ΣΧΕΔΙΑΣΜΟΣ ΚΥΚΛΩΜΑΤΟΣ ΥΠΟΛΟΓΙΣΜΟΥ ΤΕΤΡΑΓΩΝΙΚΗΣ ΡΙΖΑΣ ΑΡΙΘΜΩΝ ΓΙΑ ΣΕΙΡΙΑΚΗ ΣΕ ΕΠΙΠΕΔΟ ΨΗΦΙΟΥ (DIGIT–SERIAL) ΕΠΕΞΕΡΓΑΣΙΑ ΣΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
ΛΑΜΠΡΟΥ Σ. ΜΠΙΣΔΟΥΝΗ

ΥΠΕΥΘΥΝΟΣ ΚΑΘΗΓΗΤΗΣ:
ΓΚΟΥΤΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΣΧΕΔΙΑΣΜΟΥ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ

ΠΑΤΡΑ – ΝΟΕΜΒΡΙΟΣ 1992

Πρόλογος

Η παρούσα εργασία έχει σαν στόχο το σχεδιασμό μιας σειριακής σε επίπεδο ψηφίου (digit-serial) αρχιτεκτονικής για τον υπολογισμό της τετραγωνικής ρίζας αριθμών. Τα συστήματα που χρησιμοποιούν σειριακή σε επίπεδο ψηφίου αρχιτεκτονική επεξεργάζονται περισσότερα από ένα bit εισόδου σε κάθε μονάδα του χρόνου. Τα συστήματα αυτά προσφέρονται για εφαρμογές μέσης ταχύτητας για τις οποίες οι σειριακές σε επίπεδο bit αρχιτεκτονικές είναι αργές, ενώ οι εξ' ολοκλήρου παράλληλες είναι πιο γρήγορες απ' όσο απαιτείται.

Στο πρώτο κεφάλαιο της διπλωματικής εργασίας γίνεται μια εισαγωγή στα VLSI συστήματα επεξεργασίας σημάτων, καθώς επίσης και η περιγραφή της διαδικασίας σχεδιασμού ενός VLSI συστήματος. Στο δεύτερο κεφάλαιο αναφέρονται χρήσιμα στοιχεία που αφορούν τους αριθμητικούς επεξεργαστές, καθώς επίσης και ο τρόπος επεξεργασίας σε αλυσιδωτά συστήματα ή συστήματα αγωγού (pipeline systems). Το τρίτο κεφάλαιο περιγράφει τεχνικές επεξεργασίας σημάτων που αφορούν σύγχρονα συστήματα, σειριακές σε επίπεδο bit (bit-serial) τεχνικές και σειριακές σε επίπεδο ψηφίου (digit-serial) τεχνικές. Επίσης, στο κεφάλαιο αυτό παρουσιάζεται μια μελέτη που αφορά την απόδοση των σειριακών σε επίπεδο ψηφίου αρχιτεκτονικών. Το τέταρτο κεφάλαιο περιγράφει τον αλγόριθμο υπολογισμού τετραγωνικής ρίζας δυαδικών αριθμών που υλοποιήθηκε. Στο πέμπτο κεφάλαιο παρουσιάζεται αναλυτικά ο σχεδιασμός του κυκλώματος υπολογισμού τετραγωνικής ρίζας αριθμών για σειριακή σε επίπεδο ψηφίου επεξεργασία. Γίνεται λεπτομερής αναφορά στα υποκυκλώματα που σχεδιάστηκαν, καθώς και στο βασικό κύκλωμα υπολογισμού. Στο παράρτημα γίνεται μια συνοπτική αναφορά στα εργαλεία Mentor Graphics που χρησιμοποιήθηκαν για την υλοποίηση του κυκλώματος.

Τέλος, θα ήθελα να ευχαριστήσω τους παρακάτω συνεργάτες μου που συνέβαλαν στην εκπόνηση της εργασίας αυτής: τον καθηγητή κ. Κωνσταντίνο Γκούτη για την ευκαιρία που μου έδωσε και τον υποψήφιο διδάκτορα κ. Δημήτριο Μετάφα για τη συνεργασία, τις υποδείξεις και το υλικό που μου προσέφερε.

Λάμπρος Μπισδούνης
Πάτρα, Νοέμβριος 1992

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1. VLSI ΣΥΣΤΗΜΑΤΑ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ (VLSI SIGNAL PROCESSING SYSTEMS)

Σύμφωνα με τις συμβάσεις που έχουν καθοριστεί μέχρι σήμερα τα "chips" που περιέχουν περισσότερα από 1000 στοιχεία ονομάζονται συστήματα μεγάλης κλίμακας ολοκληρώσεως (large scale integration, LSI) και εκείνα με πάνω από 10000 στοιχεία συστήματα πολύ μεγάλης κλίμακας ολοκληρώσεως (very large scale integration, VLSI). Στη καθημερινή όμως ορολογία ο όρος "VLSI" αναφέρεται σε συστήματα με 100000 ή περισσότερα στοιχεία. Μέχρι το 1985 είχαν κατασκευαστεί για το εμπόριο "chips" IC με περισσότερα από 1000000 τρανζίστορς.

'Ενα αποτέλεσμα της μεγάλης προόδου της κατασκευής κυκλωμάτων VLSI τεχνολογίας είναι ότι σε όλους τους τύπους επεξεργασίας πληροφοριών έχουμε αξιοσημείωτη μείωση του κόστους. Το αποτέλεσμα αυτό διαφαίνεται στο πεδίο των συστημάτων επεξεργασίας σημάτων πραγματικού χρόνου, όπου η συνεχής ροή των δεδομένων μαζί με την πολυπλοκότητα των αλγορίθμων επιβάλουν αυστηρές υπολογιστικές απαιτήσεις, οι οποίες συχνά δεν μπορούν να ικανοποιηθούν από κυκλώματα ή μηχανές γενικού σκοπού. Οι εφαρμογές της

επεξεργασίας σημάτων αφθονούν στο σύγχρονο κόσμο των ηλεκτρονικών. Οι πιο συνηθισμένες από αυτές συναντώνται στα τηλέφωνα, στα συστήματα ήχου και μουσικής, και στις τηλεοράσεις. Στις εμπορικές εφαρμογές που αφορούν τηλεπικοινωνίες και έλεγχο, έχει δημιουργηθεί μια πλατιά διαδεδομένη απαίτηση για κυκλώματα επεξεργασίας σημάτων με υψηλή ποιότητα και χαμηλό κόστος. Στο ανερχόμενο πεδίο των "έμπειρων" μηχανών η διασύνδεση ανθρώπου-μηχανής κυριαρχείται από λειτουργίες επεξεργασίας σημάτων όσον αφορά την ομιλία αλλά και την εικόνα. Επίσης εφαρμογή επεξεργασίας σημάτων έχουμε και στα στρατιωτικά ραντάρ. Η ψηφιακή επεξεργασία σημάτων μέσω VLSI συστημάτων, γίνεται με διάφορες τεχνικές που θα αναφερθούν στο κεφάλαιο 3, οι οποίες επιχειρούν μια ελαστική επιλογή μεταξύ της ποιότητας και του κόστους. Το συνολικό κόστος αυτής της ισορροπίας (tradeoff) συνεχώς μειώνεται με την εξέλιξη της τεχνολογίας κατασκευής συστημάτων.

Τα τελευταία χρόνια που η σημασία των εφαρμογών ψηφιακής επεξεργασίας σημάτων είναι μεγάλη υπάρχει μια αυξανόμενη έλξη προς την VLSI υλοποίηση των συστημάτων επεξεργασίας σημάτων. Ο ρυθμός δειγμάτων (sample rate), είναι διαφορετικός για κάθε εφαρμογή και κυμαίνεται από 8 KHz για συστήματα ομιλίας, τηλεπικοινωνίες και επεξεργαστές εικόνας, μέχρι μερικές δεκάδες ή εκατοντάδες MHz για επεξεργαστές ραντάρ πραγματικού χρόνου. Υπάρχουν περιπτώσεις όπου πρέπει να υλοποιηθούν αλγόριθμοι οι οποίοι χρειάζονται χιλιάδες υπολογιστικά βήματα για να εκτελεστούν τα οποία οδηγούν σε υπολογιστικές απαιτήσεις που ξεπερνούν το ένα εκατομμύριο αριθμητικών λειτουργιών ανά δευτερόλεπτο. Οι απαιτήσεις αυτές αντιμετωπίζονται γενικά με συστήματα νέων αρχιτεκτονικών τα οποία εκμεταλέυονται τις δυνατότητες ταυτόχρονης εκτέλεσης που υπάρχουν σε πολλούς αλγόριθμους. Η VLSI τεχνολογία προσφέρει νέες δυνατότητες για την υλοποίηση τέτοιων αρχιτεκτονικών. Το επικρατέστερο μέτρο κόστους στην ανάπτυξη VLSI συστημάτων είναι η πολυπλοκότητα της σχεδίασης. Με την είσοδο της VLSI τεχνολογίας έχει επέλθει μεγάλη εξέλιξη στη μεθοδολογία σχεδίασης και στα χρησιμοποιούμενα εργαλεία (tools), με αποτέλεσμα την αύξηση της παραγωγικής ικανότητας και την ευκολία στην υλοποίηση πολύπλοκων αλγορίθμων.

Για την κατασκευή ενός πλήρους VLSI συστήματος επεξεργασίας σημάτων απαιτείται εμπειρία στο πεδίο της ανάπτυξης αλγορίθμων, στο πεδίο της αρχιτεκτονικής συστημάτων, καθώς και στο πεδίο της διαδικασίας του φυσικού σχεδιασμού (layout). Η σχεδίαση ειδικών VLSI "chips" (custom design), δεν είναι η μόνη προσσέγγιση για την υλοποίηση επεξεργαστών σημάτων πραγματικού χρόνου, αλλά έχει το πλεονέκτημα να εξασφαλίζει αριθμητικά στοιχεία και στοιχεία μνήμης φτιαγμένα με ακρίβεια ώστε να ανταπεξέρχονται στις απαιτήσεις της κάθε εφαρμογής. Τα πλεονεκτήματα αυτής της επιλογής έχουν αντισταθμιστεί στο παρελθόν από το κόστος που κατά παράδοση σχετίζεται με την μεγάλη εξέλιξη στις κλίμακες του χρόνου (timescales) και με τις πολλαπλές σχεδιαστικές επαναλήψεις.

Το υψηλό κόστος λοιπόν καθιστά την σχεδίαση ειδικών "chips" (custom design) μη ελκυστική για εφαρμογές μέσου και χαμηλού εύρους αγοράς και για προιόντα ταχείας εξέλιξης. Για το σχεδιασμό VLSI επεξεργαστών σημάτων είναι τεχνολογικά απαραίτητη η γνώση αρχιτεκτονικών αρχών, η χρήση CAD εργαλείων (CAD tools), καθώς και βιβλιοθήκες βασικών λειτουργικών στοιχείων (cell libraries). Το εύρος της γνώσης που απαιτείται για ένα ικανοποιητικό σχεδιασμό VLSI συστημάτων επεξεργασίας σημάτων περιλαμβάνει την ανάπτυξη αλγορίθμων, το λογικό σχεδιασμό συστημάτων και κυκλωμάτων, το φυσικό σχεδιασμό (layout) κυκλωμάτων καθώς και την εξομοίωση (simulation) των συστημάτων που έχουν σχεδιαστεί.

1.2 ΔΙΑΔΙΚΑΣΙΑ ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΚΑΤΑΣΚΕΥΗΣ VLSI ΣΥΣΤΗΜΑΤΩΝ

Η διαδικασία ενός VLSI σχεδιασμού συνδέει ένα ευρύ φάσμα επιστημονικών κλάδων όπως είναι η Φυσική, η Χημεία, η Ηλεκτρολογία και η επιστήμη των υπολογιστών. Λόγω της ποικιλίας των εργασιών που απαιτούνται κατά τη διάρκεια της κατασκευής ενός "chip", είναι απαραίτητος ο διαχωρισμός της διαδικασίας σε κάποια βασικά στάδια, τα οποία θα αναφερθούν παρακάτω. Πρέπει όμως να μην ξεχνιέται ότι δεν πρόκειται για σαφή διαχωρισμό μεταξύ των σταδίων αυτών και ότι δεν πρόκειται για σαφή διαδοχή στη διαδικασία σχεδιασμού. Τα στάδια αυτά αλληλεπιδρούν, το καθένα καθορίζει το επόμενο

και αναπροσαρμόζει το προηγούμενο. Δεν είναι λίγες οι φορές που για να συμπληρωθεί η διαδικασία χρειάζονται αρκετά περάσματα από το πρώτο προς το τελευταίο και αντίστροφα.

Τα οκτώ στάδια της διαδικασίας είναι τα εξής:

1. Προδιαγραφές του συστήματος:

Εδώ ορίζονται οι στόχοι και οι περιορισμοί του συστήματος που πρόκειται να σχεδιαστεί δηλαδή τι θα κάνει το σύστημα, ποιά είναι τα κριτήρια για την βελτιστοποίησή του, ποιές οι απαιτήσεις για την ταχύτητά του, το μέγεθός του, τι ενέργεια επιτρέπεται να καταναλώνει και άλλα παρόμοια στοιχεία.

2. Λειτουργικός σχεδιασμός:

Το συνολικό σύστημα πρέπει να διαιρεθεί σε υπομονάδες δηλαδή σε στοιχεία τα οποία θα έχουν διαφορετική λειτουργία και θα τα ονομάζουμε λειτουργικά στοιχεία (operators) του συστήματος. Στον λειτουργικό σχεδιασμό αποφασίζονται οι αλληλεπιδράσεις ή "λειτουργικές σχέσεις" μεταξύ των λειτουργικών στοιχείων. Επίσης η εργασία στο στάδιο αυτό του σχεδιασμού έχει να κάνει με τα χαρακτηριστικά μιάς γενικής λύσης για κάποιο ορισμένο πρόβλημα, τα οποία μετατρέπονται σε όρους μιας αλγοριθμικής διαδικασίας, καθώς επίσης και με τα χαρακτηριστικά των βασικών λειτουργικών στοιχείων και των ενδοσυνδεσεών τους, που οδηγούν στην υλοποίηση του αλγόριθμου. Στο στάδιο του λειτουργικού σχεδιασμού γίνεται επίσης η οργάνωση της ροής δεδομένων, ο καθορισμός του τύπου των λειτουργικών στοιχείων και των γεωμετρικών χαρακτηριστικών τους και ο καθορισμός της απαιτούμενης επικοινωνίας μεταξύ των λειτουργικών στοιχείων. Αν η λειτουργία είναι αρκετά πολύπλοκη μπορεί το συνολικό σύστημα να αναλυθεί σε υποσυστήματα και η συνολική διαδικασία να επαναληφθεί ιεραρχικά. Στη πράξη οι υπολειτουργίες που εκτελούν τα διάφορα λειτουργικά στοιχεία δεν είναι πλήρως ανεξάρτητες αφού οι μεταβλητές που παρουσιάζονται στη σχεδίαση είναι συνήθως κοινές σε όλες τις υπολειτουργίες. Ο συνδιασμός των δύο πρώτων σταδίων (δηλαδή ο καθορισμός των προδιαγραφών και ο λειτουργικός σχεδιασμός του συστήματος

θεωρούμενα μαζί) είναι αυτό που καλείται αρχιτεκτονικός σχεδιασμός.

3. Λογικός σχεδιασμός:

Το στάδιο του λογικού σχεδιασμού δίνει μία όψη του τελικού κυκλώματος στην οποία τα ρεύματα και οι τάσεις περιορίζονται σε διάκριτα επίπεδα. Στα ψηφιακά κυκλώματα τα δύο επίπεδα (στάθμες) των σημάτων που επιτρέπονται είναι εκείνα που παριστάνονται με το λογικό 1 και το λογικό 0. Τα κυκλώματα στο στάδιο αυτό παριστάνονται με λογικά διαγράμματα, τα οποία αποτελούνται από διακόπτες, αντιστροφείς (inverters) και λογικές πύλες που χειρίζονται λογικά σήματα. Εδώ ο σχεδιασμός του συστήματος ανάγεται σε εκφράσεις "Bool" ή σε μία αναπαράσταση μιας μηχανής πεπερασμένων καταστάσεων (finite state machine). Το βασικό στοιχείο στο στάδιο του λογικού σχεδιασμού, είναι η ορθότητα των συνολικών λογικών λειτουργιών του κυκλώματος από την άποψη των στοιχειωδών λειτουργιών που εκτελούνται από τα λογικά στοιχεία σε μια συγκεκριμένη λογική ακολουθία.

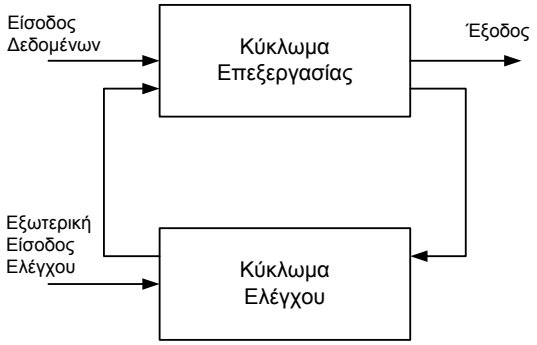
4. Σχεδιασμός κυκλωμάτων:

Τώρα το λογικό κύκλωμα μετατρέπεται σε ηλεκτρονικό κύκλωμα. Εδώ ορίζονται ποσότητες όπως η ταχύτητα, η κατανάλωση ενέργειας, το σχήμα και το μέγεθος του "chip". Με αρκετά καλή προσσέγγιση της πραγματικότητας, μπορεί να πει κανείς ότι το τελικό προιόν του σχεδιασμού κυκλωμάτων είναι ένα σύνολο από λειτουργικά κομμάτια ή στοιχεία (operators). Τα λειτουργικά στοιχεία έχουν τερματικά (terminals, pins) που αναλαμβάνουν το ρόλο σύνδεσης με το περιβάλλον τους δηλαδή ρόλο I/O. Ας σημειωθεί ότι γενικά τα κυκλώματα μπορούν να διαχωριστούν σε τρία είδη: ειδικά (custom design), συνήθη (standard cell) και διατάξεις πυλών (gate-array).

5. Σχεδιασμός συστήματος:

Η βασική εργασία εδώ είναι η σύνδεση των βασικών υποσυστημάτων μεταξύ τους καθώς επίσης η διασύνδεση και η επικοινωνία με τον εξωτερικό κόσμο. Στο στάδιο αυτό αναπτύσσονται διάφορες στρατηγικές διασύνδεσης, στρατηγικές

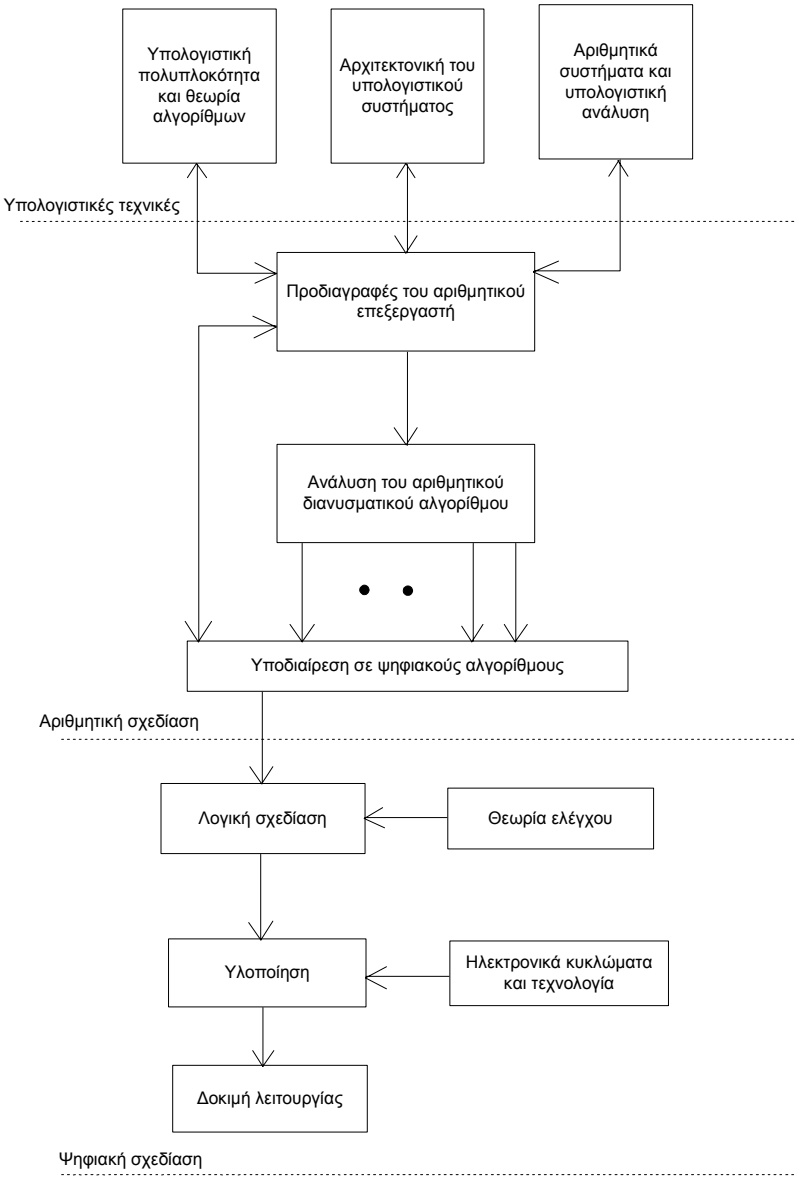
δρομολόγησης πληροφοριών και στρατηγικές ελέγχου. Το στάδιο αυτό της σχεδίασης είναι ισχυρά συνδεδεμένο με την διαδικασία η οποία είναι μια προσπάθεια συνδιασμού του σχεδιασμού συστημάτων με την τοποθέτηση και διασύνδεση βασικών υποσυστημάτων. Η διαδικασία αυτή είναι βασισμένη στις αρχικές εκτιμήσεις που αφορούν το μέγεθος του συστήματος και τις θέσεις των σημείων εισόδου/εξόδου. Για τη συστηματική σχεδίαση ψηφιακών συστημάτων θεωρείται ότι το σύστημα αποτελείται από δύο επιμέρους κυκλώματα: το κύκλωμα επεξεργασίας των πληροφοριών (δεδομένων) και το κύκλωμα ελέγχου. Το κύκλωμα ελέγχου δημιουργεί τα κατάλληλα σήματα για την πραγματοποίηση της επεξεργασίας όπως φαίνεται στο σχήμα 1.1. Τα βήματα των εργασιών στο κύκλωμα επεξεργασίας ως και η ακολουθία των σημάτων ελέγχου μπορούν να περιγραφούν βάσει των δεδομένων του προβλήματος από έναν υλικό αλγόριθμο (hardware algorithm).



Σχήμα 1.1: Διάγραμμα ψηφιακού συστήματος

Η υλοποίηση του τμήματος επεξεργασίας μπορεί να γίνει με κυκλώματα όπως καταχωρητές, μετρητές, πλήρεις αθροιστές κ.α. Για το τμήμα ελέγχου μπορούν να χρησιμοποιηθούν ακολουθιακά κυκλώματα. Επίσης η εκλογή των κατάλληλων κυκλωμάτων για την υλοποίηση του τελικού συστήματος είναι συνάρτηση της πολυπλοκότητάς του, καθώς και της εμπειρίας του σχεδιαστή. Στα υπολογιστικά συστήματα, οι βασικές μονάδες επεξεργασίας ήταν ανέκαθεν οι αριθμητικοί επεξεργαστές. Οι αριθμητικοί επεξεργαστές χρησιμοποιούνται για την εκτέλεση αριθμητικών λειτουργιών και για την παροχή αριθμητικών λύσεων σε υπολογιστικά προβλήματα. Από την έλευση της μέσης/μεγάλης κλίμακας ολοκλήρωσης ηλεκτρονικών κυκλωμάτων (MSI/LSI), όλο και πιο πολλοί περίπλοκοι αριθμητικοί επεξεργαστές έχουν γίνει κλασσικά κομμάτια υλικού για

τα σημερινά ψηφιακά υπολογιστικά συστήματα μεγάλης απόδοσης. 'Ενας σύγχρονος ψηφιακός υπολογιστής είναι εφοδιασμένος με ένα αριθμό από αριθμητικούς επεξεργαστές που χειρίζονται δεδομένα διαφορετικών τύπων ή λύνουν ειδικές αριθμητικές συναρτήσεις για γενικές ή ειδικές εφαρμογές. Η επιλογή ενός κατάλληλου αριθμητικού συστήματος αφορά την σχεδίαση της αρχιτεκτονικής του υπολογιστή και τις εφαρμογές που θα προγραμματιστούν.



Σχήμα 1.2: Διάγραμμα διαδικασιών κατασκευής υπολογιστικού συστήματος

Η αριθμητική σχεδίαση περιλαμβάνει την ανάπτυξη ενός αριθμητικού αλγόριθμου και την λογική υλοποίησή του. Στο πεδίο της αρχιτεκτονικής υπολογιστών, η αριθμητική σχεδίαση αλληλεπιδρά με την απόδοση της υλοποίησης των αριθμητικών λειτουργιών ενώ στο πεδίο της αριθμητικής ανάλυσης σχετίζεται με την ορθότητα των προσσεγγίσεων στους αριθμητικούς υπολογισμούς. Οι διαδικασίες που συμμετέχουν στην κατασκευή ενός αριθμητικού υπολογιστικού συστήματος και οι αλληλεπιδράσεις μεταξύ αυτών φαίνονται στο σχήμα 1.2. Μπορούμε να πούμε ότι η αριθμητική σχεδίαση είναι το σημαντικότερο μέρος της κατασκευής ενός υπολογιστικού συστήματος.

6. Φυσικός σχεδιασμός των κυκλωμάτων (layout):

Η διασύνδεση μεταξύ του φυσικού κόσμου και του ηλεκτρονικού κόσμου γίνεται στο στάδιο του φυσικού σχεδιασμού (layout layer), το οποίο είναι ένας σύνδεσμος μεταξύ του κυκλώματος και της διαδικασίας κατασκευής του. Το κύκλωμα δημιουργείται πάνω στην επιφάνεια ενός υποστρώματος απόπυρίτιο (Si) με ενδοσυνδέσεις από υλικά τριών επιπέδων: ένα αγώγιμο επίπεδο μετάλλου που χρησιμοποιείται για ηλεκτρικές συνδέσεις και δύο επίπεδα ημιαγωγών που είναι το πολυκρυσταλλικό πυρίτιο (polysilicon) και η διάχυση (diffusion), τα οποία χρησιμοποιούνται για την δημιουργία στοιχείων όπως διακόπτες, αντιστροφείς, πύλες καθώς και για ηλεκτρικές συνδέσεις. Τα επίπεδα αυτά απομονώνονται το ένα από το άλλο μέσω ειδικού μονωτικού υλικού. Οι ηλεκτρικές συνδέσεις μεταξύ των επιπέδων, επιτυγχάνονται με ειδικές δομές που λέγονται επαφές (contacts). Τα υλικά ημιαγωγών μπορούν να είναι η-τύπου δηλαδή τα ηλεκτρόνια να είναι τα σωματίδια που συντηρούν την ηλεκτρική αγωγή ή τύπου-ρ όπου οι οπές είναι τα σωματίδια που συντηρούν την ηλεκτρική αγωγή. Το βασικό στοιχείο του σταδίου του φυσικού σχεδιασμού είναι η βελτιστοποίηση της περιοχής στην οποία εκτείνεται το κύκλωμα. Η κατασκευή του φυσικού σχεδίου κυκλωμάτων βασίζεται σε κάποιους κανόνες σχεδίασης (design rules) οι οποίοι πρέπει να τηρούνται για την σωστή λειτουργία του κυκλώματος.

Γενικά το πρόβλημα του φυσικού σχεδιασμού (layout) είναι να τοποθετηθούν τα λειτουργικά κομμάτια σε διάφορα μέρη του "chip" και να συνδεθούν τα

απαραίτητα τερματικά ώστε να εξασφαλιστεί η απαιτούμενη επικοινωνία μεταξύ των λειτουργικών κομματιών.

7. Επαλήθευση του σχεδιασμού, έλεγχος και διόρθωση σφαλμάτων:

Στο στάδιο αυτό ελέγχεται και επαληθέυεται ότι όλοι οι κανόνες σχεδιασμού καθώς και οι λειτουργικές σχέσεις ικανοποιούνται. Το "chip" στη συνέχεια δοκιμάζεται με προσομοίωση (simulation) και αν χρειαστεί διορθώνεται.

8. Κατασκευή, έλεγχος πρωτοτύπου και παραγωγή:

Το στάδιο αυτό έχει να κάνει με τον φυσικό κόσμο των ημιαγωγών. Μία σύνθετη χημική τεχνολογία έχει αναπτυχθεί για να εκτελέσει τις φυσικές διαδικασίες που χρειάζονται για την κατασκευή των διαφόρων στοιχείων. Στη συνέχεια τα στοιχεία αυτά τοποθετούνται "μαζί" δημιουργόντας κυκλώματα, όπου ρέουν οι φορείς των πληροφοριών.

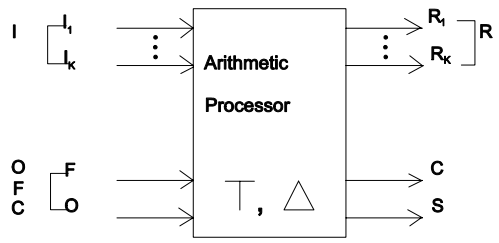
Τα βασικά στοιχεία στο στάδιο αυτό είναι η ταχύτητα του κυκλώματος που καθορίζεται από τις παραμέτρους του κυκλώματος, η πτώση τάσης στις γραμμές σύνδεσης, η ολική κατανάλωση ισχύος του κυκλώματος η οποία καθορίζει την θερμότητα που αναπτύσσεται και η ορθότητα των κυματομορφών ρεύματος και τάσης που ελέγχουν τη ροή της πληροφορίας μέσα στο κύκλωμα.

Κεφάλαιο 2

Χρήσιμα στοιχεία για τους αριθμητικούς επεξεργαστές

2.1 ΓΕΝΙΚΑ

Οι αριθμητικοί επεξεργαστές είναι τα τμήματα ενός ψηφιακού συστήματος που εκτελούν τις αριθμητικές λειτουργίες. Μία γενική περιγραφή των ψηφιακών αριθμητικών επεξεργαστών δίδεται πιό κάτω. Η μορφή τους περιγράφεται από την άποψη του σχεδιαστή. Ο σχεδιαστής πρέπει να εξετάσει τους υπάρχοντες αλγόριθμους, την αρχιτεκτονική του συστήματος, και την υλοποίηση της λογικής του επεξεργαστή. Επίσης ο σχεδιαστής πρέπει να διαχωρίσει την αριθμητική σχεδίαση από την λογική σχεδίαση, σε δύο διαδοχικές βαθμίδες. Αυτή η προσέγγιση των δύο βαθμίδων, κάνει τον αλγόριθμο ανεξάρτητο από πιθανή αλλαγή της τεχνολογίας κατασκευής του επεξεργαστή.



Σχήμα 2.1: Ψηφιακό αριθμητικό σύστημα

Το Σχήμα 2.1 δείχνει ένα αριθμητικό επεξεργαστή, ο οποίος περιγράφεται από οκτώ διανύσματα (vectors):

A = <I,R,O,F,C,S,T,Δ>, όπου:

1.I={I_1,I_2,..,I_k} είναι ένα σύνολο από ορίσματα εισόδου (input operands). Κάθε όρισμα εισόδου I_i είναι ένα ψηφιακό διάνυσμα (digital vector) σε ένα προκαθορισμένο αριθμητικό σύστημα παράστασης (number representation).

2.R={R_1,$R_{2,..,R}$t} είναι το σύνολο των ψηφιακών διανυσμάτων των αποτελεσμάτων της εξόδου (output results).

3. O.F.C (operation-format code), είναι ο κώδικας τύπου λειτουργίας, ο οποίος είναι ένα διάνυσμα, που αποτελείται από ένα αριθμό ζευγών τύπου λειτουργίας (O,F): OFC=(O_1,F_1; O_2,F_2;..;O_r,F_r;)

όπου O_i είναι το σύνολο των λειτουργικών μονάδων (operators), και F_i είναι το σύνολο των τύπων δεδομένων (data formats).

4. C είναι το διάνυσμα κατάστασης (condition vector) και S είναι το διάνυσμα ιδιομορφίας (singularity vector), τα οποία αναγνωρίζουν ειδικές καταστάσεις που σχετίζονται με το αποτέλεσμα (π.χ πρόσημο αποτελέσματος), ή υποδεικνύουν ιδιόμορφες καταστάσεις (π.χ υπερχειλιση (overflow)).

5. T είναι ο χρόνος λειτουργίας (operation time).

6. Δ είναι το διάνυσμα του ελέγχου ακρίβειας (precision control).

Η μαθηματική μορφή ενός ψηφιακού αριθμητικού επεξεργαστή που παρουσιάστηκε, υλοποιείται μέσω της σχεδίασης δύο βαθμίδων που περιγράφτηκε νωρίτερα. Μετά την εύρεση ενός αποτελεσματικού αλγόριθμου, η σημαντικότερη εργασία ενός σχεδιαστή, είναι να καταστρώσει όλα αυτά που περιγράφτηκαν στο Σχήμα 2.1 σε μία μορφή, η οποία θα μπορεί εύκολα να "μεταφραστεί" σε επίπεδο λογικού σχεδιασμού.

2.2 ΣΥΣΤΗΜΑΤΑ ΠΑΡΑΣΤΑΣΗΣ ΑΡΙΘΜΩΝ ΣΤΟΥΣ ΑΡΙΘΜΗΤΙΚΟΥΣ ΕΠΕΞΕΡΓΑΣΤΕΣ

2.2.1 Γενικά

Τα διάκριτα στοιχεία πληροφορίας παριστάνονται σε ένα ψηφιακό σύστημα, από φυσικές ποσότητες που λέγονται σήματα (signals). 'Οπως είναι γνωστό, τα

σήματα σε όλα τα σημερινά ψηφιακά ηλεκτρονικά συστήματα έχουν δύο μόνο διάκριτες τιμές και γι'αυτό τα λέμε δυαδικά (binary). Τα δυαδικά ψηφία (bits) χρησιμοποιούνται για την παράσταση δεδομένων υψηλότερου επιπέδου, όπως είναι οι χαρακτήρες, οι λέξεις, οι αριθμοί κλπ. 'Οσον αφορά τους αριθμούς, χρησιμοποιούνται κάποιοι ιδιαίτεροι τρόποι για την παράσταση τους, οι οποίοι θα αναφερθούν πιο κάτω. Κάθε σύγχρονο ψηφιακό σύστημα, είναι εφοδιασμένο με αριθμητικούς επεξεργαστές, οι οποίοι χειρίζονται δεδομένα διαφόρων τύπων και εκτελούν αριθμητικές λειτουργίες ή δίνουν αριθμητικές λύσεις σε υπολογιστικά προβλήματα. Η επιλογή ενός κατάλληλου συστήματος παράστασης αριθμών (number representation system), επιδρά τόσο στην αρχιτεκτονική ενός ψηφιακού συστήματος από την άποψη του σχεδιαστή, όσο και στις μεθόδους αριθμητικής ανάλυσης που εφαρμόζονται από τους χρήστες. Διαφορετική εσωτερική παράσταση αριθμών έχει σαν αποτέλεσμα διαφορετικό σχεδιασμό ενός αριθμητικού επεξεργαστή. Στη συνέχεια γίνεται μία παρουσίαση των βασικών συστημάτων παράστασης ακέραιων και μη ακέραιων αριθμών.

2.2.2 Σύστημα σταθερής υποδιαστολής (fixed-point representation)

Στο σύστημα αυτό η κάθε θέση ψηφίου αντιστοιχεί σε ένα αριθμό που λέγεται βάρος (weight). Για τον υπολογισμό της τιμής ενός αριθμού, πολλαπλασιάζεται η τιμή κάθε ψηφίου με το αντίστοιχο βάρος και τελικά αθροίζονται όλα τα μερικά γινόμενα. Εάν ο αριθμός D παριστάνεται από το διάνυσμα:
$D=D_{n-1}D_{n-2}...D_0.D_1D_2...D_{-m}$ τότε εάν R^i είναι το βάρος του ψηφίου της θέσης i όπου $(-m \le i \le n-1)$, η τιμή V(D) του αριθμού D είναι: $V(D) = \sum_{i=-m}^{n-1} D_i R^i$, ενώ για τα στοιχεία D_i ισχύει $0 \le D_i \le R$. Στη περίπτωση του δυαδικού συστήματος που μας ενδιαφέρει ισχύει R=2 (βάση). Για ένα αριθμητικό σύστημα με βάση R, στο οποίο διατίθενται k ψηφία, υπάρχει η δυνατότητα γιά την παράσταση R^k αριθμών. 'Οσον αφορά την παράσταση προσημασμένων αριθμών (signed numbers) σταθερής υποδιαστολής το πιο σημαντικό ψηφίο (MSB) του αριθμού χρησιμοποιείται για την παράσταση του προσήμου. Το "1" σημαίνει ότι ο αριθμός είναι αρνητικός, ενώ το "0" σημαίνει ότι ο αριθμός είναι θετικός. Υπάρχουν τρείς τρόποι για την παράσταση προσημασμένων αριθμών, σταθέρης υποδιαστολής, οι οποίοι παρουσιάζονται στις επόμενες τρείς ενότητες.

1. Παράσταση "πρόσημο και μέγεθος" (sign and magnitude representation)

Στην παράσταση αυτή το πιο σημαντικό bit του αριθμού είναι "0" ή "1", ανάλογα εάν ο αριθμός είναι θετικός ή αρνητικός. Τα υπόλοιπα (η-1) bits του αριθμού παριστάνουν το μέτρο του αριθμού σε δυαδική μορφή. Εάν π.χ η=6, ο αριθμός +12 παριστάνεται ως 001100, ενώ ο αριθμός -12 παριστάνεται ως 101100. Ο μέγιστος αριθμός που μπορεί να παρασταθεί με η bits είναι ο 2^{n-1}-1, ενώ ο ελάχιστος είναι ο -(2^{n-1}-1). Το μηδέν μπορεί να παρασταθεί με δύο τρόπους, είτε ως 000...0, είτε ως 100...0.

2. Παράσταση συμπληρώματος ως προς 1 (1's complement representation)

Στην παράσταση αυτή το MSB του αριθμού χρησιμοποιείται για την παράσταση του προσήμου. 'Οταν ο αριθμός είναι θετικός (MSB=0), τα υπόλοιπα (η-1) bits του αριθμού παριστάνουν το μέτρο του αριθμού σε δυαδική μορφή. 'Οταν όμως ο αριθμός είναι αρνητικός (MSB=1), το μέτρο του αριθμού δίδεται από το συμπλήρωμα ως προς 1 του συνόλου των ψηφίων του αριθμού. Το συμπλήρωμα ως προς 1 υπολογίζεται, εάν αντικατασταθούν όλα τα "1" του αριθμού με "0" και όλα τα "0" με "1". Για παράδειγμα το συμπλήρωμα ως προς 1 του αριθμού 101101 είναι ο αριθμός 010010. Το συμπλήρωμα ως προς 1 ενός αριθμού N, μπορεί να ορισθεί και ως 2^n-N-1. Με βάση τα παραπάνω, όταν η=6 η παράσταση συμπληρώματος ως προς 1 του αριθμού +12 είναι 001100, ενώ του αριθμού -12 είναι 110011. Ο μέγιστος αριθμός που μπορεί να παρασταθεί στο σύστημα αυτό (με λέξη μήκους η bits), είναι ο 2^n-1. Αντιστοίχως ο ελάχιστος αρνητικός αριθμός είναι ο -(2^{n-1}-1). Πρέπει επίσης να σημειωθεί ότι το μηδέν παριστάνεται με δύο τρόπους, είτε ως 000...0, είτε ως 111...1.

3. Παράσταση συμπληρώματος ως προς 2 (2's complement representation)

Στην παράσταση αυτή το MSB του αριθμού χρησιμοποιείται για την παράσταση του προσήμου. 'Οταν ο αριθμός είναι θετικός (MSB=0), τα υπόλοιπα bits του

αριθμού παριστάνουν το μέτρο του αριθμού σε δυαδική μορφή. 'Οταν όμως ο αριθμός είναι αρνητικός (MSB=1), το μέτρο του αριθμού δίδεται από το συμπλήρωμα ως προς 2 του συνόλου των ψηφίων του αριθμού. Το συμπλήρωμα ως προς 2, ορίζεται σαν το συμπλήρωμα ως προς 1 προσαυξημένο κατά μια μονάδα. 'Ετσι το συμπλήρωμα ως προς 2 του 10010 είναι το 01101+1=01110. Με βάση τα παραπάνω, όταν η=6, η παράσταση συμπληρώματος ως προς 2 του αριθμού +12 είναι 001100, ενώ η παράσταση του -12 είναι 110100. Ο μέγιστος θετικός αριθμός, που μπορεί να παρασταθεί στο σύστημα αυτό (με λέξη μήκους η bits) είναι ο 2^{n-1}-1, ενώ ο ελάχιστος αρνητικός είναι ο -2^{n-1}, ο οποίος είναι ο 100...0. Το μηδέν παριστάνεται μόνο ως 000...0.

2.2.3 Σύστημα κινητής υποδιαστολής (floating-point representation)

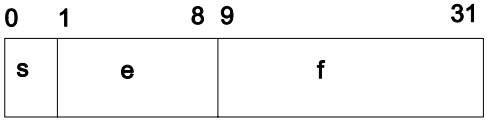
Η παράσταση αριθμών με τη μέθοδο της κινητής υποδιαστολής στηρίζεται στην εκθετική παράσταση των αριθμών και διευκολύνει την παράσταση πολύ μεγάλων ή πολύ μικρών αριθμών. Η μέθοδος αυτή χρησιμοποιείται κυρίως για την παράσταση πραγματικών μη ακεραίων αριθμών. Στην παράσταση αυτή, ο αριθμός αποτελείται από δύο τμήματα: το συντελεστή (mantissa), και τον εκθέτη (exponent). Εάν R είναι η βάση (στο δυαδικό σύστημα R=2), m ο συντελεστής και e ο εκθέτης, τότε ένας αριθμός κινητής υποδιαστολής εκλαμβάνεται ότι έχει την τιμή: $V=m*R^e$. Επειδή η βάση R είναι προκαθορισμένη, τα μόνα δεδομένα που απαιτούνται για την παράσταση ενός αριθμού κινητής υποδιαστολής είναι ο συντελεστής m και ο εκθέτης e.

