ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αποδοτική Σχεδίαση Τροποποιημένων SRT Αλγορίθμων και Προσομοίωση κατόπιν Σύνθεσής τους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Του

Μπάκα Κωνσταντίνου

Επιβλέπων : Κιαμάλ Πεκμεστζή

Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αποδοτική Σχεδίαση Τροποποιημένων SRT Αλγορίθμων και Προσομοίωση κατόπιν Σύνθεσής τους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Του

Μπάκα Κωνσταντίνου

Επιβλέπων : Κιαμάλ Πεκμεστζή

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Μαρτίου 2016

……..	……	…….
Κ. Πεκμεστζή	Δ. Σούντρης	Γ. Οικονομάκος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επ.Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2016

.....
ΜΠΑΚΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η παρούσα διπλωματική εργασία, αφορά την διερεύνηση και υλοποίηση αποδοτικών αλγορίθμων διαίρεσης. Αρχικά, αναπτύσσεται η θεωρία της διαίρεσης, και των αλγορίθμων υλοποίησής της σε ψηφιακά συστήματα. Ακολουθεί η περιγραφή δύο σχεδίων, που χρησιμοποιούν διαφοροποιήσεις του αλγόριθμου SRT με βάση το 2, και επιλέχθηκαν από τη βιβλιογραφία ως αναφορές, ενώ αναπτύσσεται μία επιπλέον διαφοροποίηση του SRT, για τις ανάγκες της εργασίας.

Η πρώτη εργασία, εστιάζει στην άμεση εκκίνηση της πράξης σε κάθε επαναληπτικό βήμα, και τον παράλληλο υπολογισμό του ψηφίου πηλίκου και του σήματος επιλογής για τους πολυπλέκτες. Το κρίσιμο μονοπάτι της δεύτερης εργασίας, περιλαμβάνει μία σχετικά δύσκολη απόφαση του ψηφίου πηλίκου, μία εν συνεχεία γρήγορη παραγωγή του ακριβούς πολλαπλάσιου του διαιρέτη, και τέλος την εκτέλεση της πράξης. Συνεπώς, δεν χρησιμοποιεί πολυπλέκτες. Το σχέδιο που αναπτύχθηκε, υπολογίζει τα πρώτα δύο bits της RSD αναπαράστασης σε συμπλήρωμα ως προς δύο. Ακόμα, χρησιμοποιεί ταυτόχρονα τους κανόνες επιλογής του SRT και του non-Restoring, απλοποιώντας περαιτέρω την απόφαση του ψηφίου πηλίκου. Κατά συνέπεια, απλοποιείται σημαντικά και ο ακριβής υπολογισμός του πολλαπλάσιου του διαιρέτη, και αποφεύγεται έτσι η χρήση σειράς από πολυπλέκτες. Λόγω της υβριδικής αναπαράστασης, ο αλγόριθμος δεν κινδυνεύει από υπερχείλιση αναπαράστασης, αλλά απαιτείται η διάδοση κρατουμένου κατά μήκος 2 bits.

Τέλος, τα τρία σχέδια αναλύθηκαν θεωρητικά, και στη συνέχεια υλοποίηθηκαν στη γλώσσα Verilog με μεταβλητό μήκος ορισμάτων. Ακολούθησε η σύνθεση και προσομοίωσή τους σε ASIC, σε τεχνολογία 65nm της TSMC, και η σύγκριση των μετρήσεων με σκοπό την εύρεση της αποδοτικότερης υλοποίησης.

Λέξεις Κλειδιά

Διαίρεση, SRT, non-restoring, Κινητής-Υποδιαστολής, Κωδικοποίηση Redundant Signed Digit, Συνδυαστικός Πίνακας, Verilog, ASIC, CMOS 65nm

Abstract

This diploma project, presents the investigation and implementation of efficient division algorithms. First, the paper states the theory of division and argorithms, which are implemented to digital systems. Subsequently, two designs that use modifications of radix two SRT drawn from bibliography are described. Particularly in this paper, an additional modification of SRT has been developed.

The first design focuses on the immediate start of the add or subtract operation on every cycle, and the concurrent calculation of the quotient selection as well as the generation of the selection signal for the multiplexers. The critical path of the second design contains a complex quotient selection, a quick but accurate generation of the divisor multiple, and finally the execution of the add/subtract operation. Consequently, multiplexers are not used. The design that has been developed, expresses the two first bits of the RSD representation in two's complement. Additionally, it combines the selection of rules of SRT and non-restoring, simplifying the quotient selection. This results in important simplification of accurate divisor multiple generation, and the use of multiplexers is avoided. Representation Overflow is avoided due to the use of hybrid representation, but two digits carry propagation is required.

Finally the three designs were analyzed in theoretical basis, and furthermore, they have been implemented in Verilog HDL with variable operands length. The three designs were synthesized and simulated in an 65nm TSMC ASIC process, and compared in order to determine the efficiency of each implementation.

Key Words

Division, SRT, non-restoring, Floating-Point, Redundant Signed Digit encoding, Combinational Array Divider, Verilog, ASIC, CMOS 65nm

ΕΥΧΑΡΙΣΤΙΕΣ

Πρώτα απ’ όλα, θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή κ. Κιαμάλ Πεκμεστζή, για την πολύτιμη βοήθεια, καθοδήγηση και την εμπιστοσύνη που μ έδειξε κατά τη διάρκεια της εκπόνησης της διπλωματικής εργασίας μου. Επίσης, είμαι ευγνώμων στα μέλη της διδακτορικής ομάδας που με βοήθησαν με τις υποδείξεις τους.

Τέλος, θέλω να ευχαριστήσω ιδιαιτέρως την οικογένεια μου, και όσους στάθηκαν δίπλα μου κατά την φοιτητική μου πορεία.

Πίνακας Περιεχομένων

1 ΕΙΣΑΓΩΓΗ

1.1 Ψηφιακή σχεδίαση

Η ψηφιακή σχεδίαση, αφορά τη σχεδίαση ψηφιακών ηλεκτρονικών κυκλωμάτων. Τις τελευταίες δεκαετίες αναπτύχθηκε ραγδαία, παράλληλα με την αύξηση της χρήσης τεχνολογικών προϊόντων. Ένας φαύλος κύκλος μαζικοποίησης της παραγωγής και μείωσης του κόστους, οδήγησε στην εξέλιξη των κινητών τηλεφώνων σε υπολογιστές τσέπης, στην εξέλιξη γενικότερα των φορητών υπολογιστικών συστημάτων, αλλά και στην δημιουργία τεράστιων υπολογιστικών συστημάτων, όπως είναι οι servers και τα data centers.

Οι σημαντικότερες παράμετροι σχεδίασης, είναι η καθυστέρηση, η επιφάνεια και η κατανάλωση, ενώ είναι φανερό πως οι απαιτήσεις ενός συστήματος καθορίζουν την τελική τους βαρύτητα. Οι φορητές συσκευές, αποτελούν μία ιδιαίτερη κατηγορία απαιτήσεων, καθώς ο κομψός σχεδιασμός, η χρήση μπαταρίας και η ανάγκη μεγάλης υπολογιστικής ισχύος, απαιτούν καθολική βελτιστοποίηση του συστήματος. Αντίθετα, οι επιτραπέζιοι υπολογιστές και οι servers απαιτούν χαμηλή καθυστέρηση, χωρίς να περιορίζονται σημαντικά από την επιφάνεια ή την κατανάλωση.

Συνέπεια της ανάγκης για ευρεία ανάπτυξη και βελτιστοποίηση ψηφιακών συστημάτων, ήταν η ανάπτυξη και τυποποίηση γλωσσών περιγραφής υλικού (Verilog, VHDL). Κατ' επέκταση, αναπτύχθηκαν σύνθετα σχεδιαστικά εργαλεία που απλοποιούν αρκετά τη διαδικασία σχεδίασης, ελέγχου και βελτιστοποίησης.

Μερικοί τομείς της ψηφιακής σχεδίασης, είναι η ψηφιακή επεξεργασία σήματος, η κρυπτογραφία, η μηχανική όραση και τα γραφικά υπολογιστών. Ενδεικτικά, η ψηφιακή επεξεργασία σήματος ασχολείται με τον μαθηματικό χειρισμό ενός σήματος και τους τρόπους επεξεργασίας του. Σημαντικά πεδία της είναι, η επεξεργασία ήχου και ομιλίας, η επεξεργασία εικόνας, η επεξεργασία ιατρικών σημάτων, τα συστήματα τοποθεσίας, και ο αυτόματος έλεγχος.

Η διαίρεση αποτελεί ένα βασικό στοιχείο των αλγορίθμων στις ανωτέρω εφαρμογές, ενώ η υλοποίησή της παραμένει πρόκληση. Προκειμένου να αξιοποιηθεί η εκτεταμένη έρευνα και θεωρία, η σχεδίαση μίας αποδοτικής υλοποίησης μπορεί να γίνει αρκετά πολύπλοκη, καθώς οι αποφάσεις και οι παράμετροι που πρέπει να εξετάσει ο σχεδιαστής, ανάλογα με τις απαιτήσεις του συστήματος, είναι πολλές.

1.2 Αντικείμενο διπλωματικής

Αντικείμενο της παρούσης διπλωματικής είναι η διερεύνηση και υλοποίηση αποδοτικών αλγορίθμων διαίρεσης.

Συγκεκριμένα, πραγματοποιήθηκε εκτεταμένη διερεύνηση της βιβλιογραφίας, και επιλέχθηκαν δύο παραλλαγές του αλγορίθμου SRT, για ανάλυση και υλοποίηση. Παράλληλα, αναπτύχθηκε και υλοποιήθηκε τρίτη παραλλαγή του αλγορίθμου, με σκοπό την περαιτέρω επιτάχυνσή του. Εν συνεχεία, τα τρία σχέδια αναλύθηκαν θεωρητικά με παράμετρο το μήκος των ορισμάτων, και συγκρίθηκαν ως προς την καθυστέρηση και την επιφάνειά τους.

Στα πλαίσια της εργασίας, υλοποιήθηκαν στη γλώσσα Verilog τα τρία σχέδια, με μεταβλητό μήκος ορισμάτων. Κάθε διαιρέτης υλοποιήθηκε σε μορφή συνδυαστικού πίνακα, και πλαισιώθηκε απο ίδιο κύκλωμα ελέγχου, έτσι ώστε να δέχεται και να επιστρέφει ορίσματα σε μορφή, κατά τα πρότυπα της IEEE. Η λειτουργία των υλοποιήσεων αυτών, προσομοιώθηκε και επαληθεύτηκε με το περιβάλλον ModelSim [1]. Έπειτα πραγματοποιήθηκε σύνθεση και προσομοίωση των παραπάνω κυκλωμάτων σε ASIC στα 65 nm, και εξήχθησαν τα αποτελέσματα καθυστέρησης, επιφάνειας και κατανάλωσης, από τον Synopsys Design Compiler [2].

1.3 Συνεισφορά διπλωματικής

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήθηκαν τα προβλήματα της σχεδίασης κυκλωμάτων διαίρεσης. Αναπτύχθηκαν οι κατηγορίες των αλγορίθμων διαίρεσης και οι τρόποι βελτίωσής τους.
2. Επιλέχθηκαν και παρουσιάστηκαν δύο διαφορετικές παραλλαγές του αλγορίθμου SRT.
3. Αναπτύχθηκε στα πλαίσια της εργασίας μία ακόμα παραλλαγή του αλγορίθμου SRT.
4. Αναλύθηκαν και συγκρίθηκαν θεωρητικά τα τρία σχέδια.
5. Υλοποιήθηκαν παραμετρικά τα τρία σχέδια.
6. Παρουσιάστηκαν τα αποτελέσματα της σύνθεσης και προσομοίωσης των υλοποιήσεων στα 65 nm. Συγκρίθηκαν πειραματικά ως προς την καθυστέρηση, την επιφάνεια και την κατανάλωση.

1.4 Οργάνωση κειμένου

Στο κεφάλαιο 2 παρουσιάζεται το απαραίτητο θεωρητικό υπόβαθρο για την κατανόηση της διπλωματικής.
Στο κεφάλαιο 3 αναλύονται οι υλοποιήσεις που εξετάστηκαν, με έμφαση στα δομικά τους στοιχεία.
Στο κεφάλαιο 4 παρουσιάζεται μια θεωρητική ανάλυση για τον υπολογισμό της επιφάνειας και της καθυστέρησης των υλοποιήσεων.
Στο κεφάλαιο 5 παρουσιάζονται τα πειραματικά αποτελέσματα για την επιφάνεια, τη καθυστέρηση και τη κατανάλωση της κάθε υλοποίησης. Επίσης, γίνεται σύγκριση των αποτελεσμάτων.
Στο κεφάλαιο 6 γίνεται μια σύνοψη της διπλωματικής μαζί με τα συμπεράσματα και πιθανές προεκτάσεις αυτής.
Στο κεφάλαιο 7 παρατίθεται η βιβλιογραφία που χρησιμοποιήθηκε.

2 ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 ΣΥΣΤΗΜΑΤΑ ΑΝΑΠΑΡΑΣΤΑΣΗΣ ΑΡΙΘΜΩΝ

Ο βασικός σκοπός του κεφαλαίου, είναι να παρουσιασθεί η θεωρία πίσω από τα συστήματα αναπαράστασης αριθμών, με τελικό στόχο την εμβάθυνση στο σύστημα *RSD*. Σημασία έχει δοθεί στη δομή της παρουσίασης, για την εύκολη κατανόηση των εννοιών. Αρχικά, παρουσιάζονται οι βασικές έννοιες με ορισμούς. Γίνεται ο διαχωρισμός μεταξύ αριθμών σταθερής και κινητής υποδιαστολής, δίνονται παραδείγματα συμβατικών και μη συμβατικών συστημάτων.

Το πιο κοινό σύστημα αναπαράστασης αριθμών είναι το δεκαδικό. Είναι το σύστημα που όλοι είμαστε εξοικειωμένοι, και βάσει αυτού μπορούμε να επεκταθούμε και να κατανοήσουμε άλλα συστήματα, αφού σε αυτό ανάγουμε τις αριθμητικές αξίες ώστε να τις αντιληφθούμε. Τα συστήματα αναπαράστασης δεν περιορίζονται στις μορφές που είμαστε εξοικειωμένοι. Γενικά, είναι ένα σύνολο κανόνων, που αντιστοιχίζουν μία ακολουθία ψηφίων σε μία συγκεκριμένη αριθμητική τιμή.

Στα ψηφιακά συστήματα, είναι βολική η χρήση συστημάτων με βάση το δύο και τις δυνάμεις του, αφού κάθε ψηφίο μπορεί να πάρει την τιμή 0 για χαμηλό επίπεδο τάσης και την τιμή 1 για υψηλό επίπεδο τάσης. Επιπλέον, οι αριθμοί αποθηκεύονται σε προκαθορισμένου μήκους *(n)* καταχωρητές. Έτσι, το σύνολο αριθμών που μπορούν να αναπαρασταθούν είναι πεπερασμένο και εξαρτάται από την μορφή αναπαράστασης. Η κατώτατη και ανώτατη δυνατή αναπαριστώμενη τιμή, συμβολίζονται V_{min} και V_{max} αντίστοιχα, οπότε το διάστημα $[V_{min}, V_{max}]$ εκφράζει την περιοχή αναπαριστώμενων αριθμών. Όταν μία αριθμητική πράξη, επιστρέφει αποτέλεσμα εκτός της περιοχής αυτής, τότε το αποτέλεσμα αυτό είναι λανθασμένο και η αριθμητική μονάδα πρέπει να υποδείξει ότι υπάρχει σφάλμα. Πρόκειται για την υπόδειξη *υπερχείλισης (overflow)* ή *υποχείλισης (underflow)*.

2.1.1 Ορισμοί

Μία αριθμητική μονάδα, μπορεί να χαρακτηριστεί σύμφωνα με:

1. Το σύνολο αριθμητικών πράξεων, όπως πρόσθεση, αφαίρεση, πολλαπλασιασμός και διαίρεση.
2. Τους τελεστέους και τα αποτελέσματα των πράξεων.
3. Τις συνθήκες, όπως στην περίπτωση μηδενικού ή αρνητικού αποτελέσματος.
4. Τη διαχείριση ιδιαιτεροτήτων, όπως αποτελέσματα που δεν μπορούν να αναπαρασταθούν. Π.χ. *υπερχείλιση (overflow)*, *υποχείλιση (underflow)* και *όχι-αριθμός (Nan)*.

Οι είσοδοι και έξοδοι των αριθμητικών πράξεων χαρακτηρίζονται από:

1. Ένα πεπερασμένο και διατεταγμένο σύνολο αριθμητικών τιμών $x \in N$.
2. Το *πεδίο τιμών* $V = [V_{min}, V_{max}]$.
3. Ακρίβεια.
4. Σύστημα αναπαράστασης αριθμών.

Σε ένα σύστημα αναπαράστασης αριθμών, οι αριθμοί αναπαρίστανται από μία διατεταγμένη ακολουθία ψηφίων, που ονομάζεται *διάνυσμα ψηφίων* και συμβολίζεται με X. Ισχύει $X \in V$ και γράφουμε, $X = (x_a, \dots, x_i, \dots, x_b)$.

Η αρίθμηση μπορεί να είναι είτε προς τα αριστερά, όπως στους ακέραιους, $X = (x_{n-1}, x_{n-2}, \dots, x_1, x_0)$, είτε προς τα δεξιά, όπως στο δεκαδικό μέρος των αριθμών $X = (x_1, x_2, \dots, x_n)$.

Κάθε ψηφίο x_i έχει ένα σύνολο τιμών D_i. Ισχύει δηλαδή $x_i \in D_i$, όπου στη γενική μορφή:

$$D_i = \{-a, -a+1, \dots, -1, 0, 1, \dots, b-1, b\} \qquad (2.1.1)$$

Σύστημα αναπαράστασης αριθμών, είναι λοιπόν ένα σύνολο κανόνων, που αντιστοιχίζουν ένα *διάνυσμα ψηφίων* X σε μία συγκεκριμένη αριθμητική τιμή. Δύο διαφορετικοί αριθμοί δεν μπορούν να έχουν την ίδια αναπαράσταση *(Unambiguity)*. Δύο ή περισσότερες διαφορετικές αναπαραστάσεις όμως, μπορούν να έχουν την ίδια αριθμητική τιμή. Τα συστήματα στα οποία συμβαίνει αυτό, ονομάζονται συστήματα με περίσσεια *(redundant)*. Τα συστήματα στα οποία κάθε αριθμητική τιμή έχει μοναδική αναπαράσταση, ονομάζονται συστήματα δίχως περίσσεια *(non-redundant)* [3].

Θεωρώντας:

- Έναν ακέραιο *x* που αναπαρίσταται από το *διάνυσμα ψηφίων* $X = (x_{n-1}, \dots, x_1, x_0)$
- Ένα *διάνυσμα συντελεστών βαρύτητας* $W = (w_{n-1}, \dots, w_1, w_0)$
- Ένα *διάνυσμα βάσεων* $R = (r_{n-1}, \dots, r_1, r_0)$

ένα σύστημα αναπαράστασης ορίζεται ως *σταθμισμένο (weighted)*, εάν:

$$X = \sum_{i=0}^{n-1} x_i\, w_i \text{, όπου } \left.\begin{matrix} w_0 = 1 \\ w_i = w_{i-1} r_{i-1} \end{matrix}\right\} \qquad (2.1.2)$$

Όταν σε ένα *σταθμισμένο* σύστημα, το *διάνυσμα βάσεων* R έχει σταθερή τιμή, δηλαδή $r_i = r\ \forall i$, τότε προκύπτει εύκολα $w_i = r^i$ και κατά συνέπεια:

$$X = \sum_{i=0}^{n-1} x_i\, r^i = x_{n-1} r^{n-1} + x_{n-2} r^{n-2} + \cdots + x_1 r + x_0 \qquad (2.1.3)$$

Αποκαλούμε αυτά τα συστήματα *σταθερής-βάσης (fixed-radix).* Για να αποφεύγονται πιθανές συγχύσεις, μπορούμε να υποδηλώσουμε τη *βάση* του συστήματος που χρησιμοποιείται, με έναν δείκτη. Για παράδειγμα, η ακολουθία 1001_{10} αναπαριστά τη δεκαδική αξία 1001, ενώ η ακολουθία 1001_2 αναπαριστά τη δεκαδική αξία 9.

Προφανώς οι *συντελεστές βαρύτητας* $w_i = r^i$ αυξάνουν από δεξιά προς τα αριστερά. Έτσι, το ψηφίο με το χαμηλότερο βάρος ή *λιγότερο σημαντικό ψηφίο,* είναι το x_0 ή γενικά το πρώτο από δεξιά *(Least Significant Bit-LSB).* Αντίστοιχα, το ψηφίο με το μεγαλύτερο βάρος ή *περισσότερο σημαντικό ψηφίο,* είναι το x_{n-1} ή γενικά το πρώτο από αριστερά *(Most Significant Bit-MSB).*

Ορίζεται ως *τυπικό σύνολο ψηφίων (Canonical digit set),* το σύνολο που προκύπτει από τη σχέση (2.1.1) για $a = 0$ και $b = |r_i| - 1$:

$$D_i = \{0, 1, \dots, |r_i| - 1\} \qquad (2.1.4)$$

Τα συστήματα με θετική σταθερή βάση και τυπικό σύνολο ψηφίων χαρακτηρίζονται *συμβατικά.* Τα υπόλοιπα χαρακτηρίζονται *μη συμβατικά.*

2.1.2 Παραδείγματα συμβατικών συστημάτων

Χαρακτηριστικά παραδείγματα *συμβατικών συστημάτων*, είναι το δυαδικό , το δεκαδικό και το δεκαεξαδικό. Το πεδίο τιμών V των συμβατικών συστημάτων είναι προφανώς θετικό. Για να αναπαρασταθούν λοιπόν αρνητικοί αριθμοί, χρησιμοποιούνται τρεις διαφορετικοί τρόποι. Η αναπαράσταση *προσήμου-μέτρου* είναι η πιο κατανοητή καθώς χρησιμοποιείται και στο δεκαδικό σύστημα.

Στο δυαδικό σύστημα, η αναπαράσταση με *συμπλήρωμα ως προς δύο (two's complement)* είναι η πιο διαδεδομένη, αφού απλουστεύει και επιταχύνει τις υλοποιήσεις αριθμητικών μονάδων. Η αναπαράσταση *με συμπλήρωμα ως προς ένα (one's complement),* αποτελεί περισσότερο ενδιάμεσο βήμα για την εύρεση του αντίθετου αριθμού σε *συμπλήρωμα ως προς δύο* [4]. Οι αναπαραστάσεις με *συμπλήρωμα ως προς δύο* και *ως προς ένα*, μπορούν να επεκταθούν σε μεγαλύτερες βάσεις και ονομάζονται *συμπλήρωμα ως προς τη βάση* και *ως προς ελαττωμένη βάση* αντίστοιχα.

Τέλος, άξιο αναφοράς είναι το *πολωμένο σύστημα αναπαράστασης (biased representation)*, αφού χρησιμοποιείται στους αριθμούς *κινητής-υποδιαστολής*.

2.1.3 Παραδείγματα μη συμβατικών συστημάτων

Μερικά παραδείγματα *μη συμβατικών συστημάτων*, είναι:

1. *Μη συμβατικά συστήματα σταθερής βάσης*
2. *Μη συμβατικά συστήματα μεικτής βάσης*
3. *Μη σταθμισμένα συστήματα*

<u>1. *Μη συμβατικά συστήματα σταθερής βάσης*</u>

- Αρνητική *βάση*
  Για παράδειγμα $r = -2$, όπου $x = \sum_{i=0}^{n-1} x_i \, (-2)^i$
  Ενδεικτικά: $1101 = -8 + 4 - 0 + 1 = -3$
  $0101 = -0 + 4 - 0 + 1 = +5$
- Μιγαδική *βάση*
  Για παράδειγμα $r = 2j$, όπου $j = \sqrt{-1}$ με:
  $x_i \in \{0, 1, 2, 3\}$
  $W: -8j - 4 + 2j + 1$
  Ενδεικτικά: $2301 = 2 \cdot (-8j) + 3 \cdot (-4) + 0 \cdot 2j + 1 = 13 - 16j$
- *Μη-τυπικό σύνολο ψηφίων*
  Η γενική μορφή είναι $D_i = \{-a, -a + 1, \dots, -1, 0, 1, \dots, b - 1, b\}$.
  Συνήθως $a = b$, οπότε το σύνολο ψηφίων είναι συμμετρικό.

Λέμε ότι έχουμε *περίσσεια* όταν $a + b + 1 > r$, ενώ ο *παράγοντας περίσσειας* υπολογίζεται:

$$\rho = \frac{a}{r-1}, \rho > \frac{1}{2}$$

2. *Μη συμβατικά συστήματα μεικτής βάσης*

Εδώ υπάρχουν i, j για τα οποία ισχύει γενικά $r_i \neq r_j$, δηλαδή το διάνυσμα βάσεων $R = (r_{n-1}, \dots, r_1, r_0)$, δεν έχει σταθερές τιμές. Παραδείγματα τέτοιων συστημάτων είναι:

- το σύστημα αναπαράστασης του χρόνου, όπου $R = (31, 24, 60, 60)$
- το παραγοντικό σύστημα, όπου
  $\left.\begin{matrix} r_i = i + 2 \\ i = 0, \dots, n-1 \end{matrix}\right\} \rightarrow R = (n+1, n, \dots, 3, 2)$
  $w_i = (i+1)!$
  Οπότε με τυπικό σύνολο ψηφίων $V = [0, (n+1)! - 1]$

3. *Μη σταθμισμένα συστήματα*

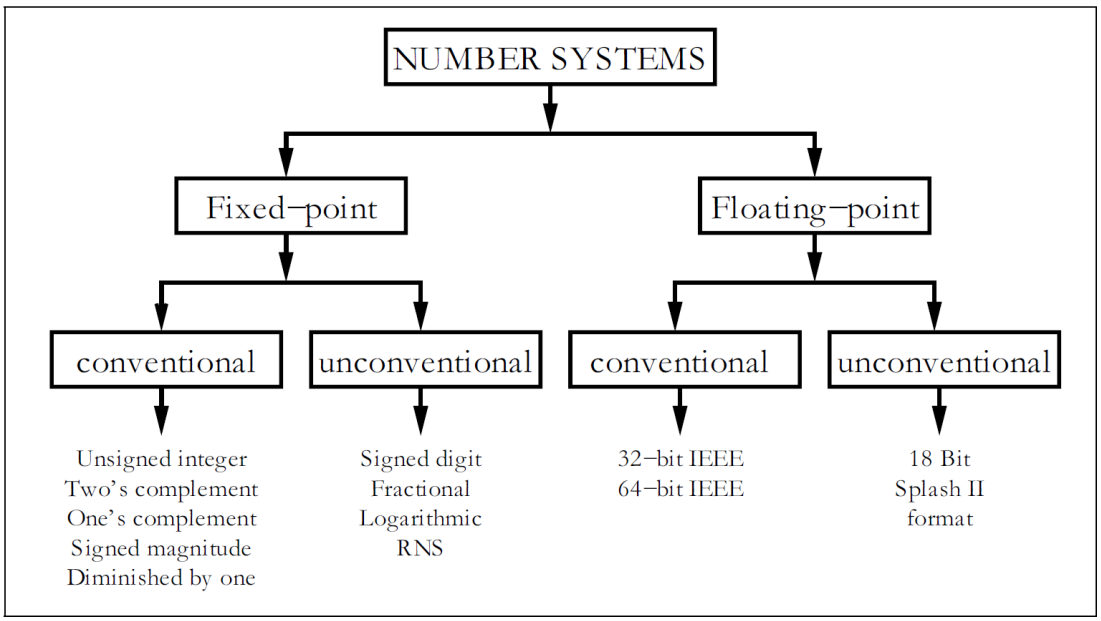
- Αριθμητικό *σύστημα υπολοίπου*(*Residue Number System-RNS*)
  Πρόκειται για ένα περίπλοκο σύστημα, που αναπαριστά έναν μεγάλο ακέραιο με τη χρήση ενός συνόλου μικρότερων ακέραιων, με στόχο την αποδοτικότερη υλοποίηση υπολογισμών. Αναπτύσσεται λεπτομερώς στα [4] [5] [6].

Στην συνέχεια θα επικεντρωθούμε στα συστήματα *σταθερής βάσης*, *συμβατικά* αλλά και *μη συμβατικά*, όπως τα συστήματα με *περίσσεια*.

2.1.4 Αναπαράσταση κλασματικών αριθμών

Όπως αναφέρθηκε, δύο διαφορετικοί αριθμοί δεν μπορούν να έχουν την ίδια αναπαράσταση. Αυτό ισχύει, εάν η θέση της υποδιαστολής είναι δεδομένη, δηλαδή στα συστήματα *σταθερής-υποδιαστολής (fixed-point)*. Υπάρχουν όμως και τα συστήματα *κινητής-υποδιαστολής (floating-point)*, όπου ένα διάνυσμα ψηφίων X, μπορεί να αναπαριστά διαφορετικούς αριθμούς, ανάλογα με τη θέση της υποδιαστολής.

Η επιλογή μεταξύ σταθερής ή κινητής υποδιαστολής, πρέπει να γίνει σε αρχικό στάδιο του σχεδιασμού ενός συστήματος. Χαρακτηριστικά όπως η απαιτούμενη καθυστέρηση και το χαμηλό κόστος, υποδεικνύουν χρήση *σταθερής-υποδιαστολής*. Χαρακτηριστικά όπως η δυναμικότητα του πεδίου τιμών και η ανάγκη κανονικοποίησης των μεταβλητών σε προκαθορισμένο πεδίο τιμών, υποδεικνύουν χρήση *κινητής-υποδιαστολής* [6]. Γενικά τα συστήματα *σταθερής-υποδιαστολής* προτιμούνται για την αναπαράσταση ακεραίων, λόγω απλότητας, όπου η υποδιαστολή θεωρείται ότι βρίσκεται στο δεξί άκρο. Από την άλλη, τα συστήματα κινητής-υποδιαστολής επικρατούν στην αναπαράσταση πραγματικών αριθμών. Στην Εικόνα 1, κατατάσσονται κάποια παραδείγματα συστημάτων αναπαράστασης. Για λόγους κατανόησης, ακολούθως αναπτύσσονται τα συστήματα *σταθερής-υποδιαστολής*. Τα συστήματα κινητής-υποδιαστολής, καλύπτονται ξεχωριστά στην επόμενη ενότητα.



Εικόνα 2.1.1 Παραδείγματα αριθμητικών συστημάτων και κατάταξή τους [6].

2.1.4. *Αριθμοί σταθερής-υποδιαστολής*

Ένα διάνυσμα ψηφίων σε σύστημα *σταθερής-υποδιαστολής*, συνήθως αναπαριστά κάποιον ακέραιο αριθμό, όπου $i = \{0,1,\dots,n-1\}$. Μπορεί όμως να αναπαριστά και δεκαδικούς αριθμούς, αν θεωρήσουμε και αρνητικά i.

Συγκεκριμένα, εάν $i = \{-m,-m+1,\dots,-1,0,1,\dots,k-1\}$, έχουμε την ακολουθία:

$$(x_{k-1}, x_{k-2}, \dots, x_1, x_0, x_{-1}, \dots, x_{-m+1}, x_{-m})$$

και η αριθμητική της αξία υπολογίζεται:

$$X = x_{k-1}r^{k-1} + x_{k-2}r^{k-2} + \cdots + x_1 r + x_0 + x_{-1}r^{-1} + \cdots + x_{-m+1}r^{-m+1} + x_{-m}r^{-m} = \sum_{i=-m}^{k-1} x_i\, r^i. \qquad (2.1.5)$$

Προφανώς, η ύπαρξη υποδιαστολής δεν είναι απαραίτητη, καθώς υπονοείται από τα βάρη των θέσεων. Γνωρίζουμε δηλαδή, ότι βρίσκεται μεταξύ x_0 και x_{-1}. Έχουμε λοιπόν k ψηφία στο ακέραιο μέρος του αριθμού και m ψηφία στο κλασματικό μέρος του αριθμού, με την υποδιαστολή να βρίσκεται σε προκαθορισμένο σημείο. Προφανώς, εάν $k + m = n$:

- για $m = 0$, αναπαρίστανται ακέραιοι, ενώ
- για $k = 0$, αναπαρίστανται καθαρά κλασματικοί αριθμοί

Το βάρος του λιγότερο σημαντικού ψηφίου r^{-m}, καταδεικνύει τη θέση της υποδιαστολής και ονομάζεται *μονάδα στην τελευταία θέση (unit in the last position)*. Γράφουμε $\text{ulp} = r^{-m}$.

2.1.5 Γενική κλάση αριθμητικών συστημάτων σταθερής-βάσης

Τα συστήματα σταθερής-βάσης, χωρίζονται σε:

- *συμβατικά*, που έχουν *θετική βάση* και *τυπικό σύνολο ψηφίων*,
- *μη συμβατικά*, όταν δεν ικανοποιούν μία από τις δύο συνθήκες.

Στη γενική τους μορφή χαρακτηρίζονται από μία τριπλέτα <*n*, *r*, *Λ*>, όπου:

- *r* είναι θετικός αριθμός που υποδηλώνει τη *βάση*,
- *Λ* είναι διάνυσμα μήκους *n* και ισχύει $\Lambda = (\lambda_{n-1}, \lambda_{n-2}, \dots, \lambda_0)$, όπου $\lambda_i \in \{-1,1\}$.

Η τιμή X μίας ακολουθίας $(x_{n-1}, x_{n-2}, \dots, x_1, x_0)$, στο σύστημα <*n*, *r*, *Λ*> υπολογίζεται:

$$X = \sum_{i=0}^{n-1} \lambda_i x_i \mathrm{r}^i \qquad (2.1.6)$$

Ο παράγοντας λ_i, διαλέγει μεταξύ των δύο πιθανών *βαρών* r^i και $-\mathrm{r}^i$, για κάθε θέση i. Βλέπουμε ότι για δεδομένα *n* και *r*, κάθε διαφορετικό διάνυσμα *Λ* συνιστά ένα διαφορετικό αριθμητικό σύστημα [7].

2.1.5. Συμβατικά συστήματα σταθερής-βάσης

Χαρακτηριστικά παραδείγματα συμβατικών συστημάτων όπως είδαμε, είναι:

- Το σύστημα *θετικής-βάσης*, όπου $\lambda_i = 1, \forall i$.
- Το *συμπλήρωμα ως προς δύο* με διάνυσμα $\Lambda = (-1,1,1, \dots,1)$.

2.1.5.. Αναπαράσταση αρνητικών αριθμών

Υπάρχουν δύο διαφορετικοί τρόποι να αναπαραστήσουμε αρνητικούς αριθμούς σταθερής υποδιαστολής. Ο προφανής, *προσημασμένο-μέτρο (signed-magnitude)*, είναι οικείος από το δεκαδικό σύστημα. Ο δεύτερος, αποκαλείται *αναπαράσταση με συμπλήρωμα* και περιλαμβάνει δύο εναλλακτικές.

2.1.5... Αναπαράσταση προσημασμένου-μέτρου

Σε αυτή τη μορφή, διαχωρίζεται ο αριθμός σε δύο μέρη. Στο πρώτο μέρος, ένα ψηφίο υποδεικνύει το πρόσημο του αριθμού. Συνήθως, χρησιμοποιείται το 0 ως θετικό πρόσημο και το $r-1$ ως αρνητικό πρόσημο, όπου *r* η *βάση* του συστήματος. Έτσι για το δυαδικό σύστημα, το 0 υποδηλώνει θετική ποσότητα, ενώ το 1 υποδηλώνει αρνητική.

