ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΘΕΜΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Σχεδίαση ελεγκτών για τη διασύνδεση μετατροπέων αναλογικού σήματος σε ψηφιακό (ADC) και ψηφιακού σήματος σε αναλογικό (DAC), σε διάταξη FPGA.

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΣΠΟΥΔΑΣΤΩΝ:
ΚΑΥΚΑΛΟΥΔΗ ΕΥΘΥΜΙΑ
ΜΟΥΣΚΕΦΤΑΡΑ ΔΗΜΗΤΡΑ

Επιβλέπων:
Ι. Καλόμοιρος, Αναπληρωτής καθηγητής

ΣΕΡΡΕΣ 2016

Υπεύθυνη Δήλωση:

Βεβαιώνουμε ότι είμαστε συγγραφείς αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης, έχουμε αναφέρει τις πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Εν κατακλείδι, βεβαιώνουμε ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμάς προσωπικά, ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής Τ.Ε. Κεντρικής Μακεδονίας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 4
Εισαγωγή στο FPGA

ΚΕΦΑΛΑΙΟ 5
Σχεδίαση ελεγκτών για την διασύνδεση μετατροπέων A/D και D/A σε διάταξη FPGA

ΣΧΗΜΑΤΑ

ΚΕΦΑΛΑΙΟ 1
Εισαγωγή στα Συστήματα A/D και D/A

ΚΕΦΑΛΑΙΟ 2
Υλοποίηση Συστημάτων σε διάτρητη πλακέτα

ΚΕΦΑΛΑΙΟ 3
Εισαγωγή στη γλώσσα VHDL και στο πρόγραμμα Quartus II

ΚΕΦΑΛΑΙΟ 4
Εισαγωγή στο FPGA

ΚΕΦΑΛΑΙΟ 5
Σχεδίαση ελεγκτών για την διασύνδεση μετατροπέων A/D και D/A σε διάταξη FPGA

ΠΙΝΑΚΕΣ

ΚΕΦΑΛΑΙΟ 1
Εισαγωγή στα Συστήματα A/D και D/A

ΚΕΦΑΛΑΙΟ 2
Υλοποίηση Συστημάτων σε διάτρητη πλακέτα

ΚΕΦΑΛΑΙΟ 3
Εισαγωγή στη γλώσσα VHDL και στο πρόγραμμα Quartus II

ΚΕΦΑΛΑΙΟ 4
Εισαγωγή στο FPGA

ΚΕΦΑΛΑΙΟ 5
Σχεδίαση ελεγκτών για την διασύνδεση μετατροπέων A/D και D/A σε διάταξη FPGA

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή στα Συστήματα A/D και D/A

1.1Αναλογικά και Ψηφιακά Σήματα

Για να κατανοήσει κανείς καλύτερα τη μετατροπή από αναλογικό σήμα σε ψηφιακό και από ψηφιακό σήμα σε αναλογικό είναι σημαντικό να δοθούν οι ορισμοί τους καθώς και οι λόγοι που μας ωθούν για τη μετατροπή των σημάτων αυτών.

Αναλογικό σήμα (Analog signal) είναι η πληροφορία με συνεχής ροή όπου δέχεται συνεχείς τιμές σ' ένα μέσω μετάδοσης σε συνάρτηση με το χρόνο. Για παράδειγμα ο ήχος που ταξιδεύει μέσω του αέρα είναι ένα αναλογικό σήμα που μεταφέρει τη διακύμανση της πίεσης που προκαλεί στον αέρα η ταλάντωση μιας ηχητικής πηγής.

Ψηφιακό σήμα (Digital signal) μπορεί να είναι είτε ένα σήμα διακριτού χρόνου και σ αυτή τη περίπτωση εννοούμε ένα επεξεργασμένο αναλογικό σήμα για την μετατροπή του σε ψηφιακό σήμα, είτε μια κυματομορφή συνεχούς χρόνου σε ψηφιακά συστήματα τα οποία μπορούν να αναπαρασταθούν σε μια σειρά από ψηφία (bits).

Οι λόγοι που μας ωθούν στη μετατροπή σημάτων παρατίθενται στη συνέχεια. Αρχικά πρέπει να πούμε πως υπάρχουν περιορισμοί στη μετάδοση αναλογικών σημάτων, λόγο της παρουσίασης θορύβου ο οποίος μπορεί να δημιουργηθεί είτε από φυσικά αίτια είτε από ανθρώπινα αίτια με αποτέλεσμα το σήμα μέχρι να φτάσει στο προορισμό του να είναι εξασθενημένο. Ένα ψηφιακό σήμα είναι λιγότερο ευαίσθητο στο θόρυβο, έτσι έχουμε μεγαλύτερη ακρίβεια αποτελεσμάτων. Επιπλέον οι ηλεκτρονικοί υπολογιστές χειρίζονται τα δεδομένα με δυαδικά ψηφία όπου υπάρχουν δυο δυνατές τιμές το 0 και το 1. Για να γίνουν αντιληπτά από τον άνθρωπο τα δεδομένα ήχου και εικόνας αλλά και να μπορέσει να γίνει επεξεργασία τους θα πρέπει να μετατραπούν από ψηφιακά σε αναλογικά σήματα.

1.2 Μετατροπή Αναλογικού Σήματος σε Ψηφιακό A/D

Η *μετατροπή αναλογικού σήματος σε ψηφιακό (Analog to Digital converter ή A/D converter ή ADC)* είναι μια σημαντική διεργασία όπου γίνεται σ' ένα ψηφιακό σύστημα και συνδέεται με τον πραγματικό κόσμο. Τα φυσικά μεγέθη όπως η πίεση, η θερμοκρασία, η τάση, η απόσταση μεταβάλλονται αναλογικά.

Ένας A/D μετατροπέας έχει διάφορες πηγές σφαλμάτων. Τη κβαντοποίηση σφάλματος και τη μη γραμμικότητα. Όλοι οι A/D μετατροπείς έχουν σφάλματα μη-γραμμικότητας που προκαλείται από φυσικές ατέλειες τους, με αποτέλεσμα στη παραγωγή τους να αποκλίνουν μια γραμμική συνάρτηση ή κάποια άλλη λειτουργία στις εισόδους τους. Αυτά τα λάθη μπορούν μερικές φορές να μετριαστούν με τη βαθμονόμηση, ή να προληφθούν με τη δοκιμή.

Ένας μετατροπέας A/D έχει μια απλή ή διαφορική είσοδο όπου εκεί υπάρχει η αναλογική τάση που θέλουμε να μετατρέψουμε σε ψηφιακή και στην έξοδο έχουμε έναν αριθμό από ακροδέκτες που θα οδηγηθούν στις κατάλληλες λογικές καταστάσεις 0 ή 1 μετά τη μετατροπή. Σε πρακτικά κυκλώματα ο ελάχιστος αριθμός δυαδικών ψηφίων από τα οποία αποτελείται η ψηφιακή λέξη μετά τη μετατροπή είναι οκτώ, οπότε λέμε ότι το σύστημα είναι οκτάμπιτο και μπορεί να έχει μέχρι και 256 στάθμες. Ισοδύναμα μπορεί να υπάρχει μετατροπέας A/D των 10 ψηφίων όπου θα έχει 1024 στάθμες , μετατροπέας των 12 ψηφίων όπου θα έχει 4096 στάθμες κ.ο.κ. Όσο μεγαλύτερος είναι ο ψηφιακός αριθμός τόσο καλύτερα μπορεί να προσεγγίσει το αναλογικό σήμα .

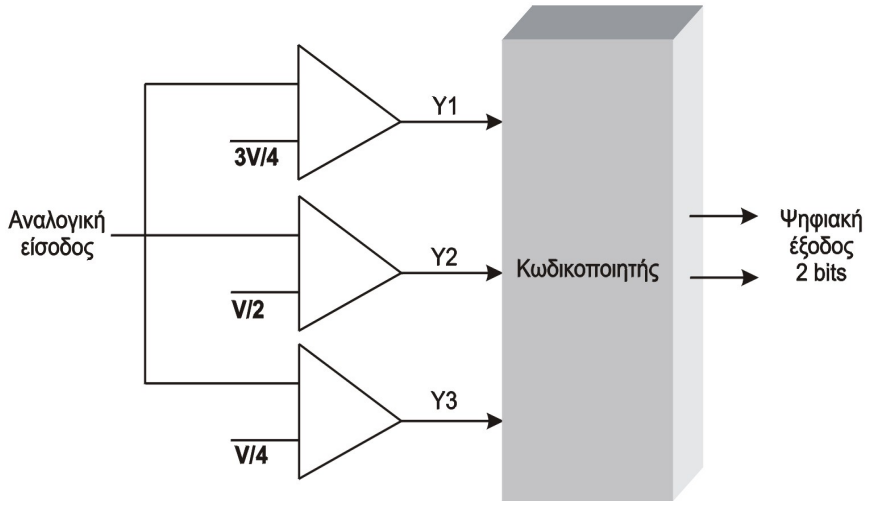
Ένας A/D μετατρέπει αναλογικό σήμα με σταθερή τιμή Vi σε ψηφιακό κώδικα και εκφράζει το λόγο Vi/Vref ,όπου Vref είναι η τάση αναφοράς η οποία συγκρίνεται με τη τάση εισόδου. Όταν η

τάση εισόδου γίνει ίση με τη τάση αναφοράς ο ψηφιακός κώδικας θα έχει τη μέγιστη τιμή. Το χρονικό διάστημα που ο A/D εκτελεί τη μετατροπή, η αναλογική τάση εισόδου πρέπει να διατηρείται σταθερή στην είσοδο του A/D. Αυτή την εργασία την υλοποιεί το κύκλωμα δειγματοληψίας και συγκράτησης που σημαίνει πως μόλις πάρει ένα δείγμα η τάση θα πρέπει να παραμείνει σταθερή μέχρι να γίνει η μετατροπή από τον A/D. Το κύκλωμα δειγματοληψίας μπορεί να είναι είτε ενσωματωμένο στον A/D μετατροπέα είτε και να είναι ένα ξεχωριστό κύκλωμα που συνδέεται με τον A/D.

Οι μετατροπείς A/D διακρίνονται σε μονοπολικούς και διπολικούς. Η έξοδος του A/D μετατροπέα δίνεται από τη σχέση:

$$\text{Out}_{10} = \frac{Vin * 2^n}{Vref} \qquad (1.10)$$

Οι πιο απλοί και γρήγοροι μετατροπείς A/D είναι αυτοί που διαθέτουν έναν αριθμό από συγκριτές. Το κύκλωμα του συγκριτή λειτουργεί με τη βοήθεια ενός τελεστικού ενισχυτή που λαμβάνει το δυναμικό αναφοράς (Vref) και συγκρίνει οποιαδήποτε στιγμή τη τάση που δέχεται από την είσοδο. Όταν η τάση αναφοράς είναι μεγαλύτερη από τη τάση εισόδου τότε η έξοδος του A/D μετατροπέα θα παράγει λογικό ένα, στην αντίθετη περίπτωση δηλαδή η τάση αναφοράς όταν είναι μικρότερη από τη τάση εισόδου θα παράγει λογικό μηδέν.



Σχήμα 1.1 Απλός Ταυτόχρονος μετατροπέας A/D με έξοδο δύο ψηφία

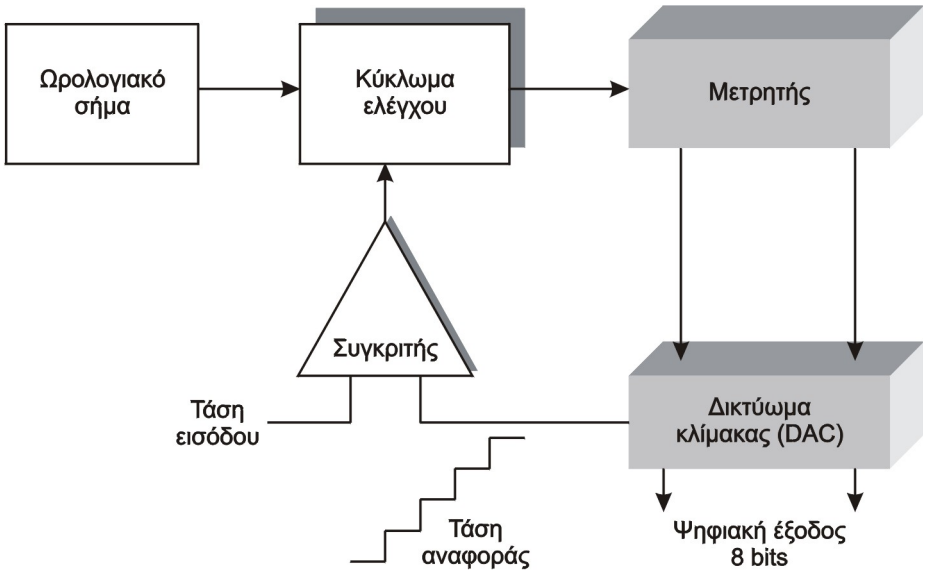
Στο Σχήμα 1.1 φαίνεται ένας ταυτόχρονος μετατροπέας με τρεις συγκριτές. Οι τάσεις αναφοράς των συγκριτών διαιρούν τη συνολική δυναμική περιοχή τάσεων (0 έως V volts) που διαβάζει ο μετατροπέας A/D σε τέσσερις περιοχές. Ανάλογα με τη περιοχή στην αναλογική τάση εισόδου, οι έξοδοι Y1, Y2 και Y3 μπορούν να βρεθούν σε μια από τις τέσσερις λογικές καταστάσεις που υπάρχουν στον παρακάτω Πίνακα 1.1. Ένας κωδικοποιητής τέσσερα προς δύο (4:2) αναλαμβάνει να μεταφέρει τη πληροφορία εξόδου σε δυο ψηφία. Ωστόσο ένας τέτοιος μετατροπέας υλοποιείται δύσκολα και έχει σημαντικό κόστος γι' αυτό και χρησιμοποιούμε άλλες τεχνικές μετατροπής.

Τάση Εισόδου	Y1	Y2	Y3
Από 0 έως V/4	0	0	0
Από V/4 έως V/2	1	0	0
Από V/2 έως 3V/4	1	1	0
Από 3V/4 έως V	1	1	1

Πίνακας 1.1 Οι έξοδοι Y1, Y2, Y3 του Σχήματος 1.1 για διάφορες τιμές αναλογικής τάσης εισόδου

1.3 Κλιμακωτή Τάση Αναφοράς - Η Τεχνική της Διαδοχικής Προσέγγισης

Η πιο γνωστή μέθοδος για την παραγωγή μιας μεταβλητής τάσης αναφοράς για σύγκριση με την αναλογική τάση εισόδου είναι η χρήση ενός μετρητή (απαριθμητή) σε συνδυασμό με ένα μετατροπέα D/A. Σε αυτή τη διαδικασία ο μετρητής απαριθμεί τις καταστάσεις, τις οποίες δέχεται στην είσοδο του ένα δικτύωμα κλίμακας. Το δικτύωμα αυτό δίνει στην έξοδο μια κλιμακωτή τάση από 0 έως V Volts όπου V είναι η μέγιστη τάση εξόδου που παράγει ο μετατροπέας D/A, η οποία εξαρτάται από τη τάση αναφοράς. Το βήμα της κλίμακας εξαρτάται από τα ψηφία του μετρητή άρα σ' ένα οκτάμπιτο σύστημα το βήμα είναι ίσο με V/255. Ένας μετατροπέας A/D με μετρητή και δικτύωμα κλίμακας εικονίζεται στο Σχήμα 1.2.



Σχήμα 1.2 Μετατροπέας A/D με μετρητή και δικτύωμα κλίμακας

Κάθε σκαλοπάτι της τάσης αναφοράς εφαρμόζεται στην μια είσοδο του συγκριτή, ενώ η αναλογική τάση που πρέπει να ψηφιοποιηθεί εφαρμόζεται στην άλλη είσοδο του συγκριτή. Όσο η τάση αναφοράς είναι μικρότερη από την αναλογική τάση εισόδου το κύκλωμα ελέγχου επιτρέπει την εφαρμογή του ωρολογιακού παλμού του ρολογιού (CLOCK) στον μετρητή. Αυτό έχει σαν αποτέλεσμα να αυξάνεται το δυαδικό περιεχόμενο της εξόδου του. Όταν η τάση αναφοράς γίνει μεγαλύτερη ή ίση με την τάση εισόδου, ο συγκριτής θα αλλάξει κατάσταση στην έξοδο του και το κύκλωμα ελέγχου διακόπτει την εφαρμογή των παλμών ρολογιού στον μετρητή. Σαν επακόλουθο οι γραμμές εξόδου του μετρητή περιέχουν τώρα τη δυαδική κωδικοποίηση της αναλογικής τάσης εισόδου και μπορούν να μεταφερθούν στη ψηφιακή έξοδο. Η παραπάνω τεχνική χαρακτηρίζεται από σχετικά μεγάλο χρόνο μετατροπής, καθώς ο χρόνος που χρησιμοποιείται δεν είναι ίδιος για

όλες τις αναλογικές τάσης εισόδου. Το πρόβλημα αυτό μπορεί να λυθεί ως ένα βαθμό με τη τεχνική *διαδοχικής προσέγγισης (successive approximation).*

Στη διαδοχική προσέγγιση ο μετρητής μηδενίζεται και στη συνέχεια το πρώτο σημαντικό ψηφίο (MSB) γίνεται μονάδα (SET). Αν ο συγκριτής βρει την τάση που βγάζει το δικτύωμα κλίμακας μικρότερη από την τάση εισόδου, το κύκλωμα ελέγχου διατηρεί τη μονάδα και γίνεται μονάδα το δεύτερο σημαντικό ψηφίο (MSB). Στη συνέχεια υπάρχει νέα σύγκριση για να καθοριστεί αν θα διατηρηθεί η δεύτερη μονάδα ή θα μηδενιστεί το αντίστοιχο ψηφίο πριν γίνει μονάδα (SET) το τρίτο MSB. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να φτάσουμε στο ελάχιστα σημαντικό ψηφίο (LSB), όπου η έξοδος του μετρητή έχει πλέον τη ζητούμενη δυαδική τάση.

Οι λογικές βαθμίδες που απαιτούνται για την καταμέτρηση διαδοχικής προσέγγισης κατασκευάζονται συνήθως σ' ένα μόνο κύκλωμα μέσης κλίμακας ολοκλήρωσης. Η διάταξη αυτή ονομάζεται *καταχωρητής διαδοχικών προσεγγίσεων ή SAR*.

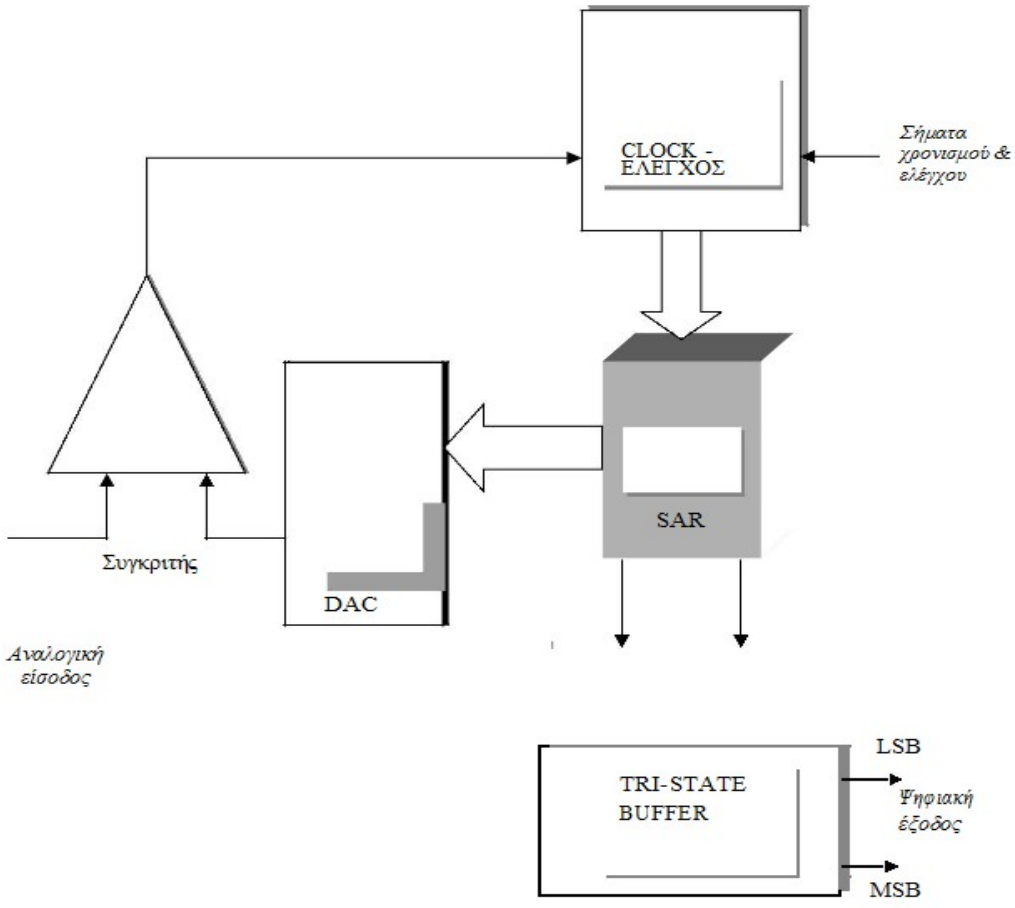
1.4 Ολοκληρωμένο Κύκλωμα ADC0804

Το ολοκληρωμένο *κύκλωμα ADC0804* είναι ένας μετατροπέας αναλογικού σήματος σε ψηφιακό (A/D converter) μήκους οκτώ ψηφίων, που βασίζεται στην τεχνική της διαδοχικής προσέγγισης και έχει συνολικά είκοσι ακροδέκτες. Είναι σχεδιασμένος να δέχεται και να στέλνει σήματα ελέγχου, ώστε η λειτουργία του να μπορεί να ελέγχεται από μικροεπεξεργαστές ή μικροελεγκτές. Παρόλο που ελέγχετε από άλλα κυκλώματα μπορεί να λειτουργήσει και αυτόνομα. Πρόκειται δηλαδή για ένα οικονομικό και αξιόπιστο κύκλωμα.

1.4.1 Περιγραφή Κυκλώματος ADC0804

Ο μετατροπέας αυτός δέχεται μια διαφορά δυναμικού ανάμεσα στους ακροδέκτες Vin(+) και Vin(-) (ακροδέκτες 6 και 7, αντίστοιχα) και παράγει μια ψηφιακή έξοδο στους ακροδέκτες 11 έως 18. Το λιγότερο σημαντικό bit (LSB) αντιστοιχεί στον ακροδέκτη 18. Ο χρονισμός επιτυγχάνεται με ένα δικτύωμα RC, που τοποθετείται ανάμεσα στους ακροδέκτες 4 και 19. Οι ακροδέκτες 1, 2, 3 και 5 προορίζονται για τον έλεγχο του μετατροπέα από εξωτερικές συσκευές. Μπορούν να συνδεθούν σε κατάλληλες λογικές στάθμες, ώστε το κύκλωμα να λειτουργεί χωρίς εξωτερικό έλεγχο. Η τάση τροφοδοσίας που δέχεται το κύκλωμα είναι V_{CC} = 5V στον ακροδέκτη 20. Ο τυπικός χρόνος μετατροπής είναι 100 μs.

Στο ολοκληρωμένο κύκλωμα ADC0804, εάν δεν ορίσουμε κάτι διαφορετικό, ο μετατροπέας ADC0804 είναι σχεδιασμένος ώστε να δέχεται στη διαφορική είσοδο διαφορές δυναμικού από 0 έως 5 Volts. Ο ακροδέκτης Vin(+) πρέπει απαραίτητα να βρίσκεται σε πιο θετικό δυναμικό από τον ακροδέκτη Vin(-). Ο μετατροπέας μπορεί να διαχειριστεί αρνητικές τάσεις με τη βοήθεια ενός κατάλληλου δικτυώματος στο κύκλωμα εισόδου.

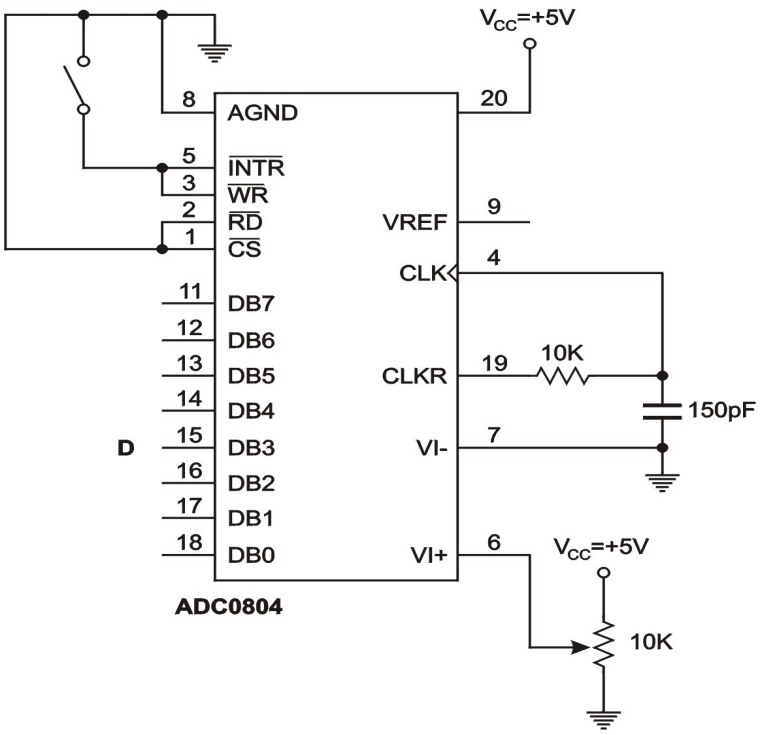


Σχήμα 1.3 Διάγραμμα βαθμίδων ενός τυπικού μετατροπέα A/D Διαδοχικών προσεγγίσεων

1.4.2 Ισοδύναμο Κύκλωμα ADC0804

Ένα ισοδύναμο κύκλωμα φαίνεται στο Σχήμα 1.4, όπου φαίνεται ένας ADC0804 που λειτουργεί χωρίς τη βοήθεια εξωτερικού ελέγχου. Για να γίνει αυτό συνδέεται στις εισόδους CLK ένα δικτύωμα RC, που εξασφαλίζει τον αυτοχρονισμό του κυκλώματος.

Αυτό το κύκλωμα αποτελεί μια τυπική εφαρμογή ελεύθερης λειτουργίας του ολοκληρωμένου κυκλώματος ADC0804. Στις εξόδους του μετατροπέα μπορούν να συνδεθούν οκτώ δίοδοι φωτοεκπομπής, τα λεγόμενα LEDs, σε σειρά με τις αντιστάσεις προστασίας έτσι ώστε να απεικονίζεται η λογική κατάσταση της ψηφιακής εξόδου.



Σχήμα 1.4 Εφαρμογή ελεύθερης λειτουργίας του μετατροπέα ADC0804

1.5 Μετατροπή Ψηφιακού Σήματος σε Αναλογικό D/A

Η *μετατροπή ψηφιακού σήματος σε αναλογικό* είναι μια διεργασία η οποία υλοποιείται με τη χρήση ψηφιακών/αναλογικών μετατροπέων (Digital to Analog converter ή D/A converter ή DAC). Δηλαδή ένα κύκλωμα που έχει ψηφιακή είσοδο και τη μετατρέπει σε αναλογική τάση ή ρεύμα είναι ουσιαστικά ένας μετατροπέας ψηφιακού σήματος σε αναλογικό. Η ψηφιακή είσοδος σε αυτά τα συστήματα αντιστοιχεί σε μια αναλογική έξοδο.

Ο D/A μετατροπέας, δημιουργεί εκ νέου μεταβολή της ψηφιακής λέξης σε αναλογική τάση χρησιμοποιώντας μια κλίμακα στην οποία η αναλογική τάση εξόδου να είναι ελάχιστη όταν όλα τα ψηφία είναι μηδέν και μέγιστη όταν όλα τα ψηφία είναι ένα. Επιπλέον ένας D/A έχει διακριτική ικανότητα, δηλαδή μπορεί να προσδιορίσει τον αριθμό n των ψηφίων που μπορεί να δεχτεί. Όσο μεγαλύτερος είναι ο αριθμός n των ψηφίων τόσο περισσότερες θα είναι οι κβαντικές στάθμες 2^n που θα έχει το σύστημα, με αποτέλεσμα να έχουμε μεγαλύτερη ακρίβεια στο αποτέλεσμα μας. Για παράδειγμα εάν έχουμε έναν D/A μετατροπέα όπου δέχεται 8 ψηφία, δηλαδή n=8 τότε θα έχουμε σαν αποτέλεσμα 2^8=256 διαφορετικές τιμές. Εάν έχουμε έναν D/A μετατροπέα όπου δέχεται 16 ψηφία δηλαδή n=16, τότε θα έχουμε σαν αποτέλεσμα 2^{16} = 65.536 διαφορετικές τιμές κ.ο.κ.

Ένας απλός μετατροπέας D/A τεσσάρων bits φαίνεται στο Σχήμα 1.5, ο οποίος αποτελείται από μία σειρά αντιστάσεων και ένα τελεστικό ενισχυτή. Δέχεται 4 ψηφία άρα θα έχουμε σαν αποτέλεσμα 2^4 =16 διαφορετικές τιμές. Το ρεύμα για κάθε κλάδο δίνεται από τη σχέση :

$$Ii=Vi/Ri \quad (1.1)$$

Γίνεται εύκολα αντιληπτό σύμφωνα με το Σχήμα 1.5 πως οι τιμές των αντιστάσεων θα είναι R, R/2, R/4 και R/8 αντίστοιχα. Η τάση εξόδου του κυκλώματος μας είναι συνάρτηση των ρευμάτων εισόδου του και φαίνεται στους παρακάτω τύπους.

$$V_0=(I_1+I_2+I_3+I_4)x(-Rf) \quad (1.2)$$

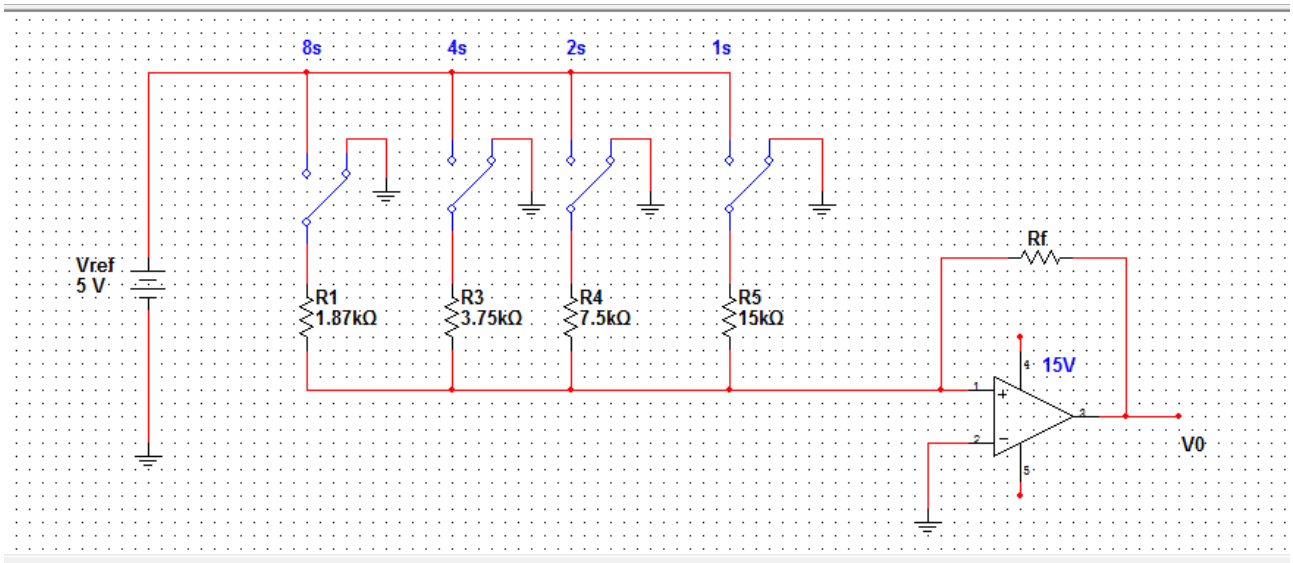
ή

$$V_0 = \left(\frac{V1}{R1} + \frac{V2}{R2} + \frac{V3}{R3} + \frac{V4}{R4} \right) x(-Rf) \, . \qquad (1.3)$$

Ο γενικός τύπος της εξόδου τάσης V_0, όπου η είσοδος δέχεται δυαδικό κώδικα $b_3b_2b_1b_0$ θα είναι

$$V_0 = A\left(\frac{Vref}{R} \right)(-Rf) \qquad (1.4)$$

όπου $A=b_3x2^3+b_2x2^2+b_1x2^1+b_0x2^0$ και το κάθε bit αντιστοιχεί στη λογική ψηφιακή τιμή 0 ή 1.



Σχήμα 1.5 Απλός μετατροπέας D/A τεσσάρων bits

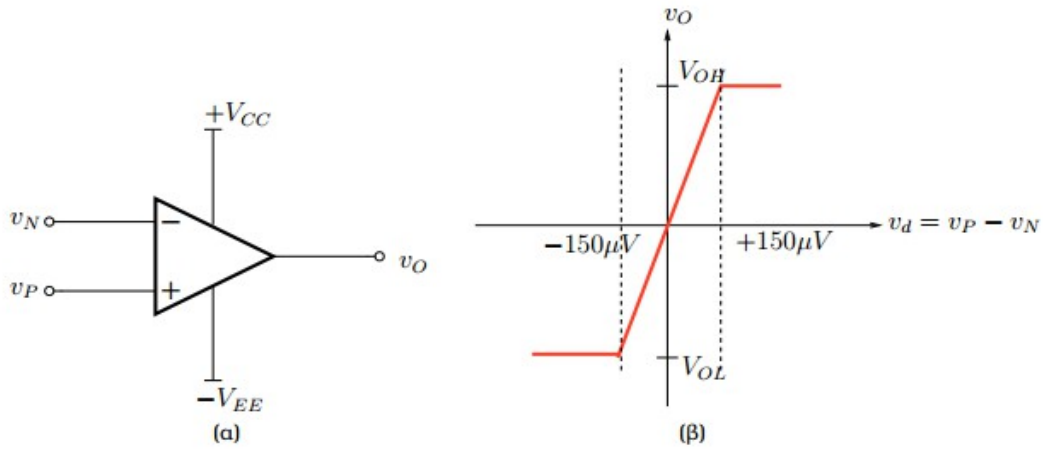
Εάν βάλουμε στη ψηφιακή μας είσοδο τον δυαδικό αριθμό 1111 η τάση εξόδου στον μετατροπέα τεσσάρων bits θα είναι ίση σύμφωνα με τον παρακάτω τύπο.

$$V_0 = (I+2I+4I+8I)(-Rf) = 15\left(\frac{Vref}{R} \right)(-Rf), \qquad (1.5)$$

όπου Vref είναι η μέγιστη τάση. Το κύκλωμα του παραδείγματος έχει μειονεκτήματα όσον αφορά τη σειρά των αντιστάσεων. Αντικαθίσταται από ένα άλλο δικτύωμα που ονομάζεται κύκλωμα κλίμακας R-2R, το οποίο θα αναλύσουμε στη συνέχεια.

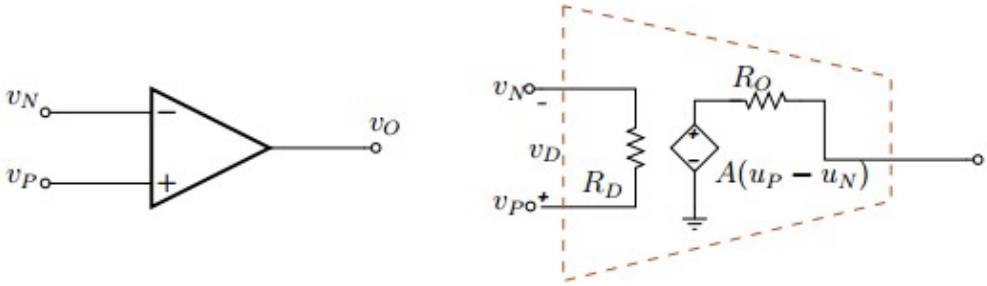
1.6 Σύντομη Περιγραφή για την Λειτουργία του Τελεστικού Ενισχυτή (ΤΕ)

Ο *τελεστικός ενισχυτής* είναι ένας ενισχυτής τάσης με πολύ μεγάλο κέρδος. Το κέρδος μπορεί να παίρνει πολύ μεγάλες τιμές, συνήθως μεταξύ 10^4 και 10^6. Ο τελεστικός ενισχυτής αποτελεί το βασικό δομικό στοιχείο σε μια πληθώρα αναλογικών εφαρμογών. Μπορεί να χρησιμοποιηθεί αυτόνομα σε ένα αναλογικό κύκλωμα, ενώ στην περίπτωση των ολοκληρωμένων αναλογικών κυκλωμάτων χρησιμεύει σαν μοντέλο *αφαιρετικής λειτουργίας (abstraction)* για τα αρχικά στάδια του σχεδιασμού, και στη συνέχεια η υλοποίηση του με τρανζίστορ προσαρμόζεται, ώστε να ταιριάζει καλύτερα στις ανάγκες της εφαρμογής. Στο Σχήμα 1.6α και 1.6β απεικονίζεται ο τελεστικός ενισχυτής αλλά και παράδειγμα εισόδου-εξόδου.