Για την παράσταση του προσήμου του συντελεστή και του εκθέτη χρησιμοποιείται συχνά από ένα επιπλέον δυαδικό ψηφίο. Το μηδέν στη θέση του προσήμου (τόσο για το συντελεστή, όσο και για τον εκθέτη), υποδηλώνει θετικό αριθμό. Αντίστοιχα, το 1 υποδηλώνει αρνητικό. Ο εκθέτης ενός δυαδικού κινητής υποδιαστολής μπορεί να είναι προσημασμένος (signed) ή πολωμένος (biased). 'Οσον αφορά το πρόσημο του εκθέτη αναφερθήκαμε προηγουμένα. Στην περίπτωση του πολωμένου εκθέτη, ορίζεται μια σταθερά που ονομάζεται πόλωση (bias), η οποία προστίθεται στην πραγματική τιμή του εκθέτη. 'Ετσι ο

πολωμένος εκθέτης έχει πάντα μη αρνητική τιμή. Συνήθως η πόλωση έχει τιμή 2^{n-1}, όπου η ο συνολικός αριθμός των δυαδικών ψηφίων του εκθέτη. Προφανώς, όταν πρέπει να βρεθεί η πραγματική (μη πολωμένη) του εκθέτη, τότε πρέπει να αφαιρεθεί η πόλωση.

'Ενας αριθμός κινητής υποδιαστολής θεωρείται ότι είναι κανονικοποιημένος (normalized), όταν το πιό σημαντικό bit του συντελεστή δεν είναι μηδέν. Για παράδειγμα ο αριθμός $0,001011*2^{-5}$ δεν είναι κανονικοποιημένος. Στην κανονικοποιημένη του μορφή παριστάνεται ως $0,1011*2^{-7}$. Η κανονικοποίηση είναι επιθυμητή επειδή επιτρέπει την καταγραφή περισσότερων ψηφίων του συντελεστή, αυξάνοντας έτσι την ακρίβεια της παράστασης.

Σύμφωνα με τα παραπάνω για τον αριθμό $-0,5078125*2^{-5}$, έχουμε:
$m = (-0,5078125)_{10} = (11000001000000000000)_2$ (μορφή "πρόσημο και μέγεθος") $e=(-2+127)_{10}=(01111101)_2$ (πόλωση = 127). Το ινστιτούτο IEEE έχει συστήσει το πρότυπο για την παράσταση κανονικοποιημένων αριθμών κινητής υποδιαστολής που φαίνεται στο Σχήμα 2.2, όπου: s (1bit) το πρόσημο του συντελεστή, e (8 bits) ο εκθέτης πολωμένος κατά 127 με την προυπόθεση ότι 0< e<255, f (23bits) το κλασματικό μέρος του συντελεστή μετά την κανονικοποιησή του.



Σχήμα 2.2

Συνολικά χρησιμοποιούνται 32 δυαδικά ψηφία. Το διάστημα των αριθμών, που μπορούν να παρασταθούν με το πρότυπο του IEEE, είναι $2^{-126} \le D \le 2^{128}$.

2.2.4 Σύγκριση των συστημάτων των αριθμών

Η επιλογή ενός συστήματος παράστασης αριθμών βασίζεται στην αποδοτικότητα του, στην ευκολία της αριθμητικής σχεδίασης και στην αξιοπιστία της λειτουργίας. Αντιπαραθέτοντας το σύστημα σταθερής υποδιαστολής και το

σύστημα κινητής υποδιαστολής μπορούμε να δούμε τα εξής: Το μεγάλο πλεονέκτημα της παράστασης κινητής υποδιαστολής είναι ότι επιτρέπει την παράσταση πολύ μεγάλων και πολύ μικρών ποσοτήτων με την χρήση περιορισμένου αριθμού δυαδικών ψηφίων. Βέβαια οι αριθμοί σταθερής υποδιαστολής έχουν απλότερη μορφή από αυτούς της κινητής υποδιαστολής και η χρήση τους σε μαθηματικούς επεξεργαστές είναι ευκολότερη. 'Ετσι οι μονάδες που χρησιμοποιούν αριθμητική σταθερής υποδιαστολής είναι απλότερες και μικρότερες από τις αντίστοιχες κινητής υποδιαστολής.

Λόγω της πολυπλοκότητας ενός συστήματος κινητής υποδιαστολής τα σύγχρονα συστήματα έχουν ενσωματωμένο ειδικό υλικό για τη διευκόλυνση του χειρισμού και της αποθήκευσης αριθμών κινητής υποδιαστολής. Λαμβάνοντας υπόψιν τα παραπάνω και αφού στην εφαρμογή μας το πεδίο τιμών των αριθμών είναι γνωστό ή προβλέπεται κατά την διάρκεια του σχεδιασμού θα χρησιμοποιήσουμε το σύστημα της σταθερής υποδιαστολής.

Στη συνέχεια πρέπει να επιλεγεί ένας τρόπος παράστασης προσημασμένων αριθμών (signed numbers) σταθερής υποδιαστολής. Οι αριθμητικές πράξεις με αριθμούς που χρησιμοποιούν την παράσταση συμπληρώματος ως προς 2 ή ως προς 1 μπορούν να υλοποιηθούν πολύ πιο εύκολα από τις πράξεις με αριθμούς που χρησιμοποιούν την παράσταση "πρόσημο και μέγεθος". Αυτό έχει οδηγήσει στην ευρεία χρήση της παράστασης του συμπληρώματος ως προς 1 και της παράστασης του συμπληρώματος ως προς 2. Μία σύγκριση μεταξύ των συμπληρωμάτων ως προς 1 και 2 θα μας δείξει τα πλεονεκτήματα και τα μειονεκτήματα του καθενός. Το συμπλήρωμα ως προς 1 ενός αριθμού έχει το πλεονέκτημα ότι υπολογίζεται ευκολότερα με ψηφιακά κυκλώματα αφού το μόνο που έχουμε να κάνουμε είναι να αλλάξουμε τα "0" σε "1" και τα "1" σε "0". Επίσης το συμπλήρωμα του 2 έχει μόνο ένα αριθμητικό μηδέν ενώ το μηδέν του συμπληρώματος ως προς 1 μπορεί να είναι "θετικό" ή "αρνητικό" μπερδεύοντας έτσι τα πράγματα.

Το κυριότερο όμως πλεονέκτημα του συστήματος του συμπληρώματος ως προς 2 το οποίο μας οδήγησε στην επιλογή του είναι ότι κατά την αφαίρεση δύο δυαδικών (binary) αριθμών χρειάζεται μία μόνο πράξη αριθμητικής πρόσθεσης

μεταξύ του μειωτέου και του συμπληρώματος ως προς 2 του αφαιρετέου. Παρατηρούμε λοιπόν ότι το σύστημα συμπληρώματος του 2 είναι γενικά ευκολόχρηστο και αποδοτικό γι'αυτό και χρησιμοποιήθηκε στην εφαρμογή μας.

2.3 ΕΠΕΞΕΡΓΑΣΙΑ ΤΥΠΟΥ ΑΓΩΓΟΥ (PIPELINED PROCESSING) ΣΤΟΥΣ ΑΡΙΘΜΗΤΙΚΟΥΣ ΕΠΕΞΕΡΓΑΣΤΕΣ

Δύο όροι χρησιμοποιούνται συχνά για τον καθορισμό της υπολογιστικής ισχύος ενός συστήματος. Ο πρώτος είναι ο αριθμός των λειτουργιών που μπορούν να εκτελεστούν στη μονάδα του χρόνου (bandwidth) και ο δεύτερος είναι το διάστημα του χρόνου που χρειάζεται για να εκτελεστεί μια απλή λειτουργία (latency). Για την μηχανή που εκτελεί μία λειτουργία στη μονάδα του χρόνου ο δεύτερος όρος είναι το αντίστροφο του πρώτου. Στη συμβατική σχεδίαση η αύξηση του πρώτου όρου (bandwidth) των αριθμητικών επεξεργαστών επιτυγχανόταν με μείωση του δεύτερου όρου (latency) μέσω ταχύτερων λογικών κυκλωμάτων. Για παράδειγμα ταχύτεροι αθροιστές σχεδιάστηκαν με την ελαχιστοποίηση του χρόνου διάδοσης του κρατουμένου (carry propagation). Ταχύτεροι πολλαπλασιαστές σχεδιάστηκαν να επιτρέπουν ταυτόχρονη πρόσθεση πολλών αριθμών.

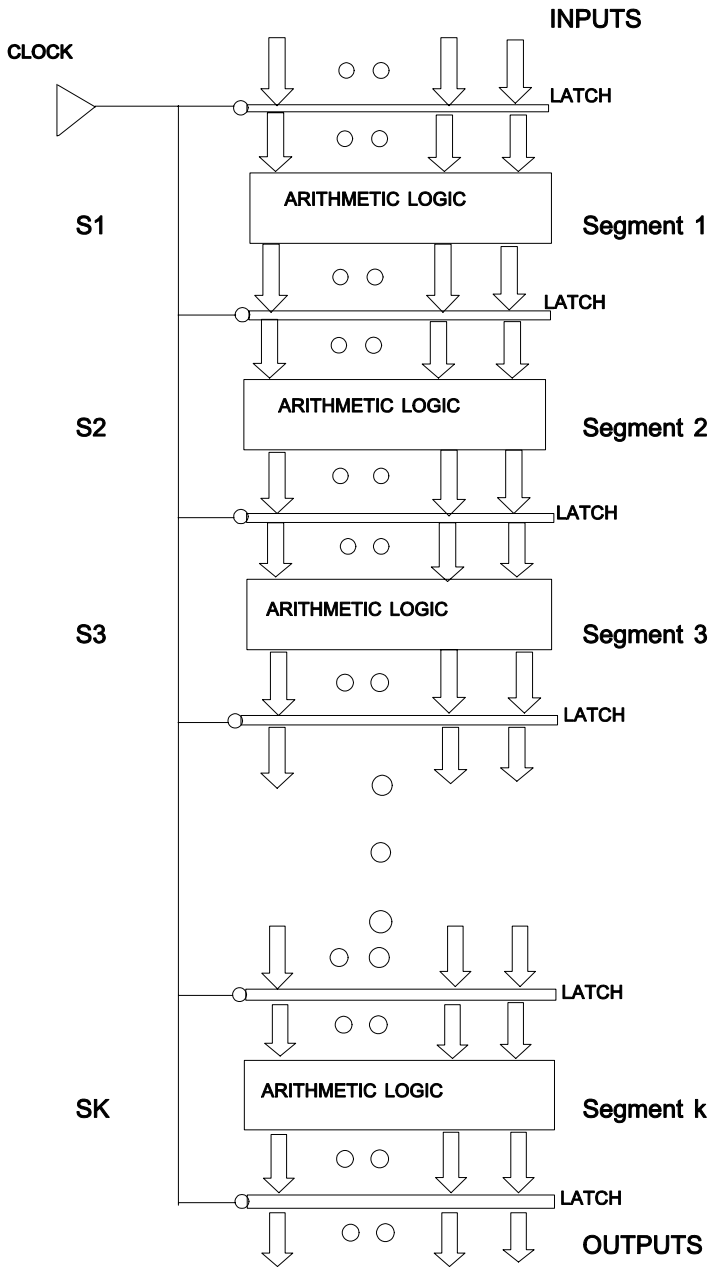
Ωστόσο η τεχνολογία κυκλωμάτων έχει σχεδόν φτάσει στο ύστατο όριο της ταχύτητας. 'Ετσι γρήγορα κυκλώματα πλέον δεν είναι αρκετά για την σημαντική αύξηση της υπολογιστικής ισχύος. Η εισαγωγή των μικροεπεξεργαστών και της τεχνολογίας VLSI επέτρεψε τόσο την ραγδαία πτώση του κόστους του υλικού, όσο και την σημαντική αύξηση της ταχύτητας των υπολογιστών. Επειδή όμως η πίεση για ακόμα ισχυρότερους υπολογιστές εξακολουθούσε να αυξάνει οι σχεδιαστές έστρεψαν την προσοχή τους σε νέα πεδία. Μια εύλογη λύση σε αυτό το πρόβλημα είναι να επιτραπεί ταυτόχρονη εκτέλεση πολλών λειτουργιών από πολλές αριθμητικές μονάδες (παράλληλη επεξεργασία). Βέβαια, η παράλληλη επεξεργασία (parallel processing) δεν είναι μια οικονομική λύση λόγω μεγάλης αύξησης του υλικού (hardware). Η περίπτωση όμως της επεξεργασίας τύπου αγωγού (pipelined processing) εξασφαλίζει σημαντική αύξηση στον αριθμό των λειτουργιών στη μονάδα του χρόνου (bandwidth) με μια μόνο λογική αύξηση του

υλικού.

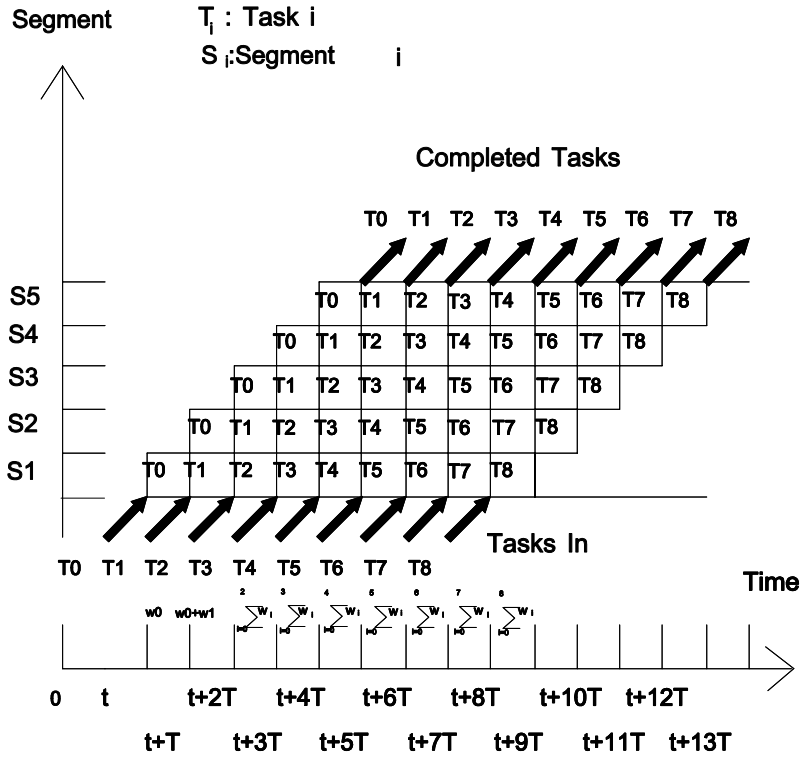
Ο υπολογισμός τύπου αγωγού (pipelined computing) στηρίζεται στην υποδιαίρεση του συνολικού υπολογισμού σε ξεχωριστές λειτουργίες, οι οποίες μπορούν να εκτελεστούν χρονικά επικαλυπτόμενες μέσω μιας αριθμητικής τύπου αγωγού (arithmetic pipeline) υψηλής ταχύτητας υπό τον περιορισμό όμως ορισμένης προτεραιότητας. Αυτή η χρονικά επικαλυπτόμενη εκτέλεση έχει χρησιμοποιηθεί στη σχεδίαση κεντρικού επεξεργαστή στον οποίο οι τέσσερις φάσεις του τυπικού κύκλου εκτέλεσης (instruction fetch, instruction decode, operand fetch, execution) της επόμενης εντολής μπορούν να είναι χρονικά επικαλυπτόμενες με τις αντίστοιχες φάσεις της τρέχουσας εντολής. Αν οι χρόνοι εκτέλεσης είναι ίσοι τότε ο υπολογισμός τύπου αγωγού (pipelined computing) μπορεί να είναι δύο φορές γρηγορότερος από τον παραδοσιακό υπολογισμό. Μία αριθμητική μονάδα τύπου αγωγού (pipelined arithmetic unit) ορίζεται γενικά σαν μια συλλογή από τμήματα υλικού (segments) τα οποία είναι οργανωμένα σε μια μορφή αγωγού με ένα σήμα ελέγχου συγχρονισμού, τέτοιο ώστε οι υπολειτουργίες να μπορούν να εκτελούνται ταυτόχρονα από τα διαδοχικά τμήματα του αγωγού (pipeline).

Στο Σχήμα 2.3 κάθε τμήμα S_J είναι βασικά ένα ειδικού σκοπού αριθμητικής λογικής κύκλωμα με μια καθυστέρηση τ_J, όπως ένας αθροιστής, ένας πολλαπλασιαστής κλπ. Τα διαδοχικά τμήματα (segments) διασυνδέονται με μανταλωτές δεδομένων (data latches), που είναι συγχρονισμένοι καταχωρητές και κρατούν τα σήματα εισόδου και εξόδου των διαδοχικών τμημάτων (σχήμα 2.3). Τα b_J bits εξόδου του τμήματος S_J φέρονται σαν τα bits εισόδου a_{J+1} του τμήματος S_{J+1}. Γενικά λοιπόν ισχύει $b_J=a_J+1$. 'Εστω τώρα W_J η ολική λέξη που χειρίζεται το τμήμα S_J μέσα σε ένα χρονικό διάστημα τ_J. Διαφορετικά τμήματα τελειώνουν τις λειτουργίες τους σε διαφορετικά χρονικά διαστήματα. Για να ρυθμίσουμε την λειτουργία του αγωγού (pipeline) το ρολόι συγχρονισμού θα πρέπει να έχει περίοδο $\tau = \mathrm{Max}\{T_i \}_{0}^{k-1} + T_l$ όπου T_l είναι η καθυστέρηση διάδοσης μέσω ενός απλού μανταλωτή (latch). Επομένως ο ρυθμός ενός συστήματος τύπου αγωγού (pipeline system) καθορίζεται από την μέγιστη καθυστέρηση ενός τμήματος. Κάθε μανταλωτής ελευθερώνει την πληροφορία στο τμήμα που ακολουθεί μόνο όταν πυροδοτηθεί από το σήμα του εξωτερικού ρολογιού. Τα

δύο ακρινά τμήματα έχουν επιπρόσθετους μανταλωτες οι οποίοι κρατούν τις εισόδους/εξόδους ολόκληρου του συστήματος των k τμημάτων. Χρειάζονται k κύκλοι ρολογιού για να "γεμίσει" ή να "αδειάσει" ο αγωγός (pipeline). Όταν ο αγωγός "γεμίσει" εντελώς, κάθε τμήμα είναι απασχολημένο αφού εκτελεί την J λειτουργία T_J η οποία απαιτεί ποσό εργασίας W_J.



Σχήμα 2.3: Αριθμητικός αγωγός με k τμήματα



Σχήμα 2.4: Διαδοχικές λειτουργίες σε αγωγό 5 τμημάτων

Ο ρυθμός εργασίας (work rate) όταν ο αγωγός βρίσκεται στην σταθερή κατάσταση (έχει "γεμίσει" εντελώς), είναι $\sum_{J=0}^{k-1} W_i$ ανά κύκλο ρολογιού. Ο αριθμός λειτουργιών στη μονάδα του χρόνου (bandwith) είναι το αντίστροφο της μέγιστης χρονικής διάρκειας τ (latency per segment), και όχι της χρονικής διάρκειας k*τ όλου του αγωγού. Επομένως η αύξηση του αριθμού των τμημάτων δεν έχει επίδραση στον αριθμό των λειτουργιών που εκτελούνται σε ένα κύκλο ρολογιού (bandwith). Ο αγωγός στη σταθερή του κατάσταση ολοκληρώνει μία λειτουργία ανά κύκλο ρολογιού. Οπότε k λειτουργίες μπορούν ταυτόχρονα να εκτελεστούν στον αγωγό σε k διαδοχικούς κύκλους ρολογιού. Το σχήμα 2.4 δείχνει τη ροή διαδοχικών λειτουργιών διαμέσου ενός αριθμητικού αγωγού (arithmetic pipeline) πέντε τμημάτων. Για την μεγαλύτερη αύξηση της υπολογιστικής ισχύος μπορούν σε ένα σύστημα να λειτουργούν παράλληλα διάφοροι αγωγοί, οι οποίοι θα χειρίζονται πολλαπλά ρεύματα δεδομένων. Στην εφαρμογή μας προσπαθήσαμε ο υπολογισμός να είναι τύπου αγωγού (pipelined computing) έτσι ώστε να αποκομίσουμε όλα τα οφέλη αυτής της αρχιτεκτονικής που αναφέρθηκαν.

Κεφάλαιο 3

Τεχνικές ψηφιακής επεξεργασίας σημάτων (Digital signal processsing techniques)

3.1 ΓΕΝΙΚΑ

Εφαρμογές επεξεργασίας σημάτων συναντώνται συχνά στο σύγχρονο κόσμο των ηλεκτρονικών. Οι τεχνικές ψηφιακής επεξεργασίας σημάτων επιχειρούν μία επιλογή μεταξύ της ποιότητας και του κόστους γεγονός του οποίου η επίδραση μειώνεται συνεχώς με την πρόοδο της τεχνολογίας κατασκευής. Η αυξανόμενη σημασία των εφαρμογών ψηφιακής επεξεργασίας σημάτων (D.S.P applications), και η αυξανόμενη έλξη προς την VLSI υλοποίηση συστημάτων ήταν το κίνητρο για την ενασχολησή μας με αυτό το πεδίο.

3.2 ΣΥΓΧΡΟΝΑ ΣΥΣΤΗΜΑΤΑ (SYNCHRONOUS SYSTEMS)

Στη σχεδίαση των ψηφιακών επεξεργαστών σε επίπεδο VLSI μία δημοφιλής προσέγγιση είναι η αρχιτεκτονική σύγχρονων συστημάτων (synchronous system architecture), στην οποία οι υπολογισμοί εκτελούνται σε δεδομένα που ρέουν σε

σταθερό γράφο ροής δεδομένων (data flow graph) μεταξύ αριθμητικών και λογικών επεξεργαστών. Κάθε επεξεργαστής στο γράφο ροής ειδικεύεται στη λειτουργία που είναι φτιαγμένος να λειτουργεί η οποία μπορεί να είναι πρόσθεση, πολλαπλασιασμός, ή κάποια λογική λειτουργία. 'Ενα σύγχρονο κύκλωμα είναι ένα σύστημα του οποίου η συμπεριφορά μπορεί να περιγραφεί πλήρως από τις τιμές των σημάτων του σε διάκριτες τιμές του χρόνου.

'Ενα σύγχρονο σύστημα πρέπει εξ'ορισμού να χρησιμοποιεί σήματα τα οποία επηρεάζουν τα στοιχεία του σε διακριτές μόνο στιγμές του χρόνου. 'Ενας τρόπος να πετύχουμε αυτό το σκοπό είναι να χρησιμοποιήσουμε παντού στο σύστημα παλμούς μιας ορισμένης διάρκειας και να συμφωνήσουμε ότι οι παλμοί μιας ορισμένης πολικότητας θα παριστάνουν το λογικό "1" και της άλλης πολικότητας (ή η έλλειψη παλμού) το λογικό "0". Ο συγχρονισμός επιτυγχάνεται μέσω μιάς "γεννήτριας κύριου ρολογιού" (master-clock generation) η οποία τροφοδοτεί το σύστημα με μία περιοδική σειρά παλμών ή κύκλων ρολογιού (clock pulses or cycles). Αυτοί οι παλμοί διανέμονται παντού στο σύστημα και χρησιμοποιούνται με τέτοιο τρόπο ώστε τα διάφορα στοιχεία να επηρεάζονται από τις εισόδους τους μόνο τις στιγμές που φθάνουν αυτοί οι παλμοί συγχρονισμού. Γενικά λοιπόν σε ένα σύγχρονο σύστημα η κάθε επικοινωνία και ο κάθε υπολογισμός γίνονται με βάση ένα κύριο ρολόι. Το ρολόι αυτό είναι σχετικά γρήγορο για κάθε δεδομένη τεχνολογία. Τυπικά η ταχύτητα του κυμαίνεται σε μερικές δεκάδες μεγακύκλους (MHz) για τεχνολογίες τριών μικρών (3-micron technologies). Βέβαια και οι ασύγχρονες αρχιτεκτονικές είναι εφικτές και θα μπορούσαν να δώσουν μια καλή λύση στα προβλήματα που δημιουργούνται με τα γρήγορα σύγχρονα συστήματα. Ανάμεσα όμως σε πολλές στρατηγικές εμείς θα ασχοληθούμε αποκλειστικά με σύγχρονα και μάλιστα σειριακά συστήματα.

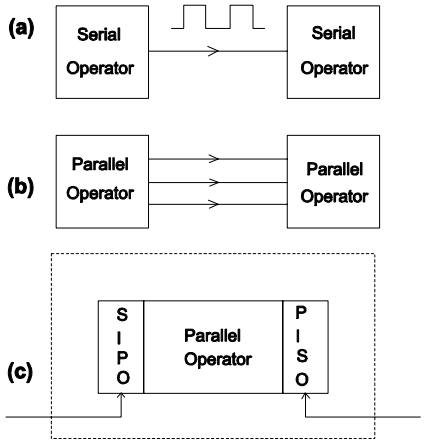
3.3 ΣΕΙΡΙΑΚΕΣ ΣΕ ΕΠΙΠΕΔΟ BIT ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ (BIT-SERIAL ARCHITECTURES)

Οι σειριακές σε επίπεδο-bit αρχιτεκτονικές διακρίνονται για την στρατηγική επικοινωνίας που ακολουθούν. Τα ψηφιακά σήματα μεταδίδονται με ακολουθιακό τρόπο bit προς bit σε απλές γραμμές, εν αντιθέση προς την παράλληλη

μετάδοση των λέξεων σε παράλληλους διαύλους (buses) (σχήμα 3.1). Πιο συγκεκριμένα στις αρχιτεκτονικές αυτές τα δεδομένα μεταδίδονται ένα bit στη μονάδα του χρόνου πάνω σε απλές γραμμές και όλες οι λειτουργικές μονάδες λειτουργούν πάνω σε απλά bits δεδομένων στη μονάδα του χρόνου. Το διακριτικό αυτό στοιχείο των σειριακών σε επίπεδο-bit αρχιτεκτονικών είναι το κλειδί για τα πολλά πλεονεκτήματα τους ως VLSI στρατηγικές.

Το πιο σημαντικό πλεονέκτημα είναι ότι οδηγούν σε αποδοτική επικοινωνία μεταξύ VLSI κυκλωμάτων (chips) και μέσα σε αυτά. Για το λόγο αυτό οι αρχιτεκτονικές αυτές επικρατούν στις εφαρμογές επεξεργασίας ψηφιακών σημάτων. 'Ενα άλλο σημαντικό πλεονέκτημα της σειριακής σε επίπεδο-bit αρχιτεκτονικής είναι το σχετικά μικρό μέγεθος των λειτουργικών μονάδων που χρησιμοποιούνται και οι ελάχιστες απαιτήσεις ενδοσυνδέσεων. 'Ενα μειονέκτημα αυτής της αρχιτεκτονικής είναι η σχετικά χαμηλή ρυθμοαπόδοση (throughput) που προκύπτει από την σειριακής φύσης μετάδοση δεδομένων.

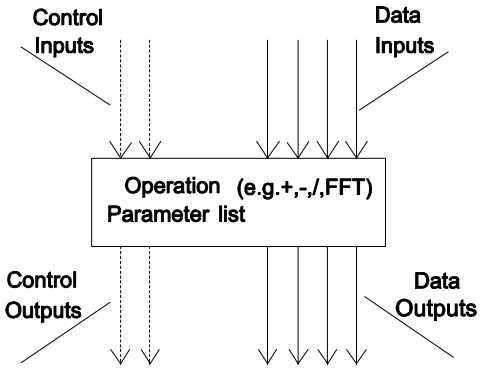
Στο σχήμα 3.1 αντιπαραθέτουμε την σειριακή σε επίπεδο-bit αρχιτεκτονική (3.1.a) με την παράλληλη σε επίπεδο-bit αρχιτεκτονική (bit-parallel architecture) (3.1.b), στην οποία όλα τα bits μιας λέξης εισόδου ή ενός δείγματος εισόδου μεταδίδονται σε μία μονάδα του χρόνου. Βέβαια αυτή η αρχιτεκτονική (bit-parallel) απαιτεί μεγαλύτερη περιοχή (area) από την σειριακή σε επίπεδο-bit αρχιτεκτονική.



Σχήμα 3.1: VLSI αρχιτεκτονικές

Επίσης για πολλές εφαρμογές είναι κατάλληλες διατάξεις παράλληλου υπολογισμού (parallel computation schemes). Σε αυτές τις περιπτώσεις τα παράλληλα στοιχεία πρέπει να χρησιμοποιούνται με σειριακές διασυνδέσεις επικοινωνίας (serialised communications interfaces), όπως φαίνεται στο σχήμα 3.1 (3.1.c).

Στο σχήμα 3.2 βλέπουμε την γενική μορφή ενός στοιχείου σειριακής σε επίπεδο-bit επεξεργασίας (bit-serial operation). Το στοιχείο αυτό μπορεί να εκτελέσει κάθε λειτουργία που αφορά επεξεργασία σήματος πάνω σε μία ή περισσότερες λέξεις εξόδου μετά από την αναγκαία καθυστέρηση για τη διαδικασία υπολογισμού. Αυτή η καθυστέρηση καλείται latency του στοιχείου και μετριέται σαν ένας ακέραιος αριθμός από bits ή λέξεις και bits.

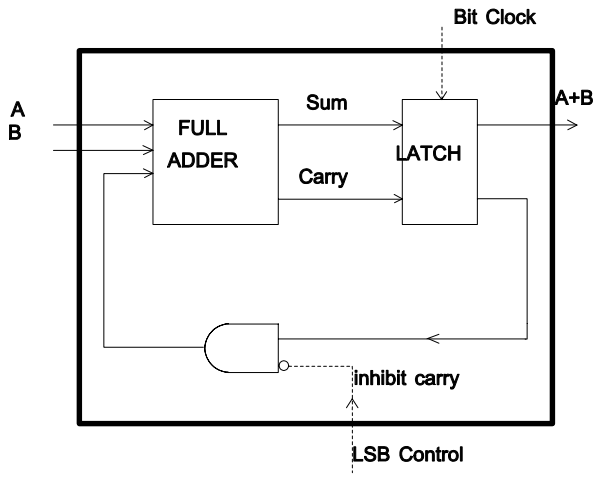


Σχήμα 3.2: Στοιχείο σειριακής-bit επεξεργασίας

Μία σημαντική αρχή είναι όλα τα λειτουργικά στοιχεία (operators) να έχουν σταθερή και γνωστό latency. Κάθε λειτουργικό στοιχείο έχει ένα προεραιτικό σύνολο από εισόδους ελέγχου (control inputs) συνήθως σε απλές γραμμές. Αυτές οι είσοδοι υποδεικνύουν την άφιξη μιας νέας λέξης ή ομάδας λέξεων και γενικά χρησιμοποιούνται για να αρχίσουν (ή να τερματίσουν) την λειτουργία για κάθε σύνολο δεδομένων. Μερικά λειτουργικά στοιχεία έχουν και εξόδους ελέγχου (control outputs). Οι είσοδοι ελέγχου καθυστερούνται κατάλληλα για να εφαρμοστούν σαν σύγχρονες είσοδοι ελέγχου στο επόμενο λειτουργικό στοιχείο. Στην ενότητα αυτή θα ασχοληθούμε μόνο με τις βασικές λειτουργίες ενός στοιχείου. Τις στρατηγικές ελέγχου θα τις δούμε σε επόμενη ενότητα. Γενικά ένα

τρίτο είδος σημάτων εισόδου είναι πιθανό: οι παράμετροι. Για παράδειγμα το κέρδος σε μια διάταξη μπορεί να θεωρηθεί ως μία παράμετρος. Στα συστήματα όμως με τα οποία θα ασχοληθούμε δεν συναντάμε συνήθως παραμέτρους παρά μόνο δύο τύπους δεδομένων: σήματα και έλεγχο.

Σαν ένα παράδειγμα σειριακού λειτουργικού στοιχείου (serial operator) δίνουμε στο σχήμα 3.3 έναν αθροιστή. Θεωρούμε ότι πρώτα εισάγεται στη διάταξη το LSB (least significant bit) των δεδομένων. 'Οσον αφορά την λειτουργία του ο αθροιστής αρχικά συνδιάζει διαδοχικά ζεύγη των bits εισόδου διαμέσου ενός πλήρους αθροιστή (full adder), κρατά το άθροισμα σαν έξοδο και ανατροφοδοτεί το κρατούμενο (carry) για τον συνδιασμό με το επόμενο (περισσότερο σημαντικό) ζεύγος των bits εισόδου. Οι απαιτήσεις ελέγχου στην περίπτωση αυτή είναι ένας παλμός συγχρόνως με την άφιξη των λιγότερο σημαντικών bits των δύο λέξεων εισόδου. Αυτό το σήμα ελέγχου χρησιμοποιείται για να εμποδίσει το κρατούμενο της προηγούμενης πρόσθεσης να έχει λανθασμένη επίδραση στη πρόσθεση δύο επόμενων λέξεων.



Σχήμα 3.3: Σειριακός-bit αθροιστής

Τελικά λοιπόν το λειτουργικό αυτό στοιχείο (αθροιστής) έχει δύο ρεύματα δεδομένων εισόδου, ένα ρεύμα δεδομένων εξόδου, μία είσοδο ελέγχου και latency ένα bit. Μπορούμε επίσης να κατασκευάσουμε συστήματα ως δίκτυα τέτοιων στοιχείων που θα εκτελούν εργασίες επεξεργασίας σημάτων με σειριακή

σε επίπεδο-bit μέθοδο όπως είναι ο πολλαπλασιαστής, το σύστημα υπολογισμού του FFT, διάφορα φίλτρα κλπ.

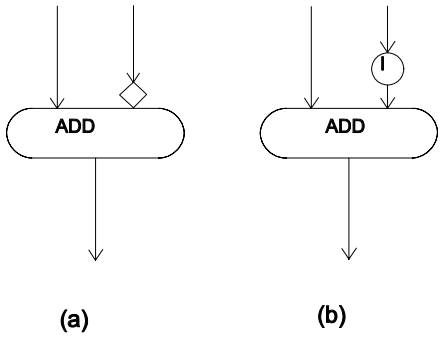
'Οσον αφορά τώρα για το ποιό bit της λέξης εισόδου (MSB ή LSB) θα εισέλθει πρώτο για επεξεργασία σε σύστημα σειριακής σε επίπεδο-bit αρχιτεκτονικής μπορούμε να αναφέρουμε τα εξής : ορισμένες λειτουργίες όπως η διαίρεση για παράδειγμα, εκτελούνται πιό εύκολα όταν εισέλθει πρώτο για σειριακή σε επίπεδο-bit επεξεργασία το MSB. Ωστόσο η πλειονότητα των εφαρμογών που ενδιαφέρουν κυριαρχείται από λειτουργίες πολλαπλασιασμού και πρόσθεσης, οι οποίες από την φύση τους εκτελούνται ευκολότερα όταν αρχίσουμε από το LSB. Γι'αυτό το λόγο υιοθετούμε αυτόν τον τύπο επεξεργασίας. Στην επιλογή ενός από αυτούς τους δύο τρόπους επεξεργασίας θα αναφερθούμε και αργότερα όταν θα αναλύουμε τις σειριακές σε επίπεδο-ψηφίου αρχιτεκτονικές (digit-serial architecture).

'Ενας άλλος σημαντικός παράγοντας είναι το μήκος της λέξης εισόδου το οποίο επηρεάζει την δυναμική περιοχή κάθε συστήματος. Το μήκος αυτό μπορεί να είναι διαφορετικό σε κάθε εφαρμογή. Ωστόσο είναι πιο βολικό το μήκος της λέξης να είναι σταθερό σε ένα ειδικό σύστημα. Τυπικές τιμές για το μήκος λέξης ενός συστήματος είναι μεταξύ 12 και 32 bits. Βέβαια και μεγαλύτερες τιμές είναι εφικτές αλλά όχι πρακτικές για συστήματα σειριακής σε επίπεδο-bit αρχιτεκτονικής.

Η σχεδίαση συστημάτων απλοποιείται αν οι είσοδοι όλων των λειτουργικών στοιχείων σε ένα σύστημα είναι σύγχρονες, δηλαδή τα LSBς όλων των εισόδων φθάνουν ταυτόχρονα. Αυτό όμως είναι κάπως επίμαχο γιατί πολλές φορές είναι πιο αποτελεσματικό σε ένα λειτουργικο στοιχείο να θεωρήσουμε ότι οι είσοδοι είναι διαθέσιμες σε διαφορετικές χρονικές στιγμές. Αυτό είναι ένα καλό παράδειγμα της ισορροπίας (tradeoff) μεταξύ της απλότητας σχεδίασης (design simplicity) και της αποδοτικότητας του υλικού (hardware) η οποία διέπεται από διάφορες συμβάσεις.

'Εστω λοιπόν ότι τα δύο σήματα εισόδου σε ένα λειτουργικό στοιχείο έχουν χρονική διαφορά 1 bit. Αυτή η κατάσταση μπορεί να βελτιωθεί αν το λειτουργικό

στοιχείο έχει μια προαιρετική προκαθυστέρηση (predelay) εισόδου ενός bit η οποία μπορεί να καλείται σαν μέρος της βασικής δομής για να αποφεχθούν επιπρόσθετες δρομολογήσεις του ενδιάμεσου σήματος. Η χρήση αυτής της ευκολίας μπορεί να είναι εσωτερική. Στο σχήμα 3.4 βλέπουμε την εφαρμογή της προκαθυστέρησης σε έναν βασικό αθροιστή. 'Οταν η επιλογή της προκαθυστέρησης χρησιμοποιείται δείχνουμε την παρουσία της με ένα σύμβολο ρόμβου στο τέλος της σχετικής εισόδου.



Σχήμα 3.4: (α) Προκαθυστέρηση (β) Ισοδύναμο κύκλωμα

Δύο βασικά πλεονεκτήματα των σειριακών σε επίπεδο-bit αρχιτεκτονικών είναι ότι δίκτυα τέτοιων αρχιτεκτονικών δρομολογούνται εύκολα πάνω στο chip χωρίς τα προβλήματα που δημιουργούνται με τους διαύλους (buses) των δικτύων που είναι κατασκευασμένα σε παράλληλες σε επίπεδο-bit (bit-parallel) αρχιτεκτονικές και ότι όλα τα σήματα εισάγονται στο chip και εξάγονται από αυτό μέσω απλών ακροδεκτών (pins). Επίσης οι δίαυλοι των παράλληλων αρχιτεκτονικών δεν εκμεταλεύονται το διαθέσιμο εύρος ζώνης της πληροφορίας (information bandwidth) τόσο αποδοτικά όσο οι απλές γραμμές των σειριακών αρχιτεκτονικών.

'Ενα μειονέκτημα των σειριακών συστημάτων όμως είναι το σταθερό μήκος των λέξεων που χρησιμοποιούν. Συγκρίνοντας τα λειτουργικά στοιχεία (operators) των σειριακών και παράλληλων σε επίπεδο-bit αρχιτεκτονικών θεωρώντας ένα κοινό ρυθμό ρολογιού (clock rate), παρατηρούμε ότι στο σειριακό τύπο έχουμε επεξεργασία μιας λέξης κάθε N παλμούς ρολογιού ενώ στον παράλληλο τύπο έχουμε επεξεργασία μιας λέξης σε ένα παλμό ρολογιού. 'Ομως στη σειριακή

αρχιτεκτονική απαιτείται Ν φορές λιγότερο υλικό (hardware). Με δεδομένα ένα κοινό ρυθμό ρολογιού και τη βασική τεχνολογία για τις δύο υλοποιήσεις μπορούμε να δημιουργήσουμε ένα μέτρο σύγκρισης περιοχής (area) και χρόνου (time) "A*T". Βέβαια η συνθήκη του κοινού ρολογιού που θεωρήσαμε προηγουμένως δεν συμβαίνει συνήθως αφού στα σειριακά σε επίπεδο-bit συστήματα επιτρέπεται η χρήση μικρότερης περιοδού ρολογιού από αυτή των παράλληλων σε επίπεδο-bit συστημάτων, στα οποία για παράδειγμα απαιτείται σχετικά μεγάλη περίοδος ρολογιού για τη διάδοση του κρατουμένου (carry propagation) στα στοιχεία πολλαπλασιασμού και πρόσθεσης.

Μετά από όλη αυτή την αναφορά στις σειριακές σε επίπεδο-bit αρχιτεκτονικές είναι χρήσιμο να δώσουμε κάποιες συμβάσεις για τις αρχιτεκτονικές αυτές που έχουν επικρατήσει μέχρι σήμερα :

α. Επικοινωνία σειριακή σε επίπεδο ενός bit.
β. Εισαγωγή στα στοιχεία, πρώτα του LSB της λέξης εισόδου.
γ. Υψηλή στάθμη σημάτων το λογικό "1".
δ. Χαμηλή στάθμη σημάτων το λογικό "0".
ε. Το σύστημα παράστασης αριθμών είναι σταθερής υποδιαστόλης συμπληρώματος του δύο (fixed-point two's complement).
στ.Το μήκος λέξης των δεδομένων είναι σταθερό για κάθε σύστημα.
ζ. Τα λειτουργικά στοιχεία έχουν σταθερό latency.
η. Οι είσοδοι στα λειτουργικά στοιχεία είναι χρονικά ευθυγραμισμένες (time aligned).