Στο δεύτερο μέρος εκφράζεται η απόλυτη τιμή του αριθμού. Από τα $n-1$ ψηφία που αναπαριστούν το μέτρο, έστω $k-1$ αφορούν το ακέραιο μέρος και m αφορούν το κλασματικό. Έτσι η μέγιστη αναπαριστώμενη τιμή είναι:

$$X_{max} = \sum_{i=-m}^{k-2} (r-1) \cdot r^i = r^{k-1} - ulp \qquad (2.1.7)$$

Παράδειγμα 2.1.1

Στο δεκαδικό σύστημα για $k-1=4$ και $m=0$, η μέγιστη αναπαριστώμενη τιμή Y είναι

$$Y_{max} = \sum_{i=0}^{3} (10-1) \cdot 10^i = 9 \cdot 10^0 + 9 \cdot 10^1 + 9 \cdot 10^2 + 9 \cdot 10^3$$
$$= 9999 = 10^4 - ulp = 10^4 - 1$$

Προκύπτουν λοιπόν:

- η περιοχή θετικών αριθμών ως $[0, r^{k-1} - ulp]$, και
- η περιοχή αρνητικών αριθμών ως $[-(r^{k-1} - ulp), -0]$

Φαίνεται ότι το 0 έχει διπλή αναπαράσταση, καθώς υπάρχει ένα θετικό και ένα αρνητικό 0. Αυτό είναι ανεπιθύμητο στους υπολογιστές καθώς προσθέτει περιττή πολυπλοκότητα στην υλοποίηση μίας αριθμητικής μονάδας. Το μεγαλύτερο μειονέκτημα όμως, είναι ότι αν μία αριθμητική πράξη εξαρτάται από τα πρόσημα των τελεστέων, απαιτείται επιπλέον έλεγχος και κατά συνέπεια επιφάνεια και καθυστέρηση [7]. Για παράδειγμα, αν προσθέσουμε δύο αριθμούς, τον A και τον $-B$ ($A, B > 0$), πρέπει πρώτα να τους συγκρίνουμε:

- Εάν $A > B$ αφαιρούμε το B από το A και το πρόσημο θα είναι θετικό.
- Εάν $A < B$ αφαιρούμε το A από το B και το πρόσημο θα είναι αρνητικό.

Όλη αυτή η διαδικασία αποφεύγεται στις *αναπαραστάσεις με συμπλήρωμα*.

2.1.5... Αναπαραστάσεις με συμπλήρωμα

Υπάρχουν δύο εναλλακτικές μορφές *αναπαράστασης με συμπλήρωμα*:

1. *Συμπλήρωμα ως προς τη βάση*, ή *συμπλήρωμα ως προς δύο* για το δυαδικό σύστημα.
2. *Συμπλήρωμα ως προς ελαττωμένη-βάση*, ή *συμπλήρωμα ως προς ένα* για το δυαδικό σύστημα.

Και στις δύο μορφές, η αναπαράσταση των θετικών αριθμών είναι ίδια με του προσημασμένου μέτρου. Ένας αρνητικός αριθμός $-X$, αναπαρίσταται με τη χρήση μίας σταθεράς R ως $R-X$. Πρέπει να ικανοποιείται η βασική ιδιότητα $-(-X) = X$, κάτι που ισχύει, αφού το αντίθετο του $(R - X)$ είναι το $R-(R-X) = X$.

Κατ' αυτόν τον τρόπο, όταν θέλουμε να εκτελέσουμε την πράξη $A-B$, έχουμε:

$$A - B = A + (R - B) = R - (B - A) \qquad (2.1.8)$$

- Εάν $A < B$, τότε ο αρνητικός $-(B - A)$ βρίσκεται ήδη στην ίδια αναπαράσταση συμπληρώματος $R - (B - A)$.
- Εάν $A > B$, τότε το σωστό αποτέλεσμα είναι $(A - B)$ και έχουμε $R - (B - A) = R + (A - B)$, όπου το R πρέπει να απορριφθεί.

Η επιλογή του R πρέπει να γίνει λοιπόν με δύο κριτήρια:

1. Να απλουστευθεί ή να αποφευχθεί η διόρθωση.
2. Να μπορεί να υπολογιστεί αποδοτικά και ταχύτατα, το συμπλήρωμα $R-X$ ενός αριθμού X.

Για την εύρεση του R ορίζουμε αρχικά το συμπλήρωμα ενός ψηφίου x_i ως:

$$\overline{x_i} = (r - 1) - x_i \qquad (2.1.9)$$

Με $\bar{X}$ συμβολίζουμε την ακολουθία $(\bar{x}_{k-1}, \bar{x}_{k-2}, \dots, \bar{x}_{-m+1}, \bar{x}_{-m})$. Εάν προσθέσουμε $X + \bar{X}$, για κάθε ψηφίο του αποτελέσματος $\bar{y}_i$, θα ισχύει εξ ορισμού $\bar{y}_i = x_i + \bar{x}_i = (r-1)$. Επομένως θα ισχύει:

$$X + \bar{X} = \sum_{i=-m}^{k-1} (r-1) \cdot r^i = r^k - ulp \qquad (2.1.10)$$

Που συνεπάγεται, ότι προσθέτοντας μία μονάδα στην τελευταία θέση:

$$X + \bar{X} + ulp = r^k \qquad (2.1.11)$$

Σε καταχωρητές σταθερού μήκους *n*, το *πιο σημαντικό ψηφίο* απορρίπτεται, οπότε το τελικό αποτέλεσμα είναι 0. Αναδιατάσσοντας την εξίσωση, έχουμε:

$$r^k - X = \bar{X} + ulp \qquad (2.1.12)$$

Αν επιλέξουμε λοιπόν $R = r^k$ προκύπτει $R - X = r^k - X = \bar{X} + ulp$

Πρόκειται για την αναπαράσταση *συμπληρώματος ως προς τη βάση* και έχει τα πλεονεκτήματα, ότι

- το συμπλήρωμα υπολογίζεται εύκολα,
- σε μία πρόσθεση δύο αριθμών A και $-B$, έχουμε $A - B = A + (R - B) = R + (A - B)$, όπου το R απορρίπτεται αυτόματα, οπότε δεν χρειάζεται διόρθωση.

Με την επιλογή $R = r^k - ulp$ προκύπτει $R - X = (r^k - ulp) - X = \bar{X}$

Πρόκειται για την αναπαράσταση *συμπληρώματος ως προς ελαττωμένη βάση* και έχει το πλεονεκτήματα, ότι το συμπλήρωμα υπολογίζεται ακόμα πιο εύκολα. Μειονεκτεί όμως, καθώς χρειάζεται διόρθωση στην πρόσθεση [7].

2.1.5.... Αναπαράσταση με συμπλήρωμα ως προς δύο

Για το δυαδικό σύστημα και εφαρμόζοντας για $r = 2$ τα ανωτέρω, προκύπτει $R = 2^k$. Η αναπαράσταση αυτή ονομάζεται *συμπλήρωμα ως προς δύο (two's complement-2C)* και είναι η πιο διαδεδομένη στους υπολογιστές λόγω των πλεονεκτημάτων που προσφέρει.

Η περιοχή αναπαριστώμενων τιμών για $k = n$, είναι $[-2^{n-1}, 2^{n-1} - ulp]$, με $ulp = 2^0 = 1$. Παρατηρούμε ότι είναι λίγο ασύμμετρη, καθώς έχει μία αρνητική τιμή παραπάνω. Έχει όμως μοναδική αναπαράσταση για το 0, όπως φαίνεται στον πίνακα 2.1.1.

Η τιμή μίας ακολουθίας $(x_{k-1}, x_{k-2}, \dots, x_1, x_0, x_{-1}, \dots, x_{-m+1}, x_{-m})$ δίνεται από την σχέση:

$$X = -x_{k-1}2^{k-1} + \sum_{i=-m}^{k-2} x_i 2^i \qquad (2.1.13)$$

Στο παράδειγμα 2.1.2, μπορούμε να δούμε πως εκτελείται μία αφαίρεση, προσθέτοντας το *συμπλήρωμα ως προς δύο* του *αφαιρετέου* στον *μειωτέο*.

Παράδειγμα 2.1.2

Έστω οι αριθμοί $5_{10} = 0101_{2C}$ και $2_{10} = 0010_{2C}$. Οι αντίθετοί τους είναι $-5_{10} = 1011_{2C}$ και $-2_{10} = 1110_{2C}$ αντίστοιχα. Η πρόσθεση και οι δύο πιθανές αφαιρέσεις των 5 και 2, φαίνονται στα πινακάκια. Παρατηρούμε ότι το R απορρίπτεται αυτόματα και το αποτέλεσμα είναι σωστό. Βέβαια το πεδίο τιμών είναι $[-8, 7)$ για $n = 4$, οπότε αποτέλεσμα εκτός του πεδίου τιμών θα προκαλούσε *υπερχείλιση*.

R=16	-8	+4	+2	+1	ulp	
	0	1	0	1		5
	1	1	0	1	1	-2
~~1~~	0	0	1	1		3

R=16	-8	+4	+2	+1	ulp	
	0	1	0	1		5
	0	0	1	0	0	+2
~~0~~	0	1	1	1		7

R=16	-8	+4	+2	+1	ulp	
	0	0	1	0		2
	1	0	1	0	1	-5
~~0~~	1	1	0	1		-3

2.1.5.... Αναπαράσταση με συμπλήρωμα ως προς ένα

Για $r = 2$ και $R = 2^k - ulp$, προκύπτει η αναπαράσταση με *συμπλήρωμα ως προς ένα (one's complement-1C)*. Εδώ η περιοχή αναπαριστώμενων τιμών για $k = n$, είναι συμμετρική και ίση με $[-(2^{n-1} - ulp), 2^{n-1} - ulp]$, με $ulp = 2^0 = 1$. Το 0 έχει δύο διαφορετικές αναπαραστάσεις όπως φαίνεται και στον πίνακα 2.1.1.

Η τιμή μίας ακολουθίας $(x_{k-1}, x_{k-2}, \dots, x_1, x_0, x_{-1}, \dots, x_{-m+1}, x_{-m})$ δίνεται από την σχέση:

$$X = -x_{k-1}(2^{k-1} - ulp) + \sum_{i=-m}^{k-2} x_i 2^i \qquad (2.1.14)$$

2.1.5... Πολωμένο σύστημα αναπαράστασης (biased representation)

Το σύστημα αυτό διαφοροποιείται κάπως από τη γενική κλάση. Προκύπτει από το σύστημα *θετικής-βάσης*, πολωμένο κατά μία ποσότητα που ονομάζεται *bias*. Η σταθερά *bias*, επιλέγεται συνήθως στη μέση του πεδίου τιμών, έτσι ώστε το σύστημα να είναι σχεδόν συμμετρικό. Η αριθμητική τιμή x του διανύσματος X, δίνεται από τη σχέση:

$$x = \sum_{i=0}^{n-1} \left(x_i r^i\right) - bias \qquad (2.1.15)$$

Ένα παράδειγμα για $n = 3$, φαίνεται στον πίνακα 2.1.1, ενώ ακολουθεί παράδειγμα υπολογισμού της σταθεράς *bias*.

Παράδειγμα 2.1.3

Για $\mathrm{r} = 2$ και $n = 8$,

$bias = 2^{n-1} - 1 = 2^7 - 1 = 128 - 1 = 127.$

Πίνακας 2.1.1

Δυαδικό	SM	1's C	2's C	bias=3
011	3	3	3	0
010	2	2	2	-1
001	1	1	1	-2
000	0	0	0	-3
111	-3	-0	-1	4
110	-2	-1	-2	3
101	-1	-2	-3	2
100	-0	-3	-4	1

2.1.5. Μη συμβατικά συστήματα σταθερής-βάσης

2.1.5.. Σύστημα αρνητικής-βάσης

Χαρακτηριστικό παράδειγμα *μη συμβατικών συστημάτων σταθερής-βάσης*, είναι το *σύστημα αρνητικής-βάσης*, όπου $\lambda_i = (-1)^i, \forall i$. Έτσι προκύπτει:

$$X = \sum_{i=0}^{n-1} x_i(-\mathrm{r})^i \qquad (2.1.16)$$

Στο σύστημα *αρνητικής-βάσης* δεν χρειάζεται ψηφίο προσήμου για την αναπαράσταση αρνητικών αριθμών. Το πρώτο μη μηδενικό ψηφίο, καθορίζει το πρόσημο του αριθμού. Καθότι θετικοί και αρνητικοί αναπαρίστανται στο ίδιο σύστημα, το πρόσημο δεν είναι απαραίτητο να ελέγχεται στις αριθμητικές πράξεις. Οι υλοποιήσεις του συστήματος όμως είναι πιο περίπλοκες, αφού τα κρατούμενα μπορεί να είναι θετικά η αρνητικά, και το πεδίο τιμών είναι ασύμμετρο όπως φαίνεται στο παράδειγμα 2.1.4.

Παράδειγμα 2.1.4

Έστω δύο αριθμοί μήκους $n = 5$, σε δυαδικό σύστημα *αρνητικής-βάσης* με $r = 2$. Εκτελώντας την πρόσθεσή τους, παρατηρούμε ότι διαδίδονται ταυτόχρονα ένα θετικό και ένα αρνητικό κρατούμενο. Στην πρώτη σειρά φαίνονται τα βάρη κάθε θέσης. Στη θέση με $w_1 = -2$, έχουμε $(-2) + (-2) = -4$, που μπορεί να αποδωθεί ως $-8 + 4 = -4$, με δύο 1 στις δύο επόμενες θέσεις.

Το πεδίο τιμών ενός τέτοιου συστήματος είναι: $[V_{min}, V_{max}] = [01010, 10101] = [-10, 21]$. Παρατηρούμε ότι είναι ασύμμετρο, με δύο φορές περισσότερες θετικές τιμές από αρνητικές. Αυτό ισχύει πάντα για περιττά μήκη *n*, ενώ το αντίθετο ισχύει για ζυγά μήκη *n*.

+16	-8	+4	-2	+1	
1	0	0	1	1	15
0	0	0	1	0	-2
1	1	1	0	1	13

Για τους ανωτέρω λόγους, το συγκεκριμένο σύστημα έχει περιορισμένη χρήση, σε αντίθεση με το *συμπλήρωμα ως προς δύο* και το *σύστημα προσημασμένου-ψηφίου με περίσσεια* που παρουσιάζεται εν συνεχεία και προσφέρει σημαντικά πλεονεκτήματα.

2.1.5.. Συστήματα σταθερής-βάσης με μη-τυπικό σύνολο ψηφίων - Συστήματα με περίσσεια (RSD)

Όπως είδαμε, η γενική μορφή συνόλου ψηφίων είναι:

$$D_i = \{-a, -a+1, \dots, -1, 0, 1, \dots, b-1, b\}$$

Σε ένα *μη-τυπικό σύνολο ψηφίων*, τα a και b μπορούν να έχουν οποιαδήποτε τιμή. Γενικά, λέμε ότι έχουμε *περίσσεια* όταν $a + b + 1 > r$, ή αλλιώς όταν ο αριθμός δυνατών τιμών ενός ψηφίου είναι μεγαλύτερος της βάσης. Σε αυτήν την περίπτωση, κάποιοι αριθμοί μπορούν να αναπαρασταθούν με περισσότερα από ένα διανύσματα ψηφίων. Αυτό μας προσφέρει διάφορα πλεονεκτήματα, όπως θα δούμε στη συνέχεια.

Ιδιαίτερης σημασίας είναι οι υλοποιήσεις με $a = b$, καθώς έχουν επικρατήσει για την γρήγορη εκτέλεση αριθμητικών πράξεων. Το σύνολο ψηφίων είναι συμμετρικό και συνηθίζεται η σημειογραφία

$$D_i = \{\bar{a}, \dots, \bar{1}, 0, 1, \dots, a\}$$

Όπου $\bar{\imath} = -i$. Όμοια με το σύστημα *αρνητικής-βάσης*, το σύστημα αρκεί για την αναπαράσταση και αρνητικών αριθμών, χωρίς τροποποιήσεις. Το πρόσημο του αριθμού καθορίζεται και εδώ, από το πρώτο μη μηδενικό ψηφίο. Ένα τέτοιο σύστημα προκύπτει από τη σχέση 2.1.6 με $\lambda_i = 1, \forall i$ και το ανωτέρω *μη-τυπικό, συμμετρικό σύνολο ψηφίων*. Ονομάζεται σύστημα *προσημασμένου-ψηφίου με περίσσεια ή Redundant Signed-Digit (RSD)* [8].

Στα ψηφιακά συστήματα, η αύξηση της *περίσσειας* μπορεί να φανεί χρήσιμη. Ωστόσο, η επιλογή του συνόλου ψηφίων πρέπει να γίνει προσεκτικά, αφού ένα μεγαλύτερο σύνολο ψηφίων χρειάζεται περισσότερα bits να αναπαρασταθεί. Γενικά προτιμούμε την ελάχιστη *περίσσεια*, που προκύπτει για $2a + 1 \geq r$. Δεδομένου ότι ο ο a επιλέγεται ακέραιος και επιλύοντας, προκύπτει:

$$\left\lceil \frac{r-1}{2} \right\rceil \leq a \leq r - 1$$

, όπου $\lceil x \rceil$ είναι ο μικρότερος ακέραιος για τον οποίο $\lceil x \rceil \geq x$.

- Εάν $a < \left\lceil \frac{r-1}{2} \right\rceil$, τότε το σύστημα είναι *non-redundant*, δηλαδή δεν έχει *περίσσεια*.
- Εάν $a > r - 1$, τότε το σύστημα είναι *over-redundant*.

Ο παράγοντας περίσσειας υπολογίζεται:

$$\rho = \frac{a}{r-1} \qquad (2.1.17)$$

Και η παραπάνω ανίσωση γίνεται $\frac{1}{2} < \rho \leq 1$. Στον πίνακα 2.1.2 μπορούμε να δούμε πως χαρακτηρίζεται ένα σύστημα ανάλογα με τον *παράγοντα περίσσειας*.

Πίνακας 2.1.2 Χαρακτηρισμός περίσσειας συστήματος ανάλογα με τον παράγοντα ρ [9].

Σύνολο ψηφίων $D_{<r,a>}$	a	Παράγοντας Περίσσειας ρ
Incomplete	$< (r-1)/2$	$< 1/2$
Complete άλλά non-redundant	$= (r-1)/2$	$= 1/2$
Redundant	$\geq \lceil r/2 \rceil$	$> 1/2$
Minimally Redundant	$= \lceil r/2 \rceil$	$> 1/2$ και < 1
Maximally Redundant	$= r-1$	$= 1$
Over-redundant	$> r-1$	> 1

Όπως φαίνεται στο παράδειγμα 2.1.5, ένας αριθμός σε σύστημα RSD μπορεί να αναπαρίσταται με διαφορετικούς τρόπους. Η αναπαράσταση με τα λιγότερα μη μηδενικά ψηφία, παρουσιάζει ιδιαίτερο ενδιαφέρον, και ονομάζεται *Κανονική παράσταση Προσημασμένου ψηφίου (Canonical Signed Digit)* [8]. Χρησιμοποιείται έμμεσα στον πολλαπλασιασμό, για να ελαχιστοποιήσει τον απαιτούμενο αριθμό τον προσθέσεων. Για παράδειγμα, αν υποθέσουμε ότι πρέπει να πολλαπλασιάσουμε επί μία μεγάλη ακολουθία από 1, έστω μήκους 9, θα πρέπει να πραγματοποιηθούν 9 προσθέσεις. Αν όμως η ακολουθία αυτή, μετατραπεί σε $100000000\bar{1}$ τότε χρειάζονται μόνο δύο προσθέσεις.

Παράδειγμα 2.1.5

Σε ένα RSD σύστημα με βάση το δύο, το σύνολο ψηφίων θα είναι $D_i = \{\bar{1}, 0, 1\}$. Ο αριθμός $X = 7$, για $n = 4$ μπορεί να αναπαρασταθεί με τέσσερις διαφορετικούς τρόπους, όπως φαίνεται στο πινακάκι.

2^i	8	4	2	1	
X	0	1	1	1	0+4+2+1
	1	-1	1	1	8-4+2+1
	1	0	-1	1	8+0-2+1
	1	0	0	-1	8+0+0-1

Ο αντίθετος $-X = -7$, βρίσκεται εύκολα εάν αλλάξουμε το πρόσημο κάθε μη μηδενικού ψηφίου.

2^i	8	4	2	1	
X	0	-1	-1	-1	-4-2-1
	-1	1	-1	-1	-8+4-2-1
	-1	0	1	-1	-8+2-1
	-1	0	0	1	-8+1

Κατά τον ίδιο τρόπο, υπήρξαν προσπάθειες να αξιοποιηθεί στην πράξη της διαίρεσης. Σκοπός ήταν η ελαχιστοποίηση του αριθμού των προσθέσεων ή αφαιρέσεων, και η

αντικατάστασή τους με μία απλή ολίσθηση. Τα οφέλη δεν είναι ίδια με τον πολλαπλασιασμό όμως, καθώς οι πράξεις γίνονται σε διαδοχικά εξαρτημένα βήματα.

Πάραυτα, η *RSD* αναπαράσταση χρησιμοποιήθηκε για να επιταχύνει με διαφορετικό τρόπο τη διαίρεση. Επιτρέποντας *RSD* αναπαράσταση στα ψηφία του πηλίκου, μπορούμε να αποφύγουμε μία πλήρη σύγκριση μεταξύ *μερικού υπολοίπου* και *διαιρέτη*, να μειώσουμε δηλαδή την ακρίβεια για να κερδίσουμε σε ταχύτητα. Ένα μικρό λάθος, μπορεί να διορθωθεί λόγω της *περίσσειας*. Για παράδειγμα, αν δύο διαδοχικές σωστές τιμές για το *πηλίκο*, είναι 01 και επιλέξουμε 1, μία διόρθωση μπορεί να ακολουθήσει επιλέγοντας $\bar{1}$. Έτσι το αποτέλεσμα θα είναι σωστό, αφού $01 = 1\bar{1}$.

Το πραγματικό κίνητρο για την ανάπτυξη της *RSD* αριθμητικής, ήταν ο περιορισμός της διάδοσης κρατουμένου στην πρόσθεση, με στόχο το αποτέλεσμα να μπορεί να γίνει ανεξάρτητο του μήκους των ορισμάτων της. Αυτό όμως προκύπτει αρκετά ακριβό σαν κύκλωμα [6]. Ωστόσο, σε ενδιάμεσα στάδια πράξεων όπου δεν ενδιαφέρει το ακριβές αποτέλεσμα, το κρατούμενο αποθηκεύεται στην αναπαράσταση αντί να διαδοθεί. Ένας επιπλέον τρόπος να επιταχυνθεί η διαίρεση λοιπόν, είναι η χρήση *RSD* αναπαράστασης για το *μερικό υπόλοιπο*.

Αυτό που πρέπει να αντιμετωπίζεται από την υλοποίηση διαίρεσης, είναι η *υπερχείλιση αναπαράστασης*. Ο αριθμός του παραδείγματος 2.1.5 μπορεί να γραφεί και ως $100\bar{1} = 1\bar{1}00\bar{1} = 1\bar{1}\bar{1}00\bar{1} = 1\bar{1}\bar{1}\bar{1}00\bar{1}$, κ.ο.κ. Μπορεί δηλαδή κατά την ολίσθηση του *μερικού υπολοίπου* να χαθούν *σημαντικά ψηφία*, επομένως απαιτείται ιδιαίτερη προσοχή.

2.2 Αριθμοί κινητής-υποδιαστολής

Σε ένα σύστημα ακεραίων με αναπαράσταση *συμπληρώματος ως προς δύο* και μήκος 32 bits, το εύρος των αναπαριστώμενων αριθμών είναι $[-2147483648, 2147483647]$. Αυτό σημαίνει ότι στις αριθμητικές πράξεις, περιοριζόμαστε σε ένα μικρό σχετικά πεδίο αναπαράστασης των ορισμάτων και των αποτελεσμάτων τους. Το πεδίο τιμών γίνεται ακόμα μικρότερο στους δεκαδικούς αριθμούς. Όσο πιο αριστερά θέτουμε την *σταθερή-υποδιαστολή*, τόσο κερδίζουμε σε ακρίβεια αναπαράστασης των πραγματικών αριθμών, αλλά χάνουμε σε εύρος αναπαράστασης. Εάν για παράδειγμα η υποδιαστολή τεθεί στο αριστερό άκρο και οι αριθμοί έχουν καθαρά κλασματικό μέρος, η μονάδα στην τελευταία θέση γίνεται $ulp = 2^{-32}$ αυξάνοντας την ακρίβεια, αλλά το πεδίο τιμών γίνεται $[-\frac{1}{2}, \frac{1}{2})$.

Τη λύση στο πρόβλημα αυτό δίνουν τα συστήματα *κινητής-υποδιαστολής*. Παρέχουν τη δυνατότητα αναπαράστασης ενός μεγάλου και δυναμικού πεδίου τιμών και σημαντικά πλεονεκτήματα στην εκτέλεση των πράξεων, χρησιμοποιώντας μία μορφή παρόμοια με την επιστημονική σημειογραφία. Συγκεκριμένα, ένας αριθμός κινητής-υποδιαστολής F αποτελείται από δύο μέρη, το *σημαντικό μέρος M (Mantissa)* και τον *εκθέτη E (Exponent)*. Η τιμή του δίνεται από τη σχέση:

$$F = M \cdot \beta^E$$

,όπου β είναι η *βάση* του *εκθέτη*, η οποία δεν συμπεριλαμβάνεται στην αναπαράσταση, καθώς είναι δεδομένη για ένα συγκεκριμένο σύστημα.

Έτσι ένας αριθμός *κινητής-υποδιαστολής* μήκους *n* bits, χωρίζεται στα δύο μέρη *M* και *E*. Είναι εμφανές, ότι η αύξηση του αριθμού bits του *εκθέτη* επιφέρει αύξηση του πεδίου τιμών, ενώ η αύξηση του αριθμού bits του *σημαντικού μέρους* αυξάνει την ακρίβεια.

Οποιαδήποτε μορφή αναπαράστασης χρησιμοποιήσουμε, θα έχουμε 2^n διαφορετικά διανύσματα ψηφίων. Έτσι αυξάνοντας το πεδίο τιμών, αυξάνεται η απόσταση μεταξύ δύο διαδοχικών τιμών και μειώνεται η ακρίβεια. Συνεπώς, ο προσδιορισμός του κατάλληλου αριθμού bits για κάθε ποσότητα εξαρτάται από τον συμβιβασμό μεταξύ πεδίου τιμών και ακρίβειας.

Η αναπαράσταση των *M* και *E*, πρέπει να περιλαμβάνει και αρνητικούς αριθμούς. Για τον *εκθέτη* χρησιμοποιείται συνήθως *πολωμένο σύστημα αναπαράστασης (biased)*. Για την *Mantissa* χρησιμοποιείται αναπαράσταση *προσημασμένου-μέτρου*, ενώ το *μέτρο* είτε θα είναι καθαρό κλάσμα είτε της μορφής $1.f$ (για $r = 2$). Μέχρι το 1980, δεν υπήρχε κάποιο πρότυπο για τους αριθμούς *κινητής-υποδιαστολής*. Κάθε επεξεργαστής χειριζόταν με δικό του τρόπο τους αριθμούς *κινητής-υποδιαστολής*, με αποτέλεσμα την εξάρτηση επιστημονικών εφαρμογών και δεδομένων από την πλατφόρμα. Αναπτύχθηκε λοιπόν το πρότυπο 754 από την IEEE, με το οποίο συμμορφώνονται όλοι οι επεξεργαστές.

2.2.1 Γενική μορφή

Οι αρχικές υλοποιήσεις μελετήθηκαν και συνέδραμαν στην έκδοση του προτύπου. Η γενική μορφή φαίνεται στο σχήμα:

Sign S	Exponent E	Unsigned Significand M

Εικόνα 2.2.1 Γενική μορφή αριθμού κινητής-υποδιαστολής

S είναι το bit *προσήμου*. Ακολουθούν *e* bits για τον *εκθέτη* και *m* bits για το *μη-προσημασμένο σημαντικό μέρος*. Η αριθμητική τιμή ενός τέτοιου αριθμού *κινητής-υποδιαστολής*, δίνεται από τη σχέση:

$$F = (-1)^S \cdot M \cdot \beta^E \qquad (2.2.1)$$

Η μέγιστη τιμή του *σημαντικού μέρους* είναι $M_{max} = 1 - ulp$. Όποτε μία αριθμητική πράξη, έχει σαν αποτέλεσμα *Mantissa* μεγαλύτερη της μέγιστης τιμής, πρέπει αυτή να διορθωθεί με αντίστοιχη ταυτόχρονη διόρθωση του *εκθέτη*, ώστε ο αριθμός να παραμείνει ο ίδιος. Η διόρθωση που γίνεται διέπεται από τη σχέση:

$$M \cdot \beta^E = \frac{M}{\beta} \cdot \beta^{E+1} \qquad (2.2.2)$$

Το β περιορίζεται σε ακέραιες δυνάμεις του δύο, έτσι ώστε η διαίρεση M/β να απαιτεί απλά ολίσθηση. Στα πλαίσια της εργασίας όπως και στο πρότυπο της IEEE, θα μας απασχολήσει για τη *βάση* του *εκθέτη*, η τιμή 2.

Η αναπαράσταση ενός αριθμού *κινητής-υποδιαστολής* δεν είναι μοναδική, αλλά διαφέρει ανάλογα με τη θέση της υποδιαστολής και τον *εκθέτη*. Για παράδειγμα, ο αριθμός 0.1010^{011} μπορεί να γραφεί και στη μορφή 0.0101^{100}. Εάν δοκιμάζαμε να μειώσουμε και άλλο την *Mantissa* και να αυξήσουμε κατά ένα τον *εκθέτη*, θα προέκυπτε ο αριθμός 0.0010^{101}. Όπως φαίνεται απορρίφθηκε ένα *σημαντικό ψηφίο*. Προτιμούμε λοιπόν την παράσταση που ξεκινάει από 1, ώστε να αναπαριστώνται όσο το δυνατόν περισσότερα *σημαντικά ψηφία*. Η παράσταση αυτή αποκαλείται *κανονικοποιημένη (normalized)* και απλοποιεί σημαντικά την εκτέλεση των αριθμητικών πράξεων. Επειδή η *Mantissa*

περιορίζεται σε ένα συγκεκριμένο, στενό πεδίο τιμών, η σύγκριση μεταξύ δύο αριθμών γίνεται εύκολα μέσω των *εκθετών* τους. Προφανώς το 0 δεν μπορεί να αναπαρασταθεί στην *κανονικοποιημένη* μορφή, οπότε θα πρέπει να αναπαρασταθεί με ειδικό τρόπο. Συνήθως, χρησιμοποιούνται $M = 0$ και $E = 0$, έτσι ώστε η αναπαράσταση να είναι παρόμοια με αυτήν της *σταθερής-υποδιαστολής* και να απλοποιείται ο έλεγχος με το 0.

Τέλος ο *εκθέτης* αναπαρίσταται κυρίως στο *πολωμένο σύστημα (biased)*. Εάν το E αναπαριστά τον *εκθέτη* πολωμένο, τότε ο σωστός *εκθέτης* exp βρίσκεται:

$$exp = E - bias \qquad (2.2.3)$$

,όπου $bias$ θετική σταθερά, που επιλέγεται συνήθως στη μέση του πεδίου τιμών, έτσι ώστε το σύστημα να είναι σχεδόν συμμετρικό. Για $\beta = 2$ θα ισχύει

$$0 \leq E \leq 2^e - 1$$

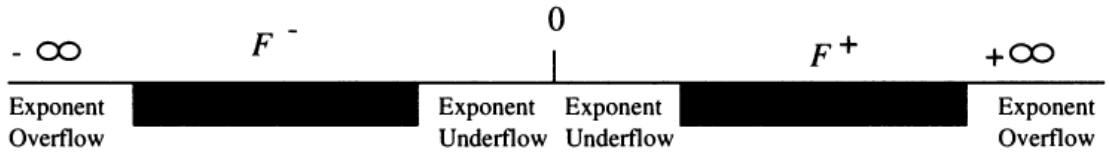
Και επιλέγοντας $bias = \frac{2^e}{2} = 2^{e-1}$, η ανισότητα για τον *εκθέτη* γίνεται

$$-2^{e-1} \leq exp \leq 2^e - 1 - 2^{e-1}$$

$$\text{ή } -2^{e-1} \leq exp \leq 2^{e-1} - 1$$

Κατά αυτόν τον τρόπο, η σύγκριση δύο *εκθετών* μπορεί να γίνει αγνοώντας τα πρόσημά τους. Κατ' επέκταση, δύο ολόκληροι αριθμοί *κινητής-υποδιαστολής* στη συγκεκριμένη διάταξη *S,E,M,* μπορούν να συγκριθούν σαν ακέραιοι σε αναπαράσταση *προσημασμένου-μέτρου*.

Το πεδίο τιμών των *κανονικοποιημένων* αριθμών *κινητής-υποδιαστολής*, είναι ταυτόσημο για θετικούς και αρνητικούς αριθμούς, αφού αναπαριστώνται σε μορφή *προσημασμένου-μέτρου*. Η ελάχιστη δυνατή αναπαριστώμενη απόλυτη τιμή, είναι $M_{min} \cdot \beta^{E_{min}}$, ενώ η μέγιστη δυνατή, είναι $M_{max} \cdot \beta^{E_{max}}$. Καθότι τα αποτελέσματα των πράξεων πρέπει να διατηρούνται σε *κανονικοποιημένη* μορφή, τυχόν υπέρβαση των ορίων του πεδίου τιμών του αριθμού, φαίνεται σε αντίστοιχη υπέρβαση του πεδίου τιμών του *εκθέτη*. Έτσι, αν ένα αποτέλεσμα απαιτεί $E > E_{max}$, λέμε ότι έχουμε *υπερχείλιση εκθέτη (exponent overflow)*. Αντίστοιχα, αν απαιτεί $E < E_{min}$, λέμε ότι έχουμε *υποχείλιση εκθέτη (exponent underflow)*. Σε περίπτωση *υπερχείλισης*, οι περισσότεροι επεξεργαστές αποδίδουν μία ειδική αναπαράσταση για το *άπειρο* στο αποτέλεσμα, άλλοι προκαλούν διακοπή, ενώ άλλοι αποδίδουν τη μεγαλύτερη αναπαριστώμενη τιμή. Σε περίπτωση *υποχείλισης*, αποδίδεται η ειδική αναπαράσταση για το *0*. Σε κάθε περίπτωση, παράγονται *ειδοποιητικά σήματα (flags)*, για την επίγνωση της κατάστασης. Το πεδίο τιμών και η πιθανή υπέρβαση των ορίων του, απεικονίζονται στην εικόνα 2.2.2.



Εικόνα 2.2.2 Πεδίο τιμών του *F* και περιοχές υπερχείλισης-υποχείλισης εκθέτη [7].

2.2.2 Αριθμητικές πράξεις με αριθμούς κινητής-υποδιαστολής

Δεδομένων δύο αριθμών στη γενική μορφή που περιγράψαμε με $\beta = 2$, $F_1 = (-1)^{S_1} \cdot M_1 \cdot 2^{E_1 - bias}$ και $F_2 = (-1)^{S_2} \cdot M_2 \cdot 2^{E_2 - bias}$, μία αριθμητική πράξη μας δίνει αποτέλεσμα $F_3 = (-1)^{S_3} \cdot M_3 \cdot 2^{E_3 - bias}$.

Στην παρούσα ενότητα, θα περιγράψουμε γενικά τις πρόσθεση/αφαίρεση και τον πολλαπλασιασμό δύο αριθμών *κινητής-υποδιαστολής*. Θα εστιάσουμε στη διαίρεση, στο αφιερωμένο σε αυτήν υπο-κεφάλαιο.

Η πρόσθεση και η αφαίρεση είναι πιο δύσκολες σε αριθμούς *κινητής-υποδιαστολής* από ότι σε *σταθερής*. Αυτό οφείλεται στην ύπαρξη των *εκθετών*, οι οποίοι πρέπει να είναι ίσοι για να ξεκινήσει μία πρόσθεση ή αφαίρεση. Συγκρίνουμε λοιπόν τους δύο *εκθέτες* και αυξάνουμε τον *εκθέτη* του μικρότερου κατά $|E_1 - E_2|$, ώστε να γίνει ίσος με τον *εκθέτη* του μεγαλύτερου. Ταυτόχρονα, ολισθαίνουμε την *Mantissa* του μικρότερου $|E_1 - E_2|$ θέσεις προς τα δεξιά. Κατά την ολίσθηση αυτή, μπορεί να απορρίψουμε κάποια λιγότερο *σημαντικά ψηφία*. Αυτό μπορεί να οδηγήσει σε κάποιο σφάλμα, που αποκαλείται *απώλεια σημαντικότητας (loss of significance)* και αντιμετωπίζεται με *φύλαξη μερικών ψηφίων (guard digits)*. Εάν μειώναμε τον *εκθέτη* του μεγαλύτερου, θα έπρεπε να ολισθήσουμε αριστερά την *Mantissa* του χάνοντας κάποια περισσότερο *σημαντικά ψηφία*, ή να αυξήσουμε το μήκος του κυκλώματος κατά άγνωστο εκ των προτέρων αριθμό ψηφίων. Συνοπτικά λοιπόν, ακολουθούμε την εξής διαδικασία:

1. Υπολογίζουμε την διαφορά d των *εκθετών*, $d = |E_1 - E_2|$.
2. Ολισθαίνουμε την *Mantissa* του μικρότερου d θέσεις προς τα δεξιά.
3. Προσθέτουμε τα ευθυγραμμισμένα M_1 και M_2. Θέτουμε τον *εκθέτη* του αποτελέσματος ίσο με τον *εκθέτη* του μεγαλύτερου τελεστέου.
4. Κανονικοποιούμε την *Mantissa* του αποτελέσματος, διορθώνοντας τον *εκθέτη* αν χρειαστεί. Ενδέχεται να προκληθεί *υπερχείλιση* ή *υποχείλιση*, για πρόσθεση ή αφαίρεση αντίστοιχα.
5. Στρογγυλοποιούμε την *Mantissa* του αποτελέσματος, διορθώνοντας τον *εκθέτη* αν χρειαστεί.