Σχήμα 1.6α Το σχηματικό του τελεστικού ενισχυτή
Σχήμα 1.6β Παράδειγμα χαρακτηριστικής εισόδου-εξόδου

Ο τελεστικός ενισχυτής όπως φαίνεται από το σχηματικό διάγραμμα του σχήματος 1.6(α), αποτελείται από δύο εισόδους, την αναστρέφουσα είσοδο U_N (συνδέεται στον ακροδέκτη − του τελεστικού ενισχυτή) και τη μη αναστρέφουσα είσοδο U_P (συνδέεται στον ακροδέκτη + του τελεστικού ενισχυτή), και μία έξοδο τη Uout. Επίσης στον τελεστικό ενισχυτή συνδέονται δύο τάσεις τροφοδοσίας +VCC και −VEE αντίστροφης πολικότητας (Υπάρχουν μοντέλα εκτελεστικών ενισχυτών που δέχονται μία τάση τροφοδοσίας). Στο Σχήμα 1.7 βλέπουμε το μοντέλο λειτουργίας του τελεστικού ενισχυτή.



Σχήμα 1.7 Μοντέλο λειτουργίας του τελεστικού ενισχυτή

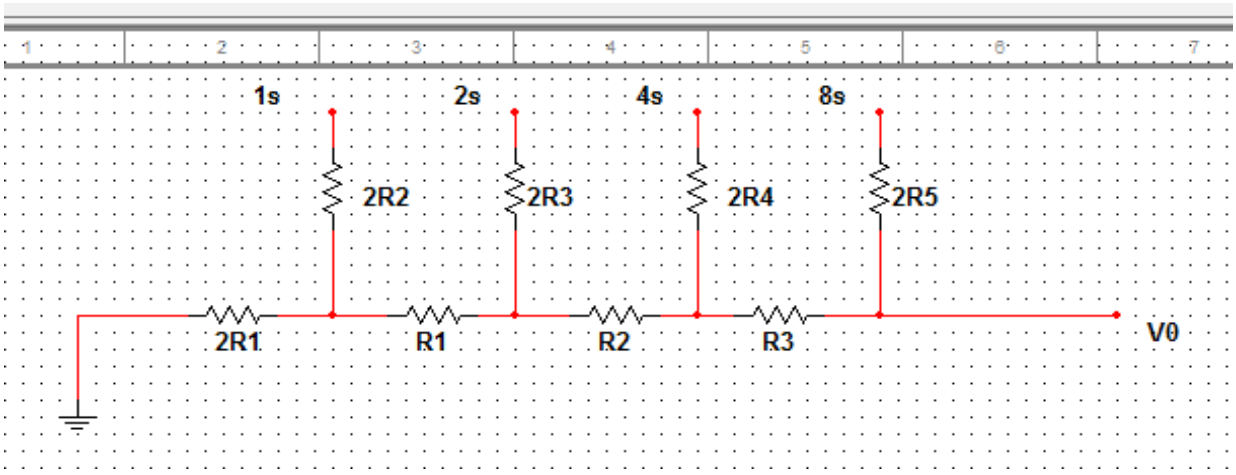
Άρα καταλήγουμε ότι η κύρια λειτουργία του τελεστικού ενισχυτή είναι να ενισχύει τη διαφορά δυναμικού μεταξύ των εισόδων του.

1.7 Κύκλωμα Κλίμακας R-2R

Ένα *δικτύωμα αντιστάσεων R-2R* τεσσάρων bits για τη μετατροπή ψηφιακού σήματος σε αναλογικό περιέχει μόνο δύο τιμές αντιστάσεων. Στο Σχήμα 1.8 παρακάτω εικονίζονται οι κάθετες αντιστάσεις οι οποίες έχουν διπλάσια τιμή από τις οριζόντιες αντιστάσεις. Λόγο αυτής της ιδιαιτερότητας προήλθε και η ονομασία κλίμακα R-2R.

Σε αυτό το κύκλωμα η είσοδος αντιστοιχεί στο πλέον σημαντικό bit, το οποίο όταν έρθει σε δυναμικό V, το ρεύμα εξόδου θα αυξάνεται κατά V/(2R) και αν και οι επόμενες είσοδοι έρθουν σε δυναμικό V τότε το ρεύμα θα αυξάνεται διαδοχικά κατά

$$\frac{V}{R2^2} \quad , \quad \frac{V}{R2^3} \quad , \quad \frac{V}{R2^4} \quad \text{κ.ο.κ.}$$



Σχήμα 1.8 Δικτύωμα αντιστάσεων R-2R

Εάν στις εισόδους έχουμε τη μέγιστη τιμή που είναι το 1 τότε έχουμε τάση Vref της οποίας το ρεύμα εξόδου I_0 μας δίνεται από τη σχέση:

$$I_0=Vref\left(\frac{b3}{2R}+\frac{b2}{4R}+\frac{b1}{8R}+\frac{b0}{16R}\right) \tag{1.6}$$

όπου $b_0b_1b_2b_3$ είναι ο δυαδικός κώδικας εισόδου με τιμές 0 ή 1. Θέτουμε με K= Vref/R και το ρεύμα εξόδου προς τον τελεστικό ενισχυτή θα είναι :

$$I_0=K\left(\frac{b3}{2}+\frac{b2}{4}+\frac{b1}{8}+\frac{b0}{16}\right). \tag{1.7}$$

Ο γενικός τύπος που μας οδηγεί σε αυτή την εξίσωση είναι :

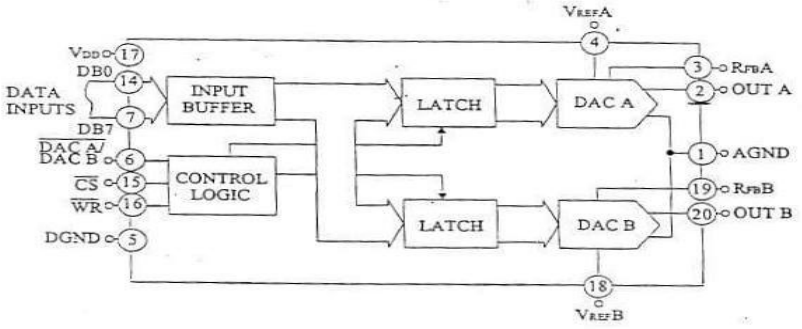
$$I_0=K\ \frac{[\psi\eta\phi\iota\alpha\kappa\acute{\eta}\ \epsilon\acute{\iota}\sigma o\delta o\varsigma]10}{2^n}\ . \tag{1.8}$$

Για να μετατραπεί το ρεύμα σε τάση με τη βοήθεια του τελεστικού ενισχυτή πρέπει να ισχύει η παρακάτω σχέση :

$$V_0=-Vref\ \frac{[\psi\eta\phi\iota\alpha\kappa\acute{\eta}\ \epsilon\acute{\iota}\sigma o\delta o\varsigma]10}{2^n}\ . \tag{1.9}$$

1.8 Ολοκληρωμένο Κύκλωμα DACES52110

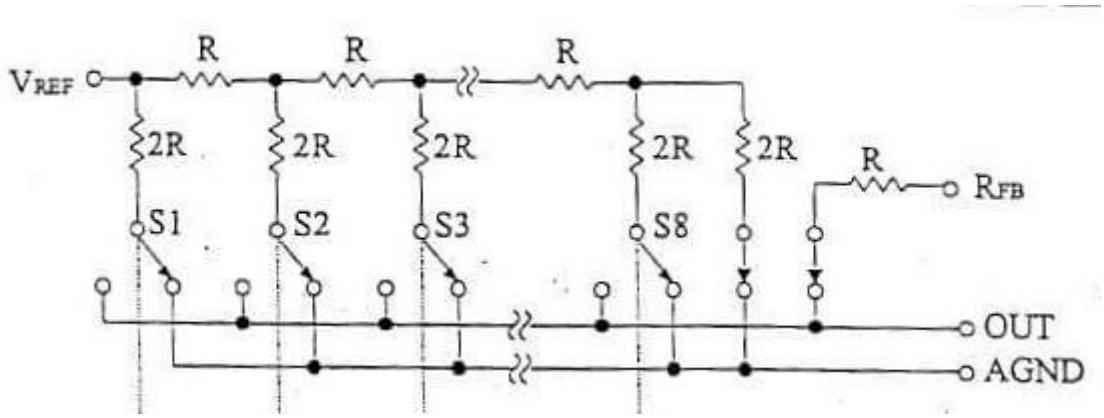
Ο ES52110 περιλαμβάνει ένα διπλό πολλαπλασιαστικό ολοκληρωμένο κύκλωμα των 8 ψηφίων. Αυτό το μονολιθικό κύκλωμα αποτελείται, από δύο δικτυώματα αντιστάσεων κλίμακας R-2R, δύο μανδαλωτές δεδομένων, έναν απομονωτή εισόδου, μια εσωτερική αντίσταση ανάδρασης για το κάθε DAC και ένα κύκλωμα λογικού ελέγχου.



Σχήμα 1.9 Σχηματικό Διάγραμμα DACES52110

1.8.1 Περιγραφή Κυκλώματος DACES52110

Ο DACES52110 περιλαμβάνει δύο πανομοιότυπους πολλαπλασιαστές μετατροπής των 8 ψηφίων, το DAC A και τον DAC B. Ο κάθε D/A μετατροπέας περιλαμβάνει ένα δικτύωμα R-2R και οχτώ διακόπτες n-κάναλιων που κατευθύνουν το ρεύμα. Το απλοποιημένο κύκλωμα D/A για τον DAC A βρίσκεται στο Σχήμα 1.10.

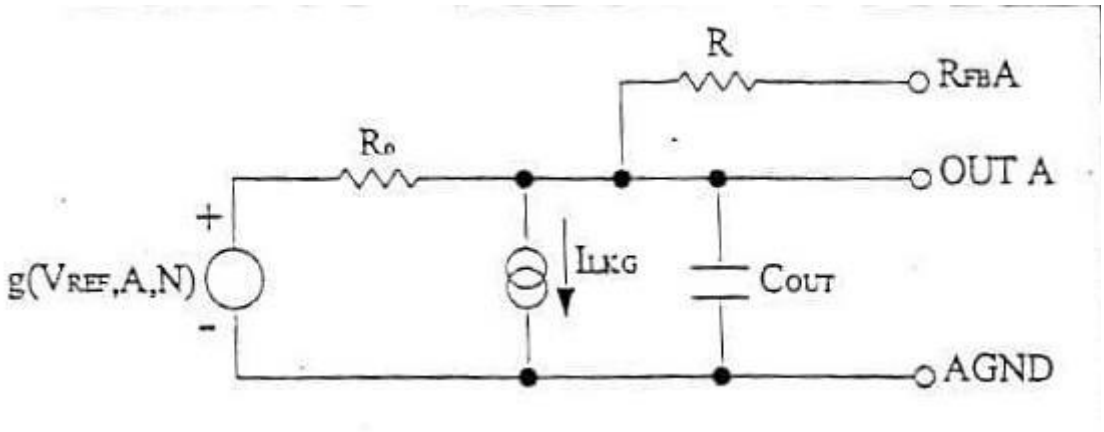


Σχήμα 1.10 Απλοποιημένο Κύκλωμα DAC A ή DAC B

Η δομή του R-2R είναι κυκλώματα δυαδικής μορφής που εναλλάσσονται μεταξύ της εξόδου DAC και της γείωσης AGND, έτσι διατηρούν τον καθορισμό του ρεύματος για κάθε δικτύωμα ανεξάρτητα από την κατάσταση του διακόπτη. Υπάρχει συνήθως ένας κλειστός διακόπτης σε σειρά με την εσωτερική αντίσταση ανάδρασης (R_{FB}).

1.8.2 Ισοδύναμο Κύκλωμα DACES52110

Ένα ισοδύναμο κύκλωμα φαίνεται στο Σχήμα 1.11 για έναν από τους μετατροπείς του D/A και συγκεκριμένα για τον DAC A. Να σημειώσουμε πως η γείωση είναι κοινή είτε πρόκειται για τον DAC A είτε για τον DAC B. Το κύκλωμα αυτό αποτελείται από μια εσωτερική αντίσταση ανάδρασης. Το ρεύμα όπως φαίνεται στο Σχήμα 1.11 είναι το I_{LKG}. Επιπλέον υπάρχει μια αντίσταση R_0 συνδυασμένη σε σειρά με την τάση αναφοράς Vref. Τέλος υπάρχει ένας πυκνωτής Cout που η χωρητικότητα του κυμαίνεται από 50pF μέχρι 120pF, ανάλογα με τη ψηφιακή είσοδο του κυκλώματος.



Σχήμα 1.11 Ισοδύναμο Κύκλωμα Αναλογικής Εξόδου του DAC A

ΚΕΦΑΛΑΙΟ 2

Υλοποίηση Συστημάτων σε διάτρητη πλακέτα

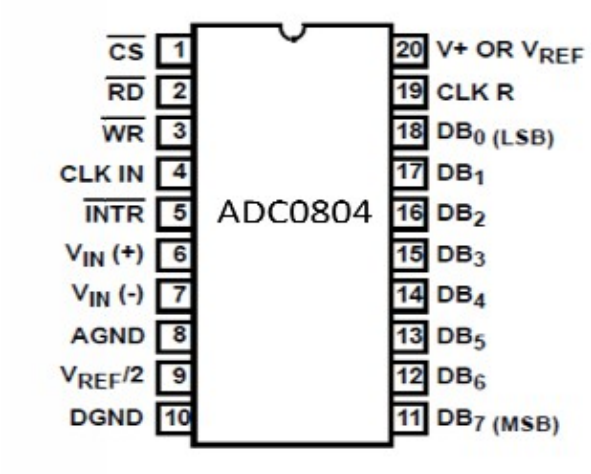
2.1 Ορισμός Διάτρητη Πλακέτας

Η διάτρητη πλακέτα ή BreadBoard ή Raster είναι μία πλακέτα ειδικά σχεδιασμένη για να μας διευκολύνει να δοκιμάσουμε τα κυκλώματα που δημιουργήσαμε, χωρίς τη χρήση κολλητηριού. Μας προσφέρουν γρήγορη και αξιόπιστη λύση για την δημιουργία πρωτοτύπων.

2.2 Υλοποίηση Συστήματος ADC0804 σε Διάτρητη Πλακέτα

Ο ADC που χρησιμοποιήσαμε στην εφαρμογή μας θεωρήθηκε αξιόπιστο κύκλωμα καθώς εκτός από το να τηρεί τις προδιαγραφές που αναφέρθηκαν, έχει χαμηλό κόστος, είναι εύκολος στην χρήση και να είναι ευρέως διαθέσιμος στην τοπική αγορά. Ο ADC0804 είναι ένας ADC των 8 bits και έχει ονομαστικό ρυθμό μετατροπής, σύμφωνα με τον κατασκευαστή, 8888conv/s.

Στο παρακάτω Σχήμα 2.1 αναγράφονται και οι συντομογραφίες των ακροδεκτών του.



Σχήμα 2.1 Συντομογραφίες ακροδεκτών του ADC 0804

Μία σύντομη περιγραφή για τις λειτουργίες των ακροδεκτών:
CS (Chip Select): Ειδοποιεί τον ADC0804 ότι έχει επιλεγεί και τίθεται σε λειτουργία.
WR (Write): Δίνει εντολή στον ADC0804 να εξάγει τα δεδομένα προς τους ακροδέκτες εξόδου.
RD (Read): Δίνει εντολή στον ADC0804 να διαβάσει μια αναλογική τιμή από την είσοδο.
INTR (Interrupt): Ενεργοποιείται όταν ο ADC0804 έχει διαβάσει μια αναλογική τιμή και την έχει αποθηκεύσει στο εσωτερικό του. Απενεργοποιείται όταν η αναλογική τιμή έχει μετατραπεί σε ψηφιακή και έχει μεταφερθεί στην έξοδο.
VIN(+), VIN(-): Οι αναλογικές είσοδοι. Είναι ζευγάρι γιατί μπορούν να χρησιμοποιηθούν και ως διαφορική είσοδος. Αλλιώς η VIN(-) μπορεί να γειωθεί και η VIN(+) να χρησιμοποιηθεί ως μοναδική απόλυτη είσοδος.
VREF/2: Η τάση αναφοράς για την αναλογική είσοδο. Ορίζει το εύρος της αναλογικής εισόδου να είναι ίσο με $V_{REF.}$ Δίνεται υποδιπλασιασμένη τάση καθώς εσωτερικά διπλασιάζεται.
AGND (Analog Ground): Η αναλογική γείωση.
DGND (Digital Ground): Η ψηφιακή γείωση. Αν δεν υπάρχει, ταυτίζεται με την αναλογική

γείωση.
V+: Η τροφοδοσία του ADC0804 (5 Volt).
DB0-7 (Data Bit 0-7): Τα bits εξόδου του ADC0804.

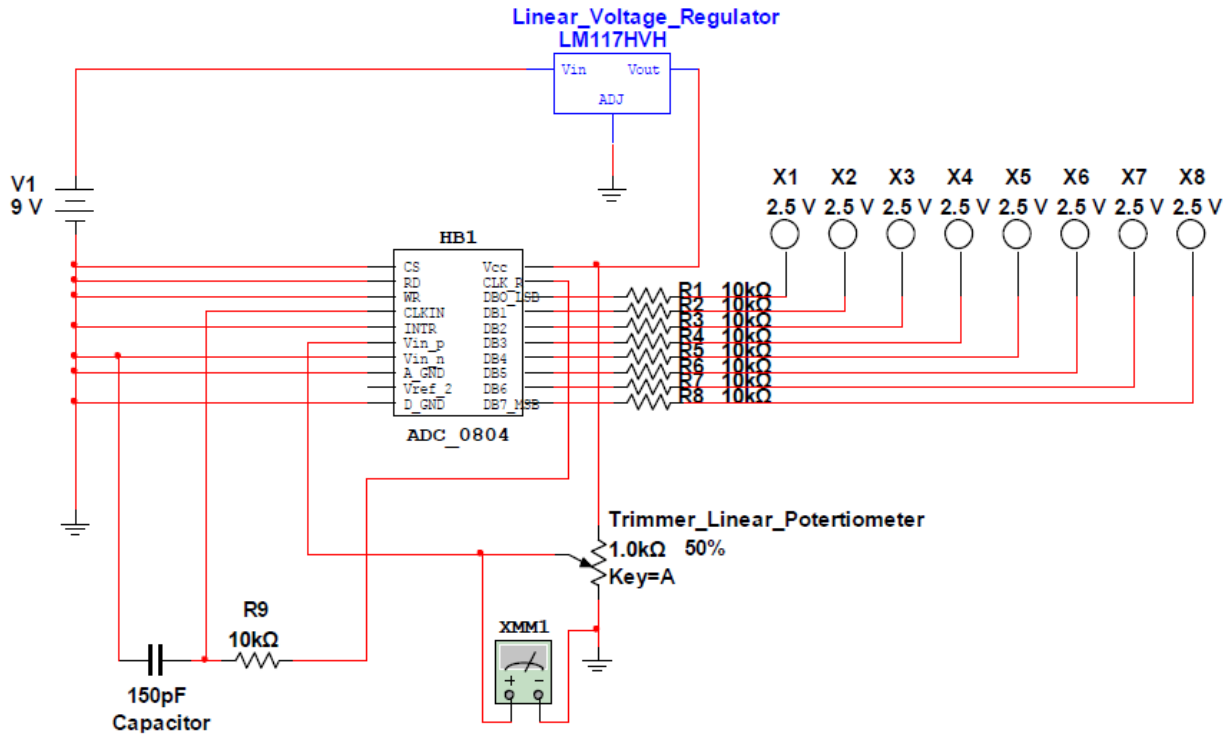
Όπως έγινε αντιληπτό από την περιγραφή τα CS, WR, VIN(+), VIN(-),VREF/2 είναι είσοδοι στον ADC0804, ενώ τα RD, INTR, DB0-7 είναι έξοδοι. Επίσης να τονισθεί ότι, όπως φαίνεται και στο Σχήμα 2.1, τα CS, WR, RD και INTR λειτουργούν με ανάστροφη λογική. Δηλαδή είναι ενεργοποιημένα όταν βρίσκονται σε χαμηλή τάση (LOW) και αντίστοιχα απενεργοποιημένα όταν βρίσκονται σε υψηλή τάση (HIGH).

Η διαδικασία μετατροπής του κυκλώματος πάνω σε ράστερ μπορεί εύκολα να δημιουργηθεί μέσα από τα διαγράμματα χρονισμού που θα περιγράψουμε σε παρακάτω κεφάλαιο.

Αρχικά θέτουμε LOW το CS για την ενεργοποίηση του ADC. Στο κύκλωμά μας αυτό θα γίνει αν συνδέσουμε με ένα καλώδιο τον ακροδέκτη 1 με την γείωση δηλαδή το λογικό μηδέν. Στη συνέχεια θέτουμε και τον ακροδέκτη 3 ,WR, σε LOW με τον ίδιο τρόπο. Όσο το CS και το WR παραμένουν LOW, ο ADC βρίσκεται σε κατάσταση επαναφοράς (reset state).

Επίσης στη γείωση θα καταλήξουν και οι ακροδέκτες 2,5,7,8,10 δηλαδή RD, INTR, Vin(+), A_GND, D_GND αντίστοιχα. Ο ακροδέκτης CLK IN συνδέεται με έναν κόμβο, και συνιστά μια αντίσταση 10kΩ και έναν πυκνωτή 150pF, όπου η άλλη άκρη του πυκνωτή θα συνδεθεί μαζί με το Vin(-) συνεπώς στην γείωση, και η άλλη άκρη της αντίστασης στον ακροδέκτη 19,CLK_R. Ο ακροδέκτης 5, Vin(+) θα συνδεθεί στο trimmer ή αλλιώς Linear Potertiometer, το οποίο είναι αυτό που ρυθμίζει την τάση που θα εισάγουμε μέσα στο κύκλωμα μας, η τάση κυμαίνεται από 0V έως 5V. Τον ακροδέκτη 20 τον συνδέουμε με τα 5V που μας δίνει το Linear Voltage Regulator. Το Linear Voltage Regulator είναι ένα τσιπάκι που του εισάγουμε τα 9V της μπαταρίας και μας επιτρέπει να χρησιμοποιήσουμε μόνο τα 5V, που είναι τα κατάλληλα για την λειτουργία το κυκλώματος. Οι ακροδέκτες 11 έως 18, ή αλλιώς DB7 με DB0 αντίστοιχα, είναι η έξοδος 8 bit του κυκλώματος μας, με πιο σημαντικό ψηφίο (Most Significant Bit) το DB7 και λιγότερο σημαντικό (Least Significant Bit) το DB0. Στη συνέχεια είναι συνδεδεμένες 8 αντιστάσεις των 10kΩ και τέλος 8 λαμπτήρες led, για την εμφανή έξοδο του κυκλώματος.

Στο Σχήμα 2.2 βλέπουμε την απεικόνιση του κυκλώματος μέσα από την εφαρμογή Multisum 11.0. Στην εφαρμογή έχουμε προσθέσει και ένα βολτόμετρο για να βλέπουμε κάθε στιγμή ποια είναι η αναλογική τάση της εισόδου.



Σχήμα 2.2 Απεικόνιση του ADC κυκλώματος στο πρόγραμμα Multisum

Από την παραπάνω περιγραφή φαίνεται το πόσο απλός είναι στην χρήση, αλλά και ευέλικτος, ο ADC0804. Γεγονός που δικαιώνει την επιλογή μας για τη χρήση του στην διάταξη μας. Για την ολοκλήρωση του ADC δίνεται ο Πίνακας 2.1 με τις προδιαγραφές χρονισμού του ADC0804, όπου δίνονται οι ακριβείς τιμές για τους διάφορους χρόνους που περιλαμβάνονται στα διαγράμματα χρονισμού.

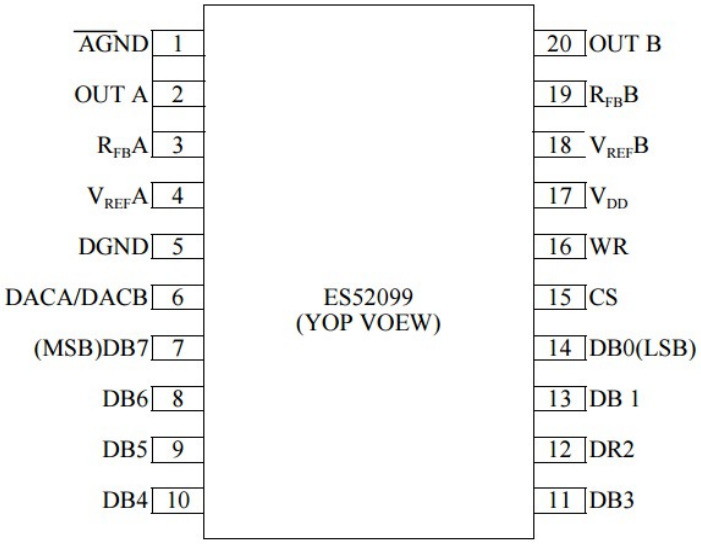
AC TIMING SPECIFICATIONS V+ = 5V, and $T_A = 25^oC$, Unless Otherwise Specified					
Clock Frequency, f_{CLK}	V+ = 6V (Note 4)	100	640	1280	kHz
	V+ = 5V	100	640	800	kHz
Clock Periods per Conversion (Note 5), t_{CONV}		62	-	73	Clocks/Conv
Conversion Rate In Free-Running Mode, CR	$\overline{INTR}$ tied to $\overline{WR}$ with $\overline{CS}$ = 0V, f_{CLK} = 640kHz	-	-	8888	Conv/s
Width of $\overline{WR}$ Input (Start Pulse Width), $t_{W(WR)I}$	$\overline{CS}$ = 0V (Note 6)	100	-	-	ns
Access Time (Delay from Falling Edge of $\overline{RD}$ to Output Data Valid), t_{ACC}	C_L = 100pF (Use Bus Driver IC for Larger $C_{L)}$	-	135	200	ns
Three-State Control (Delay from Rising Edge of $\overline{RD}$ to HI-Z State), t_{1H}, t_{0H}	C_L = 10pF, R_L = 10K (See Three-State Test Circuits)	-	125	250	ns
Delay from Falling Edge of $\overline{WR}$ to Reset of $\overline{INTR}$, t_{WI}, t_{RI}		-	300	450	ns
Input Capacitance of Logic Control Inputs, C_{IN}		-	5	-	pF
Three-State Output Capacitance (Data Buffers), C_{OUT}		-	5	-	pF

Πίνακας 2.1 Προδιαγραφές χρονισμού του ADC0804

2.3 Υλοποίηση Συστήματος DAC ES52110 σε Διάτρητη Πλακέτα

Ο Dac που αναζητούμε για την εφαρμογή μας δεν βρέθηκε στην Ελληνική αγορά, άρα δεν μπόρεσε να υλοποιηθεί η εφαρμογή σε ράστερ. Όμως αντ' αυτού κάναμε την υλοποίηση σε ράστερ με έναν άλλον Dac που ήταν ευκολότερος στην εύρεση του και πληρούσε στις προδιαγραφές.

Στο παρακάτω Σχήμα 2.3 αναγράφονται και οι συντομογραφίες των εξόδων του.



Σχήμα 2.3 Συντομογραφίες των εξόδων του DAC ES52110

Μία σύντομη περιγραφή για τις λειτουργίες των ακροδεκτών:
CS (Chip Select): Ειδοποιεί τον DAC ES52110 ότι έχει επιλεγεί και τίθεται σε λειτουργία.
WR (Write): Δίνει εντολή στον DAC ES52110 να εξάγει τα δεδομένα προς τους ακροδέκτες εξόδου.
AGND/DGND: Αναλογική και ψηφιακή γείωση.
OUTA/OUTB: Για την Ενεργοποίηση της εξόδου ανάλογα με ποια θύρα του DAC χρησιμοποιώ, δηλαδή αν χρησιμοποιήσουμε τη θύρα DAC A θα ενεργοποιηθεί η έξοδος OUTA διαφορετικά αν χρησιμοποιήσουμε τη θύρα DAC B θα ενεργοποιηθεί η έξοδος OUT B.
R_{FB} A/ R_{FB} B: Εσωτερική αντίσταση ανατροφοδότησης.
V_{REF} A/ V_{REF} B: Η τάση αναφοράς για το κύκλωμα R-2R.
DAC A/DAC B: Είναι το σήμα για την επιλογή της εξόδου από τη θύρα a ή b.
VDD: Πηγή.
DB7-DB0: 8-bit ψηφιακής εισόδου.

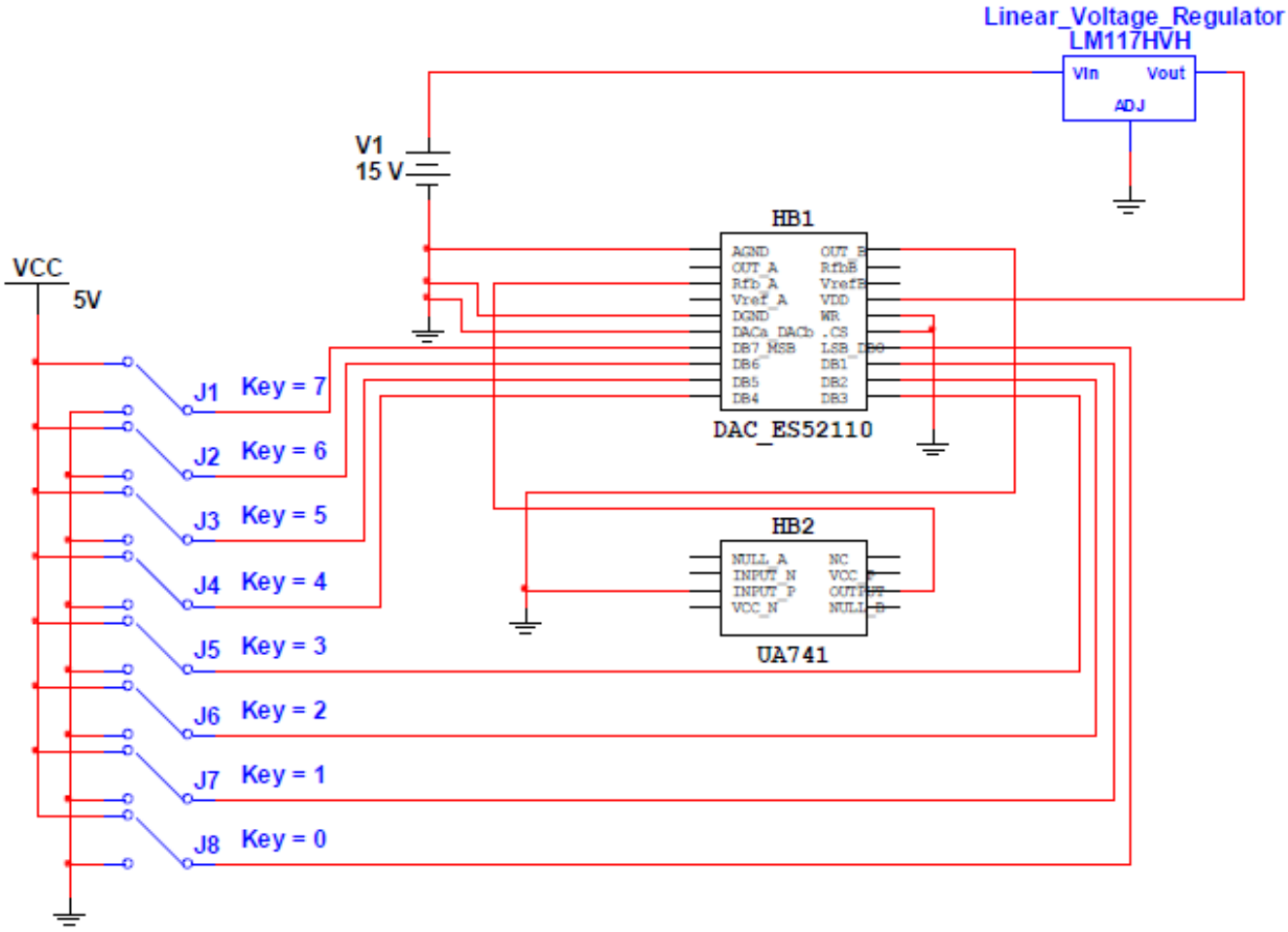
Από την περιγραφή καταλαβαίνουμε ότι τα CS, WR, DGND, V_{DD}, DB0-DB7 και DACA/DACB είναι είσοδοι στον DAC ES52110, ενώ τα R_{FB}A, R_{FB}B, OUTA, OUTB και AGND είναι έξοδοι. Επίσης να τονισθεί ότι, όπως φαίνεται και στο Σχήμα 2.3, τα CS, WR και DACA λειτουργούν με ανάστροφη λογική. Δηλαδή είναι ενεργοποιημένα όταν βρίσκονται σε χαμηλή τάση (LOW) και αντίστοιχα απενεργοποιημένα όταν βρίσκονται σε υψηλή τάση (HIGH).

Η διαδικασία μετατροπής του κυκλώματος πάνω σε ράστερ μπορεί εύκολα να δημιουργηθεί μέσα από τα διαγράμματα χρονισμού που θα περιγράψουμε σε παρακάνω κεφάλαιο.

Αρχικά θέτουμε LOW, στη γείωση, τον ακροδέκτη 15, CS για την ενεργοποίηση του Dac. Στη συνέχεια θέτουμε και τον ακροδέκτη 16, WR, σε LOW με τον ίδιο τρόπο. Όσο το CS και το WR παραμένουν LOW, ο Dac βρίσκεται σε κατάσταση επαναφοράς (reset state). Επίσης στη γείωση θα καταλήξουν και οι ακροδέκτες 1, 5, 6 δηλαδή A_GND, D_GND και DACA/

DACB αντίστοιχα. Ο ακροδέκτης 3, R_{FB}A συνδέεται στον ακροδέκτη 6 output του τελεστικού ενισχυτή (UA741). Επίσης ο ακροδέκτης του τελεστικού ενισχυτή 3, input(+) συνδέεται με τον ακροδέκτη 20, out_B και καταλήγουν μαζί στη γείωση. Ο ακροδέκτης 7, VDD συνδέεται με το Linear Voltage Regulator. Το Linear Voltage Regulator είναι ένα τσιπάκι που ενώ εισάγουμε τα 15V απο τη μπαταρία μας επιτρέπει να χρησιμοποιήσουμε μόνο τα 5V, που είναι τα κατάλληλα για την λειτουργία το κυκλώματος. Τέλος οι ακροδέκτες 7-14 ή αλλιώς DB7 με DB0 αντίστοιχα, είναι η είσοδος 8 bit του κυκλώματος μας, με πιο σημαντικό ψηφίο (Most Significant Bit) το DB7 και λιγότερο σημαντικό (Least Significant Bit) το DB0. Στη συνέχεια οι ακροδέκτες είναι συνδεδεμένοι σε 8 διακόπτες, για την εύκολη επιλογή της εισόδου του κυκλώματος.

Στο Σχήμα 2.4 βλέπουμε την απεικόνιση του κυκλώματος μέσα από την εφαρμογή Multisum 11.0.



Σχήμα 2.4 Απεικόνιση του DAC κυκλώματος στο πρόγραμμα Multisum

ΚΕΦΑΛΑΙΟ 3

Εισαγωγή στη γλώσσα VHDL και στο πρόγραμμα Quartus II

3.1 Γλώσσα περιγραφής υλικού VHDL

Η VHDL (VHSIC hardware description language ή γλώσσα περιγραφής υλικού VHSIC) είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται στον αυτοματισμό ηλεκτρονικών σχεδιάσεων (electronic design automation) για τη περιγραφή ψηφιακών και μεικτών (mixed-signal) συστημάτων, όπως η επιτόπια συστοιχία προγραμματιζόμενων πυλών (FPGA) και τα ολοκληρωμένα κυκλώματα.

Η VHDL συνήθως χρησιμοποιείται για τη συγγραφή μοντέλων σε κείμενο που περιγράφουν ένα λογικό κύκλωμα. Η VHDL έχει δομές που χειρίζονται τον παραλληλισμό που υπάρχει στις σχεδιάσεις υλικού, αλλά αυτές οι δομές (διεργασίες ή processes) διαφέρουν στη σύνταξη σε σύγκριση με τις παράλληλες δομές της Ada (εργασίες ή tasks). Η VHDL έχει ισχυρούς τύπους και δεν κάνει διάκριση μεταξύ κεφαλαίων και μικρών γραμμάτων. Επιπλέον έχει δυνατότητες εισόδου και εξόδου σε αρχεία και μπορεί να χρησιμοποιηθεί σαν γλώσσα γενικών καθηκόντων για επεξεργασία κειμένου. Για έναν προγραμματιστή χωρίς εμπειρία είναι σχετικά εύκολο να παράγει κώδικα που προσομοιώνεται με επιτυχία αλλά δε μπορεί να παραχθεί σαν πραγματική υλοποίηση, ή είναι πολύ μεγάλος για να χρησιμοποιηθεί στην πράξη.