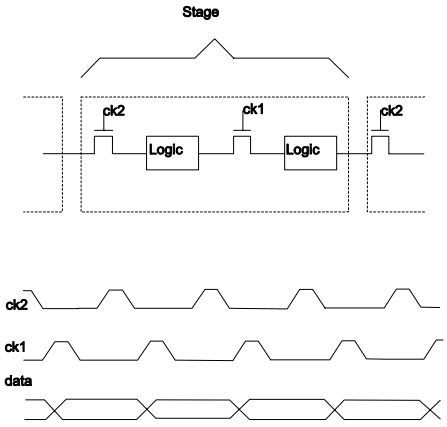
Μερικές επιμέρους συμβάσεις που αφορούν την στρατηγική ελέγχου σειριακών σε επίπεδο-bit συστημάτων θα αναφερθούν στην επόμενη ενότητα.

Τέλος κάνοντας μια αναφορά στην μεθοδολογία σχεδίασης αναφέρουμε ότι μία αποδοτική ενέργεια για το σχεδιαστή είναι η οργάνωση ενός συνόλου βασικών λειτουργικών στοιχείων (primitive operators) από το οποίο μία μεγάλη ποικιλία συστημάτων μπορεί να κατασκευαστεί. Το μεγάλο πλεονέκτημα της σχεδίασης με αυτό το τρόπο είναι ότι τα βασικά στοιχεία κατασκευάζονται μόνο μια φορά και οι χρήστες γλιτώνουν την εργασία της επαναυλοποίησης μη αναγκαίων

λεπτομερειών. Αν όλα τα λειτουργικά στοιχεία είναι του ίδιου γενικού τύπου (βλεπε σχήμα 3.2), τότε τα διάφορα συστήματα μπορούν να σχεδιαστούν ιεραρχικά. Αυτό σημαίνει ότι λειτουργικά στοιχεία υψηλότερου επιπέδου μπορούν να ορισθούν ως δίκτυα από υπάρχοντα λειτουργικά στοιχεία και μετά να χρησιμοποιηθούν στην ίδια συνοψισμένη λογική. Βέβαια τα επίπεδα ιεραρχίας μπορούν να αλλάξουν σειρά. Οπότε είναι απόλυτα σωστή η κατασκευή ενός συστήματος από λειτουργικά στοιχεία που έχουν ορισθεί προηγουμένως, αδιάφορα από το επίπεδό τους.

3.4 ΧΡΟΝΙΣΜΟΣ ΚΑΙ ΣΤΡΑΤΗΓΙΚΕΣ ΕΛΕΓΧΟΥ ΣΤΗ ΣΕΙΡΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ

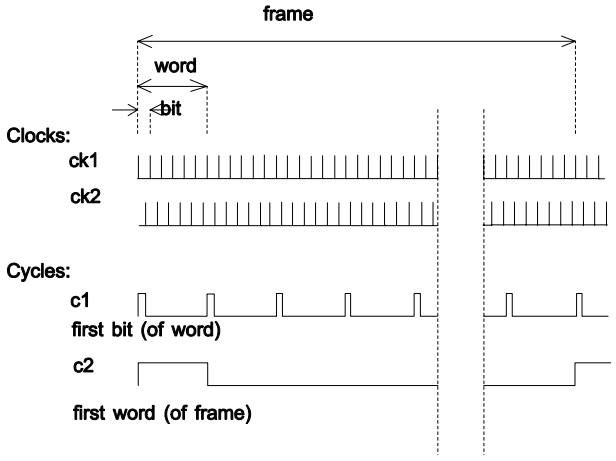
Στα σειριακά συστήματα χρειάζεται να εξασφαλίσουμε σωστό χρονισμό σημάτων καθορίζοντας το μέρος του βασικού κύκλου (basic bit-cycle) κατά τη διάρκεια του οποίου τα μεταδιδόμενα δεδομένα μεταβάλλονται και το μέρος κατά τη διάρκεια του οποίου τα δεδομένα μένουν σταθέρα, για να γίνει η λήψη τους από το επόμενο λειτουργικό στοιχείο. Το παραπάνω γεγονός εξαρτάται απο το διάγραμμα χρονισμού που θα χρησιμοποιηθεί. 'Ετσι ο βασικός κύκλος καθορίζεται από ένα μη επικαλυπτόμενο ρολόι δύο φάσεων (non-overlapping two phase clock) του τύπου που φαίνεται στο σχήμα 3.5. Η σωστή λειτουργία αυτής της διάταξης απαιτεί οι διαδοχικές βαθμίδες ρολογιού να εναλλάσσονται και συμπληρωματικά ρολόγια να χρησιμοποιούνται στην έξοδο και στην είσοδο.



Σχήμα 3.5: Με επικαλυπτόμενο ρολόι 2 φάσεων

Στη συνέχεια θα ασχοληθούμε με τον έλεγχο του χρονισμού ή της ροής των γεγο-νότων διαμέσου ενός δικτύου. Στο πιο χαμηλό επίπεδο, αυτό αφορά το κύριο σύγχρονο ρολόι που ελέγχει τη ροή όλων των δεδομένων στο σύστημα, στο επίπεδο του ενός bit. Ονομάζουμε αυτό το επίπεδο ελέγχου κύκλο 0 (cycle 0) ή πιο απλά C_0.

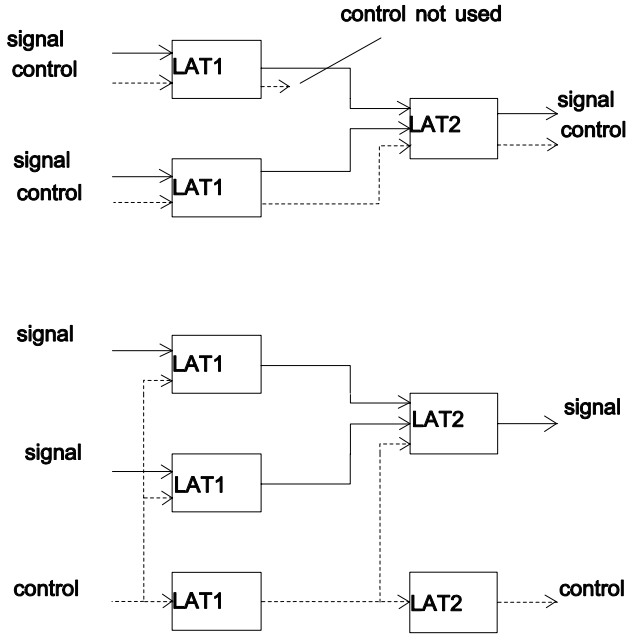
Διάφορα άλλα σήματα ελέγχου που χρειάζονται εκτός του C_0 φαίνονται στο σχήμα 3.6. Το κυριότερο και πιο συνηθισμένο από αυτά είναι το C_1 (cycle 1) που χρειάζεται για να καθορίζει την έναρξη μιας νέας λέξης δεδομένων σε όλους (ή στους περισσότερους) κόμβους του συστήματος. Μία ευνόητη μορφή γι'αυτό το σήμα είναι ένας παλμός ο οποίος συμπίπτει με την παρουσία του LSB της λέξης εισόδου. Δοθέντος του ότι το C_0 είναι ολικά σύγχρονο σήμα, δεν συμβαίνει κάτι παρόμοιο πάντα με το C_1. Αν δεν επιβάλουμε όλα τα λειτουργικά στοιχεία να έχουν latency ίση με αριθμούς ακέραιους όσον αφορά το μήκος λέξης του σήματος, τότε η άφιξη του LSB στους διάφορους κόμβους του συστήματος θα γίνεται σε διαφορετικές χρονικές στιγμές.



Σχήμα 3.6: Κύκλοι ελέγχου

Χρειάζεται λοιπόν να καθοριστεί ένα ξεχωριστό δίκτυο ελέγχου που θα διανέμει τους κύκλους ελέγχου με τον κατάλληλο χρονισμό σε κάθε λειτουργικό στοιχείο. Για το σκοπό αυτό υπάρχουν δύο εναλλακτικές στρατηγικές. Η πρώτη στρατηγική προέρχεται από την αντίληψη ότι το δίκτυο ελέγχου πρέπει να είναι όμοιο με το βασικό δίκτυο, όσον αφορά το latency και όχι όσον αφορά την

λειτουργικότητά του. 'Ενας τρόπος για να πραγματοποιήσουμε αυτή τη στρατηγική είναι να περιλάβουμε αυτή την καθυστέρηση ελέγχου σαν μέρος του κάθε βασικού λειτουργικού στοιχείου. 'Ετσι έλεγχος και δεδομένα ρέουν μέσα στο δίκτυο μαζί και κάθε φορά που ένα σήμα εισάγεται σε κάποιο νέο λειτουργικό στοιχείο εφαρμόζεται στο στοιχείο αυτό το σήμα ελέγχου C_1. Το σχήμα 3.7α δείχνει τη βασική αρχή αυτής της στρατηγικής. Ωστόσο η προσθήκη των κυκλωμάτων καθυστέρησης ελέγχου μέσα σε κάθε λειτουργικό στοιχείο μπορεί να είναι σπάταλη όσον αφορά την περιοχή (area), ειδικά όταν πολλά από τα παραγόμενα σήματα δεν χρησιμοποιούνται.

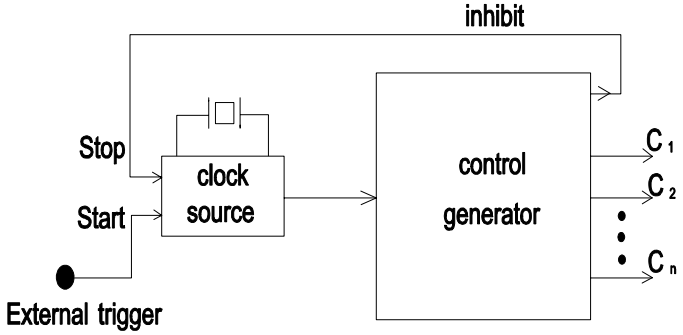


Σχήμα 3.7: Στρατηγικές υλοποίησης ελέγχου

Η δεύτερη στρατηγική είναι να πραγματοποιήσουμε ένα ανεξάρτητο δίκτυο ελέγχου χρησιμοποιόντας ειδικά στοιχεία καθυστέρησης ελέγχου. 'Ετσι χρησιμοποιείται για την υλοποίηση ελέγχου μονάχα ένας απλός "δρόμος" ελέγχου αποφεύγοντας τα προβλήματα που δημιουργούνται από την προσθήκη των κυκλωμάτων καθυστέρησης ελέγχου μέσα σε κάθε λειτουργικό στοιχείο. Το σχήμα 3.7β δείχνει αυτή την στρατηγική. Μια πρόσθετη εξοικονόμηση κυκλώματος μπορεί να γίνει αφού το C_1 είναι κυκλικό και κάθε εκδοχή του που καθυστερείται πέρα από το μήκος ενός κύκλου είναι όμοια με μια προηγούμενη

εκδοχή. 'Ετσι τίθεται ένα ανώτατο όριο στο αριθμό των διαφορετικών εκδοχών του C_1 που χρειάζεται να παραχθούν σε κάθε δίκτυο. Βέβαια εκτός από τον κύκλο 1 (C_1) μπορούμε να χρησιμοποιήσουμε και άλλους κύκλους. Για παράδειγμα ο C_2 χρησιμοποιείται για να υποδείξει τη πρώτη λέξη ενός πλαισίου (frame) λέξεων, ο C_3 για να υποδείξει το πρώτο πλαίσιο μιας ομάδας πλαισίων κλπ. Αυτοί οι "ανώτεροι" κύκλοι ελέγχου χρησιμοποιούνται συνήθως σε πολυπλεγμένες λειτουργίες ελέγχου.

Σαν μια πηγή όλων αυτών των σημάτων ελέγχου χρησιμοποιούμε στα συστήματά μας ένα ειδικό βασικό λειτουργικό στοιχείο που λέγεται γεννήτρια ελέγχου (control generator). Αυτή υλοποιείται σαν ένα σύνολο απο σύχγρονους μετρητές (counters). Η γεννήτρια ελέγχου είναι λοιπόν μία απλή πηγή από την οποία παράγονται τα σήματα ελέγχου ή οι καθυστερημένες εκδοχές τους. Η γενική στρατηγική παραγωγής ελέγχου φαίνεται στο σχήμα 3.8.



Σχήμα 3.8: Στρατηγική παραγωγης ελέγχου

Αυτή περιλαμβάνει μία γεννήτρια ελέγχου (on-chip) και μία πηγή ρολογιού (off-chip) ελεγχόμενη από κρύσταλλο. Η πηγή ρολογιού (clock source) αρχίζει να λειτουργεί όταν δέχεται σήμα από έναν εξωτερικό πυροδοτητή (trigger) και σταματάει μετά ένα ολοκληρωμένο κύκλο του συστήματος μέσω μιας γραμμής "αναστολής" ("inhibit" line). Η γενική αυτή στρατηγική ελέγχου δηλαδή η παραγωγή σημάτων ελέγχου μέσω γεννήτριας υλοποιούμενης από σύγχρονους μετρητές, χρησιμοποιήθηκε και στην εφαρμογή με την οποία ασχολείται ειδικότερα η παρούσα εργασία.

3.5 ΣΕΙΡΙΑΚΕΣ ΣΕ ΕΠΙΠΕΔΟ-ΨΗΦΙΟΥ ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ (DIGIT-SERIAL ARCHITECTURES)

3.5.1 Γενικά

'Οπως είναι γνωστό πολλά πραγματικού χρόνου συστήματα επεξεργασίας σημάτων υλοποιούνται με χρήση αρχιτεκτονικών ειδικού σκοπού. Οι απαιτήσεις όσον αφορά το ρυθμό δειγμάτων (sample rate) σε πραγματικού χρόνου συστήματα επεξεργασίας σημάτων κυμαίνονται από χαμηλές τιμές για επικοινωνίες και επεξεργασία ομιλίας μέχρι υψηλές τιμές για συστήματα ραντάρ και επεξεργασίας εικόνας. 'Ετσι συστήματα ειδικού σκοπού απαιτούν διαφορετικούς ρυθμούς δειγμάτων και χρειάζονται για να υλοποιηθούν διαφορετικούς τρόπους υλοποίησης, όπως είναι οι παράλληλες σε επίπεδο bit αρχιτεκτονικές (bit parallel) και οι σειριακές σε επίπεδο bit αρχιτεκτονικές και σε επίπεδο ψηφίου αρχιτεκτονικές (bit serial, digit serial). Στις σειριακές σε επίπεδο bit τεχνικές, όπως είδαμε στην προηγούμενη ενότητα μεταδίδεται ένα bit στην μονάδα του χρόνου. Οι τεχνικές αυτές είναι ιδανικές για χαμηλής ταχύτητας εφαρμογές και απαιτούν λιγότερες διασυνδέσεις και λιγότερο υλικό (hardware). Στα παράλληλα σε επίπεδο bit συστήματα γίνεται λήψη όλων των bits με λέξη εισόδου σε μία μονάδα του χρόνου (κύκλος ρολογιού). Τα συστήματα αυτά απαιτούν μεγαλύτερη περιοχή (area) και περισσότερες διασυνδέσεις αλλά είναι ιδανικά για εφαρμογές υψηλής ταχύτητας.

Τα συστήματα που χρησιμοποιούν σειριακή σε επίπεδο ψηφίου αρχιτεκτονική επεξεργάζονται περισσότερα από ένα bits εισόδου σε ένα κύκλο ρολογιού. Τα συστήματα αυτά προσφέρονται για εφαρμογές μέσης ταχύτητας για τις οποίες οι σειριακές σε επίπεδο bit αρχιτεκτονικές είναι αργές ενώ οι παράλληλες σε επίπεδο bit αρχιτεκτονικές είναι πιο γρήγορες απ'όσο χρειάζεται. Γενικά λοιπόν οι σειριακές σε επίπεδο ψηφίου τεχνικές (digit serial techniques) είναι ιδανικές στην επεξεργασία ψηφιακών σημάτων προσφέροντας την ελαστικότητα που χρειάζεται για να εκμεταλευτούμε τις δυνατότητες της διαθέσιμης τεχνολογίας. Ο αριθμός των bits που υποβάλλονται σε επεξεργασία σε ένα κύκλο ρολογιού αναφέρεται σαν μέγεθος ψηφίου (digit size). 'Οταν το μέγεθος ψηφίου είναι η μονάδα τότε η αρχιτεκτονική ανάγεται σε σειριακή σε επίπεδο bit, ενώ όταν το

μέγεθος ψηφίου είναι ίσο με το μήκος της λέξης η αρχιτεκτονική ανάγεται σε παράλληλη σε επίπεδο bit. Για μικρά μεγέθη ψηφίου ο ρυθμός δειγμάτων στις σειριακές σε επίπεδο ψηφίου αρχιτεκτονικές αυξάνεται γραμμικά με την αύξηση του μεγέθους ψηφίου. 'Οπως είχαμε αναφέρει και στην προηγούμενη ενότητα το μειονέκτημα των σειριακών σε επίπεδο bit υπολογισμών είναι η σχετικά χαμηλή ρυθμοαπόδοση (throughput) που είναι αποτέλεσμα της σειριακής φύσης της μετάδοσης δεδομένων.

'Ετσι έχουν αναπτυχθεί τεχνικές που επιτυγχάνουν μεγαλύτερη ρυθμοαπόδοση (throughput). Οι πιο σημαντικές είναι οι "radix-4" και "twin-pipe" αρχιτεκτονικές. Στους υπολογισμούς της "radix-4" αρχιτεκτονικής τα δεδομένα διαιρούνται σε ψηφία των δύο bits το καθένα παρέχοντας έτσι διπλάσια ταχύτητα από αυτή των σειριακών σε επίπεδο bit αρχιτεκτονικών. Στις "twin-pipe" αρχιτεκτονικές έχουμε μια αισθητή μείωση της περιοχής (area), λόγω των δύο όμοιων ρευμάτων δεδομένων (twin data streams). Μολονότι οι σειριακές σε επίπεδο bit αρχιτεκτονικές είναι γνωστές εδώ και πολύ καιρό, οι σειριακές σε επίπεδο ψηφίου διερευνήθηκαν μόλις πρόσφατα. Σε πολλές περιπτώσεις χρησιμοποιείται ο "μεταφραστής πυριτίου"(silicon compiler) που ονομάζεται CATHEDRAL III, ο οποίος χρησιμοποιεί παράλληλο σε επίπεδο λέξης υπολογισμό για να διευθήνει σειριακή σε επίπεδο ψηφίου επεξεργασία σημάτων σε εφαρμογές υψηλής ταχύτητας.

3.5.2 Τεχνικές και συμβάσεις για την υλοποίηση σειριακών σε επίπεδο-ψηφίου υπολογισμών

'Οπως αναφέρθηκε και προηγουμένως τα συστήματα που χρησιμοποιούν σειριακή σε επίπεδο ψηφίου αρχιτεκτονική επεξεργάζονται περισσότερα από ένα bit εισόδου σε ένα κύκλο ρολογιού. Ο αριθμός των bits που υποβάλονται σε επεξεργασία σε ένα κύκλο ρολογιού αναφέρεται σαν μέγεθος ψηφίου (digit size). 'Ετσι η ελάχιστη μονάδα πληροφορίας είναι το ψηφίο (digit) και οι σειριακές σε επίπεδο ψηφίου αρχιτεκτονικές επεξεργάζονται ένα ψηφίο σε κάθε κύκλο ρολογιού. Τα ψηφία σε μια πλεονάζουσα παράσταση είναι συνήθως προσεσημασμένα ψηφία (signed digits) δηλαδή κάθε ψηφίο μπορεί να παριστάνει

θετικές και αρνητικές τιμές. Στη σχεδίαση των συστημάτων που χρησιμοποιήσαμε στην εφαρμογή μας έγινε παράσταση των δεδομένων στο σύστημα σταθερής υποδιαστολής συμπληρώματος του δύο (two's complement fixed point data representation), το οποίο έχει τα πλεονεκτήματα που αναφέραμε στο κεφάλαιο 2. Η χρήση της "on line" αριθμητικής που σημαίνει εισαγωγή του πιο σημαντικού bit πρώτου στο σύστημα (MSB first), έχει γενικά υποστηριχθεί για τις σύγχρονες αρχιτεκτονικές. Το μειονέκτημα της "on line" αριθμητικής είναι το αυξημένο μέγεθος των λειτουργικών στοιχείων και η επιπλέον μετατροπή των δεδομένων. Το κατά πόσον αυτό υπερτερεί των πλεονεκτημάτων όπως η μείωση του latency είναι ένα ανοικτό ερώτημα. Επειδή όμως στην εφαρμογή μας υπερτερούν πράξεις όπως η πρόσθεση και η αφαίρεση οι οποίες εκτελούνται ευκολότερα με χρηση αριθμητικής στην οποία εισάγεται πρώτα το λιγότερο σημαντικό bit (LSB first), θα χρησιμοποιήσουμε αυτή την αριθμητική και όχι την "on line" αριθμητική. Θα αναπτύξουμε λοιπόν σειριακές σε επίπεδο ψηφίου αρχιτεκτονικες στις οποίες το μέγεθος ψηφίου (digit size) θα επιλέγεται από τον χρήστη για να εξασφαλιστεί μια καλή ισορροπία μεταξύ της περιοχής (area) και της ταχύτητας (speed) των συστημάτων. Το πιο κατάλληλο μέγεθος ψηφίου για χρήση σε αριθμητικούς υπολογισμούς βρίσκεται σε ένα σημείο μεταξύ της σειριακής σε επίπεδο bit μορφής και της παράλληλης μορφής. Το βέλτιστο μέγεθος ψηφίου εξαρτάται από τα ακριβή χαρακτηριστικά της σχεδίασης, αλλά γιά τις περισσότερες υλοποιήσεις κυμαίνεται μεταξύ των τεσσάρων και οκτώ bits. Αυτό συνεπάγεται ότι για την μέγιστη ρυθμοαπόδοση (throughput) είναι προτιμότερο να χρησιμοποιηθούν πολλαπλά σειριακά σε επίπεδο ψηφίου ρεύματα υπολογισμού από το να χρησιμοποιηθεί ένα απλό ρεύμα παράλληλης αριθμητικής. Τα συστήματα σειριακής σε επίπεδο ψηφίου αρχιτεκτονικής που θα σχεδιάσουμε θα είναι σύγχρονα συστήματα (synchronous systems), στα οποία οι υπολογισμοί θα εκτελούνται στα δεδομένα σε ένα σταθερο γράφο ροής δεδομένων που θα αποτελείται από αριθμητικούς και λογικούς επεξεργαστές. Συνήθως τα συστήματα σύγχρονων αρχιτεκτονικών εκτελούν επεξεργασία τύπου αγωγού (pipelined), έτσι ώστε κάθε επεξεργαστής (λειτουργικό στοιχείο) να απομονώνεται από γειτονικούς επεξεργαστές με μανταλωτές (latches). Οι σειριακές σε επίπεδο ψηφίου διατάξεις που θα περιγράψουμε είναι αρχιτεκτονικές για "hard wired" αλγόριθμους ροής δεδομένων βασισμένες σε μετάδοση αριθμητικών δεδομένων ένα ψηφίο στη μονάδα του χρόνου με λειτουργίες που

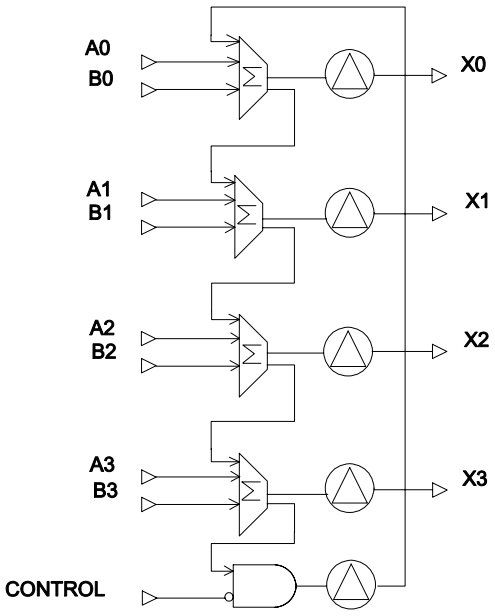
εκτελούνται στα ψηφία δεδομένων όταν αυτά φθάνουν στα λειτουργικά στοιχεία (operators).

Οι "hard wired" αλγόριθμοι ροής δεδομένων περικλείουν μια πλατιά κλίμακα εφαρμογών επεξεργασίας και ελέγχου ψηφιακών σημάτων. Γενικά κάθε αλγόριθμος υπολογισμού μπορεί να υλοποιηθεί σαν ένας "hard wired" αλγόριθμος ροής δεδομένων. Στον τρόπο σχεδίασης που θα εφαρμόσουμε κάθε λειτουργία παρουσιάζεται σαν μια εκδοχή του προγράμματος που περιγράφει ο αλγόριθμος και υλοποιείται από το δικό της λειτουργικό στοιχείο υλικού. 'Ετσι στην διάρκεια ένος κύκλου δείγματος (sample cycle) εκτελείται το ισοδύναμο μιας ολοκληρωμένης επανάληψης του αλγορίθμου.

Οι λειτουργίες εκτελούνται σε μορφή αγωγού (pipelined) κι αυτό γιατί το κύκλωμα έχει υψηλό latency. Το μεγάλο πλεονέκτημα της επεξεργασίας τύπου αγωγού (pipelined processing) είναι ότι ένας μεγάλος αριθμός επαναλήψεων του αλγόριθμου εκτελούνται ταυτόχρονα όλες σε διάφορες βαθμίδες του συστήματος. Επίσης το κύκλωμα σχεδιάζεται όσο το δυνατόν παράλληλο και έτσι ένας αριθμός λειτουργιών εκτελείται στον ίδιο χρόνο. Από τα παραπάνω συμπεραίνουμε ότι η ρυθμοαπόδοση (throughput) του κυκλώματος είναι μια επανάληψη του αλγόριθμου σε κάθε κύκλο δείγματος (sample cycle). Πιο κάτω παρουσιάζουμε μια συγκεκριμένη υλοποίηση σειριακού σε επίπεδο ψηφίου υπολογισμού, αλλά πολλές άλλες στρατηγικές είναι πιθανές. Ο σειριακός σε επίπεδο ψηφίου υπολογισμός βασίζεται στην διαίρεση λέξεων δεδομένων μήκους W bits, σε ένα έως W ψηφία (digits), όπου κάθε ψηφίο αποτελείται από N bits. Ο αριθμός N είναι σταθερός για ένα σειριακό σε επίπεδο ψηφίου κύκλωμα και ο W πρέπει να είναι ένα ακέραιο πολλαπλάσιο του N. Οι διάφορες λέξεις δεδομένων μεταδίδονται ένα ψηφίο στην μονάδα του χρόνου, το λιγότερο σημαντικό ψηφίο πρώτο διαμέσου N γραμμών από το ένα λειτουργικό στοιχείο στο επόμενο. 'Ετσι κάθε λέξη διαιρείται σε W/N ψηφία και μια ολοκληρωμένη λέξη δεδομένων μεταδίδεται κάθε W/N=P κύκλους ρολογιού. Η περίοδος των P κύκλων ρολογιού ονομάζεται περίοδος δείγματος (sample period). 'Οταν δεν υπάρχει κάποιο κενό μεταξύ δύο διαδοχικών λέξεων δεδομένων είναι απαραίτητος κάποιος μηχανισμός που θα υποδεικνύει που τελειώνει η μια λέξη και αρχιζει η επόμενη. Αυτό εξασφαλίζεται μέσω ενός περιοδικού σήματος που αναφέρεται σαν CONTROL το οποίο

περνάει στα διάφορα λειτουργικά στοιχεία. Το σήμα CONTROL είναι σε υψηλή στάθμη (high) για ένα κύκλο ρολογιού ακριβώς σε κάθε περίοδο δείγματος (sample period). Φυσικά όλες οι διαφορετικές καθυστερημένες εκδοχές του σήματος αυτού είναι διαθέσιμες σε ένα σειριακό σε επίπεδο ψηφίου κύκλωμα και τα λειτουργικά στοιχεία μπορούν να συνδέονται σε ένα ή περισσότερα από αυτά τα σήματα ελέγχου, ώστε να επιτευχθεί ο συγχρονισμός. Βέβαια μόνο P διαφορετικά σήματα ελέγχου χρειάζονται λόγω της περιοδικότητας που υπάρχει.

Οι διάφοροι υπολογισμοί εκτελούνται σε σειριακά σε επίπεδο ψηφίου δεδομένα (digit serial data), όταν το κάθε ψηφίο φθάσει σε συγκεκριμένο λειτουργικό στοιχείο. Μία τυπική λειτουργική μονάδα (operator) με σειριακή σε επίπεδο ψηφίου αρχιτεκτονική, είναι ο σειριακός σε επίπεδο-ψηφίου αθροιστής (digit serial adder) που φαίνεται στο σχήμα 3.9. Τα δύο ορίσματα (operands) A και B τροφοδοτούνται ένα ψηφίο στη μονάδα του χρόνου στον αθροιστή.



Σχήμα 3.9: Σειριακός (digit serial) αθροιστής

Στο παράδειγμα μας το μέγεθος ψηφίου είναι N=4. Προστίθενται N bits στη μονάδα του χρόνου με το κρατούμενο να διαδίδεται από την μία βαθμίδα πλήρους αθροιστή (full adder) στην επόμενη. Στο κάτω μέρος το κρατούμενο

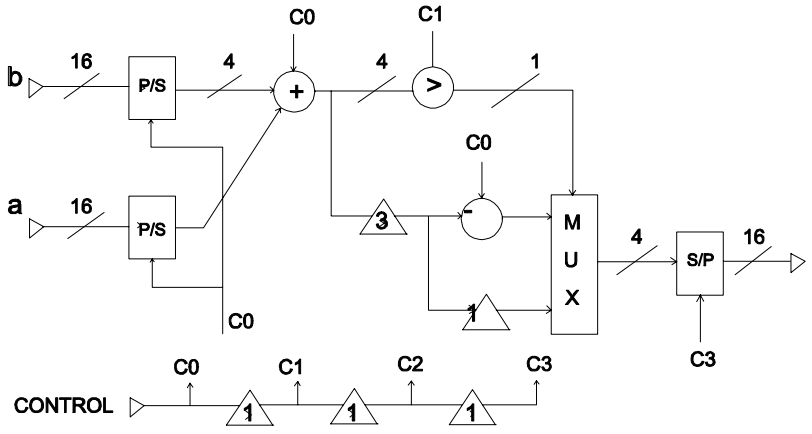
εξόδου από την τελευταία βαθμίδα του πλήρους αθροιστή, καθυστερείται έναν κύκλο ρολογιού μέσω ενός στοιχείου που συμβολίζεται D (delay). Το καθυστερημένο κρατούμενο εξόδου επιστρέφει στην πρώτη βαθμίδα πλήρους αθροιστή στη διάρκεια του επόμενου κύκλου ρολογιού όταν δηλαδή τα επόμενα ψηφία εισόδου έχουν φθάσει. Η πύλη AND στο κάτω μέρος του αθροιστή χρησιμοποιείται για να κάνει reset το κρατούμενο, έτσι ώστε να αρχίσει κανονικά η άθροιση του επόμενου ζεύγους λέξεων εισόδου. Είναι φανερό ότι για να "καθαριστεί" το κρατούμενο στον σωστό χρόνο θα πρέπει ο παλμός του σήματος CONTROL (υψηλή στάθμη) να εισαχθεί στον αθροιστή ταυτόχρονα με την άφιξη του περισσότερου σημαντικού ψηφίου της προηγούμενης λέξης εισόδου.

Σημειώνουμε ότι ο ίδιος αθροιστής μπορεί να χρησιμοποιηθεί για την εκτέλεση πρόσθεσης λέξεων με μήκος κάθε πολλαπλάσιου του μεγέθους ψηφίου (N=4). Αυτό που πρέπει να αλλάξει είναι η περίοδος του εφαρμοζόμενου σήματος ελέγχου CONTROL. Στο πιο πάνω κύκλωμα αλλά και στα περισσότερα κυκλώματα που θα κατασκευάσουμε οι έξοδοι των περισσότερων λειτουργικών μονάδων καθυστερούνται κατά ένα κύκλο ρολογιού για να επιτραπεί επεξεργασία τύπου αγωγού (pipelined) από την μία λειτουργική μονάδα στην επόμενη. Στην περίπτωση του αθροιστή υπάρχει latency ενός κύκλου ρολογιού μεταξύ των λέξεων εισόδου και εξόδου. Είναι φανερό ότι ο αθροιστής του σχήματος 3.9 μπορεί να προσαρμοστεί ώστε να χειριστεί διαφορετικά μεγέθη ψηφίου μεταβάλλοντας τον αριθμό των βαθμίδων του.

Επιπλέον η ταχύτητα ρολογιού (clock speed) της λειτουργίας καθορίζεται από το μέγεθος του χρόνου που απαιτείται για την διάδοση των σημάτων του κρατουμένου (carry signals). Ωστόσο αφού το κρατούμενο δεσμεύεται μετά από Ν διαδόσεις, η ταχύτητα ρολογιού του κυκλώματος είναι ανεξάρτητη του αριθμού των ψηφίων μιάς λέξης. Γενικά το μειωμένο μήκος της διάδοσης μερικών αποτελεσμάτων στα συνδιαστικά τμήματα των σειριακών σε επίπεδο ψηφίου κυκλώματα, συγκρινόμενα με παράλληλα σε επίπεδο bit κυκλώματα επιτρέπει στα σειριακά κυκλώματα να είναι γρηγορότερα. Στα σειριακά σε επίπεδο ψηφίου κυκλώματα δεν έχουν όλα τα σήματα εύρος N bits. Ειδικότερα μερικά σήματα φέρουν απλές λογικές τιμές "1" ή "0". Τέτοιες λογικές τιμές αν και έχουν εύρος

ένα bit ωστόσο έχουν διάρκεια μιάς περιόδου δείγματος (P κύκλους ρολογιού). 'Ετσι μιά λογική τιμή "1" παριστάνεται με P διαδοχικά "1" bits που περνούν σε μιά απλή γραμμή. Με αυτό τον τρόπο απλοποιείται ο συγχρονισμός όλων των δεδομένων στο κύκλωμα. 'Ενα παράδειγμα λειτουργικής μονάδας που παράγει ένα λογικό σήμα ως έξοδο είναι το κύκλωμα του σύγκριτη που συγκρίνει δύο αριθμητικές τιμές εισόδου A και B και δίδει ως έξοδο ένα λογικό "1" σήμα όταν A > B. Από την άλλα πλευρά ένας πολυπλέκτης δέχεται μιά λογική είσοδο σαν ένα σήμα επιλογής χρησιμοποιώντας το για να επιλέξει μεταξύ δύο αριθμητικών τιμών εισόδου αυτή που θα περάσει στην έξοδο. Είναι λοιπόν πολύ πιθανό να παρουσιαστούν σε ένα σειριακό σε επίπεδο ψηφίου κυκλωμα σήματα με μεταβλητό εύρος.

Γενικά λοιπόν είναι άσκοπο να επιμένουμε ώστε οι είσοδοι και οι έξοδοι σε ένα ολοκληρωμένο κύκλωμα (IC) να είναι σε σειριακή σε επίπεδο ψηφίου μορφή. Συχνά σήματα εισόδου/εξόδου περνούν σαν παράλληλες λέξεις και μετατρέπονται εσωτερικά σε σειριακή σε επίπεδο ψηφίου μορφή. 'Ετσι είναι αναγκαίο να σχεδιασθούν Π/Σ και Σ/Π μετατροπείς (P/S and S/P converters) οι οποίοι θα μετατρέπουν τα σειριακά σε επίπεδο ψηφίου ρεύματα δεδομένων σε ένα απλό παράλληλο ρεύμα δεδομένω και αντίστροφα.



Σχήμα 3.10: Τυπικό digit-serial κύκλωμα

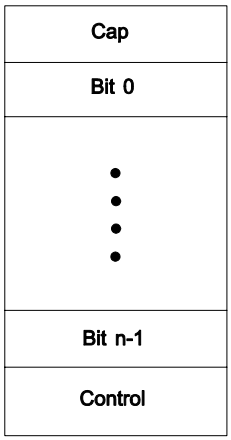
Το σχήμα 3.10 δείχνει ένα τυπικό ολοκληρωμένο σειριακό σε επίπεδο ψηφίου κύκλωμα. Το κύκλωμα αυτό δέχεται δύο 16-bit παράλληλες εισόδους A και B και

υπολογίζει την απόλυτη τιμή του αθροίσματος τους. Το μέγεθος ψηφίου σε αυτό το κύκλωμα είναι N=4 και το μήκος λέξης W=16. 'Ετσι η περίοδος δείγματος είναι τέσσερα και μία νέα είσοδος εισάγεται κάθε τέσσερους κύκλους ρολογιού. Εκτός από τις δύο εισόδους υπάρχει και ένα σήμα εισόδου CONTROL που είναι περιοδικό με περίοδο τέσσερα, ο πρώτος παλμός του συμβαίνει την στιγμή t=0 και επαναλαμβάνεται καθε τέσσερους κύκλου ρολογιού. Τέσσερα διαφορετικά σήματα ελέγχου C_0, C_1, C_2, και C_3 παράγονται με καθυστέρηση του σήματος εισόδου που αναφέρθηκε ως CONTROL. Το σήμα ελέγχου C_i έχει παλμό την στιγμή i και από τότε κάθε τέσσερις διαδοχικούς κύκλους. Για να γίνει κατανοητή η λειτουργία του κυκλώματος είναι καλό να περιγράψουμε ένα ολοκληρωμένο υπολογισμό. Την στιγμή t=0 οι δύο λέξεις εισόδου δεσμεύονται στους Π/Σ μετατροπείς και οι διαδοχικές εξόδοι-ένα ψηφίο στη μονάδα του χρόνου για τους επόμενους τέσσερεις κύκλους ρολογιού-αρχίζουν με το πρώτο ψηφίο την χρονική στιγμή t=1. Τα πρώτα ψηφία προστίθενται την στιγμή t=1 και το άθροισμα εξόδου εμφανίζεται την στιγμή t=2. Το σήμα C_0 χρησιμοποιείται για έλεγχο του αθροιστή αφού ο αθροιστής χρειάζεται ένα σήμα ελέγχου, ένα κύκλο ρολογιού πριν εισαχθεί το πρώτο ψηφίο των δεδομένων εισόδου. Την στιγμή t=2, η έξοδος του αθροιστή περνάει σε ένα συγκριτή ο οποίος καθορίζει το προσημό της. Αφού το bit προσήμου μιάς σειριακής σε επίπεδο ψηφίου τιμής περιέχεται στο τελευταίο ψηφίο το αποτέλεσμα δεν είναι γνωστό μέχρι την στιγμή t=6. Οπότε ο συγκριτής έχει τιμή latency τέσσερις κύκλους ρολογιού. Στον ίδιο χρόνο η αρνητική εκδοχή του αθροίσματος υπολογίζεται. Το αποτέλεσμα της σύγκρισης χρησιμοποιείται την στιγμή t=6 για να επιλεγεί το αρνητικό ή μη αρνητικό άθροισμα. Για να αποφύγουμε όμως τα τελευταία δύο σήματα να φθάσουν στον πολυπλέκτη πριν την έξοδο του συγκριτή, εισάγονται στο κύκλωμα με καθυστερήσεις τριών και ενός κύκλων ρολογιού στις θέσεις που φαίνονται στο σχήμα. Το πρώτο ψηφίο της εξόδου του πολυπλέκτη είναι έτοιμο τη t=7. Τη στιγμή t=10 τα τέσσερα ψηφία του αποτελέσματος έχουν διαβιβαστεί στον Σ/Π μετατροπέα. Τέλος την στιγμή t=11, τα 16 bits εξόδου εμφανίζονται στον δίαυλο (bus) εξόδου αποκρίνοντας έτσι στο σήμα ελέγχου C_3. 'Ετσι το latency του ολοκληρωμένου κυκλώματος είναι 11 κύκλοι ρολογιού. Λόγω της επεξεργασίας τύπου αγωγού (pipelined) που επικρατεί στο κύκλωμα η ρυθμοαπόδοση (throughput) του είναι ένας ολοκληρωμένος υπολογισμός κάθε τέσσερις κύκλους ρολογιού

3.5.3 Σχεδιασμός σειριακών σε επίπεδο-ψηφίου λειτουργικών μονάδων σε επίπεδο layout

Η συλλογικότητα των σειριακών σε επίπεδο ψηφίου κυκλωμάτων μπορεί να περιγραφεί με χαρακτηριστικά ολοκληρωμένων κυκλωμάτων τα οποία παράγονται από ένα "μεταφραστή" πυριτίου (silicon compiler) που ονομάζεται "PARSIFAL". 'Ενα κύκλωμα (chip) το οποίο παράγεται από τον "PARSIFAL" έχει μία γενική μορφή στην οποία σωροί κυττάρων (cell stacks) που παριστάνουν τα σειριακά σε επίπεδο ψηφίου λειτουργικά στοιχεία (digit-serial operators) είναι τακτοποιημένοι σε σειρές (rows) χωρισμένες από κανάλια διαδρομής (routing channels). Κάθε διαφορετικός σωρός κυττάρων (cell stack) αποτελείται από μικρότερα απλά κύτταρα.

Γενικά ροή δεδομένων εντός των σειρών έχουμε από τα αριστερά ή τα δεξιά, ενώ οι διάφορες συνδέσεις δημιουργούνται μέσα στα κανάλια διαδρομής. Για να επιτύχουμε ένα κανονικό και καθιερωμένο τρόπο κατασκευής του "layout" μέσω κυττάρων θα πρέπει οι σωροί κυττάρων που παρουσιάζονται στο "layout" να έχουν το ίδιο ύψος. Το μήκος τους μπορεί να διαφέρει αλλά κάθε διαφορετικός σωρός κυττάρων πρέπει να αποτελείται από κύτταρα του ίδιου μήκους. Η απαίτηση για σταθερό ύψος κυττάρων πρέπει να είναι ανεξάρτητη του μεγέθους ψηφίου έτσι ώστε να μπορεί να κατασκευαστεί κύκλωμα οποιουδήποτε μεγέθους ψηφίου.