Εάν για παράδειγμα $E_1 \geq E_2$, το αποτέλεσμα προκύπτει:

$$F_3 = F_1 \pm F_2 = \left((-1)^{S_1} \cdot M_1 \pm (-1)^{S_2} \cdot M_2 \cdot 2^{-(E_1 - E_2)}\right) \cdot 2^{E_1 - bias} \qquad (2.2.4)$$

Ο πολλαπλασιασμός των δύο αριθμών F_1 και F_2 υλοποιείται απλά όπως φαίνεται επιλύοντας:

$$F_3 = F_1 \cdot F_2$$

$$= (-1)^{S_1} \cdot M_1 \cdot 2^{E_1 - bias} \cdot (-1)^{S_2} \cdot M_2 \cdot 2^{E_2 - bias}$$

$$= (-1)^{S_1 + S_2} \cdot M_1 \cdot M_2 \cdot 2^{E_1 + E_2 - bias - bias}$$

$$= (-1)^{S_3} \cdot M_3 \cdot 2^{E_3 - bias}$$

Εξισώνοντας τις ποσότητες:

$$\left.\begin{array}{c}(-1)^{S_3} = (-1)^{S_1+S_2} \\ M_3 = M_1 \cdot M_2 \\ E_3 = E_1 + E_2 - bias\end{array}\right\} \quad (2.2.5)$$

Οι τρεις αυτοί υπολογισμοί μπορούν να γίνουν παράλληλα. Το πρόσημο S_3 είναι θετικό, εάν S_1 και S_2 είναι ίσα, αλλιώς είναι αρνητικό. Η *Mantissa* M_3 προκύπτει από πολλαπλασιασμό των M_1, M_2, σαν να είναι αριθμοί σταθερής-υποδιαστολής, αλλά πρέπει να προκύψει *κανονικοποιημένη*. Όπως θα δούμε, το πεδίο τιμών για την Mantissa στο IEEE πρότυπο, είναι [1,2) καθώς είναι της μορφής $1.f$. Το γινόμενο $M_1 \cdot M_2$ λοιπόν, θα βρίσκεται στο διάστημα [1,4). Εάν προκύψει τελικά $M_3 \geq 2$, θα χρειαστεί μία δεξιά ολίσθηση για το M_3 και η αύξηση κατά ένα του *εκθέτη* E_3. Εάν ο *εκθέτης* E_3 προκύψει μεγαλύτερος από E_{max}, πρέπει να παραχθεί σήμα *υπερχείλισης*. Εάν προκύψει μικρότερος από E_{min}, πρέπει να παραχθεί σήμα *υποχείλισης*.

2.2.3 IEEE Floating-point standard

Το IEEE Floating-point standard ορίζει τέσσερις μορφές αριθμών *κινητής-υποδιαστολής* [10]. Οι δύο βασικές μορφές, είναι η *μονής ακρίβειας* μήκους 32 bits και η *διπλής ακρίβειας* μήκους 64 bits. Οι άλλες είναι επεκτάσεις των βασικών μορφών, με τουλάχιστον 44 και 80 bits αντίστοιχα και χρησιμοποιούνται σε ενδιάμεσα αποτελέσματα.

Όπως διατυπώθηκε ανωτέρω, η *Mantissa* στο IEEE πρότυπο είναι της μορφής $1.f$, σε αντίθεση με υλοποιήσεις πριν την ανάπτυξη του προτύπου, που ήταν της μορφής $0.1f$. Αυτό παρέχει μεγαλύτερη ακρίβεια, καθώς το πρώτο bit που είναι πάντα 1 λόγω *κανονικοποιημένης* παρουσίασης, παραμένει κρυμμένο.

Μορφή *μονής-ακρίβειας*

Ο συμβιβασμός μεταξύ εύρους αναπαράστασης και ακρίβειας, έγινε όπως φαίνεται στο σχήμα, για το πρότυπο *μονής ακρίβειας*.

S	8 bits - biased exponent E	23 bits - unsigned fraction *f*

Εικόνα 2.2.3 Μορφή μονής ακρίβειας προτύπου IEEE.

Τα 23 bits στη *Mantissa* προσφέρουν ακρίβεια αντίστοιχη των 24 bits, λόγω του κρυμμένου 1. Για τον *εκθέτη* χρησιμοποιείται $bias = \frac{2^e}{2} - 1 = 2^{e-1} - 1 = 127$, δηλαδή μικρότερη κατά ένα από αυτήν που αναφέραμε, ώστε να έχει μεγαλύτερη μέγιστη τιμή ο *εκθέτης* exp. Έτσι η αριθμητική τιμή ενός διανύσματος ψηφίων κατά το IEEE πρότυπο *μονής-ακρίβειας*, βρίσκεται από τη σχέση:

$$F = (-1)^S \cdot 1.f \cdot 2^{E-127} \quad (2.2.6)$$

Οι τιμές $E = 255$ και $E = 0$ δεσμεύονται για ειδικές αναπαραστάσεις, όπως φαίνεται στον πίνακα:

Πίνακας 2.2.1 Ειδικές αναπαραστάσεις του προτύπου IEEE μονής ακρίβειας [41].

Sign S	Exponent e	Mantissa m	Meaning
0/1	0	0	± 0
0/1	0	$\neq 0$	Denormalized: $(-1)^S \cdot 0.f \cdot 2^{-127}$
0/1	$1 < e < 255$	m	Normalized: $(-1)^S \cdot 1.f \cdot 2^{E-127}$
0/1	255	0	$\pm \infty$
-	255	$\neq 0$	NaN

Οι παραστάσεις NaN, χρησιμοποιούνται όταν εκτελούνται:

- Πρόσθεση ή αφαίρεση δύο άπειρων ποσοτήτων, όπως $\infty - \infty$.
- Πολλαπλασιασμός του μηδέν με το άπειρο, $0 \cdot \infty$.
- Διαίρεση μηδενικών ή άπειρων, $0/0$ ή ∞/∞ [6].

Η μέγιστη αναπαριστώμενη τιμή αυξάνεται και από την μορφή της *Mantissa*, αφού βρίσκεται στο πεδίο τιμών $[1,2)$, αντί για $[1/2, 1)$. Ακριβέστερα, το πεδίο τιμών του *σημαντικού μέρους* είναι $[1, 2 - 2^{-23}]$, ενώ το πεδίο τιμών του *εκθέτη* για *κανονικοποιημένη Mantissa* είναι $[-126, 127]$. Έτσι το πεδίο τιμών του F θα είναι:

$$[-F_{max}, -F_{min}] \cup [F_{min}, F_{max}]$$

, όπου:

$$F_{min} = 1.0 \cdot 2^{-126} = 2^{-126}$$

$$F_{max} = (2 - 2^{-23}) \cdot 2^{127} = (1 - 2^{-24}) \cdot 2^{128}$$

Μορφή *διπλής-ακρίβειας*

Στην περίπτωση της *διπλής-ακρίβειας*, με 64 διαθέσιμα bits, ακρίβεια και πεδίο αναπαράστασης αυξάνονται σημαντικά όπως φαίνεται στο σχήμα:

S	11 bits - biased exponent E	52 bits - unsigned fraction f

Εικόνα 2.2.4 Μορφή διπλής ακρίβειας προτύπου IEEE.

Τα 52 bits στη *Mantissa* προσφέρουν ακρίβεια αντίστοιχη των 53 bits, λόγω του κρυμμένου 1. Για τον *εκθέτη* χρησιμοποιείται $bias = \frac{2^e}{2} - 1 = 2^{e-1} - 1 = 1023$. Έτσι η αριθμητική τιμή ενός διανύσματος ψηφίων κατά το IEEE πρότυπο *διπλής-ακρίβειας*, βρίσκεται από τη σχέση:

$$F = (-1)^S \cdot 1.f \cdot 2^{E-1023} \quad (2.2.7)$$

Οι τιμές $E = 2047$ και $E = 0$ δεσμεύονται παρόμοια για ειδικές αναπαραστάσεις, όπως φαίνεται στον πίνακα:

Πίνακας 2.2.2 Ειδικές αναπαραστάσεις του προτύπου IEEE διπλής ακρίβειας [6].

Sign S	Exponent e	Mantissa m	Meaning
0/1	0	0	± 0
0/1	0	$\neq 0$	Denormalized: $(-1)^S \cdot 0.f \cdot 2^{-1023}$
0/1	$1 < e < 2046$	m	Normalized: $(-1)^S \cdot 1.f \cdot 2^{E-1023}$
0/1	2047	0	$\pm\infty$
-	2047	$\neq 0$	NaN

Το πεδίο τιμών του *σημαντικού μέρους* είναι $[1, 2 - 2^{-52}]$, ενώ το πεδίο τιμών του *εκθέτη* για *κανονικοποιημένη Mantissa* είναι $[-1022, 1023]$. Έτσι το πεδίο τιμών του F θα είναι:

$$[-F_{max}, -F_{min}] \cup [F_{min}, F_{max}]$$

, όπου:

$$F_{min} = 1.0 \cdot 2^{-1022} = 2^{-1022}$$

$$F_{max} = (2 - 2^{-52}) \cdot 2^{1023} == (1 - 2^{-53}) \cdot 2^{1024}$$

2.3 Αθροιστές

Στην ανάπτυξη των συστημάτων αναπαράστασης αριθμών, δείξαμε πως με τη χρήση *συμπληρώματος ως προς δύο*, η αφαίρεση μετατρέπεται σε πρόσθεση. Επίσης πρόσθεση και αφαίρεση, χρησιμοποιούνται σε ενδιάμεσα στάδια και των υπόλοιπων πράξεων. Επομένως οι αθροιστές αποτέλεσαν πεδίο έντονης έρευνας, και έχουν αναπτυχθεί πολλοί τρόποι για την αποδοτική τους υλοποίηση. Στα πλαίσια της εργασίας θα παρουσιάσουμε τα βασικά μέρη ενός αθροιστή, καθώς και τέσσερις διαφορετικές υλοποιήσεις.

Στο παράδειγμα 2.3.1 παρουσιάζεται αναλυτικά η εκτέλεση μίας πρόσθεσης στο δυαδικό σύστημα, μεταξύ δύο ορισμάτων των 4 bits, $A + B = \{carry, S\}$, όπως θα επιλύαμε με μολύβι και χαρτί.

Παράδειγμα 2.3.1

Πρόσθεση στο δυαδικό σύστημα, μεταξύ δύο ορισμάτων των 4 bits, $A + B = \{carry, S\}$, όπως θα επιλύαμε με μολύβι και χαρτί:

Βήμα	i	4	3	2	1	0	
	2^i	16	8	4	2	1	dec
	A	0	0	1	0	1	5
	+B	0	0	0	1	1	3
1	carry	0	0	0	1	-	2
	S	-	0	0	0	0	0
2	carry	0	0	1	~~1~~	-	4
	S	-	0	0	0	0	0
3	carry	0	1	~~1~~	~~1~~	-	8
	S	-	0	0	0	0	0
4	carry	0	~~1~~	~~1~~	~~1~~	-	0
	S	-	1	0	0	0	8

1. Στο πρώτο βήμα, προσθέτουμε δύο 1 με βάρος $r^0 = 1$. Το αποτέλεσμα ισούται με 2 στο δεκαδικό σύστημα και άρα με 10 στο δυαδικό. Έτσι, το 0 που αντιστοιχεί στο ίδιο βάρος πάει στο S_0 , ενώ το 1 που αντιστοιχεί στο αμέσως μεγαλύτερο βάρος πάει στο $carry_1$.
2. Έπειτα, έχουμε δύο 1 από B και *κρατούμενο*, οπότε ομοίως $S_1 = 0$ και $carry_2 = 1$. Σημειώνεται εδώ, ότι προσθέσαμε το $carry_1$ οπότε το διαγράφουμε.
3. Ακολουθούν δύο 1 από A και *κρατούμενο*, οπότε και πάλι $S_2 = 0$ και $carry_3 = 1$. Τώρα διαγράφουμε το $carry_2$.
4. Τέλος μόνο το *κρατούμενο* είναι 1, οπότε $S_3 = 1$ και $carry_4 = 0$, διαγράφουμε το $carry_3$ και τελειώνει η πρόσθεση. Το αποτέλεσμα είναι $(carry_4, S_3, S_2, S_1, S_0) = (01000) = 8_2$.

Στο παράδειγμα βλέπουμε ότι υπάρχει ένα ακόμα bit στο αποτέλεσμα που παίρνει την τιμή $carry_4$. Διαφορετικά θα υπήρχε πιθανότητα *υπερχείλισης*. Πρέπει να προσέξουμε επίσης, ότι στην πρώτη πρόσθεση για $i = 0$, προσθέτουμε δύο bits και το αποτέλεσμα είναι επίσης δύο bits. Αυτό υλοποιείται από έναν *ημιαθροιστή (Half Adder)*, τον οποίο και θα περιγράψουμε. Στις επόμενες προσθέσεις, προσθέτουμε 3 bits και το αποτέλεσμα είναι πάλι δύο bits. Αυτό υλοποιείται από έναν *πλήρη αθροιστή (Full Adder)* που επίσης θα περιγράψουμε.

2.3.1 Δομικά στοιχεία

2.3.1. Ημιαθροιστής

Ο *ημιαθροιστής (Half Adder)* είναι ένα κύκλωμα, που προσθέτει δύο bits ίδιου βάρους και παράγει δύο εξόδους. Η μία έξοδος *S (Save)* έχει το ίδιο βάρος με τις εισόδους και η άλλη έξοδος *C (Carry)* το αμέσως μεγαλύτερο. Ο πίνακας αληθείας προσδιορίζει τις επιθυμητές εξόδους του κυκλώματος:

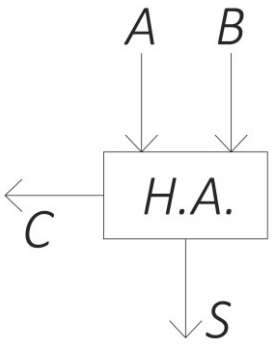
Πίνακας 2.3.1 Πίνακας αληθείας Ημιαθροιστή.

A	B	C	S	dec
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	2

Από τον πίνακα αληθείας εξάγουμε τις εξισώσεις υπολογισμού των εξόδων από τις εισόδους:

$$S = A \oplus B$$
$$C = A \cdot B$$

Και τέλος απεικονίζουμε τον ημιαθροιστή, όπως φαίνεται στην εικόνα 2.3.1.



Εικόνα 2.3.1 Ημιαθροιστής

2.3.1. *Πλήρης αθροιστής*

Όπως φάνηκε στο παράδειγμα 2.3.1, ο *ημιαθροιστής* δεν είναι αρκετός, καθώς υπάρχει η ανάγκη για την άθροιση τριών ψηφίων. Ο *πλήρης αθροιστής (Full Adder)* λοιπόν, αποτελεί το βασικό συστατικό των *αθροιστών*. Προσθέτει τρία ψηφία ίδιου βάρους και παράγει δύο εξόδους. Η μία έξοδος *S (Save)* έχει το ίδιο βάρος με τις εισόδους και η άλλη έξοδος *C (Carry)* το αμέσως μεγαλύτερο. Ο πίνακας αληθείας διαμορφώνεται:

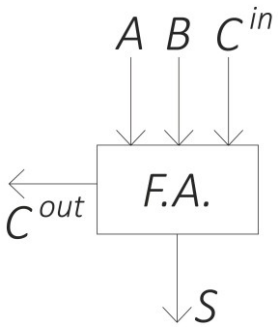
Πίνακας 2.3.2 Πίνακας αληθείας Πλήρους αθροιστή.

A	B	C^{in}	C^{out}	S	dec
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	2
1	0	0	0	1	1
1	0	1	1	0	2
1	1	0	1	0	2
1	1	1	1	1	3

Έτσι εξάγονται οι εξισώσεις υπολογισμού των εξόδων από τις εισόδους:

$$S = A \oplus B \oplus C^{in}$$
$$C^{out} = (A \cdot B) + (A \cdot C^{in}) + (B \cdot C^{in})$$

Και τέλος απεικονίζουμε τον *πλήρη αθροιστή*, όπως φαίνεται στην εικόνα 2.3.2.



Εικόνα 2.3.2 Πλήρης αθροιστής

2.3.2 Αθροιστής διάδοσης κρατουμένου

Η απλούστερη υλοποίηση *αθροιστή*, θυμίζει τον αλγόριθμο, που χρησιμοποιούμε για να εκτελέσουμε μία πρόσθεση με μολύβι και χαρτί. Εκεί προσθέτουμε δύο ευθυγραμμισμένους αριθμούς, οπότε τα ψηφία των δύο ορισμάτων είναι διατεταγμένα σύμφωνα με τα βάρη τους, και προκύπτουν ένα *άθροισμα* με το ίδιο βάρος και ένα *κρατούμενο* με το αμέσως επόμενο βάρος. Το *κρατούμενο* προστίθεται μαζί με τα ψηφία του επόμενου βάρους, παράγονται ένα *άθροισμα* και το επόμενο *κρατούμενο* και η διαδικασία επαναλαμβάνεται σε όλο το μήκος των ορισμάτων. Κατά αυτόν τον τρόπο, το *κρατούμενο* διαδίδεται αλυσιδωτά, και έτσι η υλοποίηση αυτή ονομάζεται *αθροιστής διάδοσης κρατουμένου (Carry Propagate Adder-CPA ή Ripple-Carry Adder-RCA)*.

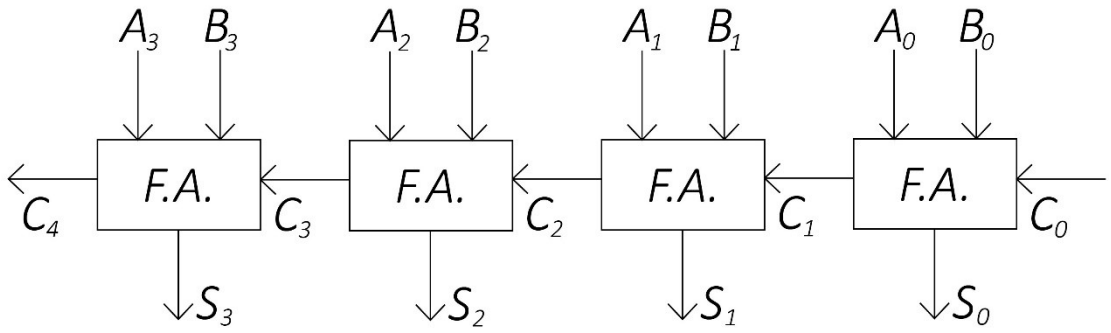
Ένας *αθροιστής διάδοσης κρατουμένου*, συνίσταται από τα ελάχιστα δομικά στοιχεία. Χρησιμοποιείται μία αλυσίδα από *πλήρεις αθροιστές*, με εισόδους τα ισοβαρή ψηφία των δύο ορισμάτων, και το *κρατούμενο* εξόδου του προηγούμενου *πλήρους αθροιστή*. Ισχύει δηλαδή για κάθε θέση i:

$$\left.\begin{matrix} C_0^{in} = C^{in} \\ C_i^{in} = C_{i-1}^{out},\ \ i = \{1,2,\dots,n\} \end{matrix}\right\}$$

Συνηθίζεται όμως να χρησιμοποιείται η σεμειογραφία:

$$\left.\begin{matrix} C_0 = C^{in} \\ C_{i+1} = C_{i+1}^{in} = C_i^{out},\ \ i = \{0,1,2,\dots,n\} \end{matrix}\right\} \qquad (2.3.1)$$

Το αποτέλεσμα μίας πράξης μήκους *n*, είναι το διάνυσμα ψηφίων $(C_n, S_{n-1}, \dots, S_1, S_0)$, ενώ για μήκος 4 bits όπως στο παράδειγμα 2.3.1, το κύκλωμα παρουσιάζεται στην εικόνα 2.3.3.



Εικόνα 2.3.3 Αθροιστής 4-bit διάδοσης κρατουμένου.

Πρόκειται για ένα *συνδυαστικό* κύκλωμα, και μπορούμε να παρατηρήσουμε τη διάδοση των δεδομένων μεταξύ των βαθμίδων, όπως περιγράψαμε στο παράδειγμα 2.3.1. Αρχικά οι τιμές των *κρατουμένων* $(C_n, C_{n-1}, \dots, C_2, C_1)$ είναι μηδενικές, και τα *Save* ψηφία $(S_{n-1}, \dots, S_2, S_1)$ λαμβάνουν τις αντίστοιχες τιμές. Όσο διαδίδεται το *κρατούμενο*, τα ψηφία *S* διορθώνονται μέχρι να καταλήξει το κύκλωμα στο σωστό αποτέλεσμα.

Εάν ο *αθροιστής* χρησιμοποιείται για απλή πρόσθεση, τότε το C_0 είναι πάντα 0, και ένας *ημιαθροιστής* είναι αρκετός στην πρώτη θέση. Μία αφαίρεση όμως μετατρέπεται σε πρόσθεση, όταν χρησιμοποιούμε *συμπλήρωμα ως προς δύο*. Έτσι αν πρέπει να αφαιρέσουμε δύο αριθμούς A, B, βρίσκουμε το *συμπλήρωμα ως προς δύο* του B και το αθροίζουμε στο A. Αντιστρέφουμε δηλαδή τα ψηφία του B και προσθέτουμε μία *ulp* μέσω του C_0. Επιπλέον για την σωστή λειτουργία του κυκλώματος, με ορίσματα σε *συμπλήρωμα ως προς δύο*, πρέπει να γίνεται έλεγχος για *υπερχείλιση*, ή απλούστερα να αντιστραφούν τα ψηφία αρνητικής αξίας A^{n-1}, B^{n-1}, C_n.

Ο *αθροιστής διάδοσης κρατουμένου* μπορεί να προσθέσει/αφαιρέσει δύο ορίσματα, αλλά και να μετατρέψει έναν αριθμό από την *carry-save* μορφή που θα δούμε, σε *συμπλήρωμα ως προς δύο*. Έχει πολύ χαμηλό κόστος, αλλά η *διάδοση κρατουμένου* κατά μήκος *n πλήρων αθροιστών*, τον καθιστά μη αποδοτικό. Προκειμένου να ελαχιστοποιηθεί η *διάδοση κρατουμένου*, έχουν αναπτυχθεί διάφορες μέθοδοι *πρόβλεψης* αλλά και *αποθήκευσης κρατουμένου* στην αναπαράσταση. Οι μέθοδοι *αποθήκευσης κρατουμένου*, χρησιμοποιούνται σε ενδιάμεσα στάδια πολυπλοκότερων πράξεων. Δεν δίνουν αποτέλεσμα σε τελική μορφή, αλλά εξαλείφουν την *διάδοση κρατουμένου*.

2.3.3 Αθροιστής πρόβλεψης κρατουμένου

Οι *αθροιστές πρόβλεψης κρατουμένου (Carry-Lookahead Adder-CLA)*, έχουν διερευνηθεί εκτεταμένα. Η κύρια ιδέα, είναι να παραχθούν όλα τα *κρατούμενα* παράλληλα με την είσοδο των ορισμάτων στο κύκλωμα, ώστε να μην εισάγεται η καθυστέρηση της *διάδοσης*. Κάτι τέτοιο μπορεί να υλοποιηθεί, εφόσον ένα κρατούμενο C_i εξαρτάται από όλες τις προηγούμενες εισόδους $(A_{i-1}, \dots, A_1, A_0)$, $(B_{i-1}, \dots, B_1, B_0)$, C^{in}. Μία τέτοια εξάρτηση μπορεί να κοστίζει αρκετά για μεγάλα i, οπότε έχουν αναπτυχθεί αρκετές διαφοροποιήσεις του βασικού σχήματος.

Η *πρόβλεψη κρατουμένου* βασίζεται στην παρατήρηση, ότι γνωρίζοντας τα δύο ψηφία μίας θέσης, A_i και B_i, μπορούμε να αποφασίσουμε αν θα *γεννηθεί κρατούμενο* ή αν θα

διαδοθεί το *κρατούμενο* από δεξιά. Από τον πίνακα αληθείας του πλήρη αθροιστή, εξαγάγαμε τις σχέσεις:

$$S = A \oplus B \oplus C^{in}$$
$$C^{out} = (A \cdot B) + (A \cdot C^{in}) + (B \cdot C^{in})$$

Διαμορφώνοντας τις σχέσεις για την θέση i:

$$S_i = A_i \oplus B_i \oplus C_i \quad (2.3.2)$$
$$C_{i+1} = (A_i \cdot B_i) + (A_i \cdot C_i) + (B_i \cdot C_i) = (A_i \cdot B_i) + C_i \cdot (A_i + B_i) \quad (2.3.3)$$

Παρατηρώντας την εξίσωση, μπορούμε να δούμε ότι *γεννάται κρατούμενο (Generate)* όταν $A_i \cdot B_i = 1$, ενώ *διαδίδεται* το *κρατούμενο* από δεξιά *(Propagate)*, όταν $A_i + B_i = 1$. Θέτοντας λοιπόν $G_i = A_i \cdot B_i$ και $P_i = A_i + B_i$, προκύπτει:

$$C_{i+1} = G_i + C_i \cdot P_i$$

Μάλιστα, αν θέσουμε τελικά $P_i = A_i \oplus B_i$, η ανωτέρω σχέση δεν αλλάζει, καθώς:

$$(A_i \cdot B_i) + C_i \cdot (A_i + B_i) = (A_i \cdot B_i) + C_i \cdot (A_i \oplus B_i)$$

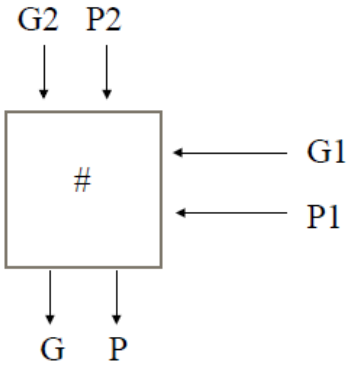
Έτσι τα σήματα P_i, G_i παράγονται με τη χρήση *ημιαθροιστών*, για εισόδους A_i, B_i. Επιπλέον, καταλήγουμε στις πιο βολικές εξισώσεις:

$$S_i = P_i \oplus C_i \quad (2.3.4)$$
$$C_{i+1} = G_i + C_i \cdot P_i \quad (2.3.5)$$

Αντικαθιστώντας διαδοχικά $C_i = G_{i-1} + C_{i-1} \cdot P_{i-1}$ στη σχέση 2.3.5, προκύπτει:

$$C_{i+1} = G_i + G_{i-1}P_i + G_{i-2}P_{i-1}P_i + C_{i-2}P_{i-2}P_{i-1}P_i = \cdots$$
$$= G_i + G_{i-1}P_i + G_{i-2}P_{i-1}P_i + \cdots + C_0P_0P_1 \cdots P_i$$

Σύμφωνα με τη σχέση αυτή, οποιοδήποτε *κρατούμενο*, μπορεί να *προβλεφθεί* με τη χρήση του κυκλώματος *πρόβλεψης κρατουμένου* (τελεστής #), που απεικονίζεται στην εικόνα 2.3.4.

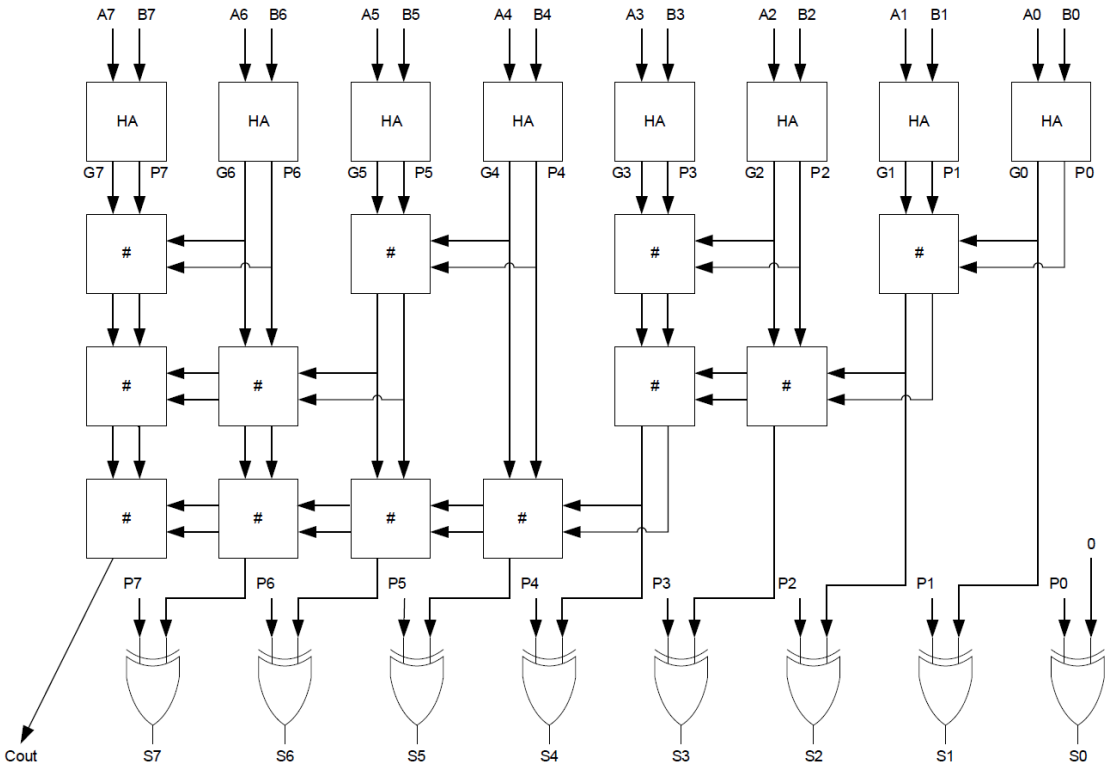


Εικόνα 2.3.4 Κύκλωμα πρόβλεψης κρατουμένου [8].

Οι έξοδοι βρίσκονται από τις σχέσεις:

$$G = G_2 + (G_1 \cdot P_2) \quad (2.3.6)$$
$$P = P_1 \cdot P_2 \quad (2.3.7)$$

Συνοψίζοντας, αν χρησιμοποιήσουμε *ημιαθροιστές* με εισόδους τα αρχικά ορίσματα, παράγονται τα σήματα P και G για κάθε θέση i. Κατόπιν, με κατάλληλη χρήση τελεστών #, μπορούμε να *προβλέψουμε* οποιοδήποτε *κρατούμενο*. Ένας τρόπος υλοποίησης *CLA* 8 bits φαίνεται στην εικόνα 2.3.5, ενώ υπάρχουν πολλοί ακομά που δεν θα μας απασχολήσουν στα πλαίσια της εργασίας. Τα τελικά ψηφία *Save*, προκύπτουν από μία απλή *XOR* όπως φαίνεται και από τη σχέση 2.3.4.



Εικόνα 2.3.5 Αθροιστής 8-bit πρόβλεψης κρατουμένου.

Όσον αφορά τη χρήση του *αθροιστή πρόβλεψης κρατουμένου*, ισχύουν τα ίδια με τον *αθροιστή διάδοσης κρατουμένου*. Μπορεί δηλαδή, να χρησιμοποιηθεί για ορίσματα σε μορφή *συμπληρώματος ως προς δύο* κατά τον ίδιο τρόπο, και να κάνει αφαίρεση συμπληρώνοντας το δεύτερο όρισμα και προσθέτοντας μία *ulp* μέσω του C_0. Μπορεί επίσης, να μετατρέψει έναν αριθμό από την *carry-save* μορφή, σε *συμπλήρωμα ως προς δύο*.

Ο εν λόγω *αθροιστής προβλέπει* τα *κρατούμενα*, με αποτέλεσμα να είναι ταχύτερος από τον *αθροιστή διάδοσης κρατουμένου*. Συγκεκριμένα, καθυστερεί όσο ένας *ημιαθροιστής* συν logn κυκλώματα # συν μια πύλη *XOR*. Το μειονέκτημα του είναι η μεγαλύτερη επιφάνεια, η οποία αυξάνει πολύ για μεγάλα μήκη *n*. Συγκεκριμένα, απαιτούνται $n \cdot log(n)/2$ τελεστές #. Παρόλα αυτά η μείωση της καθυστέρησης είναι σημαντική, και συνήθως αξίζει το επιπλέον κόστος.

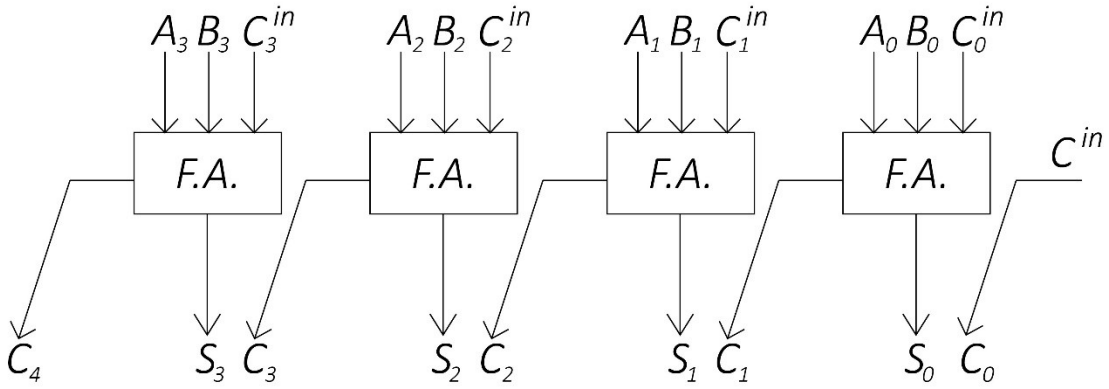
2.3.4 Αθροιστής αποθήκευσης κρατουμένου

Είναι σύνηθες, να μην μας ενδιαφέρει το ακριβές αποτέλεσμα μίας πρόσθεσης, και έτσι να μην υπάρχει η ανάγκη *διάδοσης* του *κρατουμένου*. Όποτε για παράδειγμα πρέπει να προσθέσουμε περισσότερα από δύο ορίσματα, ή να εκτελέσουμε προσθέσεις εντός άλλων πράξεων, θα ήταν άσκοπο να περιμένουμε τη *διάδοση* του *κρατουμένου*, όταν είναι αρκετή μία εκτίμηση του αποτελέσματος. Αυτό επιτυγχάνεται, *αποθηκεύοντας* το *κρατούμενο* στην αναπαράσταση της εξόδου. Μία τέτοια αναπαράσταση είναι η *RSD*, που χρησιμοποιείται στην παρούσα εργασία.

Ένας *αθροιστής αποθήκευσης κρατουμένου (Carry-Save Adder-CSA)* είναι παρόμοιος με τον *αθροιστή διάδοσης κρατουμένου*. Αποτελείται και αυτός από *n πλήρεις αθροιστές*, όπου *n* το μήκος των ορισμάτων. Διαφέρει όμως, καθώς αθροίζει τρία ορίσματα και δίνει στην έξοδο το αποτέλεσμα σε διπλή αναπαράσταση, ή όπως λέμε αναπαράσταση *αποθήκευσης κρατουμένου*. Κάθε *πλήρης αθροιστής* λοιπόν, λαμβάνει τρία ψηφία στην είσοδο και εξάγει δύο ψηφία, και έτσι ονομάζεται *3-2 συμπιεστής*. Το ψηφίο *Save* μίας βαθμίδας έχει το ίδιο βάρος με αυτήν, ενώ το ψηφίο *Carry* έχει το βάρος της αμέσως μεγαλύτερης βαθμίδας. Ισχύει δηλαδή:

$$\left.\begin{aligned} C_0 &= C^{in} \\ A_i + B_i + C_i^{in} &= 2C_{i+1} + S_i, \quad i = \{0,1,2,\dots,n\} \end{aligned}\right\} \qquad (2.3.8)$$

Στην εικόνα 2.3.6 απεικονίζεται *αθροιστής 4-bit αποθήκευσης κρατουμένου*.



Εικόνα 2.3.6 Αθροιστής 4-bit αποθήκευσης κρατουμένου.

