic_1164.all;
entity VsmALU is
```

```
   port( addsub: in std_logic;
          A3  : in std_logic;
              A2  : in std_logic;
              A1  : in std_logic;
              A0  : in std_logic;
              B3  : in std_logic;
              B2  : in std_logic;
              B1  : in std_logic;
              B0  : in std_logic;
              EnableAlu:in std_logic;
          AluOUT3 : out std_logic;
              AluOUT2 : out std_logic;
              AluOUT1 : out std_logic;
              AluOUT0 : out std_logic;
          Cout, OVERFLOW : out std_logic);
end VsmALU;

architecture bhv of VsmALU is
signal c0,c1,c2,t0,t1,t2,t3: std_logic;
signal tmp3,tmp2,tmp1,tmp0,taluout0,taluout1,taluout2,taluout3:
std_logic;

begin

      tmp0<=addsub xor b0;
      tmp1<=addsub xor b1;
      tmp2<=addsub xor b2;
      tmp3<=addsub xor b3;

      t0<=a0 xor tmp0;
      taluout0<=t0 xor addsub;
      c0<=(addsub and t0)or(tmp0 and a0);


      t1<=a1 xor tmp1;
      taluout1<=t1 xor c0;
      c1<=(c0 and t1)or(tmp1 and a1);

      t2<=a2 xor tmp2;
      taluout2<=t2 xor c1;
      c2<=(c1 and t2)or(tmp2 and a2);

      t3<=a3 xor tmp3;
      taluout3<=t3 xor c2;
      cout<=(c2 and t3)or(tmp3 and a3);
AluOUT3<=taluout3 when (EnableAlu='1') else 'Z';
AluOUT2<=taluout2 when (EnableAlu='1') else 'Z';
AluOUT1<=taluout1 when (EnableAlu='1') else 'Z';
AluOUT0<=taluout0 when (EnableAlu='1') else 'Z';
end bhv;
```

Ακολουθώντας τη διαδικασία που περιγράψαμε στην παράγραφο 4.4 δημιουργούμε από τον παραπάνω κώδικα το αντίστοιχο Macro της μονάδας πρόσθεσης/αφαίρεσης. Στην εικόνα 4.16 φαίνεται η μονάδα αυτή μετά τη δημιουργία της.



Εικόνα 4.16- Η μονάδα της πρόσθεσης/αφαίρεσης

4.5.2 Συσσωρευτής A

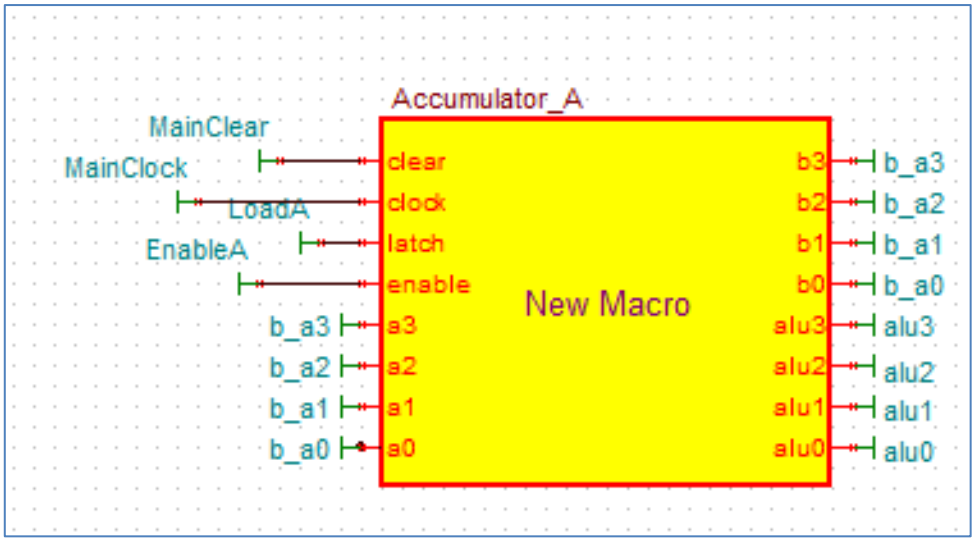
Το τμήμα του επεξεργαστή που αποθηκεύει τα άμεσα αποτελέσματα που υπολογίζονται από τον μικροεπεξεργαστή. Όπως φαίνεται από τον κώδικα που ακολουθεί συσσωρευτής έχει τα εξής σήματα εισόδου: clear και clock για την επαναφορά και το χρονισμό με το παλμό του ρολογιού, latch για το χρόνο μανδάλωσης, Enable ώστε ο συσσωρευτής να αποκτήσει το έλεγχο του διαύλου και μια 4-bit είσοδος A. Στα σήματα εξόδου διακρίνουμε την 4-bit έξοδο B,Alu καθώς και μια 4-bit temp.

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

entity accA is
```

```
port(
      clear,clock,latch,Enable,A3,A2,A1,A0: in std_logic;
      B3,B2,B1,B0,Alu3,Alu2,Alu1,Alu0: out std_logic
);
end accA;

architecture bhv of accA is
signal temp: std_logic_vector(3 downto 0);
begin
process (clock,clear)
begin
      if(clear='0') then
            temp<="0000";

      elsif rising_edge(clock) then
            if(latch='1') then
                  temp(3)<=A3;
            temp(2)<=A2;
            temp(1)<=A1;
            temp(0)<=A0;
            end if;
      end if;
end process;

Alu3 <= temp(3);
Alu2 <= temp(2);
Alu1 <= temp(1);
Alu0 <= temp(0);
B3<= temp(3) when (Enable='1') else 'Z';
B2<= temp(2) when (Enable='1') else 'Z';
B1<= temp(1) when (Enable='1') else 'Z';
B0<= temp(0) when (Enable='1') else 'Z';
end bhv;
```

Κώδικας 3: κώδικας για το συσσωρευτή Α

Με την ίδια διαδικασία που δημιουργήσαμε τη μονάδα πρόσθεσης/αφαίρεσης δημιουργούμε και το Macro του συσσωρευτή Α. Στην εικόνα που ακολουθεί φαίνεται η μονάδα του συσσωρευτή.