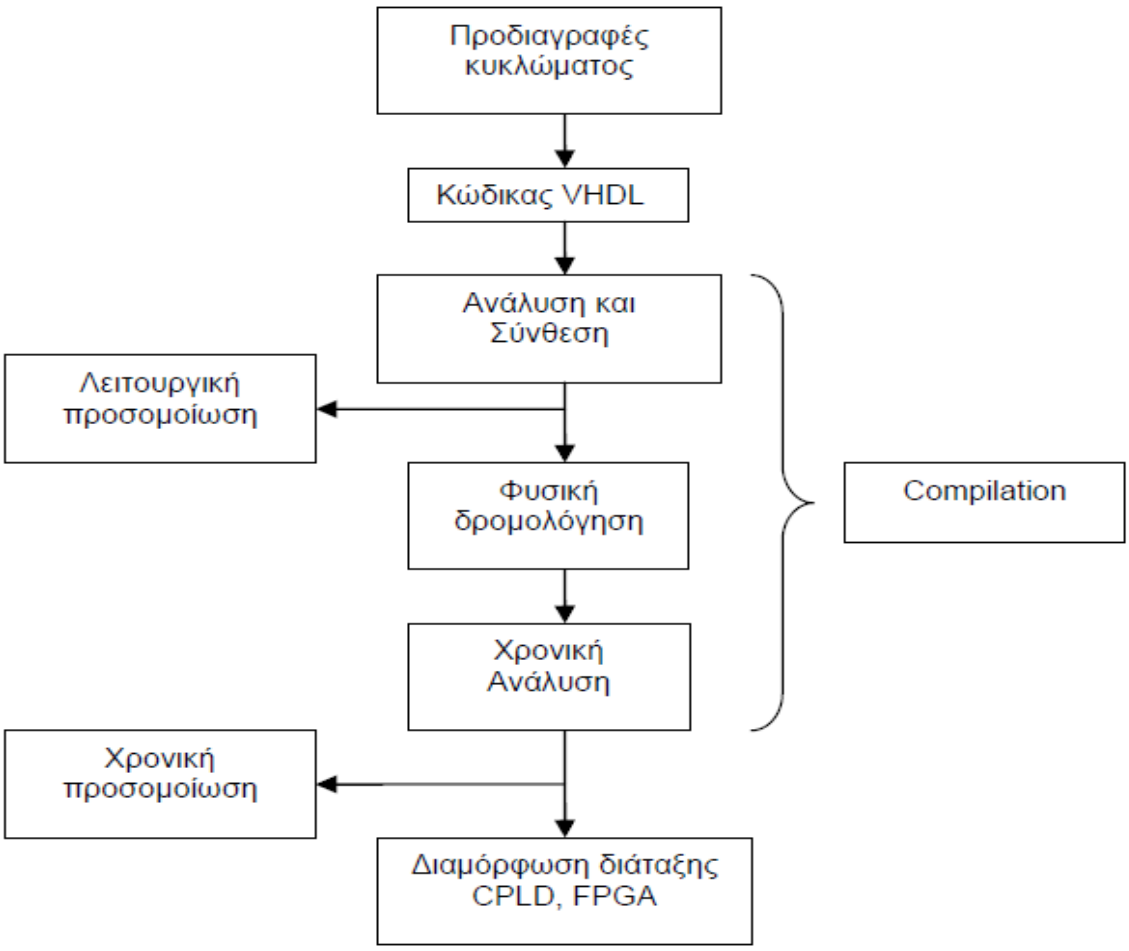
Το βασικό πλεονέκτημα της VHDL, όταν αυτή χρησιμοποιείται για σχεδίαση συστημάτων, είναι ότι επιτρέπει την περιγραφή (μοντελοποίηση) και την επαλήθευση (προσομοίωση) του επιθυμητού συστήματος, πριν τα εργαλεία σύνθεσης μεταφράσουν τη σχεδίαση σε πραγματικό υλικό (πύλες και γραμμές). Ένα άλλο όφελος της VHDL είναι ότι επιτρέπει τον ορισμό ταυτόχρονων συστημάτων. Η VHDL είναι γλώσσα ροής δεδομένων σε αντίθεση με τις διαδικαστικές γλώσσες προγραμματισμού όπως η BASIC η C και η συμβολική γλώσσα οι οποίες εκτελούνται ακολουθιακά, με κάθε εντολή να ακολουθεί την προηγούμενη. Ένα έργο σε VHDL έχει πολλές εφαρμογές.

Σε πολλές περιπτώσεις η γλώσσα VHDL χρησιμοποιεί ασυνήθιστη σύνταξη για την περιγραφή των λογικών κυκλωμάτων. Ο κύριος λόγος για αυτό είναι ότι η γλώσσα VHDL αποσκοπούσε αρχικά στην καταγραφή και προσομοίωση κυκλωμάτων και όχι στην περιγραφή κυκλωμάτων με σκοπό την σύνθεση.

Ο κώδικας VHDL αποτελεί τον βασικό τύπο αρχείου που δέχονται ως είσοδο τα λογισμικά ψηφιακής σχεδίασης κυκλωμάτων (Computer Aided Design ή CAD), για τη δημιουργία σύνθετων ολοκληρωμένων κυκλωμάτων. Η γλώσσα VHDL χρησιμοποιείται ευρύτατα για την περιγραφή και υλοποίηση ψηφιακών συστημάτων σε προγραμματιζόμενες λογικές διατάξεις,τύπου CPLDs (Complex Programmable Logic Devices-Σύνθετες προγραμματιζόμενες λογκές διατάξεις) και FPGAs (Field Programmable Gate Arrays-(ιατάξεις πυλών προγραμματιζόμενες στο πεδίο). Επίσης, έχει καθιερωθεί σαν ένα πρότυπο (standard) στη σχεδίαση ηλεκτρονικών κυκλωμάτων ASICs (Application Specific Integrated Circuits). Η καθιέρωση της ως πρότυπο μας διαβεβαιώνει ότι και οι επόμενες εκδόσεις εργαλείων σχεδίασης θα υποστηρίζουν το πρότυπο αυτό. Έτσι, ένα κύκλωμα που περιγράφηκε και αναπτύχθηκε με τα σημερινά εργαλεία σχεδίασης θα είναι μεταφέρσιμο μελλοντικά σε νέα εργαλεία σχεδίασης, με ελάχιστες ή καθόλου αλλαγές.

3.1.1 Ροή Σχεδίασης

Στο **Σ**χήμα 3.1 απεικονίζεται η γενική ροή των βημάτων που ακολουθεί ο σχεδιαστής λογικών κυκλωμάτων όταν εργάζεται με κώδικα VHDL.

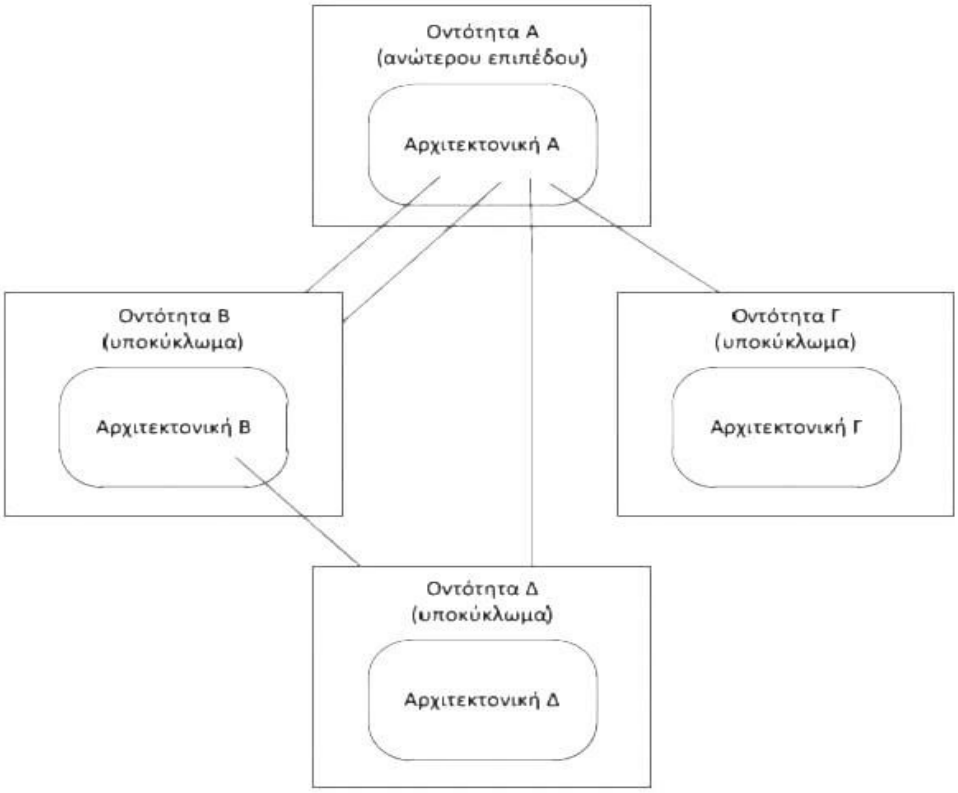


Σχήμα 3.1 Βασική ροή εργασιών κατά τη σχεδίαση με τη γλώσσα VHDL

Πρώτο βήμα είναι να δούμε τις προδιαγραφές του κυκλώματος που περιγράφουν την συμπεριφορά που είναι αναγκαία. Στο δεύτερο βήμα, το κύκλωμα χωρίζεται σε μέρη ακολουθώντας τους κανόνες ιεραρχικής δομημένης σχεδίασης. Τα μέρη για να μπορούν να συνδυαστούν λειτουργικά μεταξύ τους θα πρέπει να σχεδιαστούν με κώδικα VHDL. Ο κώδικας αποθηκεύεται με επέκταση *.vhd*. Έπειτα γίνεται η μεταγλώττιση που αποτελείται από την ανάλυση και σύνθεση, όπου σε αυτό το σημείο επεξεργάζεται ο κώδικας για συντακτικά λάθη και στη συνέχεια, ο compiler σχεδιάζει το κύκλωμα που περιγράφει ο κώδικας. Στη συνέχεια της μεταγλώττισης εκτελείται το πρώτο στάδιο λειτουργικής προσομοίωσης, όπου προστίθενται οι χρονικοί περιορισμοί του τελικού κυκλώματος και ακολουθεί το στάδιο φυσικής δρομολόγησης, όπου κάθε λογική δομή η οποία έχει παραχθεί από τη σύνθεση, βρίσκει τη φυσική της αντιστοίχηση σε μια λογική βαθμίδα μέσα στη διάταξη και ακολουθεί η χρονική προσομοίωση της σχεδίασης. Εφόσον έχουν υλοποιηθεί όλα τα στάδια με επιτυχία και τα αποτελέσματα της προσομοίωσης δείχνουν ότι το κύκλωμα λειτουργεί σύμφωνα με τις προδιαγραφές, ακολουθεί η διαμόρφωση διάταξης, όπου δέχεται σήματα εισόδου και παράγει σήματα εξόδου.

3.1.2 Χαρακτηριστικά VHDL

Η VHDL ακολουθεί τις αρχές των παράλληλων γλωσσών και ταυτόχρονα, ακολουθεί τις αρχές του δομημένου προγραμματισμού, για τη σχεδίαση ιεραρχικών κυκλωμάτων. Επιπλέον περιλαμβάνει τις αρχές του συγχρονισμού και του χρονισμού. Ένα από τα χαρακτηριστικά της VHDL είναι ότι μοντελοποιεί με ακρίβεια τις λειτουργίες του κυκλώματος, αλλά και τους χρόνους κατά τους οποίους οι λειτουργίες παράγουν αποτελέσματα. Η VHDL περιγράφει λειτουργίες σε κάθε πρόταση ή τμήμα κώδικα, οι οποίες παράγουν αποτελέσματα ταυτόχρονα με άλλες λειτουργίες.



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

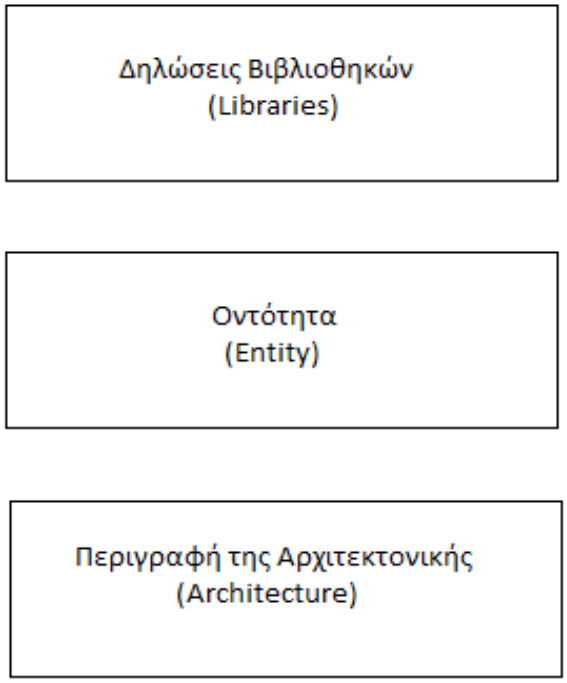
Τα αποτελέσματα της προσομοίωσης παράγονται με αυστηρές χρονικές προδιαγραφές σε διάφορα σημεία του κυκλώματος και εν τέλει στις εξόδους. Στο τέλος, ο compiler θα συνθέσει κυκλώματα που περιγράφει ο κώδικας. Η VHDL περιγράφει σύγχρονες δομές (concurrent code) αλλά και ακολουθιακές δομές (sequential code). Αυτό συμβαίνει καθώς η VHDL παράγει αποτελέσματα είτε ταυτόχρονα με την εφαρμογή των εισόδων είτε σε συγχρονισμό με συμβάντα, όπως για παράδειγμα τους παλμούς ρολογιού.

Ο ιεραρχικός τρόπος σχεδίασης είναι δυνατός στη VHDL λόγω της δομής του κώδικα, που αποτυπώνεται στη περιγραφή του κυκλώματος ως βαθμίδα (block) και στη λειτουργική περιγραφή του κυκλώματος. Όπως θα περιγραφεί παρακάτω, τα δύο αυτά μέρη του κώδικα αναφέρονται ως «οντότητα» και «αρχιτεκτονική». Έχοντας σχεδιάσει ένα κύκλωμα σε γλώσσα VHDL, έχουμε τη δυνατότητα να το συγχωνεύσουμε σε πιο πολύπλοκα κυκλώματα, κάνοντας απλώς αναφορά στην «οντότητα» του, χωρίς να χρειάζεται να επαναλάβουμε την περιγραφή της αρχιτεκτονικής. Επιπλέον μπορούμε να παράγουμε «στιγμιότυπα» (instances) ενός κυκλώματος, όσες φορές χρειάζεται, κάνοντας αναφορά σ' αυτό μέσα στην αρχιτεκτονική μιας ανώτερης ιεραρχικής βαθμίδας, όπως βλέπουμε και στο Σχήμα 3.2. Όταν ένα κύκλωμα περιγράφεται μία φορά μπορεί να χρησιμοποιηθεί ξανά σαν βαθμίδα υποκυκλώματος, σε οποιοδήποτε άλλο ψηφιακό σύστημα. Με

τον τρόπο αυτό, το hardware στην κωδική του περιγραφή, καθίσταται ευέλικτο, αφού πλέον μπορεί να διαμοιραστεί στους χρήστες σαν λογισμικό ή να χρησιμοποιηθεί για την ανάπτυξη μεγάλων συστημάτων, μέσω βιβλιοθηκών που περιλαμβάνουν βαθμίδες υποκυκλωμάτων. Τέτοιες βιβλιοθήκες υπάρχουν έτοιμες στα μεγάλα εργαλεία ψηφιακής σχεδίασης, όπως το Quartus II, αλλά μπορούν να δημιουργηθούν και από τον κάθε σχεδιαστή, ανάλογα με τις δικές του ανάγκες σχεδίασης.

3.2 Δομή Προγραμμάτων σε VHDL

Σε αυτή την παράγραφο περιγράφονται τα βασικά στοιχεία της γλώσσας VHDL, δηλαδή οι βασικοί κανόνες σύνταξης και η βασική δομή ενός προγράμματος. Τα βασικά μέρη ενός κώδικα αποτελούνται από τις *βιβλιοθήκες (library)*, την *οντότητα (entity)* και την *αρχιτεκτονική (architecture)*, όπως φαίνεται και στο Σχήμα 3.3. Στις περισσότερες περιπτώσεις, πριν το τμήμα της οντότητας θα πρέπει να δηλωθούν και κάποια πακέτα βιβλιοθηκών, που περιγράφουν τους τύπους δεδομένων ή υποκυκλώματα που χρησιμοποιεί ο σχεδιαστής.



Σχήμα 3.3 Βασικά τμήματα ενός αρχείου VHDL

Ο κώδικας VHDL συντάσσεται με μια σειρά από «λεκτικά στοιχεία» (lexical elements), δηλαδή ειδικά αλφαριθμητικά, με τη βοήθεια των οποίων διατυπώνονται δηλώσεις, γίνεται ανάθεση τιμών, ονομάζονται τμήματα κώδικα, συντάσσονται εντολές και ορίζονται τελεστές πράξεων. Ανάμεσα στα λεκτικά στοιχεία διακρίνουμε τις δεσμευμένες λέξεις (αλλιώς λέξεις-κλειδιά) και τα αναγνωριστικά.

Οι λέξεις *library, entity* και *architecture* είναι δεσμευμένες λέξεις και κάθε φορά που τις πληκτρολογούμε στον επεξεργαστή κειμένων το χρώμα των γραμμάτων θα είναι διαφορετικό από αυτό του υπόλοιπου κώδικα για να βοηθήσει τον προγραμματιστή στη σύνταξη του κώδικα. Ωστόσο δεν είναι μόνο αυτές οι λέξεις δεσμευμένες. Υπάρχουν και άλλες όπως *signal, port, proccess, downto, end, case, if, is, then, when, wait* κ.α.

Επιπλέον πολύ σημαντικό στη σύνταξη του κώδικα είναι ο χαρακτήρας τέλους γραμμής (;) που σημειώνεται όταν ολοκληρώνεται μια εντολή της VHDL.

3.2.1 Βιβλιοθήκες και πακέτα

Οι δηλώσεις των βιβλιοθηκών (libraries) είναι συλλογή χρήσιμων τμημάτων κώδικα**,** που έχουν ήδη δημιουργηθεί στο παρελθόν και χρησιμοποιούνται συχνά. Οι βιβλιοθήκες δηλώνονται με τη λέξη-κλειδί *library*. Ένα έτοιμο, τυποποιημένο πακέτο που χρησιμοποιείται πολύ συχνά είναι το *std_logic_1164* της βιβλιοθήκης *ieee*, το οποίο περιγράφει τον τύπο δεδομένων std_logic, ενώ τα πακέτα εισάγονται με τη λέξη-κλειδί *use*. Τα *πακέτα (package)* χρησιμοποιούνται για να συγκρατούν προγράμματα της VHDL που είναι γενικής χρήσης, όπως τα προγράμματα που ορίζουν ένα τύπο. Τα βασικά πακέτα που ανήκουν στη βιβλιοθήκη αυτή είναι:

- *std_logic_1164*, είναι πακέτο που εισάγει τον τύπο δεδομένων std_logic και std_logic_vector και επιτρέπει τη χρήση πολλών τιμών σε ένα σήμα εκτός από τις απλές τιμές '0' και '1'. Οι επιπλέον τιμές είναι η κατάσταση «υψηλής εμπέδησης» ('Z'), η «αδιάφορη» κατάσταση ('-'), η «άγνωστη» κατάσταση ('X') και άλλες τιμές που είναι χρήσιμες σε ορισμένες περιπτώσεις.
- *numeric_std*, που εισάγει τους τύπους *signed* και *unsigned,* με βάση τον τύπο std_logic. Το πακέτο επιτρέπει την εκτέλεση αριθμητικών πράξεων με προσημασμένους ή μη προσημασμένους αριθμούς.
- *std_logic_arith* είναι πακέτο hardware και ορίζει τύπους σημάτων signed και unsigned. Επιπλέον επιτρέπει την τέλεση αριθμητικών πράξεων (+, -, *, /) ανάμεσα σε σήματα.
- *numeric_std ή std_logic_arith* είναι πακέτο για τη χρήση προσημασμένων αριθμών σε σήματα αριθμητικών κυκλωμάτων. Τα δύο αυτά πακέτα δεν είναι ισοδύναμα, άρα δεν πρέπει να χρησιμοποιούνται ταυτόχρονα.

Στο Σχήμα 3.4 φαίνεται η δήλωση βιβλιοθηκών στο λογισμικό Quartus II.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
```

Σχήμα 3.4 Δήλωση Βιβλιοθηκών και πακέτων

3.2.2 Οντότητα

Τα σήματα εισόδου και εξόδου μιας οντότητας ορίζονται με τη λέξη - κλειδί *entity.* Το όνομα της οντότητας μπορεί να είναι οποιοδήποτε όνομα επιθυμεί ο προγραμματιστής, αρκεί να μην είναι κάποιο από τις δεσμευμένες λέξεις. Τα σήματα εισόδου και εξόδου καθορίζονται με τη βοήθεια της λέξης - κλειδί *port (θύρα)*. Τα σήματα θυρών μπορούν να οριστούν ως τύπου std_logic_vector και η μετατροπή τους να γίνει στο σώμα της αρχιτεκτονικής. Οι τύποι signed και unsigned ορίζονται σε δύο διαφορετικά πακέτα της βιβλιοθήκης ieee. Το ένα ονομάζεται std_logic_arith. Το άλλο είναι επίσημα προτυποποιημένο πακέτο και ονομάζεται numeric_std. Το *downto* στη δήλωση του τύπου μας δείχνει πόσα ψηφία έχει το σήμα. Παρακάτω στο Πίνακα 3.1 περιγράφονται οι δυνατές καταστάσεις σημάτων που είναι θύρες οντοτήτων. Η σύνταξη της οντότητας γίνεται ως εξής:

```
ENTITY όνομα_οντότητας IS
        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);
```

Κατάσταση	Σκοπός
IN	Χρησιμοποιείται για ένα σήμα που είναι είσοδος σε μια οντότητα.
OUT	Χρησιμοποιείται για ένα σήμα που είναι έξοδος σε μια οντότητα. Η κατάσταση σε σήμα σαν αυτό μπορεί να αναγνωριστεί μόνον από άλλες βαθμίδες τις οποίες τα σήματα τροφοδοτούν, όχι όμως από το εσωτερικό της οντότητας.
INOUT	Χρησιμοποιείται για ένα σήμα που είναι και είσοδος και έξοδος σε μια οντότητα.
BUFFER	Χρησιμοποιείται για ένα σήμα που είναι έξοδος σε μια οντότητα αλλά ταυτόχρονα μπορούν να διαβαστούν στο εσωτερικό της οντότητας.

Πίνακας 3.1 Δυνατές καταστάσεις σημάτων που είναι θύρες οντοτήτων

3.2.3 Αρχιτεκτονική

Η *αρχιτεκτονική (architecture)* δίνει τις λεπτομέρειες του κυκλώματος της οντότητας. Αποτελείται από δύο μέρη. Το πρώτο μέρος είναι η *περιοχή δήλωσης (declarative region)*, όπου εμφανίζεται πριν από τη λέξη-κλειδί *begin* και δηλώνονται σήματα, τύποι ορισμένοι από τον χρήστη και σταθερές. Το δεύτερο μέρος είναι το *σώμα* της *αρχιτεκτονικής (architecture body)* και περιλαμβάνει εντολές που καθορίζουν τις λογικές συναρτήσεις που εκτελεί το κύκλωμα. Η λειτουργία μιας οντότητας καθορίζεται στο σώμα της αρχιτεκτονικής μετά τη λέξη-κλειδί *begin*. Η γενική μορφή της αρχιτεκτονικής έχει ως εξής:

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

Μετά τις δηλώσεις των σημάτων και τη δεσμευμένη λέξη-κλειδί begin, μπορεί να ακολουθήσει η εντολή δ*ιαδικασίας (process)*, η οποία μπορεί να περικλείει στο εσωτερικό της άλλες εντολές, όπως case, if, loop και άλλες. Η γενική μορφή της process έχει ως εξής:

```
PROCESS (λίστα ευαισθησίας)
        BEGIN
                Εντολές που περιγράφουν συνδυαστικά ή ακολουθιακά
                κυκλώματα
        END  PROCESS;
```

Η process ανανεώνει τις αναθέσεις τιμών σε σήματα που αναφέρονται στο εσωτερικό της μόνον όταν αλλάξουν οι τιμές των σημάτων που υπάρχουν στη λίστα ευαισθησίας της. Κατά την προσομοίωση του κυκλώματος ο μεταγλωττιστής θα εκτελέσει τις εντολές στο εσωτερικό της δομής process σειριακά και θα αποδώσει στα σήματα μόνον τις τελικές τιμές τους, αφού ολοκληρωθεί η σειριακή εκτέλεση της διεργασίας. Η process αναθέτει τιμές στα σήματα και οι αναθέσεις αυτές ανανεώνονται μόνον κατά τις μεταβάσεις των σημάτων που περιλαμβάνονται στη

λίστα ευαισθησίας.

Οι εντολές if, case και loop μπορούν να χρησιμοποιηθούν για να περιγράψουν συνδυαστικά ή ακολουθιακά κυκλώματα.

Η *εντολή if* είναι εντολή διακλάδωσης υπό συνθήκη. Γράφεται μόνον μέσα σε ακολουθιακό τμήμα κώδικα, όπως είναι η διαδικασία που περιγράψαμε παραπάνω process. Ελέγχει αν μια συνθήκη είναι αληθής, οπότε εκτελεί το μέρος του κώδικα που ακολουθεί, αλλιώς ελέγχει μια νέα σειρά συνθηκών και εκτελεί το αντίστοιχο μέρος του κώδικα. Η γενική μορφή της if έχει ως εξής:

```
IF συνθήκη THEN
        Προτάσεις ανάθεσης;
ELSIF συνθήκη THEN
        Προτάσεις ανάθεσης
ELSE
        Προτάσεις ανάθεσης;
END IF;
```

Η *εντολή case* ικανοποιεί τις ανάγκες στην περιγραφή μηχανών πεπερασμένων καταστάσεων (Finite State Machines). Η μηχανή πεπερασμένων καταστάσεων είναι ακολουθιακό κύκλωμα που πραγματοποιείται με τη βοήθεια συνδυαστικής λογικής ενός ή περισσοτέρων Flip-Flops. Όπως και η if έτσι και η case θεωρούνται καθαρά ακολουθιακές εντολές και τοποθετούνται σε διεργασίες και υποπρογράμματα.

Η διαφορά της case από την if είναι ότι η case επιλέγει με βάση τη τιμή που λαμβάνει μια μοναδική έκφραση, ενώ η if εξετάζει διαδοχικά την σειρά λογικών συνθηκών για να επιλέξει την ομάδα εντολών που θα εκτελέσει. Η γενική μορφή της case έχει ως εξής:

```
CASE έκφραση IS
        WHEN τιμή => αναθέσεις;
        WHEN τιμή => αναθέσεις;
        ....
END CASE;
```

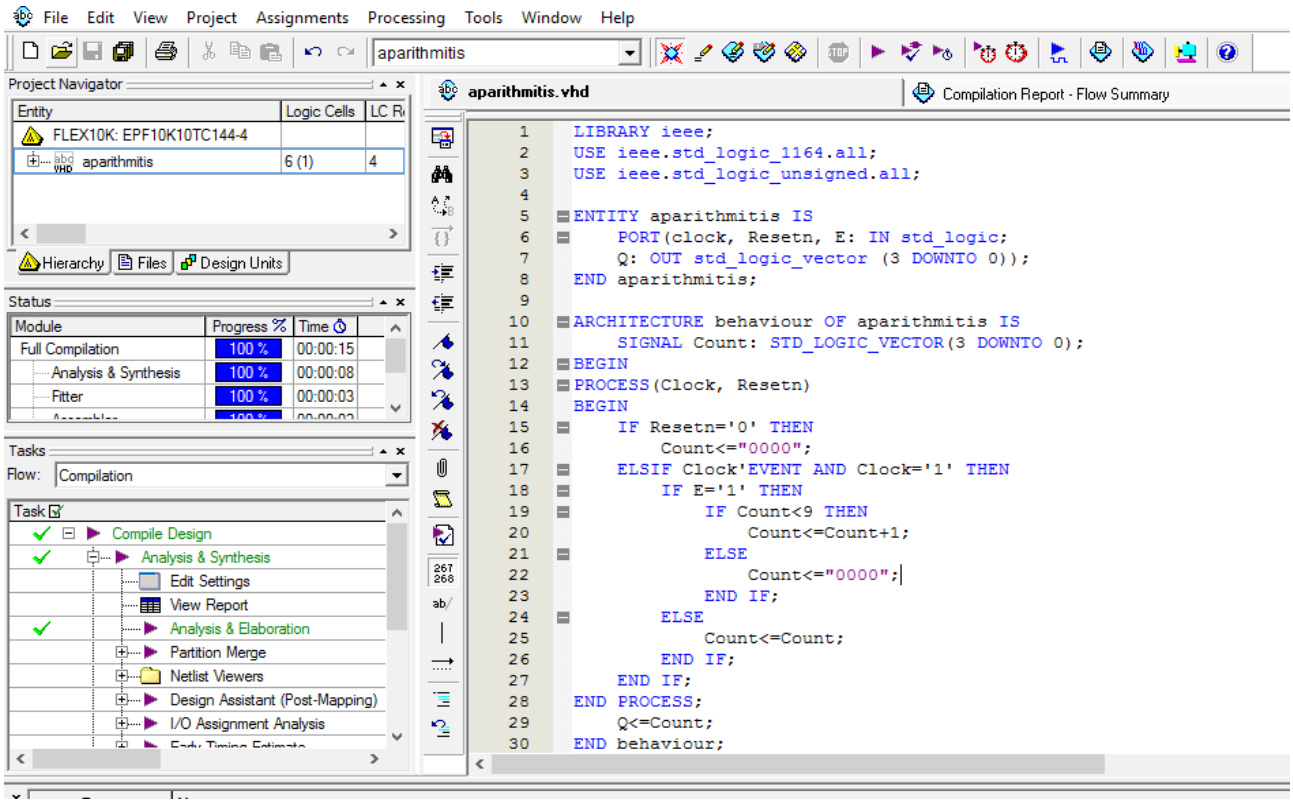
Η γλώσσα VHDL έχει δύο τύπους εντολών βρόγχου. Την *εντολή for-loop* και την *εντολή while-loop.* Οι εντολές αυτές χρησιμοποιούνται για την επανάληψη μίας ή περισσοτέρων εντολών ακολουθιακής αντιστοίχησης. Η loop πρέπει να βρίσκεται μέσα σε διεργασία ή υποπρόγραμμα. Η αντίστοιχη σύγχρονη εντολή της loop είναι η *generate.*

Η γλώσσα VHDL χειρίζεται τρία *αντικείμενα δεδομένων (data objects)*, τα οποία μεταφέρουν τη πληροφορία μέσα στο σύστημα και αποδίδουν τιμές σε διάφορα σημεία. Συνήθως αναγράφονται στην αρχιτεκτονική του κώδικα πριν από τη δεσμευμένη λέξη-κλειδί begin. Σε κάθε αντικείμενο δίνουμε ένα όνομα και το ορίζουμε σύμφωνα με κάποιο τύπο δεδομένων(data type). Επίσης, κάθε αντικείμενο έχει κάθε στιγμή κάποια τιμή. Τα αντικείμενα δεδομένων που χρησιμοποιούνται είναι: Τα *σήματα (signals)*, οι *μεταβλητές (variables)*, οι *σταθερές (constants)* και τα *αρχεία (files)* τα οποία δεν είναι για τη σύνθεση κώδικα αλλά για τη προσομοίωση.

Τα σήματα είναι τα πιο σημαντικά από τα αντικείμενα δεδομένων, αφού αποδίδουν τιμές στα καλώδια του κυκλώματος και αντιπροσωπεύουν τις διασυνδέσεις ανάμεσα σε μονάδες του κυκλώματος. Μπορούν να χρησιμοποιηθούν σε τμήματα κώδικα με σύγχρονες εντολές, αλλά και σε ακολουθιακά τμήματα κώδικα.

Οι μεταβλητές λειτουργούν ως μέσων για την προσωρινή αποθήκευση τιμών που δημιουργούνται σαν αποτέλεσμα από την τέλεση αριθμητικών πράξεων. Μια μεταβλητή δηλώνεται και χρησιμοποιείται μόνο σε τμήματα του κώδικα που περιλαμβάνουν ακολουθιακές εντολές.

Επιπλέον δηλώνεται και χρησιμοποιείται μόνον μέσα σε process ή σε υποπρόγραμμα. Για να επηρεάσει το κύκλωμα, η τιμή μιας μεταβλητής θα πρέπει να αποδοθεί σε κάποιο σήμα. Υπάρχει διαφορά στη χρήση σημάτων και μεταβλητών.



Σχήμα 3.5 Δομή προγράμματος σε VHDL ενός Απαριθμητή

Τέλος, οι σταθερές λαμβάνουν τιμή κατά τη δήλωσή τους και η τιμή αυτή παραμένει στη συνέχεια σταθερή. Άρα, η σταθερά απλά μεταφέρει μια συγκεκριμένη αριθμητική τιμή, όπου χρειαστεί.

Για να κατανοήσει κανείς καλύτερα τη δομή ενός προγράμματος στη γλώσσα VHDL παραπάνω στο Σχήμα 3.5 υπάρχει ένα παράδειγμα απαριθμητή.

3.3 Υποκυκλώματα

Μια οντότητα της γλώσσας VHDL που ορίζεται σε ένα αρχείο προγράμματος μπορεί να χρησιμοποιηθεί ως υποκύκλωμα σε ένα άλλο αρχείο προγράμματος. Ένα υποκύκλωμα στη γλώσσα της VHDL ονομάζεται συνιστώσα (component). Το υποκύκλωμα δηλώνεται με μια *δήλωση συνιστώσας (component decleration)*. Η εντολή αυτή καθορίζει το όνομα του υποκυκλώματος και δίνει ονόματα στις θύρες εισόδου και εξόδου του. Η δήλωση συνιστώσας μπορεί να υπάρχει είτε στη περιοχή δηλώσεων της αρχιτεκτονικής είτε σε μια δήλωση πακέτου. Η γενική μορφή μιας τέτοιας δήλωσης είναι ως εξής:

COMPONENT όνομα συνιστώσας
[GENERIC (όνομα παραμέτρου : integer := τιμή;]
PORT (όνομα ακροδέκτη : mode τύπος;
όνομα ακροδέκτη : mode τύπος);

3.4 Τελεστές, πράξεις και χαρακτηριστικά

Στη VHDL μπορούμε να κάνουμε αριθμητικές, λογικές και σχεσιακές πράξεις χρησιμοποιώντας τους αντίστοιχους τελεστές. Υποστηρίζονται τα εξής είδη τελεστών: Τελεστές ανάθεσης (ή αντιστοίχησης), λογικοί τελεστές, αριθμητικοί τελεστές, τελεστές σύγκρισης (σχεσιακοί τελεστές), τελεστές ολίσθησης, ο τελεστής συνένωσης και ορισμένοι άλλοι.

- *Τελεστές ανάθεσης* αποδίδουν τιμές σε σήματα μεταβλητές και σταθερές. Οι τελεστές ανάθεσης και η λειτουργία παραθέτονται στο Πίνακα 3.2.

Τελεστής	**Λειτουργία**	**Παράδειγμα**
<=	Αναθέτει τιμές σε σήματα	a1<=a2+a3;
:=	Αναθέτει τιμές σε μεταβλητές και δίνει αρχικές τιμές σε σήματα, κατά τη δήλωσή τους	g1 :="1101"; g2 := a2 and a2;
=>	Δίνει τιμές σε στοιχεία πινάκων	When start => cs<='0';

Πίνακας 3.2 Τελεστές ανάθεσης και λειτουργία τους

- *Λογικοί Τελεστές* μπορούν να υποστηρίξουν λογικές πράξεις. Τα αποτελέσματα των πράξεων είναι ίδιου τύπου και μεγέθους με τους τελεστέους. Εάν οι πράξεις έχουν διαφορετικό τύπο ή είναι διαφορετικού μεγέθους θα υπάρξει σφάλμα. Ο Πίνακας 3.3 δείχνει τους λογικούς τελεστές.

Τελεστής	**Λειτουργία**
NOT	Αντιστροφή
AND	Και
NAND	Οχι Και
OR	Ή
NOR	Ούτε
XOR	Αποκλειστικό Ή
XNOR	Αποκλειστικό Ούτε

Πίνακας 3.3 Τελεστές λογικών πράξεων

Εκτός από τον τελεστή NOT που έχει μεγαλύτερη προτεραιότητα από τους υπόλοιπους, όλες οι άλλες λογικές πράξεις έχουν την ίδια προτεραιότητα και για αυτό το λόγο καλό είναι να βάζουμε παρενθέσεις όταν σε μια δήλωση έχουμε πολλές λογικές πράξεις.

- *Σχεσιακοί Τελεστές* υλοποιούν σχεσιακές πράξεις, δηλαδή κάνουν συγκρίσεις μεταξύ σημάτων ή μεταβλητών και ελέγχουν αν υπάρχει ισότητα ή ανισότητα μεταξύ τους. Το αποτέλεσμα που προκύπτει από σχεσιακές πράξεις είναι πάντα τύπου Boole, δηλαδή True ή False. Οι σχεσιακές πράξεις έχουν την ίδια προτεραιότητα μεταξύ τους και μεγαλύτερη από τις λογικές πράξεις. Στον Πίνακα 3.4 απεικονίζονται οι τελεστές σχεσιακών πράξεων.

Τελεστής	Λειτουργία
=	Ισότητα
/=	Διάφορο
<	Μικρότερο
<=	Μικρότερο ή Ίσο
>	Μεγαλύτερο
>=	Μεγαλύτερο ή Ίσο

Πίνακας 3.4 Τελεστές Σχεσιακών Πράξεων

- *Τελεστές αριθμητικών πράξεων* οδηγούν στα αντίστοιχα αριθμητικά κυκλώματα, όταν οι τύποι δεδομένων που χρησιμοποιούμε είναι προκαθορισμένοι. Εάν οι τύποι δεν είναι προκαθορισμένοι τότε είναι αναγκαίο να χρησιμοποιήσουμε κάποια αριθμητική βιβλιοθήκη. Στο Πίνακα 3.5 φαίνονται οι τελεστές αριθμητικών πράξεων.

Τελεστής	Λειτουργία
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
**	Ύψωση σε δύναμη
ABS	Απόλυτη τιμή
REM	Υπόλοιπο
MOD	Modulo

Πίνακας 3.5 Τελεστές αριθμητικών πράξεων

3.5 Συνήθη Σφάλματα στα Προγράμματα της Γλώσσας VHDL

Στην ενότητα αυτή θα αναφερθούμε στα συνήθη σφάλματα κατά τον προγραμματισμό ενός κυκλώματος σε γλώσσα VHDL, τα οποία είναι τα εξής :

- Το όνομα που θα χρησιμοποιήσουμε στη δήλωση της οντότητας και στην αρχιτεκτονική πρέπει να είναι το ίδιο.
- Κάθε εντολή της γλώσσας VHDL πρέπει να λήγει με ένα ελληνικό ερωτηματικό.
- Οι απόστροφοι χρησιμοποιούνται για δεδομένα μήκους ενός ψηφίου, ενώ τα εισαγωγικά για δεδομένα μήκους πολλών ψηφίων.
- Οι συνδυαστικές εντολές περιλαμβάνουν αντιστοιχήσεις απλών σημάτων, αντιστοιχήσεις σημάτων επιλογής και εντολές δημιουργίας. Οι αντιστοιχίσεις απλών σημάτων μπορούν να χρησιμοποιηθούν έξω ή μέσα σε μια εντολή διαδικασίας (process). Οι άλλες μορφές συνδυαστικών εντολών μπορούν να χρησιμοποιηθούν μόνο έξω από μια εντολή διαδικασίας.

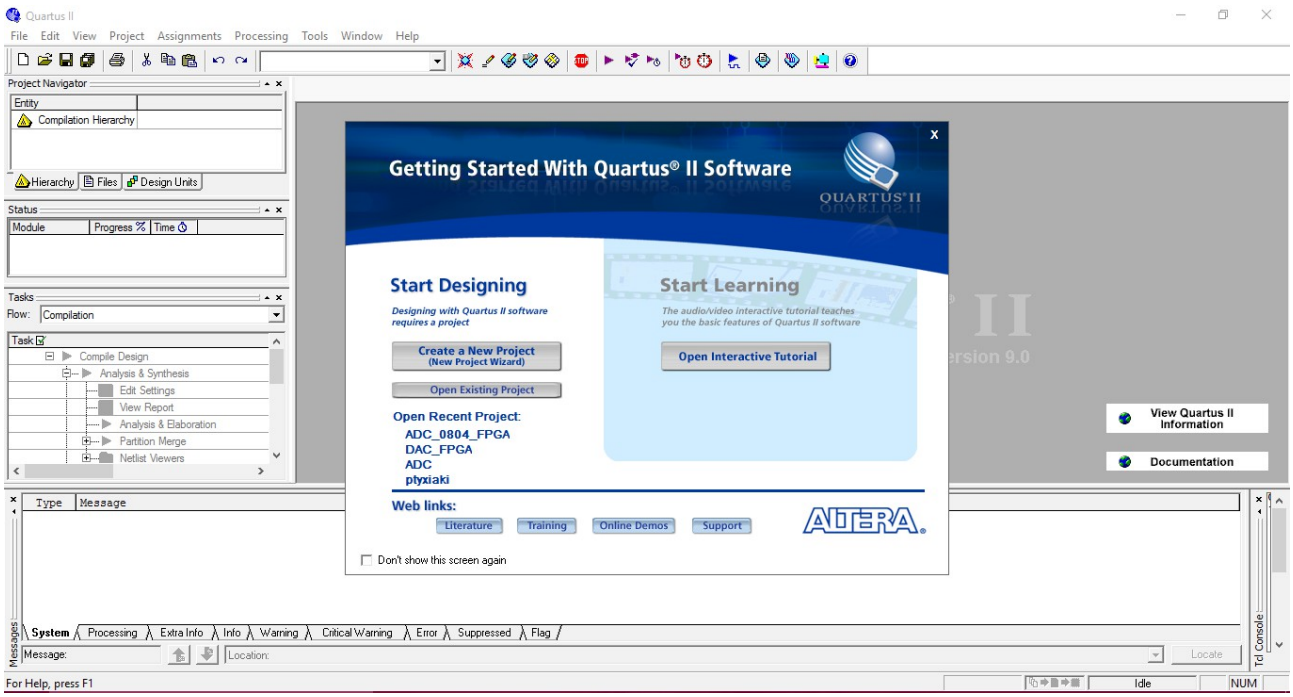
- Απαγορεύεται η χρήση λέξης-κλειδί της γλώσσας VHDL ως όνομα κάποιας επιγραφής, σήματος ή μεταβλητής. Επιπλέον, δεν είναι δυνατόν να χρησιμοποιήσουμε το ίδιο όνομα πολλές φορές για οποιαδήποτε επιγραφή, σήμα ή μεταβλητή του προγράμματος.

3.6 Λογισμικό Quartus II

Το Quartus II είναι μια προγραμματιζόμενη λογική συσκευή για τον σχεδιασμού λογισμικού που παράγεται από την εταιρεία Altera. Το Quartus II επιτρέπει την ανάλυση και τη σύνθεση σχεδίων, εκτελέσει την ανάλυση χρονισμού, εξετάζει RTL διαγράμματα, προσομοιώνει και ρυθμίζει τη συσκευή προορισμού με τον προγραμματιστή. Στο Quartus προγραμματίζουμε σε γλώσσα περιγραφής υλικού, μπορεί να γίνει οπτική επεξεργασία των λογικών κυκλωμάτων και προσομοιώσεις σε διάνυσμα και κυματομορφή.

3.6.1 Περιβάλλον Quartus

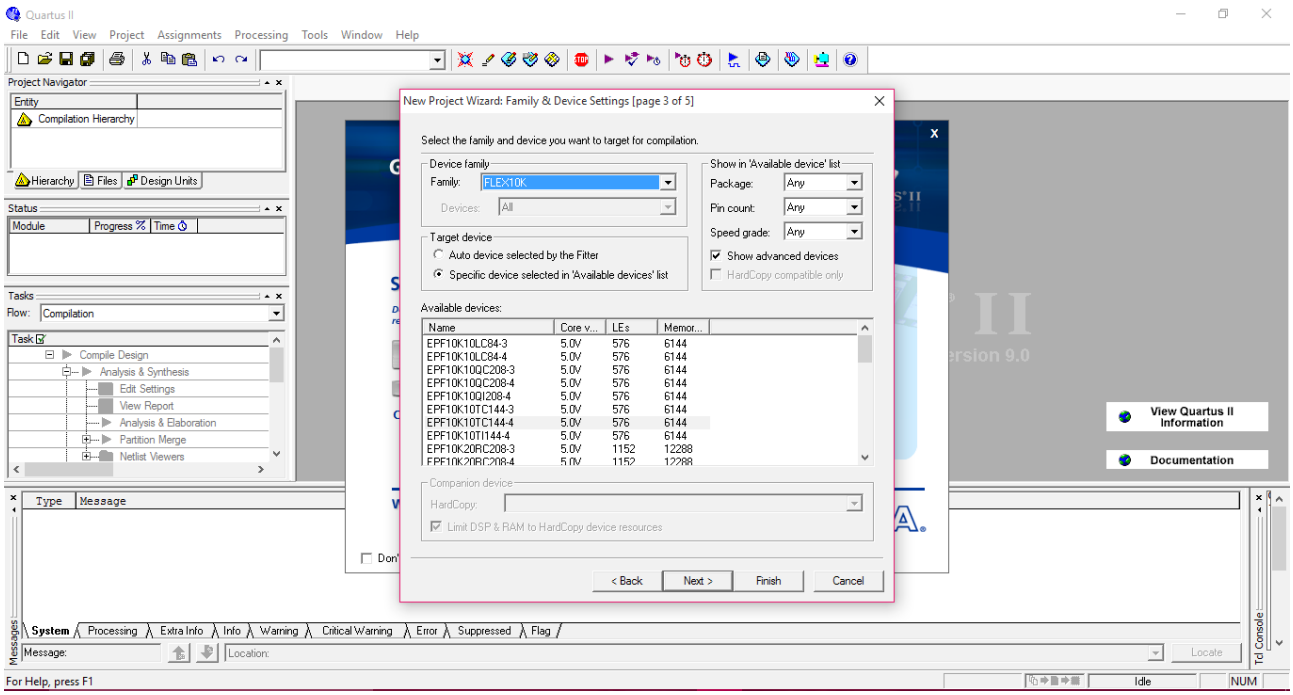
Ένα λογικό κύκλωμα ή υποκύκλωμα που σχεδιάζεται με το λογισμικό Quartus II ονομάζεται έργο (project). Το Quartus II εργάζεται σε ένα έργο κάθε φορά και κρατά τις πληροφορίες του σε ένα φάκελο του συστήματος αρχείων. Για να ξεκινήσουμε τη σχεδίαση ενός νέου έργου, το πρώτο βήμα που πρέπει να κάνουμε είναι να δημιουργήσουμε ένα φάκελο για να αποθηκεύσουμε εκεί τα αρχεία.



Σχήμα 3.6 Κεντρική οθόνη του προγράμματος Quartus II

Ξεκινώντας την εκτέλεση του προγράμματος Quartus II για τη δημιουργία ενός έργου θα πρέπει να δούμε ένα παράθυρο παρόμοιο με αυτό στο Σχήμα 3.6. Όπως φαίνεται, το κυρίως παράθυρο αποτελείται από πολλά παράθυρα που παρέχουν πρόσβαση σε όλα τα χαρακτηριστικά του λογισμικού μας. Επιλέγουμε πάνω αριστερά το μενού που ονομάζεται File, για να αρχίσουμε να εργαζόμαστε σε μια νέα σχεδίαση και στη συνέχεια επιλέγουμε NewProject wizard. Ορίζουμε το κατάλογο εργασίας, στο οποίο το έργο πρέπει να έχει ένα όνομα και στη συνέχεια δημιουργούμε

τον επιθυμητό κατάλογο. Στη συνέχεια επιλέγουμε το τύπο εξαρτήματος στο οποίο θα υλοποιηθεί το έργο όπως φαίνεται και στο Σχήμα 3.7.



Σχήμα 3.7 Καθορισμός της οικογένειας εξαρτημάτων

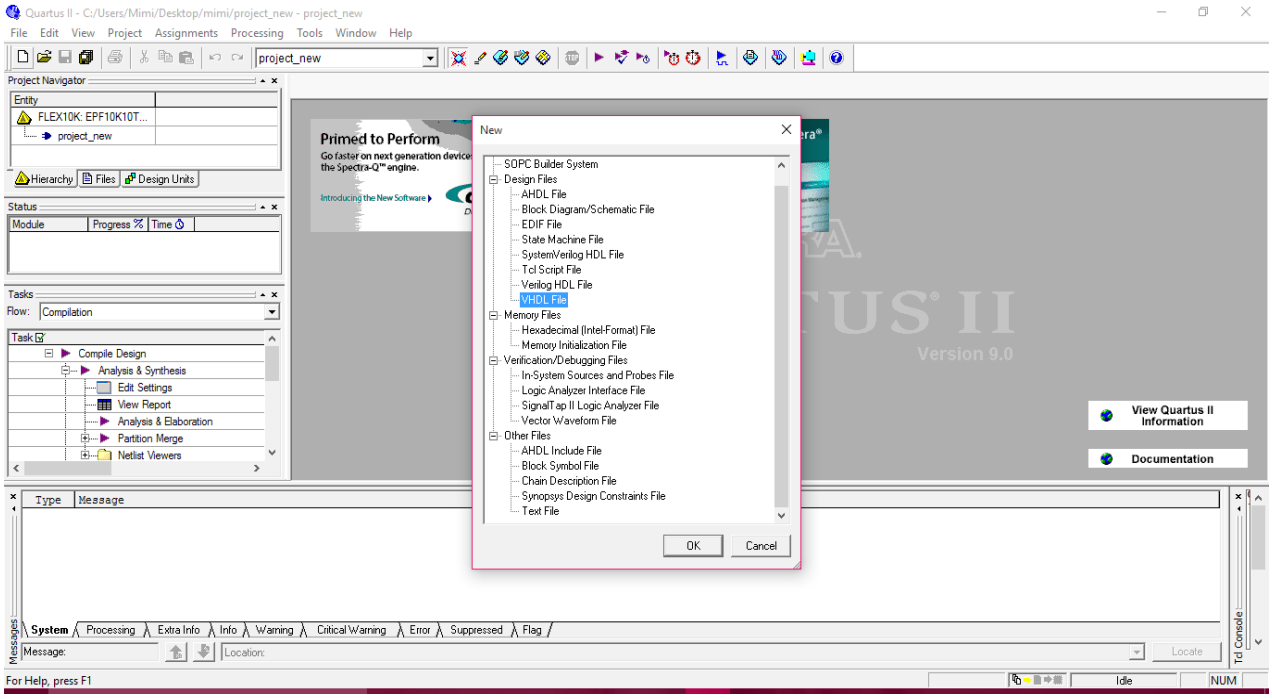
Μια ακόμα επιλογή που μας παρέχει το λογισμικό μας είναι η επιλογή για EDA (electronic design automation) εργαλεία. Τα EDA είναι εργαλεία που έχουν αναπτυχθεί και διατίθενται στο εμπόριο από άλλες εταιρείες εκτός της Altera. Εάν δε τα επιλέξουμε πατάμε την επιλογή Finish και επιστρέφουμε στο κύριο παράθυρο απεικόνισης του Quartus II.

3.6.2 Κώδικας και προσομοίωση

Για να δημιουργήσουμε ένα νέο έργο για τη σχεδίαση σε γλώσσα VHDL, το λογισμικό Quartus II περιέχει έναν επεξεργαστή κειμένων (text editor), όπως εικονίζεται και στο Σχήμα 3.8. Επιλέγουμε File → New για να παράγουμε το παράθυρο του Σχήματος 3.9 και επιλέγουμε VHDL File. Αυτή η ενέργεια μας δίνει πρόσβαση στον επεξεργαστή κειμένου.

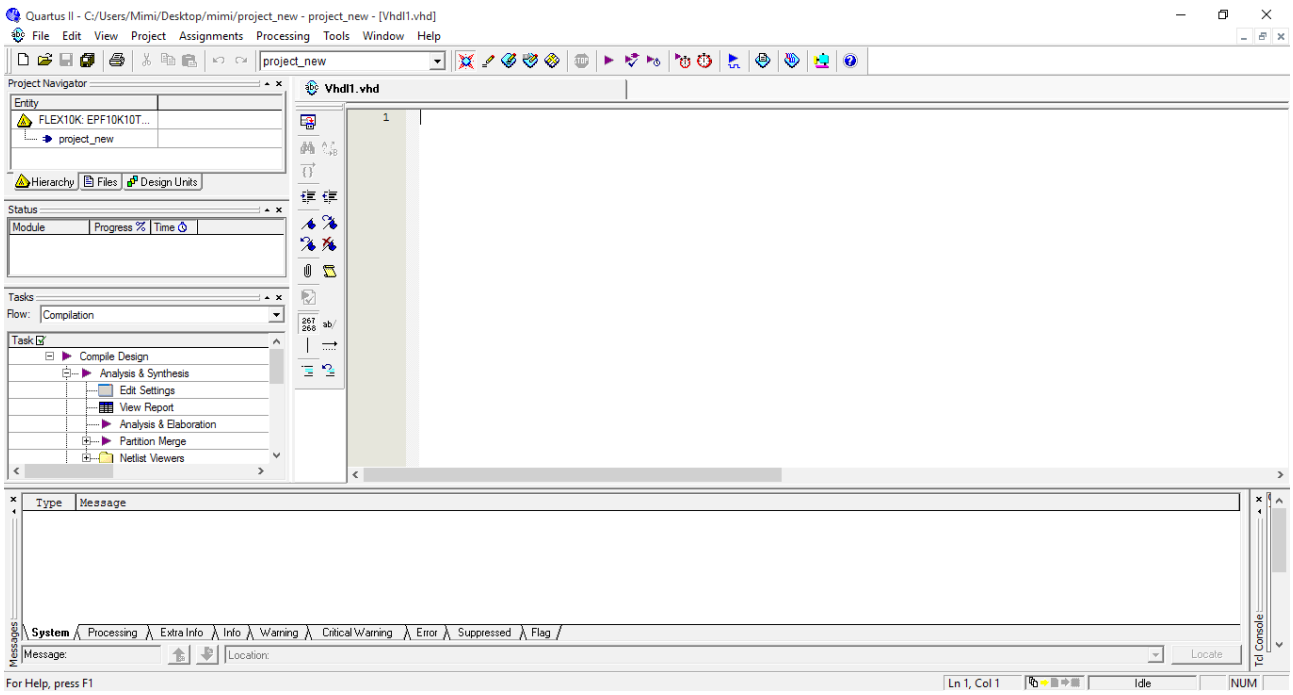
Το πρώτο βήμα είναι να καθορίσουμε ένα όνομα για το έργο. Επιλέγουμε File → Save και διαπιστώνουμε πως το αρχείο που δημιουργήσαμε έχει την επέκταση ονόματος vhd, που θα πρέπει να χρησιμοποιείται για όλα τα αρχεία που είναι σε γλώσσα VHDL.

Το κείμενο εισάγεται στο σημείο εισαγωγής (insertion point). Υπάρχουν δύο δυνατότητες του επεξεργαστή κειμένου που είναι αρκετά χρήσιμες για τη γραφή προγραμμάτων. Πρώτον, ο επεξεργαστής κειμένου μπορεί να απεικονίζει διαφορετικούς τύπους εντολών της γλώσσας VHDL με διαφορετικά χρώματα και δεύτερον, μπορεί αυτόματα να στοιχίσει το κείμενο σε μία νέα γραμμή, έτσι ώστε αυτή να βρίσκεται σε σύμπτωση με τη προηγούμενη γραμμή. Αυτές οι επιλογές μπορούν να ελέγχονται από τις ρυθμίσεις, οι οποίες βρίσκονται στο Tools → Options → Text Editor.



Σχήμα 3.8 Σύνθεση Κυκλώματος από Πρόγραμμα σε Γλώσσα VHDL

Είναι δύσκολο να μπορεί να θυμάται ο σχεδιαστής τη σύνταξη των προγραμμάτων σε γλώσσα VHDL. Γι' αυτό το λόγο ο επεξεργαστής κειμένου παρέχει ένα σύνολο Προτύπων της Γλώσσας VHDL. Τα πρότυπα αυτά έχουν παραδείγματα διαφόρων τύπων εντολών της γλώσσας VHDL, όπως είναι η δήλωση οντοτήτων, η αρχιτεκτονική και η εντολή αντιστοίχησης σήματος.



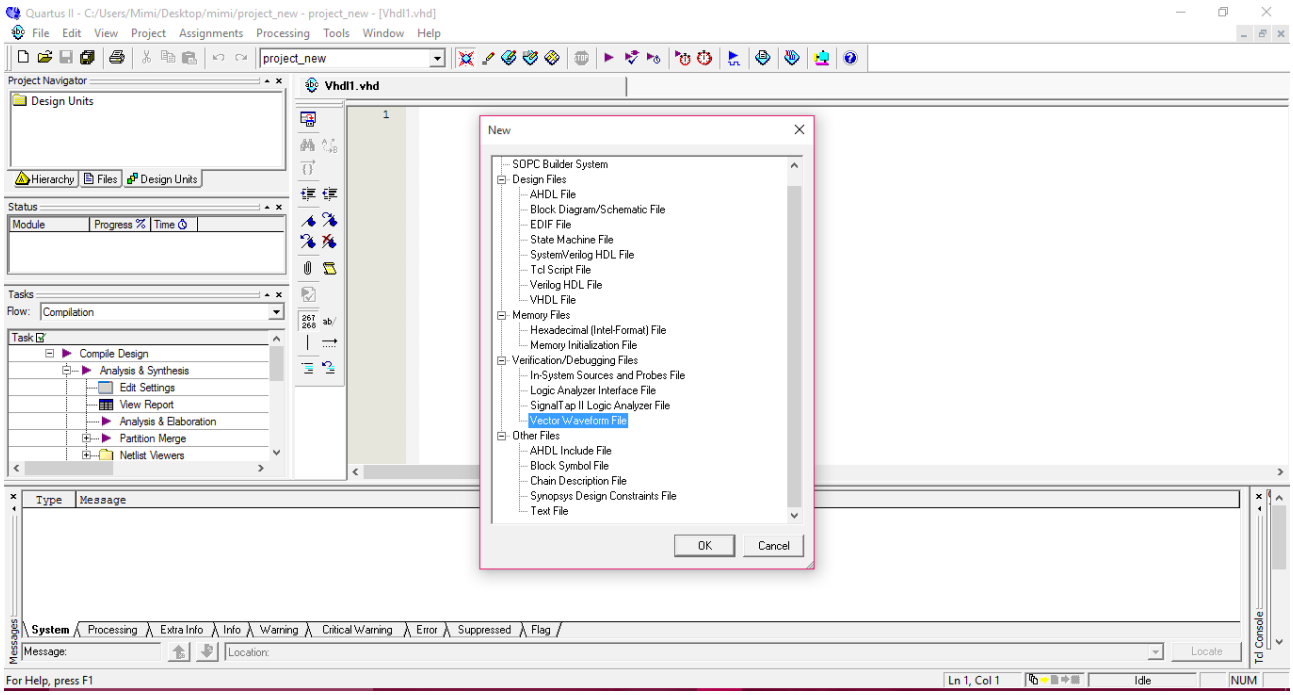
Σχήμα 3.9 Επεξεργαστής Κειμένων στο Quartus II

Για να γίνει η σύνθεση ενός κυκλώματος από πρόγραμμα σε γλώσσα VHDL, επιλέγουμε Processing → Start → Start Analysis and Synthesis. Αν το πρόγραμμα σε VHDL έχει γίνει σωστά, ο μεταγλωττιστής θα απεικονίσει ένα μήνυμα που λέει πως δεν προέκυψαν καθόλου λάθη.

Αν ο μεταγλωττιστής αναφέρει πως έγιναν κάποια λάθη, για κάθε λάθος θα εμφανιστεί ένα μήνυμα στο παράθυρο Messages. Κάνοντας διπλό κλικ σε κάποιο μήνυμα λάθους θα επισημάνει την προβληματική δήλωση στο πρόγραμμα στο παράθυρο του επεξεργαστή κειμένων.

Η λειτουργική προσομοίωση του προγράμματος σε γλώσσα VHDL γίνεται με το εργαλείο προσομοίωσης που περιέχει το λογισμικό Quartus II. Πριν το κύκλωμα μπορέσει να προσομοιωθεί, είναι απαραίτητο να δημιουργηθούν οι κατάλληλες κυματομορφές που ονομάζονται διανύσματα δοκιμής (test vectors) και αντιπροσωπεύουν τα σήματα εισόδου.

Ανοίγουμε το παράθυρο του επεξεργαστή κυματομορφών (Waveform Editor) επιλέγοντας File → New κάνουμε κλικ στη καρτέλα Other Files και επιλέγουμε Vector Waveform File όπως φαίνεται και στο Σχήμα 3.10. Αποθηκεύουμε το αρχείο και θέτουμε την επιθυμητή προσομοίωση να εκτελεστεί στο χρόνο που θέλουμε, επιλέγοντας Edit → End Time. Στον επεξεργαστή κυματομορφών επιλέγουμε View → Fit in Window για να απεικονίσουμε στο παράθυρο ολόκληρη τη περιοχή προσομοίωσης.



Σχήμα 3.10 Επιλογή για τη δημιουργία ενός αρχείου διανυσματικής δοκιμής

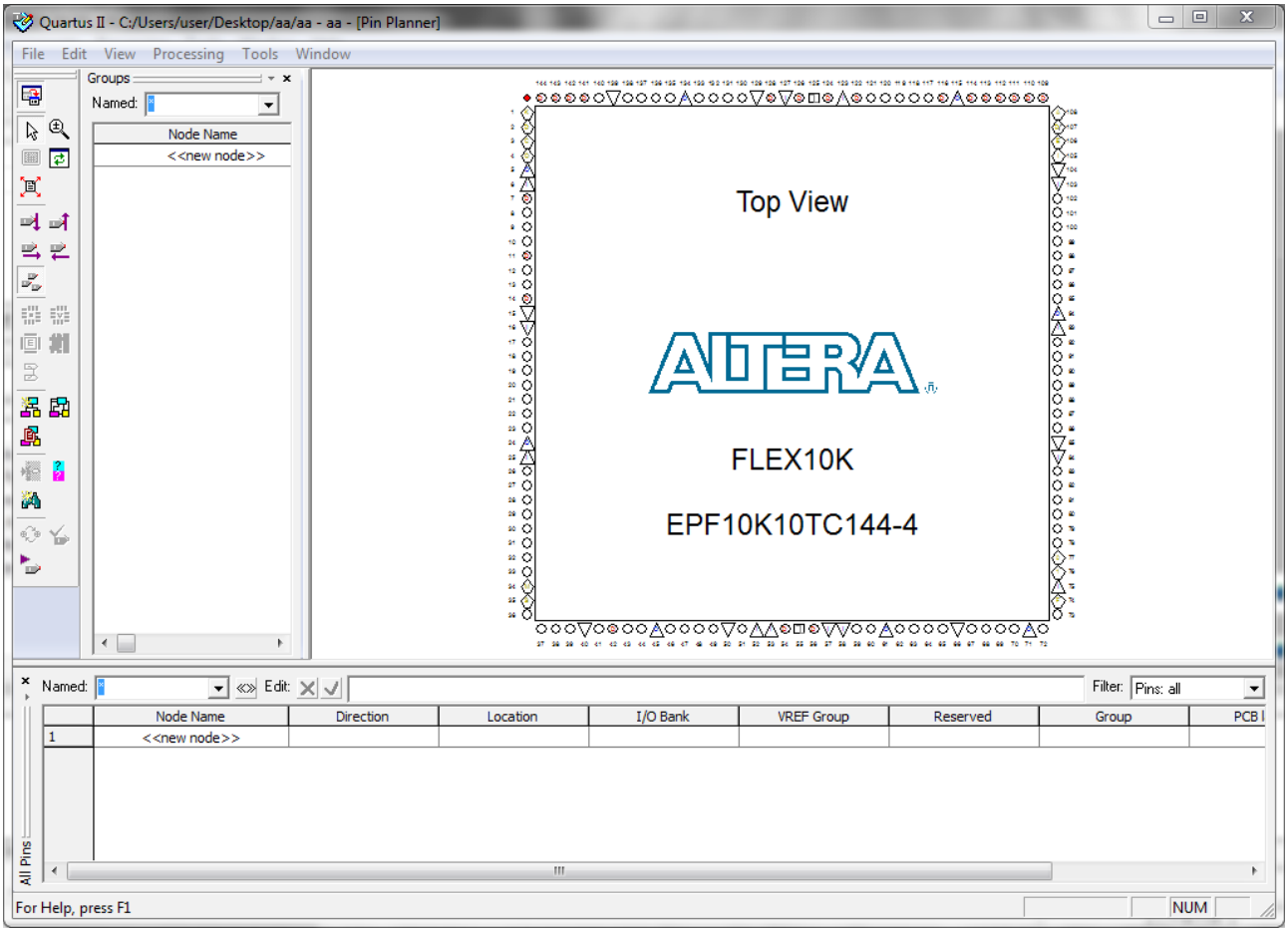
Στη συνέχεια θέλουμε να βάλουμε τους κόμβους εισόδου και εξόδου του κυκλώματος που θα προσομοιωθεί. Αυτό γίνεται με τη χρήση βοηθητικού προγράμματος Node Finder. Κάνουμε κλικ στην επιλογή Edit → Insert Node or Bus. Επειδή μας ενδιαφέρουν οι ακροδέκτες εισόδου και εξόδου θέτουμε Pins:all και κάνουμε κλικ στο κουμπί List για να βρούμε τους κόμβους εισόδου και εξόδου. Έπειτα θα προσδιορίσουμε τις λογικές τιμές που θα χρησιμοποιηθούν για τα σήματα εισόδου κατά τη διάρκεια της προσομοίωσης. Οι λογικές τιμές εξόδου θα δημιουργηθούν αυτόματα από την προσομοίωση.

Για τη προσομοίωση της λειτουργικής προσομοίωσης, επιλέγουμε Assignments → Settings, κάνουμε κλικ στον Simulator και επιλέγουμε Fuctional ως κατάσταση προσομοίωσης. Για να ολοκληρώσουμε τη ρύθμιση του προσομοιωτή επιλέγουμε την εντολή Proccessing → Generate Functional Simulation Netlist. Ο προσομοιωτής του Quartus II παίρνει τα διανύσματα δοκιμών

εισόδου και παράγει τις εξόδους στο αρχείο. Η εκτέλεση μιας προσομοίωσης ξεκινά με την επιλογή Processing → Start Simulation. Στο τέλος το Quartus δείχνει τα αποτελέσματα τις προσομοίωσης. Αφού έχει ολοκληρωθεί και η σχεδίαση επιλέγουμε File → Close Project και κλείνουμε το παρόν έργο.

3.6.3 Ορισμός ακροδεκτών (pin assignments)

Για να αντιστοιχίσουμε τις εισόδους και τις εξόδους του κυκλώματός μας με συγκεκριμένους ακροδέκτες ορίζουμε ότι η διάταξή μας είναι ένα FPGA που ανήκει στην οικογένεια FLEX10K της εταιρίας ALTERA και συγκεκριμένα η διάταξη EPF10K10TC144-4. Αυτή η διαδικασία γίνεται από το Menu Assignments/Device. Προκειμένου να ορίσουμε σε ποιους ακροδέκτες του τσιπ θα συνδεθούν οι είσοδοι και οι έξοδοι του κυκλώματος, κάνουνε τις εξής επιλογές: Επιλέγουμε Menu Assignments/Pins. Ανοίγει το παράθυρο του Σχήματος 3.11. Στο πεδίο Node Name εμφανίζονται τα ονόματα που έχουνε δώσει στους ακροδέκτες μας. Διπλοπατώντας πάνω στο πεδίο Location εμφανίζεται η αρίθμηση όλων των ακροδεκτών της διάταξης. Εκεί, κάνουνε τις επιλογές των εισόδων, εξόδων και τις αντιστοιχούμε στα pins της πλακέτας του FPGA.



Σχήμα 3.11 Παράθυρο ορισμού ακροδεκτών (pin assignments)

ΚΕΦΑΛΑΙΟ 4

Εισαγωγή στο FPGA

4.1 Λογικές Διατάξεις Πυλών Προγραμματιζόμενες στο Πεδίο (FPGA)

Η *επιτόπια συστοιχία προγραμματιζόμενων πυλών* (*FPGA* ή *Field Programmable Gate Array*) είναι τύπος προγραμματιζόμενου ολοκληρωμένου κυκλώματος γενικής χρήσης το οποίο διαθέτει πολύ μεγάλο αριθμό τυποποιημένων πυλών και άλλων ψηφιακών λειτουργιών όπως απαριθμητές, καταχωρητές μνήμης κ.α. Σε ορισμένα από αυτά ενσωματώνονται και αναλογικές λειτουργίες. Κατά τον προγραμματισμό του FPGA, ο οποίος γίνεται πάντοτε ενώ αυτό είναι τοποθετημένο στο τυπωμένο κύκλωμα, ενεργοποιούνται οι επιθυμητές λειτουργίες και διασυνδέονται μεταξύ τους έτσι ώστε το FPGA να συμπεριφέρεται ως ολοκληρωμένο κύκλωμα με συγκεκριμένη λειτουργία. Ο κώδικας με τον οποίο προγραμματίζεται το FPGA γράφεται σε γλώσσες περιγραφής υλικού (VHDL, AHDL, Verilog).

4.1.1 Χαρακτηριστικά FPGA

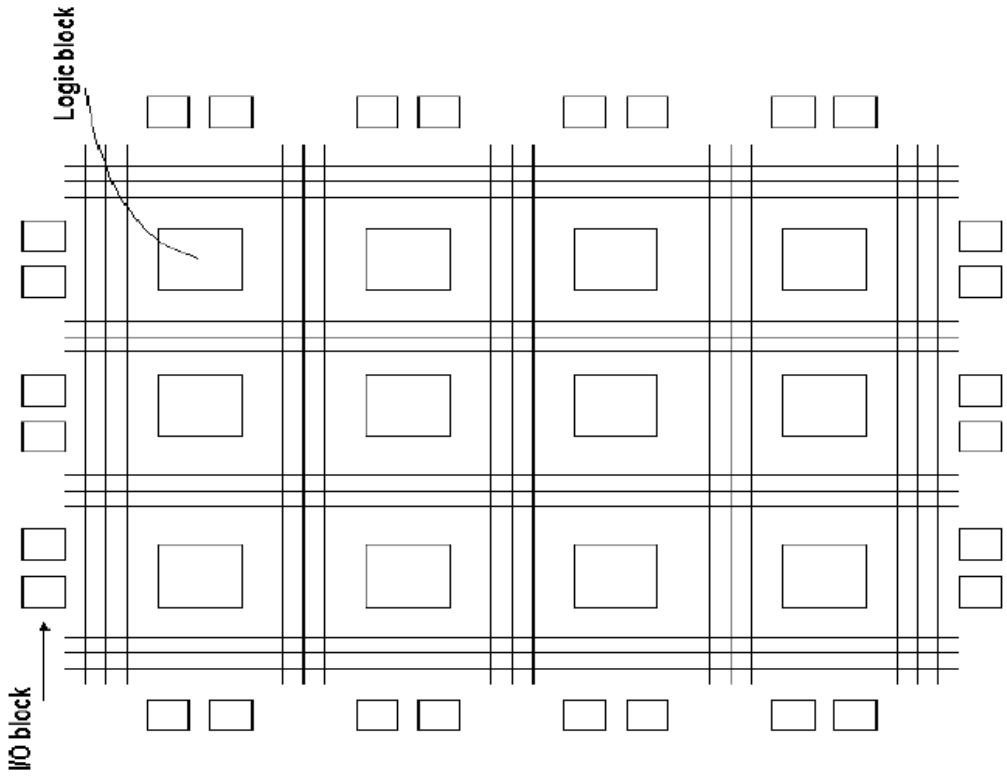
Τα ιδιαίτερα χαρακτηριστικά του FPGA είναι τα εξής:

- Το FPGA χάνει τον προγραμματισμό του κάθε φορά που διακόπτεται η τάση τροφοδοσίας του. Επομένως απαιτεί εξωτερικό μικροεπεξεργαστή ή μνήμη με μόνιμη συγκράτηση δεδομένων (non-volatile memory) από τα οποία θα προγραμματίζεται, κάθε φορά που επανέρχεται η τάση τροφοδοσίας.
- Ο προγραμματισμός του FPGA μπορεί να αλλάζει κάθε φορά που τροποποιείται το λογισμικό του μικροεπεξεργαστή ή τα δεδομένα της μνήμης που το ελέγχει.
- Δεν υπάρχει όριο στο πόσες φορές μπορεί να επαναπρογραμματιστεί.
- Η κατανάλωση ισχύος είναι σημαντικά αυξημένη, σε σχέση με τα ASIC.

Έτσι το FPGA είναι ιδιαίτερα κατάλληλο εκεί που οι παράμετροι λειτουργίας πρέπει να αλλάζουν συχνά ή σε μικρές ποσότητες παραγωγής. Βασική δομική μονάδα του FPGA είναι το λογικό μπλοκ, με τη χρήση του οποίου υλοποιούνται οι λογικές συναρτήσεις που εκφράζουν τη λειτουργία ενός ψηφιακού κυκλώματος. Ανάλογα με το μέγεθος του κυκλώματος πολλά λογικά μπλοκ συνδέονται για να υλοποιήσουν το πλήθος των απαραίτητων λογικών συναρτήσεων.

4.1.2 Γενική Δομή Διάταξης FPGA

Σύμφωνα με τα καινούρια πρότυπα για την υλοποίηση μεγάλων κυκλωμάτων είναι εύκολο να χρησιμοποιούμε ολοκληρωμένα κυκλώματα που έχουν μεγάλη χωρητικότητα, όπως οι λογικές διατάξεις προγραμματιζόμενου πεδίου. Η γενική μορφή FPGA φαίνεται στο Σχήμα 4.1. Η διάταξη αυτή περιέχει λογικές βαθμίδες, βαθμίδες εισόδου / εξόδου για τη σύνδεση με τους ακροδέκτες της συσκευασίας και διακόπτες και γραμμές εσωτερικής διασύνδεσης.

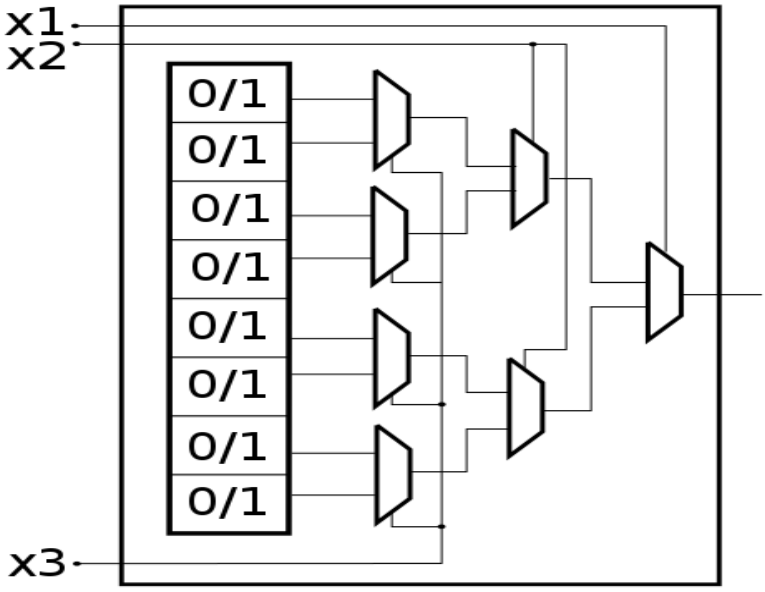


Σχήμα 4.1 Γενική Δομή μιας Διάταξης FPGA

Οι λογικές βαθμίδες οργανώνονται με τη μορφή δισδιάστατης σειράς. Επιπλέον οι γραμμές εσωτερικής διασύνδεσης οργανώνονται ως οριζόντια και κατακόρυφα κανάλια δρομολόγησης (routing channels) ανάμεσα στις γραμμές και στις στήλες των λογικών βαθμίδων. Αυτά τα κανάλια διαθέτουν στο εσωτερικό τους καλώδια και προγραμματιζόμενους διακόπτες. Όπως φαίνεται και στο Σχήμα 2.13 σε ένα FPGA τα τετράγωνα δίπλα από τις λογικές βαθμίδες περιέχουν διακόπτες που συνδέουν τους ακροδέκτες εισόδου και εξόδου των λογικών βαθμίδων με τα καλώδια διασύνδεσης, ενώ τα τετράγωνα που είναι διαγώνια των λογικών βαθμίδων συνδέουν ένα καλώδιο διασύνδεσης με ένα άλλο. Επιπλέον υπάρχουν προγραμματιζόμενες συνδέσεις ανάμεσα στις βαθμίδες εισόδου και εξόδου και τα καλώδια διασύνδεσης.

4.1.3 Λογικές Βαθμίδες FPGA

Οι λογικές βαθμίδες ενός FPGA έχουν έναν μικρό αριθμό εισόδων και μια έξοδο. Η πιο γνωστή βαθμίδα που υπάρχει στο εμπόριο είναι ο *πίνακας αναζήτησης ή αναφοράς (Lookup table ή LUT),* ο οποίος έχει *κυψέλες αποθήκευσης (storage cells)* που χρησιμοποιούνται για την υλοποίηση μιας μικρής λογικής συνάρτησης. Η κάθε κυψέλη αποθήκευσης μπορεί να περιέχει μια λογική τιμή, δηλαδή ή 0 ή 1. Αφού αποθηκευτεί η τιμή θα κατευθυνθεί στην έξοδο της κυψέλης αποθήκευσης. Στο Σχήμα 4.2 φαίνεται ένας πίνακας LUT τριών εισόδων. Ο πίνακας μας περιέχει οκτώ κυψέλες αποθήκευσης, επειδή ο πίνακας αληθείας με τρεις μεταβλητές περιέχει οκτώ γραμμές. Εάν ένας πίνακας LUT έχει τέσσερις εισόδους τότε θα έχει 16 κυψέλες αποθήκευσης. Εάν έχει πέντε εισόδους, τότε θα έχει 32 κυψέλες αποθήκευσης.



Σχήμα 4.2 Πίνακας LUT τριών εισόδων

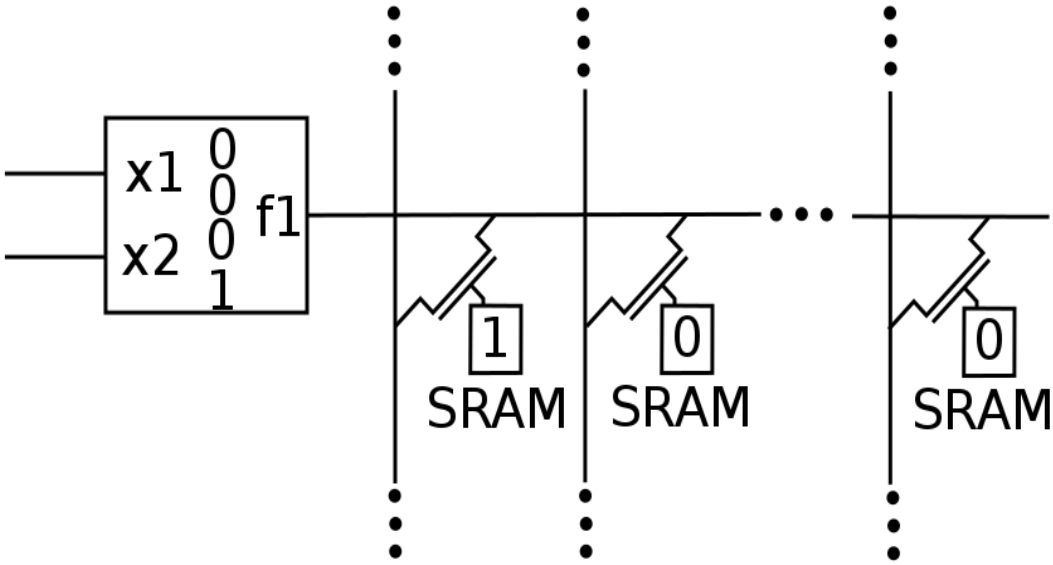
Όταν ένα κύκλωμα γίνει με FPGA, οι λογικές βαθμίδες προγραμματίζονται για να πραγματοποιήσουν τις αναγκαίες συναρτήσεις και τα κανάλια δρομολόγησης προγραμματίζονται για να κάνουν τις αναγκαίες διασυνδέσεις ανάμεσα στις λογικές βαθμίδες. Οι κυψέλες αποθήκευσης LUT είναι *πτητικές ή μη μόνιμες (Volatile)*. Αυτό σημαίνει ότι τα δεδομένα που περιέχουν οι κυψέλες χάνονται εάν διακοπεί η τροφοδοσία του κυκλώματος. Για αυτόν το λόγο το FPGA πρέπει να προγραμματίζεται κάθε φορά που εφαρμόζεται τροφοδοσία στο κύκλωμα. Για να αντιμετωπιστεί αυτό το πρόβλημα στη μητρική πλακέτα μαζί με το FPGA, υπάρχει ένα μικρό ολοκληρωμένο κύκλωμα μνήμης που ονομάζεται *προγραμματιζόμενη μνήμη μόνο ανάγνωσης (programmable read-only memory ή PROM)* και μπορεί να διατηρεί μόνιμα τα δεδομένα διάταξης. Έτσι όταν εφαρμόζεται τροφοδοσία στο κύκλωμα οι κυψέλες αποθήκευσης φορτώνονται αυτόματα από τη PROM μνήμη.

4.1.4 Προγραμματισμός και Ρύθμιση της Διάταξης του FPGA

Όταν ένα κύκλωμα έχει μεταγλωττιστεί τότε μπορεί να μεταφερθεί στο ολοκληρωμένο κύκλωμα FPGA της *πλακέτας DE2*. Αυτή η πλακέτα έχει μια λειτουργία η οποία είναι γνωστή ως *προγραμματισμός JTAG (Joint Test Action Group ή Ομάδα Κοινής Δράσης Ελέγχου)*. Το JTAG καθορίζει με έναν απλό τρόπο τον έλεγχο των ψηφιακών κυκλωμάτων και τη φόρτωση δεδομένων και αποτέλεσε ένα πρότυπο της ieee. Τα δεδομένα των ρυθμίσεων θα μεταφερθούν από τον υπολογιστή στη πλακέτα μέσω ενός καλωδίου. Για να μπορέσει να ολοκληρωθεί αυτή η σύνδεση είναι απαραίτητο να γίνει η εγκατάσταση ενός λογισμικού οδηγού USB-Blaster της Altera. Όταν λειτουργεί η JTAG τα δεδομένα των ρυθμίσεων φορτώνονται απευθείας στο FPGA. Εάν το FPGA είναι ρυθμισμένο με αυτόν τον τρόπο τότε οι πληροφορίες για τις ρυθμίσεις θα χαθούν όταν διακοπεί η τροφοδοσία.

4.1.5 Διακόπτες Διασύνδεσης σε FPGA

Στο FPGA η πληροφορία προγραμματισμού αποθηκεύεται σε στοιχεία μνήμης, τα οποία είναι μέλος μιας *στατικής μνήμης τυχαίας προσπέλασης (static random access memory ή SRAM)*. Κάθε στοιχείο μπορεί να αποθηκεύει ένα λογικό 0 ή 1 και παρέχει τη αποθηκευμένη τιμή στην έξοδο. Επιπλέον τα στοιχεία της SRAM χρησιμοποιούνται για να οργανώνουν τα καλώδια σύνδεσης στα FPGAs. Στο Σχήμα 4.3 φαίνεται ένα μικρό τμήμα του FPGA. Η λογική βαθμίδα δημιουργεί την έξοδο f1, η οποία οδηγείται στο οριζόντιο καλώδιο που αυτό μπορεί να συνδεθεί με κάποια κατακόρυφα καλώδια, με τα οποία διασταυρώνεται με τη βοήθεια προγραμματιζόμενων διακοπτών.



Σχήμα 4.3 Διακόπτες Διαβίβασης προς Τρανζίστορ σε FPGA

Ο κάθε διακόπτης φτιάχνεται με ένα τρανζίστορ τεχνολογίας NMOS, του οποίου ο ακροδέκτης πύλης ελέγχεται από ένα στοιχείο μνήμης SRAM. Αυτός ο διακόπτης είναι γνωστός ως *διακόπτης διαβίβασης προς τρανζίστορ (pass- transistor switch)*. Όταν ένα στοιχείο έχει τη τιμή 0, τότε το ελεγχόμενο από αυτό τρανζίστορ NMOS είναι απενεργοποιημένο, όταν όμως το στοιχείο έχει τη τιμή 1, το τρανζίστορ NMOS ενεργοποιείται. Ο διακόπτης σχηματίζει μία σύνδεση ανάμεσα στα δύο καλώδια που συνδέονται στους ακροδέκτες πηγής και απαγωγού. Ο αριθμός διακοπτών που έχει ένα FPGA εξαρτάται από την αρχιτεκτονική του ολοκληρωμένου κυκλώματος. Σε μερικά FPGAs οι διακόπτες δεν υλοποιούνται με τρανζίστορ διαβίβασης αλλά με τη βοήθεια απομονωτών τριών καταστάσεων.

4.2 Αναπτυξιακό κύκλωμα LEAP LP-2900

Η πλακέτα που θα χρησιμοποιήσουμε για την υλοποίηση των εφαρμογών μας είναι η LP-2900 της εταιρίας Leap Electronic Co. Η πλακέτα αυτή βασίζεται στο ολοκληρωμένο FPGA EPF10K10TC144-4 της εταιρίας Altera. Το FPGA έχει 144 Pins εισόδου/εξόδου, αποτελείται από 10.000 λογικές πύλες, οργανωμένες σε 576 λογικά στοιχεία, 720 flips-flops και 6.144 ψηφία μνήμης.

Το LP-2900 χωρίζεται σε τέσσερα μέρη. Στο μέρος που περιέχεται το FPGA, στις συσκευές εισόδου/εξόδου, στο μέρος της τροφοδοσίας και στη θύρα επικοινωνίας με τον ηλεκτρονικό υπολογιστή.

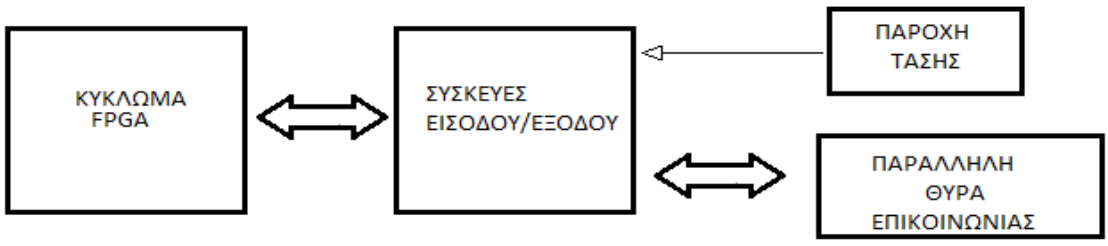
Οι συσκευές εισόδου/εξόδου
4 σετ κόκκινα, κίτρινα και πράσινα leds κοινής καθόδου
Ενδείκτες επτά τομέων κοινής ανόδου
1 buzzer
2 ηλεκτρονικά ζάρια κοινής καθόδου
8 διακόπτες τύπου push button
2x8 διακόπτες τύπου dip switches
4 διακόπτες παλμών
1 8x8 dont matrix display
1 πληκτρολόγιο 4x3
1 οθόνη LCD
Ένας μετατροπέας σήματος A/D και D/A
1 ολοκληρωμένο κύκλωμα 8051

Πίνακας 4.1 Συσκευές εισόδου/εξόδου

Στο υποκύκλωμα που είναι τοποθετημένο το FPGA υπάρχει και μία μνήμη EPROM, ένας διακόπτης reset και 144 μικρά leds που είναι συνδεδεμένα στους ακροδέκτες του FPGA, για να μπορούμε να δούμε ποιοι από αυτούς χρησιμοποιούνται αλλά και αν η τιμή τους είναι '0' ή '1'.

Η επικοινωνία με τον υπολογιστή γίνεται μέσω της παράλληλης θύρας του υπολογιστή. Για την σύνδεση του υπολογιστή με την αναπτυξιακή πλακέτα χρειαζόμαστε ένα καλώδιο που το ένα άκρο του το συνδέουμε στην παράλληλη θύρα του υπολογιστή και το άλλο στη θύρα του αναπτυξιακού.

Το σχεδιαστικό πρόγραμμα με το οποίο συνεργάζεται το LP-2900 είναι το Quartus II. Μέσω του Quartus II θα δημιουργηθούν τα προγραμματιστικά αρχεία του κάθε κυκλώματος ώστε να παραχθούν τα κατάλληλα σήματα που θα διαμορφώσουν τη διάταξη. Στο παρακάτω Σχήμα 4.4 φαίνεται το διάγραμμα βαθμίδων του LP-2900.



Σχήμα 4.4 Σχηματικό Διάγραμμα του LP-2900

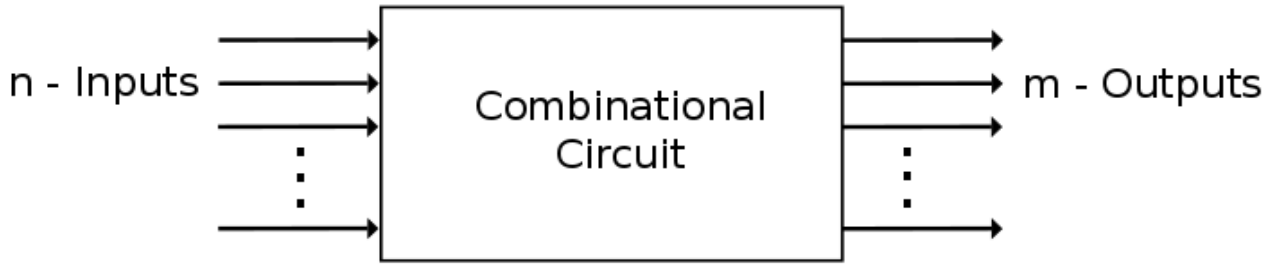
ΚΕΦΑΛΑΙΟ 5

Σχεδίαση ελεγκτών για την διασύνδεση μετατροπών A/D και D/A σε διάταξη FPGA

5.1 Σχεδιασμός Κυκλωμάτων

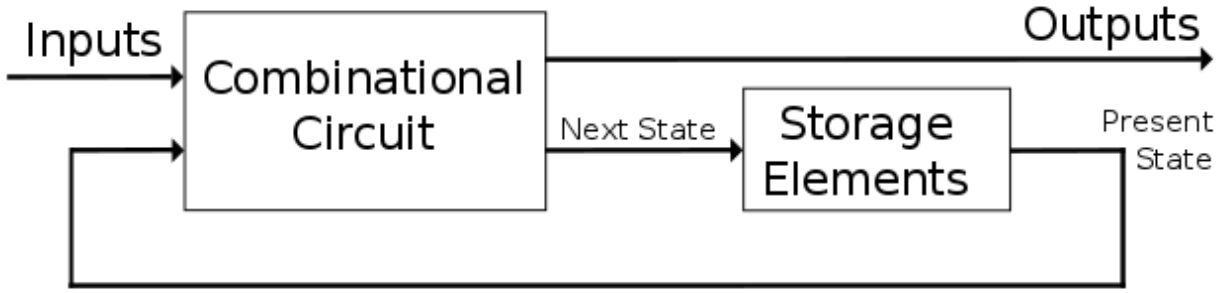
Ο σχεδιασμός των κυκλωμάτων διαχωρίζονται σε δυο κατηγορίες, τα συνδυαστικά λογικά και τα ακολουθιακά κυκλώματα.

Τα *συνδυαστικά (combinational) λογικά κυκλώματα* αποτελούνται από τις λογικές πύλες και υλοποιούν μια ή περισσότερες συναρτήσεις της άλγεβρας Boole. Δεν έχουν μνήμη αλλά έχουν έναν αριθμό εισόδων και μια ή περισσότερες εξόδους. Οι τιμές των εξόδων στα συνδυαστικά κυκλώματα εξαρτώνται μόνον από τις τρέχουσες, παρούσες τιμές των εισόδων και όχι από το παρελθόν.



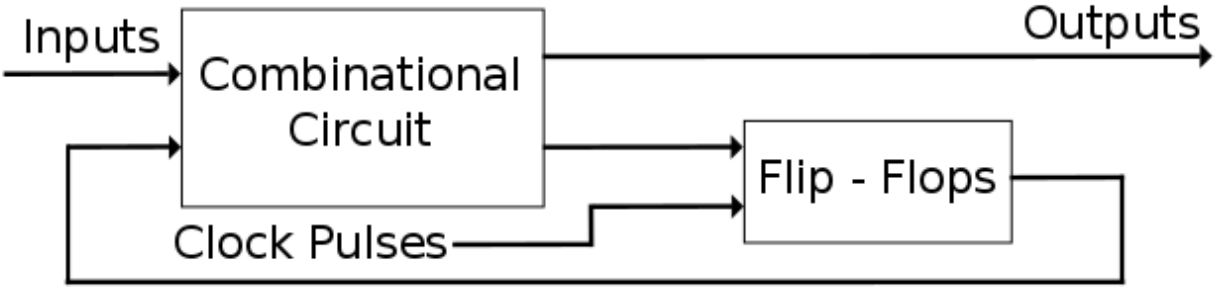
Σχήμα 5.1 Συνδυαστικό (combinational) λογικό κυκλώματα

Τα *ακολουθιακά (sequential)* κυκλώματα από την άλλη μεριά περιέχουν στοιχεία μνήμης, ώστε οι τιμές των εξόδων τους να εξαρτώνται από προηγούμενες τιμές των εισόδων, καθώς και τις τρέχουσες τιμές των εισόδων, στην κατάσταση (state). Η συμπεριφορά των ακολουθιακών κυκλωμάτων καθορίζεται από τη χρονική ακολουθία των καταστάσεων από τις οποίες περνούν διαδοχικά.



Σχήμα 5.2 Ακολουθιακά (sequential) κυκλώματα

Στις περισσότερες περιπτώσεις υπάρχει μία γεννήτρια κύριου ρολογιού (master clock generator) που ελέγχει τη λειτουργία ενός ακολουθιακού κυκλώματος και τροφοδοτεί το κύκλωμα με παλμούς που διανέμονται παντού στο κύκλωμα ώστε να επιτευχθεί ο συγχρονισμός (synchronization), ένα τέτοιο κύκλωμα λέγεται *σύγχρονο ακολουθιακό κύκλωμα (synchronous sequential circuit).* Το σήμα αυτό είναι ένα περιοδικό σήμα που αποτελείται από παλμούς. Αλλαγές μπορούν να προκύψουν κατά το θετικό ή αρνητικό μέτωπο των παλμών αυτού του σήματος, λέμε δηλαδή ότι λειτουργούν με παλμικό τρόπο (pulse mode). Τα σύγχρονα ακολουθιακά κυκλώματα υλοποιούνται με την βοήθεια συνδυαστικής λογικής και ενός ή περισσοτέρων Flip-Flops, τα οποία μπορούν να διατηρηθούν σε μια κατάσταση έως ότου κάποιο σήμα εισόδου τα κάνει να αλλάξουν κατάσταση.



Σχήμα 5.3α Σχηματικό Διάγραμμα

Σχήμα 5.3β Χρονικό Διάγραμμα Παλμών

Η άλλη περίπτωση, στην οποία δεν χρησιμοποιείται ωρολογιακό σήμα, ονομάζεται *ασύγχρονο ακολουθιακό κύκλωμα (asynchronous sequential circuit)*. Τα ασύγχρονα ακολουθιακά κυκλώματα δε λειτουργούν με παλμικό τρόπο και δε χρησιμοποιούν Flip-Flops για να παριστούν τις μεταβλητές κατάστασης. Αντίθετα, οι αλλαγές κατάστασης εξαρτώνται από την τιμή 0 ή 1, των εισόδων του κυκλώματος για κάθε χρονική στιγμή. Για να επιτευχθεί αξιόπιστη λειτουργία, οι είσοδοι του κυκλώματος πρέπει να μεταβάλλονται μία φορά κάθε φορά. Επιπρόσθετα , πρέπει να μεσολαβεί ικανό χρονικό διάστημα ανάμεσα στις αλλαγές των σημάτων εισόδου, ώστε να προλαβαίνει το κύκλωμα να φθάσει σε κάποια σταθερή κατάσταση, η οποία επιτυγχάνεται όταν παύουν να μεταβάλλονται όλα τα εσωτερικά σήματα. Ένα κύκλωμα που συμμορφώνεται με αυτούς τους περιορισμούς λέγεται ότι λειτουργεί με το θεμελιώδη τρόπο (fundamental mode).

Στη περίπτωση των δικών μας ολοκληρωμένων κυκλωμάτων, δηλαδή του ADC0804 και του DACES522110, αναφερόμαστε σε ακολουθιακά κυκλώματα.

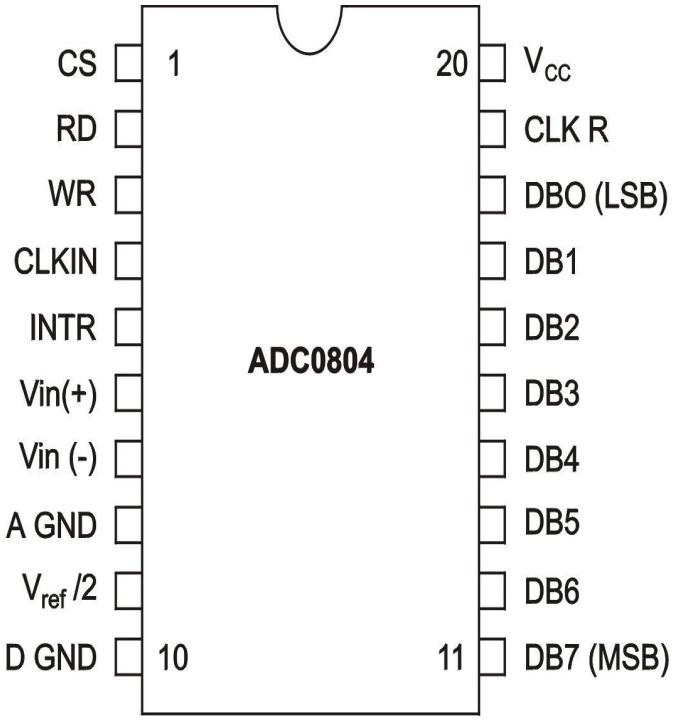
5.2 Ιεραρχική δομή σχεδίασης για τον ADC0804

Σε αυτό το κεφάλαιο θα δούμε την ιεραρχική δομή για την διασύνδεση του μετατροπέα ADC0804 σε διάταξη FPGA. Τα επίπεδα της υλοποίησης της πτυχιακής είναι: η λειτουργική περιγραφή και το διάγραμμα χρονισμού για τον ADC0804, το διάγραμμα καταστάσεων, η σχεδίαση του ελεγκτή, η προσομοίωση και τέλος η αντιστοίχηση των ακροδεκτών για τον ADC0804.

5.2.1 Λειτουργική Περιγραφή και Διάγραμμα Χρονισμού (ADC0804)

Το κύκλωμα ADC0804 στους ακροδέκτες Vin(+) και Vin(-) δηλαδή στον ακροδέκτη 6 και ακροδέκτη 7 δέχεται μια διαφορά δυναμικού και παράγει μια ψηφιακή έξοδο στους ακροδέκτες 11 έως 8. Ο χρονισμός επιτυγχάνεται με ένα δικτύωμα RC το οποίο είναι ανάμεσα στους ακροδέκτες 4 και 19. Το λιγότερο σημαντικό ψηφίο είναι στον ακροδέκτη 18 (LSB). Οι ακροδέκτες 1, 2, 3 και 5 έχουν τελική κατεύθυνση τον έλεγχο του μετατροπέα από εξωτερικές συσκευές. Ωστόσο μπορούν να συνδεθούν σε κατάλληλες λογικές στάθμες για να λειτουργεί το κύκλωμα χωρίς κάποια εξωτερικό έλεγχο . Τάση τροφοδοσίας δέχεται το κύκλωμα στον ακροδέκτη 20.

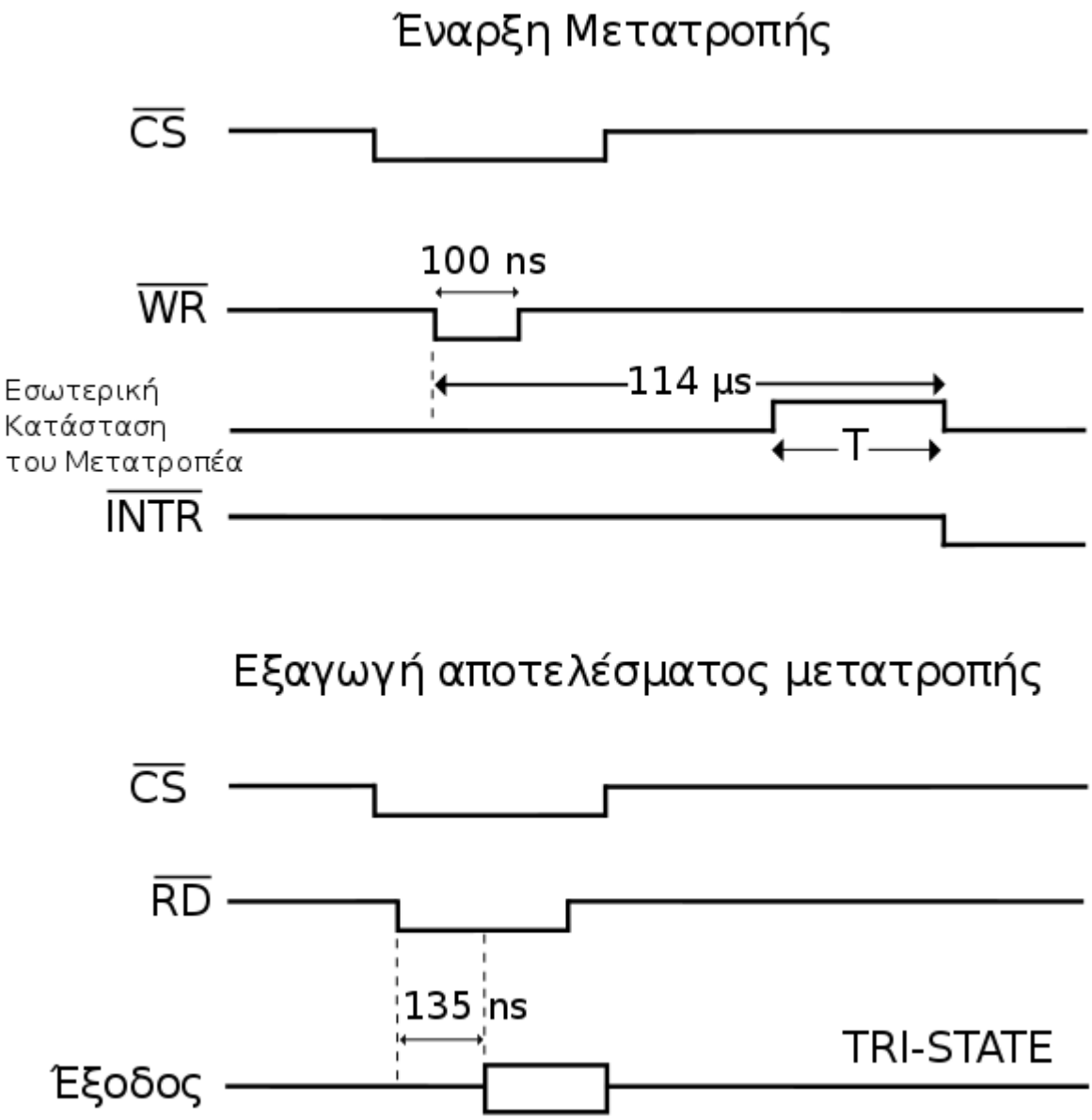
Οι ακροδέκτες CS (Chip Select), RD (Read), WR (Write) και INTR (Interrupt) ενεργοποιούνται με λογικό μηδέν και χρησιμεύουν για τον έλεγχο της μετατροπής αναλογικού σήματος σε ψηφιακό. Για να ενεργοποιήσουμε μια μετατροπή θα επιλέξουμε και θα οδηγήσουμε τον ακροδέκτη 1 (CS) σε λογικό μηδέν.



Σχήμα 5.4 Διάγραμμα ακροδεκτών του ολοκληρωμένου κυκλώματος ADC0804

Στη συνέχεια όσο ο ακροδέκτης 1 (CS) είναι σε λογικό μηδέν, οδηγούμε για λίγο και τον ακροδέκτη 3 (WR) σε λογικό μηδέν δηλαδή θα γίνει παλμός reset. Με αυτή τη δραστηριότητα έχουμε σαν αποτέλεσμα την επανεκκίνηση των καταχωρητών του μετατροπέα. Με την αρχή αυτής της δραστηριότητας ο ακροδέκτης INTR εξάγει λογικό 1. Κατόπιν, θα πάμε τον ακροδέκτη 3 (WR) σε λογικό 1. Η μετάβαση αυτή σκανδαλίζει μια μετατροπή της τάσης εισόδου. Με αυτή τη διαδικασία θα ξεκινήσει μια μετατροπή μέσα σε χρόνο ίσο με μία έως οκτώ το πολύ περιόδους του ωρολογιακού σήματος. Όταν ολοκληρωθεί η μετατροπή, ο ακροδέκτης INTR θα μεταβεί από λογικό 1 σε λογικό 0, δίνοντας έτσι τέλος στη μετατροπή.

Το αποτέλεσμα της μετατροπής για να οδηγηθεί στην έξοδο, δηλαδή στους ακροδέκτες DB0 έως DB7, θα πρέπει να κάνει μια λειτουργία ανάγνωσης, με την οποία η έξοδος του SAR μεταφέρεται στις εξόδους του απομονωτή τριών καταστάσεων (tri-state-buffer). Για να γίνει αυτό, θα χρειαστεί ο ακροδέκτης CS να μεταβεί σε λογικό 0 και στη συνέχεια ο ακροδέκτης RD πρέπει να γίνει λογικό μηδέν. Τα αποτελέσματα θα εμφανιστούν στην έξοδο. Παρακάτω στο Σχήμα 5.5 υπάρχει το διάγραμμα χρονισμού, το οποίο δείχνει με παλμούς αναλογικής τάσης και την εξαγωγή του αποτελέσματος στον μετατροπέα ADC0804.



Σχήμα 5.5 Διάγραμμα Χρονισμού για την Έναρξη και Εξαγωγή της Μετατροπής στον ADC0804

Για το παραπάνω σχήμα λαμβάνουμε κάποιες τυπικές τιμές από τον κατασκευαστή όπως:

- Η συχνότητα ρολογιού, Clock Frequency, να έχει τυπική τιμή f_{CLK} = 640kHz.
- Το Clock Periods ανά μετατροπή να έχει μέγιστη τιμή t_{CONV} = 73 Clocks/Conv.
- Με ονομαστικό ρυθμό μετατροπής, Rate In Free-Running Mode 8888conv/s.
- Ελάχιστο πλάτος παλμού WR, $t_{W(WR)I}$ = 100 ns.
- Ο τυπικός χρόνος προσπέλασης, Access Time, από την στιγμή που το RD θα γίνει μηδέν μέχρι την έξοδο των δεδομένων να είναι 135ns.
- Three-State Control χρόνος καθυστέρησης από την στιγμή που το RD θα γίνει ένα μέχρι την HI-Z κατάσταση, $t_{1H,}$ t_{0H} είναι 125 ns.
- Καθυστέρηση από την στιγμή που το WR γίνει μηδέν μέχρι την επόμενη επανεκκίνηση του RESET, t_{WI}, t_{RI} είναι 300 ns.

Παρακάτω θα ξανά αναφερθούμε στο ίδιο διάγραμμα χρονισμού του ADC με περαιτέρω στοιχεία στο σχήμα με βάση το διάγραμμα καταστάσεων.

Και στο επόμενο κεφάλαιο θα δούμε αναφορικά και τον πίνακα προδιαγραφών χρονισμού του ADC.

5.2.2 Διαγράμματα Καταστάσεων για τον ADC0804

Η κατάσταση (state) είναι ένα σύνολο τιμών των πεδίων που περιγράφουν την κατάσταση ενός αντικειμένου μια δεδομένη χρονική στιγμή. Άρα τα *διαγράμματα καταστάσεων (statechart diagram)* είναι διαγράμματα που προδιαγράφουν την δυναμική συμπεριφορά ενός αντικειμένου, μίας κλάσης και μπορούν να χρησιμοποιηθούν για την περιγραφή των υποσυστημάτων. Δεν χρειάζεται να περιγράφουμε όλες τις κλάσεις με διαγράμματα καταστάσεων, γιατί τα διαγράμματα καταστάσεων μπορεί να γίνουν πολύ πολύπλοκα. Μας δείχνουν όλες τις πιθανές καταστάσεις στις οποίες μπορεί να βρεθεί ένα αντικείμενο και πως το αντικείμενο αλλάζει καταστάσεις (state transition) ανάλογα με τα μηνύματα που λαμβάνει. Περιγράφει με λεπτομερή τρόπο τη συμπεριφορά ενός αντικειμένου.

Τα διαγράμματα καταστάσεων βοηθούν στην ανάπτυξη ενός συστήματος, και στην κατανόηση πολύπλοκων χαρακτηριστικών ή ροών εργασίας εξειδικευμένων περιοχών του συστήματος.

Τα διαγράμματα καταστάσεων είναι μια γραφική παράσταση της λειτουργίας του κυκλώματος, στο όποιο οι καταστάσεις του κυκλώματος παριστάνονται με κύκλους και οι μεταβάσεις από κατάσταση σε κατάσταση με βέλη. Κάθε βέλος σηματοδοτεί και την αντίστοιχη είσοδο ή έξοδο σε κάθε κατάσταση, ανάλογα με την κατεύθυνση του βέλους. Ένα βέλος που ξεκινά και καταλήγει στην ίδια κατάσταση σηματοδοτεί έναν βρόγχο.

Στο Σχήμα 5.6 απεικονίζεται το διάγραμμα καταστάσεων του ADC 0804, το οποίο απαρτίζεται από δέκα καταστάσεις. Σε κάθε κατάσταση ορίζουμε κάθε φορά τις τιμές των σημάτων εισόδου και εξόδου.

Πρώτη είναι η κατάσταση **start**, όπου είναι και η αρχική. Ορίζουμε τα τέσσερα σήματα εξόδων. Βάζουμε στο chip select 1 (cs1) την λογική τιμή 1, γιατί θέλουμε το chip του adc να είναι ανενεργό. Στο ready1 δίνουμε την τιμή 0, θα μείνει σε αυτή την κατάσταση έως ότου ολοκληρωθεί η μετατροπή και εξάγουμε το αποτέλεσμα στους ακροδέκτες, τότε μόνο θα αλλάξει κατάσταση και θα πάρει την τιμή 1. Στο write 1 (wr1) βάζουμε την τιμή 1 γιατί δεν θέλουμε ακόμα να μπει σε λειτουργία η εγγραφή όπως και στο σήμα read 1 (rd1) επίσης βάζουμε την τιμή 1 όπου δεν θέλουμε να μπει σε λειτουργία ανάγνωσης.

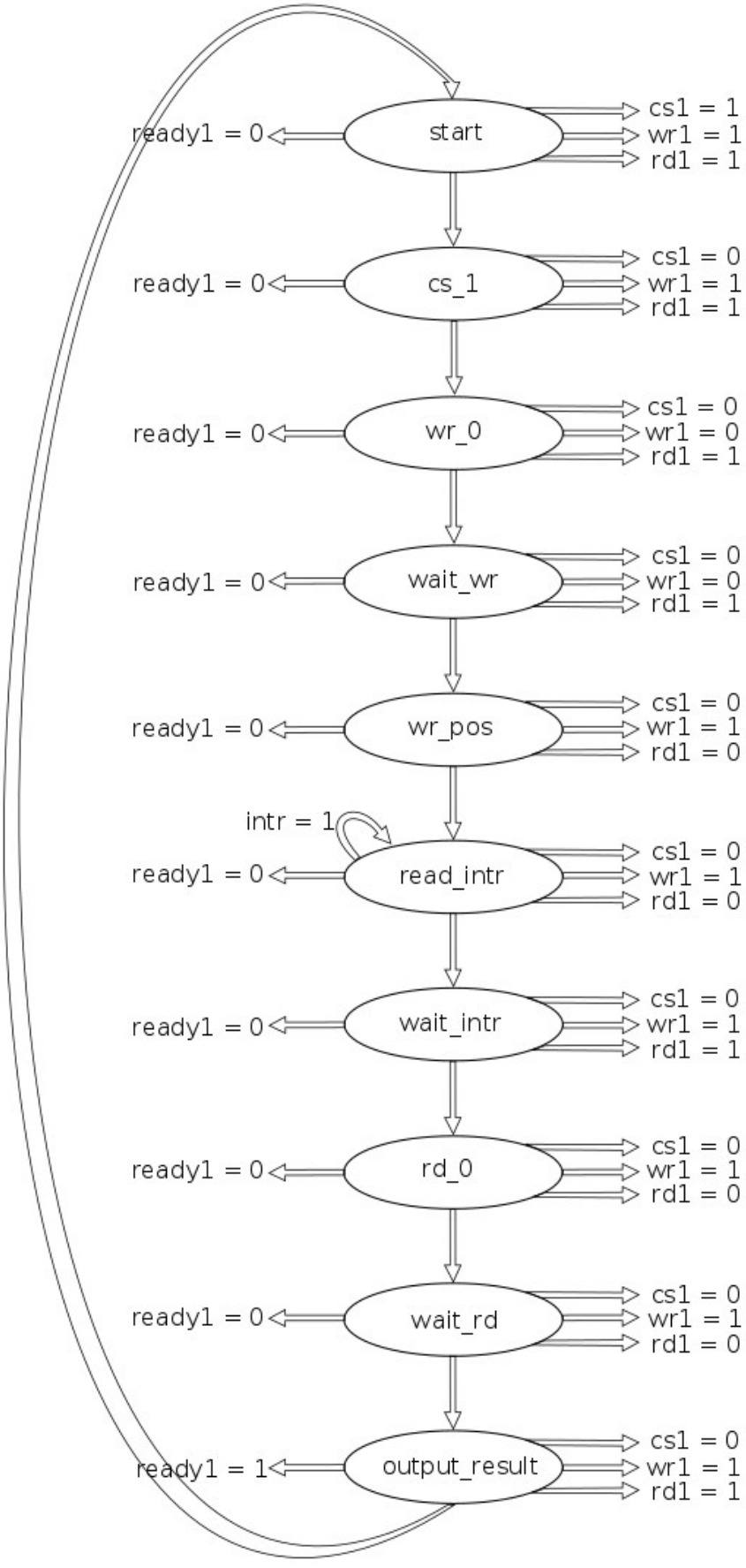
Δεύτερη κατάσταση είναι η κατάσταση **cs_0**, όπου εδώ το chip select 1 (cs1) θα πάρει την τιμή 0, για να ενεργοποιηθεί το chip του ADC 0804. Ενώ όλα τα υπόλοιπα σήματα θα παραμείνουν ίδια, δηλαδή το write 1 (wr1) έχει τιμή 1, το read 1 (rd1) επίσης 1 και το ready1 παραμένει 0.

Στην τρίτη κατάσταση με το όνομα **wr_0**, το chip select 1 (cs1) παραμένει 0, το write 1 (wr1) θα γίνει 0, το read 1 (rd1) είναι 1 και το ready1 είναι 0. Όσο το cs1 είναι 0, είδαμε ότι οδηγούμε το wr1 σε 0, η ενέργεια αυτή έχει σαν αποτέλεσμα την επανεκκίνηση (reset) των καταχωρητών του μετατροπέα. Με την έναρξη του παλμού reset ο ακροδέκτης intr εξάγει λογικό 1.

Τέταρτη κατάσταση είναι η κατάσταση με το όνομα **wait_wr**, όπου παίζει τον ρόλο του σταθεροποιητή τιμών. Δημιουργεί μια αναμονή στον controller για έναν κύκλο ρολογιού. Επαναλαμβάνουμε την ίδια τιμή σε όλα τα σήματα. Κάθε φορά που η πρόταση wait καθυστερεί τη διεργασία στη μέση της εκτέλεσης, αποδίδονται οι ίδιες τιμές στα σήματα.

Συνεχίζουμε στην πέμπτη κατάσταση με το όνομα **wr_pos**, όπου το chip select 1 (cs1) παραμένει 0, αλλάζουμε τον ακροδέκτη write 1 (wr1) από την τιμή 0 σε 1 έτσι σκανδαλίζει μία μετατροπή της τάσης εισόδου. Από την δεύτερη κατάσταση το intr είναι 1, και με τις παραπάνω ενέργειες, ο μετατροπέας θα ξεκινήσει μια μετατροπή μέσα σε χρόνο ίσο με μία έως οκτώ το πολύ περιόδους του ωρολογιακού σήματος. Το read 1 (rd1) είναι 1 και το ready1 είναι 0.

Στην έκτη κατάσταση με το όνομα **read_intr**, το chip select 1 (cs1) παραμένει 0, το write 1 (wr1) είναι 1, το read 1 (rd1) είναι 1 και το ready1 είναι 0. Επίσης δημιουργούμε έναν βρόγχος if, με συνθήκη στο intr, εάν λάβουμε την τιμή μηδέν τότε προχωράμε στην επόμενη κατάσταση wait_intr σε άλλη περίπτωση παραμένουμε στην ίδια την κατάσταση read_intr έως ότου λάβουμε την τιμή 1. Μόλις ολοκληρωθεί η μετατροπή, ο ακροδέκτης intr μεταβαίνει από λογικό ένα σε λογικό μηδέν, σηματοδοτώντας το τέλος της μετατροπής.



Σχήμα 5.6 Διάγραμμα Καταστάσεων για τον ADC 0804

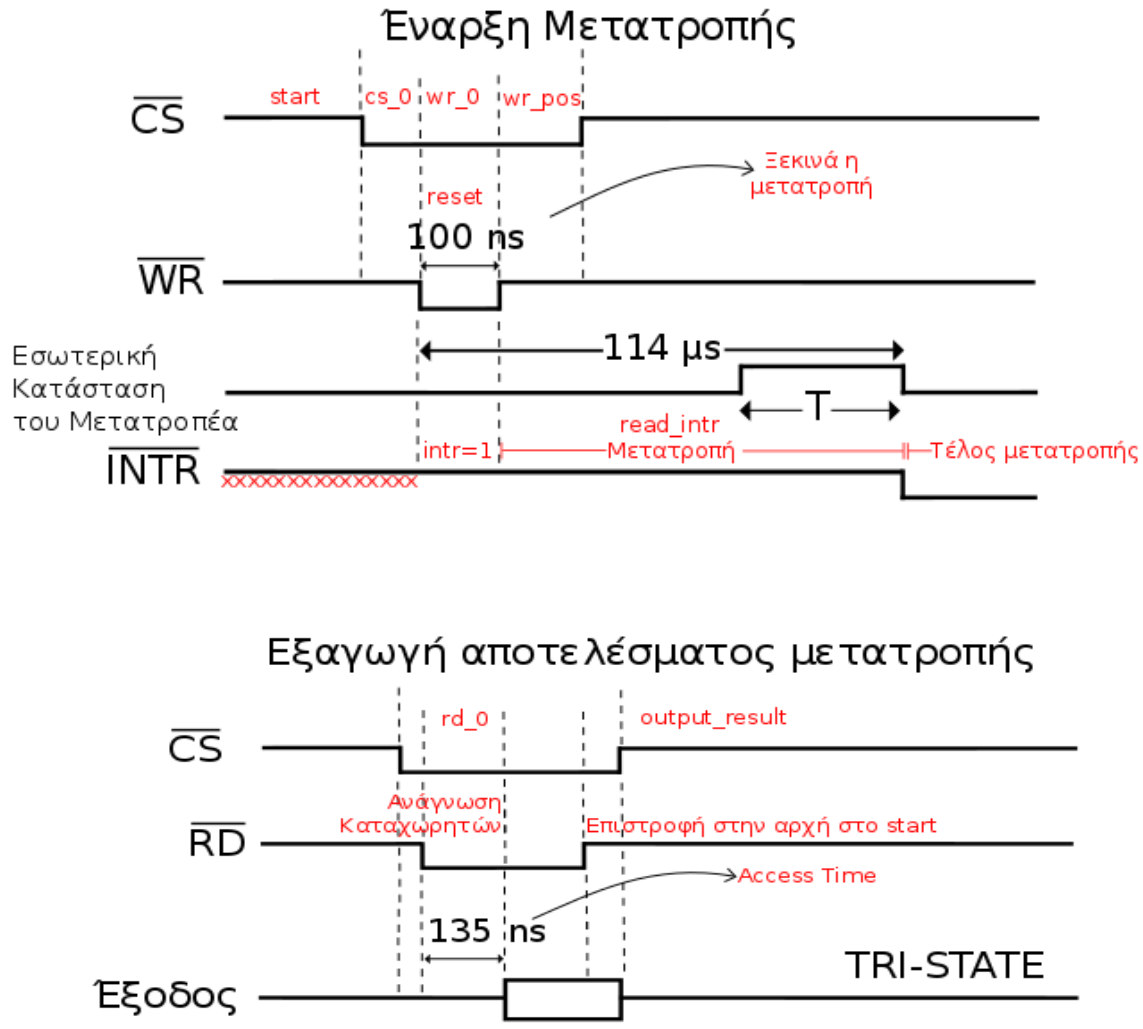
Η έβδομη κατάσταση με το όνομα **wait_intr**, παίζει πάλι τον ρόλο του σταθεροποιητή τιμών. Δημιουργεί μια αναμονή στον controller για έναν κύκλο ρολογιού. Επαναλαμβάνουμε την ίδια τιμή σε όλα τα σήματα.

Στην όγδοη κατάσταση με το όνομα **rd_0**, το chip select 1 (cs1) παραμένει 0, το write 1 (wr1) παραμένει 1, ο ακροδέκτης read 1 (rd1) θα μεταβεί από λογικό ένα σε λογικό μηδέν με αυτόν τον τρόπο ενεργοποιούμε την ανάγνωση του καταχωρητή όπου αποθηκεύσαμε το αποτέλεσμα της μετατροπής και τα εξάγουμε στους ακροδέκτες DB0 έως DB7. Το ready1 παραμένει 0.

Η ένατη κατάσταση με το όνομα **wait_rd**, παίζει πάλι τον ρόλο του σταθεροποιητή τιμών. Δημιουργεί μια αναμονή στον controller για έναν κύκλο ρολογιού. Επαναλαμβάνουμε την ίδια τιμή σε όλα τα σήματα.

Στην δέκατη και τελική κατάσταση **output_result**, το chip select 1 (cs1) παραμένει 0, το write 1 (wr1) παραμένει 1, το read 1 (rd1) θα γίνει 1 και το ready1 θα γίνει 1 όπου σηματοδοτεί την έναρξη του DAC ES52110. Ο βρόγχος κλείνει γυρίζοντας πάλι στην αρχική κατάσταση start.

Σε αυτό το σημείο αφού τελειώσαμε την ανάλυση του διαγράμματος καταστάσεων του ADC είναι χρήσιμο να αναφερθούμε ξανά στο διάγραμμα χρονισμού, αλλά αυτή τη φορά με περισσότερες πληροφορίες στο σχήμα.



Σχήμα 5.7 Διάγραμμα Χρονισμού για την Έναρξη και Εξαγωγή της Μετατροπής στον ADC0804

5.2.3 Σχεδίαση Ελεγκτή για τον ADC0804

Ο Controller (Ελεγκτής) ή αλλιώς *αντισταθμιστής* ή *ρυθμιστής* είναι το κομμάτι κώδικα που είναι κατασκευασμένο ώστε το σήμα που παράγει και τροφοδοτεί το σύστημα να αποτελεί την κατάλληλη διέγερση για να αποφέρει την επιθυμητή έξοδο του συστήματος.

Ο ελεγκτής για τον ADC έχει δημιουργηθεί σύμφωνα με το διάγραμμα καταστάσεων του. Ουσιαστικά έχουμε δημιουργήσει μία μηχανή πεπερασμένων καταστάσεων, η οποία αποτελείται από δέκα καταστάσεις.

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

entity ADC_controller is
        port (clk,rst,intr : in std_logic;
                p: in std_logic_vector(7 downto 0);
                x :out std_logic_vector(7 downto 0);
                cs,rd,wr,ready :out std_logic);
end ADC_controller;

architecture CONTROLLER of ADC_controller is
        type state is
        (start,wr_0,cs_0,wr_pos,wait_wr,read_intr,wait_intr,rd_0,wait_rd,output_result);
        signal present_state,next_state:state;
        signal cs1, wr1, rd1, ready1 :std_logic;
begin
------------Lower Section of FSM------------
        process (clk,rst)
        begin
                if (rst='1') then
                        present_state<=start;
                elsif (clk'event and clk='1') then
                        present_state<=next_state;
                end if;
        end process;

------------Upper Section of FSM------------
        process (present_state,intr)
        begin
                case present_state is
```

```
when start => --idle state
        cs1 <= '1';
        wr1 <= '1';
        rd1 <= '1';
        ready1<='0';
        next_state <= cs_0;

when cs_0=> --clear chip select
        cs1<='0';
        wr1<='1';
        rd1<='1';
        ready1<='0';
        next_state<=wr_0;

when wr_0=> --clear wr
        cs1<='0';
        wr1<='0'; --reset registers
        rd1<='1';
        ready1<='0';
        next_state<=wait_wr;

when wait_wr=> --wait for reset of INTR
        cs1<='0';
        wr1<='0';
        rd1<='1';
        ready1<='0';
        next_state<=wr_pos;

when wr_pos=> --positive edge of wr signal
        cs1<='0';
        wr1<='1'; --start conversion
        rd1<='1';
        ready1<='0';
        next_state<=read_intr;
```

```
when read_intr=> --wait for intr to go low
        cs1<='0';
        wr1<='1';
        rd1<='1';
        ready1<='0';
                if intr='1' then --intr is complemented as input
                        next_state<=read_intr;
                else
                        next_state<=wait_intr;
                end if;

when wait_intr=> --wait for intr to settle to 0
        cs1<='0';
        wr1<='1';
        rd1<='1';
        ready1<='0';
        next_state<=rd_0;

when rd_0=> --set rd to zero for output
        cs1<='0';
        wr1<='1';
        rd1<='0';
        ready1<='0';
        next_state<=wait_rd;

when wait_rd=> --keep rd clear
        cs1<='0';
        wr1<='1';
        rd1<='0';
        ready1<='0';
        next_state<=output_result;

when output_result=> --access result
        cs1<='0';
        wr1<='1';
        rd1<='1';
```

```
                    ready1<='1';
                    next_state<=start;

            end case;
        end process;



----------Output Section----------
process (clk,rst)
        begin
            if (rst='1') then
                cs <= '1';
                wr <= '1';
                rd <= '1';
                ready<='0';
                x<="00000000";
            elsif (clk'event and clk='1') then
                cs <= cs1;
                wr <= wr1;
                rd <= rd1;
                ready<=ready1;
                if ready1='1' then
                    x<=p;
                else
                    x<="00000000";
                end if;
            end if;
        end process;

end CONTROLLER;
```