Ειδικότερα για την *RSD* αναπαράσταση, οι *πλήρεις αθροιστές* μπορούν να έχουν και αρνητικές εισόδους ή εξόδους. Έτσι μπορούν να αθροίζονται δύο αριθμοί, ένας σε *RSD* αναπαράσταση και ένας σε *συμπλήρωμα ως προς δύο*, με αποτέλεσμα έναν *RSD* αριθμό.

Ο *αθροιστής αποθήκευσης κρατουμένου* έχει ίδια δομή και επιφάνεια με τον *αθροιστή διάδοσης κρατουμένου*. Η *καθυστέρησή* του όμως, είναι ανεξάρτητη του μήκους *n*, και ισούται με ένα επίπεδο *πλήρους αθροιστή*. Το μόνο μειονέκτημά του, είναι ότι το αποτέλεσμα χρειάζεται περαιτέρω επεξεργασία. Χρησιμοποιείται όμως ευρύτατα, σε ενδιάμεσα αποτελέσματα πολλαπλασιασμών, διαιρέσεων και προσθέσεων πολλαπλών ορισμάτων.

2.3.5 Αθροιστής 3 σε 1

Στα πλαίσια της εργασίας, χρησιμοποιείται αθροιστής ο οποίος αθροίζει τρία ορίσματα και επιστρέφει ένα. Πρόκειται προφανώς για συνδυασμό, ενός *αθροιστή αποθήκευσης κρατουμένου*, που ακολουθείται από έναν *αθροιστή διάδοσης κρατουμένου*. Πιο συγκεκριμένα, αθροίζει έναν *RSD* αριθμό και έναν σε *συμπλήρωμα ως προς δύο*, και επιστρέφει αποτέλεσμα σε *συμπλήρωμα ως προς δύο*. Παρουσιάζεται σχηματικά στην εικόνα 3.3.5.

2.4 Διαίρεση

2.4.1 Η διαίρεση σαν μαθηματική πράξη

Η βασική αριθμητική πράξη της διαίρεσης είναι η "αντίστροφη" του πολλαπλασιασμού, καθώς η διαίρεση ενός αριθμού X με έναν αριθμό D, είναι ο πολλαπλασιασμός του X με τον αντίστροφο του D. Αυτό φαίνεται παρακάτω:

$$\frac{X}{D} = X \cdot \frac{1}{D} = Q$$

,όπου X είναι ο *διαιρετέος (Dividend)*

,D είναι ο *διαιρέτης (Divisor)* διάφορος του 0 για να ορίζεται η διαίρεση

και Q είναι το *πηλίκο (Quotient)*

Ιδιότητες

Η διαίρεση είναι δεξιά-επιμεριστική ως προς την πρόσθεση και την αφαίρεση. Δηλαδή ισχύει

$$\frac{a+b}{c} = (a+b) \div c = \frac{a}{c} + \frac{b}{c}$$

Δεν είναι όμως αριστερά-επιμεριστική

$$\frac{a}{b+c} = a \div (b+c) \neq \frac{a}{b} + \frac{a}{c}$$

2.4.2 Οι δυσκολίες υλοποίησης της διαίρεσης σε Hardware

Από τις τέσσερις βασικές αριθμητικές πράξεις, η διαίρεση είναι η δυσκολότερη να υλοποιηθεί σε hardware, αφού για την εύρεση αποτελέσματος, πρέπει να ακολουθηθούν κατά βάση διαδοχικά εξαρτημένα βήματα. Παραλληλισμός μπορεί να υπάρξει, αλλά οι ιδιαιτερότητες μίας τέτοιας υλοποίησης, την καθιστούν ωφέλιμη υπό αρκετές προϋποθέσεις [11]. Η διαίρεση καθίσταται λοιπόν αργή σε σχέση με τις άλλες βασικές πράξεις, και έτσι το πεδίο σχεδίασης γρήγορων και αποδοτικών κυκλωμάτων διαίρεσης αποτελεί πρόκληση [12]. Αντίθετα με την πρόσθεση, την αφαίρεση και τον πολλαπλασιασμό, το σύνολο των ακεραίων δεν είναι *κλειστό* ως προς τη διαίρεση. Η διαίρεση δυο ακεραίων μπορεί να έχει *υπόλοιπο*, όπως ορίζει η *Ευκλείδια διαίρεση*:

Δεδομένων δύο ακεραίων, *X* ο *διαιρετέος* και *D* ο *διαιρέτης*, όπου $D \neq 0$, υπάρχουν μοναδικοί ακέραιοι, *Q* το *πηλίκο (Quotient)* και *R* το *υπόλοιπο (Remainder)*, ώστε:

$$\begin{gathered} \mathrm{X} = \mathrm{D} \cdot \mathrm{Q} + \mathrm{R}\,, \\ 0 \leq \mathrm{R} < |\mathrm{D}|, \end{gathered} \qquad (2.4.1)$$

όπου $|\mathrm{D}|$ η απόλυτη τιμή του διαιρέτη.

Για να διαιρεθεί και το *υπόλοιπο*, το σύστημα αριθμών επεκτείνεται ώστε να περιλαμβάνει κλάσματα, ή ρητούς αριθμούς όπως λέγονται γενικότερα [13]. Το αποτέλεσμα λοιπόν της διαίρεσης, είναι στη γενικότητά του ένας ρητός αριθμός, που δεν μπορεί να αναπαρασταθεί πάντα με ακρίβεια με έναν προκαθορισμένο αριθμό ψηφίων. Επομένως η έξοδος ενός κυκλώματος διαίρεσης, είτε θα είναι μία προσέγγιση της απάντησης, είτε μία έκφραση διαμορφωμένη κατά την *Ευκλείδια διαίρεση*.

Κατά τα ανωτέρω, υπάρχει μία ιδιαιτερότητα στον υπολογισμό του *υπολοίπου*. Εάν υλοποιούσαμε μία *ακέραια διαίρεση*, το *υπόλοιπο* θα προέκυπτε απλά. Εάν όμως μας ενδιαφέρει ο υπολογισμός *πραγματικού πηλίκου*, υπάρχουν δύο περιπτώσεις. Στην περίπτωση που χρησιμοποιούμε αριθμούς *σταθερής-υποδιαστολής*, το υπόλοιπο υπολογίζεται επίσης απλά. Στην πραγματικότητα όμως, συνηθίζεται να αναπαριστούμε τους πραγματικούς αριθμούς σε μορφή *κινητής-υποδιαστολής*, ώστε να έχουμε δυναμικό εύρος τιμών και αποδεκτή ακρίβεια. Στην περίπτωση αυτή, το *πηλίκο* υπολογίζεται παρόμοια, αλλά δεν ισχύει το ίδιο για το *υπόλοιπο*. Για το *πηλίκο*, διαιρούμε τα *σημαντικά* μέρη των αριθμών και παράλληλα υπολογίζουμε τον *εκθέτη*, όπως φαίνεται από την επίλυση:

$$Q = \frac{X}{D} = \frac{(-1)^{S_X} \cdot M_X \cdot 2^{E_X - bias}}{(-1)^{S_D} \cdot M_D \cdot 2^{E_D - bias}} = (-1)^{S_X - S_D} \cdot \frac{M_X}{M_D} \cdot 2^{E_X - E_D} = (-1)^{S_Q} \cdot M_Q \cdot 2^{E_Q - bias}$$

Εξισώνοντας τις ποσότητες:

$$\left.\begin{aligned} (-1)^{S_Q} &= (-1)^{S_X - S_D} \\ M_Q &= \frac{M_X}{M_D} \\ E_Q &= E_X - E_D + bias \end{aligned}\right\} \qquad (2.4.2)$$

Το πρόσημο S_Q είναι θετικό εάν S_X και S_D είναι ίσα, αλλιώς είναι αρνητικό. Η *Mantissa* M_Q προκύπτει από διαίρεση των M_X, M_D, που είναι *κανονικοποιημένες* ποσότητες. Σύμφωνα με το IEEE πρότυπο, η *Mantissa* είναι της μορφής $1.f$ με πεδίο τιμών $[1,2)$. Έτσι απλοποιείται η εκτέλεση, καθώς τα ορίσματα είναι ευθυγραμμισμένα. Μάλιστα, ολισθαίνοντας τα δύο ορίσματα μία θέση δεξιά, κανονικοποιούνται στη μορφή $0.1f$, με πεδίο τιμών $[1/2\,, 1)$, ενώ το αποτέλεσμα δεν αλλάζει. Θα φανεί στη συνέχεια, ότι η *κανονικοποίηση* αυτή για τον *διαιρέτη* διευκολύνει περαιτέρω τη διαίρεση. Το αποτέλεσμα M_Q, θα βρίσκεται στο πεδίο τιμών $(1/2\,, 2)$. Συνεπώς, εάν προκύψει τελικά $M_Q < 1$, θα χρειαστεί μία αριστερή ολίσθηση για το M_Q και η μείωση κατά ένα του *εκθέτη* E_Q. Εάν ο *εκθέτης* E_Q προκύψει μεγαλύτερος από E_{max}, πρέπει να παραχθεί σήμα *υπερχείλισης*, ενώ εάν προκύψει μικρότερος από E_{min}, πρέπει να παραχθεί σήμα *υποχείλισης*.

Όσον αφορά το *υπόλοιπο*, δεν υπολογίζεται άμεσα, αφού προκύπτει από τη διαίρεση των *σημαντικών μερών* και όχι ολόκληρων των αριθμών. Έτσι σε πολλές υλοποιήσεις ο υπολογισμός του *υπολοίπου* παραλείπεται. Για εφαρμογές που είναι απαραίτητος όμως, μπορεί να υπολογιστεί από τη σχέση $X\ REM\ D = X - D \cdot Int(Q)$, όπου $Int(Q)$ είναι το *πηλίκο* που έχει μετατραπεί σε ακέραιο. Η μετατροπή μπορεί να γίνει είτε με απόρριψη του κλασματικού μέρους του Q, είτε με στρογγυλοποίηση. Ωστόσο, το *πηλίκο* υπολογίζεται σε αναπαράσταση *κινητής-υποδιαστολής*, και ο εκθέτης του μπορεί να πάρει την μέγιστη

τιμή $E_{max} - E_{min}$. Έτσι, ο $Int(Q)$ υπολογίζεται με $E_X - E_D$ ολισθήσεις στο *σημαντικό μέρος* του Q, και κατόπιν απόρριψη του κλασματικού μέρους ή στρογγυλοποίηση. Στην συνέχεια πολλαπλασιάζεται με τον *διαιρέτη*, και το γινόμενο αφαιρείται από τον *διαιρετέο* δίνοντας το *υπόλοιπο* [7].

2.4.3 Γιατί είναι σημαντική η διαίρεση

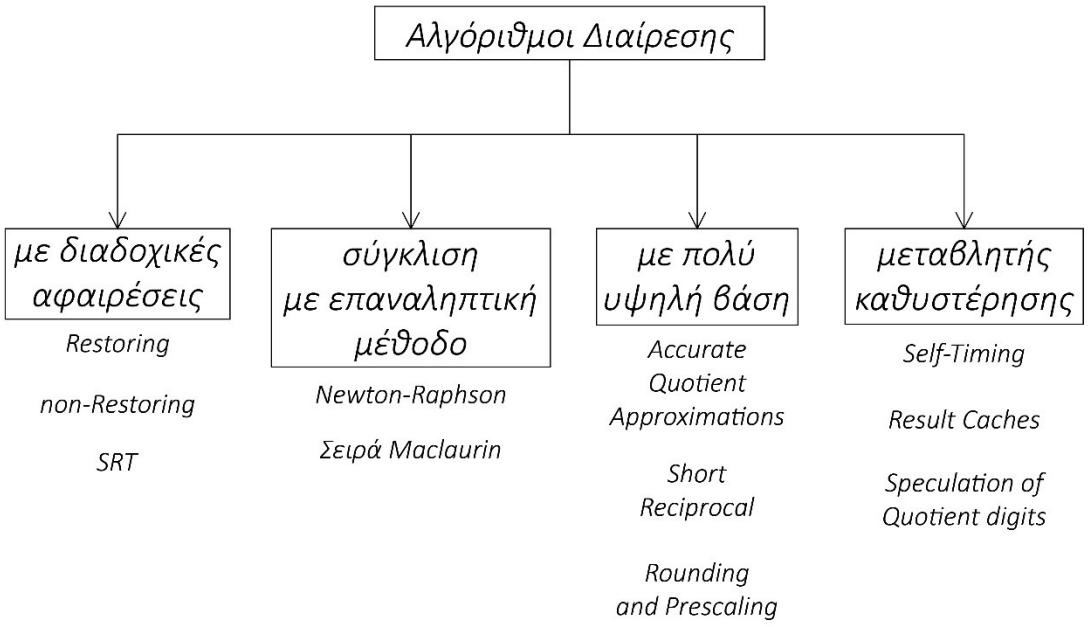
Με την ραγδαία ανάπτυξη από πλευράς υλικού, έχει επέλθει μία αντίστοιχη ανάπτυξη σε επίπεδο εφαρμογών. Οι άλλοτε εφαρμογές ειδικού σκοπού έχουν αποκτήσει γενική χρήση. Υπάρχουν πλέον χιλιάδες εφαρμογές σε κάθε πτυχή της ζωής μας. Είτε είναι επαγγελματικά εργαλεία, είτε μας διευκολύνουν καθημερινά, είτε πρόκειται για παιχνίδια, η πολυπλοκότητά τους συνεχώς αυξάνεται δημιουργώντας ζήτηση για ακόμη ταχύτερο υλικό, μεγαλύτερο όγκο δεδομένων κλπ. Για να παραμένουν λοιπόν οι επεξεργαστές ανταγωνιστικοί, πρέπει οι σχεδιαστές να δίνουν μεγάλη σημασία σε benchmarks, αλλά και να ανταπεξέρχονται στις βαριές απαιτήσεις εφαρμογών όπως τα παιχνίδια (γραφικά υψηλών απαιτήσεων) και η λήψη και επεξεργασία πολυμέσων [14] [12]. Επιπλέον, πρέπει να εκτελούν αποδοτικά αλγορίθμους μηχανικής όρασης και ψηφιακής επεξεργασίας σήματος.

Τα ανωτέρω καθιστούν απαραίτητη ακόμα και στους επεξεργαστές γενικού σκοπού την υλοποίηση *μονάδων κινητής υποδιαστολής (floating-point unit, FPU)*, σε αντίθεση με παλιότερες υλοποιήσεις που απέφευγαν να συμπεριλάβουν μονάδα διαίρεσης [15]. Ενώ όμως η μεθοδολογία για τον σχεδιασμό ταχύτατων αθροιστών και πολλαπλασιαστών είναι κατανοητή, ο σχεδιασμός μονάδων διαίρεσης αποτελεί πρόκληση. Η θεωρία πίσω από την υλοποίηση μίας τέτοιας μονάδας είναι εκτεταμένη, όπως και οι παράμετροι που πρέπει να εξετάσει ο σχεδιαστής ανάλογα με τις απαιτήσεις του συστήματος [12].

Οι αποδοτικές μονάδες διαίρεσης είναι επίσης απαραίτητες σε επεξεργαστές ειδικού σκοπού, για εφαρμογές όπως η κρυπτογραφία και η ρομποτική [3] [16]. Κατά περίπτωση, μπορεί να είναι προτεραιότητα η ταχύτητα εκτέλεσης της πράξης ή η ακρίβεια του αποτελέσματος. Ακόμα ανάλογα με τις απαιτήσεις και τους περιορισμούς του συστήματος, η μονάδα διαίρεσης μπορεί να βελτιστοποιείται σε κάποια από τα πεδία της επιφάνειας, της κατανάλωσης ισχύος και της καθυστέρησης.

2.4.4 Κατηγορίες αλγορίθμων υλοποίησης διαίρεσης-Ιστορική αναδρομή

Όπως αναφέρθηκε, η διαίρεση είναι μία σημαντική αριθμητική πράξη για τους υπολογιστές αλλά και δύσκολη στην σχεδίαση και υλοποίησή της. Υπήρξε λοιπόν έντονο το ενδιαφέρον για τη διερεύνηση γρήγορων και αποδοτικών μεθοδολογιών πραγματοποίησής της, καθ' όλη την διάρκεια της ιστορίας των υπολογιστών. Συνεπώς, η θεωρία γύρω από τη διαίρεση είναι εκτεταμένη όπως και η βιβλιογραφία. Οι αλγόριθμοι διαίρεσης χωρίζονται σε τέσσερις κατηγορίες όπως φαίνεται στην εικόνα 2.4.1. Αρχικά αναπτύσσονται οι πρώτες δύο κατηγορίες. Στην συνέχεια γίνεται αναφορά στις επόμενες δύο, που χρησιμοποιούν κατά περίπτωση και τις δύο πρώτες.



Εικόνα 2.4.1 Ταξινόμηση αλγορίθμων διαίρεσης.

2.4.4. Αλγόριθμοι με διαδοχικές αφαιρέσεις

Όπως φαίνεται στην εικόνα 2.4.1 οι διαιρέτες μπορούν να ταξινομηθούν σε τέσσερις κατηγορίες. Η πρώτη κατηγορία, χρησιμοποιεί διαδοχικές αφαιρέσεις του πολλαπλάσιου του *διαιρέτη* από το *μερικό υπόλοιπο*, και είναι ίσως η πιο διαδεδομένη με ένα ευρύ φάσμα διαφοροποιήσεων. Συγκλίνει γραμμικά στο αποτέλεσμα, κάτι που την καθιστά γενικά πιο αργή από την σύγκλιση σε ρίζα με επαναληπτική μέθοδο, που συγκλίνει τετραγωνικά. Οι τυπικές υλοποιήσεις όμως, είναι απλούστερες και απαιτούν λιγότερους πόρους. Επίσης, με την ολοκλήρωσή τους επιστρέφουν το *υπόλοιπο*, σε αντίθεση με τους διαιρέτες επαναληπτικών μεθόδων.

Η κύρια αρχή λειτουργίας βασίζεται στη μακρά διαίρεση, που χρησιμοποιούμε όταν επιλύουμε τη διαίρεση με μολύβι και χαρτί. Αρχικά παρουσιάζονται παραδείγματα για την ευκολότερη κατανόησή της. Κατόπιν, ορίζονται οι αρχές λειτουργίας και αναπτύσσονται οι διάφορες υλοποιήσεις της κατηγορίας.

2.4.4.. Μακρά διαίρεση

Για την κατανόηση του αλγόριθμου της μακράς διαίρεσης, θα παρουσιασθούν μία σειρά από παραδείγματα.

Παράδειγμα 2.4.1

Στο παράδειγμα αυτό, εκτελείται η διαίρεση $7027/3$. Συμβολίζουμε με q_k, τα ψηφία του *πηλίκου* που βρίσκουμε σε κάθε επανάληψη. Το k, υποδεικνύει εδώ το βάρος του αριθμού 10^k, σε αντίθεση με τη σημειογραφία που θα δούμε παρακάτω.

$X = PR_4$	7	0	2	7	D	3
$-q_3 \cdot D \cdot 10^3$	6				q_3	2
PR_3	1	0			q_2	3
$-q_2 \cdot D \cdot 10^2$		9			q_1	4
PR_2		1	2		q_0	2
$-q_1 \cdot D \cdot 10^1$		1	2		Q	2342
PR_1			0	7		
$-q_0 \cdot D \cdot 10^0$				6		
PR_0				1		

Η επίλυση του παραδείγματος 2.4.1, χρησιμοποιεί συντομογραφίες. Στην πραγματικότητα, δεν αφαιρείται το 6 από το 7, αλλά το 6000 από το 7027, που έχει σαν αποτέλεσμα $PR_3 = 1027$. Η λεπτομερής εκτέλεση της ίδιας διαίρεσης, φαίνεται στο παράδειγμα 2.4.2.

Παράδειγμα 2.4.2

Επανάληψη του παραδείγματος 2.4.1, χωρίς τη χρήση συντομογραφιών.

$X = PR_4$	7	0	2	7	D	3
$-q_3 \cdot D \cdot 10^3$	6	0	0	0	q_3	2
PR_3	1	0	2	7	q_2	3
$-q_2 \cdot D \cdot 10^2$		9	0	0	q_1	4
PR_2		1	2	7	q_0	2
$-q_1 \cdot D \cdot 10^1$		1	2	0	Q	2342
PR_1				7		
$-q_0 \cdot D \cdot 10^0$				6		
PR_0				1		

Παρατηρώντας τα ενδιάμεσα στάδια του *υπολοίπου*:

$$7027 - 2 \cdot 3 \cdot 10^3 = 1027 \quad \text{ή} \quad PR_4 - q_3 \cdot D \cdot 10^3 = PR_3$$
$$1027 - 3 \cdot 3 \cdot 10^2 = 0127 \quad \text{ή} \quad PR_3 - q_2 \cdot D \cdot 10^2 = PR_2$$
$$0127 - 4 \cdot 3 \cdot 10^1 = 0007 \quad \text{ή} \quad PR_2 - q_1 \cdot D \cdot 10^1 = PR_1$$
$$0007 - 2 \cdot 3 \cdot 10^0 = 0001 \quad \text{ή} \quad PR_1 - q_0 \cdot D \cdot 10^0 = PR_0$$

μπορούμε να δούμε ότι το υπόλοιπο PR_k, προκύπτει γενικά από τη σχέση:

$$PR_k = PR_{k+1} - q_k \cdot D \cdot 10^k$$

Ο συγκεκριμένος τρόπος εκτέλεσης όμως, δεν είναι βολικός για την υλοποίηση της διαίρεσης σε υπολογιστή, αφού δεν γνωρίζουμε εκ των προτέρων το βάρος στην ποσότητα $q_k \cdot D \cdot 10^k$. Εάν συγκρίνουμε τα πρώτα ψηφία των ορισμάτων στα παραπάνω παραδείγματα, θα είναι 7 από τον *διαιρετέο* και 0 από τον *διαιρέτη*, αφού η πλήρης αναπαράστασή του *διαιρέτη* με ίσο αριθμό ψηφίων, είναι 0003. Θα πρέπει λοιπόν να ελέγξουμε και το επόμενο ψηφίο του *διαιρέτη* πολλαπλασιάζοντας τον με το 10 ή απλά ολισθαίνοντας αριστερά. Και πάλι το ψηφίο του *διαιρέτη* θα είναι 0. Θα πρέπει να εκτελέσουμε τρεις ολισθήσεις τελικά, ώσπου να μπορούμε να δούμε το ψηφίο 3. Η

διαδικασία αυτή εύρεσης της δύναμης εκκίνησης του αλγορίθμου, ισοδυναμεί με την *κανονικοποίηση* του *διαιρέτη*. Προετοιμάζουμε δηλαδή τη διαίρεση, ολισθαίνοντας αριστερά $x = 3$ φορές, ώσπου να μην ηγούνται μηδενικά ψηφία. Σημειώνουμε ότι η *κανονικοποίηση* του *διαιρετέου*, δεν είναι απαραίτητη. Εάν ο *διαιρέτης* ήταν μεγαλύτερος του *διαιρετέου*, θα έπρεπε να ελέγξουμε τα πρώτα ψηφία και απλά να βάλουμε 0 και υποδιαστολή στο *πηλίκο*. Στο παράδειγμα 2.4.3, επαναλαμβάνουμε την εκτέλεση της διαίρεσης $7027/3$, χρησιμοποιώντας αυτήν την τροποποίηση, ενώ αλλάζει και η φορά της αρίθμησης των ψηφίων του *πηλίκου*:

Παράδειγμα 2.4.3

Επανάληψη του παραδείγματος 2.4.1, με αρχική ευθυγράμμιση των ορισμάτων. Τα ορίσματα $X = 7027$ και $D = 0003,$ γίνονται $X = 7027$ και $D = 3000$. Προφανώς για την εύρεση του αποτελέσματος, θα πραγματοποιηθούν $x = 3$ ολισθήσεις για την προσαρμογή της υποδιαστολής.

$X = 10 \cdot PR_0$		7	0	2	7	D	3000
$-q_1 \cdot D$		6	0	0	0	q_1	2
PR_1		1	0	2	7	q_2	3
$10 \cdot PR_1$	1	0	2	7	0	q_3	4
$-q_2 \cdot D$		9	0	0	0	q_4	2
PR_2		1	2	7	0	Q	2,342
$10 \cdot PR_2$	1	2	7	0	0		
$-q_3 \cdot D$	1	2	0	0	0		
PR_3			7	0	0		
$10 \cdot PR_3$		7	0	0	0		
$-q_4 \cdot D$		6	0	0	0		
PR_4		1	0	0	0		
$PR_4/1000$					1		

Επειδή έχουμε ευθυγραμμίσει τα ορίσματα, η υποδιαστολή βρίσκεται υποχρεωτικά μετά το πρώτο ψηφίο. Θα πρέπει έτσι να πολλαπλασιάσουμε το *πηλίκο* με το 10^3, να ολισθήσουμε δηλαδή τρεις θέσεις αριστερά, όπως κάναμε κατά την *κανονικοποίηση* του *διαιρέτη*.

Εάν η διαίρεση πραγματοποιείται σε αριθμούς *σταθερής-υποδιαστολής*, το *τελικό υπόλοιπο* μπορεί να μας ενδιαφέρει. Υπολογίζεται εύκολα, με x ολισθήσεις δεξιά, όσες δηλαδή ολισθήσαμε αριστερά το *μερικό υπόλοιπο* κατά την εκτέλεση της διαίρεσης.

Η σχέση του *μερικού υπολοίπου* PR_j, προκύπτει έτσι σε πιο βολική μορφή:

$$\left.\begin{array}{c} \beta\acute{\alpha}\sigma\eta \cdot PR_0 = \Delta\iota\alpha\iota\rho\varepsilon\tau\acute{\varepsilon}o\varsigma \\ PR_{j+1} = \beta\acute{\alpha}\sigma\eta \cdot PR_j - q_{j+1} \cdot D, \;\; j = \{0,1,\dots,n-1\} \end{array}\right\} \qquad (2.4.3)$$

Πρόκειται για μία ακολουθιακή διαδικασία, με διαδοχικά εξαρτημένα βήματα αφαιρέσεων και ολισθήσεων, όπου:

- $j + 1$ είναι το βήμα της επανάληψης

- PR_{j+1} είναι το *μερικό υπόλοιπο (Partial Remainder)*
- q_{j+1} είναι το ψηφίο του *πηλίκου*
- και D ο *διαιρέτης*

Αρχικά ισχύει $r \cdot PR_0 = X = Διαιρετέος$. Κατόπιν, ακολουθείται επαναληπτικά η παρακάτω διαδικασία:

1. εξέταση προηγούμενου *μερικού υπολοίπου* ($r \cdot PR_j$) και *διαιρέτη (D)*, και απόφαση για το ψηφίο του *πηλίκου* (q_{j+1}).
2. σχηματισμός με πολλαπλασιασμό ή ολίσθηση του γινομένου *ψηφίου πηλίκου-διαιρέτη* ($q_{j+1} \cdot D$).
3. αφαίρεση του πολλαπλάσιου του *διαιρέτη* από το προηγούμενο *μερικό υπόλοιπο* $\left(r \cdot PR_j - q_{j+1} \cdot D\right)$, ώστε να προκύψει το νέο *μερικό υπόλοιπο* (PR_{j+1}).

Η καθυστέρηση σε αλγόριθμους τέτοιου τύπου, είναι γραμμική ως προς το μήκος των ορισμάτων, αφού σε κάθε επανάληψη υπολογίζεται σταθερός αριθμός bits, που εξαρτάται από τη *βάση* του αριθμητικού συστήματος. Συγκεκριμένα, υπολογίζονται $\log_2 Radix$ bits σε κάθε βήμα.

Για να λειτουργεί σωστά ο αλγόριθμος, πρέπει το *μερικό υπόλοιπο* μετά από κάθε αφαίρεση να είναι μικρότερο του *διαιρέτη*. Πρέπει δηλαδή στον κύκλο j, να ισχύει $r \cdot PR_{j-1} < r \cdot D$. Για να συμβεί αυτό, είναι απαραίτητη η επιλογή του σωστού *ψηφίου πηλίκου*, έπειτα από μία διαδικασία σύγκρισης μεταξύ *μερικού υπολοίπου* και *διαιρέτη*, που ονομάζεται *συνάρτηση επιλογής* [17]. Γράφουμε:

$$q_{j+1} = SEL(radix \cdot PRj, D) \qquad (2.4.4)$$

Η ανωτέρω επαναληπτική διαδικασία ισχύει για διαίρεση ακεραίων, όπως και για διαίρεση καθαρά κλασματικών μερών. Ακολουθεί η απόδειξη, για διαίρεση καθαρά κλασματικών αριθμών *σταθερής-υποδιαστολής* με βάση το 2. Για την ευκολία της απόδειξης θέτουμε $PR_0 = X$ αντί για $r \cdot PR_0 = X$. Δηλαδή ολισθαίνουμε το *διαιρετέο* πριν το πρώτο βήμα, σε αντίθεση με την σχέση 2.4.3 που το πρώτο βήμα αποτελεί εξαίρεση.

Απόδειξη [7]
Η σχέση 2.4.3 για $m = n$ ($n = k + m, k = 0$), $βάση = 2$, γίνεται:

$$\left.\begin{array}{c} PR_0 = X \\ PR_j = 2 \cdot PR_{j-1} - q_j \cdot D, \;\; j = \{1, \dots, m\} \end{array}\right\}$$

Για να λειτουργεί σωστά ο αλγόριθμος πρέπει $X < D$, οπότε το πρώτο ψηφίο του Q είναι 0 και ακολουθεί υποδιαστολή. Επομένως $Q = 0.q_1 q_2 \dots q_m$.

Το μερικό υπόλοιπο στο τελευταίο βήμα, είναι PR_m, και αντικαθιστώντας διαδοχικά:

$$\begin{aligned} PR_m &= 2PR_{m-1} - q_m \cdot D = 2(2PR_{m-2} - q_{m-1} \cdot D) - q_m \cdot D = \cdots \\ &= 2^m PR_0 - (q_m + 2q_{m-1} + \cdots + 2^{m-1} q_1) \cdot D \end{aligned}$$

Αντικαθιστώντας $PR_0 = \textit{Διαιρετέος} = X$:

$$PR_m = 2^m X - (q_m + 2q_{m-1} + \cdots + 2^{m-1}q_1) \cdot D$$

Διαιρώντας με 2^m:

$$PR_m 2^{-m} = X - (2^{-1}q_1 + 2^{-2}q_2 + \cdots + 2^{-m}q_m) \cdot D$$

Οπότε:

$$PR_m 2^{-m} = R = X - Q \cdot D$$

Όπου R το πραγματικό *τελικό υπόλοιπο*.

Η ευθυγράμμιση των ορισμάτων που αναφέρθηκε, δεν μας απασχολεί στην παρούσα εργασία, αφού ασχολούμαστε με την υλοποίηση διαίρεσης αριθμών *κινητής-υποδιαστολής*. Στην περίπτωση αυτή διαιρούνται τα *σημαντικά μέρη* των αριθμών, τα οποία είναι ήδη *κανονικοποιημένα* κατά το IEEE πρότυπο. Αρχικά δεν υπάρχει έλεγχος εάν $X < D$ όπως στην απόδειξη, όμως επειδή $X = 0.1x_2 \dots x_m$ και $D = 0.1d_2 \dots d_m$, διασφαλίζεται ότι:

$$X < 2D \Rightarrow 2PR_0 < 2D \Rightarrow PR_0 < D$$

Επομένως το πρώτο ψηφίο του *πηλίκου* μπορεί να είναι 0 ή 1. Η υποδιαστολή τίθεται πάντα μετά το πρώτο ψηφίο, ενώ το *υπόλοιπο* δεν μας ενδιαφέρει καθώς υπολογίζεται διαφορετικά για τους αριθμούς *κινητής-υποδιαστολής*.

2.4.4.. Restoring

Για να εκτελεστεί η μακρά διαίρεση, πρέπει σε κάθε βήμα να αποφασίζουμε το q, χρησιμοποιώντας την προπαίδεια και δοκιμάζοντας τιμές με το μυαλό μας. Ένας επεξεργαστής όμως, θα πρέπει να προσομοιώσει αυτή τη διαδικασία. Έτσι προέκυψε η αρχική και απλούστερη μορφή των αλγορίθμων με διαδοχικές αφαιρέσεις, ο *Restoring* αλγόριθμος. Στο παράδειγμα 2.4.4, επανεκτελείται η διαίρεση του παραδείγματος 2.4.3 με χρήση του *Restoring*.

Παράδειγμα 2.4.4

Επανάληψη του παραδείγματος 2.4.3, με χρήση του *Restoring*. Εκκινούμε την διαδικασία με *κανονικοποίηση* του *διαιρέτη*, εάν δεν είναι ήδη σε *κανονικοποιημένη* μορφή. Αφαιρούμε τον *διαιρέτη* από το *μερικό υπόλοιπο*, διαδοχικά μέχρι αυτό να προκύψει αρνητικό, οπότε και το *επαναφέρουμε (restore)* στην αμέσως προηγούμενη σωστή τιμή.

$+7027 - 3 \cdot 10^3 = +4027$	$+7027 - 3000 = +4027$		
$+4027 - 3 \cdot 10^3 = +1027$	$+4027 - 3000 = +1027$		
$+1027 - 3 \cdot 10^3 = -1973$	$+1027 - 3000 = -1973$		
$-1973 + 3 \cdot 10^3 = +1027$	$-1973 + 3000 = +1027$	*Επαναφορά*	$q_1 = 2$
	Αριστερή ολίσθηση μερικού υπολοίπου (Lsh PR)		
$+1027 - 3 \cdot 10^2 = +727$	$+10270 - 3000 = +7270$		
$+727 - 3 \cdot 10^2 = +427$	$+7270 - 3000 = +4270$		
$+427 - 3 \cdot 10^2 = +127$	$+4270 - 3000 = +1270$		
$+127 - 3 \cdot 10^2 = -173$	$+1270 - 3000 = -1730$		

$-173 + 3 \cdot 10^2 = +127$	$-1730 + 3000 = +1270$	*Επαναφορά*	$q_2 = 3$
	Lsh PR		
$+127 - 3 \cdot 10^1 = +97$	$+12700 - 3000 = +9700$		
$+97 - 3 \cdot 10^1 = +67$	$+9700 - 3000 = +6700$		
$+67 - 3 \cdot 10^1 = +37$	$+6700 - 3000 = +3700$		
$+37 - 3 \cdot 10^1 = +7$	$+3700 - 3000 = +700$		
$+7 - 3 \cdot 10^1 = -23$	$+700 - 3000 = -2300$		
$-23 + 3 \cdot 10^1 = +7$	$-2300 + 3000 = +700$	*Επαναφορά*	$q_3 = 4$
	Lsh PR		
$+7 - 3 \cdot 10^0 = +4$	$+7000 - 3000 = +4000$		
$+4 - 3 \cdot 10^0 = +1$	$+4000 - 3000 = +1000$		
$+1 - 3 \cdot 10^0 = -2$	$+1000 - 3000 = -2000$		
$-2 + 3 \cdot 10^0 = +1$	$-2000 + 3000 = +1000$	*Επαναφορά*	$q_4 = 2$

Προκύπτει λοιπόν, $Q = 2{,}342$ και $R = 1000$. Ολισθαίνουμε τρεις θέσεις αριστερά το *πηλίκο*, και τρεις θέσεις δεξιά το *τελικό υπόλοιπο*, όσες ολισθήσαμε δηλαδή για την *κανονικοποίηση* του διαιρέτη. Τα σωστά αποτελέσματα προκύπτουν, $Q = 2342$ και $R = 1$.

Στο δυαδικό σύστημα, η διαδικασία αυτή είναι απλούστερη, αφού το σύνολο ψηφίων του *πηλίκου* είναι το $\{0, 1\}$. Κάθε βήμα ξεκινάει με την υπόθεση $q_j = 1$, γίνεται η αφαίρεση, και αν το *μερικό υπόλοιπο* προκύψει θετικό, σημαίνει ότι ήταν σωστή. Αν προκύψει αρνητικό, τίθεται $q_j = 0$ και *επαναφέρεται* το *μερικό υπόλοιπο*. Αυτό φαίνεται στο παράδειγμα 2.4.5, που εκτελείται η διαίρεση $(17/3)_{10} = (10001/00011)_2$.