Εικόνα 4.17- Συσσωρευτής Α

4.5.3 Συσσωρευτής Β

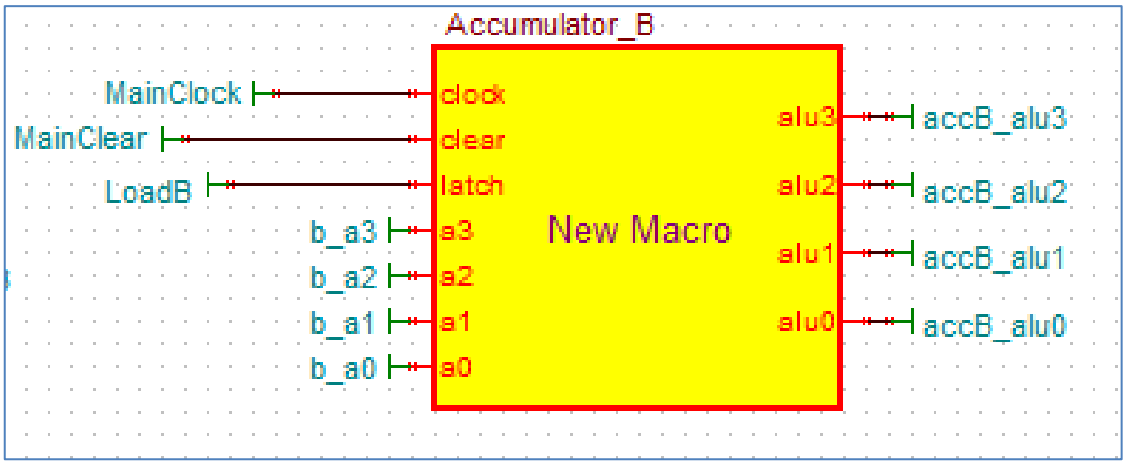
Το τμήμα του επεξεργαστή που χρησιμοποιείται κατα κόρον για να παράγει τον προστιθέμενο ή αφαιρούμενο από τον συσσωρεύτη Β αριθμό με σκοπό να εκτελεστεί η πρόσθεση ή η αφαίρεση. Όπως φαίνεται από τον κώδικα που ακολουθεί συσσωρευτής έχει τα εξής σήματα εισόδου: clear , clock,latch και Enable και μια 4-bit είσοδος Α. Στα σήματα εξόδου διακρίνουμε την 4-bit έξοδο Alu .

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

entity accA is

port(
      clear,clock,latch,Enable,A3,A2,A1,A0: in std_logic;
      B3,B2,B1,B0,Alu3,Alu2,Alu1,Alu0: out std_logic
);
end accA;

architecture bhv of accA is
signal temp: std_logic_vector(3 downto 0);
begin
process (clock,clear)
begin
      if(clear='0') then
            temp<="0000";

      elsif rising_edge(clock) then
            if(latch='1') then
                  temp(3)<=A3;
            temp(2)<=A2;
            temp(1)<=A1;
```

```
            temp(0)<=A0;
            end if;
      end if;
end process;

Alu3 <= temp(3);
Alu2 <= temp(2);
Alu1 <= temp(1);
Alu0 <= temp(0);
B3<= temp(3) when (Enable='1') else 'Z';
B2<= temp(2) when (Enable='1') else 'Z';
B1<= temp(1) when (Enable='1') else 'Z';
B0<= temp(0) when (Enable='1') else 'Z';
end bhv;
```

Κώδικας 3: κώδικας για το Συσσωρευτή B

Από τον παραπάνω κώδικα δημιουργείται το αντίστοιχο Macro του συσσωρευτή B, στην εικόνα 4.18 φαίνεται η μονάδα αυτή μετά τη δημιουργία της.



Εικόνα 4.18- Συσσωρευτής B

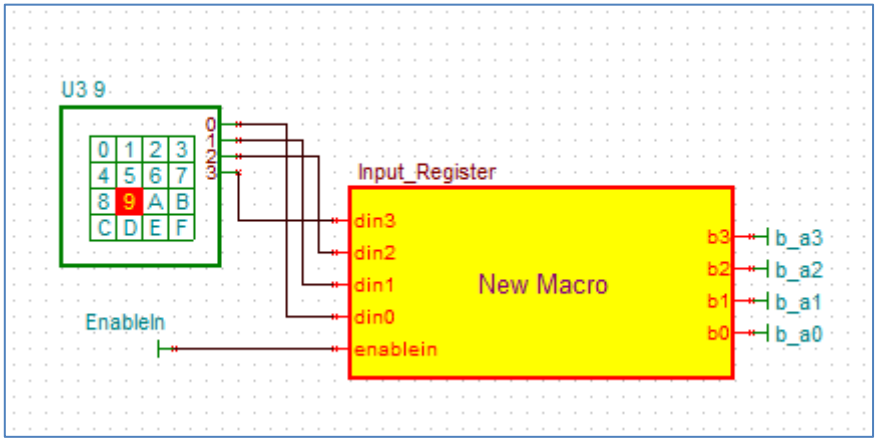
4.5.4 Καταχωρητής Εισόδου

Στο τμήμα του καταχωρητή εισόδου μεταφέρεται στον μικροεπεξεργαστή εξωτερικά δεδομένα. Όπως φαίνεται από τον κώδικα που ακολουθεί ο μονάδα αυτή έχει τα εξής σήματα εισόδου: EnableIn και 4-bit Din, στα σήματα εξόδου διακρίνουμε την 4-bit έξοδο B.

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

entity InputReg is

port(
      Din3 : in std_logic;
      Din2 : in std_logic;
      Din1 : in std_logic;
      Din0 : in std_logic;
      B3 : out std_logic;
      B2 : out std_logic;
      B1 : out std_logic;
      B0 : out std_logic;
      EnableIn: in std_logic
);
end InputReg;

architecture bhv of InputReg is
begin
B3 <= Din3 when (EnableIn='1') else 'Z';
B2 <= Din2 when (EnableIn='1') else 'Z';
B1 <= Din1 when (EnableIn='1') else 'Z';
B0 <= Din0 when (EnableIn='1') else 'Z';
end bhv;
```

Κώδικας 4: κώδικας για το Καταχωρητή Εισόδου

Με την υλοποίηση του κώδικα πραγματοποιείται το Macro του καταχωρητή εισόδου. Στην εικόνα 4,19 φαίνεται η μονάδα αυτή μετά τη δημιουργία της.



Εικόνα 4.19- Καταχωρητής Εισόδου

.