Σχήμα 3.11: Περίγραμμα κλασσικού κυττάρου

Για να κατασκευαστεί μια βιβλιοθήκη με αριθμητικές και λογικές λειτουργικές μονάδες οι οποίες θα προσαρμόζονται σε διαφορετικά μεγέθη ψηφίων και θα εξασφαλίζουν σταθερό ύψος όλων των σωρών κυττάρων, υιοθετούμε ένα περίγραμμα κλασσικού κυττάρου (standard cell template). 'Ετσι κάθε λειτουργική μονάδα υλοποιείται με έναν ή περισσότερους σωρούς κυττάρων και κάθε σωρός αποτελείται το ελάχιστο από τρείς διαφορετικούς τύπους απλών κυττάρων. Το σχήμα 3.11 δείχνει το περίγραμμα κλασσικού κυττάρου (standard cell template). 'Οπως μπορούμε να δούμε το περίγραμμα αυτό περιλαμβάνει ένα κύτταρο ελέγχου (control cell), το οποίο φέρει τον VDD δίαυλο (VDD bus), κυκλώματα οδήγησης του ρολογιού (clock buffers) και τυπικές λειτουργίες που απαιτούνται μία για κάθε ψηφίο, όπως είναι η δέσμευση κρατουμένων (latching of carries) κλπ. Πάνω από το κύτταρο ελέγχου, υπάρχουν διατεταγμένα N-bit κύτταρα τα οποία εκτελούν απλές λειτουργίες σε επίπεδο-bit και τέλος στην κορυφή του σωρού υπάρχει το κύτταρο κορυφής (CAP cell), το οποίο φέρει τον VSS δίαυλο (VSS bus) και μικρές συνδέσεις διαδρομής. Κάθε ξεχωριστή λειτουργική μονάδα που λειτουργεί σειριακά σε επίπεδο ψηφίου συγκλίνει όσο αυτό είναι δυνατό στη κλασσική μορφή που αναφέραμε.

Βέβαια διάφορες στρατηγικές μπορούν να προκύψουν από το κλασσικό περίγραμμα. Για παράδειγμα ορισμένες λειτουργικές μονάδες δεν χρειάζονται τόσο μεγάλο κύτταρο ελέγχου (control cell) όπως δείχνεται στο κλασσικό περίγραμμα και έτσι ένα επιπλέον κύτταρο του ενός bit δημιουργείται αν δανειστούμε τον υπολειπόμενο χώρο του κυττάρου ελέγχου. Αυτό δεν είναι χρήσιμο σε αριθμητικά στοιχεία τα οποία εκτελούν λειτουργίες σε ψηφία σταθερού μεγέθους, αλλά είναι πολύ χρήσιμο σε περιπτώσεις που έχουμε καθυστέρηση λογικών σημάτων όπου ένας αριθμός από καθυστερήσεις ενός bit, πρέπει να οργανωθούν σε έναν σωρό κυττάρων. 'Αλλες στρατηγικές σχεδίασης σε επίπεδο "layout" θα δούμε αργότερα, κατά τη διάρκεια της σχεδίασης σειριακών σε επίπεδο ψηφίου λειτουργικών στοιχείων, ειδικού σκοπού.

3.5.4 Απόδοση της σειριακής σε επίπεδο-ψηφίου επεξεργασίας

Για τον υπολογισμό της αποδοτικότητας των διαφόρων αρχιτεκτονικών που

χρησιμοποιούνται για την κατασκευή επεξεργαστών μπορούν να χρησιμοποιηθούν αρκετά μεγέθη. Συνήθως χρησιμοποιούνται η περιοχή (area) στην οποία εξελίσσεται το κύκλωμα και ο χρόνος που απαιτείται για να εκτελεστούν οι υπολογισμοί. Το μέγεθος του χρόνου στα σύγχρονα συστήματα αφορά τον αριθμό των κύκλων ρολογιού που χρειάζονται για την επεξεργασία των δεδομένων και τον ρυθμό με τον οποίο "τρέχει" το ρολόι. Επίσης σε πολλά συστήματα η απόδοση μετριέται με βάση το latency δηλαδή το χρόνο που απαιτείται για την εμφάνιση δεδομένων εξόδου μετά την ανάγνωση των δεδομένων εισόδου και με βάση την ρυθμοαπόδοση (throughput), δηλαδή το ρυθμό με τον οποίο τα δείγματα εισόδου διαβάζονται και παράγονται τα δείγματα εξόδου. Η ρυθμοαπόδοση είναι μία παράμετρος υψηλής σημασίας στα συστήματα επεξεργασίας σημάτων. Επίσης μειώνοντας το latency σε ένα κύκλωμα έχουμε μεγάλο κέρδος κι αυτό επειδή το latency είναι σε πολλές εφαρμογές μία κρίσιμη παράμετρος.

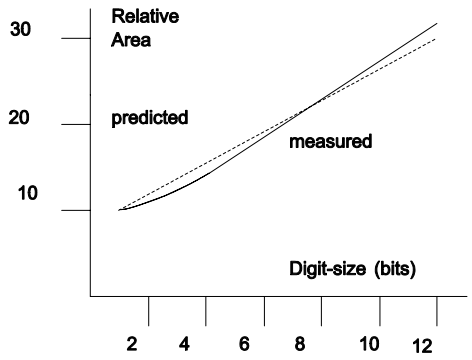
Τυπικά μέτρα της αποδοτικότητας είναι τα γινόμενα που προκύπτουν από την παράσταση: A^k*T^m, όπου A είναι το μέτρο της περιοχής (area), T το μέτρο του χρόνου και κ,m είναι πραγματικές σταθερές. Στη παρούσα ανάλυση θεωρούμε ότι κ=m=1. Αυτή η επιλογή δείχνει μια ίση βαρύτητα της περιοχής (area) και του χρόνου (time). Το αντίστροφο του γινομένου A*T παριστάνει την αποδοτικότητα μιας υλοποίησης σε πυρίτιο (silicon implementation) ενός αλγόριθμου, σαν έναν όρο ισοδύναμο με τη ρυθμοαπόδοση (throughput) ανά μονάδα επιφάνειας πυριτίου.

Υπάρχουν δύο βασικές δυνατότητες για την υλοποίηση λειτουργικών στοιχείων (operators) σε σειριακή αρχιτεκτονική. Στην απλούστερη τα λειτουργικά στοιχεία είναι "bit-sliced" μορφής και ευρύτεροι ή μικρότεροι επεξεργαστές κατασκευάζονται προσθέτοντας ή αφαιρόντας βαθμίδες του ενός bit. Σε αυτή την περίπτωση η μεταβολή της περιοχής (area) με το μέγεθος ψηφίου (digit size) είναι γραμμική. Επίσης η μεταβολή στη περίοδο του ρολογιού η οποία προέκυψε από αλλαγή του μεγέθους ψηφίου είναι γραμμική όταν ο χρόνος που απαιτείται για τη διάδοση των μερικών αποτελεσμάτων διαμέσου των βαθμίδων του ενός bit είναι ανάλογος του αριθμού των βαθμίδων αυτών.

Μία εναλλακτική λύση δημιουργείται όταν τα λειτουργικά στοιχεία χρησιμοποιούν τεχνικές πρόβλεψης (look-ahead techniques). Αυτές γενικά απαιτούν 0(N*logN) περιοχή, αλλά λειτουργούν σε 0(logN) χρόνο. Παραδείγματα είναι οι αθροιστές με πρόβλεψη κρατουμένου (carry look-ahead adders) και οι αθροιστές με ιεραρχική επιλογή κρατουμένου (hierarhical carry select adders). Λειτουργικά στοιχεία που χρησιμοποιούν τις δύο τεχνικές (ripple & look-ahead) θα εξεταστούν όσον αφορά την επίδρασή τους στη σειριακή σε επίπεδο ψηφίου επεξεργασία. Η ανάλυση που ακολουθεί αναφέρεται στον σειριακό σε επίπεδο ψηφίου αθροιστή του σχήματος 3.9. που είναι το βασικό μας παράδειγμα.

Λειτουργικά στοιχεία διάδοσης κρατουμένου (Ripple-carry operators):

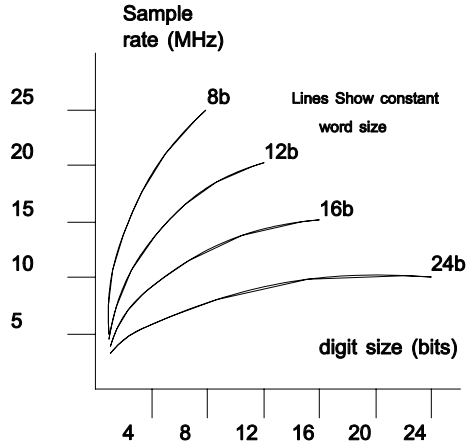
Αρχικά θεωρούμε την "bit-sliced" περίπτωση. Για να καθορίσουμε την περιοχή (area) σημειώνουμε ότι μια σταθερή περιοχή α_0, απαιτείται ανεξάρτητα από το μέγεθος ψηφίου για την εκτέλεση των αρχικών σειριακών λειτουργιών και για να διανέμει τις γραμμές της τροφοδοσίας και του ρολογιού στους επεξεργαστές. Επίσης κάθε βαθμίδα ενός bit απαιτεί περιοχή α_s. Η συνολική περιοχή είναι: $\alpha_0+N\alpha_s$. Μία λογική εκτίμηση είναι $a_0 \cong 4a_s$, βασισμένη στην εξέταση του υλικού μερικών τυπικών λειτουργικών στοιχείων. Το σχήμα 3.12 δείχνει την γραφική παράσταση της συνάρτησης της περιοχής (relative area) με το μέγεθος ψηφίου (digit size), όπου έχουμε χρησιμοποιήσει συντελεστή εκτίμησης α, $a_0 \cong 4$. Επίσης στους ίδιους άξονες έχει σχεδιαστεί το πραγματικό μέγεθος του chip σε συνάρτηση με το μέγεθος ψηφίου, μετρημένο για αλγόριθμο δείγματος που "μεταφράστηκε" από τον "PARSIFAL".



Σχήμα 3.12: Area vs digit-size

Η πραγματική καμπύλη κείται αισθητά κάτω από την προβλεπόμενη καμπύλη για χαμηλές τιμές του μεγέθους ψηφίου. Αυτό συμβαίνει επειδή για μεγάλες τιμές του μεγέθους ψηφίου μειώνεται το μέγεθος διαφόρων μερών του κυκλώματος. Σημειώνουμε ότι για μέγεθος ψηφίου ίσο με 2 το κύκλωμα είναι ασήμαντα μεγαλύτερο από το κύκλωμα με μέγεθος ψηφίου ίσο με 1 (bit-serial) και παρόλα αυτά έχει διπλάσια ρυθμοαπόδοση (throughput). Ακόμη για μέγεθος ψηφίου ίσο με 4 το κύκλωμα είναι μονάχα λίγο μεγαλύτερο. Αυτό είναι ένα πειστικό επιχείρημα για την χρησιμοποίηση σειριακής σε επίπεδο ψηφίου αρχιτεκτονικής.

Μετά τη μελέτη που αφορούσε την περιοχή (area) των κυκλωμάτων μελετάμε αμέσως πιο κάτω την ρυθμοαπόδοση (throughput) των σειριακών σε επίπεδο ψηφίου κυκλωμάτων. Η περίοδος ρολογιού απαρτίζεται από δύο συστατικά. 'Εναν αρχικό χρόνο t_0, και έναν χρόνο t_s για κάθε bit. Προσθέτοντας τα δύο αυτά συστατικά έχουμε την συνολική περίοδο του ρολογιού: t_0+Nt_s. Στη σχεδίαση μας θέτουμε για μία διάδοση χρόνο t_s=2.5ns και αρχικό χρόνο t_0=20ns για νεκρό χρόνο κύκλου, χρόνο αρχικού υπολογισμού και χρόνο διάδοσης σημάτων από το ένα λειτουργικό στοιχείο στο άλλο λόγω πιθανής μεγάλης απόστασης των ενδοσυνδέσεων.

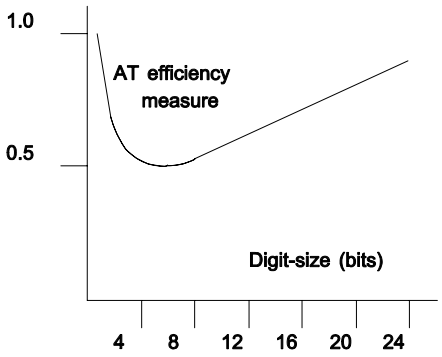


Σχήμα 3.13: Throughput vs digit-size

Σε αυτή τη βάση μια λογική εκτίμηση είναι $t_0 \cong 8t_s$. Το σχήμα 3.13 δείχνει την ρυθμοαπόδοση (throughput) σε συνάρτηση με το μέγεθος ψηφίου για ένα πλήθος από μεγέθη λέξης. Η περίοδος δείγματος υπολογίζεται ως το γινόμενο του αριθμού των κύκλων ανά λέξη δεδομένων και της περιόδου του ρολογιού.

'Ετσι λοιπόν το μέγεθος που δείχνεται στο σχήμα 3.14 είναι το αντίστροφο της περιόδου δείγματος (sample period). 'Οπως προαναφέραμε το αντίστροφο του γινομένου A*T, είναι ένα μέτρο για την αποδοτικότητα της σχεδίασης. 'Ετσι αφού $A=a_0+Na_s$ και ο συνολικός χρόνος ανά δείγμα είναι $T = (t_0 + Nt_s) * \frac{W}{N}$, η αντίστροφη αποδοτικότητα AT είναι $A*T = W (\alpha_0+Nt_s) * (t_0+Nt_s) / N$.

Σημειώνουμε ότι το A*T είναι ανάλογο του W και έτσι η θέση της ελάχιστης τιμής του είναι ανεξάρτητη από το μήκος λέξης. Το σχήμα 3.14 δείχνει το γινόμενο A*T συναρτήσει του μεγέθους ψηφίου για μήκος λέξης ίσο με 24. Για άλλες τιμές του μήκους λέξης η γραφική παράσταση έχει την ίδια μορφή. Μπορεί να καθοριστεί αναλυτικά ότι η ελάχιστη τιμή του A*T επιτυγχάνεται γιά μέγεθος ψηφίου $N = \sqrt{\frac{a_0 t_0}{a_s t_s}}$.



Σχήμα 3.14: Area-time product

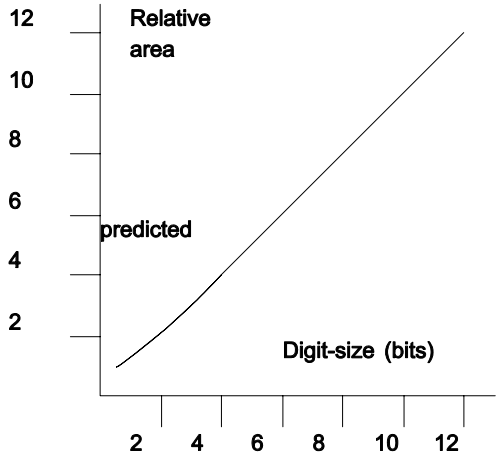
'Ετσι παρατηρούμε ότι η πιο αποδοτική επεξεργασία συμβαίνει όταν το μέγεθος ψηφίου κυμαίνεται από 4 έως 8 bits. Για τις δικές μας υποθέσεις $\alpha_0=4\alpha_s$ και $t_0= 8\ t_s$, η βέλτιστη απόδοση ανεξάρτητα από το μήκος λέξης επιτυγχάνεται όταν το μέγεθος ψηφίου είναι $N \cong 5,6$.

<u>Λειτουργικά στοιχεία πρόβλεψης (look-ahead operators):</u>

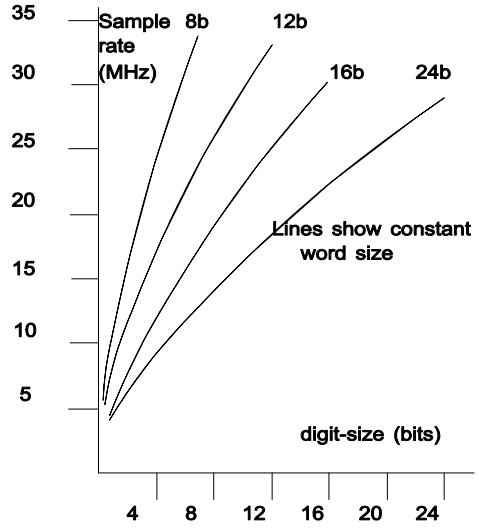
Τα λειτουργικά στοιχεία πρόβλεψης μπορούν να αναλυθούν σε παρόμοια μορφή με τα λειτουργικά στοιχεία διάδοσης, στα οποία αναφερθήκαμε προηγούμενα. Στα στοιχεία αυτά η μεταβολή της περιοχής (area) με το μέγεθος ψηφίου δεν μπορεί να εκτιμηθεί με τόση ακρίβεια όπως στη γραμμική περίπτωση.

Επιλέγοντας: $A= \alpha_0 + \alpha_s N \log_2 2N$ και $T = (t_0 + t_s \log_2 2N) * W/N$ παίρνουμε την επιθυμητή εξάρτηση της περιοχής και του χρόνου από το μέγεθος ψηφίου. Τα σχήματα 3.15, 3.16 και 3.17 είναι τα ανάλογα των σχημάτων 3.12, 3.13 και 3.14 για την περίπτωση της πρόβλεψης. Το σχήμα 3.18 είναι η γραφική παράσταση της αντίστροφης αποδοτικότητας με το μέγεθος του ψηφίου για μήκος λέξης 24 bits. Η μορφή της καμπύλης και η θέση του ελάχιστου είναι και εδώ ανεξάρτητη από το μήκος λέξης, όπως φαίνεται και στον πιο κάτω τύπο:

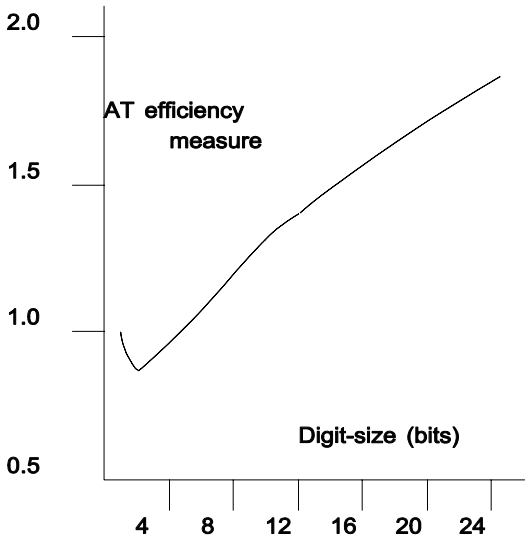
$$A*T = W (\alpha_0 + \alpha_s N \log_2 2N) * (t_0 + t_s \log_2 2N) / N.$$



Σχήμα 3.15



Σχήμα 3.16



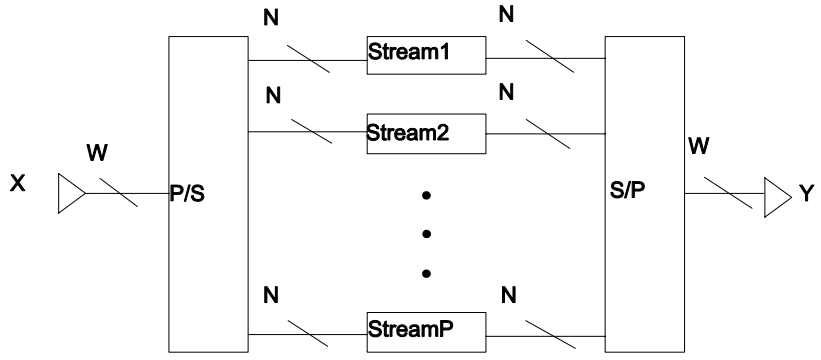
Σχήμα 3.17

Για τις τιμές των λόγων που εκτιμήθηκαν πιο πάνω το ελάχιστο αυτής της έκφρασης ανεξάρτητα του W, μπορεί να υπολογιστεί γραφικά στη τιμή $N \cong 2$, αν και αυτή η βέλτιστη τιμή του μεγέθους ψηφίου εξαρτάται από τις τιμές πραγματικών παραμέτρων.

Επομένως και στις δύο περιπτώσεις (γραμμική και λογαριθμική), υπάρχει μια ισχυρή δικαιολογία για την χρησιμοποίηση σειριακών σε επίπεδο ψηφίου επεξεργαστών, για την επίτευξη αποδοτικής επεξεργασίας. Παρόλο που όλες οι τιμές εδώ είναι βασισμένες στους λόγους a_0/a_s και t_0/t_s που εκτιμήθηκαν αρχικά, τα αποτελέσματα με διαφορετικούς λόγους θα είναι παρόμοια. Αυτό φαίνεται εύκολα στη γραμμική περίπτωση όπου το βέλτιστο μέγεθος ψηφίου (5.6) εξαρτάται μόνο από τη τετραγωνική ρίζα των λόγων περιοχής και χρόνου και συνεπώς είναι σχετικά ανεπηρέαστο σε μια αλλαγή αυτών των λόγων. Στις απλές υλοποιήσεις ένα σειριακό σε επίπεδο ψηφίου σύστημα έχει μικρότερη ρυθμοαπόδοση (throughput) από ένα παράλληλο σε επίπεδο-bit σύστημα. Ωστόσο αυτή η απώλεια μετριάζεται με μια αύξηση στο ρυθμό του ρολογιού του συστήματος. Γενικά η αποδοτικότητα αυξάνεται όταν υπάρχει μείωση στην περιοχή (area) των λειτουργικών στοιχείων και των ενδοσυνδέσεων.

3.5.5 Χρήση "ανάπτυξης" (unfolding) για την αύξηση της απόδοσης και της ταχύτητας κυκλωμάτων

'Οπως είδαμε στη προηγούμενη ενότητα η βελτίωση στο ρυθμό ρολογιού, η οποία επιτυγχάνεται με την χρησιμοποίηση μεγέθους ψηφίου μικρότερου από το μήκος λέξης είναι μία από τις βασικές αιτίες για την χρησιμοποίηση σειριακών σε επίπεδο ψηφίου αρχιτεκτονικών έναντι στις παράλληλες σε επίπεδο-bit αρχιτεκτονικές. Από την άλλη πλευρά όσον αφορά την ρυθμοαπόδοση (throughput) υπάρχει πρόβλημα επειδή ο ρυθμός λειτουργίας διαιρείται με τον αριθμό των ψηφίων μιας λέξης (P). Επομένως η ολική απόδοση θα είναι μικρότερη σε ένα σειριακό σε επίπεδο ψηφίου κύκλωμα από αυτή ενός παρράλληλου σε επίπεδο-bit κυκλώματος εκτός αν ο ρυθμός ρολογιού γίνει μεγαλύτερος. Η απώλεια αυτή μετριάζεται αν δημιουργήσουμε μια οργάνωση του κυκλώματος τέτοια ώστε οι υπολογισμοί να εκτελούνται σε P ταυτόχρονα σειριακά σε επίπεδο ψηφίου ρεύματα δεδομένων (data streams). Η μέθοδος αυτή αναφέρεται ως "ανάπτυξη" (unfolding).



Σχήμα 3.18: Παράλληλα ρεύματα υπολογισμού

Η "ανάπτυξη" επιτρέπει την επίτευξη του υψηλότερου ρυθμού ρολογιού με χρήση ψηφίων μεγέθους N=W/P και τη λήψη αποτελεσμάτων κάθε κύκλο ρολογιού, όπως ακριβώς και στην παράλληλη σε επίπεδο-bit περίπτωση. Στην ουσία η μέθοδος αυτή απαιτεί τη χρησιμοποίηση Π/Σ και Σ/Π μετατροπείς (parallel/serial & serial/parallel converters) για να δημιουργηθούν και μετά να συγχωνευτούν τα παράλληλα ρεύματα υπολογισμού (parallel computation streams), όπως φαίνεται στο σχήμα 3.18. Χρησιμοποιώντας αυτούς τους

μετατροπείς είναι εύκολο να διαιρεθεί μία λέξη σε P ψηφία στη συνέχεια να εκτελεστούν P ταυτόχρονοι υπολογισμοί και τέλος να συγχωνευτούν τα αποτελέσματα στην έξοδο. Σημειώνουμε ότι η λειτουργία της δημιουργίας και της συγχώνευσης δεν είναι ορατή από τον έξω κόσμο.

'Ενας Π/Σ μετατροπέας σαν αυτούς του σχήματος 3.19, αποτελείται από P διαφορετικούς καταχωρητές με παράλληλη είσοδο και σειριακή σε επίπεδο ψηφίου έξοδο, συνδεδεμένους στον ίδιο δίαυλο εισόδου (input bus). Κάθε τέτοιος καταχωρητής λειτουργεί δεσμεύοντας την τιμή που υπάρχει στο δίαυλο εισόδου και στέλνοντάς την έπειτα στην έξοδο- ένα ψηφίο στη μονάδα του χρόνου. Αυτό γίνεται για τους επόμενους P κύκλους ρολογιού μετά τους οποίους η νέα τιμή εισόδου δεσμέυεται στους καταχωρητές. 'Ενας Σ/Π μετατροπέας εκτελεί την "δυαδική" λειτουργία αυτής που εκτελεί ο Π/Σ μετατροπέας. Στην περίπτωση αυτή P διαφορετικά σειριακά σε επίπεδο ψηφίου ρεύματα εισόδου τροφοδοτούνται σε καταχωρητές με σειριακή είσοδο και παράλληλη έξοδο και περνούν με κυκλική σειρά σε ένα κοινό δίαυλο εξόδου τριών καταστάσεων (tri-state output bus). Τα διαδοχικά σειριακά σε επίπεδο ψηφίου ρεύματα εισόδου καθυστερούνται κατά ένα κύκλο ρολογιού το καθένα έτσι ώστε να διατηρείται ο απαραίτητος συγχρονισμός μεταξύ των σειριακών σημάτων εισόδου και των σημάτων που οδηγούν τον δίαυλο (bus), του Σ/Π μετατροπέα.

Στη πιο πάνω περιγραφή υποθέσαμε ότι ο αριθμός των ρευμάτων υπολογισμού είναι ίσος με τον αριθμό των ψηφίων κάθε λέξης (P). Αυτό δεν είναι απαραίτητο αφού γενικά ο αριθμός των ρευμάτων υπολογισμού μπορεί να είναι οποιοσδήποτε αριθμός μικρότερος του P. Στη περίπτωση αυτή τα δεδομένα δεν θα δεσμεύονται από το δίαυλο εισόδου (input bus) σε κάθε κύκλο ρολογιού αλλά με κάποια περιοδικότητα. Παρόμοια τα δεδομένα εξόδου δεν τοποθετούνται στον δίαυλο εξόδου τριών καταστάσεων σε κάθε κύκλο ρολογιού.

Μία συνηθισμένη περίπτωση είναι αυτή στην οποία ο αριθμός των ρευμάτων είναι διαιρέτης του P και ο ρυθμός του εσωτερικού ρολογιού του κυκλώματος είναι μεγαλύτερος από το ρυθμό δεδομένων μέσα στο κύκλωμα. Για παράδειγμα με μήκος λέξης 24 και μέγεθος ψηφίου 3, το P είναι ίσο με 8. Με δύο παράλληλους υπολογισμούς τα δεδομένα διαβάζονται από το δίαυλο εισόδου

κάθε 4 κύκλους ρολογιού. 'Ετσι η περίοδος δείγματος (sample period) του κυκλώματος είναι 8 κύκλοι ρολογιού ανά ρεύμα δεδομένων, ενώ η περίοδος δείγματος παράλληλων δεδομένων είναι 4 κύκλοι ρολογιού. Εναλλακτικά 4 ταυτόχρονοι υπολογισμοί επιτρέπουν νέα είσοδο κάθε δεύτερο κύκλο ρολογιού.

Στη γενική περίπτωση όμως η κατάσταση δεν είναι τόσο απλή όπως φαίνεται στο σχήμα 3.19 και συνήθως δεν μπορούμε να πάρουμε αποτέλεσμα με ανεξάρτητα ρεύματα υπολογισμού. Γενικά τα παράλληλα ρεύματα υπολογισμού αλληλεπιδρούν μεταξύ τους. Για το λόγο αυτό έχουν μελετηθεί γενικότερες και αυτόματες μέθοδοι "ανάπτυξης" (unfolding).

3.5.6 Συμπεράσματα

Στις προηγούμενες ενότητες περιγράψαμε μια νέα τεχνική για την σχεδίαση αριθμητικών επεξεργαστών οι οποίοι εκτελούν επεξεργασία τύπου αγωγού (pipeline processing). Τα βασικά χαρακτηριστικά αυτής της τεχνικής είναι η υψηλή απόδοση (=500 Mops per IC), σε συνδιασμό με την ευκολία σχεδίασης. Το βασικό κίνητρο για την χρησιμοποίηση των σειριακών σε επίπεδο ψηφίου συστημάτων έναντι στα σειριακά σε επίπεδο-bit και στα παράλληλα σε επίπεδο-bit συστήματα, είναι η υψηλή αποδοτικότητά τους. Αυτή όπως είδαμε έχει άμεση σχέση με το γινόμενο A*T (area*time). Επίσης στα σειριακά σε επίπεδο ψηφίου κυκλώματα έχουμε αισθητή μείωση της περιοχής των λειτουργικών στοιχείων και των ενδοσυνδέσεων.

Κεφάλαιο 4

Υπολογισμός τετραγωνικής ρίζας δυαδικών αριθμών

4.1 ΓΕΝΙΚΗ ΘΕΩΡΙΑ ΑΡΙΘΜΗΤΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ

Αριθμητικός αλγόριθμος είναι ένα σύνολο λειτουργιών οι οποίες ακολουθούν η μια μετά την άλλη ή δημιουργούν βρόγχους (loops) κατά έναν ορισμένο τρόπο, ώστε με την εκτέλεσή τους να επιλύουν ένα ορισμένο αριθμητικό πρόβλημα. Οι στοιχειώδεις αυτές λειτουργίες πρέπει να είναι άμεσα εκτελέσιμες από το υλικό που χρησιμοποιείται για τη λύση του προβλήματος. Επίσης οι λειτουργίες αυτές πρέπει να εκτελούνται σε διάκριτες χρονικές στιγμές. Οι απεριόριστοι βρόγχοι δεν επιτρέπονται κι αυτό γιατί οι αλγόριθμοι πρέπει να είναι αυτοτερματιζόμενοι στο τέλος του υπολογισμού.

'Οπως έχουμε αναφέρει και σε προηγούμενες ενότητες κάθε αριθμητική σχεδίαση μπορεί να διαιρεθεί σε δύο φάσεις. Η πρώτη είναι η ανάπτυξη αποδοτικών αλγόριθμων και η δεύτερη είναι η ανάπτυξη της λογικής υλοποίησης των αλγόριθμων αυτών. Και οι δύο φάσεις απαιτούν εκτεταμένη σχεδιαστική προσπάθεια. Η ανάπτυξη του αλγόριθμου είναι πιο δημιουργική εργασία ενώ η υλοποίηση απαιτεί κάποιο υπόβαθρο στον τομέα της αρχιτεκτονικής συστημάτων. Ωστόσο είναι σωστό να αναφέρουμε ότι η μία φάση στηρίζει την άλλη και ότι μονάχα με την μία δεν μπορούμε να επιτύχουμε μια

αποδοτική σχεδίαση. Οι αριθμητικοί αλγόριθμοι ταξινομούνται σε δύο γενικές κατηγορίες σύμφωνα με τα μεγέθη δεδομένων (data sizes).

Α.Ψηφιακοί αλγόριθμοι (digital algorithms):

Οι αλγόριθμοι αυτοί χειρίζονται ξεχωριστά ψηφία (digits) των ορισμάτων εισόδου και παράγουν το αποτέλεσμα σε μια ολική βάση. Οι αριθμητικοί κανόνες που διέπουν την λειτουργία ενός πλήρους αθροιστή 1-bit (F.A) δίνουν ένα παράδειγμα ψηφιακού αλγόριθμου. Συνήθως οι ψηφιακοί αλγόριθμοι υλοποιούνται με συνδιαστικά ή ακολουθιακά κυκλώματα μικρού ή μέσου μεγέθους, με περιορισμένες εισόδους/εξόδους. Στη περίπτωση των μη δυαδικών συστημάτων κάθε ψηφίο πρέπει να παριστάνεται εσωτερικά μέσω μιάς δυαδικής κωδικοποίησης. Η πολυπλοκότητα των ψηφιακών αλγορίθμων δεν αυξάνεται απαραιτήτως με την αύξηση της τιμής βάσης (radix) ενός συστήματος. Ωστόσο η πολυπλοκότητα του κυκλώματος αυξάνεται αρκετά στα συστήματα με υψηλή βάση.

Β. Διανυσματικοί αλγόριθμοι (vector algorithms):

Οι αλγόριθμοι αυτοί χειρίζονται τα ορίσματα εισόδου σαν διανυσματικές ποσότητες. 'Ετσι μπορεί κάποιος να οργανώσει έναν αριθμό από ψηφιακούς αλγόριθμους ώστε να δημιουργήσει έναν διανυσματικό αλγόριθμο. Επίσης μπορούμε να δούμε τον ψηφιακό αλγόριθμο σαν ένα μικροσκοπικό επίπεδο επεξεργασίας, το οποίο αντιστοιχεί στο επίπεδο της λογικής σχεδίασης με όρους "Boolean" εξισώσεων ή με πίνακες αληθείας. Οι διανυσματικοί αλγόριθμοι χειρίζονται αριθμητικά δεδομένα σε μακροσκοπικό επίπεδο με όρους διανυσματικών αριθμητικών εξισώσεων. Το αποτέλεσμα εμφανίζεται σε μια μορφή "ψηφίο-διάνυσμα" ("digit-vector") μετά από την κανονική εκτέλεση στοιχειωδών αριθμητικών λειτουργιών. Επίσης μπορεί να προκύψει στο αποτέλεσμα και ένα διάνυσμα κατάστασης (condition vector). Για παράδειγμα ένα διάνυσμα κατάστασης 1-bit μπορεί να δηλώνει καταστάσεις όπως υπερχείλιση (overflow), ανίχνευση μηδενός (zero detection), κλπ.

'Οσον αφορά τώρα τον τρόπο παράστασης, αναφέρουμε ότι οι διανυσματικοι αλγόριθμοι παριστάνονται μέσω διαγραμμάτων ροής (flow charts) και οι ψηφια-

κοί αλγόριθμοι αν χρειάζεται, μέσω "Boolean" εξισώσεων.

4.2 ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΣΕΙΡΙΑΚΕΣ ΣΕ ΕΠΙΠΕΔΟ ΨΗΦΙΟΥ ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ

Οι σειριακές σε επίπεδο ψηφίου αρχιτεκτονικές είναι αρχιτεκτονικές για "hard-wired" αλγόριθμους ροής δεδομένων, βασισμένους στη μετάδοση δεδομένων ένα ψηφίο στη μονάδα του χρόνου με λειτουργίες που εκτελούνται στα ψηφία δεδομένων όταν αυτά φθάσουν στα λειτουργικά στοιχεία. Οι "hard-wired" αλγόριθμοι ροής δεδομένων περικλύουν ένα μεγάλο φάσμα εφαρμογών ψηφιακής επεξεργασίας σημάτων και εφαρμογών ελέγχου. Γενικά κάθε αριθμητικός υπολογιστικός αλγόριθμος μπορεί να υλοποιηθεί σαν "hard-wired" αλγόριθμος ροής δεδομένων. Στις περιπτώσεις της σειριακής σε επίπεδο ψηφίου αρχιτεκτονικής που χρησιμοποιούμε στην εφαρμογή μας κάθε λειτουργία παρουσιάζεται σαν μία "in-line" εκδοχή του προγράμματος που περιγράφεται από τον αλγόριθμο. Η εκδοχή αυτή υλοποιείται από το αντίστοιχο λειτουργικό στοιχείο. 'Ετσι στη διάρκεια κάθε κύκλου δείγματος (sample cycle) εκτελείται το ισοδύναμο μιάς ολοκληρωμένης επενάληψης του αλγόριθμου.

Η ρυθμοαπόδοση (throughput) του κυκλώματος είναι μια επανάληψη του αλγόριθμου ανά κύκλο δείγματος. Στην επόμενη ενότητα θα ασχοληθούμε με τον σειριακό σε επίπεδο ψηφίου υπολογισμό της τετραγωνικής ρίζας δυαδικών αριθμών ο οποίος είναι βασισμένος στο γενικό πλαίσιο των σύγχρονων αλγόριθμων σταθερού δυκτίου (fixed-network synchronous algorithms). Ο αλγόριθμος που θα χρησιμοποιηθεί έχει προσαρμοστεί έτσι ώστε να υλοποιείται σε κύκλωμα το οποίο θα εκτελεί επεξεργασία τύπου αγωγού (pipelined processing).

4.3 ΑΛΓΟΡΙΘΜΟΣ ΥΠΟΛΟΓΙΣΜΟΥ ΤΗΣ ΤΕΤΡΑΓΩΝΙΚΗΣ ΡΙΖΑΣ ΔΥΑΔΙΚΩΝ ΑΡΙΘΜΩΝ

Στον τομέα των αριθμητικών επεξεργαστών για να επιτύχουμε βελτιώσεις στην ταχύτητα αλλά και στο μέγεθος των συστημάτων έχει κριθεί αναγκαία η

κατασκευή κυκλωμάτων τα οποία θα υπολογίζουν ειδικές αριθμητικές συναρτήσεις. Μία από αυτές με την οποία και θα ασχοληθούμε είναι η τετραγωνική ρίζα δυαδικών αριθμών. Πιο κάτω περιγράφουμε ένα γενικό αλγόριθμο, για τον υπολογισμό της τετραγωνικής ρίζας δυαδικών αριθμών.

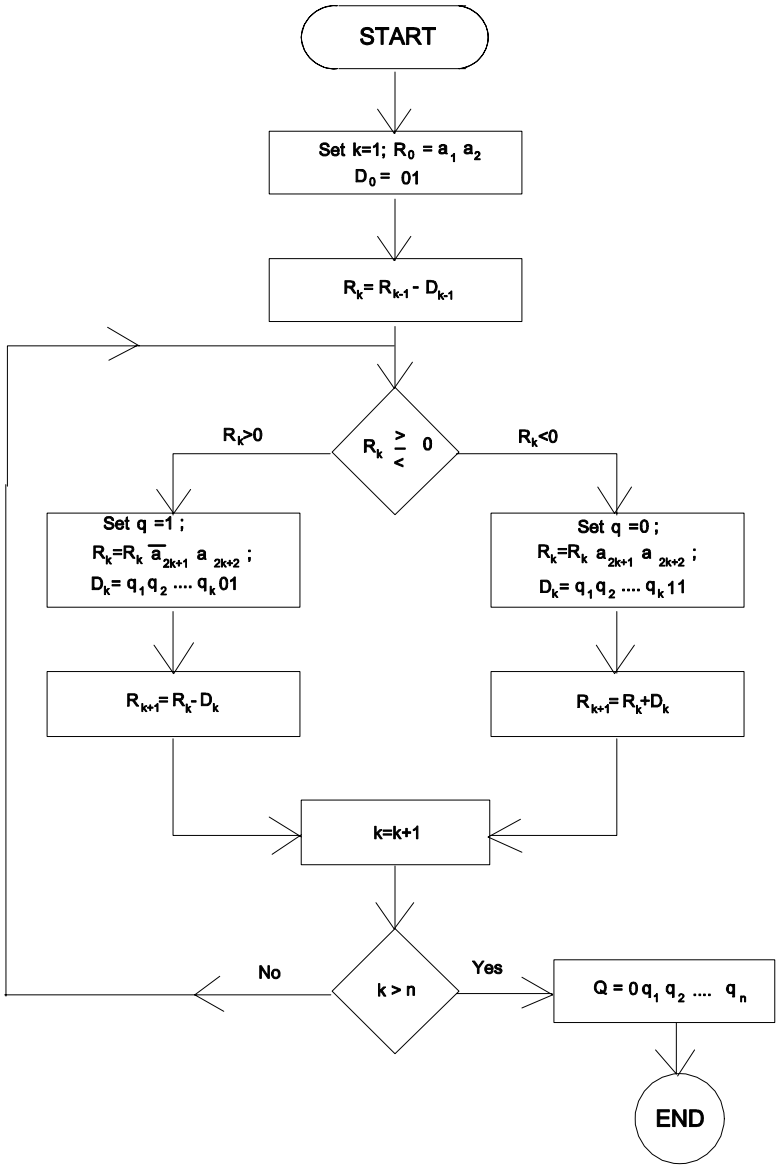
Ο υπολογισμός της τετραγωνικής ρίζας γενικά απαιτεί ελεγχόμενες λειτουργίες πρόσθεσης και αφαίρεσης. Χρησιμοποιούμε εδώ τον αλγόριθμο "μη-αποκατάστασης" (non-restoring) κι αυτό επειδή το μερικό υπόλοιπο που προκύπτει για λειτουργίες πρόσθεσης ή αφαίρεσης με αρνητικό αποτέλεσμα (sign-bit=1) δεν αποκαθίσταται εξασφαλίζοντας έτσι τον έλεγχο της επόμενης λειτουργίας. Θεωρούμε δύο θετικούς κλασματικούς δυαδικούς αριθμούς A και Q, έτσι ώστε $Q=\sqrt{A}$ ή $A=Q^2$. Ορίζουμε λοιπόν το Q σαν την τετραγωνική ρίζα του A. Σημειώνουμε λοιπόν τα εξής: $Q=\sqrt{A}=0.q_1q_2...q_n \quad A=Q^2=0.a_1a_2...a_{2n-1}a_{2n}$ όπου $0 < A < 1$ και $0 < Q < 1$.