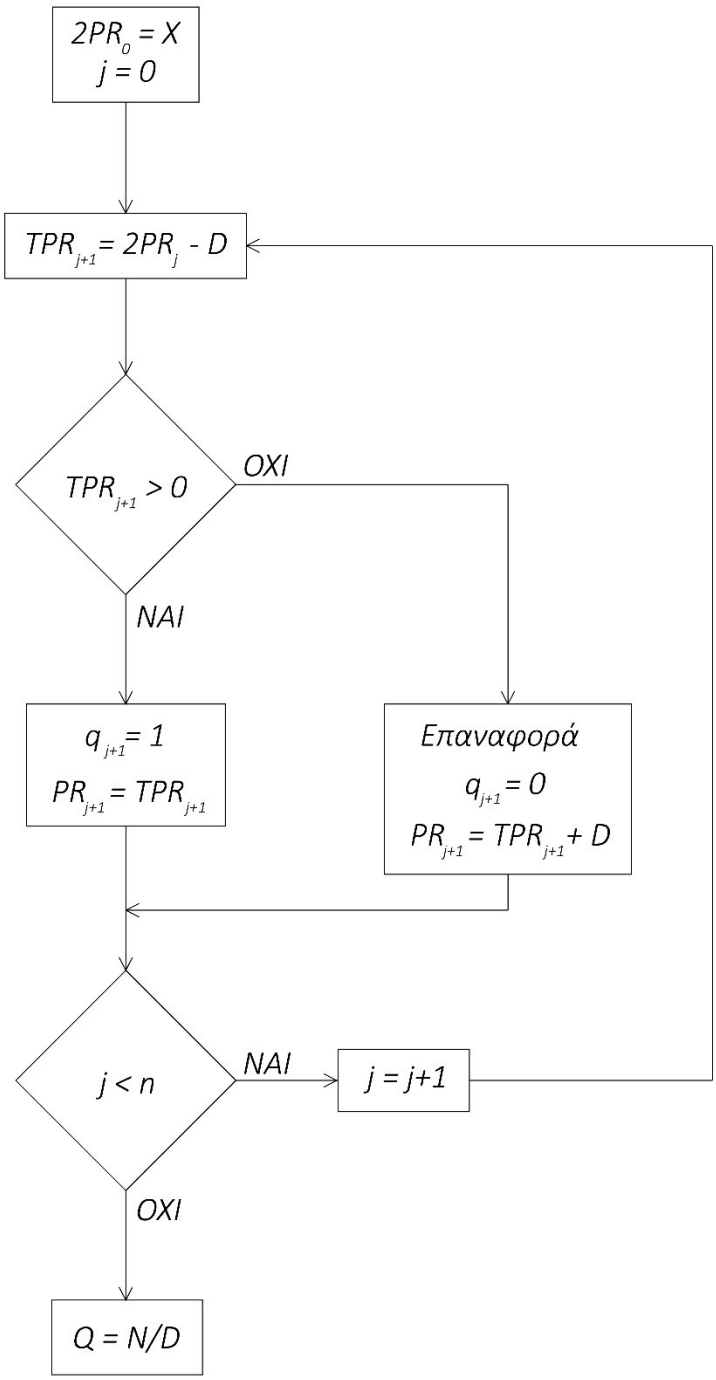
Παράδειγμα 2.4.5

Εκτέλεση της διαίρεσης $(17/3)_{10} = (10001/00011)_2$. Για λόγους ευκολίας, οι αριθμητικές τιμές είναι γραμμένες στο δεκαδικό σύστημα. Μία αριστερή ολίσθηση όμως, αφορά πολλαπλασιασμό με το δύο. *Κανονικοποιούμε* το διαιρέτη με τρεις ολισθήσεις αριστερά, που γίνεται $11000_2 = 24_{10}$ και ξεκινάμε τη διαδικασία.

$+17 - 3 \cdot 2^3 = -7$	$+17 - 24 = -7$		$q_1 = 1$
$-7 + 3 \cdot 2^3 = +17$	$-7 + 24 = +17$	*Επαναφορά*	$q_1 = 0$
	Lsh		
$+17 - 3 \cdot 2^2 = +5$	$+34 - 24 = +10$		$q_2 = 1$
	Lsh		
$+5 - 3 \cdot 2^1 = -1$	$+20 - 24 = -4$		$q_3 = 1$
$-1 + 3 \cdot 2^1 = +5$	$-4 + 24 = +20$	*Επαναφορά*	$q_3 = 0$
	Lsh		
$+5 - 3 \cdot 2^0 = +2$	$+40 - 24 = +16$		$q_4 = 1$

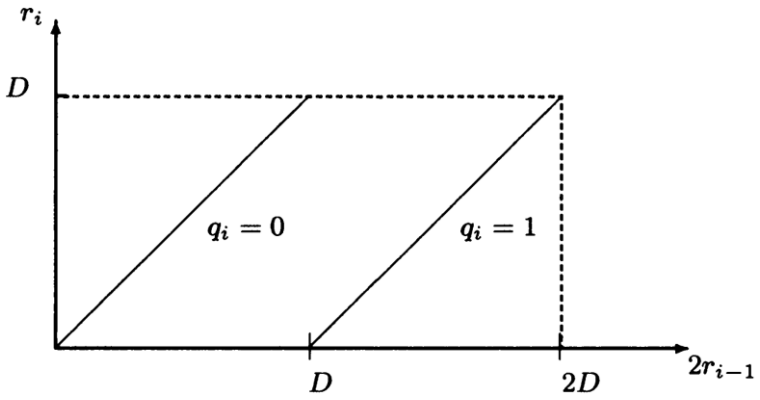
Προκύπτουν λοιπόν $Q = 0.101_2$ και $R = 10000_2 = 16_{10}$. Διορθώνουμε *πηλίκο* με τρεις ολισθήσεις αριστερά και *υπόλοιπο* με τρεις ολισθήσεις δεξιά, οπότε τελικά $Q = 101_2 = 5_{10}$ και $R = 00010_2 = 2_{10}$.

Στην εικόνα 2.4.2, απεικονίζεται το διάγραμμα ροής του αλγορίθμου Restoring. Η μεταβλητή TPR_{j+1}, εκπροσωπεί το *προσωρινό μερικό υπόλοιπο*.



Εικόνα 2.4.2 Διάγραμμα ροής του αλγορίθμου Restoring.

Στην εικόνα 2.4.3 απεικονίζεται το διάγραμμα Robertson του αλγορίθμου. Το διάγραμμα αυτό απεικονίζει ότι εάν $2PR_{j-1} < 2D$, το q_j επιλέγεται έτσι ώστε $PR_j < D$. Εφόσον $X < 2D \Rightarrow 2PR_0 < 2D$, ο αλγόριθμος θα συγκλίνει στο σωστό *τελικό υπόλοιπο* $R < D$.



Εικόνα 2.4.3 Διάγραμμα Robertson αλγορίθμου Restoring [18] [7].

Στο παράδειγμα 2.4.6, αναπτύσσεται η πλήρης εκτέλεση της διαίρεσης $0.1010/0.1111$, με την δυαδική αναπαράσταση σε *συμπλήρωμα ως προς δύο*. Το *πιο σημαντικό ψηφίο* έχει αρνητική αξία, και βρίσκεται στην δύναμη 2^1. Έτσι μπορούν να αναπρασταθούν αριθμοί στο πεδίο τιμών [-2, 1.9375], που είναι υπερσύνολο του πεδίου αναπαράστασης του *μερικού υπολοίπου*, για τον αλγόριθμο *Restoring* (-D, 2D). Εάν ήταν υποσύνολο θα υπήρχε κίνδυνος για *υπερχείλιση*.

Παράδειγμα 2.4.6

Εκτέλεση της διαίρεσης $0.1010/0.1111$, με χρήση του αλγορίθμου *Restoring*.

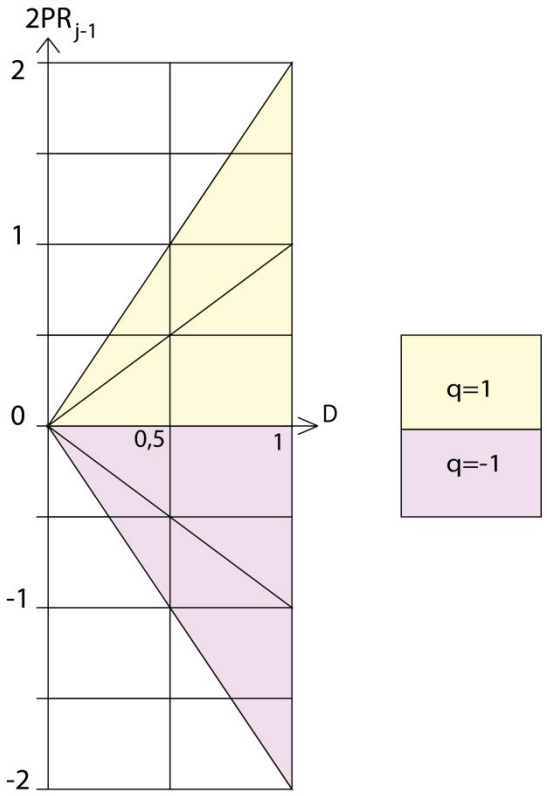
	2's comp								ulp	decimal		
X		0	0	,	1	0	1	0		0,625		
D		0	0	,	1	1	1	1		0,9375		
X=2PR0		0	0	,	1	0	1	0		0,625		
-D	+	1	1	,	0	0	0	0	1	-0,9375		q1=1
TPR1		1	1	,	1	0	1	1		-0,3125	TPR1=2PR0-D<0	Λάθος
+D	+	0	0	,	1	1	1	1		0,9375		Επαναφορά
PR1		0	0	,	1	0	1	0		0,625	PR1=TPR1+D	q1=0
2PR1		0	1	,	0	1	0	0		1,25		
-D	+	1	1	,	0	0	0	0	1	-0,9375		q2=1
PR2		0	0	,	0	1	0	1		0,3125	PR2=TPR2=2PR1-D>0	Σωστό
2PR2		0	0	,	1	0	1	0		0,625		
-D	+	1	1	,	0	0	0	0	1	-0,9375		q3=1
TPR3		1	1	,	1	0	1	1		-0,3125	TPR3=2PR2-D<0	Λάθος
+D	+	0	0	,	1	1	1	1		0,9375		Επαναφορά
PR3		0	0	,	1	0	1	0		0,625	PR3=TPR3+D	q3=0
2PR3		0	1	,	0	1	0	0		1,25		
-D	+	1	1	,	0	0	0	0	1	-0,9375		q4=1
PR4		0	0	,	0	1	0	1		0,3125	PR4=TPR4=2PR3-D>0	Σωστό

Προκύπτουν λοιπόν $Q = 0.101_2 = 0.625_{10}$ και $R = 0.0101_2 = 0.3125_{10}$. Με τρεις ολισθήσεις δεξιά το *τελικό υπόλοιπο* γίνεται $R = 0.0000101_2 = 0.0390625_{10}$.

2.4.4.. Non-Restoring

Είναι κατανοητό, ότι ο *Restoring* αλγόριθμος χρειάζεται έως *2n* κύκλους για την εκτέλεσή του, όπου *n* το μήκος των ορισμάτων. Χρειάζονται τουλάχιστον *n* αφαιρέσεις, ενώ μπορεί να χρειαστούν μέχρι *n επαναφορές*. Με σκοπό την ελαχιστοποίηση των κύκλων, ο αλγόριθμος εξελίχθηκε στον *non-Restoring*, που αντί να διορθώνει άμεσα *πηλίκο* και *υπόλοιπο*, αποθηκεύει το «λάθος» στην αναπαράσταση του *πηλίκου*, διορθώνοντάς το στην επόμενη επανάληψη. Αυτό επιτυγχάνεται με τη χρήση αναπαράστασης *προσημασμένου ψηφίου (Signed Digit)* για το *πηλίκο*, όπου το σύνολο ψηφίων γίνεται το $\{\bar{1}, 1\}$. Αν αποφασιστεί $q_j = 1$, εκτελείται αφαίρεση του *διαιρέτη* από το *μερικό υπόλοιπο*, ενώ αν αποφασιστεί $q_j = \bar{1}$, ο διαιρέτης προστίθεται στο *μερικό υπόλοιπο*, διορθώνοντας έτσι το προηγούμενο λάθος. Ο *κανόνας επιλογής* φαίνεται στην σχέση 2.4.5, και στο P-D γράφημα της εικόνας 2.4.4.

$$q_j = \begin{cases} 1 \ \alpha\nu \ PR_j \geq 0 \\ \bar{1} \ \alpha\nu \ PR_j < 0 \end{cases} \qquad (2.4.5)$$



Εικόνα 2.4.4

Ας εκτελέσουμε ξανά το παράδειγμα 2.4.5 με την μέθοδο *non-Restoring* για να παρατηρήσουμε τις διαφορές:

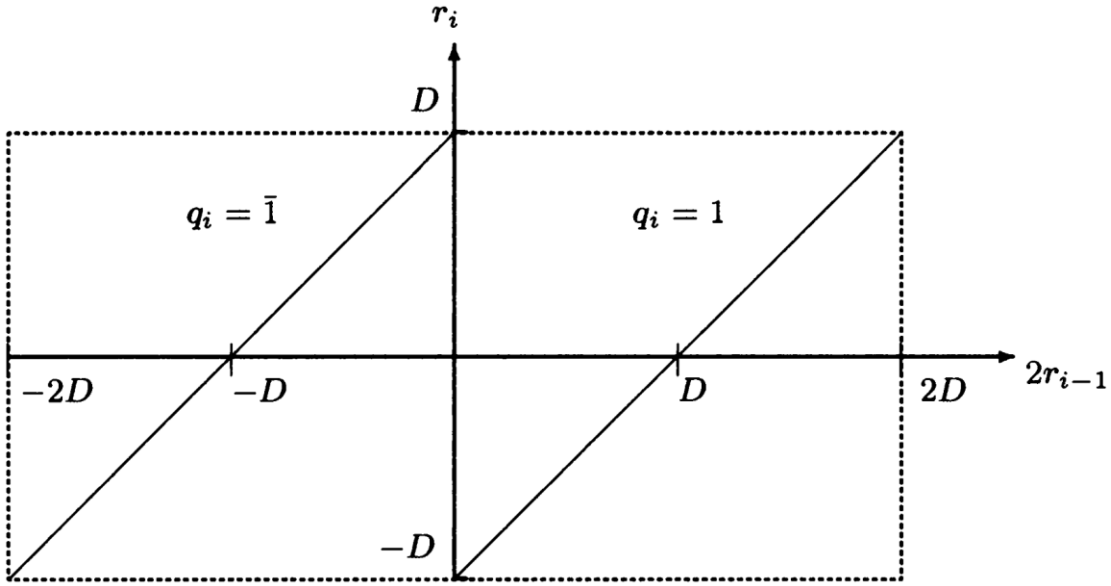
Παράδειγμα 2.4.7

Εκτέλεση της διαίρεσης $(17/3)_{10} = (10001/00011)_2$. Κανονικοποιούμε το *διαιρέτη* με τρεις ολισθήσεις αριστερά, που γίνεται $11000_2 = 24_{10}$ και ξεκινάμε τη διαδικασία.

$+17-3\cdot 2^3=-7$	$+17-24=-7$	$q_1=1$
	Lsh	
$-7+3\cdot 2^2=+5$	$-14+24=+10$	$q_2=\bar{1}$
	Lsh	
$+5-3\cdot 2^1=-1$	$+20-24=-4$	$q_3=1$
	Lsh	
$-1+3\cdot 2^0=+2$	$-8+24=+16$	$q_4=\bar{1}$

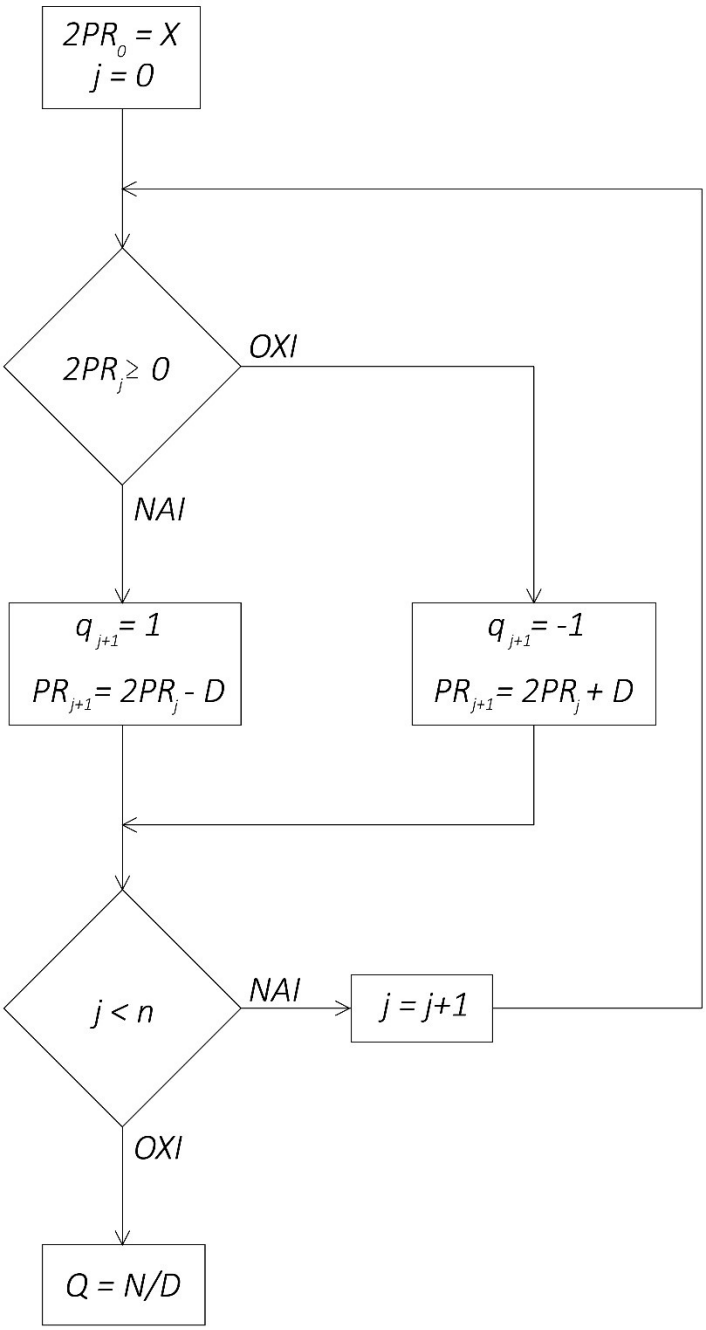
Προκύπτει λοιπόν $Q=1.\bar{1}1\bar{1}_2$ και $R=10000_2=16_{10}$. Διορθώνουμε *πηλίκο* με τρεις ολισθήσεις αριστερά και *υπόλοιπο* με τρεις ολισθήσεις δεξιά, οπότε τελικά $Q=1\bar{1}1\bar{1}_2=5_{10}$ και $R=00010_2=2_{10}$.

Πρέπει να σημειώσουμε ότι το αποτέλεσμα είναι σωστό, αλλά βρίσκεται σε διαφορετική αναπαράσταση, αφού ισχύει ότι $Q=1\bar{1}1\bar{1}_2=0101_2=5_{10}$. Το μειονέκτημα είναι ότι θα πρέπει το *πηλίκο* να μετατραπεί σε *συμπλήρωμα ως προς δύο*, αλλά αυτό μπορεί να γίνει αποδοτικά όπως θα δειχθεί παρακάτω. Το πλεονέκτημα όμως είναι, ότι αποφασίζεται ταχύτερα το q_j αφού η επιλογή βασίζεται στο πρόσημο του *μερικού υπολοίπου*, και όχι στην σύγκρισή του με το D. Επειδή επιτρέπονται και αρνητικές τιμές για το *μερικό υπόλοιπο*, το κριτήριο σύγκλισης του αλγορίθμου γίνεται $|2PR_0|<2D$. Αυτά φαίνονται στο διάγραμμα Robertson του αλγορίθμου *non-Restoring*, που παρουσιάζεται στην εικόνα 2.4.5.



Εικόνα 2.4.5 Διάγραμμα Robertson αλγορίθμου non-Restoring [18] [7].

Στην εικόνα 2.4.6, απεικονίζεται το διάγραμμα ροής του non-Restoring αλγορίθμου.



Εικόνα 2.4.6 Διάγραμμα ροής του αλγορίθμου non-Restoring.

Στο παράδειγμα 2.4.8 επανεκτελείται η διαίρεση $0.1010/0.1111$, με τη χρήση του *non-Restoring*. Η δυαδική αναπαράσταση είναι και πάλι σε *συμπλήρωμα ως προς δύο*. Όπως πριν, μπορούν να αναπρασταθούν αριθμοί στο πεδίο τιμών [-2, 1.9375], που είναι υπερσύνολο του πεδίου αναπαράστασης του *μερικού υπολοίπου*, για τον αλγόριθμο *non-Restoring* (-2D, 2D).

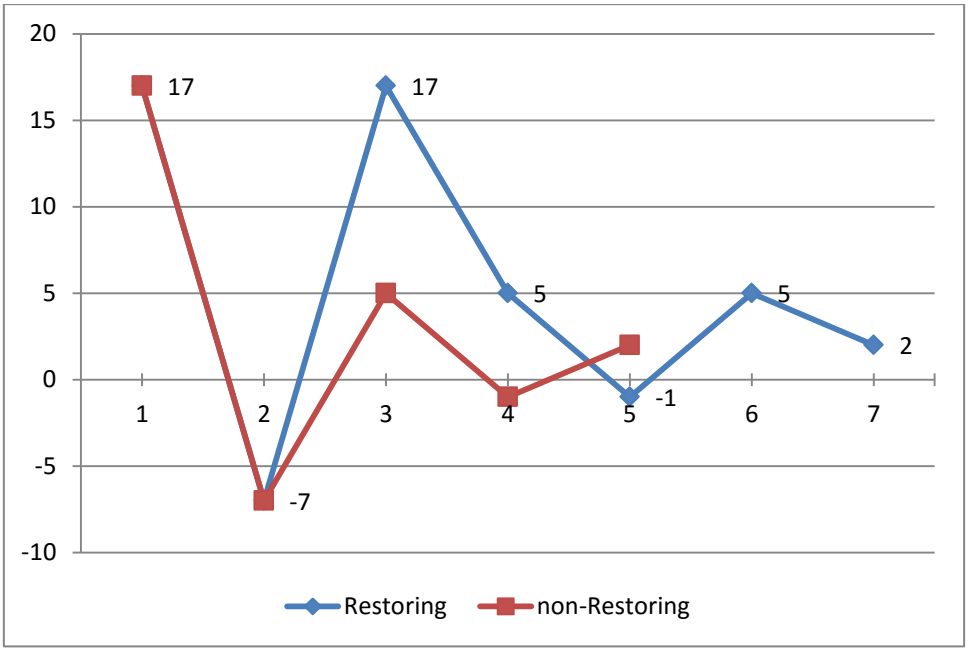
Παράδειγμα 2.4.8

Εκτέλεση της διαίρεσης $0.1010/0.1111$, με χρήση του αλγορίθμου *non-Restoring*.

	2's comp							ulp	decimal		
X		0	0	,	1	0	1	0		0,625		
D		0	0	,	1	1	1	1		0,9375		
X=2PR0		0	0	,	1	0	1	0		0,625	2PR0>0	q1=1
-D	+	1	1	,	0	0	0	0	1	-0,9375		
PR1		1	1	,	1	0	1	1		-0,3125	PR1=2PR0-D	
2PR1		1	1	,	0	1	1	0		-0,625	2PR1<0	q2=-1
+D	+	0	0	,	1	1	1	1		0,9375		
PR2		0	0	,	0	1	0	1		0,3125	PR2=2PR1+D	
2PR2		0	0	,	1	0	1	0		0,625	2PR2>0	q3=1
-D	+	1	1	,	0	0	0	0	1	-0,9375		
PR3		1	1	,	1	0	1	1		-0,3125	PR3=2PR2-D	
2PR3		1	1	,	0	1	1	0		-0,625	2PR3<0	q4=-1
+D	+	0	0	,	1	1	1	1		0,9375		
PR4		0	0	,	0	1	0	1		0,3125	PR4=2PR3+D	

Προκύπτουν λοιπόν $Q = 1.\bar{1}1\bar{1}_2 = 0.625_{10}$ και $R = 0.0101_2 = 0.3125_{10}$. Με τρεις ολισθήσεις δεξιά το *τελικό υπόλοιπο* γίνεται $R = 0.0000101_2 = 0.0390625_{10}$.

Τέλος στην εικόνα 2.4.7, παρουσιάζεται γραφικά η σύγκριση της σύγκλισης των δύο αλγορίθμων.



Εικόνα 2.4.7 Γραφική σύγκριση της σύγκλισης των αλγορίθμων Restoring και non-Restoring.

Παρατηρούμε ότι ο αλγόριθμος *non-Restoring* συγκλίνει ταχύτερα στο αποτέλεσμα. Το μειονέκτημά του, είναι η ανάγκη μετατροπής του πηλίκου από *SD* αναπαράσταση σε *συμπλήρωμα ως προς δύο*. Ένας τρόπος να γίνει αυτό, είναι να αφαιρεθεί η συνιστώσα αρνητικής αξίας του *πηλίκου* από την θετικής αξίας. Απαιτεί όμως τον υπολογισμό του *λιγότερο σημαντικού ψηφίου* προτού ξεκινήσει η αφαίρεση, και την *διάδοση κρατουμένου* καθ' όλο το μήκος του *πηλίκου*. Ένας προτιμότερος τρόπος, είναι η εφαρμογή αλγορίθμου

που μετατρέπει τα ψηφία *SD* σε *συμπλήρωμα ως προς δύο*, αμέσως μετά την απόφασή τους *(on the fly)* [19]. Στην περίπτωση του non-*Restoring* προκύπτει ακόμα απλούστερος, αφού το *πηλίκο* με σύνολο ψηφίων $\{\bar{1}, 1\}$ μπορεί να αναπαρασταθεί από τις τιμές 0 και 1 αντίστοιχα, και αρκεί ένα μόνο bit.

Έστω το αποτέλεσμα στην ανωτέρω αναπαράσταση $(0.p_1 \dots p_m)$, όπου $p_i = \frac{1}{2}(q_i + 1)$. Ο αριθμός μετατρέπεται σε *συμπλήρωμα ως προς δύο* με την ακόλουθη διαδικασία:

1. Αριστερή ολίσθηση κατά μία θέση.
2. Αναστροφή του *πιο σημαντικού ψηφίου*.
3. Ολίσθηση ενός 1 στην *λιγότερο σημαντική θέση*.

Έτσι το αποτέλεσμα προκύπτει σε *συμπλήρωμα ως προς δύο* $(1 - p_1).p_2 p_3 \dots p_m 1$ [7].

2.4.4.. SRT

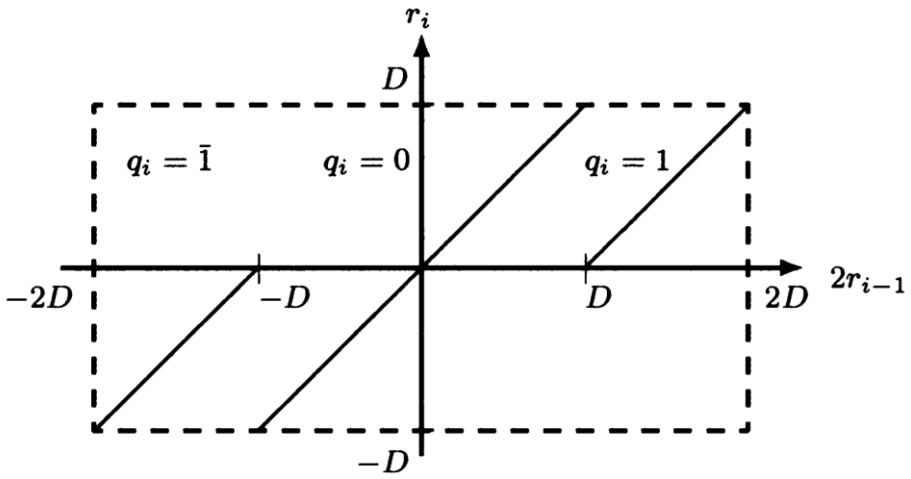
Παρά την γρήγορη απόφαση του q, ο τυπικός *non-Restoring* δεν είναι ο αποδοτικότερος αλγόριθμος, αφού υστερεί σε δύο σημεία. Πρώτον, πρέπει να πραγματοποιείται πρόσθεση ή αφαίρεση για κάθε επανάληψη. Δεύτερον, δεν μπορεί να χρησιμοποιηθεί αριθμητική με *περίσσεια*, καθώς ο έλεγχος προσήμου γίνεται σε έναν δεδομένο αριθμό *πιο σημαντικών ψηφίων*. Εάν τα πρώτα ψηφία είναι μηδενικά και το πρόσημο «κρύβεται» στα επόμενα ψηφία, δεν μπορεί να αποφασισθεί το πρόσημο. Κατά συνέπεια κάθε βήμα, πρέπει να περιμένει τη *διάδοση κρατουμένου* από το τελευταίο στο πρώτο bit [20]. Αναπτύχθηκε λοιπόν μία διαφοροποίησή του, που επιτρέπει την χρήση τέτοιας αριθμητικής εισάγοντας στο σύνολο ψηφίων του q το 0, κάνοντάς το $\{\bar{1}, 0, 1\}$. Το *κρατούμενο* κάθε πράξης, *αποθηκεύεται* έτσι στην αναπαράσταση του *υπολοίπου* και *διαδίδεται κάθετα* αντί για *οριζόντια*. Επιπλέον, κάθε φορά που το q προκύπτει 0, ολισθαίνουμε το *υπόλοιπο* χωρίς να γίνει πράξη, μειώνοντας έτσι τη μέση καθυστέρηση.

Η διαφοροποίηση αυτή εισήχθη από τους Sweeney, Robertson (1958) και Tocher (1956), ολοκληρώθηκε από τον MacSorley (1961) και αναλύθηκε από τον Freiman (1961), που απέδωσε στον αλγόριθμο το όνομα *SRT* [21] [18] [22]. Η εργασία του Atkins [23], αποτελεί την πρώτη σημαντική ανάλυση των αλγορίθμων *SRT*, ενώ η εργασία του Tan [24] αποτελεί σημαντική πηγή των *SRT υψηλότερης-βάσης*, και μία αναλυτική μέθοδο *πινάκων αναφοράς SRT*. Τέλος, αναπτύσσεται εκτενώς η θεωρία της διαίρεσης SRT, από τους Ercegovac και Lang [25].

Εάν διαφοροποιηθεί ο *non-Restoring* , έτσι ώστε το 0 να ανήκει στο σύνολο ψηφίων του q, ο *κανόνας επιλογής* γίνεται :

$$q_j = \begin{cases} 1 \ \varepsilon\acute{\alpha}\nu \ 2PR_{j-1} \geq D \\ 0 \ \ \varepsilon\acute{\alpha}\nu \ -D \leq 2PR_{j-1} < D \\ \bar{1} \ \ \varepsilon\acute{\alpha}\nu \ 2PR_{j-1} < -D \end{cases} \qquad (2.4.6)$$

Όπως φαίνεται από τον *κανόνα επιλογής* και το διάγραμμα Robertson της εικόνας 2.4.8, για την απόφαση του q_j χρειάζεται πλήρης σύγκριση μεταξύ του $2PR_{j-1}$ και του D ή του $-D$.

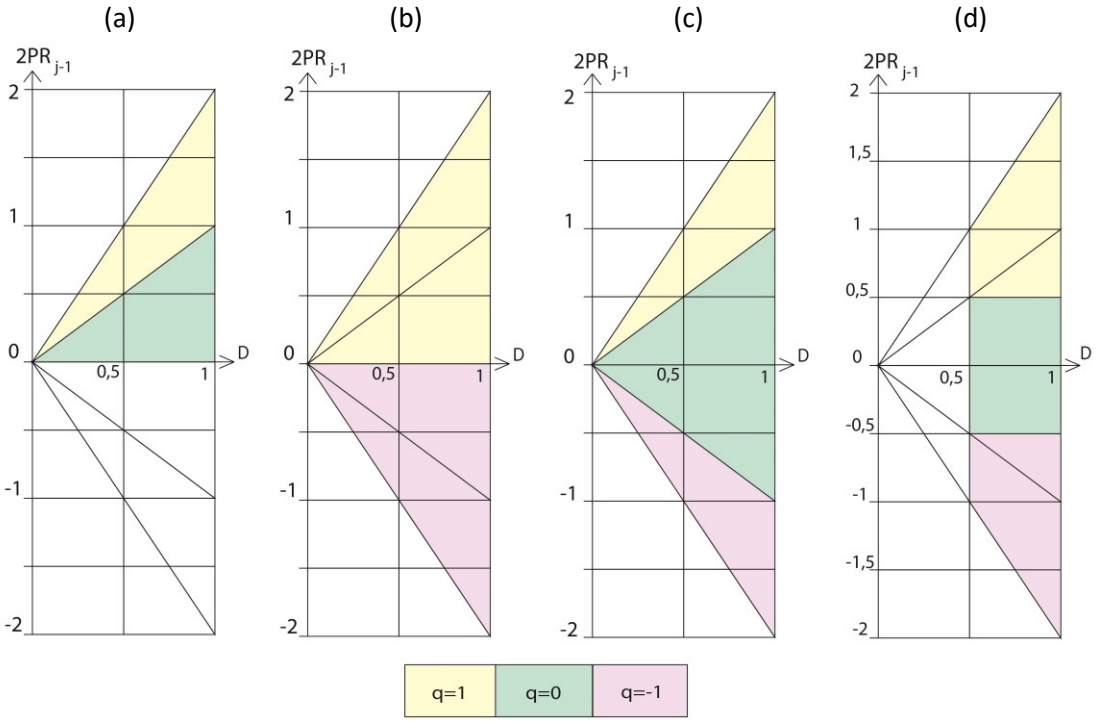


Εικόνα 2.4.8 Διάγραμμα Robertson αλγορίθμου non-Restoring με $q_i = 0$ [18] [7].

Εάν όμως περιορίσουμε τον *διαιρέτη* να βρίσκεται σε *κανονικοποιημένη* μορφή, τότε θα ισχύει $\frac{1}{2} \leq |D| < 1$. Έτσι η περιοχή του $2PR_{j-1}$, όπου $q_j = 0$, περιορίζεται σύμφωνα με την ανισότητα $-D \leq -\frac{1}{2} \leq 2r_{i-1} < \frac{1}{2} \leq D$. Κρατώντας τις σταθερές $\frac{1}{2}$ και $-\frac{1}{2}$ ο *κανόνας επιλογής* διαμορφώνεται σύμφωνα με τον αλγόριθμο *SRT*:

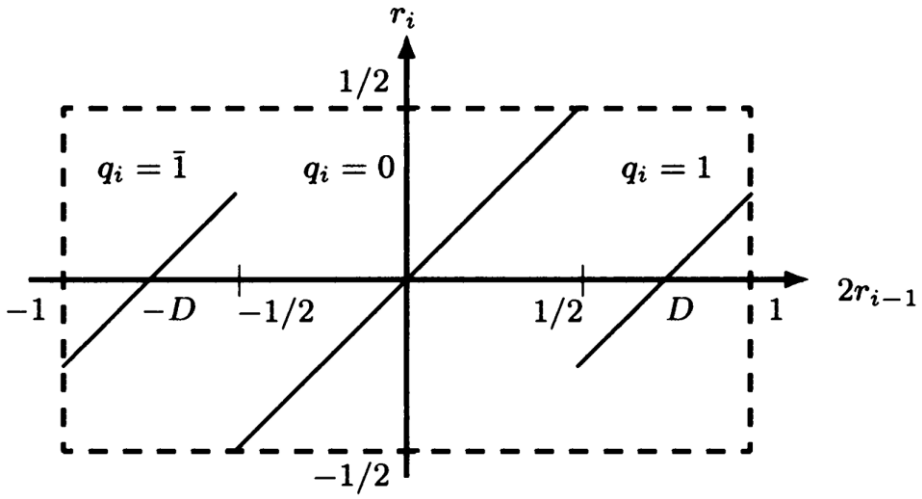
$$q_i = \begin{cases} 1 \text{ εάν } 2PR_{j-1} \geq \frac{1}{2} \\ 0 \text{ εάν } -\frac{1}{2} \leq 2PR_{j-1} < \frac{1}{2} \\ \bar{1} \text{ εάν } 2PR_{j-1} < -\frac{1}{2} \end{cases} \quad (2.4.7)$$

Οι κανόνες επιλογής των διάφορων παραλλαγών αλγορίθμων με διαδοχικές αφαιρέσεις που αναπτύχθηκαν, παρουσιάζονται συγκεντρωτικά στα P-D διαγράμματα της εικόνας 2.4.9.



Εικόνα 2.4.9 P-D διαγράμματα: (a) Restoring, (b) non-Restoring, (c) non-Restoring με $q_i = \{\bar{1}, 0, 1\}$, (d) SRT.