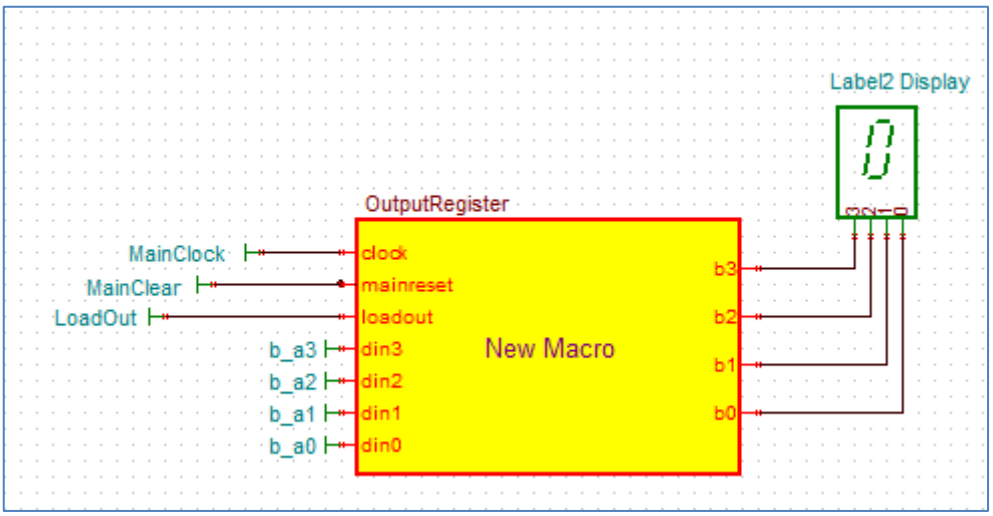
4.5.5 Καταχωρητής Εξόδου

Στο καταχωρητή εξόδου μεταφέρονται τα περιεχόμενα του εσωτερικού διαύλου στο εξωτερικό περιβάλλον. Όπως φαίνεται από τον κώδικα που ακολουθεί ο μονάδα αυτή έχει τα εξής σήματα εισόδου: clock, MainReset, LoadOut καθώς και την 4-bit είσοδο Din .Στα σήματα εξόδου διακρίνουμε την 4-bit έξοδο B

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

entity outputReg is

port(
      clock,MainReset,LoadOut: in std_logic;
      Din3 : in std_logic;
      Din2 : in std_logic;
      Din1 : in std_logic;
      Din0 : in std_logic;
      B3: out std_logic;
      B2: out std_logic;
      B1: out std_logic;
      B0: out std_logic
);
end outputReg;

architecture bhv of outputReg is
signal temp: std_logic_vector(3 downto 0);
begin
process (clock,MainReset,LoadOut)
begin
      if(MainReset='0') then
            temp<="0000";
      else
            if rising_edge(clock) then
                  if(LoadOut='1') then
                        temp(3)<=Din3;
                        temp(2)<=Din2;
                        temp(1)<=Din1;
                        temp(0)<=Din0;
                  end if;
            end if;
      end if;

end process;

B3 <= temp(3);
B2 <= temp(2);
B1 <= temp(1);
B0 <= temp(0);

end bhv;
```

Κώδικας 5: κώδικας για το Καταχωρητή Εξόδου

Σύμφωνα με το κώδικα εμφανίζεται το αντίστοιχο Macro του καταχωρητή εξόδου. Στην εικόνα 4.20 είναι η μονάδα αυτή.



Εικόνα 4.20: Καταχωρητής Εξόδου

4.5.6 Μετρητής Προγράμματος

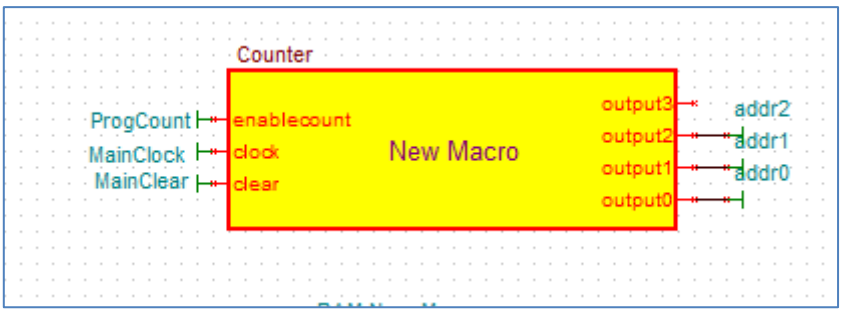
Ο μετρητής προγράμματος μετράει από το 0000 ως το 1111, κάθε χρονική στιγμή περιέχει την τιμή της ενεργούς διεύθυνσης. Όπως φαίνεται από τον κώδικα που ακολουθεί ο μονάδα αυτή έχει τα εξής σήματα εισόδου: EnableCount ,Clock και Clear.Στα σήματα εξόδου διακρίνουμε την 4-bit έξοδο Outpout.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity programCounter is
   port( EnableCount: in std_logic;
```

```
        Clock: in std_logic;
        Clear: in std_logic;
        Output3: out std_logic;
        Output2: out std_logic;
        Output1: out std_logic;
        Output0: out std_logic);
end programCounter;

architecture Behavioral of programCounter is
   signal temp: std_logic_vector(3 downto 0);

begin   process(Clock,Clear)
   begin
      if Clear='0' then
         temp <= "0000";
      elsif(falling_edge(Clock)) then
         if EnableCount='1' then
               temp <= temp + 1;
         end if;
      end if;
   end process;
   Output3 <= temp(3);
   Output2 <= temp(2);
   Output1 <= temp(1);
   Output0 <= temp(0);
end Behavioral;
```

Κώδικας 6: κώδικας για το Μετρητή προγράμματος

Στην εικόνα 4.21 φαίνεται η μονάδα αυτή μετά τη δημιουργία της.