Ο κλασσικός αλγόριθμος "μη-αποκατάστασης" (non-restoring algorithm) για τον υπολογισμό της τετραγωνικής ρίζας δυαδικών αριθμών φαίνεται στο σχήμα 4.1 στη μορφή διαγράμματος ροής. Ο δυαδικός αριθμός A διαιρείται σε ζεύγη από την υποδιαστολή και μετά ως εξής: $a_1a_2,a_3a_4,...,a_{2n-1}a_{2n}$. Η πρώτη πράξη είναι πάντα αφαίρεση. Ο αφαιρετέος στην πράξη αυτή είναι ο $D_0=0.01$, ο οποίος αφαιρείται από τον a_1a_2. Αν το υπόλοιπο R_1 είναι μη αρνητικό τότε $q_1=1$ και το επόμενο ζεύγος a_3a_4 προσαρτάται στο R_1. Η επόμενη πράξη είναι αφαίρεση, όπου ο $D_1=0.q_101$ αφαιρείται από τον R_1. Αν όμως το R_1 είναι αρνητικό, τότε $q_1=0$, το επόμενο ζεύγος a_3a_4 προσαρτάται στο R_1 και ο $D_1=0.q_111$ προστίθεται σε αυτό. Γενικά το κ-οστο ενδιάμεσο βήμα του αλγόριθμου χρειάζεται τις ακόλουθες λειτουργίες, πρίν το καθορισμό του κ-οστου bit αποτελέσματος q_k:

$R_{k+1} \leftarrow R_k \cdot a_{2k+1}a_{2k+2} - q_1q_2...q_k01$, αν $q_k=1$

$R_{k+1} \leftarrow R_k \cdot a_{2k+1}a_{2k+2} + q_1q_2...q_k11$, αν $q_k=0$

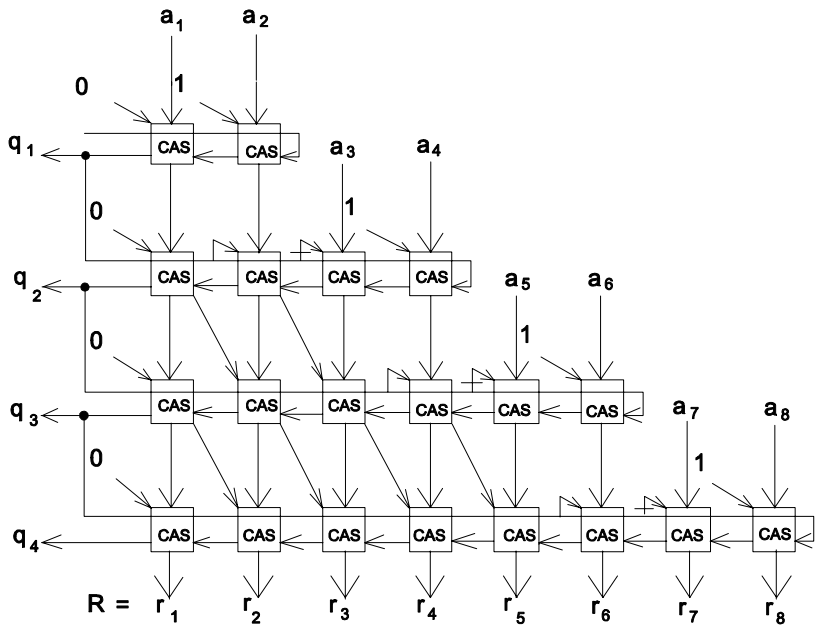
όπου η λειτουργία προσάρτησης που συμβολίζεται με "." μπορεί να επιτευχθεί με ολίσθηση του προηγούμενου υπολοίπου δύο bits προς τα αριστερά και με το νέο ζεύγος να εισάγεται από το δεξιό άκρο.



Σχήμα 4.1: Αλγόριθμος μη-αποκατάστασης

Στη συνέχεια για να δούμε τον τρόπο με τον οποίο υλοποιείται ένας τέτοιος αλγόριθμος δίνουμε στο σχήμα 4.2 μία διάταξη αποτελούμενη από C.A.S κύτταρα (controlled add-subtract cells). 'Οπως βλέπουμε σε κάθε σειρά της διάταξης μεταδίδεται ένα νέο ζεύγος των bits εισόδου. Χρησιμοποιείται εδώ αριθμητική συμπληρώματος του 2 (2's complement arithmetic) για την εκτέλεση της λειτουργίας της αφαίρεσης, μέσω πρόσθεσης με το συμπλήρωμα του 2. Η λειτουργία ελέγχου μπορεί να επιτευχθεί αν μεταδώσουμε τα κρατούμενα

εξόδου της κάθε σειράς της διάταξης στα C.A.S κύτταρα της επόμενης γραμμής. Η διάταξη που φαίνεται στο σχήμα 4.2, υπολογίζει την τετραγωνική ρίζα ενός δυαδικού αριθμού των 9 bits ($A=0.a_1a_2...a_9$) και δίνει αποτέλεσμα με μήκος 5 bits $(Q=\sqrt{A}=0.q_1q_2q_3q_4)$.



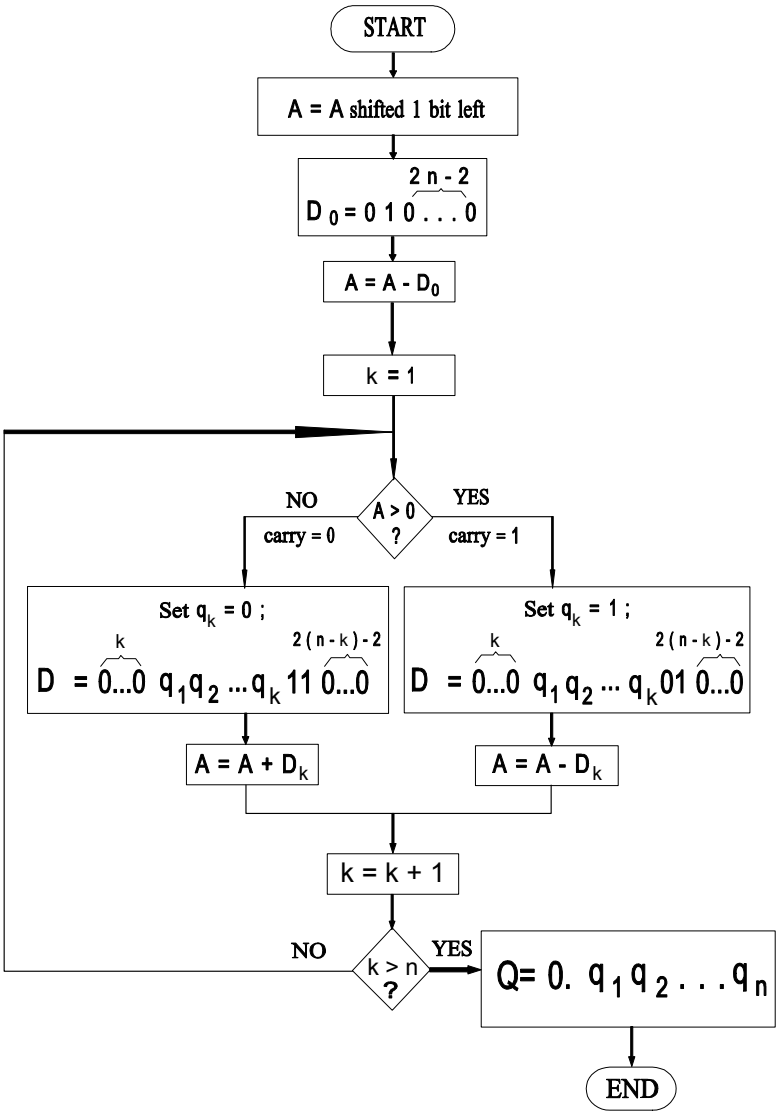
Σχήμα 4.2: Διάταξη CAS κυττάρων

Σαν ένα παράδειγμα έχουμε τον υπολογισμό της τετραγωνικής ρίζας του αριθμού Α με: $A=(0.10101001)_2=(\frac{169}{256})_{10}$ όπου με χρήση του αλγόριθμου του σχήματος 4.1 έχουμε:

10101001
11 [αφαίρεση με χρήση συμπληρώματος του 2]
q_1=1_0110
011 [αφαίρεση με χρήση συμπληρώματος του 2]
q_2=1_00110
0011 [αφαίρεση με χρήση συμπληρώματος του 2]
q_3=0_100101
11011 [πρόσθεση]
q_4=1_00000

Τελικά: $Q = (0.1101)_2 = (\frac{13}{16})_{10}$.

Είναι όμως αναγκαίο να γίνουν κάποιες μετατροπές στον κλασσικό αλγόριθμο υπολογισμού της τετραγωνικής ρίζας δυαδικών αριθμών που παρουσιάστηκε ώστε να προσαρμοστεί αυτός στις σχεδιαστικές ανάγκες της σειριακής σε επίπεδο ψηφίου τεχνικής (digit-serial technique) που θα χρησιμοποιήσουμε για την κατασκευή του τελικού κυκλώματος το οποίο είναι και το βασικό αντικείμενο της παρούσας εργασίας (βλέπε κεφ.5).



Σχήμα 4.3

Θεωρούμε και πάλι δύο θετικούς κλασματικούς δυαδικούς αριθμούς A και Q, έτσι ώστε $Q = \sqrt{A}$. Η μορφή των δύο αυτών αριθμών φαίνεται πιο κάτω: $Q=0.q_1q_2...q_n$ και $A=0.a_1a_2...a_{2n-1}$ όπου: $0 < A < 1$ και $0 < Q < 1$. Παρατηρούμε, ότι το μήκος του αριθμού A είναι 2η bits και το μήκος της τετραγωνικής του ρίζας (Q), είναι (η+1) bits. Στη περίπτωση του κυκλώματος που θα υλοποιήσουμε στο επόμενο κεφάλαιο το μήκος του αριθμού A θα είναι 16 bits, ενώ το μήκος του Q θα είναι 9 bits.

Ο αλγόριθμος που τελικά θα χρησιμοποιηθεί για την κατασκευή του κυκλώματος που θα γίνει στο επόμενο κεφάλαιο φαίνεται στο σχήμα 4.3 σε μορφή διαγράμματος ροής. Τα ορίσματα D_k που προστίθονται ή αφαιρούνται από τον αρχικό αριθμό A και από τα μερικά υπόλοιπα που προκύπτουν κατά τη διάρκεια του υπολογισμού απαρτίζονται στις περισσότερο σημαντικές τους θέσεις από κ μηδενικά. Δηλαδή στο D_1 το MSB είναι μηδενικό, στο D_2 τα δύο πιο σημαντικά bits είναι μηδενικά κ.ο.κ.

Επίσης στις λιγότερο σημαντικές θέσεις των ορισμάτων, προσαρτούμε τόσα μηδενικά ώστε να επιτύχουμε το μήκος τους να είναι ίσο με αυτό του αρχικού αριθμού A και των μερικών υπολοίπων που προκύπτουν κατά την διάρκεια του υπολογισμού, δηλαδή 2η bits (στη περίπτωση μας 16 bits). Με τον συμβολισμό A=A<<1, στο διάγραμμα ροής εννοούμε ολίσθηση του A κατά ένα bit αριστερά, οπότε το bit-πρόσημο (sign-bit) "φεύγει" και προσαρτάται στη λιγότερο σημαντική θέση του A ένα μηδενικό. Δηλαδή: $0.a_1a_2\cdots a_{2n-1} \rightarrow a_1a_2\cdots a_{2n-1}0$. Η ολίσθηση αυτή γίνεται στην αρχή του υπολογισμού έτσι ώστε να εφαρμοστεί από εκεί και πέρα ο αλγόριθμος στα υπόλοιπα bits και όχι στο πρόσημο του A. 'Οσον αφορά τώρα τα bits του αποτελέσματος, δηλαδή του αριθμού Q, παρατηρούμε ότι το q_k bit είναι το κρατούμενο (carry) της κ-οστης πράξης, που μπορεί να είναι πρόσθεση ή αφαίρεση.

Στον υπολογισμό χρησιμοποιούμε αριθμητική συμπληρώματος του 2, για την εκτέλεση της λειτουργίας της αφαίρεσης μέσω πρόσθεσης με το συμπλήρωμα του 2. Τέλος πιο κάτω δίνουμε ένα παράδειγμα υπολογισμού τετραγωνικής ρίζας ενός 16-bit αριθμού, χρησιμοποιόντας τον αλγόριθμο του σχήματος 4.3.

'Εχουμε λοιπόν: $A = (0.101010010000000)_2 = (\frac{169}{256})_{10}$

0.1010100100000000 [ολίσθηση αριστερά 1 bit]
1010100100000000
1100000000000000 [αφαίρεση]
q_1=1_0110100100000000
1011000000000000 [αφαίρεση]
q_2=1_0001100100000000
1100110000000000 [αφαίρεση]
q_3=0_1110010100000000
0001101100000000 [πρόσθεση]
q_4=1_0000000000000000
1110010011000000 [αφαίρεση]
q_5=0_1110010011000000
0000001101011000 [πρόσθεση]
q_6=0_1110110000011000
0000001101000000 [πρόσθεση]
q_7=0_0111100101011000
0000000110100000 [πρόσθεση]
q_8=0_1111101011111000

Μετά από 8 επαναλήψεις του αλγόριθμου, καταλήγουμε λοιπόν στο εξής αποτέλεσμα: $Q = (0.11010000)_2 = (\frac{13}{16})_{10}$. Παρατηρούμε επίσης ότι ο αριθμός των bits του αποτελέσματος Q (πλην του προσήμου) καθώς και ο αριθμός των αφαιρέσεων και των προσθέσεων που εκτελούνται σε μία επανάληψη του αλγόριθμου, είναι ίσος με τον αριθμό των ζευγών των bits του αριθμού A.

Κεφάλαιο 5

Το κύκλωμα υπολογισμού τετραγωνικής ρίζας

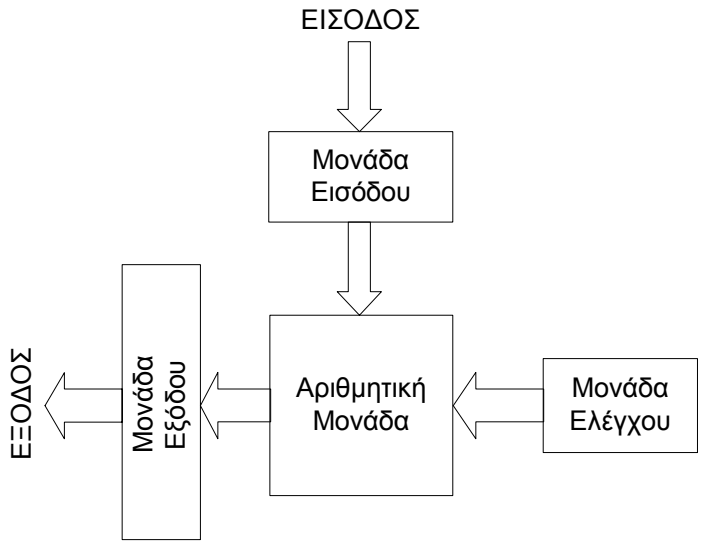
5.1 ΠΡΟΔΙΑΓΡΑΦΕΣ ΤΟΥ ΚΥΚΛΩΜΑΤΟΣ

5.1.1 Γενικά

Μελετώντας προσεκτικά τον αλγόριθμο που χρησιμοποιούμε για την εξαγωγή της τετραγωνικής ρίζας, μπορούμε να καταλήξουμε σε ορισμένα βασικά χαρακτηριστικά και προδιαγραφές που θα πρέπει να πληρεί το κύκλωμα μας. Οι βασικότερες λειτουργίες που ξεχωρίζουν με μια πρώτη ματιά είναι τέσσερις:

1.Η προετοιμασία και είσοδος του αριθμού ο οποίος πρόκειται να επεξεργαστεί καθώς και η είσοδος των ορισμάτων D0, D1,.., D7 που πρόκειται να προστεθούν ή να αφαιρεθούν με τα επιμέρους αποτελέσματα κάθε βαθμίδας της αριθμητικής μονάδας.
2.Η αριθμητική μονάδα που θα αποτελείται από οκτώ όμοιες βαθμίδες οι οποίες θα υλοποιούν τις οκτώ προσθέσεις ή αφαιρέσεις που επιβάλει ο αλγόριθμος.
3.Ο συγχρονισμός και έλεγχος των λειτουργιών του κυκλώματος. Είναι μια θεμελιώδης λειτουργία που απαιτεί πολύ μεγάλη προσοχή στη σχεδίαση. Η λειτουργία αυτή εκτελείται από την μονάδα ελέγχου.

4.Η έξοδος του αποτελέσματος. Η φύση του κυκλώματος επιβάλει τον σχεδιασμό αυτής της βαθμίδας πού σαν βασικό σκοπό έχει τον συγχρονισμό των bits του αποτελέσματος.



Σχήμα 5.1: Συνοπτικό block διάγραμμα του κυκλώματος

Οι μονάδες που περιγράψαμε πιο πάνω φαίνονται στο Σχήμα 5.1 που απεικονίζει το κύκλωμα μας υπό μορφή block διαγράμματος.

5.1.2 Η μονάδα εισόδου

'Οπως προαναφέραμε η μονάδα εισόδου είναι υπεύθυνη για την είσοδο του αριθμού που θα επεξεργαστεί καθώς και για την είσοδο των ορισμάτων D0, D1,.., D7. Μελετώντας τον αλγόριθμο βλέπουμε ότι οι αριθμοί που θα εισαχθούν στο κύκλωμα είναι δεκαεξάμπιτοι ενώ η σειριακή σε επίπεδο ψηφίου αρχιτεκτονική που υιοθετούμε επιβάλει η ροή των δεδομένων στο κύκλωμα να γίνεται σε ψηφία των τεσσάρων bits. Οδηγούμαστε λοιπόν στη χρησιμοποίηση πολυπλεκτών (multiplexers) που θα διαιρούν τις εισόδους των 16 bits του κυκλώματος σε ψηφία των τεσσάρων bits. Χρησιμοποιούμε για αυτό τον σκοπό εννιά πολυπλέκτες 16 σε 4 για την είσοδο του αριθμού προς επεξεργασία και

των D0,D1,..D7.

Μία ακόμη λειτουργία που υπαγορεύει ο αλγόριθμος είναι η ολίσθηση του αριθμού εισόδου κατά ένα bit αριστερά. Αυτή η λειτουργία επιτυγχάνεται μέσω του ολισθητή (shifter), ένα κύκλωμα που σχεδιάσαμε για να εκτελεί αυτή ακριβώς την λειτουργία ειδικά γιά σειριακές σε επίπεδο ψηφίου αρχιτεκτονικές. 'Ολα τα κυκλώματα που αναφέραμε θα περιγραφούν πλήρως σε επόμενη παράγραφο.

5.1.3 Η αριθμητική μονάδα

Η αριθμητική μονάδα είναι η μονάδα όπου εκτελούνται οι αριθμητικές λειτουργίες (προσθέσεις/αφαιρέσεις). 'Οπως φαίνεται στον αλγόριθμο προτού φθάσουμε στο τελικό αποτέλεσμα πρέπει να εκτελεσθούν οκτώ πράξεις, προσθέσεις ή αφαιρέσεις ανάλογα με το carryout της προηγούμενης πράξης. Το carryout κάθε πράξης επιδρά στο κύκλωμα με δύο διαφορετικούς τρόπους. Πρώτον, είναι ένα από τα bits του αποτελέσματος και για αυτό οδηγείται προς την έξοδο, και δεύτερον ελέγχει την πράξη που θα εκτελέσει η επόμενη βαθμίδα. Αν το carryout είναι "0" τότε η επόμενη βαθμίδα θα πρέπει να εκτελέσει πρόσθεση και αντίστροφα αν το carryout είναι "1" τότε η επόμενη βαθμίδα θα πρέπει να εκτελέσει αφαίρεση. Παρουσιάζεται λοιπόν, η ανάγκη ύπαρξης ενός κυκλώματος που θα πρέπει να εκτελεί πρόσθεση ή αφαίρεση ανάλογα με την τιμή του carryout προηγούμενης λειτουργίας. Αυτό το κύκλωμα που είναι και ταυτόχρονα το βασικό κύκλωμα του επεξεργαστή που σχεδιάσαμε, είναι ο σειριακός σε επίπεδο ψηφίου αθροιστής/αφαιρέτης (ADDSUB). Η διάδοση των δεδομένων από βαθμίδα σε βαθμίδα γίνεται μέσω ενός καταχωρητή (REG4x4) ο οποίος επίσης έχει σχεδιαστεί ειδικά για τις ανάγκες της σειριακής σε επίπεδο ψηφίου αρχιτεκτονικής που υιοθετούμε. Ο καταχωρητής αυτός κρατάει ολόκληρο το αποτέλεσμα ενός αθροιστή/αφαιρέτη (ADDSUB) που είναι ένας αριθμός των 16 bits διαιρεμένος σε τέσσερα ψηφία των τεσσάρων bits, και τροφοδοτεί τον επόμενο αθροιστή/αφαιρέτη με το περιεχόμενό την κατάλληλη χρονική στιγμή. Με αυτό τον τρόπο αποθηκεύουμε τα δεδομένα για το χρονικό διάστημα που μεσολαβεί, από την στιγμή που βγαίνουν ως την στιγμή που μπαίνουν από τον ένα αθροιστή/αφαιρέτη στον άλλο.

5.1.4 Η μονάδα ελέγχου

Η μονάδα αυτή έχει σαν σκοπό τον συγχρονισμό των κυκλωμάτων που περιλαμβάνει ο επεξεργαστής καθώς και την παραγωγή των σημάτων που απαιτούν ορισμένα κυκλώματα για να λειτουργήσουν. Μια λειτουργία της μονάδας ελέγχου είναι η οδήγηση του ρολογιού στα flip flops του κυκλωματος μέσω οδηγητών (buffers) για να επιτύχουμε μια ικανοποιητική ταχύτητα. Οδηγητές (buffers) επίσης χρησιμοποιήθηκαν για την οδήγηση του σήματος reset στα flip flops του κυκλώματος. Ο τρόπος που έγινε η οδήγηση των κυκλωμάτων (buffering) αναλύεται διεξοδικά σε επόμενη παράγραφο. Κυκλώματα όπως ο ολισθητής (SHIFTER) και ο αθροιστής/αφαιρέτης (ADDSUB) χρειάζονται ορισμένα σήματα ελέγχου, τα σήματα control1, control2, control3 και control4 που παράγονται απο την γεννήτρια ελέγχου (Con.Gen).

5.1.5 Η μονάδα εξόδου

Κατά τον σχεδιασμό του κυκλώματος θεωρήσαμε ότι τα bits του αποτελέσματος θα πρέπει να παρουσιάζονται στην έξοδο του κύκλωματος ταυτόχρονα. Η φύση του κυκλώματος είναι τέτοια ώστε το πρώτο bit του αποτελέσματος να εξάγεται στον δέκατο παλμό ρολογιού και στη συνέχεια να εξάγεται ένα bit αποτελέσματος κάθε πέντε παλμούς ρολογιού. 'Ετσι λοιπόν σχεδιάσαμε ένα κύκλωμα τον DELAYER που καθυστερεί τα επτά πρώτα bits του αποτελέσματος σε σχέση με το τελευταίο με σκοπό να εμφανίζονται όλα μαζί στην έξοδο.

5.2 ΑΝΑΛΥΤΙΚΗ ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΚΥΚΛΩΜΑΤΩΝ

5.2.1 Το κύκλωμα του αθροιστή/αφαιρέτη (ADDSUB)

Το κύκλωμα αθροιστή/αφαιρέτη δέχεται σαν είσοδο δύο ψηφία των τεσσάρων bits και η έξοδος του είναι αντίστοιχα ένα ψηφίο τεσσάρων bits. Εκτός από τις γραμμές που χρησιμοποιούνται για την είσοδο και έξοδο των δεδομένων, στο κύκλωμα του αθροιστή/αφαιρέτη έχουμε δύο εισόδους και μία έξοδο ακόμη που

χρησιμοποιούνται για τα σήματα ελέγχου. Τα σήματα αυτά είναι τα: carryout, mode και control.Το σήμα carryout είναι το τελικό κρατούμενο της πράξης. Το σήμα mode είναι το σήμα που επιλέγει αν θα γίνει πρόσθεση ή αφαίρεση. Στο σημείο αυτό είναι χρήσιμο να τονίσουμε ορίσμενα σημεία που είναι απαραίτητα για να γίνει κατανοητή η λειτουργία του αθροιστή/αφαιρέτη.

'Οπως έχουμε αναφέρει και προηγουμένως οι διαδοχικές προσθέσεις ή αφαιρέσεις που υπαγορεύει ο αλγόριθμος πραγματοποιούνται από μια διάταξη οκτώ αθροιστών/αφαιρετών. Η φύση μιας πράξης είναι άμεσα συνδεδεμένη από το τελικό κρατούμενο της προηγούμενης. Ως παράδειγμα αναφέρουμε ότι αν η πρώτη πράξη έχει τελικό κρατούμενο "0" τότε η δεύτερη πράξη θα είναι πρόσθεση. Το γεγονός αυτό στον σχεδιασμό του κυκλώματος μας μεταφράζεται με απευθείας σύνδεση της εξόδου carryout του πρώτου αθροιστή/αφαιρέτη με την είσοδο mode του δεύτερου αθροιστή/αφαιρέτη και την έξοδο carryout του δεύτερου αθροιστή/αφαιρέτη με την είσοδο mode του τρίτου κ.ο.κ. Επειδή μια ολοκληρωμένη πράξη εκτελείται από τον αθροιστή/αφαιρέτη σε τέσσερις διαδοχικούς παλμούς ρολογιού (όσο χρειάζεται ώστε να πραγματοποιηθεί η είσοδος των τεσσάρων ψηφίων του αριθμού) είναι φανερό ότι η είσοδος mode πρέπει να διατηρείται σταθερή στο διάστημα αυτών των τεσσάρων παλμών ρολογιού. Το σήμα control ελέγχει την έξοδο carryout και την αρχική τιμή που θα πάρει το carryin του πρώτου πλήρους αθροιστού (F.A) κατά την έναρξη της πράξης. 'Εχουμε επίσης δύο εισόδους για το σήμα του ρολογιού και το reset που συνδέονται στα flip flops του αθροιστή/αφαιρέτη.'Οπως φαίνεται και στο σχήμα (βλέπε τμήμα 5.4), το κύκλωμα του αθροιστή/ αφαιρέτη αποτελείται από μια διάταξη τεσσάρων F.As μεταξύ των οποίων διαδίδεται το κρατούμενο (carry). Δηλαδή ο αθροιστής ειναι διάδοσης κρατουμένου (ripple carry). Στις εξόδους των F.As υπάρχει ένα D flip flop για να αποθηκεύεται το αποτέλεσμα. Πριν την είσοδο του ενός ορίσματος στους F.As παρεμβάλεται μια πύλη XOR, που σαν σκοπό έχει την συμπλήρωση του αριθμού στην περίπτωση της αφαίρεσης. Η μία είσοδος της κάθε XOR πύλης είναι συνδεμένη στην είσοδο mode. Αν η τιμή του mode είναι "0" τότε τα bits του αριθμού περνούν αναλοίωτα μέσα από τις πύλες XOR και γίνεται η πράξη της πρόσθεσης. Αντίθετα αν η τιμή του mode είναι "1" τότε τα bits του αριθμού καθώς περνούν από τις πύλες XOR συμπληρώνονται και γίνεται η πράξη της αφαίρεσης.

Στο κύκλωμα υπάρχουν επίσης δύο ακόμα D flip flops και δύο πολυπλέκτες 2 σε 1. Ο πολυπλέκτης που η εξοδός του είναι συνδεδεμένη στο carryin του πρώτου F.A έχει σαν σκοπό να "καθαρίζει" το carryin κατά την έναρξη της πράξης. Αν η πράξη είναι πρόσθεση τότε μέσω του πολυπλέκτη περνάει "0", ενώ αν είναι αφαίρεση περνάει "1". Οι τιμές αυτές συμπίπτουν με τις τιμές του mode για κάθε περίπτωση και γι'αυτό το λόγο η είσοδος mode είναι συνδεμένη στην μία είσοδο του πολυπλέκτη. Η επιλογή ενός εκ των δύο σημάτων του πολυπλέκτη γίνεται μέσω του σήματος ελέγχου control. Αφού κατά την έναρξη "καθαριστεί" το carryin, κατά την επεξεργασία των τριών ψηφίων που θα ακολουθήσουν ο πολυπλέκτης περνάει στην είσοδο carryin του πρώτου F.A το carryout που προέρχεται από την επεξεργασία του προηγούμενου ψηφίου. Το carryout περνάει μέσα από ένα D flip flop για λόγους συγχρονισμού.

Ο πολυπλέκτης που η εξοδός του είναι συνδεδεμένη στο D flip flop εξόδου του carryout έχει σαν σκοπό να επιτρέπει μόνο στο τελικό carryout να εμφανίζεται στην έξοδο. Αυτό το carryout είναι ένα από τα bits του αποτελέσματος που θα συνδεθεί επίσης στην είσοδο mode του επόμενου αθροιστή/αφαιρέτη για να ελέγξει την επόμενη πράξη. Ο πολυπλέκτης αυτός δέχεται την έξοδό του σαν ανατροφοδότηση μέσα από ένα D flip flop. Αυτή η σύνδεση έχει σαν αποτέλεσμα η τιμή του τελικού carryout και κατά συνέπεια η τιμή του mode του επόμενου αθροιστή/αφαιρέτη να διατηρείται σταθερή για χρονική διάρκεια ίση με τέσσερους παλμούς ρολογιού. 'Ετσι εξασφαλίζουμε την σωστή λειτουργία και επικοινωνία των αθροιστών/αφαιρετών του κυκλώματος. Ο αθροιστής/αφαιρέτης έχει latency τέσσερις παλμούς ρολογιού για τα αποτελέσματα της πράξης και πέντε παλμούς ρολογιού για το τελικό carryout. Το κύκλωμα του αθροιστή/ αφαιρέτη (ADDSUB) όπως το σχεδιάσαμε στον υπολογιστή καθώς και τα αποτελέσματα της εξομοίωσης που εκθέτουν καθαρά όλα τα σήματα του κυκλώματος, φαίνονται στο τμήμα 5.4. Επίσης αναφέρουμε ότι το κύκλωμα περιλαμβάνει τριάντα επτά βασικά στοιχεία (instances).

5.2.2 Το κύκλωμα του ολισθητή (SHIFTER)

Βασικός σκοπός του κυκλώματος είναι να ολισθαίνει έναν αριθμό των 16 bits

κατά ένα bit αριστερά. Η ιδιαιτερότητα του κυκλώματος έγκειται στο γεγονός ότι λειτουργεί σειριακάσε επίπεδο ψηφίου (digit serial), δηλαδή η πλήρης ολίσθηση του αριθμού ολοκληρώνεται σε τέσσερις παλμούς ρολογιού οπότε χρειάζεται και κάποιο κατάλληλο σήμα ελέγχου. Το κύκλωμα του ολισθητή (SHIFTER) αποτελείται βασικά από μια 4χ4 διάταξη D flip flops. Το κύκλωμα αυτό δέχεται σαν είσοδο έναν αριθμό των 16 bits σε ψηφία των τεσσάρων bits και του προκαλεί τις παρακάτω αλλαγές:

1. Το MSB του αριθμού έχει χαθεί.
2. Τα υπόλοιπα δεκαπέντε bits του αριθμού έχουν μετατοπισθεί κατά μία θέση αριστερά.
3. Το λιγότερο σημαντικό bit (LSB) του αριθμού γίνεται 0.

Υπάρχουν ακόμη τα σήματα clk και reset που συνδέονται στα flip flops του κυκλώματος και το σήμα control. Για να εξηγήσουμε καλύτερα την λειτουργία του κυκλώματος θεωρούμε ότι στην είσοδο του κυκλώματος έχουμε το λιγότερο σημαντικό ψηφίο ενός αριθμού. Στον πρώτο παλμό το λιγότερο σημαντικό ψηφίο (LSD) θα βρίσκεται στις εξόδους της πρώτης τετράδας των flip flops, και στον τρίτο παλμό θα βρίσκεται στίς εξόδους της τρίτης τετράδας των flip flops. Πριν έρθει η ανερχόμενη παρυφή του τέταρτου παλμού γίνεται η ολίσθηση (shifting). Ο πολυπλέκτης που στην είσοδο του select έχει συνδεθεί το σήμα control έχει έξοδο "0" και έτσι γίνεται η προσθήκη του "0" στην θέση του LSB. 'Οπως παρατηρούμε στο κύκλωμα (βλέπε τμήμα 5.4), οι έξοδοι των flip flops της τρίτης τετράδας δεν συνδέονται με τα αντίστοιχα flip flops της τέταρτης τετράδας. Με αυτό τον τρόπο πετυχαίνεται η ολίσθηση του αριθμού. Το bit που θέλουμε να τοποθετήσουμε στο επόμενο ψηφίο, αφού το περνάμε μέσα από ένα D flip flop για να συγχρονισθεί με το επόμενο ψηφίο, το θέτουμε στην είσοδο του πολυπλέκτη που τώρα όμως λόγω της αλλαγής της στάθμης του σήματος control βγάζει στην έξοδό του το bit για το οποίο γίνεται λόγος. Για να λειτουργήσει σύμφωνα με τον επιθυμητό τρόπο ο πολυπλέκτης απαιτεί σαν σήμα ελέγχου το control που φαίνεται στην εξομοίωση του κυκλώματος που βρίσκεται στο τμήμα 5.4 μαζί με το πλήρες σχέδιο του κυκλώματος του ολισθητή όπως σχεδιάστηκε στον υπολογιστή. Το κύκλωμα του ολισθητή (SHIFTER) περιλαμβάνει 24 βασικά στοιχεία (instances).

5.2.3 Το κύκλωμα του καταχωρητή (REG4x4)

Ο σκοπός του κυκλώματος του καταχωρητή είναι να εξασφαλίζει την ασφαλή και σωστή διάδοση των δεδομένων από βαθμίδα σε βαθμίδα. Επειδή η διάδοση των αριθμών γίνεται σε μορφή τεσσάρων ψηφίων των τεσσάρων bits, οδηγηθήκαμε στην μορφή του καταχωρητή που φαίνεται στο αντίστοιχο σχήμα που βρίσκεται στο τμήμα 5.4. Ο καταχωρητής αποτελείται από μια 4χ4 διάταξη D flip flops πετυχαίνοντας έτσι τον διαχωρισμό των ψηφίων (digits). Η μορφή αυτή του καταχωρητή μας δίνει την δυνατότητα να μιλάμε για αρχιτεκτονική τύπου αγωγού (pipelined architecture). 'Εχει λοιπόν την δυνατότητα ο επεξεργαστής να δεχθεί νέα είσοδο τέσσερις παλμούς μετά την τελευταία είσοδο. Επίσης στο κύκλωμα του καταχωρητή παρατηρούμε ότι υπάρχουν ακόμη οι είσοδοι του ρολογιού και του reset για τα flip flops του κυκλώματος. Ο καταχωρητής (REG4x4) περιλαμβάνει 21 βασικά στοιχεία (instances).

5.2.4 Οι πολυπλέκτες (multiplexers) εισόδου 16 σε 4 (MUX16:4)

Σκοπός των πολυπλεκτών 16 σε 4 (MUX 16:4) είναι η μετατροπή των εισόδων των 16 bits σε ψηφία των τεσσάρων bits. Τέτοιοι πολυπλέκτες χρησιμοποιήθηκαν στην είσοδο του αριθμού που επεξεργάζεται το κύκλωμα και στην είσοδο των ορισμάτων D0, D1,.., D7. Το κύκλωμα του πολυπλέκτη 16 σε 4 αποτελείται από μία παράλληλη σύνδεση τεσσάρων πολυπλεκτών 4 σε 1. Ο τρόπος με τον οποίο τροφοδοτείται ο αριθμός στις εισόδους των πολυπλεκτών 4 σε 1 σε συνδιασμό με την σύνδεση των σημάτων επιλογής κάνουν το κύκλωμα να λειτουργεί κατά τον επιθυμητό τρόπο. Πιο συγκεκριμένα τα σήματα επιλογής των πολυπλεκτών 4 σε 1 παράγονται από ένα σύγχρονο μετρητή τεσσάρων καταστάσεων. Καθώς ο μετρητής αλλάζει κατάσταση με κάθε παλμό ρολογιού, στην έξοδο του πολυπλέκτη εμφανίζεται και ένα διαφορετικό ψηφίο του αριθμού των 16 bits. Η διαδικασία αυτή επαναλαμβάνεται για τέσσερις παλμούς ρολογιού (όσες και οι καταστάσεις του μετρητή) και στον πέμπτο παλμό έχουμε στην έξοδο του πολυπλέκτη το ίδιο ψηφίο που είχαμε στην έξοδο του στον πρώτο παλμό. Μπορεί να γίνει εύκολα αντιληπτό ότι η αρχική κατάσταση του μετρητή παίζει πρωταρχικό ρόλο στον συγχρονισμό. 'Επειτα από μελέτη καταλήξαμε στις αρχικές καταστάσεις των μετρητών των πολυπλεκτών 16 σε 4 ώστε να

επιτύχουμε συγχρονισμό στο κύκλωμα. Οι αρχικές καταστάσεις των μετρητών επιτυγχάνονται με ενεργοποίηση των set ή reset εισόδων των flip flops των μετρητών. Οι είσοδοι set και reset των μετρητών είναι είσοδοι στο συνολικό κύκλωμα και αναφέρονται με τις εξής ονομασίες: SET0A, RST0A, SET1A, RST1A, SET0B, RST0B, SET1B, RST1B, SET0C, RST0C, SET1C, RST1C, SET0D, RST0D, SET1D, RST1D. Στις τιμές που θα πρέπει να δοθούν στις εισόδους αυτές θα αναφερθούμε στην παράγραφο όπου θα σχολιασθεί το συνολικό κύκλωμα. Στο τμήμα 5.4 βρίσκεται το κύκλωμα του πολυπλέκτη 16 σε 4 (MUX16:4) καθώς και η εξομοίωση του. Ο πολυπλέκτης 16 σε 4 (MUX 16:4) περιλαμβάνει οκτώ βασικά στοιχεία (instances).

5.2.5 Το κύκλωμα του καταχωρητή (REG)

Ο καταχωρητής REG είναι ένας απλός παράλληλος καταχωρητής τεσσάρων flip flops που χρησιμοποιείται στις εξόδους των πολυπλεκτών 16 σε 4 που χρησιμοποιούμε για την είσοδο των ορισμάτων D0, D1,.., D7. Χρησιμοποιήθηκε για να έχουμε μεγαλύτερη ασφάλεια στην έξοδο των πολυπλεκτών.

5.2.6 Το κύκλωμα της γεννήτριας σημάτων ελέγχου (CON.GEN)

Το κύκλωμα αυτό παράγει τα σήματα ελέγχου που χρησιμοποιούμε για τον έλεγχο λειτουργίας των αθροιστών/αφαιρετών (ADDSUB) και του ολισθητή (SHIFTER). Τα σήματα αυτά είναι τα σήματα που αναφέρθηκαν στις παραγράφους 5.2.1 και 5.2.2 ως σήματα control. Κάθε σήμα ελέγχου είναι ένα περιοδικό σήμα με περίοδο τεσσάρων παλμών που διατηρείται στην στάθμη "0" για χρονικό διάστημα τριών παλμών και ανεβαίνει σε στάθμη "1" για χρονικό διάστημα ενός παλμού ρολογιού. Η συμμετρία του κυκλώματος του επεξεργαστή μας επιτρέπει να χρησιμοποιούμε την ίδια μορφή σήματος ελέγχου στις διάφορες βαθμίδες. 'Ομως το γεγονός ότι η απόκριση του carryout από το κύκλωμα του αθροιστή/αφαιρέτη χρειάζεται πέντε παλμούς ρολογιού μας αναγκάζει να χρησιμοποιήσουμε και τις τέσσερις παραλλαγές του σήματος που φαίνονται στην εξομοίωση της γεννήτριας σημάτων ελέγχου (τμήμα 5.4). Η

απλότητα του σήματος ελέγχου που απαιτείται κάνει τον σχεδιασμό της γεννήτριας ελέγχου απλό.

'Οπως φαίνεται και στο κύκλωμα της γεννήτριας ελέγχου (CON.GEN) που υπάρχει στο τμήμα 5.4 το ρόλο της γεννήτριας παίζει ένας μετρητής δακτυλίου. Εδώ είναι χρήσιμο να κάνουμε μιά αναφορά στην λειτουργία του μετρητή δακτυλίου που χρησιμοποιήσαμε. 'Ενας μετρητής δακτυλίου των δύο bits έχει ένα bit άσσου που το κυκλοφορεί ανάμεσα στα flip flops, κι έτσι παράγει δύο διαφορετικές καταστάσεις. Μπορούμε να διπλασιάσουμε τον αριθμό τών δυνατών καταστάσεων, εάν συνδέσουμε τον καταχωρητή ολίσθησης σαν μετρητή δακτυλίου με "αντιστροφή ουράς" ("switch tail"). Αυτός είναι ένας κυκλικός καταχωρητής ολίσθησης, όπου όμως συνδέουμε την συμπληρωματική έξοδο του τελευταίου flip flop στην είσοδο του πρώτου. Τα περιεχόμενα του καταχωρητή ολισθαίνουν κατά μία θέση δεξιά με τον κάθε παλμό ρολογιού και συγχρόνως το συμπλήρωμα της τιμής του δεύτερου flip flop μεταφέρεται στο πρώτο. Αρχίζοντας από την κατάσταση μηδενισμού ο μετρητής δακτυλίου με αντιστροφή ουράς περνάει από τις καταστάσεις 00, 10, 11, 01. Ξεκινώντας με όλα τα bits να είναι 0, η κάθε ολίσθηση εισάγει και από έναν άσσο από τα αριστερά, έως ότου ο καταχωρητής γεμίσει με άσσους. Μετά αρχίζουν να εισάγονται μηδενικά από τα αριστερά, έως ότου ο καταχωρητής γεμίσει ξανά με μηδενικά. Η γεννήτρια των σημάτων ελέγχου του κυκλώματος αποτελείται από έναν μετρητή δακτυλίου με αντιστροφή ουράς των δύο bits και από τέσσερις πύλες αποκωδικοποίησης που δίνουν σαν εξόδους τα τέσσερα σήματα ελέγχου που χρειαζόμαστε. Αφού η κάθε πύλη ενεργοποιείται από μία μόνο κατάσταση, οι έξοδοι των πυλών δημιουργούν τέσσερις συνεχόμενους παλμούς χρονισμού. Αυτό φαίνεται στην εξομοίωση της γεννήτριας ελέγχου, η οποία δίδεται στο τμήμα 5.4. Η γεννήτρια ελέγχου περιλαμβάνει επτά βασικά στοιχεία (instances). Το αναλυτικό κύκλωμα όπως σχεδιάστηκε στον υπολογιστή βρίσκεται επίσης στο τμήμα 5.4.