Ακολουθεί το διάγραμμα Robertson για τον *SRT*, όπου μπορούμε να διαπιστώσουμε ότι για κάθε *υπόλοιπο* $2PR_{j-1}$, πρέπει να ισχύει $\left|2PR_{j-1}\right| < 1$. Αυτή είναι η συνθήκη, ώστε να συγκλίνει ο *SRT* σε *τελικό υπόλοιπο* $R < D$.

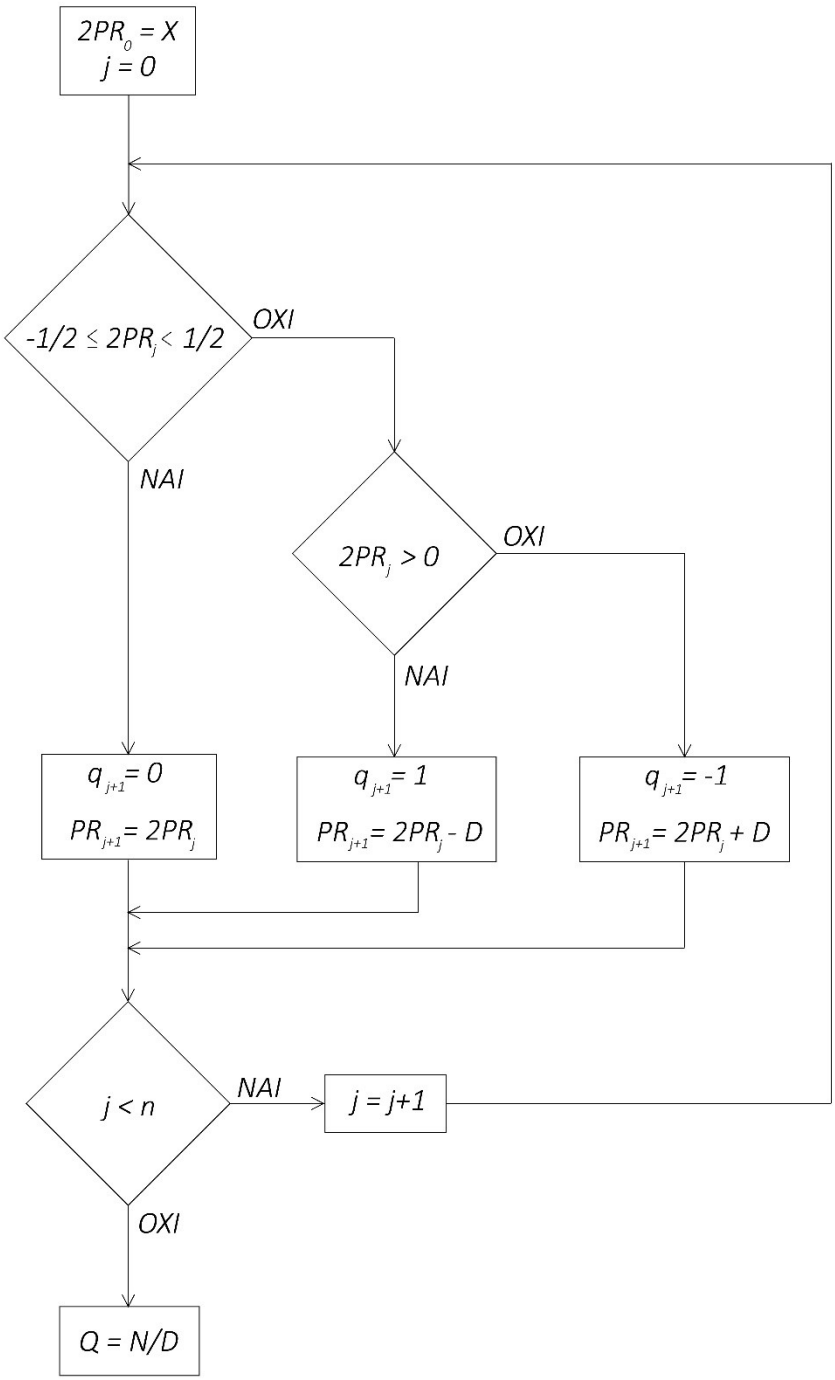


Εικόνα 2.4.10 Διάγραμμα Robertson αλγορίθμου non-Restoring με $q_i = 0$ [18] [7].

Σύμφωνα με τα ανωτέρω, για:

- $q_i = 1$ ο *διαιρέτης* αφαιρείται από το *υπόλοιπο* και το αποτέλεσμα ολισθαίνει αριστερά.
- $q_i = 0$ το *υπόλοιπο* ολισθαίνει αριστερά.
- $q_i = \bar{1}$ ο *διαιρέτης* προστίθεται στο *υπόλοιπο* και το αποτέλεσμα ολισθαίνει αριστερά.

Αυτά συνοψίζονται στο διάγραμμα ροής του αλγορίθμου *SRT*, στην εικόνα 2.4.11.



Εικόνα 2.4.11 Διάγραμμα ροής του αλγορίθμου SRT.

Στο παράδειγμα 2.4.9, εκτελείται η διαίρεση $0.1110/0.1010$ με χρήση του *SRT*. Η δυαδική αναπαράσταση είναι σε *συμπλήρωμα ως προς δύο*. Το πεδίο αναπαράστασης του *μερικού υπολοίπου* για τον αλγόριθμο *SRT*, είναι $[-1, 1)$. Με ένα ψηφίο αριστερά και τέσσερα δεξιά της υποδιαστολής, μπορούν να αναπρασταθούν αριθμοί στο πεδίο τιμών [-1, 0.9375], που είναι ίδιο με το πεδίο αναπαράστασης του *μερικού υπολοίπου*, για τον αλγόριθμο *SRT* και με ίδια διάταξη ψηφίων.

παράδειγμα 2.4.9
Εκτέλεση της διαίρεσης $0.1110/0.1010$, με χρήση του αλγορίθμου *SRT*.

	2's comp						ulp	decimal		
X		0	,	1	1	1	0	0,875		
D		0	,	1	0	1	0	0,625		
X=2PR0		0	,	1	1	1	0	0,875	2PR0>=1/2	q1=1
-D	+	1	,	0	1	0	1	1	-0,625		
PR1		0	,	0	1	0	0	0,25	PR1=2PR0-D	
2PR1		0	,	1	0	0	0	0,5	2PR1>=1/2	q2=1
-D	+	1	,	0	1	0	1	1	-0,625	
PR2		1	,	1	1	1	0	-0,125	PR2=2PR1-D	
PR3=2PR2		1	,	1	1	0	0	-0,25	-1/2<= 2PR2 <1/2	q3=0
PR4=2PR3		1	,	1	0	0	0	-0,5	-1/2<= 2PR3 <1/2	q4=0
2PR4		1	,	0	0	0	0	-1	2PR4<-1/2	q5=-1
+D	+	0	,	1	0	1	0	0,625		
PR5		1	,	1	0	1	0	-0,375	PR5=2PR4+D	

Στο συγκεκριμένο παράδειγμα εκτελείται ένα επιπλέον βήμα, ώστε να φάνει ότι αν και καθυστερημένα συγκλίνει στο σωστό αποτέλεσμα. Σε 4 κύκλους το πηλίκο προκύπτει $Q = 1.100$ αντί για $Q = 1.011$, αλλά σε 5 κύκλους γίνεται $Q = 1.100\bar{1} = 1.1\bar{1}11 = 1.0111$. Το *τελικό υπόλοιπο* προκύπτει $R = 1.1010_2 = -0.375_{10}$. Με τέσσερις ολισθήσεις δεξιά και κρατώντας την αρνητική αξία της πρώτης μονάδας, το *τελικό υπόλοιπο* γίνεται $R = 0.000\bar{1}101_2 = -0.0234375_{10}$.

Σε αντίθεση με τον *non-Restoring*, το σύνολο ψηφίων για το πηλίκο στον *SRT* είναι $\{\bar{1}, 0, 1\}$. Απαιτούνται λοιπόν δύο bits για την αναπαράσταση κάθε *RSD* ψηφίου του *πηλίκου*, και εκτός της «αργής» άθροισης των δύο συνιστωσών του, η άμεση μετατροπή τους γίνεται από τον αλγόριθμο *on-the-fly* [19].

Σύμφωνα με τον αλγόριθμο, εκφράζουμε δύο αριθμούς Q και Z μέχρι τη θέση j, ως $Q_j = 0.q_1q_2 \dots q_j$ και $Z_j = 0.z_1z_2 \dots z_j$. Ο αριθμός Z_j είναι η αντίστοιχη δυαδική αναπαράσταση του Q_j, εάν το μέρος του Q μετά τη θέση j είναι θετικό, και του $Q_j - 2^{-j}$, εάν το μέρος του Q μετά τη θέση j είναι αρνητικό. Κρατώντας λοιπόν τις δυαδικές αναπαραστάσεις Q_j και $Q_j - 2^{-j}$, μπορούμε να βρούμε τον Z_j για κάθε βήμα j. Έστω ZP_j και ZN_j αντίστοιχα, υπολογίζονται σε κάθε βήμα ως εξής:

- Εάν $q_j = 1$, $\begin{cases} ZP_j = ZP_{j-1} + 2^{-j} \\ ZN_j = ZP_{j-1} \end{cases}$
- Εάν $q_j = 0$, $\begin{cases} ZP_j = ZP_{j-1} \\ ZN_j = ZN_{j-1} + 2^{-j} \end{cases}$
- Εάν $q_j = \bar{1}$, $\begin{cases} ZP_j = ZN_{j-1} + 2^{-j} \\ ZN_j = ZN_{j-1} \end{cases}$

Στην εικόνα 2.4.12 παρουσιάζεται παράδειγμα εκτέλεσης του αλγορίθμου μετατροπής.

$q_1 = 1$ ZP_1 0. 1
ZN_1 0. 0

$q_2 = 1$ ZP_2 0. 1 1
ZN_2 0. 1 0

$q_3 = 1$ ZP_3 0. 1 1 1
ZN_3 0. 1 1 0

$q_4 = \bar{1}$ ZP_4 0. 1 1 0 1
ZN_4 0. 1 1 0 0

$q_5 = 0$ ZP_5 0. 1 1 0 1 0
ZN_5 0. 1 1 0 0 1

$q_6 = \bar{1}$ ZP_6 0. 1 1 0 0 1 1
ZN_6 0. 1 1 0 0 1 0

Quotient: $Z = ZP_6 = [0.110011]$

Εικόνα 2.4.12 Παράδειγμα εκτέλεσης του αλγορίθμου on-the-fly conversion [26].

Τέλος, απαιτείται προσοχή στην αντιμετώπιση της *υπερχείλισης αναπαράστασης*, που αναφέρθηκε στα συστήματα *RSD*.

2.4.4.. Σημαντικές παράμετροι σχεδιασμού

Οι βασικές αποφάσεις στην σχεδίαση ενός τυπικού διαιρέτη με διαδοχικές αφαιρέσεις είναι, η επιλογή βάσης, οι επιτρεπόμενες τιμές για το *ψηφίο πηλίκου* και το σύστημα αναπαράστασης του *μερικού υπολοίπου*. Πρέπει να βρεθεί ένας συμβιβασμός, μεταξύ του αριθμού των ψηφίων που υπολογίζονται σε κάθε κύκλο και της καθυστέρησης που προσθέτει αυτό. Ακόμα, είναι σημαντική η κατάλληλη επιλογή του συνόλου ψηφίων για το *πηλίκο*. Τέλος, μία πολυπλοκότερη αναπαράσταση του *υπολοίπου* μπορεί να μειώσει τον χρόνο κάθε επανάληψης, επιτρέποντας την αποφυγή *διάδοσης κρατουμένου* με την αποθήκευσή του στο *υπόλοιπο* [12].

2.4.4... Επιλογή συνόλου ψηφίων του πηλίκου

Το *σύνολο ψηφίων* για το *πηλίκο*, είναι καίριο για την διαμόρφωση ενός διαιρέτη. Όπως είδαμε, οι τρεις αλγόριθμοι με διαδοχικές αφαιρέσεις διαφοροποιούνται σύμφωνα με το σύνολο ψηφίων του *πηλίκου*. Ο αλγόριθμος *Restoring*, χρησιμοποιεί το *τυπικό σύνολο ψηφίων* $\{0, \dots, r-1\}$. Ο αλγόριθμος *non-Restoring*, χρησιμοποιεί προ*σημασμένα ψηφία δίχως περίσσεια*, εφόσον απουσιάζει το 0, $\{\bar{1}, 1\}$. Ο αλγόριθμος *SRT*, χρησιμοποιεί *προσημασμένα ψηφία με περίσσεια* $\{\bar{\alpha}, \dots, \bar{1}, 0, 1, \dots, \alpha\}$. Συνεπώς το *σύνολο ψηφίων*, καθορίζει αν ο διαιρέτης θα μπορεί να αναπαραστήσει το *υπόλοιπο* με *περίσσεια*, ενώ το α δεν είναι δεδομένο για κάθε *βάση*. Γενικά εάν $\alpha = r - 1$, το σύστημα έχει *μέγιστη περίσσεια*, και η *συνάρτηση επιλογής* προκύπτει απλούστερη σε σχέση με την *ελάχιστη περίσσεια*, καθώς τα διαστήματα επικάλυψης είναι μεγαλύτερα.

Μπορούμε να παρατηρήσουμε την διαφορά αυτή στα διαγράμματα Taylor της εικόνας 2.4.13 (2a,3a), όπου για $βάση = 4$, τα διαστήματα επικάλυψης είναι μεγαλύτερα για $\alpha = 3$ από αυτά που προκύπτουν για $\alpha = 2$. Αντίστοιχα, μπορούμε να προσέξουμε στα έγχρωμα διαγράμματα Taylor της εικόνας 2.4.13 (2b,3b), ότι χρειαζόμαστε μικρότερο *πινέλο (brush)* για $\alpha = 2$, και συνεπώς πολυπλοκότερη *συνάρτηση επιλογής*. Εντούτοις, τα

πολλαπλάσια του *διαιρέτη*, παράγονται πιο δύσκολα με την αύξηση της *περίσσειας*. Παρότι για *μέγιστη περίσσεια*, η επιλογή του ψηφίου είναι πάνω από 20% ταχύτερη [27], ο υπολογισμός του $3 \cdot D$ απαιτεί επιπλέον καθυστέρηση και επιφάνεια [17].

2.4.4... Επιλογή αναπαράστασης υπολοίπου

Το *μερικό υπόλοιπο* σε έναν διαιρέτη *SRT*, μπορεί να αναπαρασταθεί είτε *δίχως περίσσεια*, είτε με αναπαράσταση *προσημασμένου ψηφίου με περίσσεια* [3] [28] [12] [29] [25] [30]. Σε μία αναπαράσταση *δίχως περίσσεια*, κάθε πρόσθεση ή αφαίρεση πρέπει να *διαδόσει το κρατούμενο* από το *λιγότερο* στο *περισσότερο σημαντικό bit*, αυξάνοντας τον χρόνο κάθε κύκλου [17]. Σε μία αναπαράσταση *με περίσσεια* όμως, το *κρατουμένο* αποθηκεύεται στην αναπαράσταση του αποτελέσματος και διαδίδεται κάθετα αντί για οριζόντια. Χρησιμοποιείται δηλαδή *αθροιστής αποθήκευσης κρατουμένου*, και κάθε πρόσθεση/αφαίρεση καθυστερεί όσο ένας *πλήρης αθροιστής*.

Από την άλλη, σε μία αναπαράσταση *με περίσσεια*, το *σύνολο ψηφίων* είναι μεγαλύτερο από το *τυπικό*. Έτσι, κάθε ψηφίο χρειάζεται περισσότερα bits για να αναπαρασταθεί. Επιπλέον, πρέπει να παραχθούν περισσότερα πολλαπλάσια του *διαιρέτη* και να συγκριθούν με το ολισθημένο *μερικό υπόλοιπο* [20], οπότε η *συνάρτηση επιλογής* είναι πιο περίπλοκη [12].

Τέλος, τα δεδομένα στην είσοδο και στην έξοδο, αναπαρίστανται γενικά σε *συμπλήρωμα ως προς δύο*, οπότε είναι απαραίτητη η μετατροπή τους από και προς τη μορφή αυτή. Ειδικότερα για τους αριθμούς *κινητής-υποδιαστολής*, που διαιρούνται τα *μη προσημασμένα σημαντικά μέρη*, χρειάζεται μόνο η μετατροπή του *πηλίκου* από αναπαράσταση *προσημασμένου ψηφίου με περίσσεια* σε *συμπλήρωμα ως προς δύο*.

Συνοψίζοντας , ο αλγόριθμος *SRT* επιταχύνει τη διαδικασία της διαίρεσης σε σχέση με τον *non-Restoring*, καθώς η *διάδοση κρατουμένου* σε όλο το μήκος μίας άθροισης, απαιτεί περισσότερο χρόνο από μία σχετικά αργή *συνάρτηση επιλογής* του q.

2.4.4... Επιλογή βάσης

Κάθε αλγόριθμος με διαδοχικές αφαιρέσεις, υπολογίζει ένα ψηφίο σε κάθε κύκλο της επαναληπτικής διαδικασίας. Με τη χρήση του δυαδικού συστήματος, κάθε ψηφίο αναπαρίσταται από ένα bit. Σε οποιοδήποτε άλλο σύστημα όμως, που χρησιμοποιείται στις ψηφιακές υλοποιήσεις διαιρετών, με $βάση = 2^{\delta}$, κάθε ψηφίο αναπαρίσταται από $\delta = log_2 βάση$ bits. Δηλαδή σε κάθε κύκλο, υπολογίζονται δ bits. Ο απαραίτητος χρόνος για να ολοκληρωθεί η διαίρεση είναι:

$$T = t_j \cdot N \qquad (2.4.8)$$

, όπου:

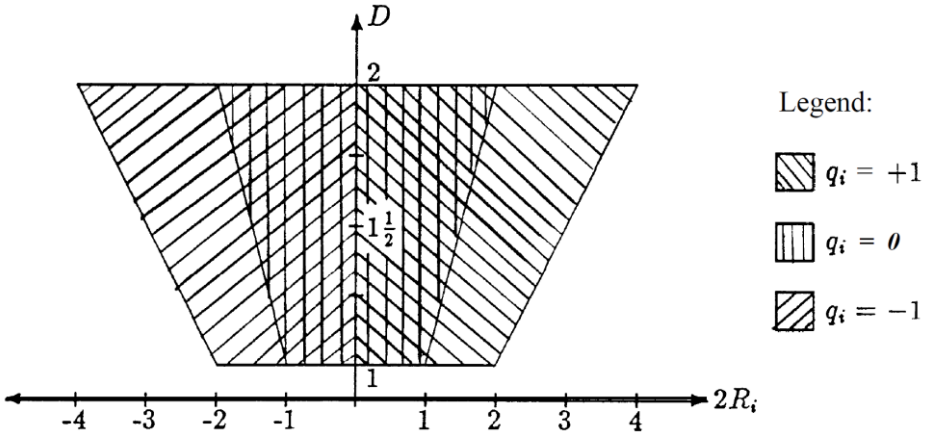
- t_j η καθυστέρηση κάθε βήματος.
- $N = αριθμός\ απαραίτητων\ βημάτων = n/\delta$.

Το t_j εξαρτάται κυρίως, από την πολυπλοκότητα της *συνάρτησης επιλογής* του *ψηφίου πηλίκου*, και την εύρεση πολλαπλάσιων του *διαιρέτη* που δεν προκύπτουν με απλή ολίσθηση. Η σύγκριση μεταξύ *διαιρέτη* και *μερικού υπολοίπου*, γίνεται όλο και πιο πολύπλοκη με την αύξηση της *βάσης* άρα και περισσότερο χρονοβόρα, όπως φαίνεται στα

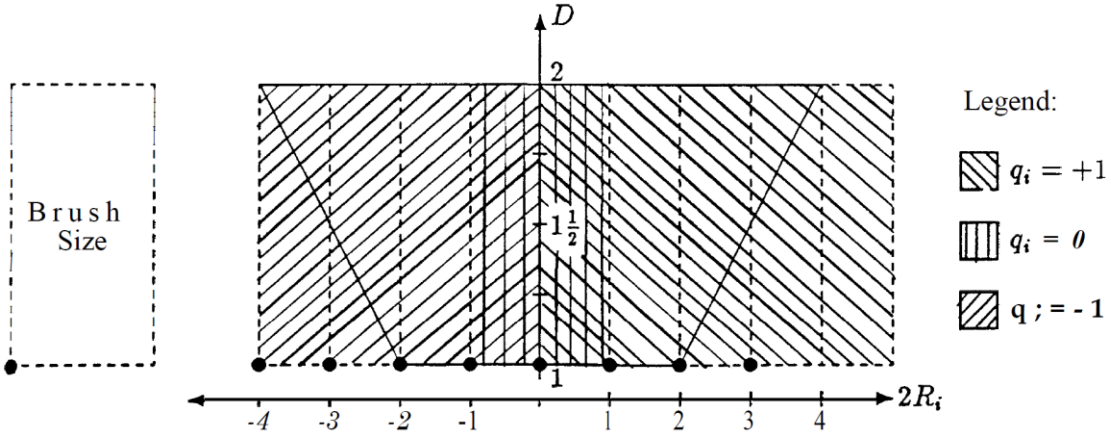
διαγράμματα Taylor της εικόνας 2.4.13 (1a σε σχέση με 2a ή 3a). Συνηθίζεται λοιπόν να χρησιμοποιείται *πίνακας αναφοράς* για την εύρεση του q *(lookup table)*, ο οποίος υλοποιείται σε μνήμη *ROM*. Επίσης με την αύξηση της *βάσης*, αυξάνεται και το *σύνολο ψηφίων* του q. Τα πολλαπλάσια του q που είναι διάφορα κάποιας δύναμης του δύο, απαιτούν πολλαπλασιασμό με τον *διαιρέτη* αντί για ολίσθηση, που αυξάνει επιπλέον την καθυστέρηση.

Από την άλλη, ο προφανής τρόπος να μειωθεί η καθυστέρηση του κυκλώματος, είναι να μειωθεί το N, που μπορεί να επιτευχθεί με τη χρήση μεγαλύτερων *βάσεων*. Μία τετραγωνική αύξηση της *βάσης*, υποδιπλασιάζει το N. Παρ' όλα αυτά δεν υποδιπλασιάζει την καθυστέρηση, καθώς αυξάνεται το t_j. Οπότε τελικά η ελαχιστοποίηση της καθυστέρησης, δεν είναι απλή διαδικασία και πρέπει να βρεθεί ένας συμβιβασμός που να ικανοποιεί τις απαιτήσεις της υλοποίησης.

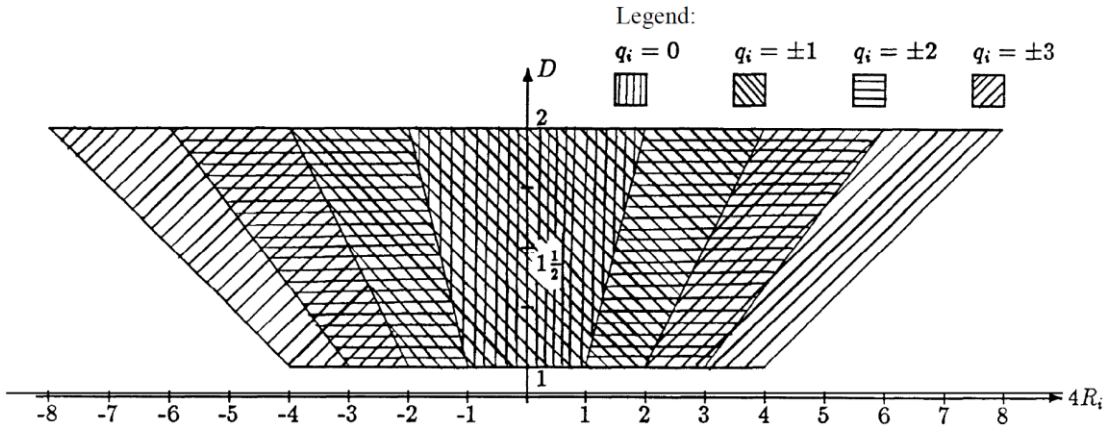
Τέλος, η καθυστέρηση του *πίνακα αναφοράς* αυξάνει γραμμικά με την αύξηση της *βάσης*, ενώ η επιφάνεια κυκλώματος αυξάνει τετραγωνικά [27]. Όπως παρουσιάζεται στη συνέχεια, το πρόβλημα αυτό αντιμετωπίζεται είτε με *Prescaling* είτε με τις υλοποιήσεις *μεταβλητής καθυστέρησης*.



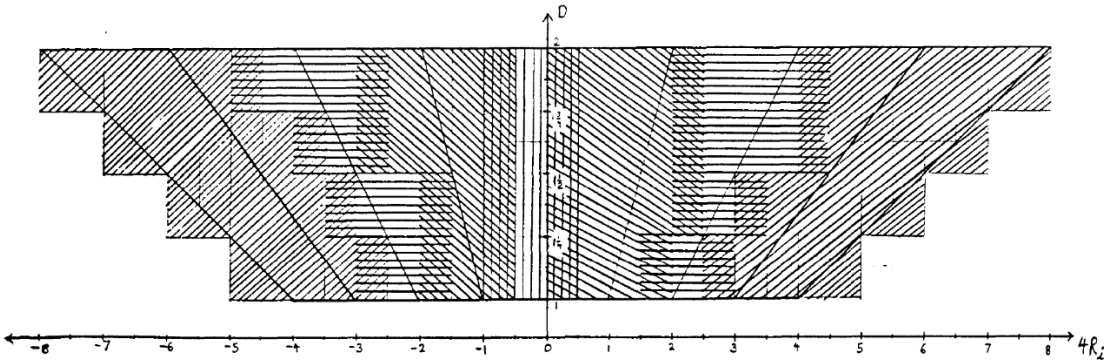
(1a) Διάγραμμα Taylor για βάση 2, με σύνολο ψηφίων $q_i = \{\bar{1}, 0, 1\}$.



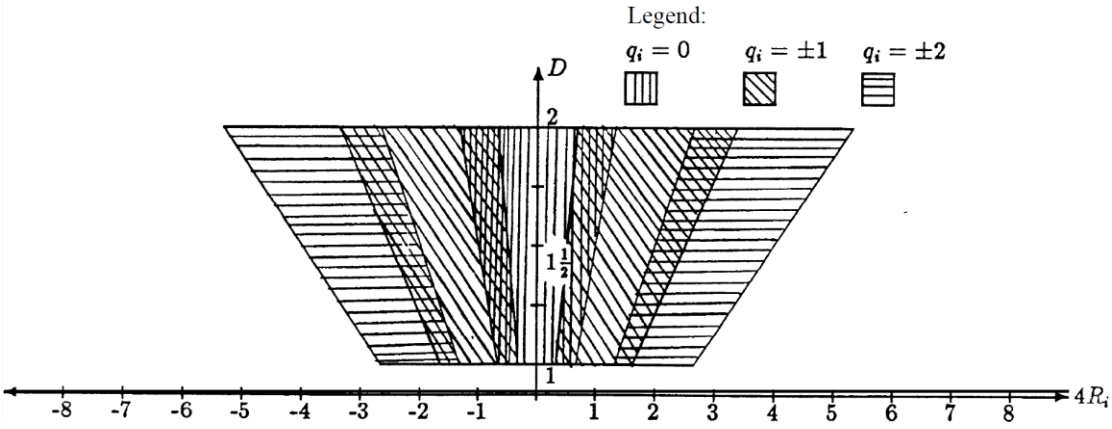
(1b) Έγχρωμο διάγραμμα Taylor για βάση 2, με σύνολο ψηφίων $q_i = \{\bar{1}, 0, 1\}$.



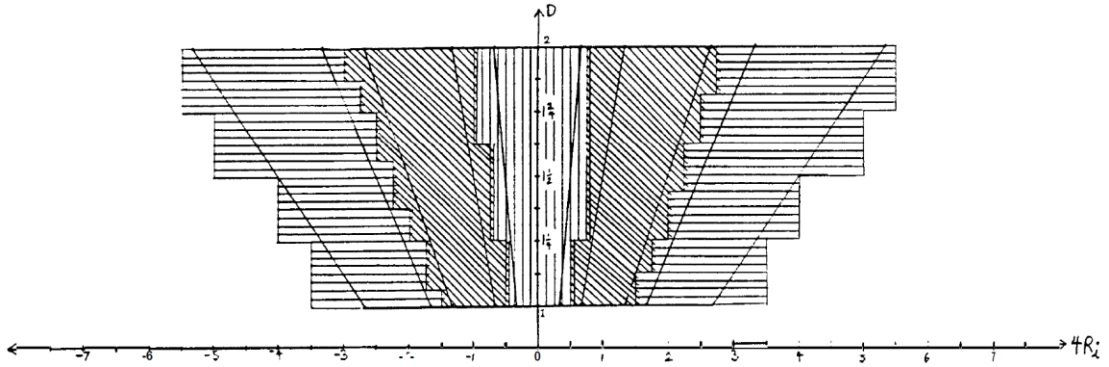
(2a) Διάγραμμα Taylor για βάση 4, με σύνολο ψηφίων $q_i = \{\bar{3}, \bar{2}, \bar{1}, 0,1,2,3\}$.



(2b) Έγχρωμο διάγραμμα Taylor για βάση 4, με σύνολο ψηφίων $q_i = \{\bar{3}, \bar{2}, \bar{1}, 0,1,2,3\}$.



(3a) Διάγραμμα Taylor για βάση 4, με σύνολο ψηφίων $q_i = \{\bar{2}, \bar{1}, 0,1,2\}$.

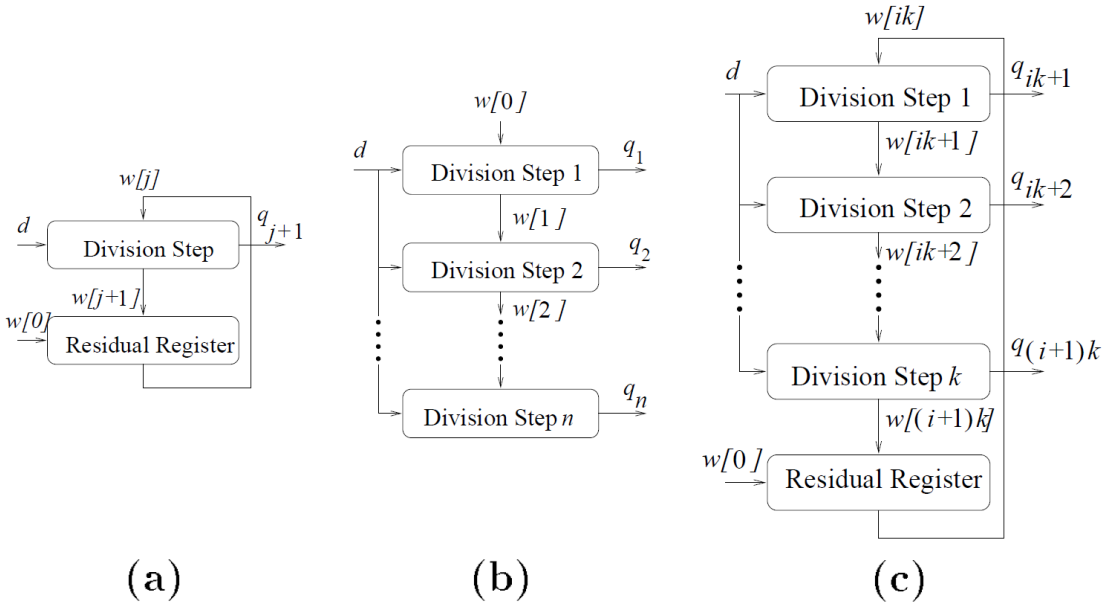


(3b) Έγχρωμο διάγραμμα Taylor για βάση 4, με σύνολο ψηφίων $q_i = \{\bar{2}, \bar{1}, 0,1,2\}$.

Εικόνα 2.4.13 Διαγράμματα Taylor (a), έγχρωμα Taylor (b), για βάση 2 (1), και βάση 4 με σύνολο ψηφίων qi={-3,-2 ,-1 ,0,1,2,3} (2) και qi={-2 ,-1 ,0,1,2} (3). Ο διαιρέτης είναι κανονικοποιημένος στο διάστημα [1,2) [31].

2.4.4.. Τρόποι υλοποίησης

Υπάρχουν τρεις διαφορετικοί τρόποι υλοποίησης, όπως φαίνονται στο σχήμα. Η επιλογή εξαρτάται από το κόστος, την καθυστέρηση και την *διεκπεραιωτικότητα (throughput)*. Η υλοποίηση μπορεί να είναι *καθαρά ακολουθιακή* (a), όταν το υλικό που εκτελεί ένα βήμα της διαίρεσης επαναχρησιμοποιείται σε κάθε επανάληψη, και το *μερικό υπόλοιπο* ανανεώνεται σε έναν καταχωρητή. Εκτελείται έτσι σε N κύκλους. Μπορεί να είναι *καθαρά συνδυαστική* (b), όταν το υλικό αυτό επαναλαμβάνεται διαμορφώνοντας έτσι έναν πίνακα, οπότε και εκτελείται σε έναν μεγάλο κύκλο. Ή μπορεί να είναι *συνδυασμός* των δύο (c), όπου το υλικό επαναλαμβάνεται k φορές και σε κάθε κύκλο εξάγονται k *ψηφία πηλίκου*. Έτσι ο συνολικός αριθμός κύκλων γίνεται N/k [32].



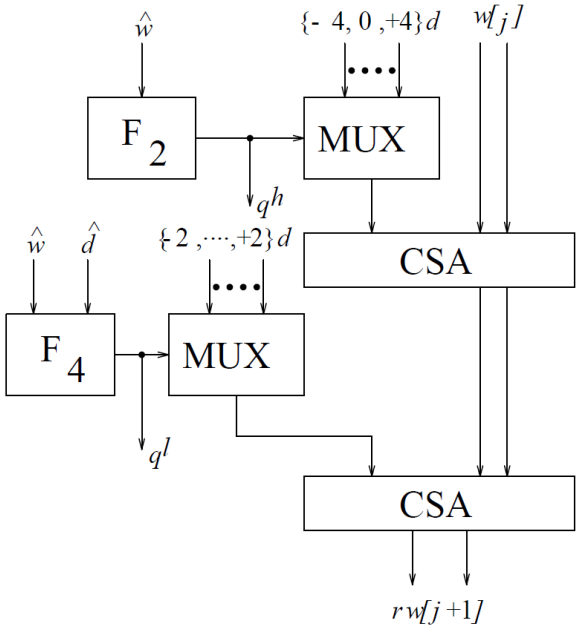
Εικόνα 2.4.14 Τρόποι υλοποίησης διαίρεσης: καθαρά ακολουθιακή (a), καθαρά συνδυαστική (b), συνδυασμός των δύο (c) [32].

Σε χρήση μεγαλύτερων *βάσεων*, είναι ασύμφορο να χρησιμοποιηθεί συνδυαστική λογική για την *συνάρτηση απόφασης* του q. Έτσι, οι συνδυαστικοί πίνακες περιορίζονται σε απλές υλοποιήσεις με βάση το 2. Σε εφαρμογές που είναι σημαντική η *διεκπεραιωτικότητα* και πρέπει να ολοκληρώνεται μία διαίρεση ανά κύκλο, μπορούν να είναι χρήσιμοι. Ένας συνδυαστικός πίνακας με διοχέτευση, ικανοποιεί τη συνθήκη αυτή, ενώ η πολυπλοκότητα και η καθυστέρηση κάθε βήματος είναι πολύ χαμηλές [14].

Η *ακολουθιακή* υλοποίηση όμως, πλεονεκτεί στο κόστος ,αφού χρειάζεται το ελάχιστο υλικό. Για την υλοποίηση μεγαλύτερων *βάσεων*, χρησιμοποιείται ένας *πίνακας αναφοράς* για την επιλογή του q. Επίσης έχουν αναπτυχθεί υλοποιήσεις μεταβλητής καθυστέρησης, που χρησιμοποιούν *ακολουθιακή* υλοποίηση. Σε αυτές, αντί για τον ακριβή προσδιορισμό του *ψηφίου πηλίκου*, βρίσκεται μία γρήγορη προσέγγιση. Κατόπιν γίνεται έλεγχος για λάθος και αν υπάρχει διορθώνεται. Οι υλοποιήσεις αυτές εκμεταλλεύονται τη δυναμική δρομολόγηση εντολών στους σύγχρονους επεξεργαστές και στοχεύουν στη μείωση της συνολικής καθυστέρησης πλήθους εντολών. Αυξάνεται η καθυστέρηση στη χειρότερη περίπτωση, αλλά μειώνεται αρκετά στην καλύτερη περίπτωση. Έτσι, μπορεί να επιτευχθεί καλύτερη μέση καθυστέρηση εκτέλεσης ανά διαίρεση, σε σχέση με την αντίστοιχη υλοποίηση σταθερής καθυστέρησης.