Εικόνα 4.21: Μετρητής Προγράμματος

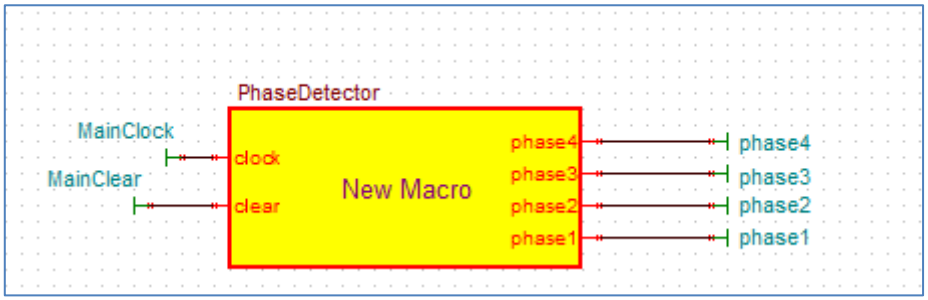
4.5.7 Μετρητής φάσεων

Ο Μετρητής φάσεων πρέπει να παράγει τα σήματα τεσσάρων φάσεων Phase0 έως Phase3, στο αρνητικό άκρο του ρολογιού με σκοπό το πλήρως προγραμματιζόμενο μικροεπεξεργαστή. Όπως φαίνεται από τον κώδικα που ακολουθεί ο μονάδα αυτή έχει τα εξής σήματα εισόδου: Clock και Clear.Στα σήματα εξόδου διακρίνουμε την 4-bit έξοδο phase.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity PhaseCounter2 is
    Port ( CLOCK : in  STD_LOGIC;
           Clear : in  STD_LOGIC;
           phase4,phase3,phase2,phase1 : out  STD_LOGIC);
end PhaseCounter2;

architecture Behavioral of PhaseCounter2 is
signal q_tmp: std_logic_vector(3 downto 0):= "0000";
begin
process(CLOCK,Clear)
begin
if Clear = '0' then
    q_tmp <= "0001";
elsif falling_edge(CLOCK) then
    q_tmp(1) <= q_tmp(0);
    q_tmp(2) <= q_tmp(1);
    q_tmp(3) <= q_tmp(2);
    q_tmp(0) <= q_tmp(3);

end if;
end process;
phase4 <= q_tmp(3);
phase3 <= q_tmp(2);
phase2 <= q_tmp(1);
phase1 <= q_tmp(0);
end Behavioral;
```

Κώδικας 7: κώδικας για το Μετρητή φάσεων

Στην εικόνα 4.22 φαίνεται η μονάδα αυτή μετά τη δημιουργία της.



Εικόνα 4.22: Μετρητής φάσεων

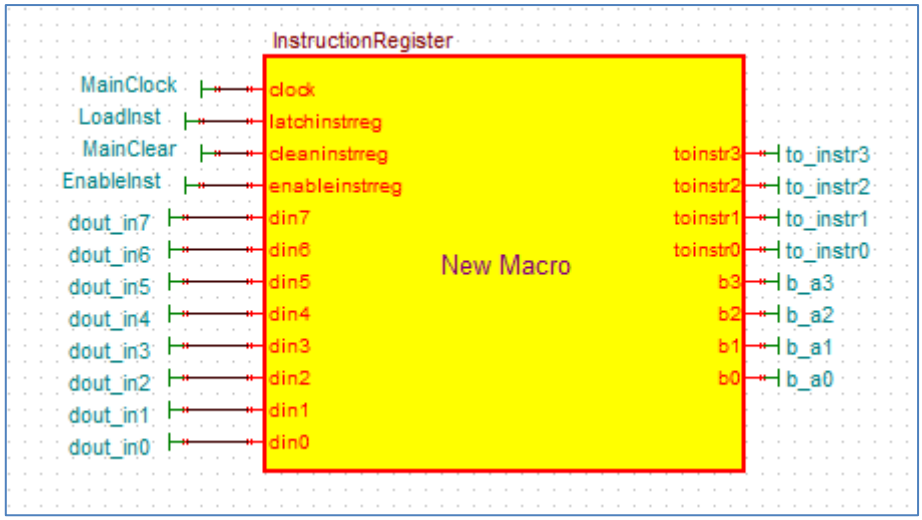
4.5.8 Καταχωρητής εντολών

Ο καταχωρητής εντολών έχει ως κύριο σκοπό να αποθηκεύει τα περιεχόμενα της μνήμης προγράμματος. Όπως φαίνεται από τον κώδικα που ακολουθεί ο μονάδα αυτή έχει τα εξής σήματα εισόδου: Clock, LatchInstrReg, CleanInstrReg, EnableInstrReg καθώς και ένα 8-bit είσοδοι Din, στα σήματα εξόδου διακρίνουμε την 4-bit έξοδο ToInstr και την 4bit έξοδο B.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity PhaseCounter2 is
    Port ( CLOCK : in  STD_LOGIC;
           Clear : in  STD_LOGIC;
           phase4,phase3,phase2,phase1 : out  STD_LOGIC);
end PhaseCounter2;

architecture Behavioral of PhaseCounter2 is
signal q_tmp: std_logic_vector(3 downto 0):= "0000";
begin
process(CLOCK,Clear)
begin
if Clear = '0' then
    q_tmp <= "0001";
elsif falling_edge(CLOCK) then
    q_tmp(1) <= q_tmp(0);
    q_tmp(2) <= q_tmp(1);
    q_tmp(3) <= q_tmp(2);
    q_tmp(0) <= q_tmp(3);

end if;
end process;
phase4 <= q_tmp(3);
phase3 <= q_tmp(2);
phase2 <= q_tmp(1);
phase1 <= q_tmp(0);
end Behavioral;
```

Κώδικας 8: κώδικας για το Μετρητή φάσεων

Στην εικόνα 4.23 φαίνεται η μονάδα αυτή μετά τη δημιουργία της.



Εικόνα 4.23: Μετρητής φάσεων

4.5.9 Μνήμη RAM

Η μνήμη RAM αποθηκεύει τις εντολές προγράμματος. Έχει 8 θέσεις μνήμης μήκους λέξης 1byte. Τα σήματα ελέγχου όπως φαίνεται από τον κώδικα που ακολουθεί είναι τα εξής: Clock, Wr, Rd, addr(2:0), Din(7:0), Dout7:0).