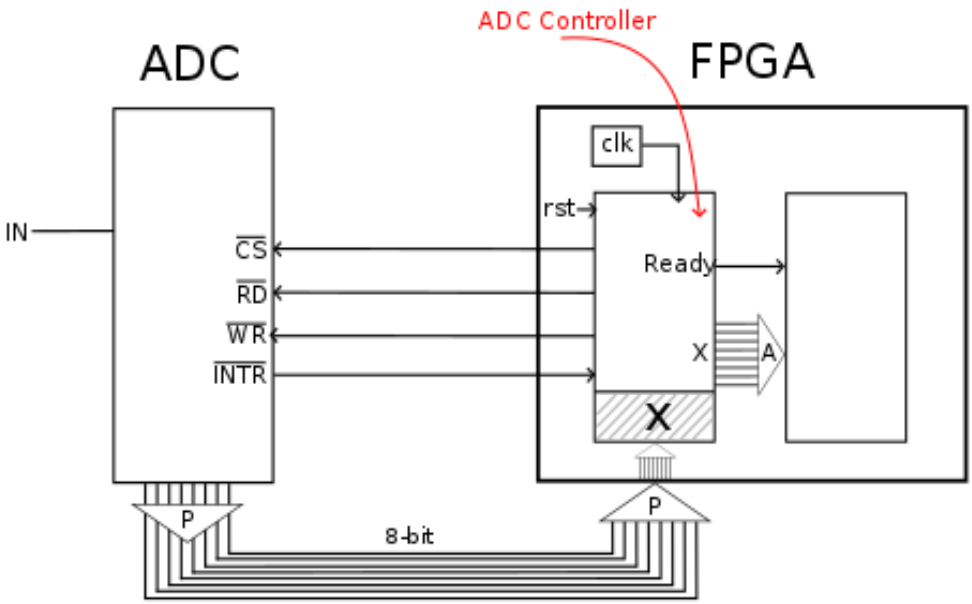
Στον παραπάνω κώδικα χρησιμοποιούμε τα πακέτα των βιβλιοθηκών std_logic_1164 και numeric_std για την εκτέλεση των αριθμητικών πράξεων. Στην δήλωση της οντότητας (Entity) η οποία έχει το όνομα ADC_controller, αντιστοιχούμε τις θέσεις των εισόδων και των εξόδων.