2.4.4.. Τρόποι βελτίωσης αλγορίθμων με διαδοχικές αφαιρέσεις

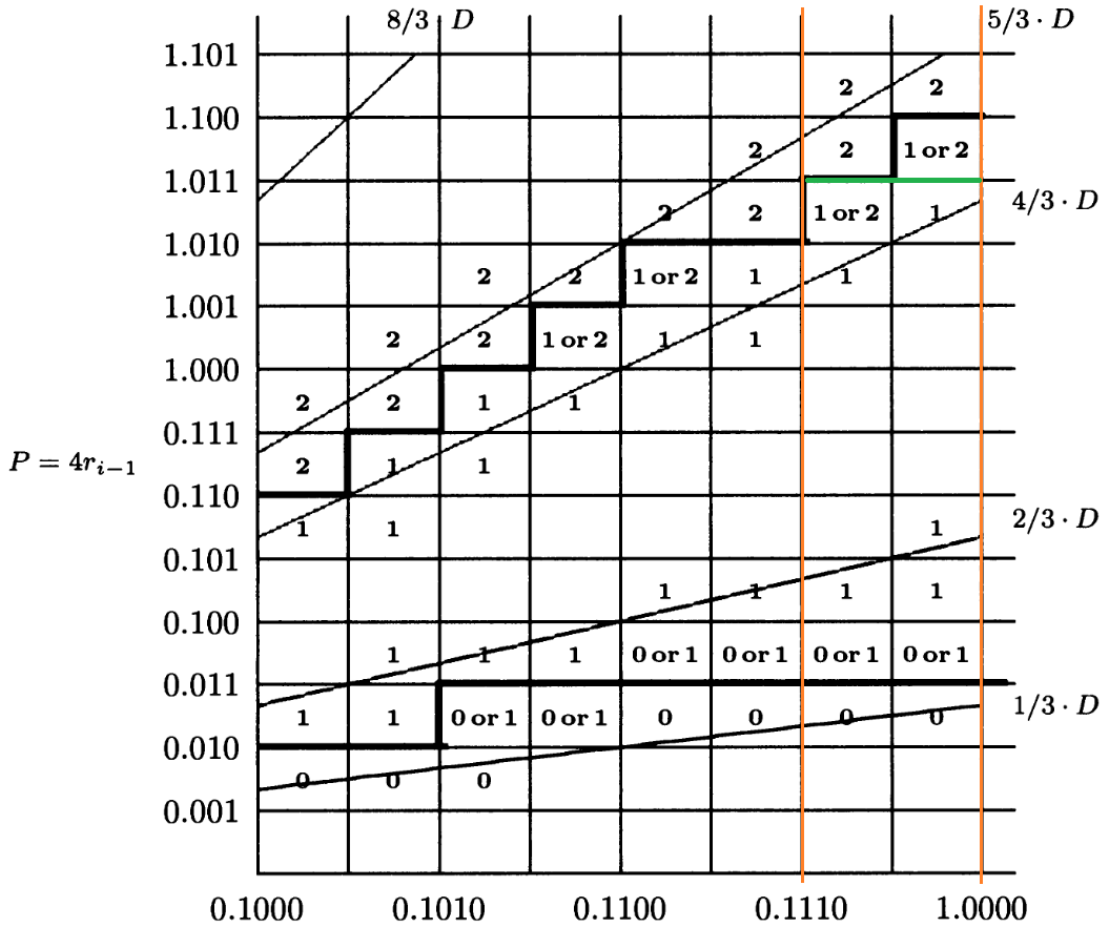
Έχουν αναπτυχθεί διάφορες τεχνικές, για την επιτάχυνση των αλγορίθμων με διαδοχικές αφαιρέσεις [12] [32] [17]. Μία από αυτές, είναι η αύξηση της *βάσης*. Όπως αναφέρθηκε, οι πρακτικές υλοποιήσεις περιορίζονται στα συστήματα με *βάσεις* 2 ή 4, μπορούμε όμως να συνθέσουμε διαιρέτες μεγαλύτερης *βάσης*, συνδυάζοντας στάδια μικρότερης *βάσης* [33] [34]. Για παράδειγμα, θα μπορούσαμε να υλοποιήσουμε ένα στάδιο της διαίρεσης με *βάση* το 8 και $\rho = \frac{6}{7}$, επικαλύπτοντας δύο στάδια με βάση το 2 και το 4, όπως στην εικόνα 2.4.15.



Εικόνα 2.4.15 Υλοποίηση SRT με βάση το 8 [32].

Το ψηφίο του *πηλίκου*, προκύπτει από το άθροισμα των δύο συνισταμένων, $q_{j+1} = q^h + q^l$. Το ψηφίο $q^h \in \{\bar{4}, 0, 4\}$ παράγεται από την *συνάρτηση επιλογής* F_2, ενώ το ψηφίο $q^l \in \{\bar{2}, \bar{1}, 0, 1, 2\}$ παράγεται από την *συνάρτηση επιλογής* F_4. Συνεπώς, $q_{j+1} \in \{\bar{6}, \dots, 6\}$.

Μία άλλη τεχνική για την αύξηση της *βάσης*, είναι η μείωση της πολυπλοκότητας της *συνάρτησης επιλογής*, μέσω της *κανονικοποίησης* των ορισμάτων σε προκαθορισμένο πεδίο τιμών (operands Prescaling) [35] [36] [37] [38] [39]. Ας υποθέσουμε για παράδειγμα ότι έχουμε το *P-D* γράφημα της εικόνας 2.4.16, που αφορά *SRT* με $\beta\acute{\alpha}\sigma\eta = 4, \alpha = 2$, $D \in [0.5, 1)$. Εάν *κανονικοποιήσουμε* τα δύο ορίσματα, έτσι ώστε να ισχύει $D \in [0.875, 1)$ (πορτοκαλί), η τεθλασμένη γραμμή στη συγκεκριμένη περιοχή, μπορεί να αντικατασταθεί από μία ευθεία (πράσινο). Κατ' αυτόν τον τρόπο, η *συνάρτηση επιλογής* καθίσταται ανεξάρτητη από τον *διαιρέτη* -με κόστος την επιπλέον καθυστέρηση της *κανονικοποίησης*- και μπορεί να υλοποιηθεί πολύ απλούστερα, και άρα να είναι ταχύτερη. Πάραυτα, η καθυστέρηση υπολογισμού πολλαπλάσιων του *διαιρέτη*, που δεν προκύπτουν με ολίσθηση, περιορίζει τις υλοποιήσεις στις *βάσεις* 2 ή 4 [17] [40].



Εικόνα 2.4.16 P-D γράφημα αλγορίθμου *SRT* με βάση = 4, α = 2, D∈[0.5,1), τροποποιημένο ώστε να φαίνεται ο ρόλος του Prescaling [7].

Μία επιπλέον σημαντική τεχνική, είναι η επικάλυψη τμημάτων κάθε σταδίου για την μείωση της καθυστέρησής του, με χρήση επιπλέον υλικού, που αναλύεται στην εργασία [17]. Η επικάλυψη της *συνάρτησης επιλογής* διαδοχικών σταδίων προτείνεται από τον Taylor [34]. Οι Oberman [27] και Quach [41] συζητούν την επικάλυψη του υπολογισμού του *μερικού υπολοίπου*. Επιπλέον, αναλύεται ο συνδυασμός της επικάλυψης του *μερικού υπολοίπου* και της *συνάρτησης επιλογής*. Ακόμα, μία υβριδική επικάλυψη επιταχύνει περαιτέρω τον αλγόριθμο επικαλύπτοντας *μερικό υπόλοιπο* και *συνάρτηση επιλογής*, για τα *πιο σημαντικά bits*. Τέλος, εκτός από τις ανωτέρω τεχνικές, μπορεί να απλοποιηθεί η *συνάρτηση επιλογής* του *ψηφίου πηλίκου*, εάν υπολογίζουμε μία εκτίμηση του *μερικού υπολοίπου* σε *συμπλήρωμα ως προς δύο* [32]. Μειώνουμε δηλαδή στο μισό τις εισόδους της συνάρτησης επιλογής, με κόστος την διάδοση κρατουμένου σε λίγα ψηφία.

2.4.4. Αλγόριθμοι σύγκλισης με επαναληπτική μέθοδο

Η ανάπτυξη ταχύτατων δυαδικών πολλαπλασιαστών [42], αναζωπύρωσε το ενδιαφέρον για την διερεύνηση αλγορίθμων διαίρεσης, που χρησιμοποιούν σαν κύρια επαναληπτική πράξη τον πολλαπλασιασμό [15]. Στην κατηγορία αυτή αλγορίθμων, το *πηλίκο* ή ο *αντίστροφος του διαιρέτη*, εκπροσωπείται από μία συνάρτηση ή από ένα ανάπτυγμα σειράς. Το πρόβλημα ανάγεται έτσι σε μία επαναληπτική διαδικασία εύρεσης της ρίζας της συνάρτησης ή του αναπτύγματος της σειράς. Οι σημαντικότερες υλοποιήσεις,

κάνουν χρήση της μεθόδου Newton-Raphson και της σειράς Maclaurin (ειδική περίπτωση των σειρών Taylor) [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53].

Το πλεονέκτημα των αλγορίθμων αυτών, είναι ότι συγκλίνουν στο αποτέλεσμα ταχύτερα από γραμμικά, συνήθως τετραγωνικά, και έτσι διπλασιάζουν τον αριθμό ευρισκόμενων bits ανά κύκλο. Μάλιστα, έχει παρουσιασθεί μέθοδος που συγκλίνει κυβικά στο αποτέλεσμα [54], αλλά απαιτεί πολυπλοκότερη αριθμητική, που περιορίζει το κέρδος της σε σχέση με τις απλούστερες μεθόδους.

Το μειονέκτημά τους είναι η αυξημένη πολυπλοκότητα, αφού οι τυπικές υλοποιήσεις απαιτούν δύο πολλαπλασιασμούς ανά κύκλο αντί για μία απλή αφαίρεση. Ακόμη, επειδή το αποτέλεσμα αποτελεί ρίζα της συνάρτησης, το *τελικό υπόλοιπο* δεν είναι άμεσα διαθέσιμο. Παρότι όμως κάθε βήμα είναι απαιτητικό σε υλικό, και το *τελικό υπόλοιπο* δεν διατίθεται, τα κέρδη σε ταχύτητα μπορεί να είναι σημαντικά και υπάρχουν περιπτώσεις που το *υπόλοιπο* δεν είναι απαραίτητο, με αποτέλεσμα να συναντώνται αρκετές εμπορικές υλοποιήσεις [12].

2.4.4.. Διαίρεση με ανάπτυγμα σειράς Maclaurin

Η διαίρεση αυτή βασίζεται στις σειρές Maclaurin, που είναι ειδική περίπτωση των σειρών Taylor. Η υλοποίηση, μπορεί να αναπτύσσει μία σειρά που συγκλίνει στον *αντίστροφο του διαιρέτη*, τον οποίο θα πολλαπλασιάσει με τον *διαιρετέο* στο τέλος της πράξης. Ο *διαιρέτης* τίθεται ίσος με $1 + x$, και αναπτύσσεται ο *αντίστροφός* του. Έχουμε:

$$g(X) = \frac{1}{b} = \frac{1}{1+X} = 1 - X + X^2 - X^3 + X^4 - \cdots$$

Που μπορεί να παραγοντοποιηθεί ως:

$$g(X) = \frac{1}{b} = \frac{1}{1+X} = (1-X)(1+X^2)(1+X^4)(1+X^8)(1+X^{16}) \ldots$$

αφού $0.5 \leq b < 1$, και έτσι όλοι οι όροι είναι διάφοροι του μηδενός.

Το *συμπλήρωμα ως προς δύο* του $1 + X^n$, είναι $2 - (1 + X^n) = 1 - X^n$. Αντίστροφα, το συμπλήρωμα ως προς δύο του $1 - X^n$, είναι $2 - (1 - X^n) = 1 + X^n$. Επειδή ο αλγόριθμος συγκλίνει τετραγωνικά, μπορεί να επιταχυνθεί με μία αρχική εκτίμηση. Σε κάθε βήμα του αλγορίθμου, πολλαπλασιάζουμε στο *μερικό γινόμενο* έναν επιπλέον όρο, τον οποίο πρέπει πρώτα να σχηματίσουμε. Αυτά γίνονται αποδοτικά, όπως στον IBM 360/91 [55] [56], όπου για ακρίβεια 32 bits, ακολουθείται η παρακάτω διαδικασία:

1. Βρίσκεται μία εκτίμηση για το γινόμενο $(1 - X)(1 + X^2)(1 + X^4)$.
2. Πολλαπλασιάζεται με το $1 + X$, οπότε $(1 - X)(1 + X^2)(1 + X^4)(1 + X) = 1 - X^8$.
3. Με συμπλήρωμα ως προς δύο προκύπτει $1 + X^8$.
4. Με μεταξύ τους πολλαπλασιασμό $1 - X^{16} = (1 - X^8)(1 + X^8)$.
5. Με συμπλήρωμα ως προς δύο προκύπτει $1 + X^{16}$.
6. Με μεταξύ τους πολλαπλασιασμό $1 - X^{32} = (1 - X^{16})(1 + X^{16})$.
7. Με συμπλήρωμα ως προς δύο προκύπτει $1 + X^{32}$.

Παράλληλα με κάθε πολλαπλασιασμό στην ανωτέρω διαδικασία, με την εύρεση κάθε όρου ανανεώνεται το γινόμενο, που προκύπτει:

$$g(X) = \frac{1}{b} = \frac{1}{1+X} = (1-X)(1+X^2)(1+X^4)(1+X^8)(1+X^{16})(1+X^{32})$$

2.4.4.. Διαίρεση με την μέθοδο Newton-Raphson

Η μέθοδος Newton-Raphson είναι μία επαναληπτική διαδικασία, εύρεσης καλύτερων διαδοχικά προσεγγίσεων, για τη ρίζα μίας συνάρτησης $f(X) = 0$. Η διαδικασία ξεκινάει με μία αρχική εκτίμηση X_0. Η επόμενη εκτίμηση X_1, είναι το σημείο τομής της εφαπτομένης της $f(X)$ στο σημείο $\big(X_0, f(X_0)\big)$, με τον άξονα των Χ. Μπορούμε να γράψουμε:

$$f'(X_0) = \frac{\delta Y}{\delta X} = \frac{Y - f(X_0)}{X - X_0} = \frac{0 - f(X_0)}{X_1 - X_0}$$

Επιλύοντας ως προς X_1, προκύπτει:

$$X_1 = X_0 - \frac{f(X_0)}{f'(X_0)}$$

Και γενικότερα:

$$X_{i+1} = X_i - \frac{f(X_i)}{f'(X_i)}$$

Στην προκειμένη περίπτωση, ενδιαφέρει η εύρεση του *αντίστροφου του διαιρέτη*, και άρα η εύρεση της ρίζας της συνάρτησης $f(X) = \frac{1}{X} - b$. Η παράγωγος προκύπτει λοιπόν $f'(X) = -\left(\frac{1}{X}\right)^2$, και έτσι:

$$X_{i+1} = X_i - \frac{\frac{1}{X_i} - b}{-\left(\frac{1}{X_i}\right)^2} = X_i + X_i - bX_i{}^2 = X_i(2 - bX_i)$$

Μία βολική αρχική εκτίμηση, είναι $X_0 = 1$. Συνήθως όμως χρησιμοποιείται *πίνακας αναφοράς* για την επιτάχυνση του αλγορίθμου.

Και οι δύο μέθοδοι που αναφέρθηκαν, υλοποιούν αποδοτικά την διαίρεση. Κάθε βήμα της διαδικασίας απαιτεί δύο πολλαπλασιασμούς, αλλά η διαδικασία συγκλίνει τετραγωνικά στο αποτέλεσμα. Στην μέθοδο Newton-Raphson οι πολλαπλασιασμοί είναι εξαρτημένοι, σε αντίθεση με το ανάπτυγμα σειράς, που μπορούν να εκτελεστούν παράλληλα και να επιταχύνουν περαιτέρω την εκτέλεση.

Η αρχική εκτίμηση για τον *αντίστροφο του διαιρέτη*, βρίσκεται συνήθως με τη χρήση *πίνακα αναφοράς*. Μεγαλύτερη ακρίβεια στην αρχική εκτίμηση παρέχει ταχύτερη σύγκλιση, αλλά ταυτόχρονα απαιτεί περισσότερα bits στην είσοδο του πίνακα. Το μέγεθος του πίνακα όμως, αυξάνεται εκθετικά με τον αριθμό των εισόδων, και πρέπει να γίνει ένας

συμβιβασμός μεταξύ κόστους και ταχύτητας. Μία εναλλακτική για την αρχική εκτίμηση, είναι η χρήση πίνακα *μερικών γινομένων* [57]. Μπορεί να υλοποιηθεί με έναν υπάρχον πολλαπλασιαστή *κινητής-υποδιαστολής*, και παρέχει μεγάλη ακρίβεια χωρίς επιπλέον κόστος.

Οι αλγόριθμοι σύγκλισης με επαναληπτική μέθοδο, μπορούν να υλοποιηθούν με χρήση του υπάρχοντος υλικού, χωρίς να υποβαθμίσουν την ευρύτερη απόδοση του επεξεργαστή. Μπορούν να επιτύχουν χαμηλότερη καθυστέρηση από μία υλοποίηση *SRT*, αλλά δεν παρέχουν άμεσα *τελικό υπόλοιπο*, και η στρογγυλοποίηση του αποτελέσματος είναι πιο δύσκολη.

2.4.4. Διαιρέτες πολύ υψηλού radix

Οι συμβατικοί αλγόριθμοι με διαδοχικές αφαιρέσεις, αδυνατούν πρακτικά να χρησιμοποιήσουν πολύ υψηλή *βάση*, λόγω περιορισμών κόστους και καθυστέρησης. Έτσι, περιορίζονται στην εύρεση μικρού αριθμού ψηφίων, σε κάθε απλό σχετικά κύκλο. Από την άλλη, οι αλγόριθμοι τετραγωνικής σύγκλισης μπορούν να επιτύχουν χαμηλότερη καθυστέρηση, με τίμημα το *τελικό υπόλοιπο* και την ακρίβεια στρογγυλοποίησης [12].

Όποτε η ταχύτητα, η εύρεση *υπολοίπου* και η ακρίβεια του αποτελέσματος είναι ταυτόχρονα σημαντικές, μπορεί να υλοποιηθεί ένας αλγόριθμος με διαδοχικές αφαιρέσεις αλλά υπό διαφορετική οπτική γωνία. Η φιλοσοφία διαδοχικών αφαιρέσεων από το *μερικό υπόλοιπο* παραμένει ίδια, άλλα ο υπολογισμός *ψηφίου πηλίκου* και *πολλαπλάσιου του διαιρέτη*, θυμίζει τεχνικές των αλγορίθμων σύγκλισης. Η ταχύτητα επιτυγχάνεται με τη χρήση πολύ υψηλής *βάσης*, συνήθως μεγαλύτερης του 2^{10}.

Για την επίτευξη υλοποίησης τέτοιων διαιρετών, με πολύ υψηλή *βάση*, αποδεκτή καθυστέρηση κύκλου, και αποδεκτό κόστος, χρησιμοποιείται αποδοτικός πολλαπλασιασμός σε κάθε κύκλο, για τον σχηματισμό του *ψηφίου πηλίκου* και των *πολλαπλάσιων του διαιρέτη*. Ακόμη, χρησιμοποιείται ένας *πίνακας αναφοράς* για μία αρχική εκτίμηση του *αντίστροφου του διαιρέτη*. Οι υλοποιήσεις διαφέρουν στον αριθμό και τον τύπο των πράξεων σε κάθε κύκλο, και στην *συνάρτηση επιλογής* του q.

Το πλεονέκτημα των διαιρετών πολύ υψηλής *βάσης*, είναι η απόφαση πολλών bits του *πηλίκου* σε κάθε κύκλο, και ο υπολογισμός του *τελικού υπολοίπου*. Απαιτούν μεγαλύτερη πολυπλοκότητα σε σχέση με τους αλγορίθμους χαμηλής *βάσης*, αλλά λιγότερη σε σχέση με τους αλγορίθμους σύγκλισης. Σημαντικοί αλγόριθμοι της κατηγορίας αυτής, είναι ο Accurate Quotient Approximations του Wong [58], ο Short Reciprocal που χρησιμοποιείται στον αριθμητικό συνεπεξεργαστή Cyrix 83D87 [59], και ο Rounding and Prescaling από τους Ercegovac και Lang [60] [61].

2.4.4. Διαιρέτες μεταβλητής καθυστέρησης

Όλοι οι διαιρέτες που εξετάσαμε μέχρι τώρα, παράγουν το αποτέλεσμα σε συγκεκριμένο χρόνο και χαρακτηρίζονται από την καθυστέρηση της χειρότερης περίπτωσης. Υπάρχουν όμως ορίσματα, για τα οποία η διαίρεση, είτε εκτελείται ταχύτερα, είτε έχει ξαναϋπολογιστεί πρόσφατα. Οι σύγχρονοι επεξεργαστές μπορούν να εκμεταλλευτούν το γεγονός αυτό, μέσω της δυναμικής δρομολόγησης εντολών που

διαθέτουν. Μπορούν έτσι να εκτελούν μία διαίρεση με μεταβλητή καθυστέρηση, και να χρησιμοποιούν το αποτέλεσμα μόλις είναι έτοιμο. Κατ' αυτόν τον τρόπο, είναι πιθανό να βελτιώνουν τη μέση καθυστέρηση, και άρα τη συνολική απόδοση του συστήματος.

Ένας τρόπος υλοποίησης διαιρετών μεταβλητής καθυστέρησης, είναι με χρήση αυτόνομου χρονισμού (self-timing) [62]. Έτσι, ένας *SRT* διαιρέτης, θα μπορούσε να εκμεταλλευτεί τις ολισθήσεις όταν το *ψηφίο πηλίκου* επιλέγεται 0. Ένας άλλος τρόπος, είναι η αποθήκευση του αποτελέσματος κάθε διαίρεσης σε μία *προσωρινή μνήμη (result cache)*. Σε έναν τέτοιο διαιρέτη διαδοχικών αφαιρέσεων, θα πρέπει να φυλάσσονται τα ορίσματα και το αποτέλεσμα, και η απαιτούμενη επένδυση σε υλικό είναι μεγάλη. Σε μία υλοποίηση που υπολογίζει τον *αντίστροφο του διαιρέτη* όμως, πρέπει να αποθηκεύονται μόνο ο *διαιρέτης* και ο αντίστροφός του. Έτσι κάθε φορά που χρησιμοποιείται ο ίδιος *διαιρέτης*, η αναζήτηση στην *cache* θα είναι επιτυχής, σε αντίθεση με τους διαιρέτες διαδοχικών αφαιρέσεων, που πρέπει να ταυτίζονται και τα δύο ορίσματα [12].

Η τελευταία και πολύ σημαντική μέθοδος, είναι η *υπόθεση του ψηφίου πηλίκου (quotient digit speculation)* [63] [64] [65] [66] [67] [32]. Στους διαιρέτες αυτής της μορφής, εισάγεται η έννοια της υπόθεσης και άρα της ανίχνευσης πιθανού σφάλματος. Όταν ανιχνευθεί κάποιο σφάλμα, το *μερικό υπόλοιπο* και το *ψηφίο του πηλίκου* διορθώνονται, οπότε η καθυστέρηση αυξάνεται. Εάν όμως η πιθανότητα σφάλματος είναι μικρή, η διόρθωση αυτή καθίσταται ανεκτή, εφόσον η *συνάρτηση επιλογής* του *πηλίκου* γίνεται πολύ απλούστερη. Αντί λοιπόν μίας ακριβούς και αργής *συνάρτησης επιλογής* του q, χρησιμοποιείται μία αρκετά καλή προσέγγιση, που μπορεί να μειώσει τη μέση καθυστέρηση του κυκλώματος. Τέλος, όλες οι παραπάνω μέθοδοι μπορούν να συνδυαστούν, και να βελτιώσουν αθροιστικά την συνολική απόδοση του διαιρέτη.

3 ΣΧΕΔΙΑΣΜΟΣ ΥΛΟΠΟΙΗΣΕΩΝ

Για τις ανάγκες της εργασίας αναπτύχθηκαν τρεις διαφορετικές υλοποιήσεις, που αφορούν παραλλαγή του βασικού σχήματος του SRT. Αναπτύχθηκαν στην μορφή *συνδυαστικού πίνακα*, αλλά θα μπορούσαν με την ίδια δομική σειρά, να αναπτυχθούν στη μορφή *ακολουθιακού* κυκλώματος.

3.1 Εργασία [68]

Αρχικά, υλοποιήθηκε το σχέδιο που προτείνεται στην εργασία [68]. Κάθε σειρά του *συνδυαστικού πίνακα*, βασίζεται στον πίνακα αληθείας της εικόνας 3.1.1. Σύμφωνα με αυτόν, εξετάζοντας τα δύο πρώτα ψηφία του *μερικού υπολοίπου*, αποφασίζεται το *πηλίκο* και κατ' επέκταση η πράξη που θα πραγματοποιηθεί.

$\hat{2}$	q	a/s	restore	compress	$\hat{2}-q.D_2$
00	0	x	1	0	00
01	0	x	1	0	01
01	0	x	1	0	01
10	1	0	0	x	00
11	1	0	0	x	01
11	0	x	1	1	01
10	1	1	0	x	00
11	0	x	1	1	01
11	1	1	0	x	01

x = Don't Care

Εικόνα 3.1.1 Πίνακας αληθείας απόφασης ψηφίου πηλίκου και σημάτων ελέγχου της εργασίας [68].

Καθώς το μερικό υπόλοιπο βρίσκεται σε *Carry-Save* μορφή, το ψηφίο του *πηλίκου* αποφασίζεται από μία εκτίμησή του $\hat{z}$, σύμφωνα με τον κανόνα επιλογής:

$$q_j = \begin{cases} 1 \text{ εάν } 1/2 \le \hat{z} \le 1 \\ 0 \text{ εάν } -1/4 \le \hat{z} \le 1/4 \\ \bar{1} \text{ εάν } -1 \le \hat{z} \le -1/2 \end{cases}$$

Κάθε *μερικό υπόλοιπο* z αρχίζει με την εκτίμηση $\hat{z}$, που αποτελείται από δύο ψηφία $z1, z2$. Στην πραγματικότητα το z, βρίσκεται στο διάστημα $\left(\hat{z}-\frac{1}{4}, \hat{z}+\frac{1}{4}\right)$, με το κατώτερο διάστημα να προκύπτει όταν κάθε άλλο ψηφίο ισούται με $\bar{1}$, και το ανώτερο διάστημα να προκύπτει όταν κάθε άλλο ψηφίο ισούται με 1. Στον πίνακα 3.1.1, συνοψίζονται τα πεδία τιμών ενός *μερικού υπολοίπου* z, για κάθε πιθανή εκτίμηση $\hat{z}$.

Πίνακας 3.1.1

$\hat{z}_2$		$0.z_1z_2$	$\hat{z}_{10}$	$z = \left(z - \frac{1}{4}, z + \frac{1}{4}\right)$	q
0	0	0.00	0	-0,25 , 0,25	0
0	1	0.01	0,25	0 , 0,5	0
0	-1	$0.0\bar{1}$	-0,25	-0,5 , 0	0
1	0	0.10	0,5	0,25 , 0,75	1
1	1	0.11	0,75	0,5 , 1	1
1	-1	$0.1\bar{1}$	0,25	0 , 0,5	0
-1	0	$0.\bar{1}0$	-0,5	-0,75 , -0,25	-1
-1	1	$0.\bar{1}1$	-0,25	-0,5 , 0	0
-1	-1	$0.\bar{1}\bar{1}$	-0,75	-1 , -0,5	-1

Από την εκτίμηση του *μερικού υπόλοιπου* αποφασίζονται, εκτός από το *ψηφίο πηλίκου* q, τα σήματα a/s, $restore$ και $compress$. Το πρώτο καθορίζει εάν θα γίνει πρόσθεση ή αφαίρεση του *διαιρέτη* από το *μερικό υπόλοιπο*.

- $a/s = 0$ αντιστοιχεί σε αφαίρεση,
- $a/s = 1$ αντιστοιχεί σε πρόσθεση,

Το σήμα $restore$ χρησιμοποιείται σαν επιλογή, σε μία σειρά από πολυπλέκτες στο τέλος κάθε σειράς.

- Εάν $restore = 1$, το αποτέλεσμα της πράξης αγνοείται και τα ψηφία του *μερικού υπολοίπου* προωθούνται στην επόμενη σειρά. Το *μερικό υπόλοιπο*, ολισθαίνει έτσι κατά μία θέση αριστερά.
- Εάν $restore = 0$, επιλέγεται το αποτέλεσμα της πράξης, το οποίο εν συνεχεία ολισθαίνει κατά μία θέση αριστερά.

Το σήμα $compress$ είναι απαραίτητο για την αποφυγή *υπερχείλισης αναπαράστασης*. Όταν $restore = 1$ και το *μερικό υπόλοιπο* προωθείται στην επόμενη σειρά, υπάρχει κίνδυνος να χαθεί το *πιο σημαντικό ψηφίο*. Αυτό μπορεί να συμβεί σε δύο περιπτώσεις, αλλά αντιμετωπίζεται κατά τον ακόλουθο τρόπο:

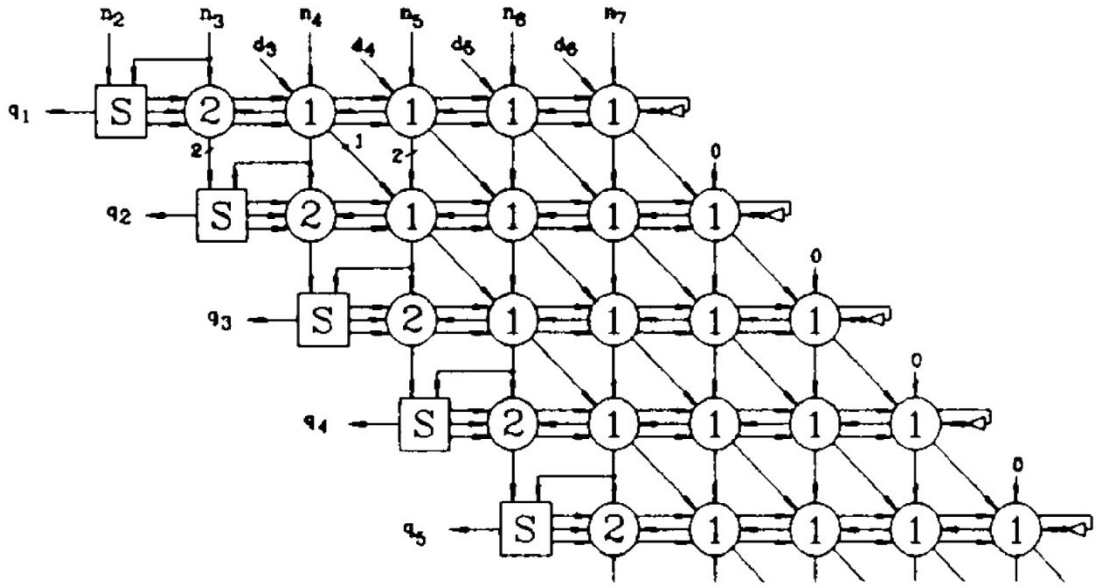
- Όταν $\hat{z} = 1\bar{1}$, το σήμα $compress$ γίνεται 1 και μετατρέπει σε $\hat{z} = 01$, που είναι ίσο.
- Όταν $\hat{z} = \bar{1}1$, το σήμα $compress$ γίνεται 1 και μετατρέπει σε $\hat{z} = 0\bar{1}$, που είναι ίσο.
- Όταν $restore = 1$ και $z1 = 0$, το $compress$ είναι 0, αφού δεν υπάρχει κίνδυνος *υπερχείλισης αναπαράστασης*.
- Όταν $restore = 0$, το σήμα $compress$ είναι αδιάφορο καθώς δεν επηρεάζει το αποτέλεσμα.

Η τελευταία στήλη του πίνακα προσδιορίζει το επιθυμητό αποτέλεσμα, για τα δύο ψηφία του *μερικού υπολοίπου* στο τέλος της σειράς. Το πρώτο ψηφίο θα απορριφθεί με την ολίσθηση. Πρέπει να εξασφαλίζεται λοιπόν ότι είναι πάντα 0, όπως αναφέρθηκε, ώστε να μην υπάρξει *υπερχείλιση αναπαράστασης*. Για να τηρούνται αυτές οι προσδοκίες, πρέπει τα δύο πρώτα ψηφία του *διαιρέτη* D_2, να είναι 10. Πρέπει δηλαδή ο *διαιρέτης* να είναι της

μορφής $0.10d_3d_4 \dots d_n$, και να ανήκει στο διάστημα $[0.5, 0.75)$. Ο *διαιρέτης* εισάγεται στο κύκλωμα σε *κανονικοποιήμενη* μορφή $0.1d_2d_3 \dots d_n$, έτσι:

- Εάν $d_2 = 0$, η διαίρεση μπορεί να εκτελεστεί άμεσα με τα ορίσματα ως έχουν.
- Εάν όμως $d_2 = 1$, τα ορίσματα πρέπει να πολλαπλασιασθούν με το $\frac{3}{4}$. Αυτό γίνεται με απλές ολισθήσεις και μία πρόσθεση, $\alpha \cdot \frac{3}{4} = \frac{\alpha}{2} + \frac{\alpha}{4}$.

Τα ανωτέρω υλοποιούνται απο μία δομική σειρά, η οποία επαναλαμβάνεται *n* φορές, όπως φαίνεται στην εικόνα 3.1.2.



Εικόνα 3.1.2 Συνδυαστικός πίνακας εργασίας [68].

Όπως φαίνεται, στην αρχή της σειράς υπάρχει το *κελί S*. Εκεί γίνεται ο έλεγχος της εκτίμησης του *μερικού υπολοίπου* $\hat{z}$ και παράγονται τα απαιτούμενα σήματα ελέγχου των υπόλοιπων *κελιών*. Ακολουθεί το *κελί 2*, που αναλαμβάνει την συμπίεση μίας εκτίμησης που πρόκειται να υπερχειλίσει. Όταν τα σήματα $restore$ και $compress$ είναι 1, το ψηφίο z_2 αντιστρέφεται μέσω των *XOR*, και οι πολυπλέκτες με $\overline{restore} = 0$ επιλέγουν την ολίσθηση της συμπιεσμένης εκτίμησης. Στην περίπτωση που $\overline{restore} = 1$, το *κελί 2* και όλα τα *κελιά 1* που έπονται, αποτελούν *αθροιστή αποθήκευσης κρατουμένου*.