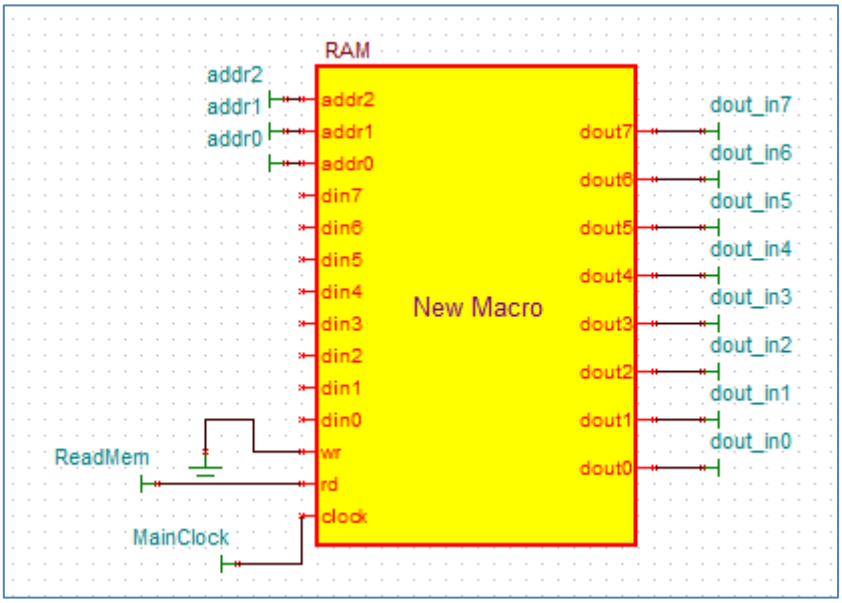
```
library ieee;
 use ieee.std_logic_1164.all;
 use ieee.numeric_std.all;
 entity RAM is
 port(
        addr2: in std_logic;
      addr1: in std_logic;
        addr0: in std_logic;

        Din7: in std_logic;     Din6: in std_logic;
        Din5: in std_logic;     Din4: in std_logic;
      Din3: in std_logic;       Din2: in std_logic;
        Din1: in std_logic;     Din0: in std_logic;

        Dout7: out std_logic;   Dout6: out std_logic;
        Dout5: out std_logic;   Dout4: out std_logic;
      Dout3: out std_logic;     Dout2: out std_logic;
        Dout1: out std_logic;   Dout0: out std_logic;
        Wr, Rd,Clock: in std_logic);
 end entity RAM;
architecture behavior of RAM is
      signal
PROG0,PROG1,PROG2,PROG3,PROG4,PROG5,PROG6,PROG7:STD_LOGIC_VECTOR(7
DOWNTO 0);
      signal
tDout0,tDout1,tDout2,tDout3,tDout4,tDout5,tDout6,tDout7:STD_LOGIC;
      signal tempd:std_logic_vector(7 downto 0);
    begin
            PROG0<="00000000"; --NOP
```

```
              PROG1<="01010011"; --LDA 3
            PROG2<="00100001"; --SUB 1
            PROG3<="00100001"; --SUB 1
            PROG4<="00010011"; --ADD 3
            PROG5<="00100010"; --SUB 2
            PROG6<="00110000"; --SUB 1
            PROG7<="00000000"; --NOP

            tempd(7)<=Din7;
            tempd(6)<=Din6;
            tempd(5)<=Din5;
            tempd(4)<=Din4;
            tempd(3)<=Din3;
            tempd(2)<=Din2;
            tempd(1)<=Din1;
            tempd(0)<=Din0;

            process (Clock,Rd)
    begin
            if rising_edge(Clock) then
                  if (Rd= '1') then
                        if(addr2='0' and addr1='0' and addr0='0') then
                              tDout7<=PROG0(7);
      tDout6<=PROG0(6);
                              tDout5<=PROG0(5);
      tDout4<=PROG0(4);
                              tDout3<=PROG0(3);
      tDout2<=PROG0(2);
                              tDout1<=PROG0(1);
      tDout0<=PROG0(0);
                        elsif(addr2='0' and addr1='0' and addr0='1')
then
                              tDout7<=PROG1(7);
      tDout6<=PROG1(6);
                              tDout5<=PROG1(5);
      tDout4<=PROG1(4);
                              tDout3<=PROG1(3);
      tDout2<=PROG1(2);
                              tDout1<=PROG1(1);
      tDout0<=PROG1(0);
                        elsif(addr2='0' and addr1='1' and addr0='0')
then
                              tDout7<=PROG2(7);
      tDout6<=PROG2(6);
                              tDout5<=PROG2(5);
      tDout4<=PROG2(4);
                              tDout3<=PROG2(3);
      tDout2<=PROG2(2);
                              tDout1<=PROG2(1);
      tDout0<=PROG2(0);
                        elsif(addr2='0' and addr1='1' and addr0='1')
then
                              tDout7<=PROG3(7);
      tDout6<=PROG3(6);
                              tDout5<=PROG3(5);
      tDout4<=PROG3(4);
                              tDout3<=PROG3(3);
      tDout2<=PROG3(2);
                              tDout1<=PROG3(1);
      tDout0<=PROG3(0);
```

```
                        elsif(addr2='1' and addr1='0' and addr0='0')
then
                                tDout7<=PROG4(7);
        tDout6<=PROG4(6);
                                tDout5<=PROG4(5);
        tDout4<=PROG4(4);
                                tDout3<=PROG4(3);
        tDout2<=PROG4(2);
                                tDout1<=PROG4(1);
        tDout0<=PROG4(0);
                        elsif(addr2='1' and addr1='0' and addr0='1')
then
                                tDout7<=PROG5(7);
        tDout6<=PROG5(6);
                                tDout5<=PROG5(5);
        tDout4<=PROG5(4);
                                tDout3<=PROG5(3);
        tDout2<=PROG5(2);
                                tDout1<=PROG5(1);
        tDout0<=PROG5(0);
                        elsif(addr2='1' and addr1='1' and addr0='0')
then
                                tDout7<=PROG6(7);
        tDout6<=PROG6(6);
                                tDout5<=PROG6(5);
        tDout4<=PROG6(4);
                                tDout3<=PROG6(3);
        tDout2<=PROG6(2);
                                tDout1<=PROG6(1);
        tDout0<=PROG6(0);
                        elsif(addr2='1' and addr1='1' and addr0='1')
then
                                tDout7<=PROG7(7);
        tDout6<=PROG7(6);
                                tDout5<=PROG7(5);
        tDout4<=PROG7(4);
                                tDout3<=PROG7(3);
        tDout2<=PROG7(2);
                                tDout1<=PROG7(1);
        tDout0<=PROG7(0);
                        end if;
                end if;
                end if;
      end process;
      dout0<=tDout0;
      dout1<=tDout1;
      dout2<=tDout2;
      dout3<=tDout3;
      dout4<=tDout4;
      dout5<=tDout5;
      dout6<=tDout6;
      dout7<=tDout7;
    end behavior;
```

Κώδικας 9: κώδικας για τη μνήμη RAM

Ακολουθώντας τη διαδικασία που περιγράψαμε δημιουργούμε από τον παραπάνω κώδικα το αντίστοιχο Macro της μνήμης RAM. Στην εικόνα 4.24 φαίνεται η μονάδα αυτή μετά τη δημιουργία της.