5.2.7 Το κύκλωμα καθυστέρησης (DELAYER)

Αν μελετήσουμε το κύκλωμα του επεξεργαστή από την πλευρά των χρονισμών

παρατηρούμε ότι το πρώτο bit του αποτελέσματος βγαίνει στην ανερχόμενη παρυφή του δέκατου παλμού, το δεύτερο στον δέκατο πέμπτο και κατά την ίδια λογική ένα bit αποτελέσματος εξάγεται ανά πέντε παλμούς ρολογιού. Στο στάδιο της σχεδίασης με την σκέψη ότι ο επεξεργαστής πρόκειται να εκτελέσει επεξεργασία τύπου αγωγού (pipelined processing), θεωρήσαμε απαραίτητο ένα κύκλωμα που να συγχρονίζει τα bit του αποτελέσματος στην έξοδο. Αυτό πρέπει να γίνει επειδή την στιγμή που παράγει η τελευταία βαθμίδα του επεξεργαστή το όγδοο bit του αποτελέσματος τα προηγούμενα bits του αποτελέσματος θα έχουν επικαλυφθεί από τα bits αποτελέσματος του επόμενου αριθμού που εισέρχεται στον επεξεργαστή. Δίνοντας ένα παράδειγμα το γεγονός αυτό θα γίνει πιο κατανοητό. Την χρονικη στιγμή "0" εισάγουμε προς επεξεργασία τον αριθμό Α. Το πρώτο bit αποτελέσματος του αριθμού Α αναμένεται στον δέκατο παλμό και το όγδοο στον τεσσερακοστό. Την χρονική στιγμή "4" εισάγουμε προς επεξεργασία τον αριθμό Β. Το πρώτο bit αποτελέσματος του αριθμού Β το αναμένουμε στον δέκατο τέταρτο παλμό και το όγδοο στον τεσσερακοστό τέταρτο. Παρατηρείται λοιπόν μια επικάλυψη των bits του αποτελέσματος ενός αριθμού από τον επόμενο. Η πρόληψη αυτής της ανεπιθύμητης επικάλυψης αποτελεσμάτων που οδηγεί τον επεξεργαστή σε λανθασμένη λειτουργία, προλαμβάνεται με το σχεδιασμό του κυκλώματος του DELAYER. Το κύκλωμα αυτό αποτελείται από επτά σειρές καθυστέρησης από D flip flops μία για κάθε ένα από τα πρώτα επτά bits του αποτελέσματος. 'Ετσι λοιπόν πετυχαίνουμε μία καθυστέρηση τριάντα πέντε παλμών για το πρώτο bit αποτελέσματος, τριάντα για το δεύτερο, είκοσι πέντε για το τρίτο, καταλήγοντας στο έβδομο bit αποτελέσματος το οποίο καθυστερείται πέντε παλμούς ρολογιού. Πέρα από τον συγχρονισμό των bits του αποτελέσματος στην έξοδο, το κύκλωμα του DELAYER χρησιμοποιείται για την προσάρτηση των απαραίτητων τελικών carryouts που προέρχονται από τους αθροιστές/αφαιρέτες, στα ορίσματα D0, D1,..., D7. Επιλέγοντας τις κατάλληλες εξόδους από τις αλυσίδες των flip flops του DELAYER πέρνουμε τα carryout σήματα ακριβώς τις χρονικές στιγμές που τα χρειαζόμαστε. Το κύκλωμα του DELAYER έχει εισόδους τα carryout σήματα από τους επτά πρώτους αθροιστές/αφαιρέτες. Οι έξοδοι του DELAYER είναι τα carryout σήματα που αφού καθυστερηθούν κατάλληλα συνδέονται στον δίαυλο εξόδου (output bus) του επεξεργαστή. Επιπλέον έχουμε είκοσι επτά εξόδους του DELAYER που είναι τα σήματα carryout των αθροιστών/αφαιρετών που συνδέο-

νται στους πολυπλέκτες είσοδου των ορισμάτων D0, D1,...,D7. Το πλήρες κύκλωμα του DELAYER όπως σχεδιάστηκε στον υπολογιστή βρίσκεται στο τμήμα 5.4. Ο DELAYER περιλαμβάνει εκατόν εξήντα επτά βασικά στοιχεία (instances).

5.2.8 Το κύκλωμα παραγωγής λογικών σταθμών (CONSTANT)

Μελετώντας τον αλγόριθμο βλέπουμε πως οι αριθμοί D0, D1, .., D7, απαιτούν για τον σχηματισμό τους ορισμένα bits που έχουν σταθερή τιμή "1" ή "0". Για την εξυπηρέτηση αυτής της ανάγκης χρησιμοποιήσαμε έναν αριθμό στοιχείων παραγωγής λογικών σταθμών κάτω από την ονομασία CONSTANT. 'Ετσι οι είσοδοι των πολυπλεκτών 16 σε 4 που πρέπει να είναι σε σταθερές λογικές στάθμες τροφοδοτούνται από ένα τέτοιο κύκλωμα παραγωγής λογικών σταθμών. Το κύκλωμα παραγωγής λογικών σταθμών (CONSTANT) βρίσκεται στο τμήμα 5.4 και περιλαμβάνει δεκαέξι βασικά στοιχεία (instances).

5.2.9 Τα κυκλώματα οδήγησης

Ο μεγάλος αριθμός των flip flops που χρησιμοποιούμε επιβάλλει την χρησιμοποίηση ειδικών στοιχείων για την οδήγηση των σημάτων του ρολογιού και reset στα flip flops. Για αυτό το σκοπό χρησιμοποιήθηκε το κύκλωμα οδήγησης της βιβλιοθήκης BUF39. Το κύκλωμα αυτό έχει fanout 11pF. 'Εχοντας υπόψιν ότι η χωρητικότητα εισόδου των flip flops για την είσοδο clk είναι 0.08pF και για την είσοδο reset είναι 0.14pF καταλήξαμε σε μια στρατηγική οδήγησης που να μας εξασφαλίζει την γρήγορη λειτουργία του κυκλώματος. Η οδήγηση για το σήμα του ρολογιού έχει μορφή δένδρου με τέσσερις βαθμίδες διακλαδώσεων.Ο BUF39 της πρώτης βαθμίδας έχει τρείς διακλαδώσεις, κάθε BUF39 της δεύτερης βαθμίδας έχει δύο διακλαδώσεις κάθε BUF39 της τρίτης βαθμίδας έχει επτά διακλαδώσεις και κάθε BUF39 τέταρτης βαθμίδας οδηγεί οκτώ flip flops. Η οδήγηση για το σήμα reset έχει μορφή δένδρου με τρεις βαθμίδες. Ο BUF39 της πρώτης βαθμίδας έχει τρείς διακλαδώσεις, κάθε BUF39 της δεύτερης βαθμίδας έχει έξι διακλαδώσεις και κάθε BUF39 της τρίτης βαθμίδας οδηγεί δεκαεννιά flip flops.

5.3 ΤΟ ΣΥΝΟΛΙΚΟ ΚΥΚΛΩΜΑ

5.3.1 Περιγραφή του συνολικού κυκλώματος

Μετά την λεπτομερή περιγραφή όλων των επιμέρους κυκλωμάτων που συνθέτουν τον επεξεργαστή, δεν μένει παρά η παρουσίαση του συνολικού κυκλώματος. Το σχηματικό του επεξεργαστή όπως σχεδιάστηκε στον υπολογιστή βρίσκεται στο τμήμα 5.4 με την ονομασία SQ.ROOT. Με μια πρώτη ματιά φαίνεται η διάταξη των οκτώ αθροιστών/αφαιρετών (ADDSUB) που αποτελούν τις οκτώ βαθμίδες της αριθμητικής μονάδας του επεξεργαστή. Η διάδοση των δεδομένων από βαθμίδα σε βαθμίδα γίνεται μέσω καταχωρητή (REG4x4). Παρατηρούμε τον τρόπο με τον οποίο έχουν συνδεθεί η έξοδος carryout του κάθε αθροιστή/αφαιρέτη με την είσοδο mode του επόμενου αθροιστή/αφαιρέτη. Με αυτόν τον τρόπο επιδρά το τελικό κρατούμενο μιάς πράξης στην επόμενη. Οι έξοδοι carryout των αθροιστών/αφαιρετών είναι συνδεμένοι στον DELAYER. Οι συνδέσεις αυτές δεν δηλώνονται με γραμμές στο σχέδιο, όπως και πολλές άλλες γιά λόγους ευκρίνειας του σχεδίου. Το σχεδιαστικό εργαλείο του υπολογιστή μας έδωσε την δυνατότητα να πραγματοποιούμε συνδέσεις χωρίς γραμμές. (Μπορούμε να ενώσουμε δύο ή περισσότερα σημεία δηλώνοντας τα στο πρόγραμμα με το ίδιο property name, βλέπε παράρτημα.) Ο DELAYER διανέμει τα σήματα carryouts σε όποιο σημείο του κυκλώματος απαιτούνται και επιπλέον τροφοδοτεί και τον δίαυλο εξόδου.

Μπορούμε επίσης να διακρίνουμε τους πολυπλέκτες 16 σε 4 (MUX 16:4) που χρησιμοποιούνται για την είσοδο και μετατροπή των εισόδων. Στην αριστερή πλευρά του σχεδίου βλέπουμε τον δίαυλο εισόδου I(15:0) να συνδέεται σε έναν πολυπλέκτη 16 σε 4. Η έξοδος αυτού του πολυπλέκτη συνδέεται με τον ολισθητή (SHIFTER) για να ολισθηθεί ο αριθμός εισόδου κατά ένα bit αριστερά. Επίσης μπορούμε να παρατηρήσουμε άλλους οκτώ πολυπλέκτες 16 σε 4 που εισάγουν στον επεξεργαστή τα ορίσματα D0, D1, D2, D3, D4, D5, D6, D7. Στην έξοδο των πολυπλεκτών υπάρχουν συνδεδεμένοι παράλληλοι καταχωρητές τεσσάρων flip flops (REG). Βλέπουμε επίσης τα κυκλώματα παραγωγής λογικών σταθμών (CONSTANT) που τροφοδοτούν τους πολυπλέκτες με τα σταθερά μηδενικά ή τους σταθερούς άσσους. Οι δεκαέξι είσοδοι SET0A,

RSTOA, SET1A, RST1A, SET0B, RST0B, SET1B, RST1B, SET0C, RST0C, SET1C, RST1C, SET0D, RST0D, SET1D, RST1D, έχουν τοποθετηθεί για να δίνουμε τις αρχικές τιμές στους μετρητές των πολυπλεκτών 16 σε 4 ώστε να πετυχαίνουμε συγχρονισμό των D0, D1, .., D7, με τον αριθμό πρός επεξεργασία (A).

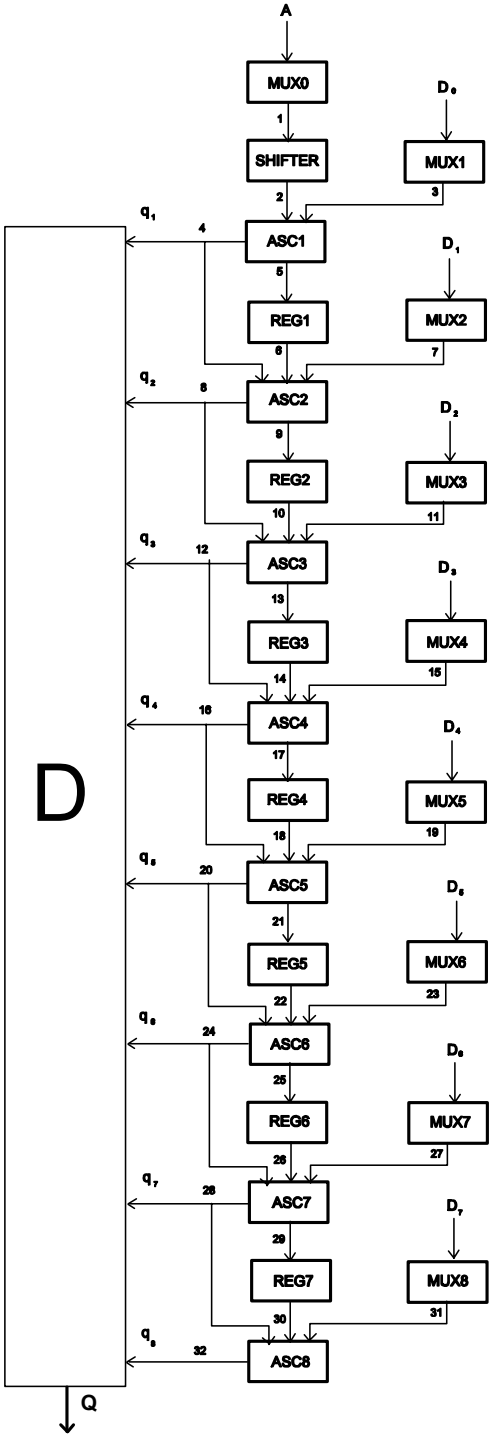
Οι είσοδοι των σημάτων ελέγχου των αθροιστών/αφαιρετών (ADDSUB) και του ολισθητή (SHIFTER) είναι συνδεδεμένες στις εξόδους της γεννήτριας σημάτων ελέγχου (CON.GEN)Ανάλογα με πιο από τα τέσσερα σήματα της γεννήτριας απαιτείται σε κάθε περίπτωση οι έξοδοι της γεννήτριας συνδέονται στις εισόδους control των αθροιστών/αφαιρετών και του shifter. Στο σχέδιο πραγματοποιήσαμε τις συνδέσεις αυτές με γραμμές ώστε ο αναγνώστης να μπορεί να διακρίνει πιο σήμα ελέγχου απαιτεί κάθε κύκλωμα. Τέλος ο δίαυλος εξόδου αποτελείται από εννιά bits, οκτώ που είναι τα τελικά carryout σήματα των αθροιστών/αφαιρετών και ένα "0" που είναι το sign bit του αποτελέσματος και δηλώνει ότι είναι θετικό.

Στην εξομοίωση του κυκλώματος (βλέπε τμήμα 5.4) δώσαμε τέσσερις αριθμούς επαληθεύοντας έτσι και την σωστή λειτουργία του επεξεργαστή αλλά και την δυνατότητα pipelined λειτουργίας. Σαν εισόδους δώσαμε τους αριθμούς 5480, 2000, 0800 και 4800 σε δεκαεξαδικό αριθμητικό σύστημα στον πρώτο, πέμπτο, έννατο και δέκατο τρίτο παλμό αντίστοιχα. Ο επεξεργαστής αποκρίθηκε στον τεσσερακοστό πέμπτο, τεσσερακοστό έννατο, πεντηκοστό τρίτο και πεντηκοστό έβδομο παλμό με τα σωστά αποτελέσματα 0D0, 080, 040 και 0C0 αντίστοιχα. Η εξομοίωση έγινε με περίοδο ρολογιού 12nsec. Το κύκλωμα του επεξεργαστή έχει latency σαράντα πέντε παλμούς ρολογιού και throughput τέσσερις παλμούς ρολογιού. Περιλαμβάνει συνολικά εννιακόσια τρία βασικά στοιχεία (instances).

5.3.2 Θεωρητικός έλεγχος λειτουργίας του κυκλώματος

Το αντικείμενο της παραγράφου είναι ο θεωρητικός έλεγχος της συμπεριφοράς του κυκλώματος. Συγκεκριμένα θα εισάγουμε έναν δεκαεξάμπιτο αριθμό στο κύκλωμα και θα υπολογίσουμε τι αναμένουμε να εμφανισθεί σε κάθε κόμβο του κυκλώματος σε μία δεδομένη χρονική στιγμή. Αυτή η μελέτη πιστεύουμε ότι είναι πολύ χρήσιμη στην καλύτερη κατανόηση της λειτουργίας του κυκλώματος, ενώ

παράλληλα θα είναι μία βοηθητική προεργασία για την φάση της εξομοίωσης με την βοήθεια υπολογιστή.



Σχήμα 5.2: Block διάγραμμα του κυκλώματος

Το κύκλωμα απεικονίζεται σε μορφή μπλοκ διαγράμματος στο σχήμα 5.2. Για λόγους απλότητας στο block διάγραμμα στις εξόδους των multiplexers 16 σε 4 έχουμε παραλείψει τους καταχωρητές. 'Ετσι η έξοδος του multiplexer και η αντίστοιχη είσοδος του αθροιστή/αφαιρέτη περιγράφονται από τον ίδιο κόμβο. Κάθε κόμβος που μελετάται έχει αριθμηθεί, ενώ οι είσοδοι που χρειάζονται σε διάφορα στοιχεία του κυκλώματος έχουν σημειωθεί. Επίσης θεωρούμε ότι η μονάδα ελέγχου μας παρέχει τον απαραίτητο χρονισμό και τα κατάλληλα σήματα για την σωστή λειτουργία του κυκλώματος.

Ο αριθμός που δίνεται σαν είσοδος στο κύκλωμα είναι το κλάσμα A=169/256 που σε δυαδική μορφή είναι A=0101010010000000. Το αναμενόμενο αποτέλεσμα είναι ο αριθμός Q=13/16 ή Q=011010000. Η τετραγωνική ρίζα του αριθμού αυτού είχε υπολογιστεί και στο παράδειγμα του κεφαλαίου 4 για τον αλγόριθμο του σχήματος 4.3. Σε αυτό το σημείο είναι σκόπιμο να θυμηθούμε τους αριθμούς που προέκυψαν κατά την εφαρμογή του αλγορίθμου. Αριθμός εισόδου, A :0101010010000000
Αριθμός A μετά τον shifter, A0:1010100100000000
Αποτέλεσμα πρώτης πράξης, A1:0110100100000000
Αποτέλεσμα δεύτερης πράξης, A2:0001100100000000
Αποτέλεσμα τρίτης πράξης, A3:1110010100000000
Αποτέλεσμα τέταρτης πράξης, A4:0000000000000000
Αποτέλεσμα πέμπτης πράξης, A5:1111001011000000
Αποτέλεσμα έκτης πράξης, A6:1111100101110000
Αποτέλεσμα έβδομης πράξης, A7:1111110010111100
Αποτέλεσμα όγδοης πράξης, A8:1111111001011111
'Ορισμα πρώτο, DO:0100000000000000
'Ορισμα δεύτερο, D1:0101000000000000
'Ορισμα τρίτο, D2:0011010000000000
'Ορισμα τέταρτο, D3:0001101100000000
'Ορισμα πέμπτο, D4:0000110101000000
'Ορισμα έκτο, D5:0000011010110000
'Ορισμα έβδομο, D6:0000001101001100
'Ορισμα όγδοο, D7:0000000110100011
Πρώτο τελικό κρατούμενο, Q1:1

Δεύτερο τελικό κρατούμενο, Q2:1
Τρίτο τελικό κρατούμενο, Q3:0
Τέταρτο τελικό κρατούμενο, Q4:1
Πέμπτο τελικό κρατούμενο, Q5:0
'Εκτο τελικό κραρούμενο, Q6:0
'Εβδομο τελικό κρατούμενο, Q7:0
'Ογδοο τελικό κρατούμενο, Q8:0

Η είσοδος του αριθμού Α γίνεται μέσω του multiplexer 16 σε 4, MUX0. Αν αφιερώσουμε τον πρώτο παλμό στην λειτουργία reset του κυκλώματος τότε μπορούμε να πούμε ότι ο αριθμός εμφανίζεται σε τέσσερεις διαδοχικούς παλμούς στον κόμβο 1, σε digits των τεσσάρων bits.Δηλαδή στον κόμβο 1 έχουμε στον δεύτερο παλμό:0000, στον τρίτο παλμό:1000, στον τέταρτο παλμό:0100 και στον πέμπτο παλμό:0101. Ο κόμβος 2 είναι η έξοδος του SHIFTER άρα θα πάρουμε τον αριθμό ολισθημένο κατά ένα bit αριστερά. Ο SHIFTER έχει latency ίση με 4. 'Αρα στον κόμβο 2 έχουμε στον πέμπτο παλμό:0000, στον έκτο παλμό:0000, στον έβδομο παλμό:1001 και στον όγδοο παλμό:1010. Ο κόμβος 3 είναι η έξοδος του multiplexer 16 σε 4, MUX1. Μέσω αυτού του multiplexer γίνεται η είσοδος του D0=0100000000000000, που θα αφαιρεθεί από τον A0 (δηλαδη τον ολισθημένο Α). Ο D0 πρέπει να εισαχθεί στον αθροιστή/αφαιρέτη ADDSUB1 ταυτόχρονα με τον A0. Αυτό επιτυγχάνεται με το κατάλληλο σήμα ελέγχου του multiplexer MUX1. 'Ετσι λοιπόν στον κόμβο 3 έχουμε στον πέμπτο παλμό:0000, στον έκτο παλμό:0000, στον έβδομο παλμό:0000 και στον όγδοο παλμό:0100. Ο κόμβος 4 είναι η έξοδος carryout του ADDSUB1. Επειδη όπως γνωρίζουμε η έξοδος carryout του ADDSUB βγαίνει πέντε παλμούς μετά την εισαγωγή των δεδομένων, στον κόμβο 4 στον δέκατο παλμό θα έχουμε έξοδο Q1:1. Ο κόμβος 5 είναι η έξοδος του ADDSUB1. Σε αυτόν τον κόμβο αναμένουμε το αποτέλεσμα A1. Στον έκτο παλμό θα έχουμε 0000, στον έβδομο 0000, στον όγδοο 1001 και στον έννατο 0110. Ο κόμβος 6 είναι η έξοδος του καταχωρητή REG1. Σε αυτό τον κόμβο ανμένουμε να παρουσιαστεί στον δέκατο παλμό 0000, στον ενδέκατο 0000, στον δωδέκατο 1001 και στον δέκατο τρίτο 0110. Ο κόμβος 7 είναι είναι η έξοδος του multiplexer MUX2. Σε αυτόν τον κόμβο εμφανίζεται ο αριθμός D1. Στον δέκατο παλμό αναμένουμε 0000, στον ενδέκατο 0000, στον δωδέκατο 0000 και στον δέκατο

τρίτο 0101. Ο κόμβος 8 είναι η έξοδος του ADDSUB2 όπου παρουσιάζεται ο αριθμός A2. Στον κόμβο αυτό αναμένουμε στον ενδέκατο παλμό 0000, στον δωδέκατο 0000, στον δέκατο τρίτο 1001 και στον δέκατο τέταρτο 0001. Ο κόμβος 9 είναι η έξοδος carryout του ADDSUB2 όπου θα εμφανιστεί στον δέκατο πέμπτο παλμό το τελικό carryout που θα είναι Q2=1. Ο κόμβος 10 είναι η έξοδος του καταχωρητή REG2. Σε αυτό τον κόμβο αναμένουμε να παρουσιαστεί στον δέκατο πέμπτο παλμό 0000, στον δέκατο έκτο 0000, στον δέκατο έβδομο 1001 και στον δέκατο όγδοο 0001. Ο κόμβος 11 είναι η έξοδος του multiplexer 16 σε 4 MUX3. Στον κόμβο αυτό εμφανίζεται ο αριθμός D2. Στον δέκατο πέμπτο παλμό έχουμε 0000, στον δέκατο έκτο 0000, στον δέκατο έβδομο 0100 και στον δέκατο όγδοο 0011. Ο κόμβος 12 είναι η έξοδος του ADDSUB3 όπου θα εμφανιστεί ο αριθμός A3. Στον δέκατο έκτο παλμό έχουμε 0000, στον δέκατο έβδομο 0000, στον δέκατο όγδοο 0101 και στον δέκατο έννατο 1110. Ο κόμβος 13 είναι η έξοδος carryout του ADDSUB3 όπου θα εμφανιστεί στον εικοστό παλμό το τελικό carryout που θα είναι Q3=0. Ο κόμβος 14 είναι η έξοδος του καταχωρητή REG3. Σε αυτό τον κόμβο αναμένουμε να παρουσιαστεί στον εικοστό παλμό 0000, στον εικοστό πρώτο 0000, στον εικοστό δεύτερο 0101 και στον εικοστό τρίτο 1110. Ο κόμβος 15 είναι η έξοδος του multiplexer 16 σε 4 MUX4. Στον κόμβο αυτό εμφανίζεται ο αριθμός D3. Στον εικοστό παλμό έχουμε 0000, στον εικοστό πρώτο 0000, στον εικοστό δεύτερο 1011 και στον εικοστό τρίτο 0001. Ο κόμβος 16 είναι η έξοδος carryout του ADDSUB4 όπου θα εμφανιστεί στον εικοστό πέμπτο παλμό το τελικό carryout που θα είναι Q4=1. Ο κόμβος 17 είναι η έξοδος του ADDSUB4 όπου θα εμφανιστεί ο αριθμός A4. Στον εικοστό πρώτο παλμό έχουμε 0000, στον εικοστό δεύτερο 0000, στον εικοστό τρίτο 0000 και στον εικοστό τέταρτο 0000. Ο κόμβος 18 είναι η έξοδος του καταχωρητή REG4. Σε αυτό τον κόμβο αναμένουμε να παρουσιαστεί στον εικοστό πέμπτο παλμό 0000, στον εικοστό έκτο 0000, στον εικοστό έβδομο 0000 και στον εικοστό όγδοο 0000. Ο κόμβος 19 είναι η έξοδος του multiplexer 16 σε 4 MUX5. Στον κόμβο αυτό εμφανίζεται ο αριθμός D4. Στον εικοστό πέμπτο παλμό έχουμε 0000, στον εικοστό έκτο 0100, στον εικοστό έβδομο 1101 και στον εικοστό όγδοο 0000. Ο κόμβος 20 είναι η έξοδος carryout του ADDSUB5 όπου θα εμφανιστεί στον τριακοστό παλμό το τελικό carryout που θα είναι Q5=0. Ο κόμβος 21 είναι η έξοδος του ADDSUB5 όπου θα εμφανιστεί ο αριθμός A5. Στον εικοστό έκτο παλμό έχουμε 0000, στον εικοστό έβδομο 1100,

στον εικοστό όγδοο 0010 και στον εικοστό έννατο 1111. Ο κόμβος 22 είναι η έξοδος του καταχωρητή REG5. Σε αυτό τον κόμβο αναμένουμε να παρουσιαστεί στον τριακοστό παλμό 0000, στον τριακοστό πρώτο 1100, στον τριακοστό δεύτερο 0010 και στον τριακοστό τρίτο 1111. Ο κόμβος 23 είναι η έξοδος του multiplexer 16 σε 4 MUX6. Στον κόμβο αυτό εμφανίζεται ο αριθμός D5. Στον τριακοστό παλμό έχουμε 0000, στον τριακοστό πρώτο 1011, στον τριακοστό δεύτερο 0110 και στον τριακοστό τρίτο 0000. Ο κόμβος 24 είναι η έξοδος carryout του ADDSUB6 όπου θα εμφανιστεί στον τριακοστό παλμό το τελικό carryout που θα είναι Q6=0. Ο κόμβος 25 είναι η έξοδος του ADDSUB6 όπου θα εμφανιστεί ο αριθμός A6. Στον τριακοστό πρώτο παλμό έχουμε 0000, στον τριακοστό δεύτερο 0111, στον τριακοστό τρίτο 1001 και στον τριακοστό τέταρτο 1111. Ο κόμβος 26 είναι η έξοδος του καταχωρητή REG6. Σε αυτό τον κόμβο αναμένουμε να παρουσιαστεί στον τριακοστό πέμπτο παλμό 0000, στον τριακοστό έκτο 0111, στον τριακοστό έβδομο 1001 και στον τριακοστό όγδοο 1111. Ο κόμβος 27 είναι η έξοδος του multiplexer 16 σε 4 MUX7. Στον κόμβο αυτό εμφανίζεται ο αριθμός D6. Στον τριακοστό πέμπτο παλμό έχουμε 1100, στον τριακοστό έκτο 0100, στον τριακοστό έβδομο 0011 και στον τριακοστό όγδοο 0000. Ο κόμβος 28 είναι η έξοδος carryout του ADDSUB7 όπου θα εμφανιστεί στον τεσσαρακοστό παλμό το τελικό carryout που θα είναι Q7=0. Ο κόμβος 29 είναι η έξοδος του ADDSUB7 όπου θα εμφανιστεί ο αριθμός A7. Στον τριακοστό έκτο παλμό έχουμε 1100, στον τριακοστό έβδομο 1011, στον τριακοστό όγδοο 1100 και στον τριακοστό έννατο 1111.Ο κόμβος 30 είναι η έξοδος του καταχωρητή REG7. Σε αυτό τον κόμβο αναμένουμε να παρουσιαστεί στον τεσσαροκοστό παλμό 1100, στον τεσσαρακοστό πρώτο 1011, στον τεσσαρακοστό δεύτερο 1100 και στον τεσσαρακοστό τρίτο 1111. Ο κόμβος 31 είναι η έξοδος του multiplexer 16 σε 4 MUX8. Στον κόμβο αυτό εμφανίζεται ο αριθμός D7. Στον τεσσαρακοστό παλμό έχουμε 0011, στον τεσσαρακοστό πρώτο 1010, στον τεσσαρακοστό δεύτερο 0001 και στον τεσσαρακοστό τρίτο 0000. Ο κόμβος 32 είναι η έξοδος carryout του ADDSUB8 όπου βγαίνει το στον τεσσαρακοστό πέμπτο παλμό το τελικό carryout που θα είναι Q8=0.

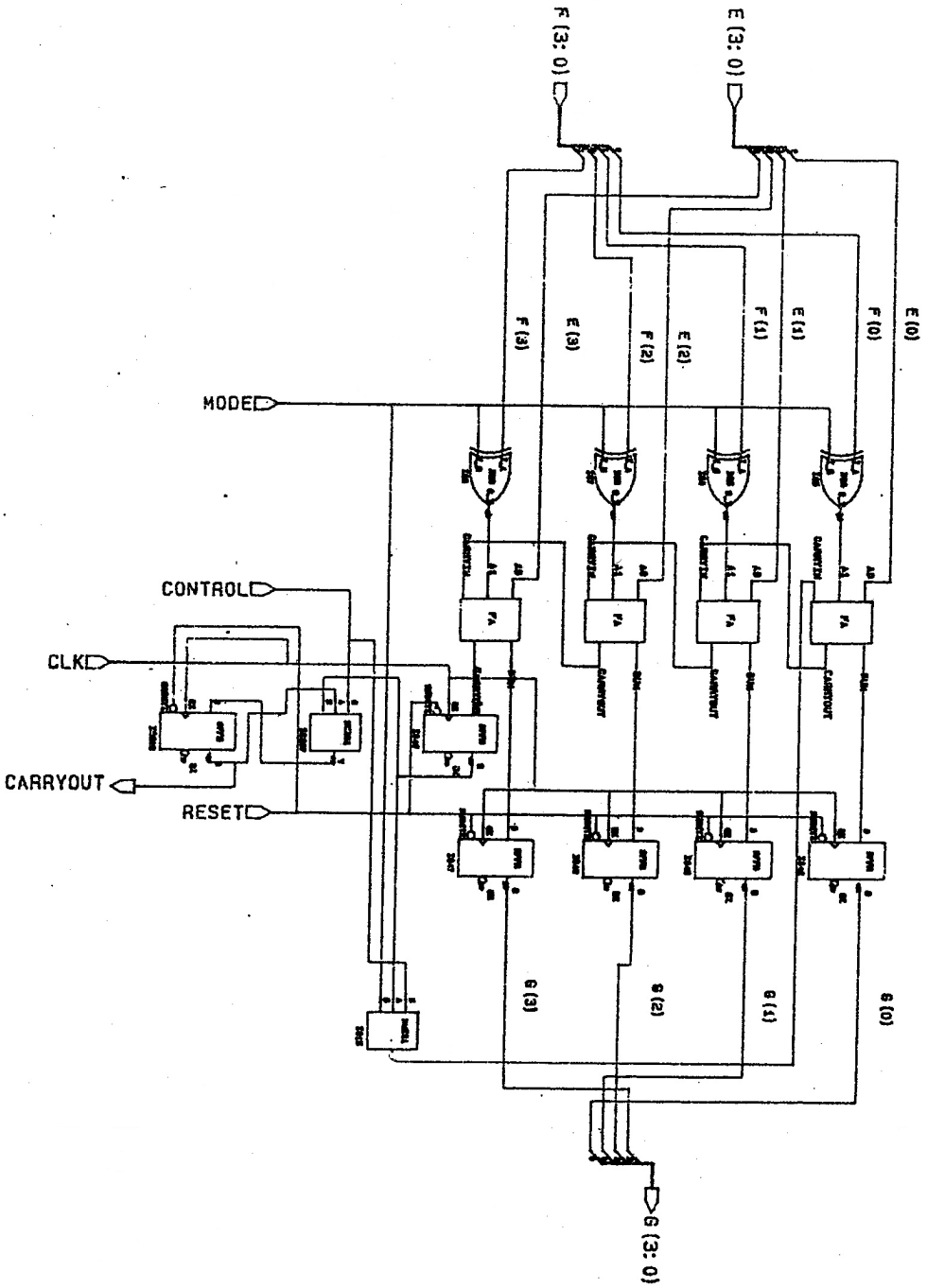
Μετά από αυτό το παράδειγμα κι αφού εξαντλήσαμε κάθε περιθώριο θεωρητικής μελέτης όσον αφορά την λειτουργία του κυκλώματος, περάσαμε στην εξομοίωση στον υπολογιστή όπου τελικά είδαμε τις προσπάθειές μας να τερματίζονται με

επιτυχία.

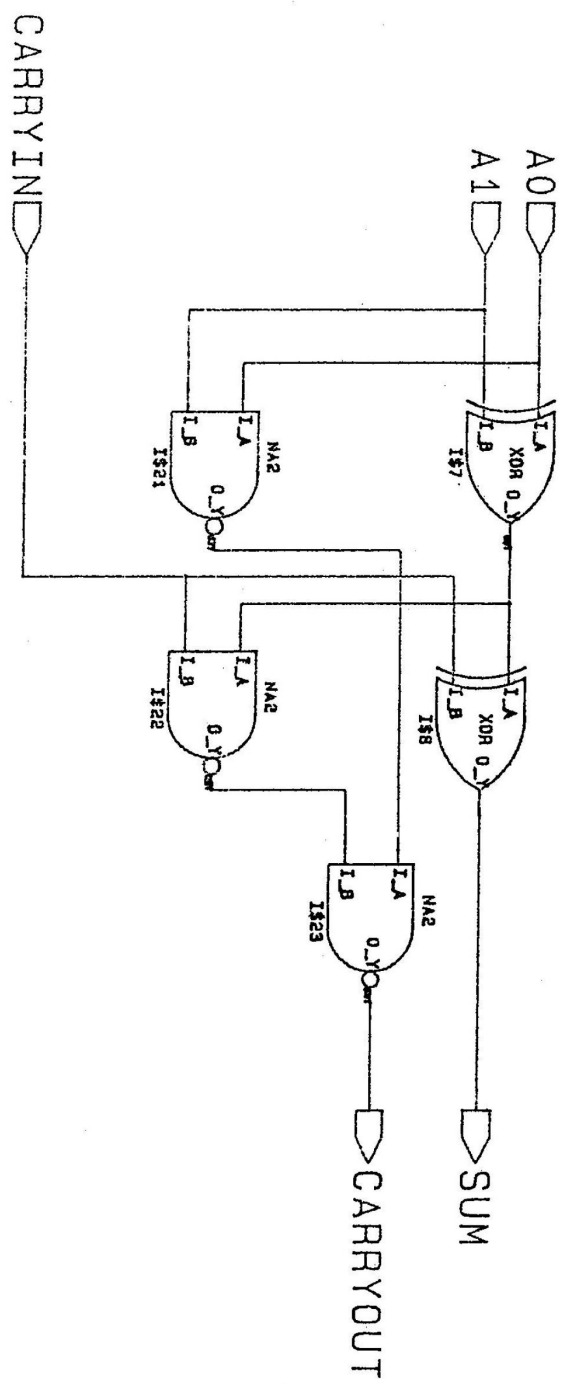
5.4 ΤΑ ΣΧΕΔΙΑ ΤΩΝ ΚΥΚΛΩΜΑΤΩΝ ΚΑΙ ΟΙ ΕΞΟΜΟΙΩΣΕΙΣ

Σε αυτή την παράγραφο βρίσκονται όλα τα σχέδια των κυκλωμάτων όπως έγιναν με την βοήθεια του σχεδιαστικού εργαλείου στον υπολογιστή. Επίσης μετά από κάθε κύκλωμα υπάρχει και η εκτύπωση της εξομοίωσης του, για όποιο κύκλωμα χρειάστηκε να γίνει εξομοίωση. Η σειρά με την οποία δίνονται τα σχέδια των κυκλωμάτων είναι η ίδια σειρά με την οποία αναφέρθηκαν στην ενότητα 5.2.