Οι είσοδοι του ADC Controller είναι το Clock, το Reset, το Interrupt και το P.

Clock: Είναι το σήμα του ρολογιού για τον συγχρονισμό του κυκλώματος (μορφή std_logic).
Reset: Είναι το σήμα για την επανεκκίνησης του κυκλώματος (μορφή std_logic).
Interrupt: Είναι το σήμα διακοπής που το έχουμε σε συγκεκριμένο βρόγχο του κυκλώματος, ενεργοποιείται όταν ο ADC έχει διαβάσει μια αναλογική τιμή και την έχει αποθηκεύσει στο εσωτερικό του. Απενεργοποιείται όταν η αναλογική τιμή έχει μετατραπεί σε ψηφιακή και έχει μεταφερθεί στην έξοδο. Αν λάβουμε ως σήμα την τιμή 0 τότε προχωρούμε στην επόμενη κατάσταση, αν λάβουμε 1 μένουμε στην παρούσα κατάσταση μέχρι να έρθει η τιμή 0 (μορφή std_logic).
P: 8-bit ψηφιακή είσοδος (μορφή std_logic_vector (7 downto 0)).

Οι έξοδοι του ADC Controller είναι το Chip Select, Write, Read, X και το Ready

Chip Select (CS): Ακροδέκτης για την επιλογή μνήμης, έχουμε ενεργή τιμή σε κατάσταση LOW. Το Chip Select είναι ένα σήμα ελέγχου, το οποίο επιλέγει το συγκεκριμένο Chip μνήμης, στην περίπτωσή μας το Chip του ADC. (μορφή std_logic).
Write - Enable (WE): Ακροδέκτης που επιτρέπει την εγγραφή, έχουμε ενεργή τιμή σε κατάσταση LOW. Σήμα ελέγχου, που υποδηλώνει τον τύπο προσπέλασης (ενεργό = εγγραφή, ανενεργό = δεν εγγράφει) (μορφή std_logic).
Read - Enable (RE): Ακροδέκτης που επιτρέπει την ανάγνωση, έχουμε ενεργή τιμή σε κατάσταση LOW. Σήμα ελέγχου, δηλαδή ενεργοποιεί την δυνατότητα να διαβάσει μια αναλογική τιμή από την είσοδο (ενεργό = ανάγνωση, ανενεργό = δεν διαβάζει). (μορφή std_logic).
X: 8-bit ψηφιακή έξοδος (μορφή std_logic_vector (7 downto 0)).
Ready: Είναι ένα σήμα εξόδου όπου είναι σε λογικό ένα στον ελεγκτή του ADC και χρησιμοποιείται για να μεταβεί το σήμα από το FPGA στον ελεγκτή του DAC (μορφή std_logic).
Στο Σχήμα 5.8 φαίνονται οι είσοδοι και οι έξοδοι του ADC προς το FPGA



Σχήμα 5.8 Είσοδοι και έξοδοι του ADC Controller

Στην συνέχεια έχουμε την δήλωση της αρχιτεκτονικής του ADC Controller με όνομα CONTROLLER. Στην αρχή του τμήματος της αρχιτεκτονικής γίνεται η δήλωση των καταστάσεων και των σημάτων. Οι καταστάσεις είναι start, wr_0, cs_0, wr_pos, wait_wr, read_intr, wait_intr, rd_0, wait_rd, output_result. Τα σήματα που περιλαμβάνει ο κώδικας μας είναι present_state, next_state, cs1, wr1, rd1, ready1. Στη συνέχεια αρχίζει η πρώτη διεργασία όπου μέσα στην λίστα ευαισθησίας είναι ο παλμός Clock και το Reset και ξεκινά η εντολή ακολουθιακής αντιστοίχισης if. Εάν το reset λάβει την λογική τιμή ένα τότε παραμένει στην αρχική μας κατάσταση start, διαφορετικά αν ο παλμός ρολογιού κάνει μετάβαση από το μηδέν στο ένα και πηγαίνει στην επόμενη κατάσταση. Έτσι τελειώνει η πρώτη διεργασία.

Ακολουθεί η δεύτερη διεργασία όπου μέσα στην λίστα ευαισθησίας υπάρχει το present_state και το interrupt. Για την περιγραφή των μηχανών πεπερασμένων καταστάσεων χρησιμοποιούμε την ακολουθιακή εντολή case. Σε κάθε διαδοχικό παλμό ρολογιού το κύκλωμα βρίσκεται σε μία από τις δυνατές καταστάσεις του. Η μετάβαση στην επόμενη κατάσταση συμβαίνει με βάση την τρέχουσα κατάσταση και την τιμή του σήματος εισόδου. Τα σήματα εξόδου εξαρτώνται από την τρέχουσα κατάσταση.

Στην πρώτη κατάσταση με το όνομα start, το cs1, wr1, rd1 παίρνουν την τιμή 1, ενώ το ready1 λαμβάνει την τιμή 0, και πηγαίνουμε στην επόμενη κατάσταση που έχει όνομα cs_0. Σε αυτή την κατάσταση το cs1 και το ready1 παίρνουν την τιμή 0, ενώ το wr1 και το rd1 παίρνουν την τιμή 1. Πηγαίνουμε στην επόμενη κατάσταση η οποία έχει το όνομα wr_0, σε αυτήν την κατάσταση λογικό μηδέν λαμβάνουν τα σήματα cs1, wr1 και ready1, ενώ το rd1 παραμένει 1. Στην επόμενη κατάσταση με το όνομα wait_wr, λαμβάνει τις ίδιες τιμές με την κατάσταση wr_0, για την σταθεροποίηση των τιμών και μεταβαίνουμε στην επόμενη κατάσταση. Αυτό που αλλάζει στην επόμενη κατάσταση με το όνομα wr_pos, είναι ότι το wr θα λάβει την τιμή 1 για να ξεκινήσει η μετατροπή. Και προχωράμε στην επόμενη κατάσταση η οποία έχει το όνομα read_intr. Σε αυτή τη κατάσταση τα σήματα έχουν τις ίδιες τιμές με την κατάσταση wr_pos με την διαφορά ότι υπάρχει η ακολουθιακή εντολή if, στην οποία όταν το intr γίνει ένα, μένουμε στην κατάσταση read_intr διαφορετικά θα πάμε στην επόμενη κατάσταση, η οποία έχει το όνομα wait_intr. Αυτή η κατάσταση λειτουργεί πάλι για σταθεροποίηση των τιμών και μεταβαίνουμε στη επόμενη κατάσταση η οποία έχει το όνομα rd_0. Σε αυτή την κατάσταση το read1 θα λάβει την τιμή μηδέν και θα μεταβούμε στην επόμενη κατάσταση με όνομα wait_rd. Και αυτή η κατάσταση λειτουργεί για την σταθεροποίηση των τιμών και μεταβαίνουμε την τελευταία μας κατάσταση με το όνομα output_result. Σε αυτή την κατάσταση το cs1 λαμβάνει την τιμή μηδέν, ενώ το wr1, rd1 και το ready1 λαμβάνουν την τιμή ένα. Η επόμενη κατάσταση που θα μεταβούμε θα είναι η αρχική μας, δηλαδή η start για να ξεκινήσει και πάλι η διαδικασία μετατροπής. Και τελειώνει η δεύτερη διεργασία.

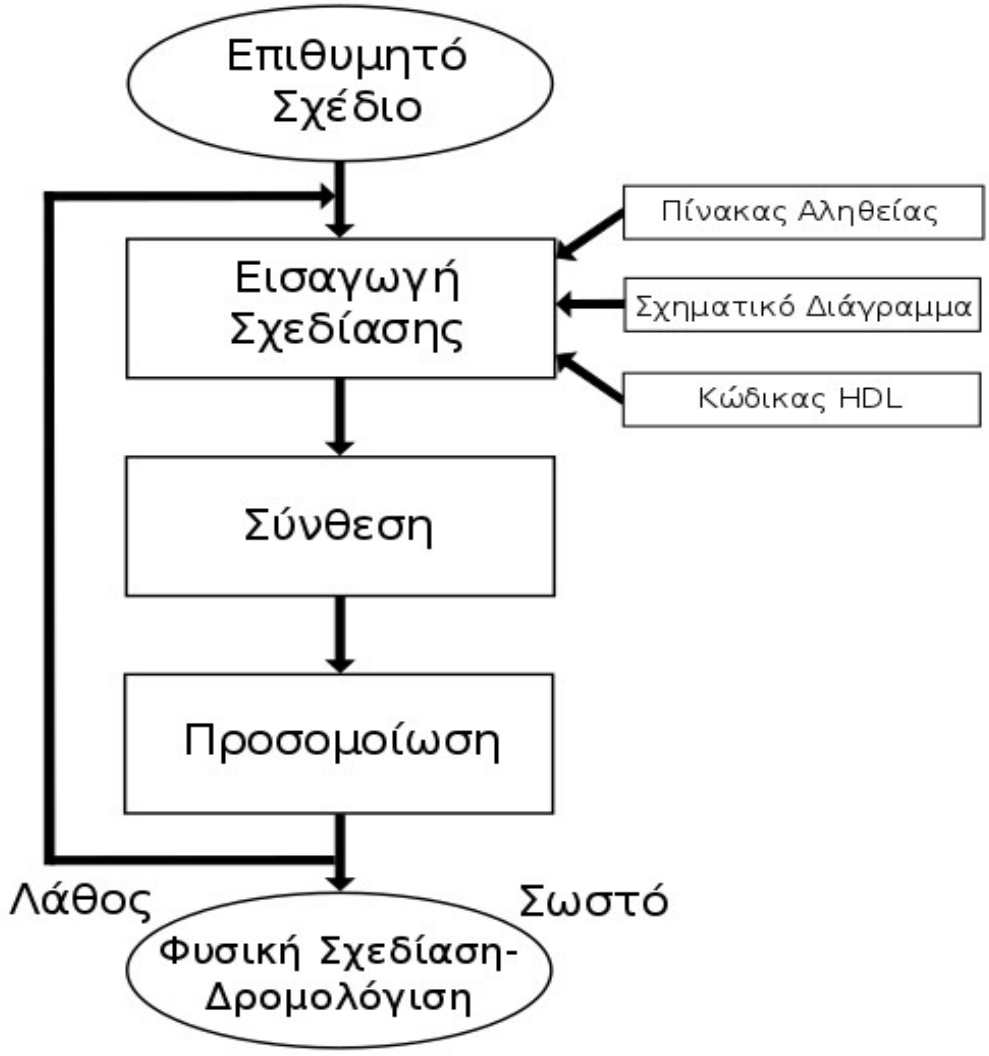
Ακολουθεί μια προαιρετική διεργασία για να βεβαιωθούμε ότι τα αποτελέσματα είναι απαλλαγμένα από δυσλειτουργίες. Στην λίστα ευαισθησίας αυτής της διεργασίας περιλαμβάνεται το Clock και το Reset. Ξεκινά η ακολουθιακή εντολή If, όπου όταν το reset είναι ίσο με ένα, τότε το cs, wr και το rd λαμβάνουν την τιμή ένα, ενώ το ready λαμβάνει την τιμή μηδέν και αποδίδουμε στην έξοδο X την 8-ψήφια τιμή “00000000”. Διαφορετικά εάν το Clock μεταβεί από την τιμή μηδέν στο ένα, αναθέτουμε στο cs το cs1, στο wr το wr1, στο rd το rd1, στο ready το ready1 και αν το ready1 γίνει ίσο με ένα τότε αναθέτουμε στο X την τιμή του P διαφορετικά το X θα πάρει την τιμή “00000000”. Και τελειώνει η ακολουθιακή εντολή If, η διεργασία μας και ο κώδικας του Controller μας.

5.2.4 Προσομοίωση του ADC0804

Η *προσομοίωση (simulation)* είναι ένα από τα τελικά στάδια της υλοποίησης ενός ολοκληρωμένου κυκλώματος. Με απλά λόγια η προσομοίωση είναι η ανάπτυξη προγραμμάτων σε ηλεκτρονικό υπολογιστή τα οποία "μιμούνται" τις πραγματικές καταστάσεις των κυκλωμάτων που περιγράψαμε παραπάνω. Η προσομοίωση ουσιαστικά αντικαθιστά την πειραματική προσέγγιση. Έτσι μπορούν να γίνουν πειράματα με ελάχιστο κόστος, πολύ γρήγορα και με ασφάλεια σε σενάρια που μέχρι τώρα δεν είναι εφικτά.

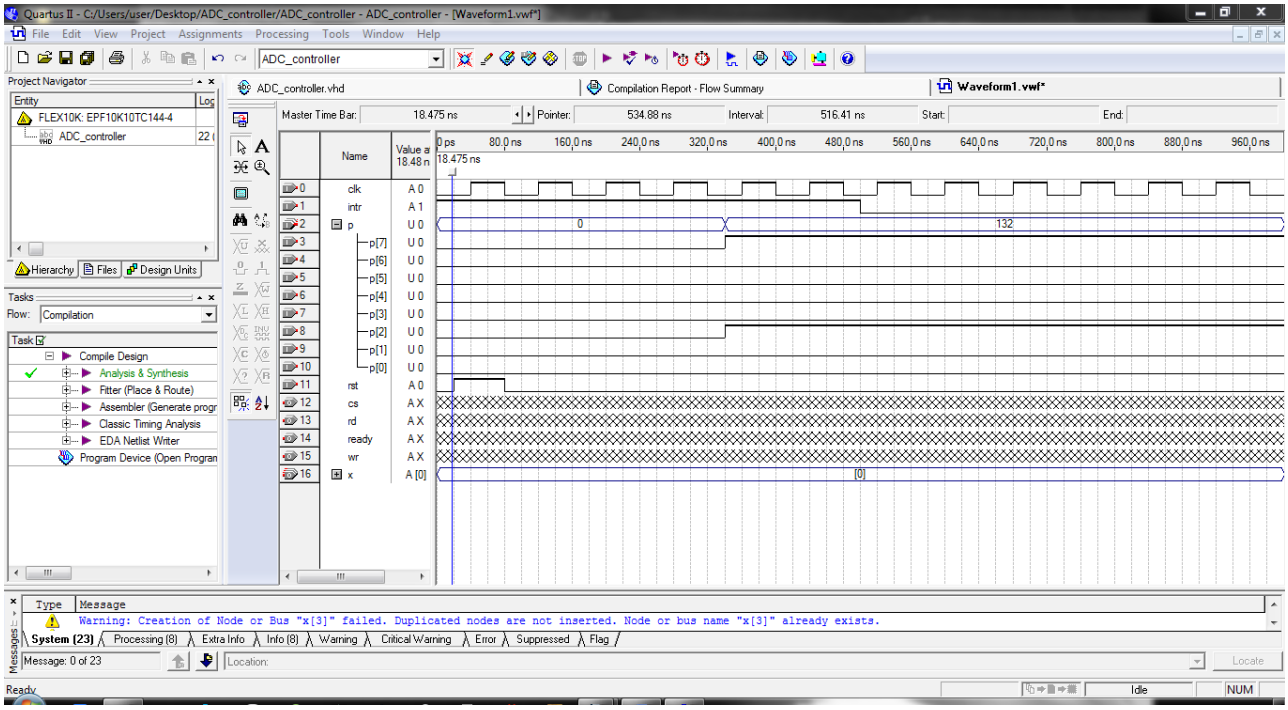
Ο κύκλος εκτέλεσης αποτελείται από την φάση αρχικοποίησης, ακολουθούμενη από έναν επαναλαμβανόμενο κύκλο εξομοίωσης. Η φάση αρχικοποίησης γίνεται στον χρόνο 0 και όλοι οι οδηγοί σημάτων παίρνουν αρχική τιμή. Στην συνέχεια κάθε process ή συν τρέχουσα πρόταση ξεκινάει και εκτελεί τις ακολουθιακές εντολές της και συνήθως προγραμματίζει αναθέσεις τιμών σε σήματα. Όλες οι processes διακόπτουν την λειτουργία τους κάποια στιγμή και περιμένουν να ενεργοποιηθούν από κάποια σήματα που ανήκουν στην λίστα ευαισθησίας τους. Εκτελούνται οι πρώτες αναθέσεις σημάτων (νωρίτερα στον χρόνο) και ενεργοποιούνται οι processes που είναι ευαίσθητες.

Στο Σχήμα 5.9 που ακολουθεί, δίνεται μια απλή περιγραφή της διαδικασίας ανάπτυξης μοντέλων προσομοίωσης.



Σχήμα 5.9 Περιγραφή της διαδικασίας ανάπτυξης μοντέλων προσομοίωσης

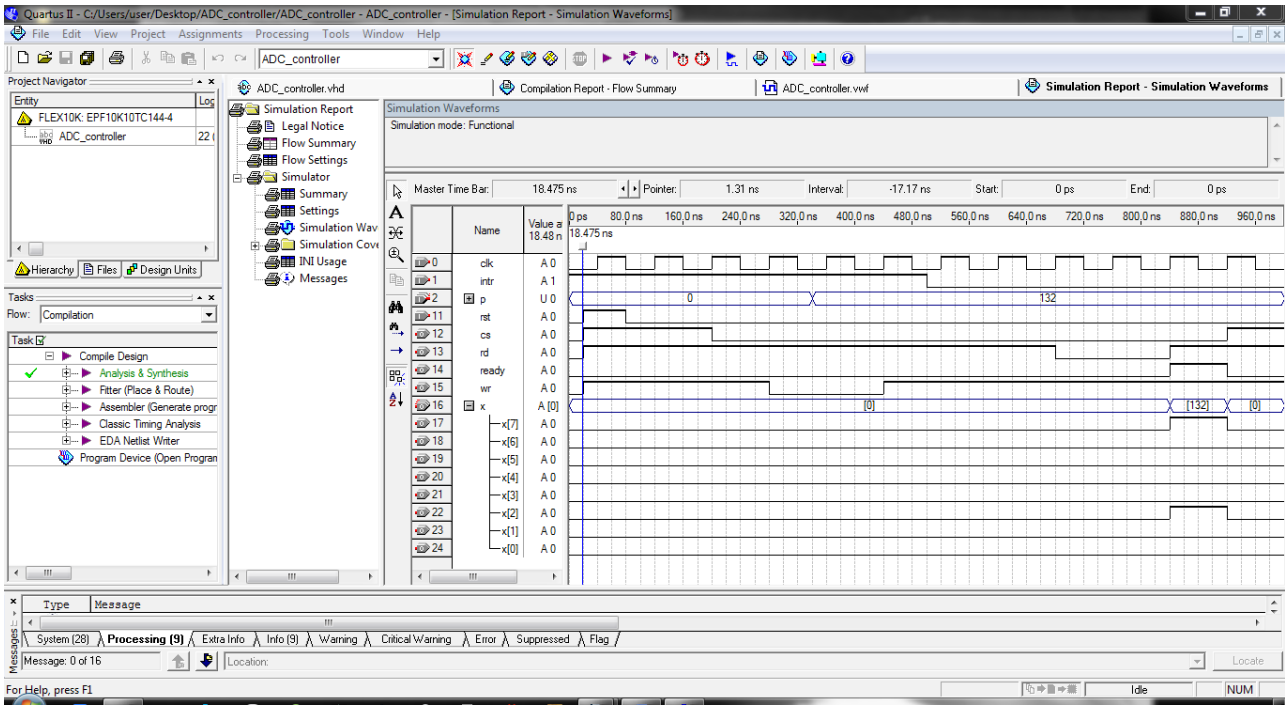
Όπως αναφέραμε αναλυτικά στο κεφάλαιο 3.6.2 η προσομοίωση γίνεται από το Waveform File του Quartus II. Ανοίγοντας το εισάγουμε τις εισόδους και τις εξόδους του ADC_Controller. Από την γραμμή εργαλείων ρυθμίζουμε τα σήματα σε 0 ή 1, όπως φαίνεται και στο Σχήμα 5.10, τα οποία εύκολα μπορούμε να δούμε στο κομμάτι της οντότητας του κώδικα. Στον κώδικα της αρχιτεκτονικής υπάρχουν δέκα καταστάσεις, άρα αντίστοιχα, μέσα στο περιβάλλον σχεδίασης της προσομοίωσης θα δημιουργήσουμε τουλάχιστον δέκα παλμούς ρολογιού. Στην περίπτωση μας χρησιμοποιήσαμε δώδεκα παλμούς ρολογιού για να εξασφαλίσουμε την εγκυρότητα των αποτελεσμάτων, με τη συνολική διάρκεια της προσομοίωσης να την έχουμε επιλέξει στα 1000 ns. Ο κάθε παλμός ρολογιού είναι 80 ns, άρα συνολικά έχουμε 960 ns για την ολοκλήρωση των ωρολογιακών παλμών.



Σχήμα 5.10 Προσομοίωση του ADC 0804 μόνο με τιμές εισόδων και άγνωστες τις τιμές εξόδων

Από το διάγραμμα χρονισμού καταλαβαίνουμε ότι ο ρόλος του interrupt στον κώδικα είναι για να μας δείξει το τέλος της μετατροπής, με την μετάβαση του από το 1 στο 0. Γιαυτό το λόγο θα δώσουμε την τιμή 1 στο interrupt από 20ns έως 500ns, και από 500ns έως 1000ns θα δώσουμε στο interrupt την τιμή 0. Για να γίνει η ενεργοποίηση της μετατροπής του κυκλώματος θα δώσουμε στο reset την τιμή 1 από 20ns έως 80ns. Τέλος από τα 0ns έως τα 340ns η είσοδος p θα έχει την δεκαδική τιμή 0, ενώ από 340ns έως 1000ns θα δώσουμε την τυχαία δεκαδική τιμή 132, η οποία αντιστοιχεί στην δυαδική τιμή 1000,0100.

Εφόσον έχουμε δώσει τιμές σε όλες τις εισόδους μας τρέχουμε την προσομοίωση από το Menu Processing / Start Simulation για να διαπιστώσουμε αν γίνεται σωστά η λειτουργία του κώδικα από την εξαγωγή των αποτελεσμάτων της προσομοίωσης. Όπως βλέπουμε και στο Σχήμα 5.11 ο κώδικας μας λειτουργεί σωστά.



Σχήμα 5.11 Προσομοίωση του ADC 0804

Η ενεργοποίηση του cs γίνεται στις επιθυμητές καταστάσεις σύμφωνα με το διάγραμμα καταστάσεων δηλαδή από 20ns έως 200ns πέρνουμε στην έξοδό μας την τιμή 1, ενώ από 200ns έως 920ns παίρνουμε την τιμή 0. Επιπλέον το rd από 20ns έως 680ns βγάζει στην έξοδο την τιμή 1, ενώ από 680ns έως 840ns έχει την τιμή 0. Το ready έχει παντού την τιμή 0 έως την εξαγωγή των αποτελεσμάτων που γίνεται 1, δηλαδή από τα 840ns έως 920ns (το ready χρησιμοποιείται για σηματοδοτήσει την έναρξη του ελεγκτή του DAC, όπου θα δούμε στο παρακάτω κεφάλαιο). Εφόσον το cs είναι ήδη 0 έχουμε την επανεκκίνηση των καταχωρητών με το wr από 280ns έως 440ns να μας βγάζει στην έξοδο την τιμή 0, δηλαδή το πλάτος παλμού του wr είναι 160ns, με ελάχιστο πλάτος παλμού από τον κατασκευαστή του ADC0804 να είναι 100ns. Τέλος βλέπουμε ότι στην έξοδο μας x παίρνουμε την δεκαδική τιμή 132, δηλαδή την τιμή που είχαμε δώσει στην είσοδο p.