Εικόνα 4.24 Μνήμη RAM

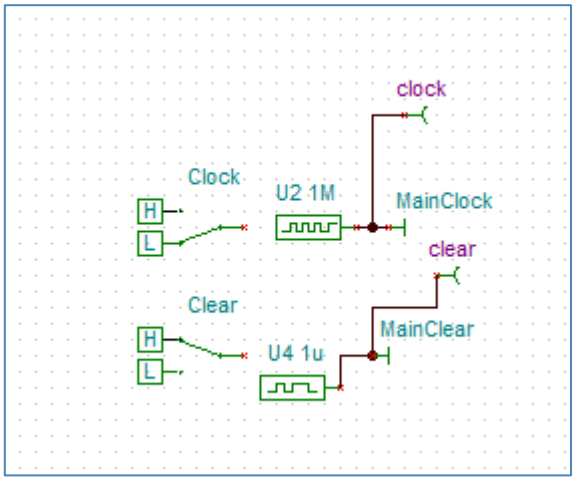
ΚΕΦΑΛΑΙΟ 5 - ΣΕΝΑΡΙΑ ΧΡΗΣΗΣ - ΑΠΟΤΕΛΕΣΜΑΤΑ

Στο τελευταίο κεφάλαιο της πτυχιακής μας εργασίας ασχολούμαστε με την εξαγωγή των κυματομορφών και ανάλυση των αποτελεσμάτων που προκύπτουν από αυτές.

5.1 Δημιουργία των κυματομορφών

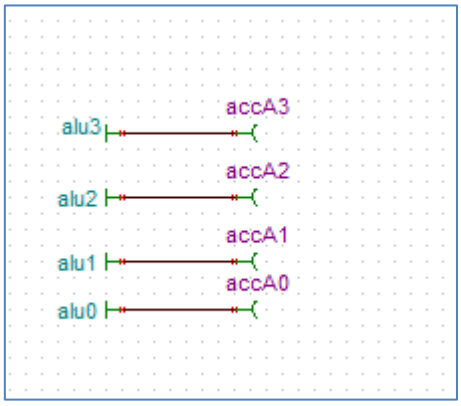
Στην ενότητα αυτή της παρούσας πτυχιακής περιγράφουμε τη διαδικασία που ακολουθήσαμε για την εξαγωγή των κυματομορφών βασικών σημάτων του επεξεργαστή επαληθεύοντας με αυτό τον τρόπο την ορθή λειτουργία του. Η διαδικασία την οποία ακολουθήσαμε συνολικά περιγράφεται στη συνέχεια :

- Τοποθετούμε πηγής παλμών στο χρονισμό(clock) και στην επαναφορά(reset)



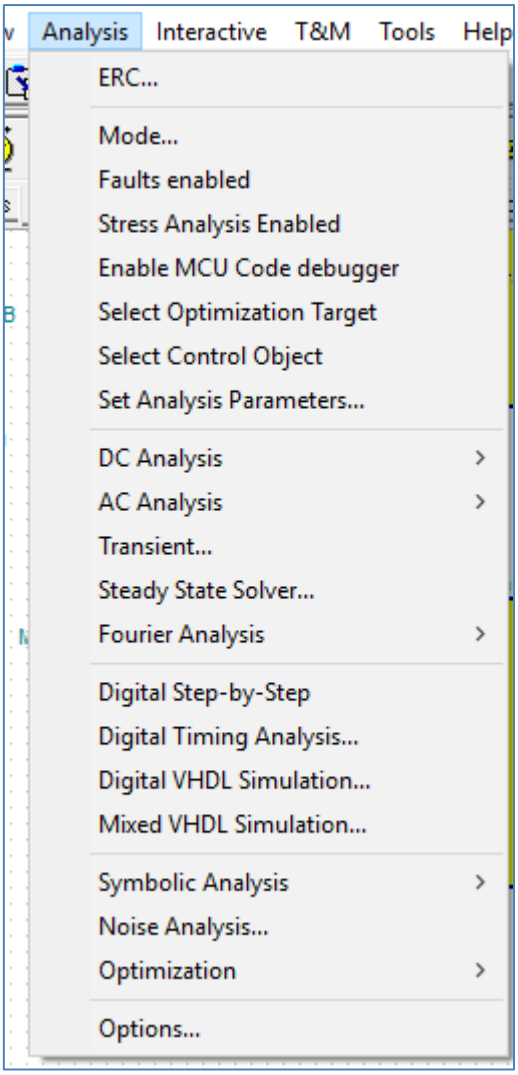
Εικόνα 5.1.1

- Τοποθετούμε ακροδέκτες εξόδου στις εξόδους της αριθμητικής και λογικής μονάδας



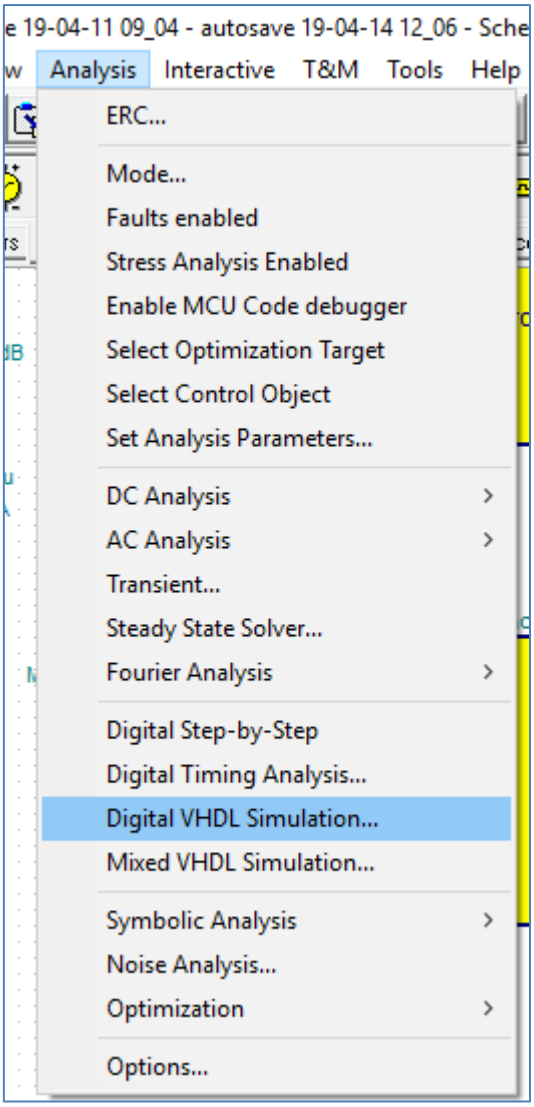
Εικόνα 5.1.2

- Από τη γραμμή μενού επιλέγουμε την καρτέλα “Analysis”