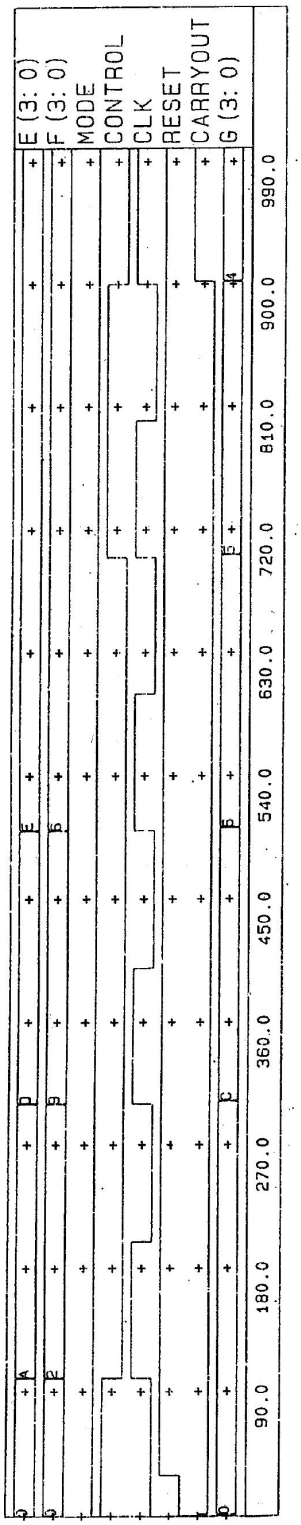
Σχέδιο του κυκλώματος αθροιστή/αφαιρέτη (ADDSUB)



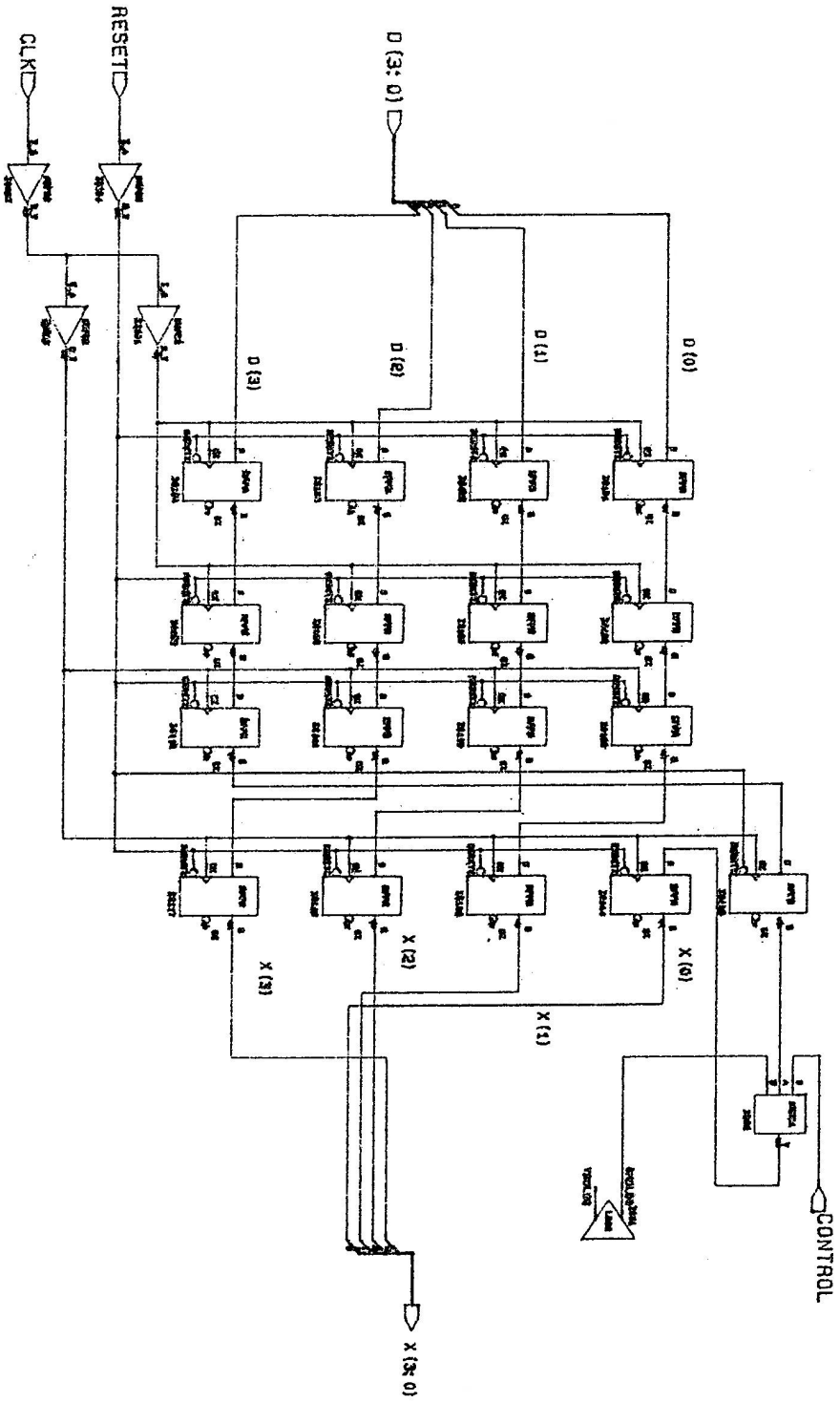
Σχέδιο του κυκλώματος πλήρους αθροιστή



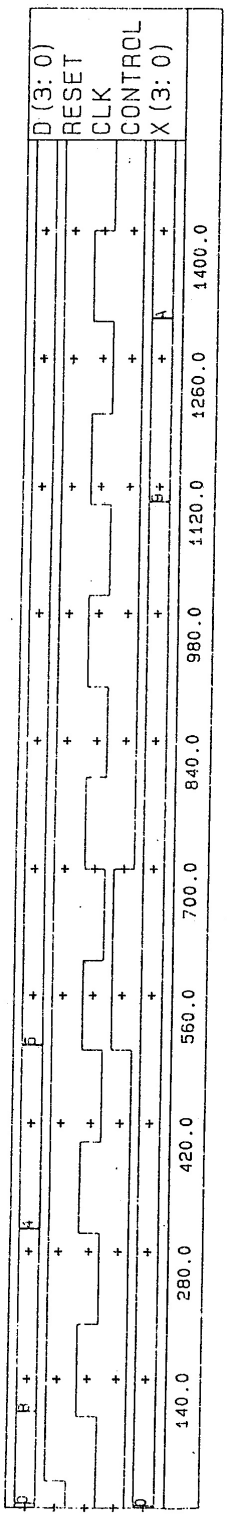
Εξομοίωση του κυκλώματος αθροιστή/αφαιρέτη (ADDSUB)



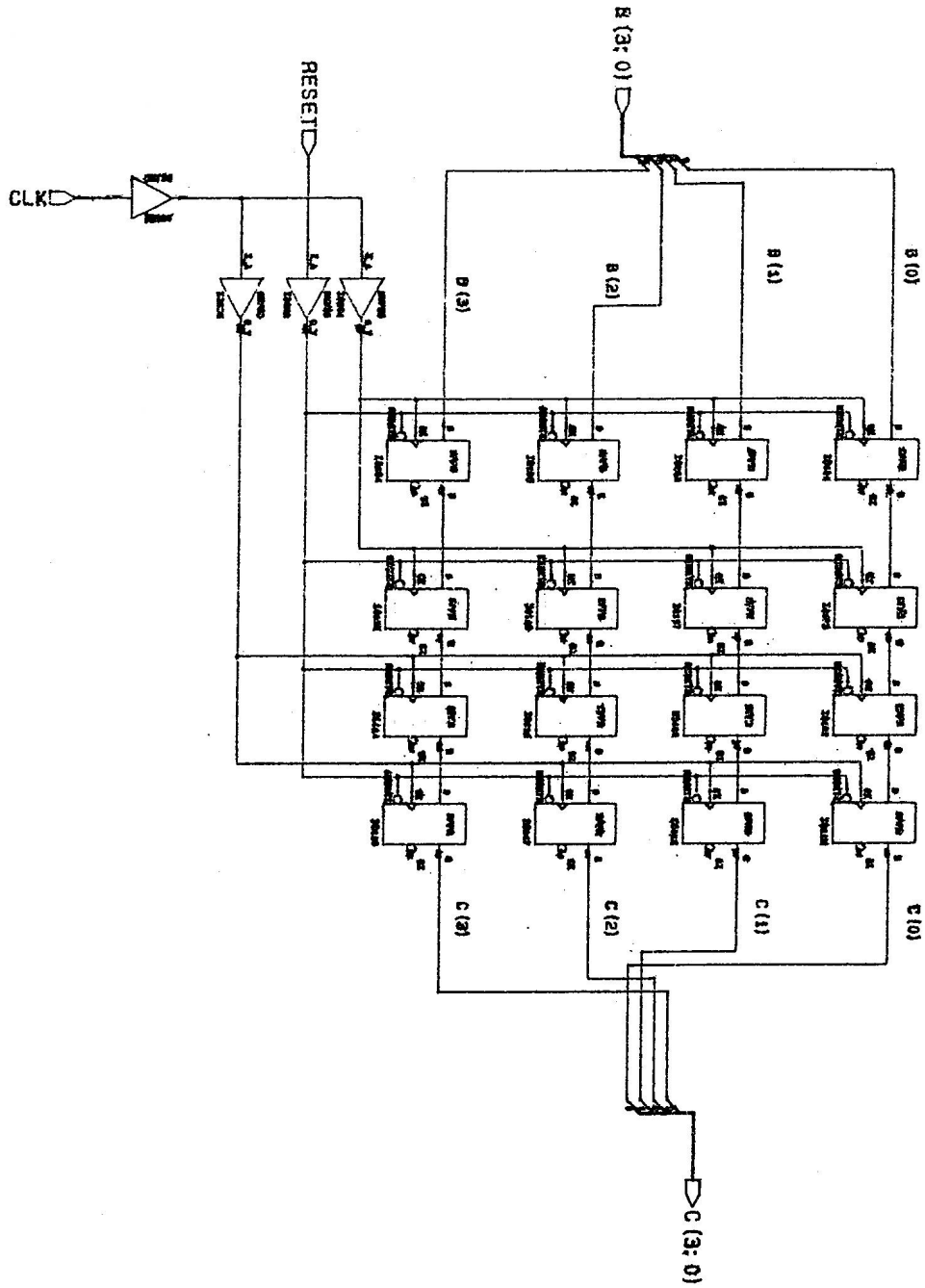
Σχέδιο του κυκλώματος ολισθητή (shifter)



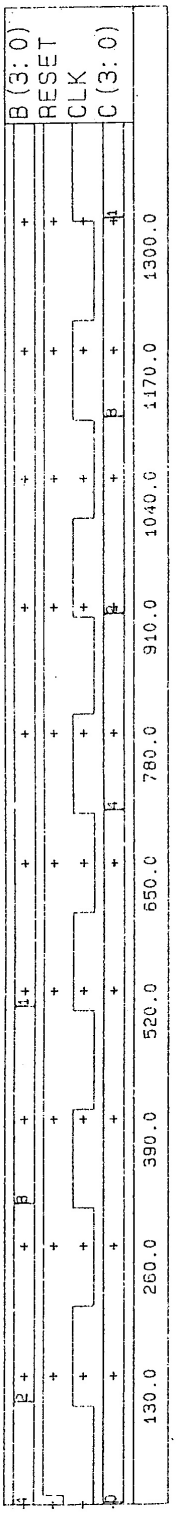
Εξομοίωση του κυκλώματος ολισθητή (shifter)



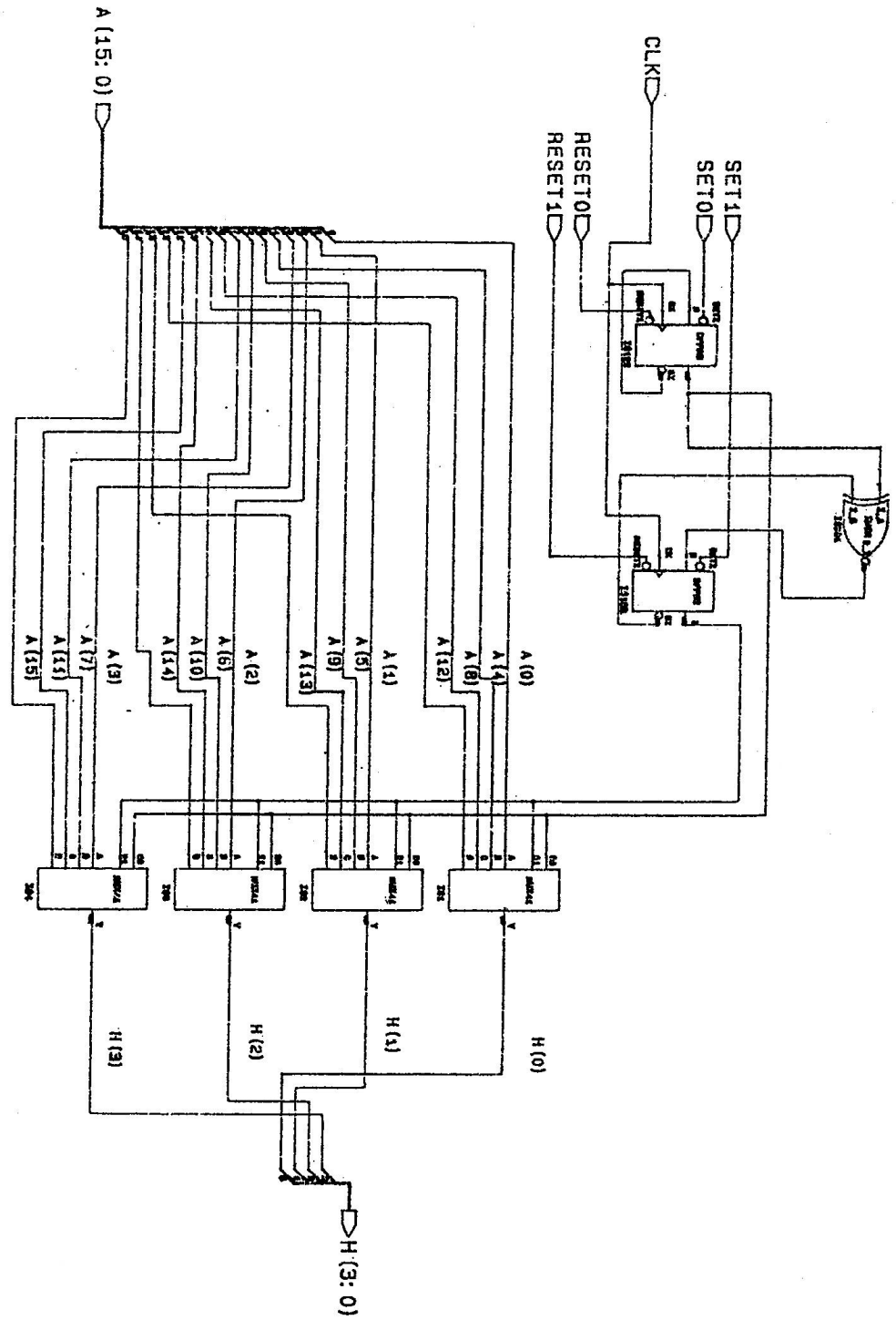
Σχέδιο του κυκλώματος καταχωρητή (REG4x4)



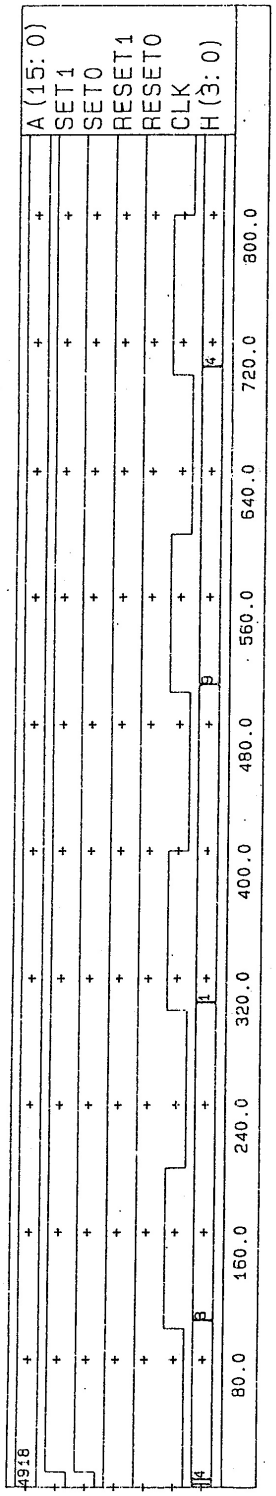
Εξομοίωση του κυκλώματος καταχωρητή (REG4x4)



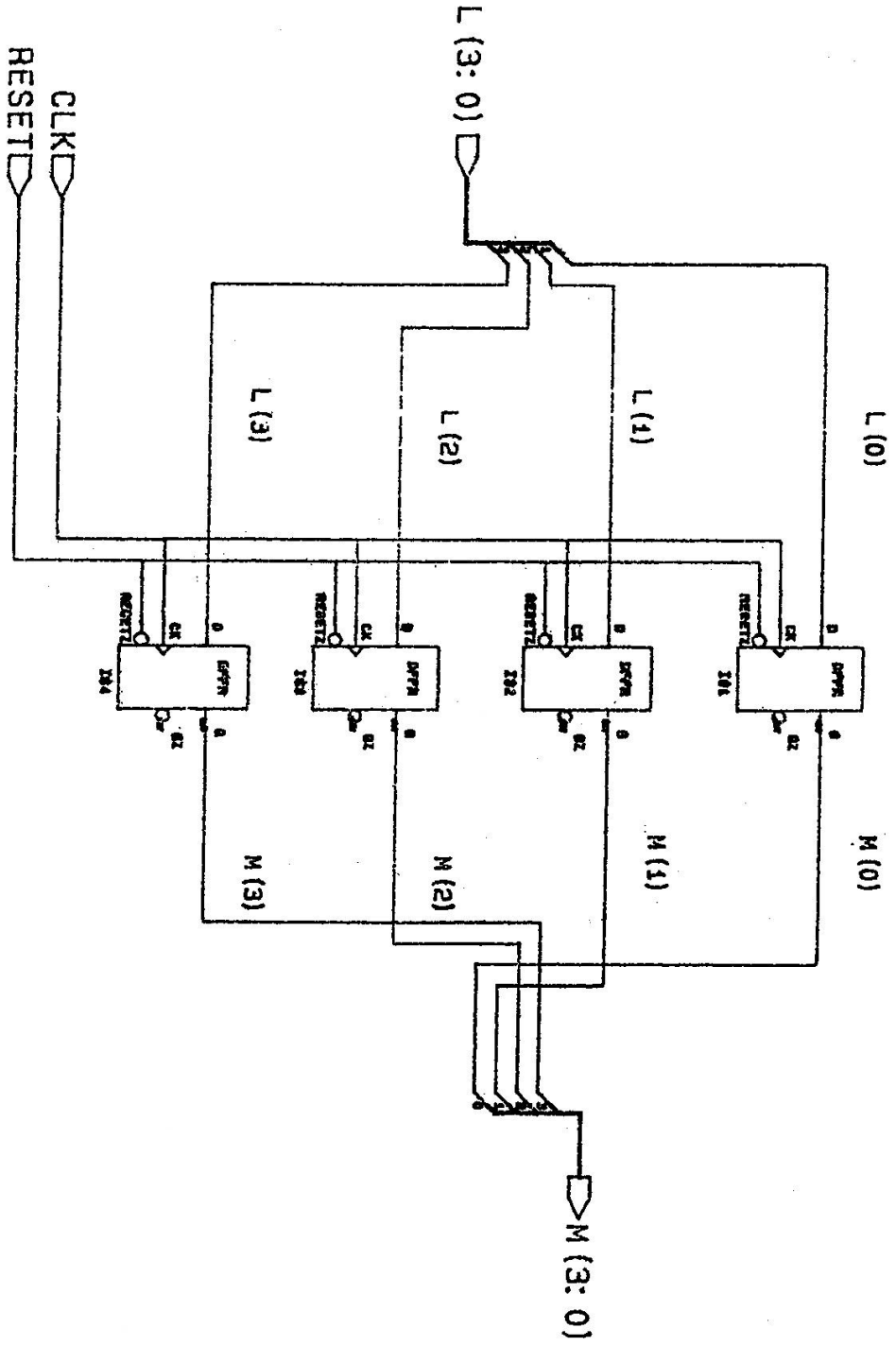
Σχέδιο του κυκλώματος πολυπλέκτη (MUX16:4)



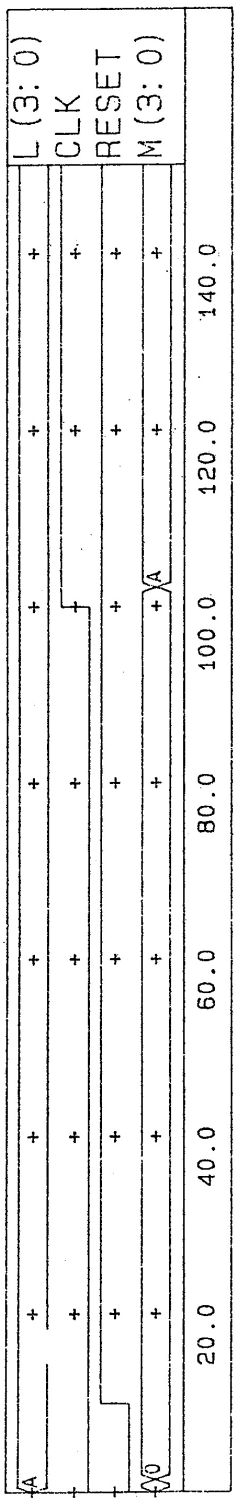
Εξομοίωση του κυκλώματος πολυπλέκτη (MUX16:4)



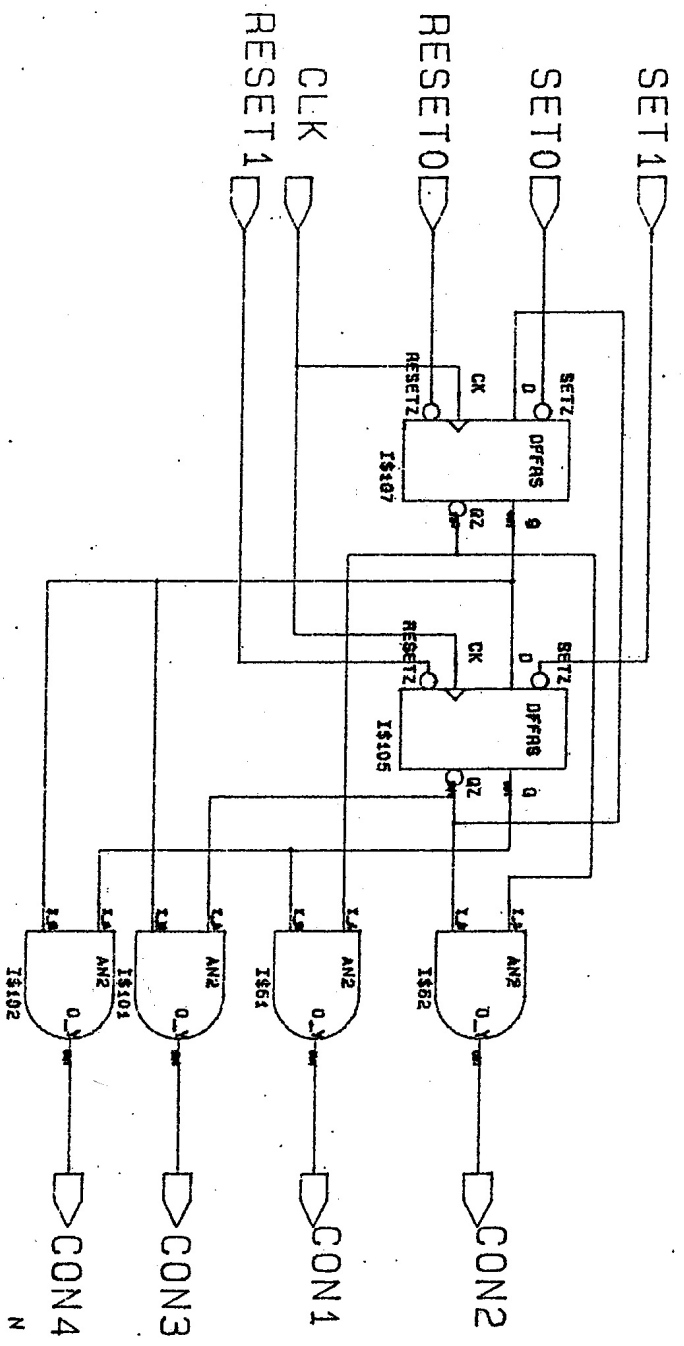
Σχέδιο του κυκλώματος καταχωρητή (REG)



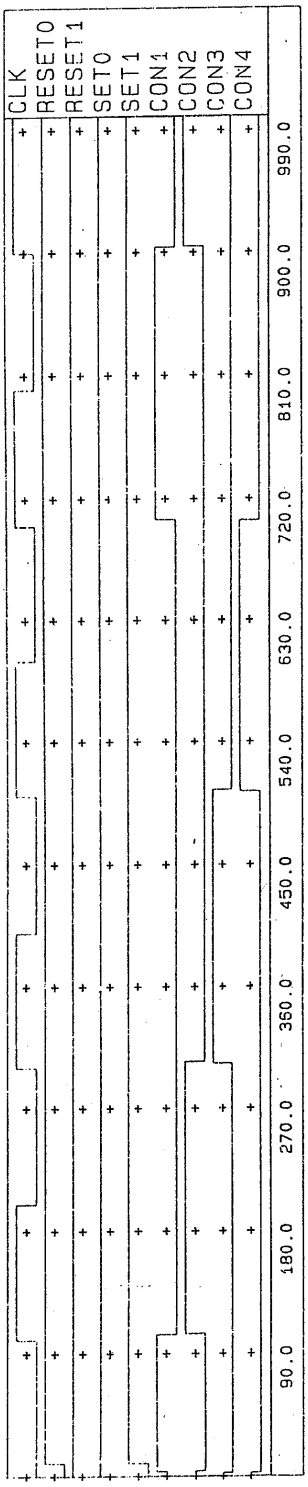
Εξομοίωση του κυκλώματος καταχωρητή (REG)



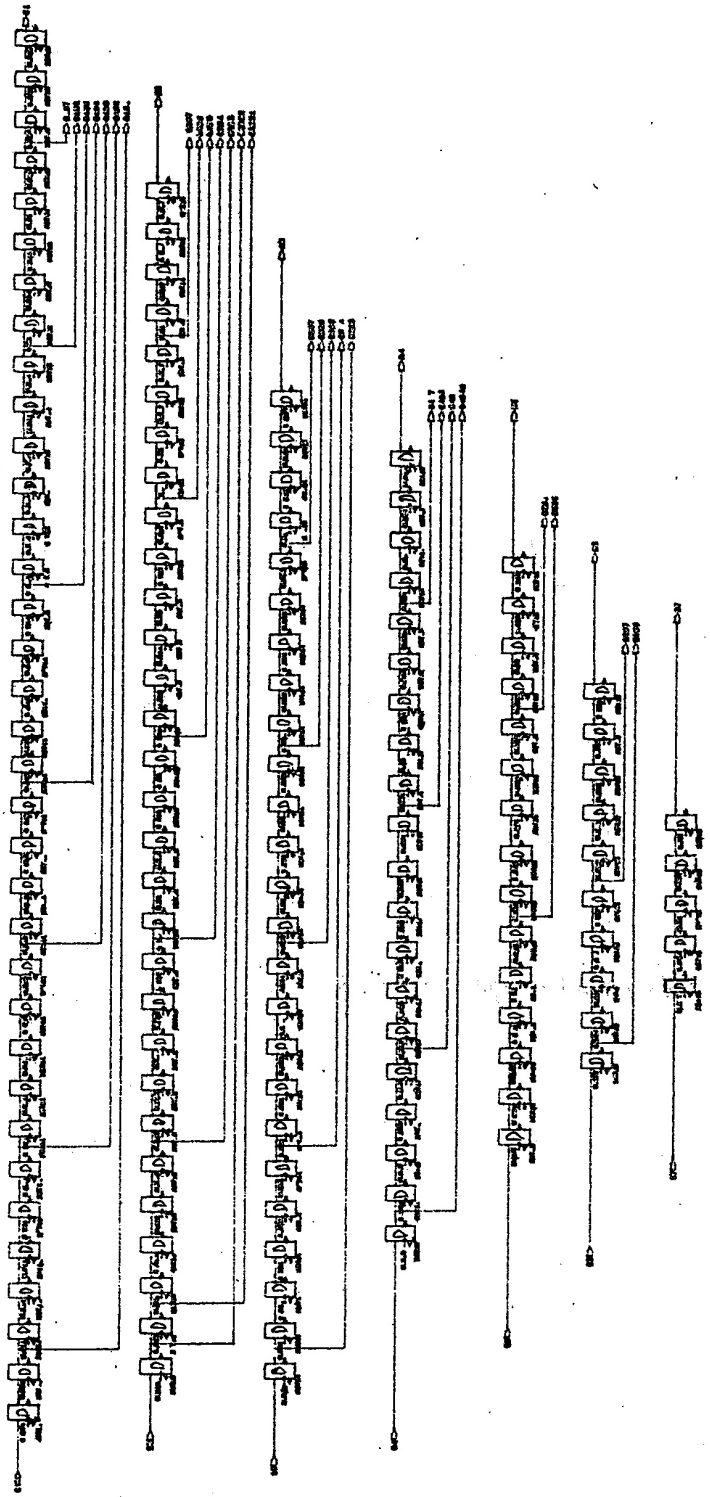
Σχέδιο του κυκλώματος γεννήτριας σημάτων ελέγχου (CON.GEN)



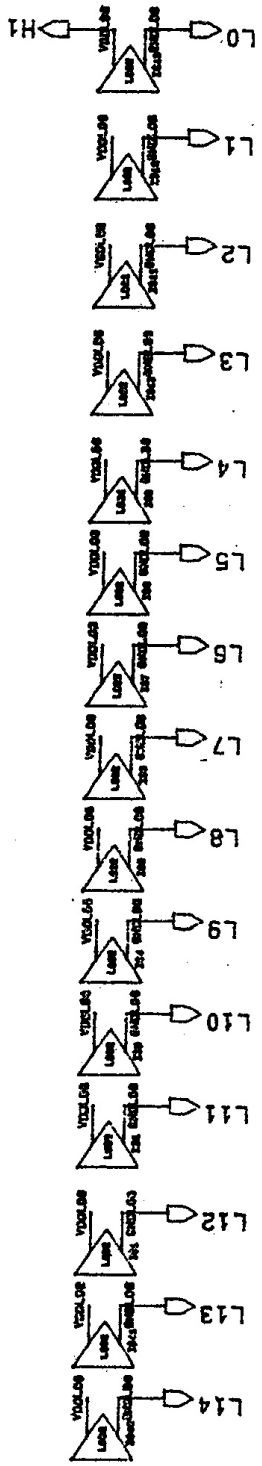
Εξομοίωση του κυκλώματος γεννήτριας σημάτων ελέγχου (CON.GEN)



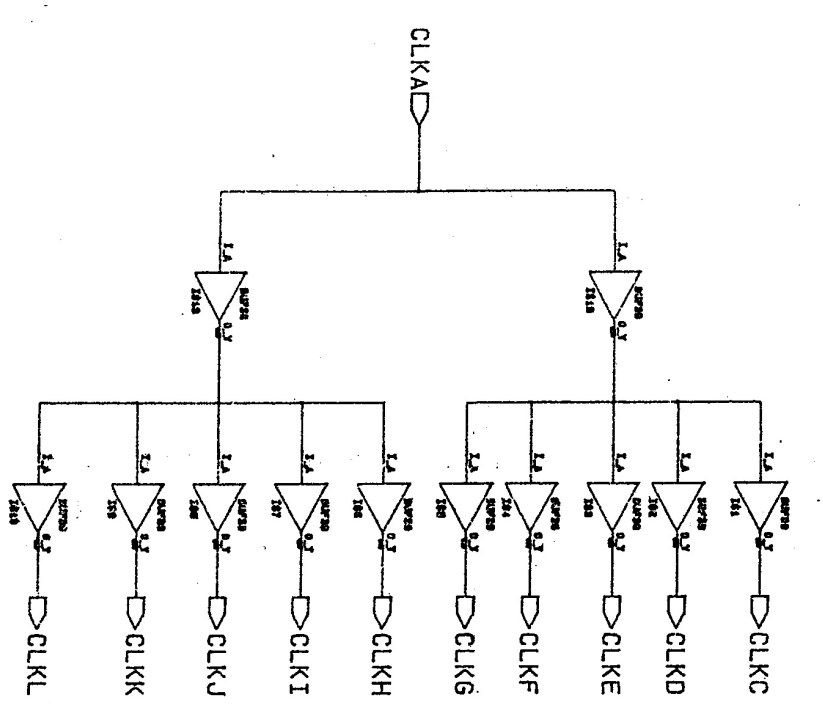
Σχέδιο του κυκλώματος καθυστέρησης (DELAYER)

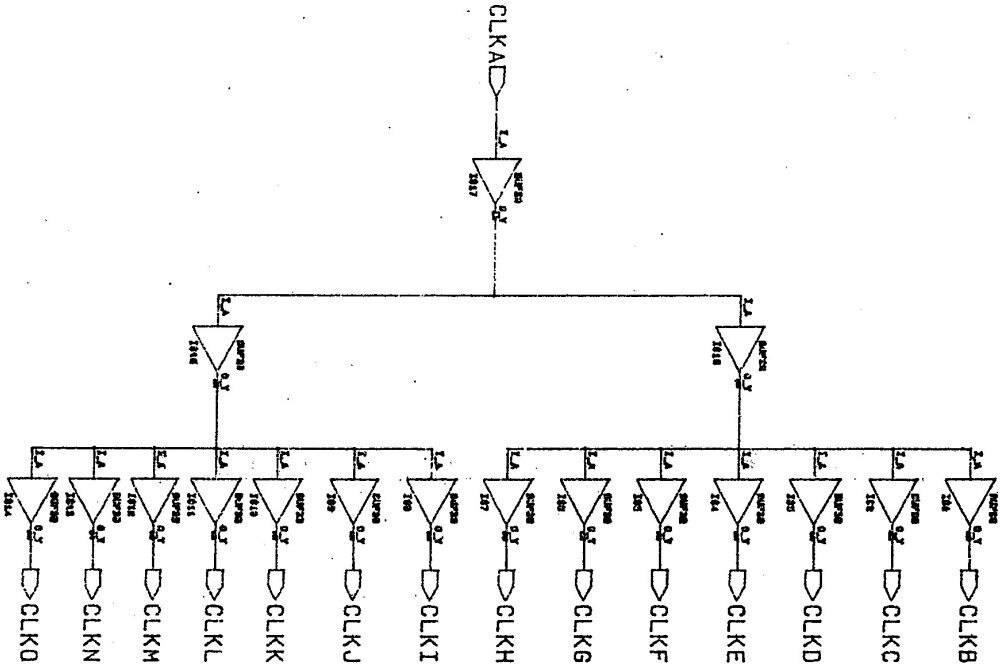


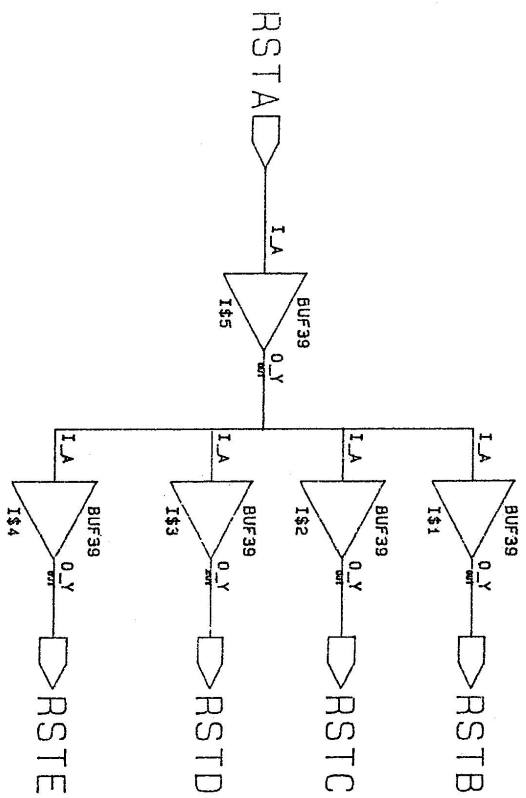
Σχέδιο του κυκλώματος παραγωγής λογικών σταθμών

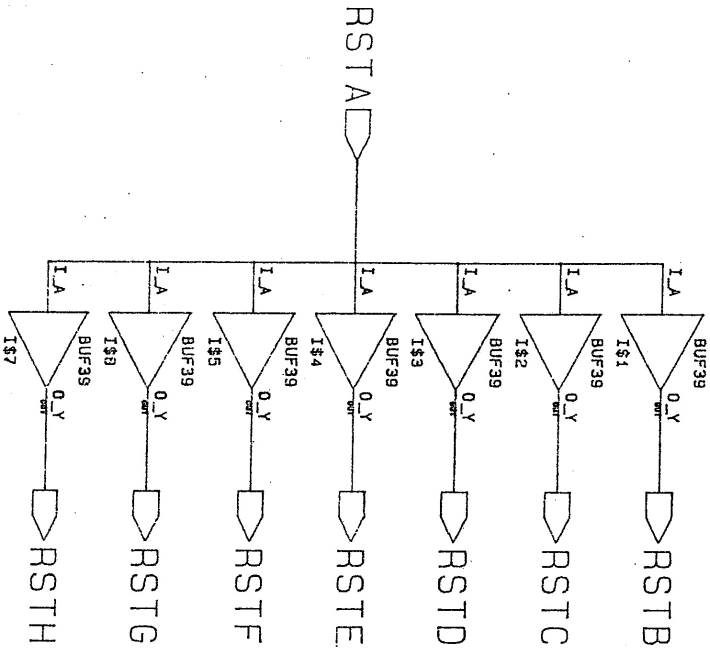


Σχέδια των κυκλωμάτων οδήγησης των σημάτων CLK και RST

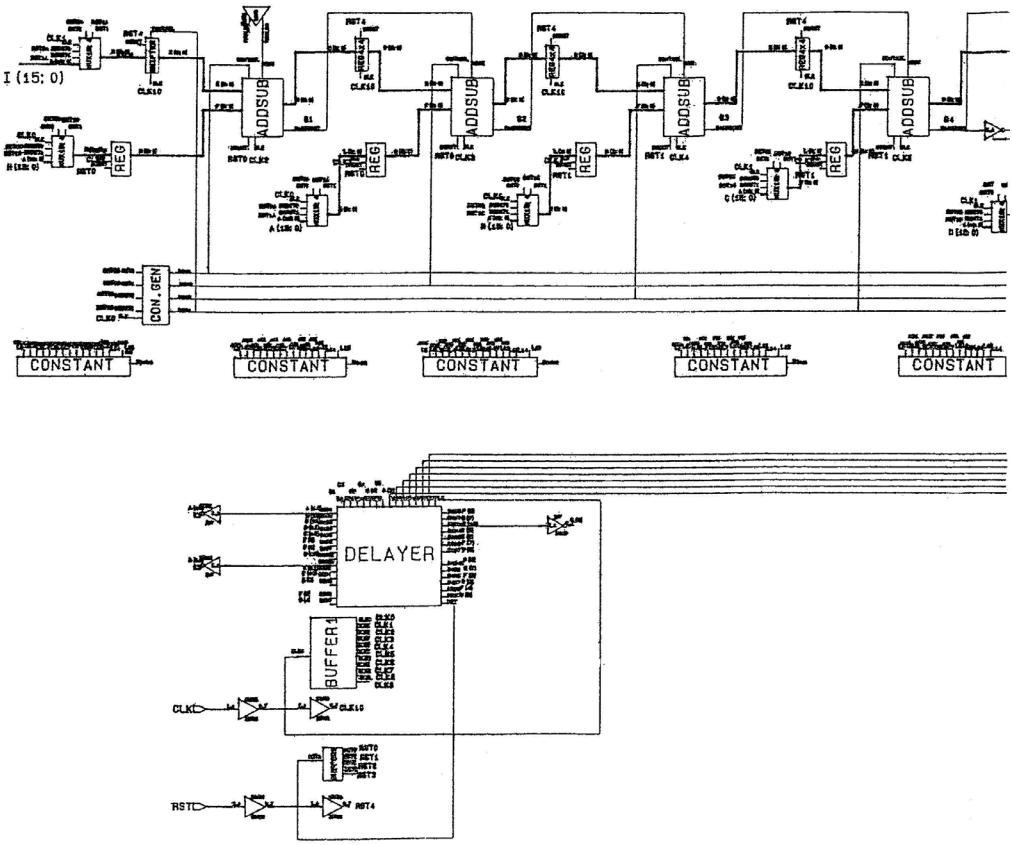





RSTA
I_A
I$5
BUF39
O_Y
OUT
I_A
I_A
I_A
I_A
I$4
I$3
I$2
I$1
BUF39
BUF39
BUF39
BUF39
O_Y
O_Y
O_Y
O_Y
OUT
OUT
OUT
OUT
RSTE
RSTD
RSTC
RSTB

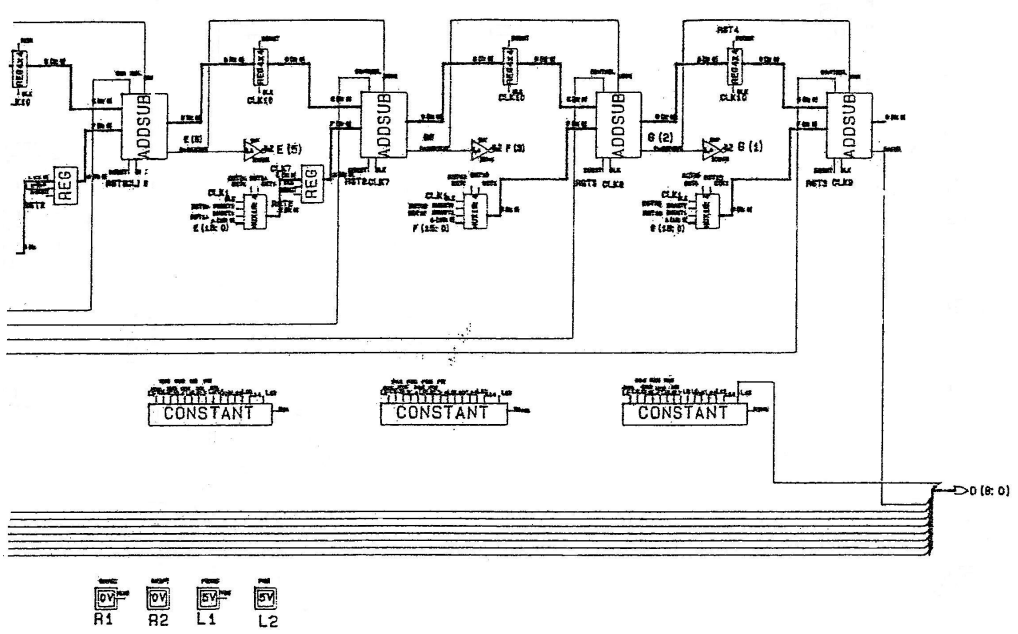

RSTA
I_A
I_A
I_A
I_A
I_A
I_A
I_A
I_A
I$7
I$6
I$5
I$4
I$3
I$2
I$1
BUF39
BUF39
BUF39
BUF39
BUF39
BUF39
BUF39
O_Y
O_Y
O_Y
O_Y
O_Y
O_Y
O_Y
OUT
OUT
OUT
OUT
OUT
OUT
OUT
RSTH
RSTG
RSTF
RSTE
RSTD
RSTC
RSTB

Σχέδιο του συνολικού κυκλώματος

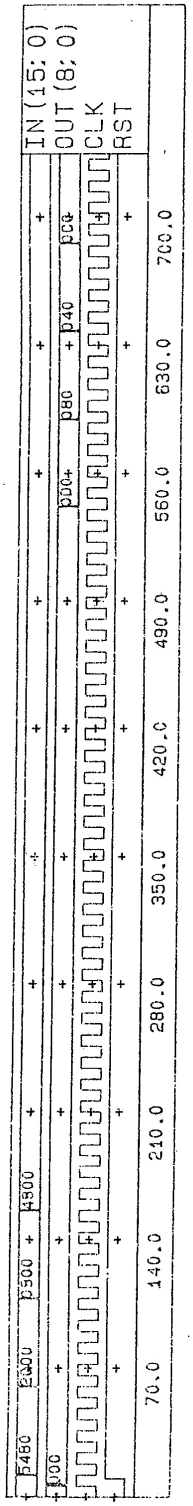


Η συνέχεια του σχεδίου του συνολικού κυκλώματος
ακολουθεί στην επόμενη σελίδα

Συνέχεια του σχεδίου του συνολικού κυκλώματος



Εξομοίωση του συνολικού κυκλώματος



Παράρτημα

Εργαλεία υλοποίησης

Αξίζει να αναφερθούμε με συνοπτικό τρόπο στα εργαλεία που χρησιμοποιήσαμε για την υλοποίηση του κυκλώματος υπολογισμού τετραγωνικής ρίζας αριθμών γιά σειριακή σε επίπεδο ψηφίου (digit-serial) επεξεργασία σημάτων. Η υλοποίηση του κυκλώματος έγινε στο σύστημα Mentor Graphics όπου χρησιμοποιήθηκε η ES2 1.5 μm βιβλιοθήκη βασικών κυττάρων (standard cell library).

1.ΓΕΝΙΚΑ

Το σύνολο των εργαλείων σχεδιασμού ES2 υποστηρίζει μια διαδρομή στη σχεδίαση συστημάτων που ξεκινά από τη σχηματική εκδοχή και καταλήγει στο φυσικό σχεδιασμό (physical layout). Η περιγραφή της διαδρομής αυτής μπορεί να διαιρεθεί σε τρείς βασικές περιοχές:

α. Σχηματικό μέρος

β. Εξομοίωση (simulation)

γ. Φυσικός σχεδιασμός (layout)

Πιο κάτω θα περιγράψουμε την συνολική διαδικασία σχεδίασης και τα εργαλεία που απαιτούνται για το καθένα από τα στάδια της διαδικασίας αυτής.

2.ΔΙΑΔΙΚΑΣΙΑ ΣΧΕΔΙΑΣΜΟΥ

Στην ενότητα αυτή παρουσιάζουμε την διαδικασία σχεδίασης στο σύστημα Mentor Graphics. Στο σύστημα αυτό έχουν δημιουργηθεί εργαλεία τα οποία υποστηρίζουν όλα τα βήματα της διαδικασίας σχεδιασμού. Τα βασικά βήματα αυτής της διαδικασίας είναι τα εξής:

1. Σχηματικό μέρος:

Κατά την διάρκεια αυτού του βήματος ο σχεδιαστής καθορίζει τη λογική του σχεδιασμού. Ο σχεδιαστής χρησιμοποιεί τα σύμβολα της βιβλιοθήκης λογικών εξαρτημάτων (logic components library) για να υλοποιήσει το σχηματικό μέρος ενός κυκλώματος. Τα λογικά σύμβολα αποτελούν την απαραίτητη πληροφορία που οδηγεί έπειτα τα εργαλεία της εξομοίωσης και του φυσικού σχεδισμού (layout). Η υλοποίηση του σχηματικού μέρους πραγματοποιείται σε κατάλληλους σταθμούς εργασίας (workstations). Το σύνολο εργαλείων ES2 παρέχει τις ES2 βιβλιοθήκες συμβόλων για την κατασκευή του σχηματικού.