5.2.5 Αντιστοίχηση ακροδεκτών του ADC0804

Εφόσον έχει γίνει στοιχειώδη κατανόηση της αναπτυξιακής πλακέτας LP-2900 και του ADC0804, στον Πίνακα 5.1 παρουσιάζονται οι ακροδέκτες του διαμορφωμένου κυκλώματος FPGA που συνδέονται με διακόπτες εισόδου και leds απεικόνισης της εξόδου.

A/D → ADC0804

Code	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7
Device	ADC0804							
Pin	Pin 131(LS)	Pin 132	Pin 133	Pin 135	Pin 136	Pin 137	Pin 138	Pin 140(MS)

› rst → Pin 124
› DB0~DB7 also connect D0~D7 of LCD

Code	/CS	/RD	DE1	DE2	WR_OUT	AD_INTR
Device	ADC0804					
Pin	Pin 38	Pin 128	Pin 33	Pin 36	Pin 37	Pin 143

› AD_WR is the output end Y6 of 3-to-8 decoder
› DE0, DE1 and DE2 are the input ends of 3-to-8 decoder
› /RD also connects the RW of LCD

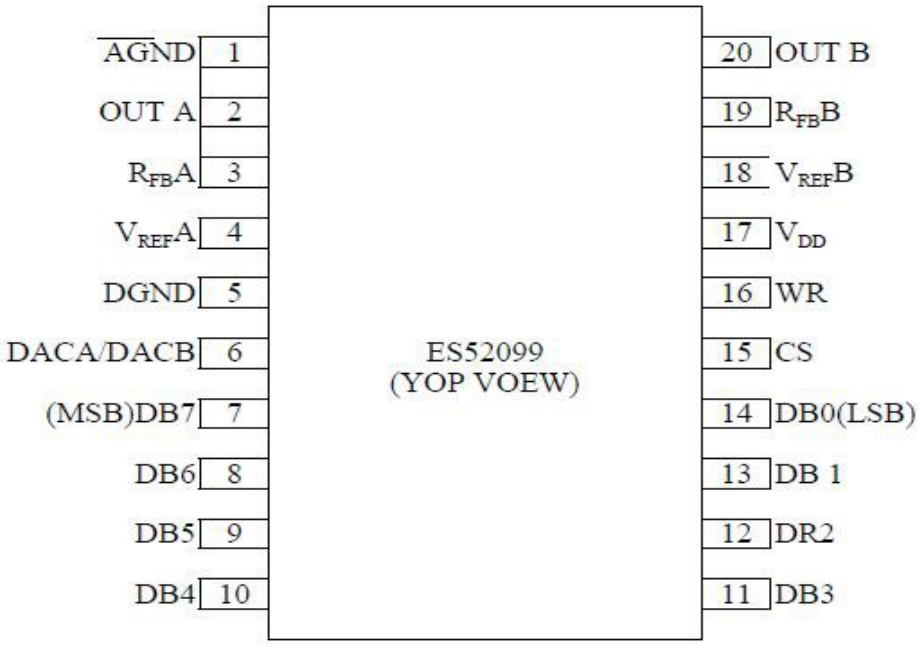
Πίνακας 5.1 Αντιστοίχηση ακροδεκτών του ADC0804

5.3 Ιεραρχική δομή σχεδίασης για τον DACES52110

Σε αυτό το κεφάλαιο θα δούμε την ιεραρχική δομή για την διασύνδεση του μετατροπέα DACES52110 σε διάταξη FPGA. Τα επίπεδα της υλοποίησης της πτυχιακής είναι: η λειτουργική περιγραφή και το διάγραμμα χρονισμού για τον DACES52110, το διάγραμμα καταστάσεων, η σχεδίαση του ελεγκτή, η προσομοίωση και τέλος η αντιστοίχηση των ακροδεκτών για τον DACES52110.

5.3.1 Λειτουργική Περιγραφή και Διάγραμμα Χρονισμού (DACES52110)

Το κύκλωμα DACES52110 στον ακροδέκτη 1 AGND έχει την αναλογική γείωση. Ο δεύτερος ακροδέκτης, με το όνομα OUT A, όπως και ο ακροδέκτης 20, OUT B είναι οι ακροδέκτες που εξάγουν τα αποτελέσματα ανάλογα ποια από τα δυο ενσωματωμένα τσιπ επιλέξαμε. Στους ακροδέκτες 3 και 19 βρίσκεται εσωτερική αντίσταση ανάδρασης.



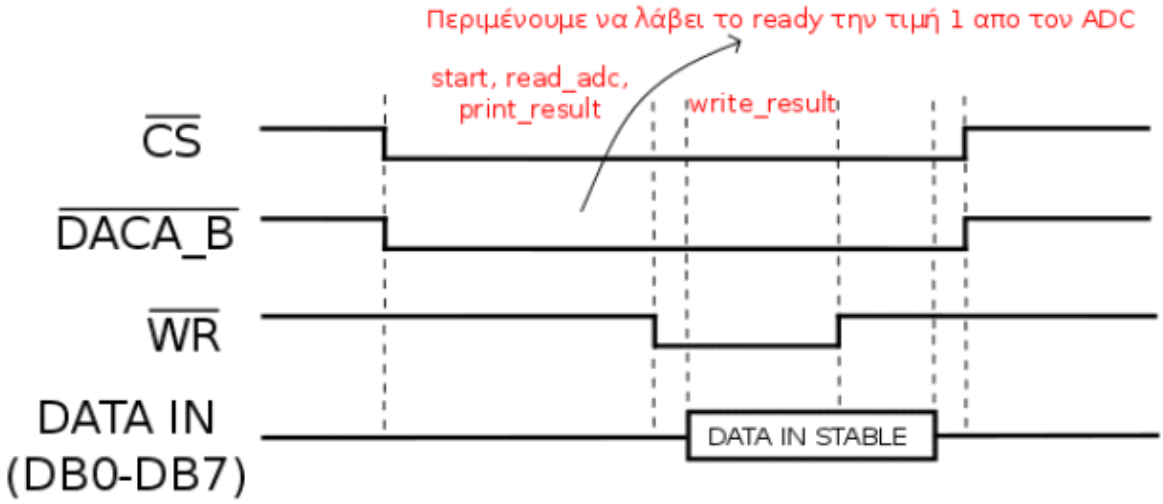
Σχήμα 5.12 Διάγραμμα ακροδεκτών του μετατροπέα D/A DACES52110

Στους ακροδέκτες 4 και 18 βρίσκεται η τάση αναφοράς για το δικτύωμα κλίμακας R-2R. Στον ακροδέκτη 5 DGND έχει την ψηφιακή γείωση. Ο ακροδέκτης 6, DAC A / DAC B είναι ο ακροδέκτης που μας δίνει την δυνατότητα να επιλέξουμε ποιον από τους δύο ενσωματωμένους dac

θα επιλέξουμε. Από τον ακροδέκτη 7 που αντιστοιχεί στον DB0 μέχρι και 14 που αντιστοιχεί στον DB7 είναι ο δίαυλος δεδομένων TTL/CMO. Ο ακροδέκτης DB0 να δέχεται το λιγότερο σημαντικό bit, και ο ακροδέκτης 14 το σημαντικότερο bit. Στον ακροδέκτη 15 και 16 βρίσκονται οι είσοδοι CS και WR για τον έλεγχο λειτουργίας του κυκλώματος. Το CS αφορά την ενεργοποίηση και απενεργοποίηση του κυκλώματος. Ενώ το WR αφορά την ενεργοποίηση και απενεργοποίηση της λειτουργίας της εγγραφής. Τέλος στον ακροδέκτη 17 βρίσκεται η τροφοδοσία ρεύματος. Οι ακροδέκτες του κυκλώματος DACES52110 παρουσιάζονται στο Σχήμα 5.12.

Οι ακροδέκτες CS (Chip Select), WR (Write) και DACA/B ενεργοποιούνται με λογικό μηδέν και χρησιμεύουν για τον έλεγχο της μετατροπής ψηφιακού σήματος σε αναλογικό. Για να γίνει η μετατροπή θα επιλέξουμε και θα οδηγήσουμε τον ακροδέκτη CS σε λογικό μηδέν.
Στη συνέχεια όσο ο ακροδέκτης CS είναι σε λογικό μηδέν, οδηγούμε για λίγο και τον ακροδέκτη WR σε λογικό 1. Με αυτή τη δραστηριότητα έχουμε σαν αποτέλεσμα την επανεκκίνηση των καταχωρητών του μετατροπέα. Έτσι θα ξεκινήσει μια μετατροπή μέσα σε χρόνο ίσο με μία έως οκτώ το πολύ περιόδους του ωρολογιακού σήματος.

Το αποτέλεσμα της μετατροπής για να οδηγηθεί στην έξοδο, δηλαδή στους ακροδέκτες DB0 έως DB7, θα πρέπει να κάνει μια λειτουργία ανάγνωσης, με την οποία η έξοδος του SAR μεταφέρετε στις εξόδους του απομονωτή τριών καταστάσεων (tri-state-buffer). Για να γίνει αυτό, θα χρειαστεί ο ακροδέκτης WR να μεταβεί σε λογικό 0. Τα αποτελέσματα θα εμφανιστούν στην έξοδο. Παρακάτω στο Σχήμα 5.13 υπάρχει το διάγραμμα χρονισμού, το οποίο δείχνει με παλμούς την εξαγωγή και τη μετατροπή του DACES5211.



Σχήμα 5.13 Διάγραμμα Χρονισμού για την Μετατροπή του DACES52110

Για το παραπάνω σχήμα λαμβάνουμε τις τυπικές τιμές από τον κατασκευαστή όπως είναι γραμμένες στα φύλλα δεδομένων στου chip.

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Chip Select to Write Set-Up Time	T_{CS}		60		—	ns
Chip Select to Write Hold Time	T_{CH}		10	—	—	ns
DAC Select to Write Set-Up Time	T_{AS}		60	—	—	ns
DAC Select to Write Hold Time	T_{AH}		10	—	—	ns
Data Select to Write Set-Up Time	T_{DS}		70	—	—	ns
Data Select to Write Hold Time	T_{DH}		10	—	—	ns
Write Pulse Width	T_{WR}		60	—	—	ns

Πίνακας 5.2 Τυπικές τιμές του κατασκευαστή για το διάγραμμα χρονισμού του DACES52110

5.3.2 Διάγραμμα Καταστάσεων για τον DAC ES52110

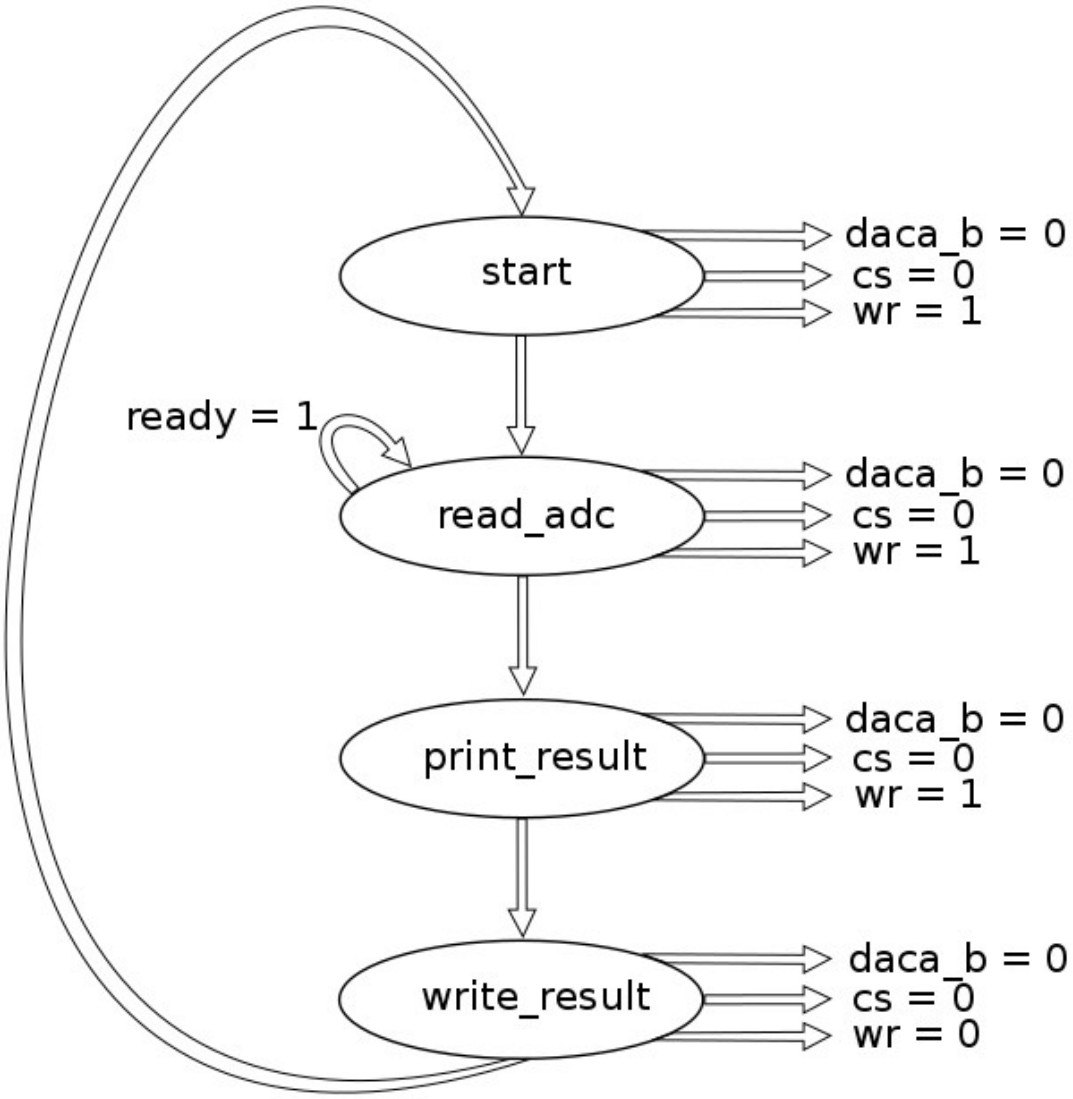
Το διάγραμμα καταστάσεων του DAC ES52110 απαρτίζεται από τέσσερις καταστάσεις και απεικονίζεται στο Σχήμα 5.14. Σε κάθε κατάσταση ορίζουμε κάθε φορά τις τιμές των σημάτων εισόδου και εξόδου.

Στην πρώτη κατάσταση με το όνομα **start**, δηλώνουμε τον ακροδέκτη DACa/DACb (daca_b) ίσο με το μηδέν. Είναι ο ακροδέκτης που ορίζει ποιος από τους δύο ενσωματωμένους dac θα λειτουργήσει, στην προκειμένη περίπτωση ο dac a γιατί θέσαμε την τιμή σε όλες τις καταστάσεις σε λογικό μηδέν. Επίσης θέτουμε την τιμή 0 στον ακροδέκτη chip select (cs) για την λειτουργία που κυκλώματος, και την τιμή του write (wr) σε λογικό ένα για να είναι ανενεργή η εγγραφή. Επίσης δεχόμαστε και την τιμή του ready από τον adc. Η τιμή του ready είναι 1, αυτός είναι και ο λόγος εκκίνησης του dac controller.

Στην δεύτερη κατάσταση με το όνομα **read_adc**, δημιουργούμε έναν βρόγχο με συνθήκη στο ready, εάν λάβουμε την τιμή ένα, σημαίνει ότι έλαβα το αποτέλεσμα της μετατροπής από τον ADC και προχωράμε στην επόμενη κατάσταση print_result σε άλλη περίπτωση παραμένουμε στην ίδια την κατάσταση read_adc έως ότου λάβουμε την τιμή 1. Ο ακροδέκτης DACa/DACb (daca_b) παραμένει με την λογική τιμή 0, το chip select (cs) με λογικό 0 και το write (wr) σε λογικό ένα.

Στην τρίτη κατάσταση με το όνομα **print result** επαναλαμβάνουμε τις ίδιες τιμές σε όλες τις εισόδους, και καθυστερώ για έναν κύκλο ρολογιού.

Στην τέταρτη και τελευταία κατάσταση με το όνομα **write**, αλλάζουμε την τιμή του ακροδέκτη write (wr) από την λογική τιμή 1 σε 0, αυτό έχει ως συνέπεια το αποτέλεσμα της μετατροπής να εμφανίζεται στην έξοδο στους 7 ακροδέκτες , (MSB)DB7 με 14 (LSB)DB0.



Σχήμα 5.14 Διάγραμμα Καταστάσεων για τον DAC ES52110

5.3.3 Σχεδίαση ελεγκτή για τον DACES522110

Ο ελεγκτής για τον DACES52110 έχει δημιουργηθεί σύμφωνα με το διάγραμμα καταστάσεων του. Ουσιαστικά έχουμε δημιουργήσει μια μηχανή πεπερασμένων καταστάσεων η οποία αποτελείται από τέσσερις καταστάσεις.

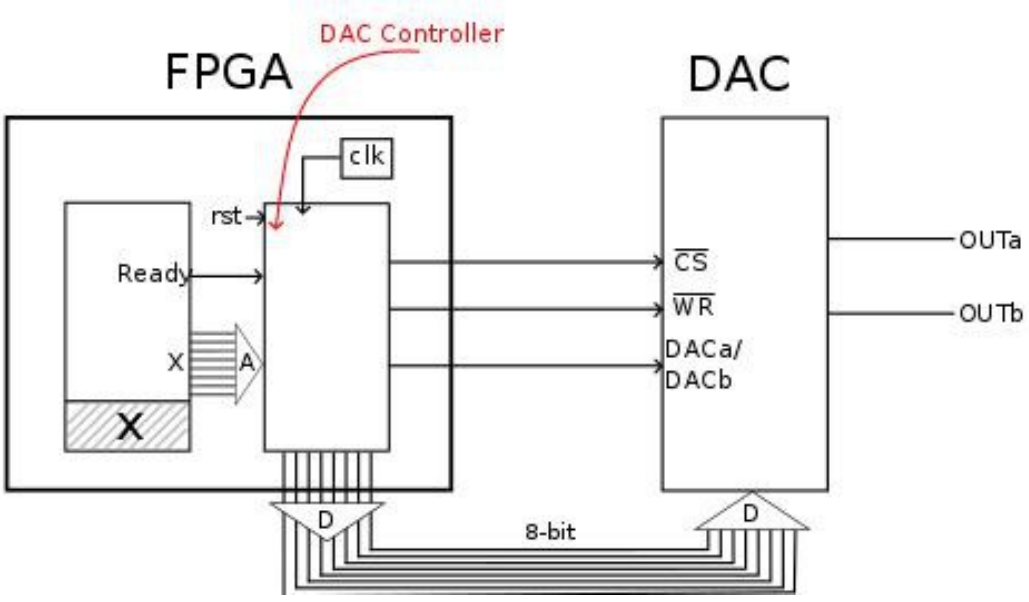
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
----------------------------------------
entity DAC_controller is
	port (clk,rst,ready:in std_logic;
		A:in std_logic_vector (7 downto 0);
		D:out std_logic_vector (7 downto 0);
		cs,wr,daca_b:out std_logic);
end DAC_controller;
----------------------------------------
```

```
architecture controller of DAC_controller is
        type state is (start,read_adc,print_result,write_result);
        signal present_state,next_state:state;
begin
 process(clk,rst)
        begin
                if (rst='1') then
                        present_state<=start;
                elsif(clk'event and clk='1')then
                       present_state<=next_state;
                 end if;
         end process;

 process(present_state,ready)
         begin
                case present_state is

                when start=>
                        daca_b<='0';
                        cs<='0';
                        wr<='1';
                        next_state<=read_adc;

                when read_adc=>
                       daca_b<='0';
                        cs<='0';
                        wr<='1';
                         if ready='1' then
                                next_state<=print_result;
                        else
                                next_state<=read_adc;
                          end if;

                 when print_result=>
                         daca_b<='0';
                         cs<='0';
                         wr<='1';
                         D<=A(7 downto 0);
                          next_state<=write_result;

                 when write_result=>
                          daca_b<='0';
                         cs<='0';
                         wr<='0';
                          next_state<=start;

                end case;
        end process;
end controller;
```

Στον παραπάνω κώδικα χρησιμοποιούμε τα πακέτα των βιβλιοθηκών std_logic_1164 και numeric_std για την εκτέλεση αριθμητικών πράξεων. Στην δήλωση της οντότητας (Entity) η οποία έχει το όνομα DAC_controller αντιστοιχούμε τι θέσεις εισόδου-εξόδου. Η είσοδος του DAC_controller είναι clk, rst, ready και A.

Από την πλευρά του DAC Controller οι *είσοδοι* του είναι οι εξής:
Clock: Είναι το σήμα του ρολογιού για τον συγχρονισμό του κυκλώματος (μορφή std_logic).
Reset: Είναι το σήμα για την επανεκκίνησης του κυκλώματος (μορφή std_logic).
Ready: Είναι το σήμα διακοπής που το έχουμε σε συγκεκριμένο βρόγχο του κυκλώματος, ενεργοποιείται όταν ο DAC έχει διαβάσει μια ψηφιακή τιμή και την έχει αποθηκεύσει στο εσωτερικό του. Απενεργοποιείται όταν η ψηφιακή τιμή έχει μετατραπεί σε αναλογική και έχει μεταφερθεί στην έξοδο. Αν λάβουμε ως σήμα την τιμή 1 τότε προχωρούμε στην επόμενη κατάσταση, αν λάβουμε 0 μένουμε στην παρούσα κατάσταση μέχρι να έρθει η τιμή 1 (μορφή std_logic).
A: 8-bit ψηφιακή είσοδος (μορφή std_logic_vector (7 downto 0)).

Οι *έξοδοι* του DAC Controller είναι chip select, write, DACa_DACb και D:
Chip Select (CS): Ακροδέκτης για την επιλογή μνήμης, έχουμε ενεργή τιμή σε κατάσταση LOW. Το Chip Select είναι ένα σήμα ελέγχου, το οποίο επιλέγει το συγκεκριμένο Chip μνήμης, στην περίπτωσή μας το Chip του ADC. (μορφή STD_LOGIC).
Write - Enable (WE): Ακροδέκτης που επιτρέπει την εγγραφή, έχουμε ενεργή τιμή σε κατάσταση LOW. Σήμα ελέγχου, που υποδηλώνει τον τύπο προσπέλασης (ενεργό = εγγραφή, ανενεργό = δεν εγγράφει) (μορφή STD_LOGIC).
DACa_DACb: Είναι το σήμα για την επιλογή της εξόδου από τη θύρα a ή b, ανάλογα ποιον DAC μετατροπέα διαλέξαμε από τους δυο που συμπεριλαμβάνονται μέσα στο Chip. (μορφή STD_LOGIC).
D: 8-bit ψηφιακή έξοδος (μορφή std_logic_vector (7 downto 0)). Στο Σχήμα 3.5 φαίνονται οι είσοδοι και οι έξοδοι του DAC Controller προς το FPGA.



Σχήμα 5.15 Είσοδοι και έξοδοι του DAC Controller

Στην συνέχεια έχουμε την δήλωση της αρχιτεκτονικής με όνομα CONTROLLER. Στην αρχή του τμήματος της αρχιτεκτονικής γίνεται η δήλωση των καταστάσεων και των σημάτων. Οι καταστάσεις είναι start, read_adc, print_result, write_result. Τα σήματα που περιλαμβάνει ο κώδικας μας είναι present_state και next_state. Στη συνέχεια αρχίζει η πρώτη διεργασία όπου μέσα στην λίστα ευαισθησίας είναι ο παλμός Clock και το Reset και ξεκινά η εντολή ακολουθιακής αντιστοίχισης if. Εάν το reset λάβει την λογική τιμή ένα τότε παραμένει στην αρχική μας κατάσταση start, διαφορετικά αν ο παλμός ρολογιού κάνει μετάβαση από το μηδέν στο ένα πηγαίνει στην επόμενη κατάσταση. Και έτσι τελειώνει η πρώτη διεργασία.

Ακολουθεί η δεύτερη διεργασία όπου μέσα στη λίστα ευαισθησίας υπάρχει το present_state και Read. Για την περιγραφή των πεπερασμένων μηχανών καταστάσεων χρησιμοποιήσουμε την ακολουθιακή εντολή case. Σε κάθε διαδοχικό παλμό ρολογιού το κύκλωμα βρίσκεται σε μία από τις δυνατές κατάστασης του. Η μετάβαση στην επόμενη κατάσταση συμβαίνει με βάση την τρέχουσα κατάσταση και την τιμή του σήματος εισόδου. Τα σήματα εξόδου εξαρτώνται από την τρέχουσα κατάσταση.

Στη πρώτη κατάσταση με το όνομα start το daca_b και το cs παίρνουν την τιμή 0, ενώ το wr λαμβάνει την τιμή 1 και πηγαίνουμε στην επόμενη κατάσταση που έχει όνομα read_adc. Σε αυτή την κατάσταση οι τιμές των σημάτων είναι ίδιες με την διαφορά ότι υπάρχει η ακολουθιακή τιμή if στην οποία όταν το ready γίνει ένα, τότε πηγαίνουμε στη επόμενη κατάσταση που ονομάζεται print_result διαφορετικά πηγαίνουμε στην κατάσταση read_adc. Η κατάσταση print_result το daca_b και τα cs είναι μηδέν, το wr είναι ένα, και η έξοδος των οκτώ ψηφίων D λαμβάνει την είσοδο, επίσης οκτώ ψηφίων A και προχωράμε στην τελευταία κατάσταση με όνομα write_result. Σε αυτή την κατάσταση το daca_b, το cs και το wr λαμβάνουν την τιμή μηδέν και προχωράμε στη αρχική ας κατάσταση η οποία είναι η start για να ξεκινήσει και πάλι η διαδικασία μετατροπής. Τελειώνει η δεύτερη διεργασία καθώς και ο κώδικας του controller για τον DACES52110.

5.3.4 Προσομοίωση για τον DACES52110

Η ίδια διαδικασία για την προσομοίωση επαναλαμβάνεται για ακόμα μία φορά για το κύκλωμα του DAC. Η προσομοίωση γίνεται από το Waveform File του Quartus II. Ανοίγοντας το εισάγουμε τις εισόδους και τις εξόδους του DAC_controller. Από την γραμμή εργαλείων ρυθμίζουμε τα σήματα σε 0 ή 1, όπως φαίνεται και στο Σχήμα 5.16 τα οποία εύκολα μπορούμε να δούμε στο κομμάτι της οντότητας του κώδικα. Στον κώδικα της αρχιτεκτονικής υπάρχουν τέσσερις καταστάσεις, άρα αντίστοιχα, μέσα στο περιβάλλον σχεδίασης της προσομοίωσης θα δημιουργήσουμε τουλάχιστον τέσσερις παλμούς ρολογιού. Στην περίπτωση μας χρησιμοποιήσαμε εφτά παλμούς ρολογιού για να εξασφαλίσουμε την εγκυρότητα των αποτελεσμάτων, με τη συνολική διάρκεια της προσομοίωσης να την έχουμε επιλέξει στα 1000 ns. Ο κάθε παλμός ρολογιού είναι 80 ns, άρα συνολικά έχουμε 560 ns για την ολοκλήρωση των ωρολογιακών παλμών.

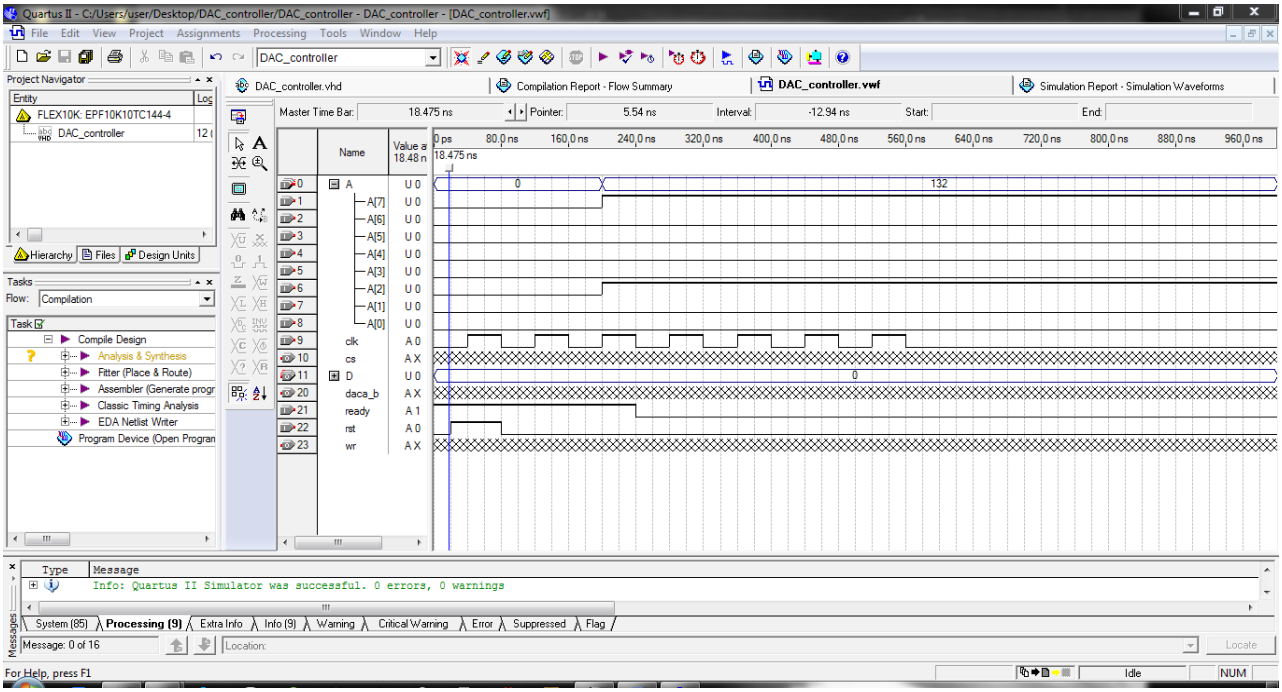
Ολοκληρώνουμε επιτυχώς την ανάλυση και σύνθεση (Analysis & Synthesis) και στη συνέχεια σχεδιάζουμε στο Waveform File τις κυματομορφές. Επιλέγουμε την Λειτουργική (Functional) και θέτουμε την συνολική διάρκεια της προσομοίωσης στο 1μs (End Simulation at 1μs) όπως πριν.

Σχεδιάζουμε τις εισόδους μας και μας εμφανίζει την παρακάτω Σχήμα 5.16.

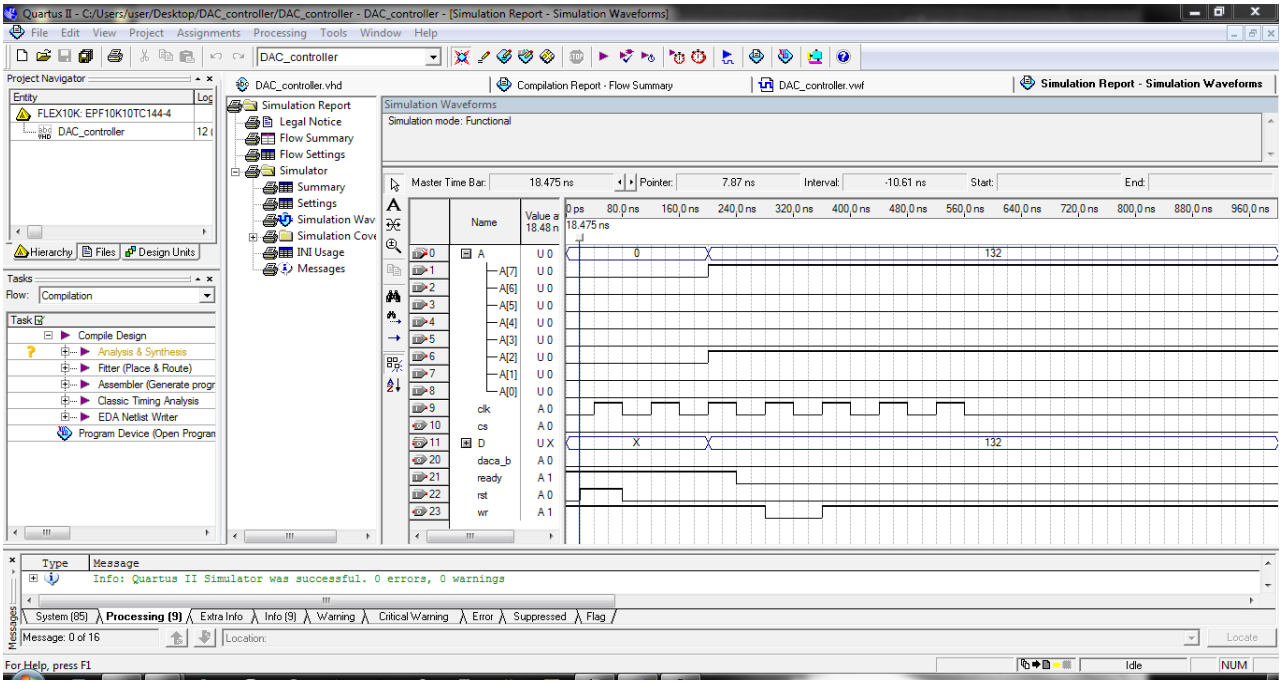
Από το διάγραμμα χρονισμού καταλαβαίνουμε ότι ο ρόλος του ready είναι για να σηματοδοτήσει την έναρξη της λειτουργίας του DAC, γι'αυτό το λόγο από 0ns έως 240ns του δίνουμε την τιμή 1. Για να γίνει η ενεργοποίηση της μετατροπής του κυκλώματος θα δώσουμε στο reset την τιμή 1 από 20ns έως 80ns. Η είσοδος A θα έχει την δεκαδική τιμή 0, από 0ns έως 200ns, ενώ από 200ns έως 1000ns θα δώσουμε την τυχαία δεκαδική τιμή 132.

Εφόσον έχουμε δώσει τιμές σε όλες τις εισόδους μας τρέχουμε την προσομοίωση από το

Menu Processing / Start Simulation για να διαπιστώσουμε αν γίνεται σωστά η λειτουργία του κώδικα από την εξαγωγή των αποτεσμάτων της προσομοίωσης. Όπως βλέπουμε και στο Σχήμα 5.17 ο κώδικας μας λειτουργεί σωστά.



Σχήμα 5.16 Προσομοίωση του DACES52110 μόνο με τιμές εισόδων και άγνωστες τις τιμές εξόδων



Σχήμα 5.17 Προσομοίωση του DAC ES52110

Το cs είναι ενεργό καθ' όλη την διάρκεια της προσομοίωσης, όπως και το daca_b, με λογική τιμή 0. Το wr γίνεται 0 από 280ns έως 360ns. Τέλος βλέπουμε ότι στην έξοδο μας D παίρνουμε την δεκαδική τιμή 132, δηλαδή την τιμή που είχαμε δώσει στην είσοδο A.

5.3.5 Αντιστοίχηση ακροδεκτών του DACES52110

Εφόσον έχει γίνει στοιχειώδη κατανόηση της αναπτυξιακής πλακέτας LP-2900 και του DACES52110, στον Πίνακα 5.3 παρουσιάζονται οι ακροδέκτες του διαμορφωμένου κυκλώματος FPGA που συνδέονται με διακόπτες εισόδου και leds απεικόνισης της εξόδου.

D/A → DACES52110

Code	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7
Device	ADC0804							
Pin	Pin 131(LS)	Pin 132	Pin 133	Pin 135	Pin 136	Pin 137	Pin 138	Pin 140(MS)

› DB0~DB7 also connect D0~D7 of LCD
› rst → Pin 63
› ready → Pin18
› clk_div_out → Pin 17

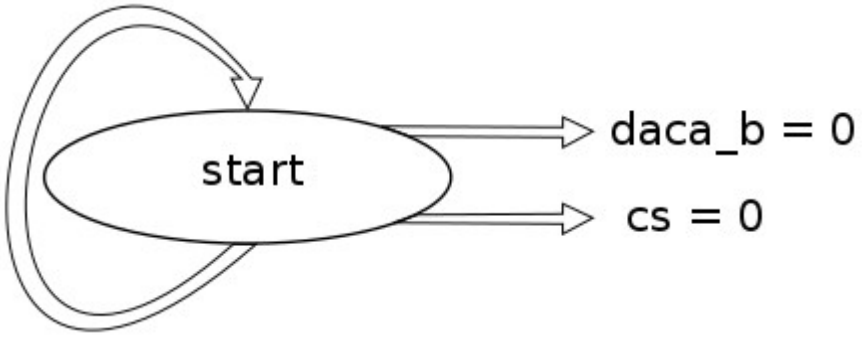
Code	/CS_DAC	/WR	/DACA	-	-	-
Device	DACES52110			-		
Pin	Pin 39	Pin 128	Pin 122	-	-	-

› /WR also connects the RW of LCD
› /DACA also connects the RS of LCD

Πίνακας 5.3 Αντιστοίχηση ακροδεκτών του DACES52110

5.3.6 Διάγραμμα Καταστάσεων του DAC σύμφωνα με το FPGA

Υλοποιώντας την τελική συνδεσμολογία στο FPGA LP 2900 για τον ADC και τον DAC παρατηρήσαμε ότι εκτός από τα pin του led που είναι τα ίδια, δηλαδή 131, 132, 133, 135, 136, 137, 138, 140 το RD του ADC και το WR του DAC χρησιμοποιούν το ίδιο pin, δηλαδή το 128. Αυτό έχει σαν αποτέλεσμα ο DAC να λειτουργεί σε μία μόνο κατάσταση (state) για την διαδικασία της μετατροπής, με όνομα start και όχι σε τέσσερις καταστάσεις που έχουμε δείξει παραπάνω. Στο κεφάλαιο 5.3.2 έχει σχεδιαστεί μία γενική μορφή του διαγράμματος καταστάσεων για τον DAC. Η τελική μορφή του DAC για το FPGA LP 2900 είναι αυτή που φαίνεται στο Σχήμα 5.18.



Σχήμα 5.18 Γενική μορφή του διαγράμματος καταστάσεων για τον DAC

Στην μια και μοναδική κατάσταση start οι τιμές τον σημάτων λαμβάνουν τις εξής τιμές, daca_b με λογική τιμή 0, και το cs επίσης με λογική τιμή 0. Μετά το πέρασμα της κατάστασης start επιστρέφουμε ξανά στην ίδια και επαναλαμβάνεται η ανάθεση τιμών στα σήματα.

5.4 Διαίρεση συχνότητας – Divider

Η διαίρεση συχνότητας (frequency divider ή clock divider) είναι ένα κύκλωμα όπου παίρνει σήμα εισόδου από μία συχνότητα Fin και παράγει ένα σήμα εξόδου Fout = $\frac{Fin}{n}$, όπου n είναι ένας ακέραιος αριθμός.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
-----------------------------------------
ENTITY divider IS
        PORT(clock, rst :in std_logic;
                Q : out std_logic);
        END divider;
-----------------------------------------
ARCHITECTURE behaviour OF divider IS
        signal count :std_logic_vector(3 downto 0);
BEGIN
        PROCESS (clock, rst)
                BEGIN
                        IF rst='1' THEN
                                count<="0000";
                        ELSIF clock'EVENT AND clock='1' THEN
                                count<=count+1;
                        END IF;
                END PROCESS;
                Q<=count(0);--divide frequency by 2
END behaviour;
```

Στο παραπάνω κώδικα χρησιμοποιούμε τα πακέτα std_logic_1164 και std_logic_unsigned για την εκτέλεση αριθμητικών πράξεων με μη προσημασμένους αριθμούς. Στη δήλωση της οντότητας (entity) αντιστοιχούμε τις θέσεις των εισόδων και τον εξόδων. Η είσοδος μας είναι το clock και το rst ενώ η έξοδος μας είναι το Q. Στο μέρος της αρχιτεκτονικής (architecture) δηλώνουμε ένα σήμα αριθμητή τεσσάρων ψηφίων και στο τμήμα της διεργασίας (process) βάζουμε στη λίστα ευαισθησίας τις εισόδους μας, δηλαδή το clock και το rst. Ξεκινά η εντολή

ακολουθιακής αντιστοίχησης if. Αν το rst είναι ίσο με ένα τότε ο αριθμητής θα γίνει μηδέν. Διαφορετικά η συνθήκη της if να αληθεύει αν έχει συμβεί μετάβαση ρολογιού και η μετάβαση αυτή είναι από το '0' στο '1'. Τέλος η έξοδος Q πέρνει τη τιμή του αριθμητή μας. Ουσιαστικά με αυτή τη διαδικασία η συχνότητα του ρολογιού μας διαιρείται με το δυο και έχει σαν αποτέλεσμα να έχουμε πιο αργό ωρολογιακό παλμό.