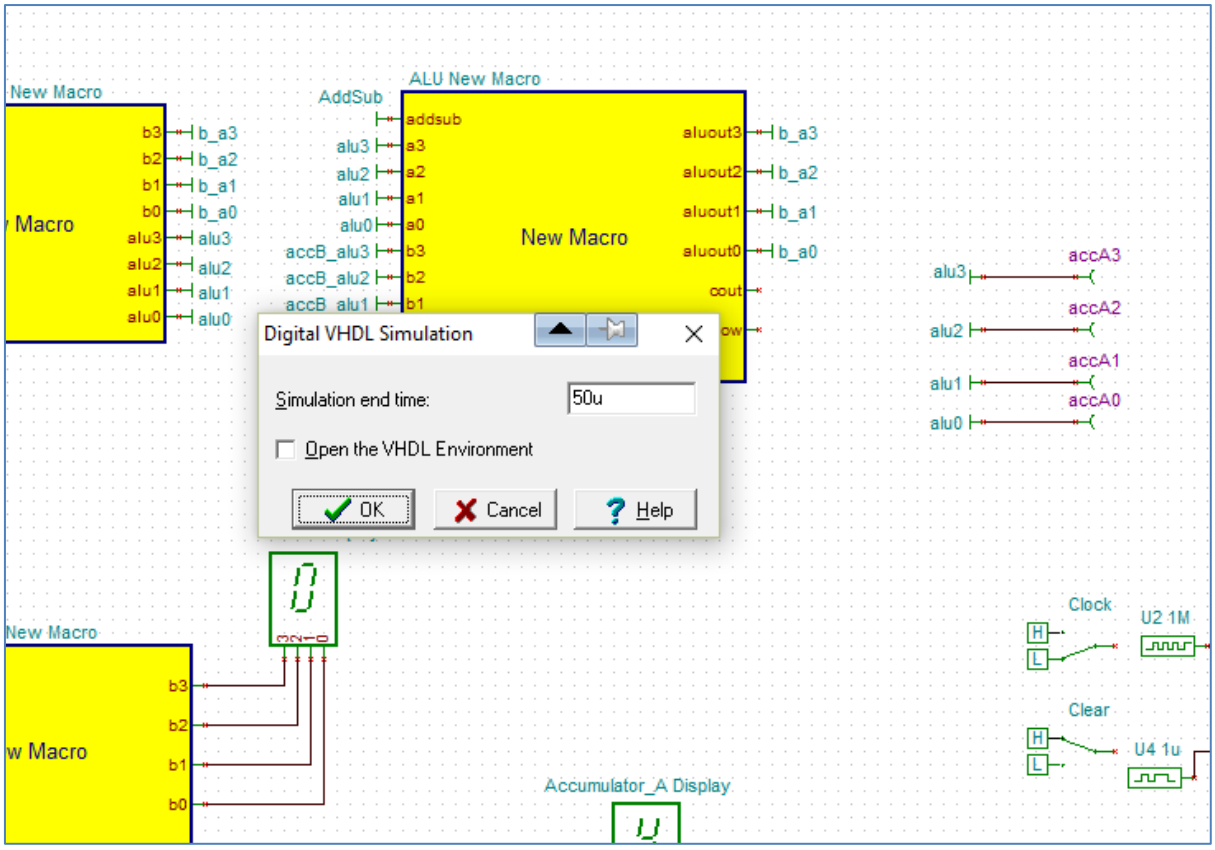
Εικόνα 5.1.3

- Στη συνέχεια επιλέγουμε “Digital VHDL Simulation”



Εικόνα 5.1.4

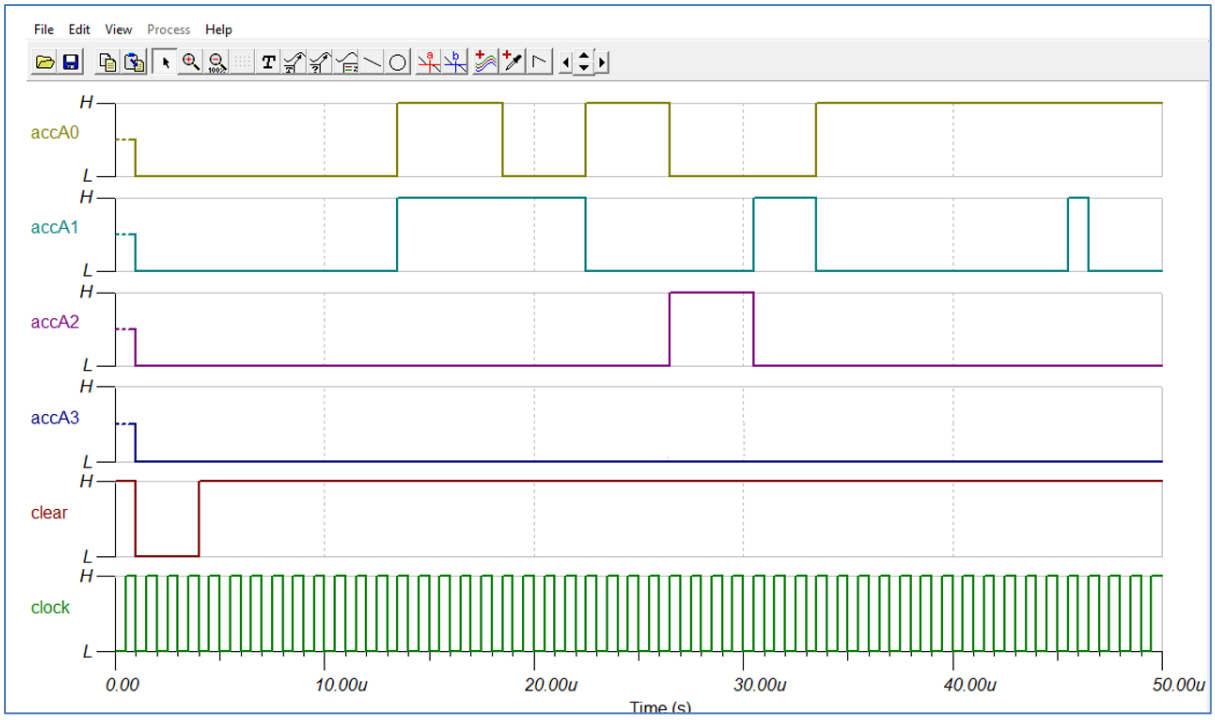
- Το παράθυρο το οποίο εμφανίζεται μετά την επιλογή του “Digital VHDL Simulation” είναι :



Εικόνα 5.1.5

Στο οποίο καθορίζουμε τη χρονική περίοδο των κυματομορφών που τα παρατηρήσουμε .