2. Εξομοίωση (simulation):

Η εξομοίωση χρησιμοποιεί τα δεδομένα που αφορούν την συνδεσμολογία του σχηματικού και την πληροφορία που αφορά τον χρονισμό, για να διαπλάσει την λογική λειτουργία του συστήματος που σχεδιάζεται. Επίσης κατά την διάρκεια της εξομοίωσης αναλύονται οι εσωτερικές καθυστερήσεις των διαφόρων πυλών και χρησιμοποιείται η πληροφορία αυτή με σκοπό την πρόβλεψη της συμπεριφοράς του συστήματος. Το σύνολο εργαλείων ES2 εφοδιάζει το σχεδιαστή με την ES2 λογική βιβλιοθήκη που περιέχει σύμβολα με λειτουργικά δεδομένα και δεδομένα χρονισμού για όλα τα κύτταρα (cells). Η βιβλιοθήκη αυτή περιλαμβάνει επίσης τα απαραίτητα δεδομένα που εξομοιώνουν τις καθυστε-

ρήσεις που δημιουργούνται από τη διαδικασία κατασκευής και την χωρητικότητα των γραμμών σύνδεσης. Οι διάφορες καθυστερήσεις υπολογίζονται αυτόματα, όταν μιά κατάσταση αλλάζει και τα αρχεία εξομοίωσης ανανεώνονται αυτόματα.

3. Φυσικός σχεδιασμός (layout):

Στη διαδικασία του φυσικού σχεδιασμού τα δεδομένα που αφορούν την συνδεσμολογία, χρησιμοποιούνται για την υλοποίηση του chip. Με βάση τους κανόνες της τεχνολογίας σχεδίασης, τοποθετούνται αρχικά τα καθορισμένα βασικά κύτταρα (standard cells). Στην συνέχεια ακολουθεί η ενδοσύνδεση των κυττάρων. Η τοποθέτηση (placement) και η δρομολόγηση (routing) μπορούν να εκτελεστούν αυτόματα ή μέσω ειδικών γραφικών. Για την εκτέλεση της διαδικασίας του φυσικού σχεδιασμού ο σχεδιαστής χρησιμοποιεί τα εργαλεία τοποθέτησης και δρομολόγησης (placement & routing tools) του συστήματος Mentor Graphics.

4. Επανόρθωση:

'Οταν έχει ολοκληρωθεί ο φυσικός σχεδισμός (layout), μπορεί να παραχθεί ένα πιο ακριβές μοντέλο των καθυστερήσεων των σημάτων. Στο σημείο αυτό μπορούν επίσης να υπολογιστούν όλες χωρητικότητες. 'Ετσι μπορεί η εξομοίωση να ανανεωθεί και να περικλύει πλέον, τις fanout καθυστερήσεις, τις πραγματικές τιμές ή τις τιμές που έχουν εκτιμηθεί για τις χωρητικότητες μετάλλου και την διαδικασία κατασκευής, αυξάνοντας έτσι την ακρίβεια.

5. Δοκιμή (testing):

Μετά την κατασκευή το σχέδιο δοκιμάζεται με ένα σύνολο διέγερσεων εισόδου. Η εξομοίωση παράγει τα πρότυπα δοκιμής, τα οποία μπορούν να μετατραπούν ώστε να ταιριάζουν στη μορφή που χρησιμοποιείται από το συγκεκριμένο chip.

6. 'Ελεγχος κανόνων σχεδίασης:

Ο έλεγχος κανόνων σχεδιασμού είναι ένα απαραίτητο βήμα στη συνολική

διαδικασία σχεδίασης. Αν υπάρξει κάποια παραβίαση των ES2 κανόνων σχεδιασμού τότε προκαλούνται καθυστερήσεις στη παραγωγή του τελικού προιόντος.

3.ΕΡΓΑΛΕΙΑ ΣΧΕΔΙΑΣΗΣ

'Οσον αφορά τώρα τα εργαλεία του συστήματος Mentor Graphics που χρησιμοποιήσαμε, αυτά είναι τα εξής:

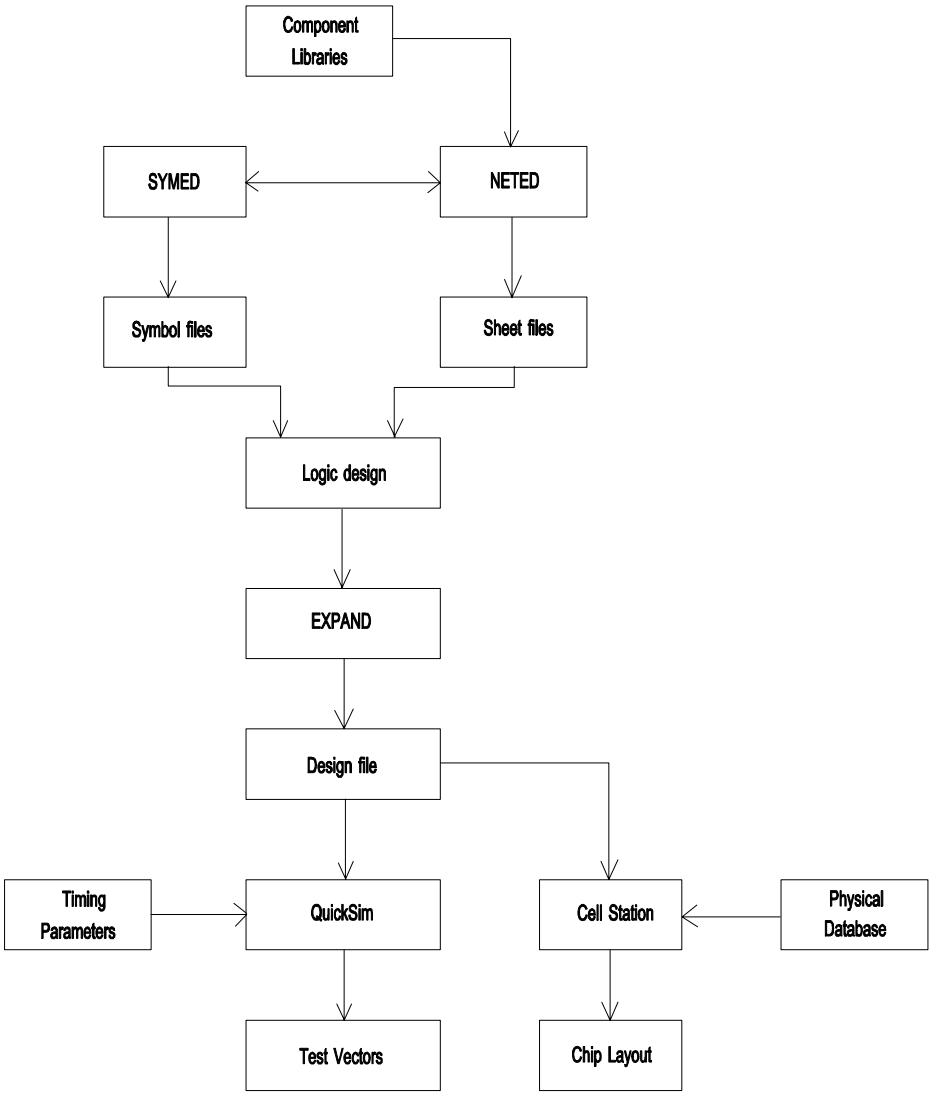
1. NETED εργαλείο για κατασκευή σχηματικού.

2. SYMED εργαλείο για κατασκευή συμβόλων, που είναι η βάση για ιεραρχική σχεδίαση.

3. EXPAND εργαλείο για σωστή προετοιμασία, ώστε να ακολουθήσει εξομοίωση και αυτόματος φυσικός σχεδισμός (layout).

4. QuickSim εργαλείο για την εκτέλεση εξομοίωσης στα ψηφιακά ηλεκτρονικά κυκλώματα.

5. Cell Station εργαλείο για τη διαδικασία του φυσικού σχεδισμού (layout).



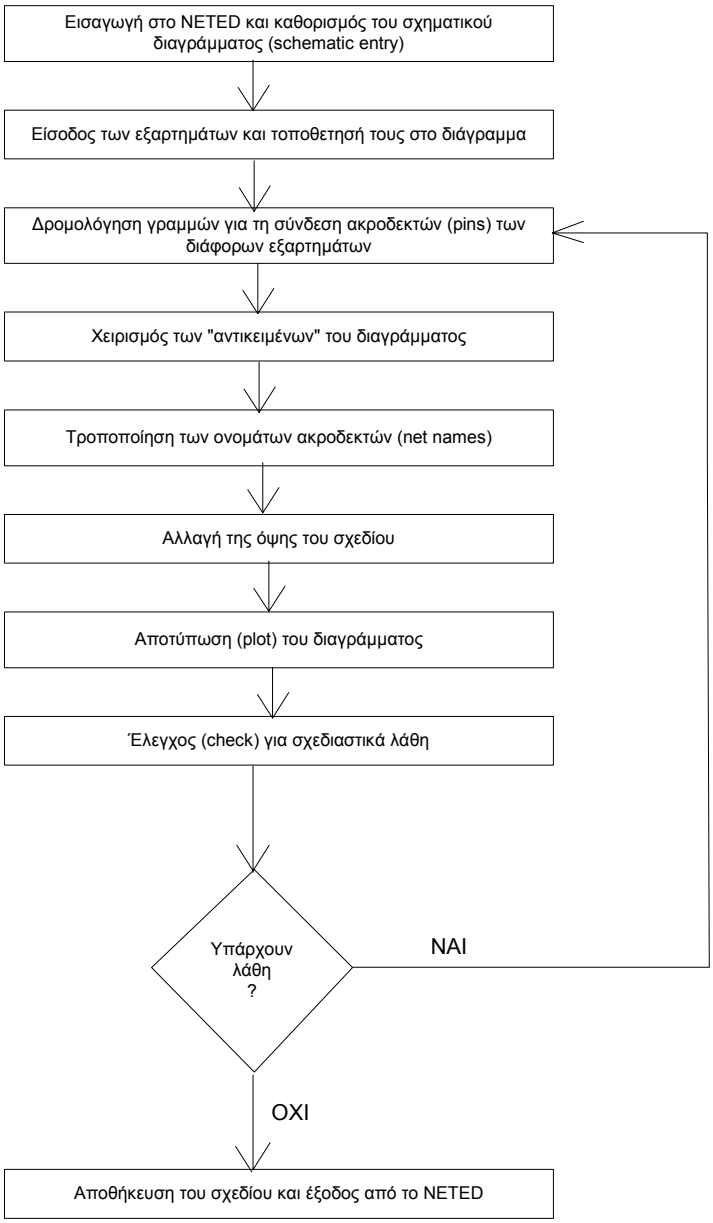
Σχήμα 1: Εργαλεία υλοποίησης

Πρίν προχωρήσουμε σε μία συνοπτική περιγραφή του κάθε εργαλείου χωριστά, δίδουμε ένα διάγραμμα στο σχήμα 1 το οποίο περιέχει όλα τα εργαλεία που χρησιμοποιήσαμε, καθώς και τις αλληλεπιδράσεις μεταξύ αυτών.

3.1 NETED

Το πρόγραμμα NETED είναι ένα από τα Shematic Entry εργαλεία που παρέχονται από το σύστημα Mentor Graphics. Μπορούμε να πούμε ότι το

NETED "παίρνει" την θέση του χαρτιού και του μολυβιού, σχεδιάζοντας σχηματικά διαγράμματα. Μπορεί να χρησιμοποιηθεί μαζί με το πρόγραμμα SYMED στο οποίο θα αναφερθούμε πιο κάτω, καθώς και μαζί με άλλα προγράμματα σχηματικού και βιβλιοθήκες εξαρτημάτων (components libraries), ώστε να δημιουργηθούν ολοκληρωμένα σχηματικά διαγράμματα κυκλωμάτων.



Σχήμα 2: Διαδικασία του NETED

Επίσης είναι δυνατόν, να ορίσουμε με το NETED διάφορα φυσικά ή λειτουργικά χαρακτηριστικά, με χρήση των "properties", τα οποία δεν θα ήταν ορατά σε άλλα σχηματικά διαγράμματα. Στα σχηματικά διαγράμματα (shematic sheets) που δημιουργούμε με το NETED, τα οποία είναι ανάλογα με τα σχηματικά διαγράμματα που σχεδιάζονται σε χαρτί, τοποθετούμε τα διάφορα εξαρτήματα, σχεδιάζουμε τις γραμμές σύνδεσης και τέλος προσθέτουμε στο κείμενο πληροφορίες κειμένου (π.χ ονόματα). Η χρήση λοιπόν του NETED έχει ως συνέπεια μια διαδικασία η οποία αποτελείται από τα βήματα που φαίνονται στο διάγραμμα του σχήματος 2.

Τα περισσότερα εξαρτήματα τυπικά προέρχονται από βιβλιοθήκες που ονομάζονται Parts Libraries. Οι Parts Libraries δεν είναι τίποτε άλλο, εκτός από καταλόγους που αποτελούνται από διάφορους τύπους εξαρτημάτων. Υπάρχουν πολλές διαφορετικές τέτοιες βιβλιοθήκες, οι οποίες είναι διαθέσιμες στο σύστημα Mentor Graphics. Επίσης μπορούν να δημιουργηθούν ειδικές βιβλιοθήκες (custom libraries), ώστε να εξυπηρετήσουν ειδικούς χρήστες. Μία Part Library περιέχει συνήθως γενικά εξαρτήματα όπως πύλες NAND, OR, αντιστροφείς και άλλα συνήθη εξαρτήματα που απαιτούνται από τον χρήστη, για την δημιουργία εξαρτημάτων ανωτέρου επιπέδου.

'Οσον αφορά τώρα το χειρισμό των διαφόρων "αντικειμένων" ενός σχηματικού διαγράμματος, υπάρχουν τρία βήματα που πρέπει να εκτελεστούν. Τα βήματα αυτά είναι τα εξής:
α. Επιλέγουμε το αντικείμενο που θα υποστεί κάποιο χειρισμό.
β. Κάνουμε τις διάφορες ενέργειες που επιθυμούμε (μετακίνηση, αντιγραφή, περιστροφή, διαγραφή) πάνω στο συγκεκριμένο "αντικείμενο".
γ. Αναιρούμε την επιλογή που είχαμε κάνει στο βήμα (α)

'Οπως βλέπουμε και στο διάγραμμα του σχήματος 2 μια απαραίτητη λειτουργία που γίνεται κατά τη χρήση του NETED, είναι η τροποποίηση των ονομάτων ακροδεκτών (Net names). Αυτό γίνεται επειδή το NETED χρησιμοποιεί δύο εξαρτήματα στο τερματισμό μιας γραμμής στο σημείο εισόδου ή εξόδου. Τα εξαρτήματα αυτά είναι τα $portin και $portout. Αρχικά, με την τοποθέτηση αυτών των εξαρτημάτων, έχουν όλα το ίδιο όνομα (NET). 'Ετσι ο εξομοιωτής (simulator)

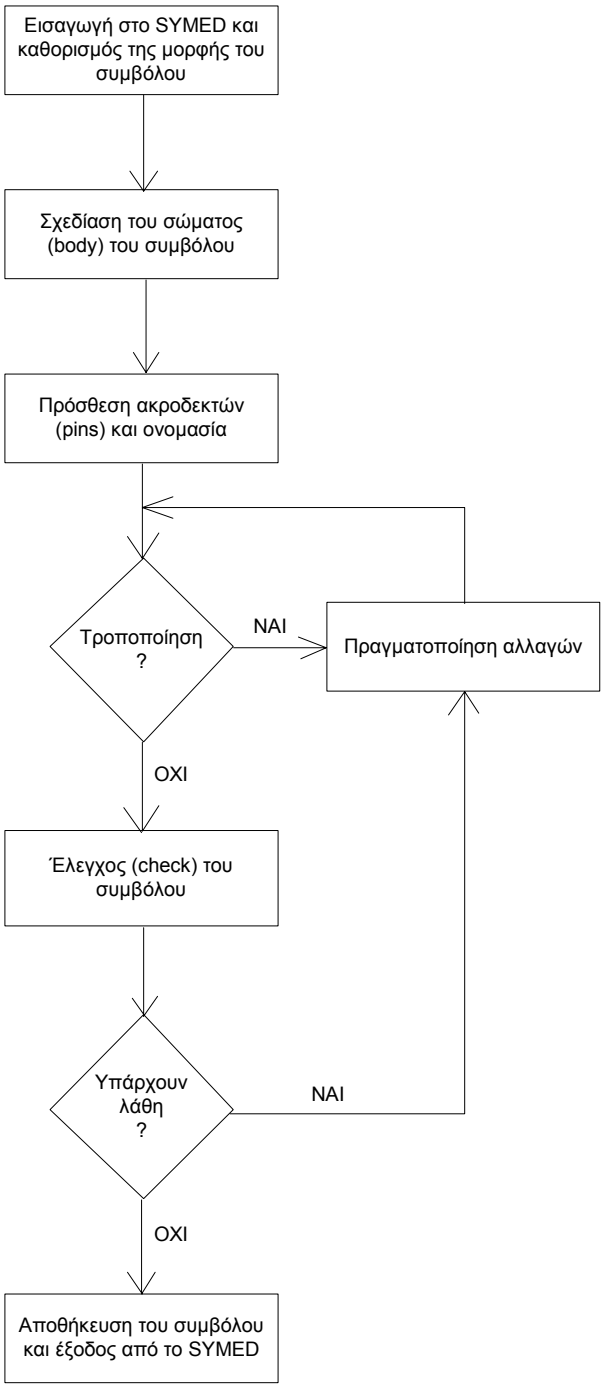
θα θεωρήσει όλα αυτά τα εξαρτήματα συνδεδεμένα. Για να μην γίνει αυτό λοιπόν, θα πρέπει να αλλάξουμε τα ονόματα, ώστε κάθε ακροδέκτης να έχει διαφορετικό και μάλιστα μοναδικό όνομα. Πολλές φορές είναι επιθυμητό από το χρήστη να δει ένα μέρος του σχηματικού διαγράμματος, ώστε να είναι ορατές όλες οι λεπτομέρειες. Το NETED δίνει τη δυνατότητα μεγέθυνσης ενός μέρους του διαγράμματος. Επίσης υπάρχει η δυνατότητα σμίκρινσης και ακόμα η δυνατότητα αλλαγής της όψης του σχηματικού διαγράματος.

Τέλος, είναι αναγκαίος ο έλεγχος του σχηματικού διαγράμματος, για τυχόν λάθη στη σχεδίαση. Αυτό φυσικά πρέπει να γίνει πριν την εξομοίωση. Η λειτουργία ελέγχου που προσφέρει το NETED, παρέχει πληροφορίες για μή συνδεδεμένους ακροδέκτες (unconnected pins), για διακοπτόμενες γραμμές, για λανθασμένα net names, για μη ισχύοντα properties κ.α. Μέτα από αυτόν τον αναγκαίο έλεγχο, πρέπει να ακολουθήσει σώσιμο (save) του σχηματικού διαγράμματος και έξοδος από το πρόγραμμα NETED.

3.2 SYMED

Το SYMED είναι και αυτό ένα από τα Shematic Entry εργαλεία που μας παρέχει το σύστημα Mentor Graphics. Μπορούμε με αυτό το πρόγραμμα να δημιουργήσουμε σύμβολα. Το SYMED συνεργάζεται με το NETED, μέσα στο οποίο μπορούν να χρησιμοποιηθούν σύμβολα που έχουν δημιουπγηθεί από το SYMED. Επίσης το πρόγραμμα SYMED μπορεί να χρησιμοποιηθεί μέσα από το πρόγραμμα NETED, ώστε να δημιουργηθεί το σύμβολο ενός σχηματικού διαγράμματος που σχεδιάστηκε στο NETED. Βέβαια είναι προφανές, ότι ο διαχωρισμός στα δύο προγράμματα έπεται από το ότι το NETED χρησιμοποιείται για να περιγράψει την λειτουργικότητα των διαφόρων εξαρτημάτων, ενώ το SYMED απλώς τα συμβολίζει. 'Ετσι μέσω των συμβόλων που παράγονται με τη χρήση του SYMED, μπορούμε εύκολα να προχωρήσουμε σε ένα ιεραρχικό τρόπο σχεδίασης. Η λειτουργικότητα ορισμένων χαμηλού επιπέδου εξαρτημάτων, όπως οι απλές πύλες, είναι γνωστή στον εξομοιωτή, οπότε χρησιμοποιείται απευθείας το συμβολό τους, χωρίς την ύπαρξη κάποιου σχηματικού διαγράμματος. Τα εξαρτήματα αυτά αναφέρονται ως PRIMITIVES.

Η διαδικασία της δημιουργίας συμβόλου κάποιου εξαρτήματος, αποτελείται από ορισμένα βήματα τα οποία φαίνονται στο διάγραμμα του σχήματος 3.



Σχήμα 3: Διαδικασία του SYMED

'Οσον αφορά το σχεδιασμό του "σώματος" (body) του συμβόλου, μπορούμε να αναφέρουμε ότι το σύμβολο μπορεί να έχει διάφορα σχήματα (ορθογώνιο, τετράγωνο, κυκλικό κλπ). Επίσης μπορούν να χρησιμοποιηθούν διάφοροι συμβολισμοί για να υποδείξουν τις θέσεις ακροδεκτών ή τον τύπο των σημάτων που θα εισέλθουν ή θα εξέλθουν από το συγκεκριμένο ακροδέκτη. 'Επειτα από το σχεδιασμό του "σώματος", το σύμβολο ονομάζεται και τοποθετούνται οι ακροδέκτες, οι οποίοι ονομάζονται κατά παρόμοιο τρόπο. Μετά από αυτά μπορούν να γίνουν στο σύμβολο διάφορες αλλαγές όπως τροποποίηση του σχήματος, τροποποίηση των θέσεων ακροδεκτών, τροποποίηση ονομάτων κ.α. Επίσης μπορούν να γίνουν χειρισμοί όπως αντιγραφή, μετακίνηση, διαγραφή και περιστροφή.

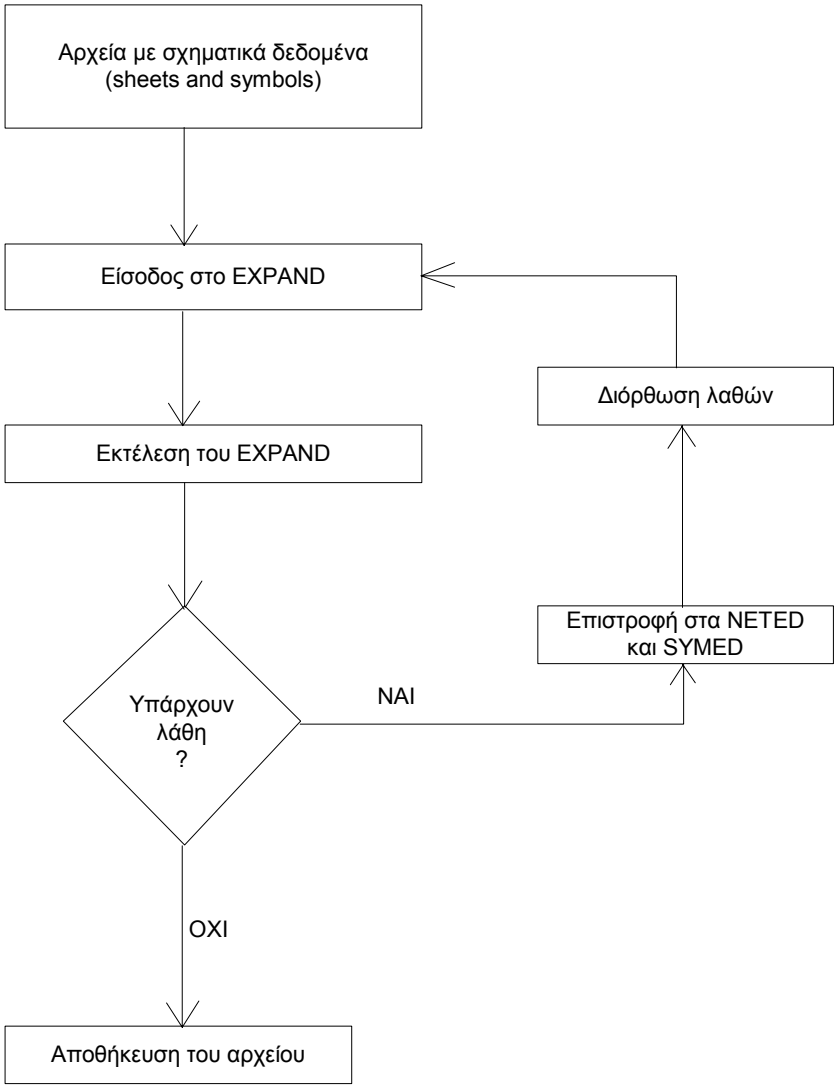
'Οπως και στη περίπτωση του NETED, πρέπει και εδώ στη συνέχεια να γίνει έλεγχος του σύμβόλου. Πιθανά λάθη που μπορεί να υποδείξει ο έλεγχος αυτός είναι η ανυπαρξία κάποιων ακροδεκτών στο σύμβολο, η ύπαρξη ενός ονόματος περισσότερες από μία φορές, καθώς και μη ισχύουσες τιμές σε κάποια properties. Τέλος, πρέπει να σώσουμε (save) το σύμβολο και να γίνει έξοδος από το πρόγραμμα SYMED.

3.3 EXPAND

Μετά την δημιουργία του σχηματικού μέρους, που έγινε με χρήση των προγραμμάτων NETED & SYMED και διαφόρων βιβλιοθηκών εξαρτημάτων (component libraries), η βάση δεδομένων περιλαμβάνει όλα τα δεδομένα και τις πληροφορίες, που απαιτούνται ώστε να επεκταθεί το σχέδιο και να δημιουργηθεί ένα αρχείο σχεδίου (design file). Το αρχείο αυτό είναι το αποτέλεσμα της χρήσης του EXPAND, που ουσιαστικά είναι το στάδιο προετοιμασίας της εξομοίωσης (simulation) και της αυτόματης παραγωγής του φυσικού σχεδίου (layout).

Η βασική λειτουργία του προγράματος EXPAND είναι να μεταφράζει τα αρχεία που περιέχουν σχηματικά μόνο δεδομένα και να θέτει τα δεδομένα αυτά σε κάποια ιεραρχική σειρά. Επίσης το EXPAND επαληθεύει τους ηλεκτρικούς περιορισμούς και τα υπάρχοντα εξαρτήματα των σχηματικών διαγραμμάτων

(sheets) του σχεδίου. Στο διάγραμμα του σχήματος 4 φαίνεται η διαδικασία που ακολουθείται κατά τη χρήση του EXPAND. Αρχικά, με την είσοδο στο πρόγραμμα EXPAND, εισάγονται και οι κανόνες για την δημιουργία του αρχείου σχεδίου (design file). Κατά την εκτέλεση του προγράμματος, γίνεται επεξεργασία των σχηματικών διαγραμμάτων (sheets) και των συμβόλων, ώστε να παραχθούν κάποιες χρήσιμες πληροφορίες γι'αυτά. Αν κατά την επεξεργασία αυτή προκύψουν λάθη, τότε αυτά διορθώνονται με επιστροφή στα προγράμματα NETED & SYMED.



Σχήμα 4: Διαδικασία EXPAND

Μετά την ανάλυση χωρίς λάθη, όλων των εξαρτημάτων, σώζουμε το νέο αρχείο σχεδίου (design file) και έτσι έχει γίνει πλέον η κατάλληλη επεξεργασία ώστε να ακολουθήσει εξομοίωση (simulation) ή αυτόματη παραγωγή του φυσικού σχεδίου (layout).

'Οσον αφορά την εκτέλεση του EXPAND, αναφέρουμε πιο κάτω τις δυνατότητες του προγράμματος αυτού:

α. Επαληθεύει το ότι τα ονόματα των ακροδεκτών (pins) του συμβόλου είναι τα ίδια με τα ονόματα των αντίστοιχων ακροδεκτών του σχηματικού διαγράμματος (shematic sheets).

β. Ελέγχει την εγκυρότητα των διαφόρων εξαρτημάτων, ώστε να δούμε αν τα εξαρτήματα αυτά υπάρχουν στη βάση δεδομένων. Για παράδειγμα ελέγχει αν για κάποιο σχηματικό διάγραμμα υπάρχει το αντίστοιχο σύμβολο.

γ. Εκτελεί άλλους έλεγχους, όπως είναι η ονομασία ενός bus, επειδή ορισμένες τιμές δέν έχουν ερμηνευτεί μέχρι την εκτέλεση του EXPAND.

δ. Ελέγχει για μία ακόμη φορά τα properties του σχηματικού μέρους.

Τέλος αναφέρουμε ότι η εκτέλεση του EXPAND, βασίζεται σε μία μόνο εντολή, στην οποία αναφέρουμε το όνομα του προγράμματος και το όνομα του αρχείου, το οποίο θέλουμε να επεκτείνουμε.

3.4 QuickSim

Το πρόγραμμα QuickSim εκτελεί λογική εξομοίωση (logic simulation) για ψηφιακά ηλεκτρονικά κυκλώματα. Για κυκλώματα άλλου είδους (π.χ αναλογικά) υπάρχουν άλλα προγράμματα εξομοίωσης όπως τα AccuSim, MSIMON κ.α. Το QuickSim πρόγραμμα παρέχει εξομοίωση, χρησιμοποιόντας software & hardware μοντέλα ψηφιακών κυκλωμάτων και επιτρέπει γρήγορη και ακριβή ανάλυση της απόδοσης του κυκλώματος σε κανονικούς τύπους λειτουργίας,

βασισμένη σε μοντέλα ακρίβειας. Το QuickSim μπορεί να χρησιμοποιεί πολλαπλές τεχνικές μοντελοποίησης ώστε να αναλύει τα δεδομένα του σχεδίου. Οι τύποι των μοντέλων που είναι διαθέσιμοι είναι οι εξής:

α. Built-in Primitives: SSI πύλες τις οποίες καταλαβαίνει απευθείας το Quick Sim.

β. QuickParts: Μοντέλα που μεταφράζονται μέσω ενός αρχείου χρονισμού (timing file).

γ. BLMs: Προγράμματα γραμμένα σε Pascal ή C τα οποία "μιμούνται" τη λειτουργία ενός στοιχείου.

δ. HLMs: Πραγματικά εξαρτήματα μαζί με ένα αρχείο χρονισμού, χρησιμοποιούνται ως μοντέλα.

Το QuickSim είναι ένας εξομοιωτής κυκλικού χρονισμού (timing wheel), 12 καταστάσεων, για MOS, TTL και ECL λογική. Ενώ το QuickSim εξομοιώνει την λειτουργία των ψηφιακών κυκλωμάτων, δεν μπορεί να αναγνωρίσει ρεύμα, τάση ή fanout.

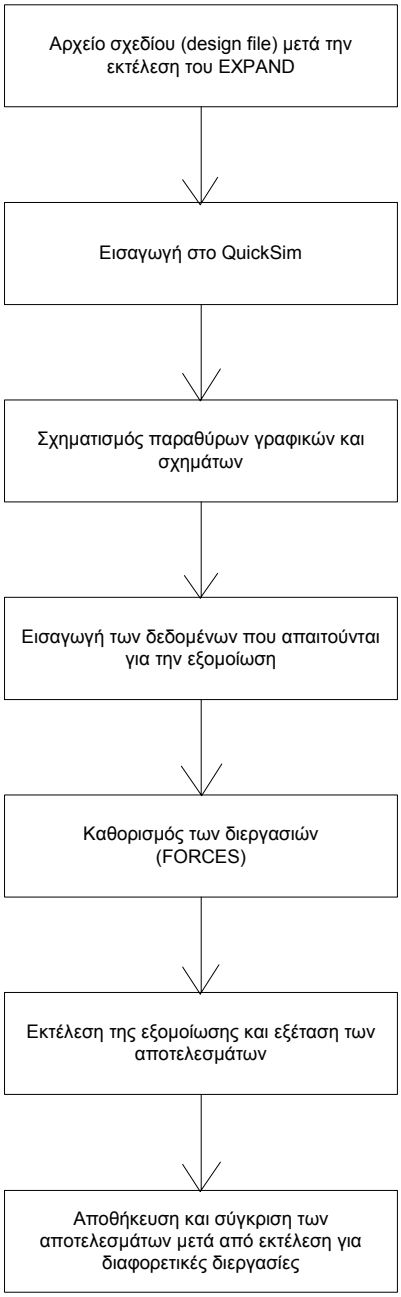
Στο διάγραμμα του σχήματος 5, φαίνεται η διαδικασία της εξομοίωσης, με χρήση του προγράμματος QuickSim του συστήματος Mentor Graphics.Αρχικά, μετά την εισαγωγή στο πρόγραμμα QuickSim, σχηματίζονται διάφορα παράθυρα μέσα στο περιβάλλον του προγράμματος, μέσω των οποίων βλέπουμε τα δεδομένα και τα αποτελέσματα σε διάφορες μορφές. Τα παράθυρα αυτά είναι τα εξής:

α. Transcript: είναι το παράθυρο όπου παρουσιάζονται οι εντολές που έχουν εισαχθεί προηγούμενα.

β. Status: είναι το παράθυρο που υποδεικνύει την κλίμακα χρόνου και την βάση του συστήματος παράστασης των αριθμητικών δεδομένων (Hex, binary κλπ).

γ. Trace: είναι το παράθυρο, όπου δίδονται οι κυματομορφές των σημάτων.

δ. List: είναι το παράθυρο που περιλαμβάνει την κατάσταση των διαφόρων δεδομένων σε αριθμητική μορφή. Οι τιμές μπορούν να είναι σε δεκαεξαδικό, δεκαδικό, οχταδικό ή δυαδικό σύστημα.



Σχήμα 5: Διαδικασία του QuickSim

ε. Monitor: είναι το παράθυρο που υποδεικνύει την τρέχουσα κατάσταση κάθε σήματος.

στ. View: είναι το παράθυρο που δείχνει το σχηματικό διάγραμμα (sheet) του κυκλώματος που αναλύεται. Μετά την εμφάνιση των 6 παραθύρων, εισάγουμε τα δεδομένα και κατόπιν κάνοντας χρήση κυρίως της εντολής FORCE εισάγουμε τις διάφορες διεγέρσεις που απαιτούνται από το κύκλωμα ώστε να εξομοιώσουμε την λειτουργία του.

Τέλος, μετά την εκτέλεση της εξομοίωσης, σώζουμε τα αποτελέσματα και τα συγκρίνουμε με αποτελέσματα από προηγούμενες εξομοιώσεις (με διαφορετικά δεδομένα) του ίδιου κυκλώματος.

3.5 Cell Station

Το Cell Station είναι ένα σύστημα από software εργαλεία για την παραγωγή του φυσικού σχεδίου τύπου cell-based. Το σύστημα Cell Station περιλαμβάνει έναν editor τοποθέτησης (placement) και δρομολόγησης (routing) που ονομά-ζεται CellGraph. Επίσης έχουμε στη διαθεσή μας τα ακόλουθα αυτόματα ερ-γαλεία:

α. CellPlace, το οποίο περιλαμβάνει τα CellFloor & CellPlace προγράμματα που εκτελούν αυτόματη τοποθέτηση (auto-placement).

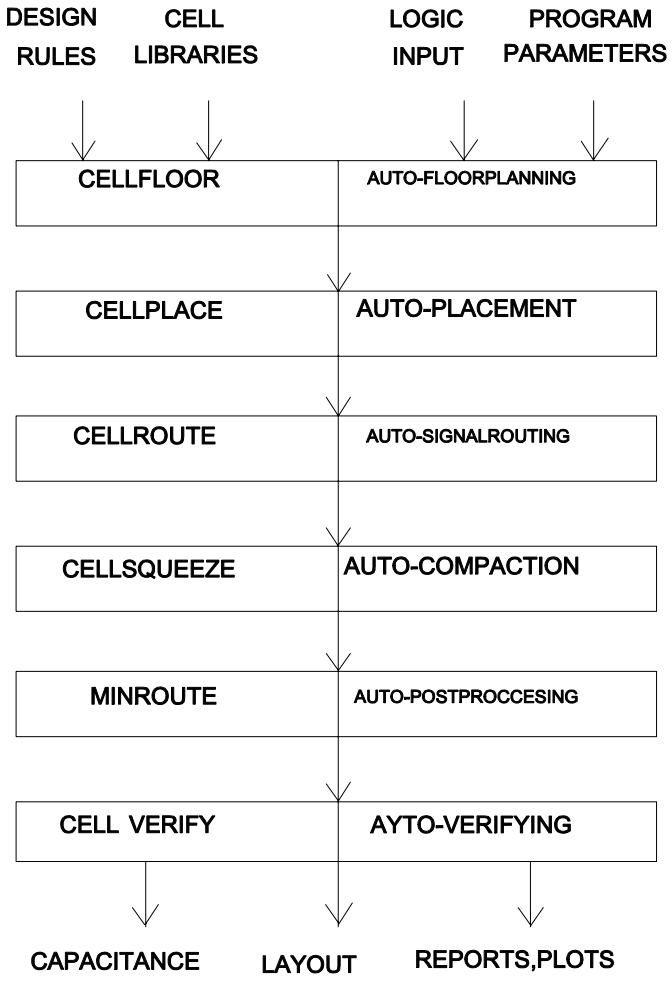
β. CellRoute, το οποίο περιλαμβάνει τα CellRoute, Cellsqueeze & Minroute προγράμματα για αυτόματη δρομολόγηση (auto-routing).

Μπορούμε γενικά να παράγουμε το φυσικό σχεδισμό (layout), μόνο με τη χρήση του CellGraph. Ωστόσο, τυπικά χρησιμοποιούνται και τα αυτόματα προγράμματα μαζί με το CellGraph.

Στο διάγραμμα του σχήματος 6 φαίνεται η ροή της διαδικασίας χρήσης του συστήματος CellStation. Το σύστημα Cell Station απαιτεί ως εισόδους τα εξής:

α. Κανόνες σχεδίασης: Πρέπει αρχικά να γίνει ο καθορισμός των κανόνων σχεδιασμού και ο καθορισμός της τεχνολογίας της βιβλιοθήκης κυττάρων (cell library).

β. Βιβλιοθήκες κυττάρων (cell libraries): Το σύστημα Cell Station υποστηρίζει ένα σύνολο από κατάλληλες βιβλιοθήκες κυττάρων.



Σχήμα 6: Διαδικασία του CellStation

γ. Λογική είσοδος: Η λογική είσοδος αποθηκεύεται στον Design Directory του συστήματος Cell Station.

δ. Παράμετροι προγράμματος (program parameters): Πολλά από τα εργαελία φυσικού σχεδισμού παρέχουν παραμέτρους εισόδου, οι οποίες επιτρέπουν τον ελέγχο της διαδικασίας και τη μορφή των αποτελεσμάτων του φυσικού σχεδιασμού (layout).

Η επεξεργασία αρχίζει με κατασκευή του "floorplan" του chip. 'Επειτα γίνεται η τοποθέτηση (placement), η οποία ακολουθείται από τη δρομολόγηση των γραμμών (routing). Κάθε ένα από αυτά τα βήματα της επεξεργασίας βασίζεται στη λογική είσοδο και σε τεχνολογικούς περιορισμούς. Στη συνέχεια γίνεται συμπαγοποίηση (compaction) του chip και ελαχιστοποίηση της περιοχής του. Τέλος το chip επαληθεύεται (verifying). Στο τέλος της διαδικασίας παραγωγής του φυσικού σχεδίου (layout), το σύστημα Cell Station εξασφαλίζει ένα σύνολο από χρήσιμες εξόδους. Αυτές είναι οι εξής:

α. Χωρητικότητες: Οι τιμές τους βασίζονται στη φυσική σχεδίαση. Μπορούν να χρησιμοποιηθούν στη επαναεξομοίωση (resimulation), ώστε να γίνει ένας ολοκληρωμένος έλεγχος του chip.

β. Αρχείο φυσικού σχεδιασμού (layout file): Μπορούμε να εγγράψουμε το αρχείο αυτό σε κάποια ταινία, ώστε να οδηγηθεί για βιομηχανική παραγωγή.

γ. Απολογιστικά στοιχεία: Τα στοιχεία αυτά μπορούν να οδηγήσουν σε στατιστική ανάλυση.

δ. Εκτύπωση του φυσικού σχεδιασμού: Φυσικά αυτό θα γίνει με την βοήθεια κάποιας σχεδιαστικής μηχανής (plotter).

Βιβλιογραφία

[1] R. Hartley and P. Corbett, "Digit-serial Processing Techniques", *IEEE Trans. on Circuits and systems*, vol.37, no.6, pp. 707-719, June 1990.

[2] K. Parhi, "A Systematic Approach for Design of Digit-Serial Signal Processing Architectures", *IEEE Trans. on Circuits and Systems*, vol.38, no.4, pp. 358-375, April 1991.

[3] P. Denyer and D. Renshaw, "VLSI Signal Processing, A Bit Approach", *Addison-Wesley*, 1985.

[4] K. Hwang, "Computer Arithmetic: Principles, Architecture and Design", *Wiley*, 1979.

[5] K. Hwang and F.A. Briggs, "Computer Architecture and Parallel Processing", *Mc Graw-Hill*, 1984.

[6] M.M. Mano, "Digital Design", *Prentice-Hall*, 1984.

[7] J. Millman and A. Grabel, "Microelectronics", *Mc Graw-Hill*, 1987.

[8] A. Mukherjee, "Introduction to nMOS and CMOS VLSI Systems Design", *Prentice-Hall*, 1986.

[9] Γ. Παπακωνσταντίνου, Π.Τσανάκας, Γ.Φραγκάκης, "Αρχιτεκτονική Υπολογιστών", *Συμμετρία*, Αθήνα, 1987