5.5 Δομική σχεδίαση ελεγκτών ADC και DAC και αντιστοίχηση ακροδεκτών

Η VHDL είναι μια γλώσσα που επιτρέπει την ιεραρχική σχεδίαση κυκλωμάτων. Ο χρήστης μπορεί να περιγράψει ένα σύνθετο κύκλωμα με βάση τα *υποκυκλώματα* που το αποτελούν. Τα υποκυκλώματα που συμπεριλαμβάνονται σε μια ανώτερη οντότητα (top-level entity) ονομάζονται *δομικά στοιχεία* (components). Τα στοιχεία είναι μικρότερα κυκλώματα, που έχουν ήδη περιγραφεί σε VHDL.

Όπως πολλά κυκλώματα έτσι και το δικός μας, χρησιμοποιήθηκαν τρία μικρότερα υποκυκλώματα. Ο κώδικας που περιλαμβάνει τα υποκυκλώματα ονομάζεται 'top_level' και τα δομικά στοιχεία μας είναι ο ADC_controller, DAC_controller που είδαμε την αναλυτική του λειτουργία στο κεφάλαιο 4 και ο Divider που είδαμε την αναλυτική του λειτουργία στο κεφάλαιο 5.

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

entity top_level IS
port (clk,rst,intr : in std_logic;
        dig_in: in std_logic_vector(7 downto 0);
        cs,rd,wr_out, pin33, pin36, clk_div_out, ready,CS_DACA, DACA, dac_wr:out  std_logic;
        leds :out std_logic_vector(7 downto 0));
end top_level;

ARCHITECTURE my_top_level of top_level IS
      SIGNAL r, n_wr,n_intr: std_logic;
      SIGNAL clk_div: std_logic;
      SIGNAL d,d2 :std_logic_vector(7 downto 0);--intermediate output

------COMPONENTS------------
COMPONENT DAC_controller
PORT(clk,rst,ready : in std_logic;
              A: in std_logic_vector(7 downto 0);
              D:out std_logic_vector(7 downto 0);
              cs,wr,daca_b :out std_logic);
END COMPONENT;

COMPONENT ADC_controller
PORT(clk,rst,intr : in std_logic;
              p: in std_logic_vector(7 downto 0);
              x :out std_logic_vector(7 downto 0);
              cs,rd,wr,ready :out std_logic);
END COMPONENT;
```

```
COMPONENT divider --clock divider using a simple counter
PORT(clock, rst :in std_logic;
                        Q  : out std_logic);
END COMPONENT;
-------------------------------------------------------------

BEGIN
        pin33<='1';--DE0
        pin36<='1';--DE1
        inst0: divider PORT MAP(clk, rst, clk_div);
        inst1:ADC_controller PORT MAP(clk_div,rst,intr,dig_in,d2,cs,rd,n_wr,r);
        inst2:DAC_controller PORT MAP (clk_div,rst,r,d2,d,CS_DACA,dac_wr, DACA);
        wr_out<=NOT(n_wr);--wr_out acts on DE2 of 74138. Y7 is low on 111
        clk_div_out<=clk_div;
        ready<=r;

--output result when ready-----

process (r)
begin
if (r'EVENT AND r='1') then
leds<=d;
end if;
end process;

END my_top_level;
```

Στο παραπάνω κώδικα για το υποκύλωμα μας χρησιμοποιούμε κάποια βασικά πακέτα της βιβλιοθήκης ieee, το std_logic_1164 που εισάγει τον τύπο δεδομένων std_logic και std_logic_vector και επιτρέπει τη χρήση περισσοτέρων τιμών σε ένα σήμα εκτός από τις απλές τιμές '0' και '1' και το numeric_std που εισάγει τους τύπους signed και unsigned, με βάση τον τύπο std_logic.

Στη δήλωση της οντότητας (entity), η οποία έχει το όνομα 'top_level' αντιστοιχούμε τις θέσεις των εισόδων και τον εξόδων. Οι είσοδοι μας είναι το clk, rst, intr και το dig_in. Το dig_in έχει εύρος 8 ψηφία και είναι η θέση των leds για το FPGA. Οι έξοδοι μας είναι το cs, rd, wr_out, , pin36, clk_div_out, ready, CS_DACA, DACA, dac_wr και τα leds. Τα leds έχουν εύρος επίσης 8 ψηφία και το wr_out είναι ακροδέκτης του FPGA σε αντίθεση με το wr που είναι του ADC .

Στη συνέχεια έχουμε τη δήλωση της αρχιτεκτονικής (architecture) του 'top_level' με όνομα 'my_top_level'. Στο τμήμα της αρχιτεκτονικής αρχικά δηλώνουμε τα σήματα, τα οποία αποδίδουν τιμές στα καλώδια του κυκλώματος και αντιπροσωπεύουν τις διασυνδέσεις ανάμεσα σε μονάδες του κυκλώματος. Τα σήματα αυτά είναι το r, n_wr, n_intr, clk_div, d και το d2. Το r παίρνει τη τιμή του ready και θα χρειαστεί για την σύνδεση για τον DAC. Το n_wr (not write) πρόκειται για τον ακροδέκτη wr στον οποίο έχουμε προσθέσει μία πύλη not, κάνοντας μία αντιστροφή για να πάρουμε στην έξοδο του FPGA τα αποτελέσματα που επιθυμούμε. Αυτός ο ακροδέκτης του FPGA είναι ο wr_out.

Έπειτα στο τμήμα της αρχιτεκτονικής ακολουθεί η δήλωση των υποκυκλωμάτων μας (components). Ουσιαστικά είναι το κομμάτι του κώδικα όπου 'καλεί' το κύκλωμα του ADC_controller, DAC_controller και divider για να ολοκληρωθεί το κύκλωμα μας. Αυτό επιτυγχάνεται με το να γράψουμε το κομμάτι του port από το entity του κάθε υποκυκλώματος, μέσα στα components του top_level.

Στη συνέχεια στο begin ξεκινάν οι εντολές που υλοποιούν την αρχιτεκτονική του κυκλώματος. Για να χρησιμοποιηθούν τα components που φτιάξαμε πρέπει να δημιουργήσουμε ένα στιγμιότυπο (instance) για κάθε ένα component που θέλουμε να χρησιμοποιήσουμε και να συνδέσουμε τις εισόδους και τις εξόδους με τα αντίστοιχα καλώδια. Κάθε δήλωση ενός υποκυκλώματος περιλαμβάνει ένα μοναδικό όνομα για να ξεχωρίζει από τα άλλα, το όνομα του υποκυκλώματος που χρησιμοποιούμε, καθώς και την αντίστοιχη των καλωδίων (των εισόδων, των εξόδων και των εσωτερικών σημάτων) στις πόρτες, μέσω της λίστας αντιστοίχησης port map. Η διασύνδεση λαμβάνει χώρα σύμφωνα με τη σειρά που οι θύρες δηλώνονται στην οντότητα της υπομονάδας.

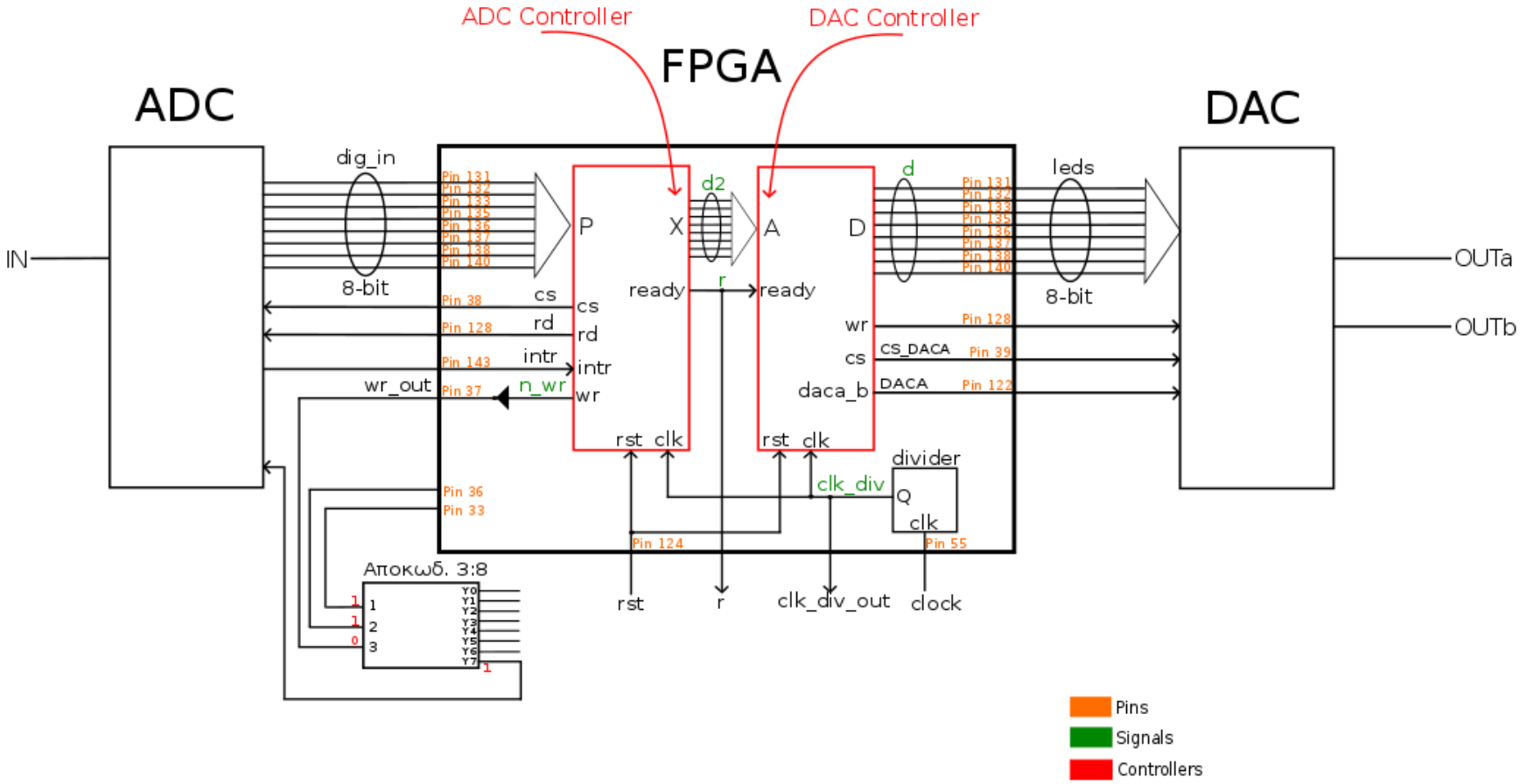
Στον κώδικά μας το πρώτο στιγμιότυπο έχει το όνομα “inst0” και είναι για την διασύνδεση με τον divider, για την διαίρεση συχνότητας. Η αντιστοίχηση καλωδίων στις πόρτες είναι η εξής:
clock → clk, rst → rst, Q → clk_div.

Το δεύτερο στιγμιότυπο έχει το όνομα “inst1” και είναι για την διασύνδεση με τον ελεγκτή του ADC, τον ADC_controller. Η αντιστοίχηση καλωδίων στις πόρτες είναι η εξής:
clk → clk_div, rst → rst, intr → intr, p → dig_in, x → d2, cs → cs, rd → rd, wr → n_wr, και ready → r.

Το τρίτο στιγμιότυπο έχει το όνομα “inst2” και είναι για την διασύνδεση με τον ελεγκτή του DAC, τον DAC_controller. Η αντιστοίχηση καλωδίων στις πόρτες είναι η εξής:
clk → clk_div, rst → rst, ready → r, A → d2, D → d, cs → CS_DACA, wr → dac_wr και daca_b → DACA.

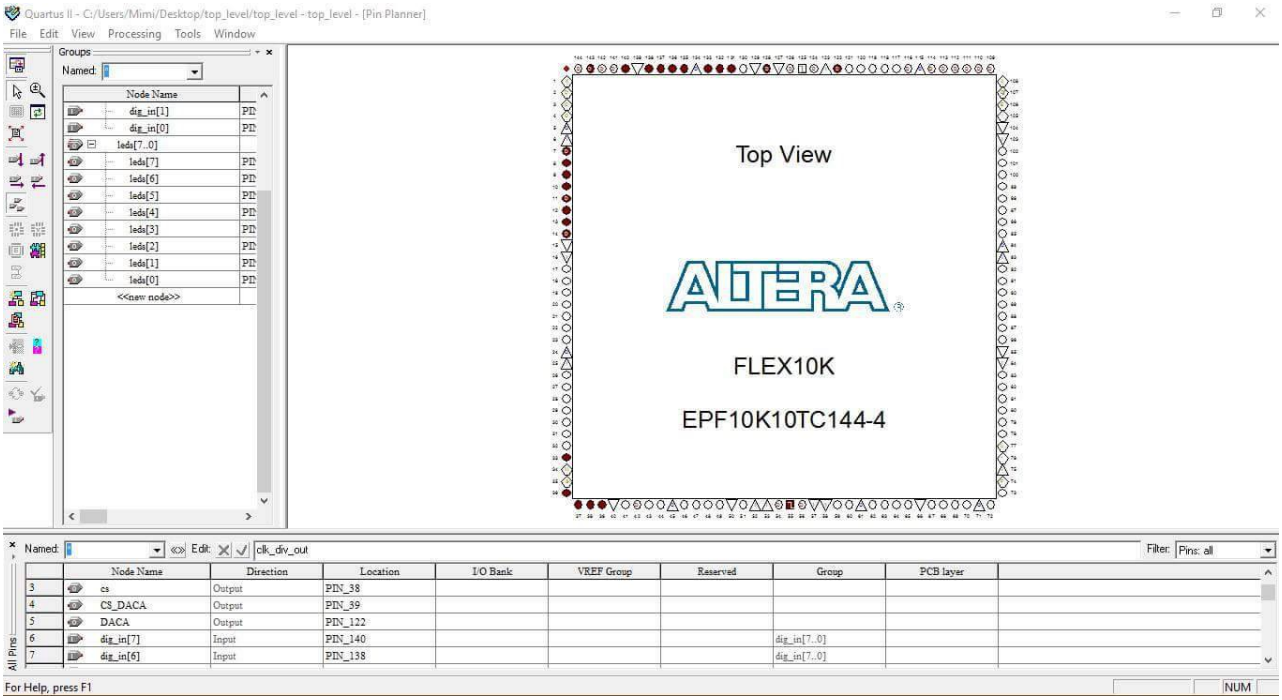
Το pin33 και το pin36 δέχονται τη τιμή '1'. Είναι δύο ακροδέκτες του FPGA όπου συνδέονται στον δεύτερο και πρώτο ακροδέκτη ενός αποκωδικοποιητή 3:8. Ο ακροδέκτης της τρίτης εισόδου συνδέεται με το σήμα wr_out. Ο αποκωδικοποιητής αυτός έχει 8 εξόδους, τα Y1, Y2, Y3, Y4, Y5, Y6, Y7. Θέλουμε το pin33 και το pin36 να έχουν την λογική τιμή “1”, γιατί αν το σήμα wr_out λάβει την τιμή “0” τότε στην έξοδο του αποκωδικοποιητή Y7 έχω την επιθυμητή τιμή “1”. Οι υπόλοιποι ακροδέκτες από τον αποκωδικοποιητή δηλαδή Y1, Y2, Y3, Y4, Y5, Y6 είναι αδιάφορες καταστάσεις για όλους τους διαφορετικούς συνδυασμούς στην είσοδο. Ο κώδικας της αρχιτεκτονικής τελειώνει στο τμήμα της διεργασίας (process) όπου βάζουμε στη λίστα ευαισθησίας το r. Ξεκινά η εντολή ακολουθιακής αντιστοίχησης if. Αν το r κάνει μετάβαση από το '0' στο '1' τα leds θα πάρουν τη τιμή της d.

Όλη η παραπάνω περιγραφή του κώδικα μπορεί εύκολα να γίνει πιο κατανοητή μέσα από το παρακάτω σχήμα 5.19



Σχήμα 5.19: Απεικόνιση σχεδίασης ελεγκτων

Εφοσον έχει ολοκληρωθεί η σύνταξη του κώδικα για τη δομική σχεδίαση των ελεγκτών για το FPGA το επόμενο βήμα είναι να αντιστοιχίσουμε τις εισόδους και τις εξόδους του κυκλώματός μας με συγκεκριμένους ακροδέκτες. Αναλυτικά έχουμε αναφέρει στο Κεφάλαιο 3.6.3 τη διαδικασία που ακολουθήσαμε για την αντιστοίχηση ακροδεκτών. Στο Σχήμα 5.20 βλέπουμε το αποτέλεσμα της αντιστοίχησης ακροδεκτών του ADC και DAC στο FPGA μέσω του προγράμματος Quartus.



Σχήμα 5.20 Το αποτέλεσμα της αντιστοίχησης ακροδεκτών του ADC και DAC στο FPGA

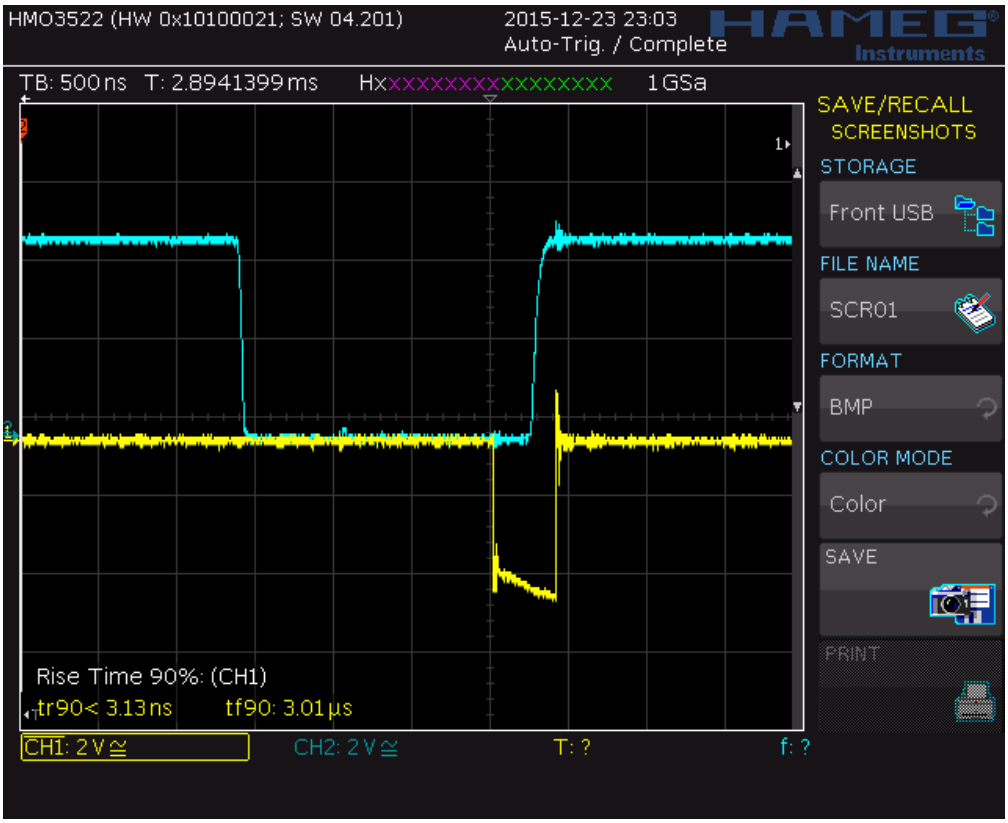
5.6 Αποτελέσματα του παλμογράφου

Για να διαπιστώσουμε πως το κύκλωμα μας λειτουργεί σωστά είδαμε τα αποτελέσματα του στον παλμογράφο. Ο παλμογράφος είναι ένα όργανο για τον έλεγχο ηλεκτρονικών κυκλωμάτων καθώς μας επιτρέπει να αναγνωρίσουμε διάφορα σήματα σε διαφορετικά σημεία ενός κυκλώματος, δημιουργώντας μια ορατή δισδιάστατη γραφική παράσταση των σημάτων. Ο οριζόντιος άξονας αντιπροσωπεύει το χρόνο, και κάθετος άξονας παρουσιάζει την συχνότητα.

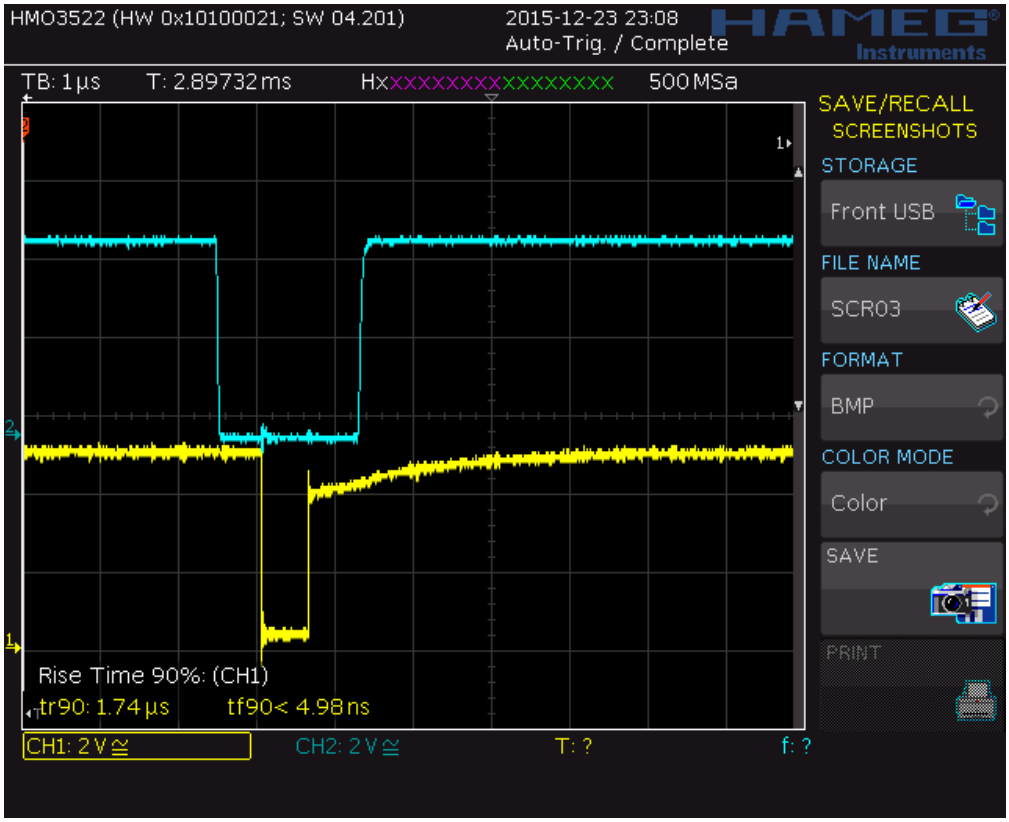
Στο Σχήμα 5.21 και 5.22 με γαλάζιο χρώμα έχουμε το interrupt και στα δύο σχήματα. Στο 5.21 με κίτρινο χρώμα είναι το write ενώ στο 5.22 είναι το read.

Όπως είδαμε και παραπάνω στο κεφάλαιο 5.2.2 που αφορά τα διαγράμματα καταστάσεων, όσο το cs1 είναι 0, οδηγούμε το wr1 σε 0, η ενέργεια αυτή έχει σαν αποτέλεσμα την επανεκκίνηση (reset) των καταχωρητών του μετατροπέα. Με την έναρξη του παλμού reset ο ακροδέκτης intr εξάγει λογικό 1. Και όντως αυτό φαίνεται μέσα από το σχήμα 5.21.

Στη συνέχεια με την ολοκλήρωση της μετατροπής ο ακροδέκτης intr θα ξαναγίνει 0. Όταν γίνει 0 τότε ο ελεγκτής παράγει σήμα read και ο ακροδέκτης read 1 (rd1) θα μεταβεί από λογικό ένα σε λογικό μηδέν με αυτόν τον τρόπο ενεργοποιούμε την ανάγνωση του καταχωρητή όπου αποθηκεύσαμε το αποτέλεσμα της μετατροπής και τα εξάγουμε στους ακροδέκτες DB0 έως DB7. Αυτή η λειτουργία φαίνεται στο σχήμα 5.22.



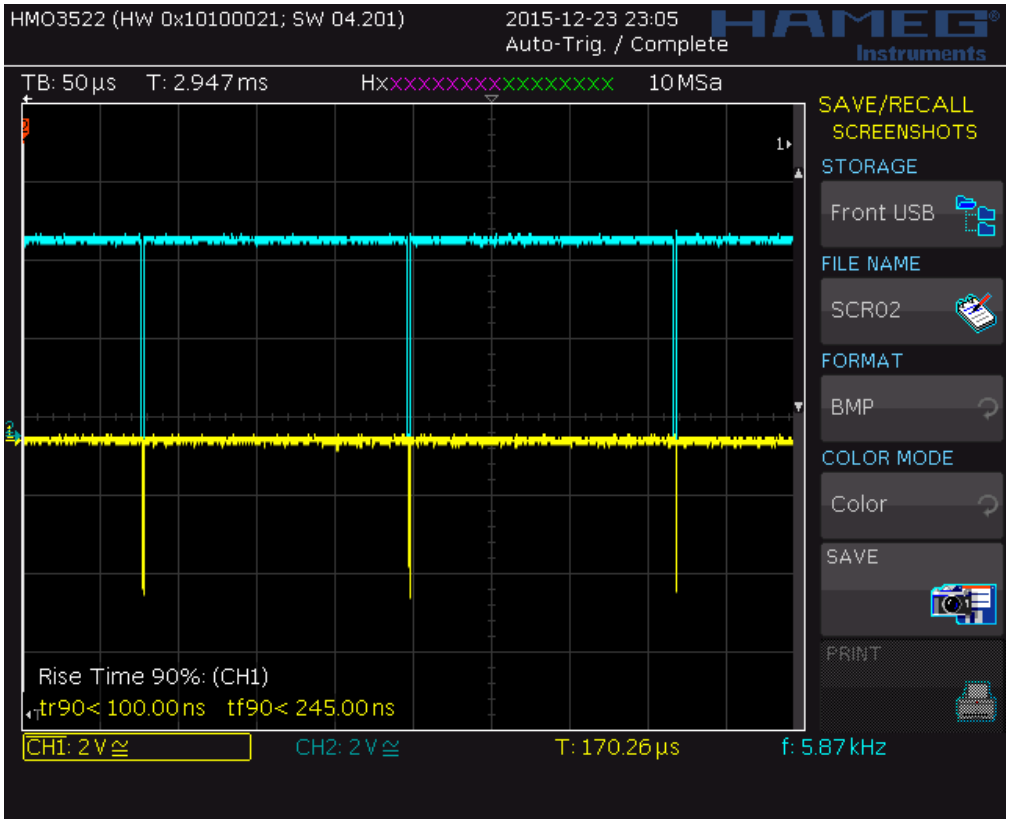
Σχήμα 5.21 Απεικόνιση σήματος interrupt και write



Σχήμα 5.22 Απεικόνιση σήματος interrupt και read

Στο Σχήμα 5.23 απεικονίζονται οι κυματομορφές του σήματος interrupt με γαλάζιο χρώμα και του write με κίτρινο χρώμα.

Αν λέγαμε ότι όλο αυτό που περιγράψαμε για την παραπάνω απεικόνιση θέλαμε να το δούμε από λίγο πιο "μακρυά" σαν μια σμίκρυνση τότε θα βλέπαμε την παρακάτω εικόνα. Όπου το interrupt γίνεται 0 γιατί δείχνει την ολοκληρώσει της μετατροπής, και κατόπιν γίνεται 1 γιατί έχω κατάσταση reset. Παραμένει 1 μέχρι να ολοκληρωθεί η μετατροπή. Δηλαδή βλέπουμε τον χρόνο τριών μετατροπών.



Σχήμα 5.23 Απεικόνιση σήματος interrupt και write

5.7 Συμπεράσματα και προτάσεις πτυχιακής εργασίας

Το παρακάτω κείμενο αναφέρεται στα συμπεράσματα που βγήκαν μετά την ολοκλήρωση της παρούσας πτυχιακής εργασίας. Επίσης υπάρχουν κάποιες προτάσεις γα την συνέχιση της .

Με τη παρούσα εργασία μελετήσαμε και κατανοήσαμε εις βάθος τη γλώσσα περιγραφής υλικού VHDL καθώς και τη σχεδίαση ελεγκτών αναλογικού σήματος σε ψηφιακό και ψηφιακού σήματος σε αναλογικό σε διάταξη FPGA.

Ο ελεγκτής του ADC0804 αλλά και ο ελεγκτής του DACES52110 λειτουργούν. Το συμπέρασμα αυτό προέκυψε καθώς τα αποτελέσματα των προσομοιώσεων που υλοποιήσαμε στο λογισμικό Quartus II είναι τα επιθυμητά, σύμφωνα με τα φύλλα δεδομένων του κατασκευαστή των ολοκληρωμένων κυκλωμάτων.

Στη διασύνδεση μετατροπέων στη συγκεκριμένη διάταξη FPGA λειτουργεί μόνο ο ADC0804 και αυτό γιατί ο κατασκευαστής του FPGA έχει ορίσει ο ακροδέκτης 128 να δέχεται το σήμα RD του ADC0804 αλλά και το σήμα WR του DACES52110.

Ένα ακόμα συμπέρασμα που βγήκε από την παρούσα εργασία είναι ότι κατά τη σχεδίαση οποιουδήποτε κυκλώματος είναι πολύ χρήσιμο πάνω στα σχέδια να αναγράφονται ονομασίες των διάφορων εισόδων, εξόδων και γενικότερα των καλωδίων που συνδέουν τα διάφορα σημεία του

κυκλώματος. Με αυτό τον τρόπο θα μπορούν να αποφευχθούν λάθη, ή τουλάχιστον θα είναι πολύ πιο εύκολο να διορθωθούν.

Μια βελτιστοποίηση που θα μπορούσε να γίνει στη παρούσα πτυχιακή εργασία είναι η διερεύνηση της διασύνδεσης σε διάταξη FPGA με διαφορετικούς μετατροπείς ADC και DAC. Όσο μεγαλύτερος είναι ο ψηφιακός αριθμός τόσο καλύτερα μπορεί να προσεγγίσει το αναλογικό σήμα. Ο ADC0804 και ο DACES52110 που χρησιμοποιήσαμε είναι των 8 bit. Μπορούν να χρησιμοποιηθούν μετατροπείς 12, 16 ή 32 bits. Για παράδειγμα θα μπορούσαν να μελετηθούν οι μετατροπείς αναλογικού σε ψηφιακό σήμα ADS1262 των 32bit, ADS1110 των 16 bit, AD7992 των 12bit και οι μετατροπείς από ψηφιακό σε αναλογικό σήμα DACPCM1795 των 32 bit, DAC7631 των 16 bit, DAC8512 των 12bit.

Βιβλιογραφία

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ
ΜΕΛΕΤΗ ΜΕΤΑΤΡΟΠΕΩΝ ΨΗΦΙΑΚΟΥ ΣΗΜΑΤΟΣ ΣΕ ΑΝΑΛΟΓΙΚΟ ΜΕ ΤΗ ΤΕΧΝΙΚΗ ΣΙΓΜΑ ΔΕΛΤΑ
ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΤΟΥ ΤΣΙΑΤΟΥΡΑ ΚΩΝΣΤΑΝΤΙΝΟΥ
ΕΠΙΒΛΕΠΩΝ : Κ. ΕΥΣΤΑΘΙΟΥ Πάτρα 2007

Σχεδίαση Ψηφιακών Συστημάτων με τη γλώσσα VHDL
Τρίτη έκδοση – Εκδόσεις Τζιόλα
Stephen Brown και Zvonko Vranesic
Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών Πανεπιστήμιο του Τορόντο

ΤΕΙ ΣΕΡΡΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΠΡΟΗΓΜΕΝΑ ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΣΗΜΕΙΩΣΕΙΣ ΕΡΓΑΣΤΗΡΙΟΥ ΣΤ΄ ΕΞΑΜΗΝΟ
Έκδοση 2η Ιωάννης Καλόμοιρος Είκουρος καθηγητής ΤΕΙ Σερρών
Συνεργασία: Μαδεμλής Ιωάννης Εργαστηριακός Συνεργάτης Σέρρες 2010

Circuit Design and Simulation with VHDL
second edition Volnei A. Pedroni
The MIT Press Cambridge, Massachusetts London, England

ΤΕΙ Σερρών
Τμήμα Πληροφορικής και Επικοινωνιών
Εισαγωγή στη γλώσσα VHDL
Ιωάννης Α. Καλόμοιρος

ΤΕΙ Σερρών
Τμήμα Πληροφορικής και Επικοινωνιών
Συστηματα συλλογης πληροφοριων και μετρησεων
Ιωάννης Α. Καλόμοιρος

Ανάλυση και σχεδίαση κυκλωμάτων ψηφιακής λογικής
Victor P. Nelson, **H. Troy Nagle**, **J. David Irwin**, **Bill D. Carroll**
μετάφραση: **Φώτης Κοκαβέσης**
επιμέλεια: **Μάνος Ρουμελιώτης**
Επίκεντρο, 2007

ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΤΕΡΕΑΣ ΕΛΛΑΔΑΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΣΗΜΕΙΩΣΕΙΣ
ΗΛΕΚΤΡΟΝΙΚΕΣ & ΗΛΕΚΤΡΙΚΕΣ ΜΕΤΡΗΣΕΙΣ
ΧΡΗΣΤΟΣ ΤΣΩΝΟΣ
ΛΑΜΙΑ 2010

Operating Systems Design and Implementation (3rd Edition) 3rd Edition
by Andrew S Tanenbaum (Author), Albert S Woodhull (Author)

ΤΕΙ Χαλκίδας
Τμήμα Ηλεκτρολογίας
Εργαστήριο Τεχνολογίας Μετρήσεων
Αρχές ψηφιοποίησης ADC DAC Conversion
Μέρος 1ο Ροβινσών Ισμίνογλου
Χαλκίδα 2008

Sites

https://en.wikipedia.org/wiki/Digital-to-analog_converter

https://es.wikipedia.org/wiki/Quartus_II

https://el.wikipedia.org/wiki/VHDL

https://el.wikipedia.org/wiki/FPGA

https://en.wikipedia.org/wiki/Frequency_divider

http://www.datasheetcatalog.com/datasheets_pdf/E/S/5/2/ES52110.shtml

http://www.engineersgarage.com/sites/default/files/ADC0804.pdf

https://el.wikipedia.org/wiki/Αναλογικό_σήμα

https://el.wikipedia.org/wiki/Ψηφιακό_σήμα

https://el.wikipedia.org/wiki/Ψηφιακά_ηλεκτρονικά