- Μετά την ανάλυση σε επίπεδο VHDL προκύπτουν οι κυματομορφές όπως αυτές αποτυπώνονται στην εικόνα 4.24



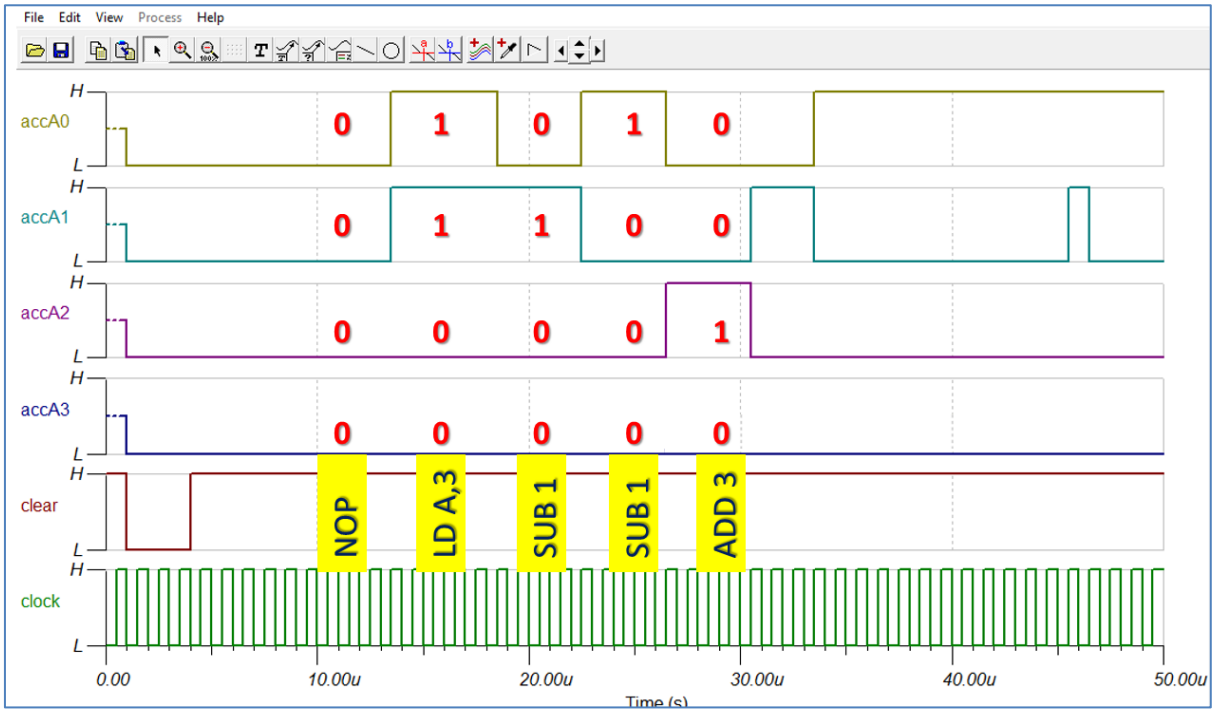
Εικόνα 5.1.7

5.2 Ανάλυση τελικών αποτελεσμάτων

Στην παράγραφο αυτή μελετάμε και αναλύουμε τα αποτελέσματα όπως αυτά προέκυψαν από τη προσομοίωση του απλού επεξεργαστή. Στο σενάριο που ακολουθήσαμε τον έλεγχο ορθότητας της λειτουργίας του επεξεργαστή μας δημιουργήσαμε ένα απλό πρόγραμμα βασικών αριθμητικών πράξεων και μεταφοράς δεδομένων. Συγκεκριμένα το πρόγραμμα που αποθηκεύτηκε στην μνήμη του επεξεργαστή έχει τις ακόλουθες εντολές:

```
-    NOP ;καμία λειτουργία
-    LD A,3 ;φόρτωση στον συσσωρευτή της τιμής 3
-    SUB 1 ;μείωση του περιεχομένου του συσσωρευτή κατά 1
-    SUB 1 ;μείωση του περιεχομένου του συσσωρευτή κατά 1
```

```
-    ADD 3 ;αύξηση του περιεχομένου του συσσωρευτή κατά 3
```



Εικόνα 5.1.8 Αποτελέσματα κατά την εκτέλεση πράξεων

Στις κυματομορφές εξόδου που παρουσιάζονται στην εικόνα 4.24 διαπιστώνουμε τη σωστή εκτέλεση του προγράμματος μας, με την πρώτη τιμή που φορτώνεται στο συσσωρευτή A (μετά την εκτέλεση της NOP να είναι η 0011=3, στη συνέχεια αυτή μειώνεται κατά 1 (τιμή 0010=2) επίσης μειώνεται κατά 1 (0001=1) και στη συνέχεια γίνεται 0100=4

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Etienne SICARD www.microwind.org, "A Very Simple Microprocessor"

[2] A.P. Malvino, J.A. Brown "Digital computer electronics", Third Edition, Glenco-Macmillan, ISBN 0-02-800594-5,1992, USA

[3] M. Morris Mano, Michael D. Ciletti, "Digital Design",

Prentice Hall of India Pvt. Ltd., 2008.

[4] Brian Holdsworth, Clive Woods, "Digital Logic Design",

Elsevier India Pvt. Ltd., 2005.

[5] Samir Palnitkar, "Verilog HDL, A Guide to Digital Design and Synthesis",

Prentice Hall of India Pvt. Ltd., 2005.

[6] Douglas L. Perry "VHDL SEOND EDITION", McGraw-Hill International Editions

[7] www.tina.com

[8] "TINA Users Manual", DesignSoftware.

[9] J. F. Wakerly, Digital Design Principles & Practices, 3rd Ed., Prentice Hall, Upper Saddle River, New Jersey, 2001.

[10] S. Yalamanchili, Introductory VHDL From Simulation to Synthesis, Prentice Hall,

Upper Saddle River, New Jersey, 2001.

[11] The VHDL Reference: A Practical Guide to Computer-Aided Integrated Circuit Design (Including VHDL-AMS) with Other, ISBN:0471899720