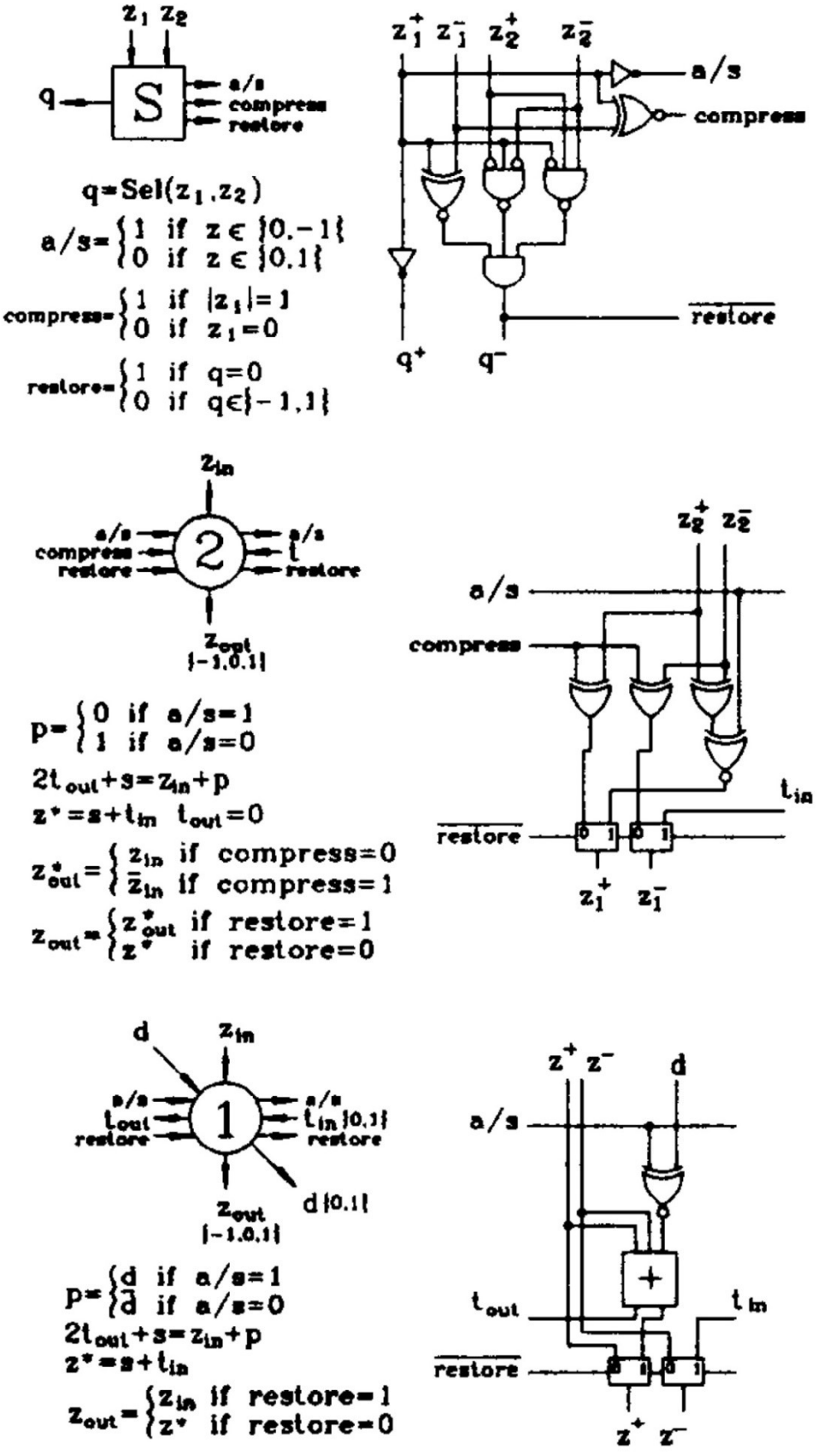
Τρίτο όρισμα της άθροισης έιναι η *XOR* του σήματος a/s με τον *διαιρέτη*, ενώ το *κρατούμενο εισόδου* στο τελευταίο *κελί 1* ισούται με $\overline{a/s}$, διαμορφώνοντας τον αντίθετο του *διαιρέτη* σε *συμπλήρωμα ως προς δύο* όποτε χρειαστεί:

- Εάν ο *διαιρέτης* προστίθεται στο *μερικό υπόλοιπο*, ισχύει για το τρίτο όρισμα $ci = di$.
- Εάν ο *διαιρέτης* αφαιρείται από το *μερικό υπόλοιπο*, προστίθεται το *συμπλήρωμα ως προς δύο του διαιρέτη*. Ισχύει δηλαδή $ci = di'$ και $ulp = \overline{a/s} = 1$.

Συνοψίζοντας:

$$ci = (di \oplus as)' = \begin{cases} di \text{ εάν } a/s = 1 \\ di' \text{εάν } a/s = 0 \end{cases}$$

Στην εικόνα 3.1.3, φαίνονται τα σχέδια για κάθε τύπο *κελιού*.



Εικόνα 3.1.3 Σχέδια για κάθε τύπο κελιού της Εργασίας [68].

Επίσης σημαντική είναι η κωδικοποίηση των *SD ψηφίων* του *μερικού υπολοίπου* και του *πηλίκου*. Όσον αφορά το *μερικό υπόλοιπο*, στην εργασία χρησιμοποιείται η κωδικοποίηση που φαίνεται στους πίνακες της εικόνας 3.1.4, για την υλοποίηση με συμβατικούς *πλήρεις αθροιστές*.

z^+	Code
0	0
1	1

z^-	Code
−1	0
0	1

Εικόνα 3.1.4 κωδικοποίηση SD ψηφίων μερικού υπολοίπου [68].

Σύμφωνα με τους πίνακες, η τιμή κάθε ψηφίου z_i δίνεται από τα bits z_i^+ και z_i^-, όπως στον πίνακα 3.1.2. Παρατηρούμε ότι ισχύει $z_i = z_i^+ - \overline{z_i^-}$.

Πίνακας 3.1.2 Τιμή κάθε SD ψηφίου του μερικού υπολοίπου.

z_i^+	z_i^-	z_i
0	0	$\bar{1}$
0	1	0
1	0	0
1	1	1

Όσον αφορά τα ψηφία του *πηλίκου*, δεν αναφέρεται κάπου στην εργασία η κωδικοποίηση που χρησιμοποιείται. Χρησιμοποιήθηκε λοιπόν η κωδικοποίηση, για την οποία $q_i = q_i^+ - q_i^-$, καθώς έτσι είναι εύκολη η μετατροπή σε *συμπλήρωμα ως προς δύο*. Εξάγοντας όμως το σχέδιο του *κελιού S* από την εργασία, σε λογικές συναρτήσεις και πίνακα αληθείας (εικόνα 3.1.5-a), μπορούμε να δούμε ότι χρησιμοποιεί κωδικοποίηση κατά την οποία το q_i προκύπτει σύμφωνα με τον πίνακα 3.1.3.

Πίνακας 3.1.3 Τιμή κάθε SD ψηφίου του πηλίκου.

q_i^+	q_i^-	q_i
0	0	0
0	1	1
1	0	0
1	1	$\bar{1}$

(a)

z1p	z1m	z2p	z2m	=>	qp	qm	nres	as	com
0	0	0	0		1	1	1	1	1
0	0	0	1		1	1	1	1	1
0	0	1	0		1	1	1	1	1
0	0	1	1		1	0	0	1	1
0	1	0	0		1	0	0	1	0
0	1	0	1		1	0	0	1	0
0	1	1	0		1	0	0	1	0
0	1	1	1		1	0	0	1	0
1	0	0	0		0	0	0	0	0
1	0	0	1		0	0	0	0	0
1	0	1	0		0	0	0	0	0
1	0	1	1		0	0	0	0	0
1	1	0	0		0	0	0	0	1
1	1	0	1		0	1	1	0	1
1	1	1	0		0	1	1	0	1
1	1	1	1		0	1	1	0	1

(b)

z1p	z1m	z2p	z2m	=>	qp	qm	as	nres	com
0	0	0	0		0	1	1	1	X
0	0	0	1		0	1	1	1	X
0	0	1	0		0	1	1	1	X
0	0	1	1		0	0	X	0	1
0	1	0	0		0	0	X	0	0
0	1	0	1		0	0	X	0	0
0	1	1	0		0	0	X	0	0
0	1	1	1		0	0	X	0	0
1	0	0	0		0	0	X	0	0
1	0	0	1		0	0	X	0	0
1	0	1	0		0	0	X	0	0
1	0	1	1		0	0	X	0	0
1	1	0	0		0	0	X	0	1
1	1	0	1		1	0	0	1	X
1	1	1	0		1	0	0	1	X
1	1	1	1		1	0	0	1	X

Εικόνα 3.1.5 Πίνακες αληθείας, (a) όπως προκύπτει από το σχέδιο του κελιού S, (b) όπως υλοποιήθηκε.

3.2 Εργασία [29]

Στην εργασία αυτή, περιγράφεται η δομική σειρά μίας υλοποίησης διαιρέτη. Για δίκαιη σύγκριση των τριών υλοποιήσεων, και εδώ αναπτύχθηκε σε *συνδυαστικό πίνακα*. Το *ψηφίο πηλίκου* εξάγεται από τον πίνακα αληθείας, που απεικονίζεται στις δύο πρώτες στήλες του πίνακα της εικόνας 3.2.1. Σημειώνεται ότι υπάρχει ένα λάθος στην στήλη q_j. Συγκεκριμένα, η τιμή που αντιστοιχεί στο $0.\overline{11}$, είναι $\overline{1}$.

r_0	r_1	r_2	q_j	$d_2 = 0$ r_0^{**}	r_0^{*}	r_1^{**}	$d_2 = 1$ r_0^{**}	r_0^{*}	r_1^{**}
1	0	1	1				0	1	1
1	0	0	1				0	0	0
1	0	$\overline{1}$	1				0	0	1
1	$\overline{1}$	1	1				1	1	1
1	$\overline{1}$	0	1				1	0	0
1	$\overline{1}$	$\overline{1}$	0				0	0	0
0	1	1	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	0	0
0	1	$\overline{1}$	0	0	0	0	0	0	0
0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	1	1	1	1	1
0	0	0	$\overline{0}$	0	0	0	0	0	0
0	0	$\overline{1}$	$\overline{0}$	0	0	1	0	0	1
0	$\overline{1}$	1	$\overline{0}$	1	1	1	1	1	1
0	$\overline{1}$	0	$\overline{1}$	0	0	0	0	1	1
0	$\overline{1}$	$\overline{1}$	$\overline{0}$	0	0	1	0	0	0
$\overline{1}$	1	1	$\overline{0}$				1	1	1
$\overline{1}$	1	0	$\overline{1}$				0	1	1
$\overline{1}$	1	$\overline{1}$	$\overline{1}$				0	0	0
$\overline{1}$	0	1	$\overline{1}$				1	1	0
$\overline{1}$	0	0	$\overline{1}$				1	1	1
$\overline{1}$	0	$\overline{1}$	$\overline{1}$				1	0	0

Εικόνα 3.2.1 Πίνακας αληθείας και μερικό υπόλοιπο (R_{j+1}) που προκύπτει, εξετάζοντας τα 3 MSD ψηφία του R_j [29].

Στον πίνακα της εικόνας 3.2.1 παρουσιάζονται επίσης τα πρώτα 3 bits που θα προκύψουν από την πράξη, ανάλογα με το δεύτερο ψηφίο του *διαιρέτη*.

- για $d_2 = 0$, η περιοχή λειτουργίας του αλγορίθμου είναι $-2|D| < R^{j+1} < 2|D|$, όπου $D_{min} = 0.5$, άρα $-1 < R^{j+1} < 1$. Όπως παρατηρούμε, το επόμενο ψηφίο r_0 θα είναι πάντα 0. Συνεπώς, οι υπόλοιπες τιμές του *μερικού υπολοίπου*, είναι αδιάφορες στην περίπτωση αυτή.
- για $d_2 = 1$, η περιοχή λειτουργίας του αλγορίθμου είναι $-2|D| < R^{j+1} < 2|D|$, όπου $D_{min} = 0.75$, άρα $-1.5 < R^{j+1} < 1.5$.

Το *μερικό υπόλοιπο* όπως και το *πηλίκο*, αναπαρίστανται σε σύστημα *RSD*. Κάθε ψηφίο r_i συνίσταται από δύο bits $r_i^*, r_i^{**} \in \{0,1\}$, και ισχύει $r_i = r_i^* - r_i^{**}$. Κάθε πιθανή τιμή του r_i φαίνεται στον πίνακα 3.2.1.

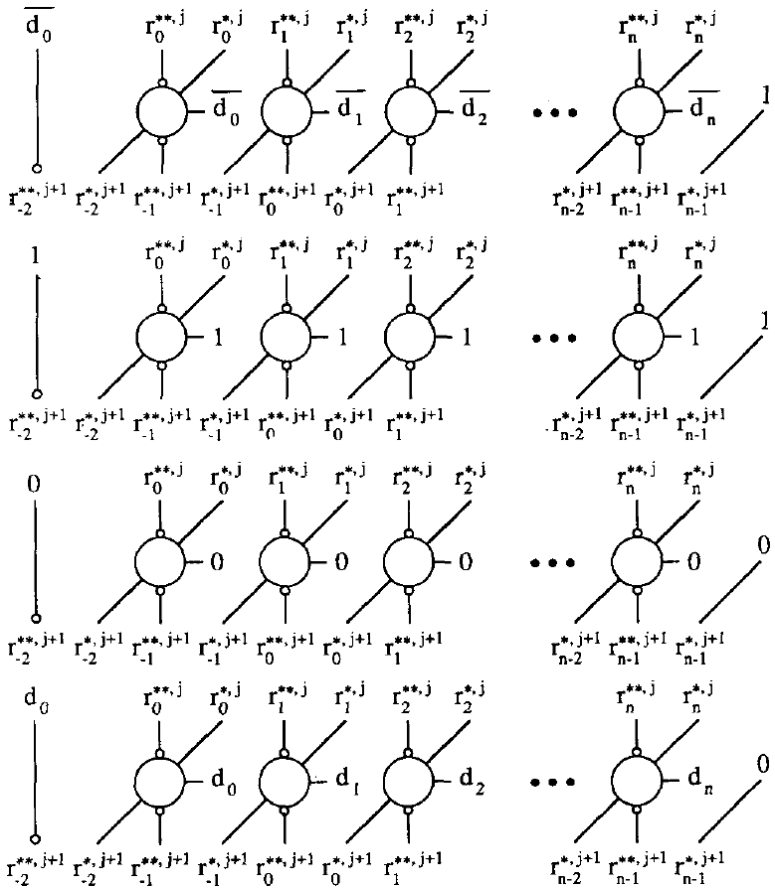
Πίνακας 3.2.1 Τιμή κάθε SD ψηφίου του μερικού υπολοίπου. Τα ίδια ισχύουν για το πηλίκο.

r_i^*	r_i^{**}	r_i
0	0	0
0	1	-1
1	0	1
1	1	0

Μετά την απόφαση του q, υπολογίζεται παράλληλα καθ' όλο το μήκος του αθροιστή, το τρίτο όρισμά του c_i, που καθορίζει την πράξη που θα εκτελεστεί. Τονίζεται εδώ, ότι δεν χρησιμοποιούνται πολυπλέκτες για την επιλογή μεταξύ του αποτελέσματος της πράξης και του *μερικού υπολοίπου*. Αντ' αυτού, όταν δεν πραγματοποιείται πράξη προστίθεται το 0. Σύμφωνα με τα ανωτέρω και την εικόνα 3.2.2:

- Εάν $q_j = 1$, αφαιρείται ο *διαιρέτης*, οπότε $c_i = \overline{d_i}$ και $ulp = 1$
- Εάν $q_j = \overline{1}$, προστίθεται ο *διαιρέτης*, οπότε $c_i = d_i$ και $ulp = 0$
- Εάν $q_j = 0$, προστίθεται το *συμπλήρωμα ως προς δύο του* 0, οπότε $c_i = 1$ και $ulp = 1$
- Εάν $q_j = \overline{0}$, προστίθεται το 0, οπότε $c_i = 0$ και $ulp = 0$

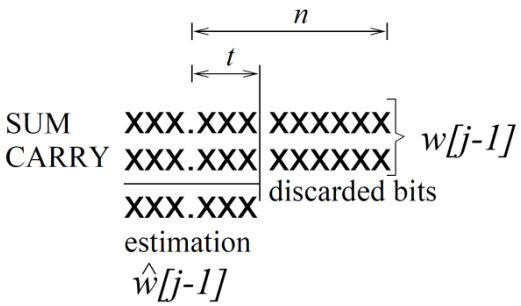
Παρατηρούμε ότι χρησιμοποιούνται δύο διαφορετικές αναπαραστάσεις για το 0. Κατ' αυτόν τον τρόπο, αποφεύγεται η *υπερχείλιση αναπαράστασης*.



Εικόνα 3.2.2 Βήματα διαίρεσης για $q_j = 1, 0, \overline{0}, \overline{1}$ [29].

3.3 Παρούσα Εργασία

Εκτός από τα προαναφερθέντα σχέδια, αναπτύχθηκε στα πλαίσια της εργασίας μία διαφοροποίηση του *SRT*. Κατόπιν της ανάπτυξης της υλοποίησης, παρατηρήθηκε ότι η βασική ιδέα θυμίζει αυτήν που περιγράφεται στην εικόνα 2 της εργασίας [32], που παρατίθεται στην εικόνα 3.3.1. Παρόμοια υλοποίηση όμως δεν βρέθηκε σε σχετική αναζήτηση της βιβλιογραφίας.



Εικόνα 3.3.1 Εικόνα 2 της εργασίας [32].

Η συγκεκριμένη υλοποίηση, διαφοροποιείται καθώς χρησιμοποιεί μία υβριδική αναπαράσταση για το *μερικό υπόλοιπο*. Η *RSD* αναπαράσταση του *μερικού υπολοίπου* χρησιμοποιεί την κωδικοποίηση του πίνακα 3.2.1, και επεκτείνεται με δύο ψηφία σε *συμπλήρωμα ως προς δύο* στα αριστερά της υποδιαστολής. Το ίδιο σύστημα *RSD* χρησιμοποιείται για την αναπαράσταση του *πηλίκου*.

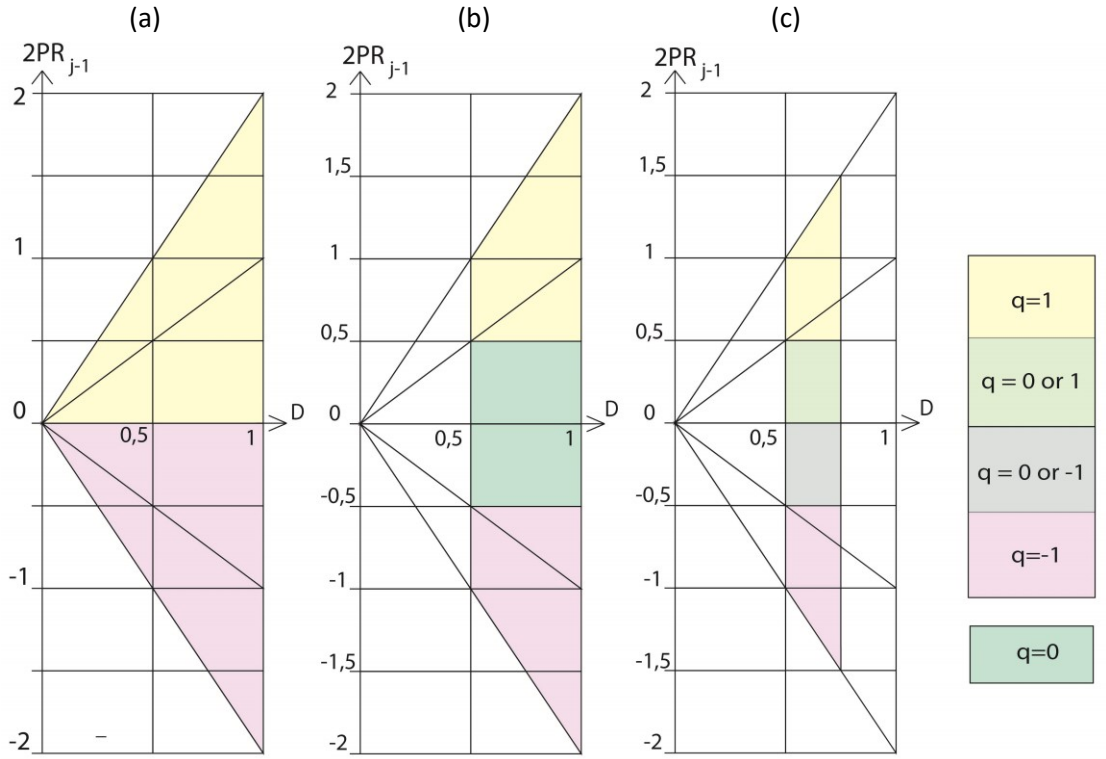
Η υβριδική αυτή αναπαράσταση, προσφέρει τρία οφέλη:

1. Χρησιμοποιούνται *αθροιστές αποθήκευσης κρατουμένου* κατά το μεγαλύτερο μήκος του *μερικού υπολοίπου*, με αποτέλεσμα την αποφυγή της *διάδοσης κρατουμένου*.
2. Η *συνάρτηση απόφασης* του *ψηφίου πηλίκου* απλοποιείται σημαντικά, καθώς εξαρτάται από λιγότερα bits.
3. Η αναπαράσταση των πρώτων δύο ψηφίων σε *συμπλήρωμα ως προς δύο* αποκλείει το πρόβλημα *υπερχείλισης αναπαράστασης*. Έτσι, όταν το *ψηφίο πηλίκου* είναι 0, το 0 προστίθεται στο *μερικό υπόλοιπο*, και σε οποιαδήποτε από τις δύο μορφές του. Αυτό συνεπάγεται ότι δεν απαιτούνται πολυπλέκτες, ενώ η παραγωγή του τρίτου ορίσματος που καθορίζει την εκτελούμενη πράξη Y_i, απλοποιείται επίσης σημαντικά.

Το μόνο μειονέκτημα της συγκεκριμένης αναπαράστασης, είναι η *διάδοση κρατουμένου* κατά μήκος τριών bits. Για την ελαχιστοποίηση της *διάδοσης*, τα ορίσματα *κανονικοποιούνται*, έτσι ώστε ο *διαιρέτης* να είναι στο διάστημα $[0.5, 0.75)$, όπως στην εργασία [68].

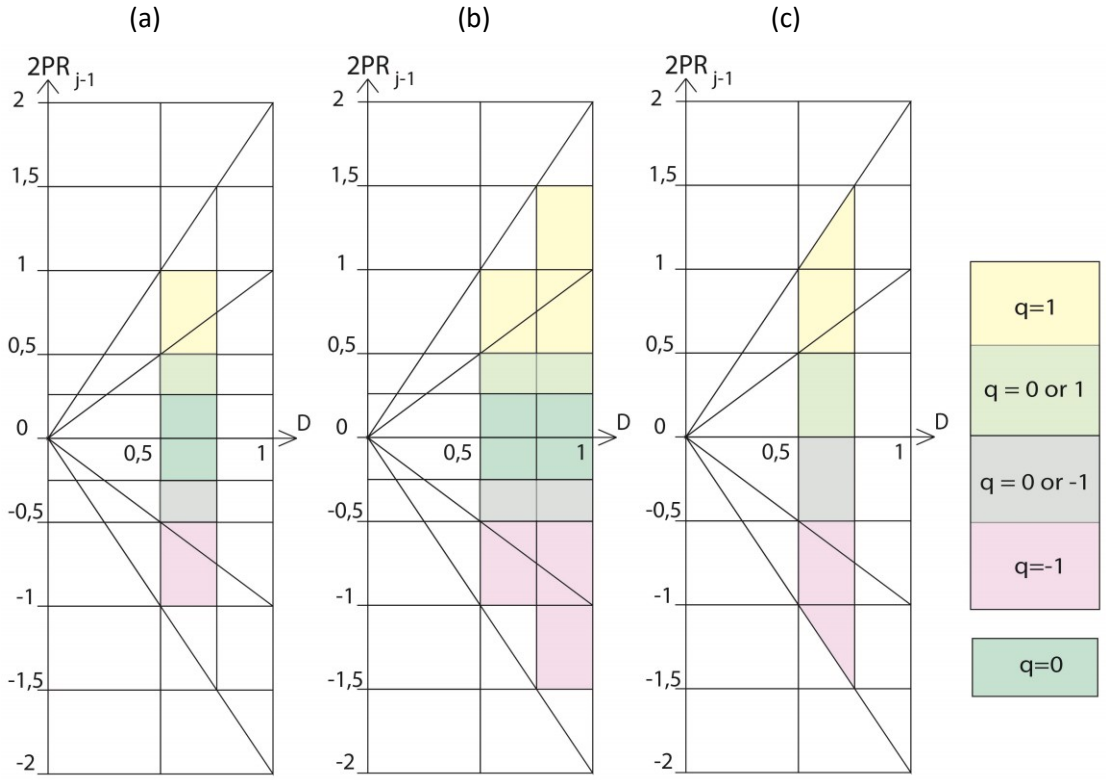
Εμβαθύνοντας στις λεπτομέρειες του σχεδίου, πρόκειται για την εκτέλεση μίας διαφοροποίησης του *SRT*, που προκύπτει από την σύνθεση του *SRT* και του *Non-Restoring*, όπως φαίνεται και στα P-D διαγράμματα της εικόνας 3.3.2. Μεγεθύνοντας τα διαστήματα επιλογής κατ' αυτόν τον τρόπο, απαιτούνται λιγότερα ψηφία για την απόφαση του q, με αποτέλεσμα να απλοποιείται περαιτέρω η *συνάρτηση επιλογής*. Ο *κανόνας επιλογής* του *ψηφίου πηλίκου* γίνεται:

$$q_j = \begin{cases} 1 \;\; \varepsilon\acute{\alpha}\nu \; 2r_{j-1} > 0 \\ 0 \;\; \varepsilon\acute{\alpha}\nu -\frac{1}{2} < 2r_{j-1} < \frac{1}{2} \\ \bar{1} \;\; \varepsilon\acute{\alpha}\nu \; 2r_{j-1} < 0 \end{cases} \qquad (3.3.1)$$



Εικόνα 3.3.2 Περιοχές επιλογής q για τους αλγορίθμους: (a) non-Restoring, (b) SRT, (c) τροποποιημένος SRT παρούσας εργασίας.

Τα P-D διαγράμματα των τριών παραλλαγών του SRT που περιγράφηκαν, παρουσιάζονται συγκεντρωτικά στην εικόνα 3.3.3.



Εικόνα 3.3.3 Περιοχές επιλογής q για τους τροποποιημένους αλγορίθμους SRT των: (a) Εργασία [68] , (b) Εργασία [29], (c) Παρούσα εργασία.

Τα τρία πρώτα ψηφία του *μερικού υπολοίπου* είναι αρκετά για την σωστή απόφαση, όπως φαίνεται και από τον πίνακα 3.3.1. Τα πρώτα δύο ψηφία αναπαρίστανται σε *συμπλήρωμα ως προς δύο*, ενώ το τρίτο στο σύστημα *RSD*. Έτσι αρκούν τέσσερα bits για την *συνάρτηση επιλογής*, και απαιτείται *διάδοση κρατουμένου* κατά μήκος τριών ψηφίων. Η εκτίμηση του αριθμού δίνεται από τη σχέση:

$$R^{est} = -2R_1 + R_0 + \frac{R_{-1}}{2}$$

Επειδή τα ακόλουθα ψηφία, μπορούν να έχουν τις τιμές $\bar{1}, 0, 1$, το *μερικό υπόλοιπο* βρίσκεται εντός του διαστήματος $\left(R^{est} - \frac{1}{2}, R^{est} + \frac{1}{2}\right)$.

Πίνακας 3.3.1 Πίνακας αληθείας σύμφωνα με την εκτίμηση 3 ψηφίων του μερικού υπολοίπου.

R_1	R_0	R_{-1}^{+}	R_{-1}^{-}	R^{est}	R	q
0	0	0	0	0	(-0,5 , 0,5)	0
0	0	0	1	-0,5	(-1 , 0)	-1
0	0	1	0	0,5	(0 , 1)	1
0	0	1	1	0	(-0,5 , 0,5)	0
0	1	0	0	1	(0,5 , 1,5)	1
0	1	0	1	0,5	(0 , 1)	1
0	1	1	0	1,5	(1 , 2)	x
0	1	1	1	1	(0,5 , 1,5)	1
1	0	0	0	-2	(-2,5 , -1,5)	x
1	0	0	1	-2,5	(-3 , -2)	x
1	0	1	0	-1,5	(-2 , -1)	-1
1	0	1	1	-2	(-2,5 , -1,5)	x
1	1	0	0	-1	(-1,5 , -0,5)	-1
1	1	0	1	-1,5	(-2 , -1)	-1
1	1	1	0	-0,5	(-1 , 0)	-1
1	1	1	1	-1	(-1,5 , -0,5)	-1

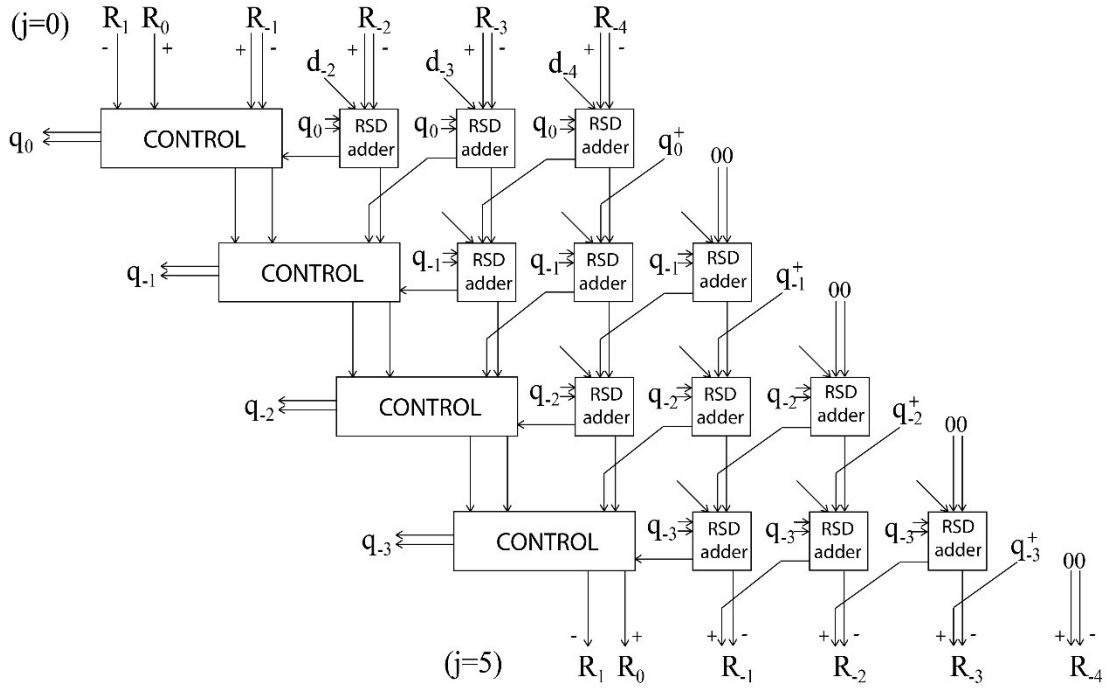
Κατά τα γνωστά, το τρίτο όρισμα των *RSD* αθροιστών προκύπτει από τον πίνακα 3.3.2. Οι τιμές για $q^+ = 1, q^- = 1$, είναι αδιάφορες, καθώς η συγκεκριμένη αναπαράσταση του 0 δεν αποδίδεται στο q.

Πίνακας 3.3.2 Πίνακας αληθείας Y_i και ulp.

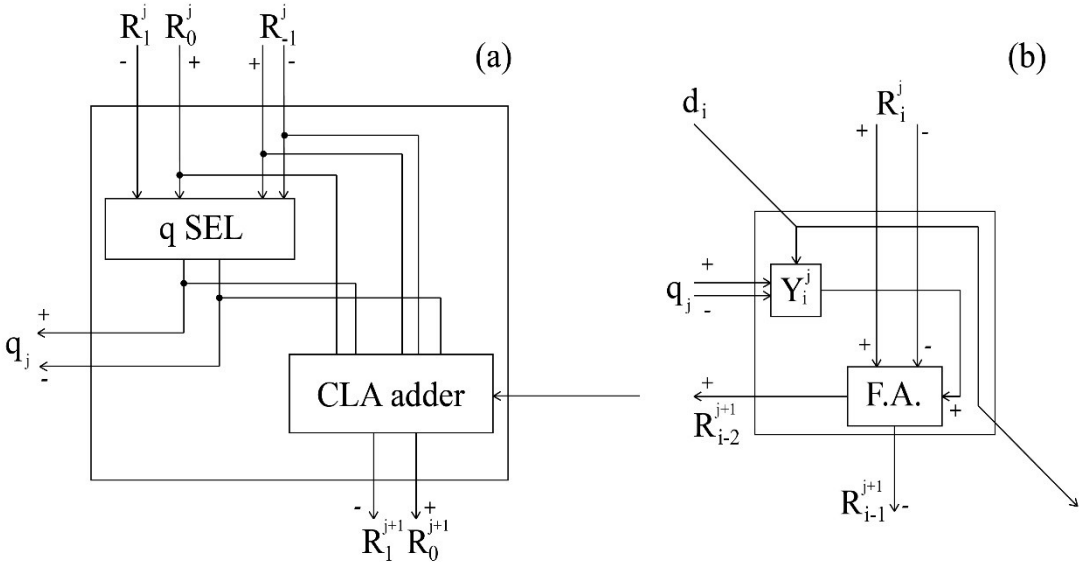
q^+	q^-	q	Y_i	ulp
0	0	0	0	0
0	1	-1	d_i	0
1	0	1	$\overline{d_i}$	1
1	1	0	x	x

Η υλοποίηση αναπτύχθηκε σε *συνδυαστικό πίνακα*, όπως και τα άλλα δύο σχέδια, και παρουσιάζεται στην εικόνα 3.3.4, όπου η αρίθμηση υποδεικνύει το *βάρος* κάθε θέσης. Στην εικόνα 3.3.5, αναπτύσσονται το *κελί ελέγχου (Control)* και ένα τυπικό κελί του αθροιστή. Σημειώνεται, ότι θέτοντας αντιστροφείς στις εισόδους και εξόδους αρνητικής αξίας,

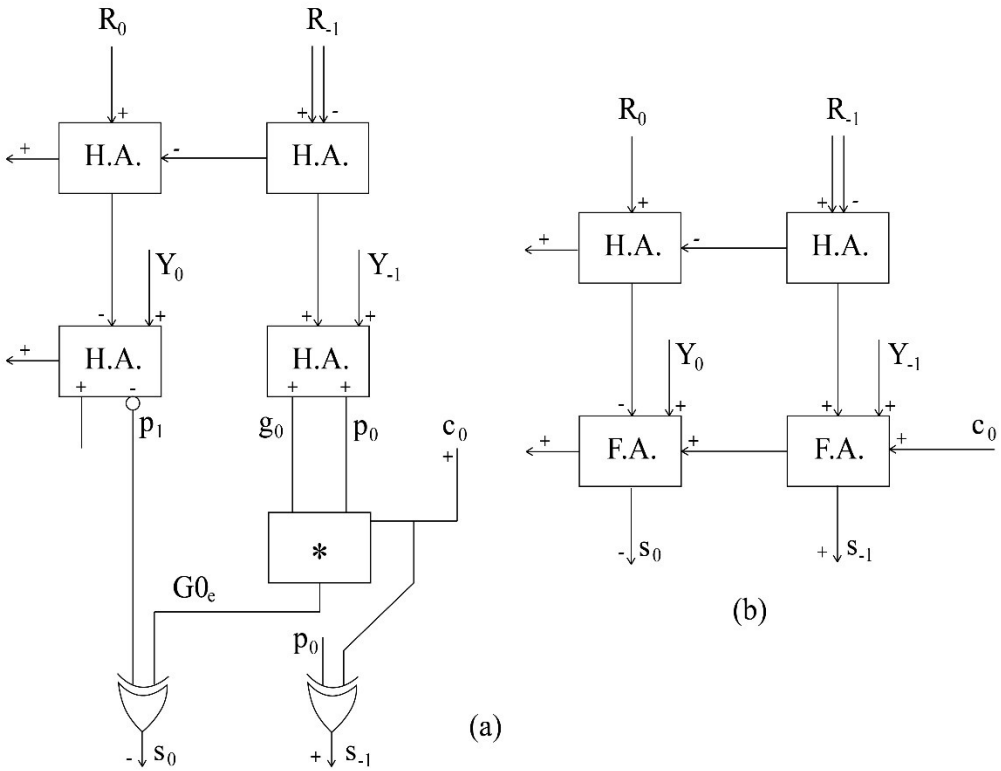
χρησιμοποιούνται συμβατικοί *πλήρεις αθροιστές*. Τέλος, στην εικόνα 3.3.6-a απεικονίζεται ο *αθροιστής πρόβλεψης κρατουμένου 3 σε 1* δύο ψηφίων, που αναλύεται στη συνέχεια και θεωρητικά. Στην εικόνα 3.3.6-b, φαίνεται ο αντίστοιχος *αθροιστής διάδοσης κρατουμένου*. Σημειώνουμε ότι τα Y_0, Y_{-1}, τοποθετήθηκαν στην δεύτερη σειρά επειδή εξαρτώνται από το *πηλίκο*.



Εικόνα 3.3.4 Συνδυαστικός πίνακας της παρούσης υλοποίησης.



Εικόνα 3.3.5 (a) Σχέδιο κελιού Control, (b) Σχέδιο κελιού RSD.



Εικόνα 3.3.6 (a) CLA 3 σε 1, δύο ψηφίων, (b) αντίστοιχος CPA.

4 ΘΕΩΡΗΤΙΚΗ ΑΝΑΛΥΣΗ ΣΧΕΔΙΩΝ

Προτού συγκρίνουμε πειραματικά τα τρία σχέδια που αναπτύχθηκαν, θα τα συγκρίνουμε θεωρητικά. Για τον σκοπό αυτό, υπολογίσαμε θεωρητικά το κρίσιμο μονοπάτι και την επιφάνεια των σχεδίων, μέσω αναλυτικής εξέτασης των δομικών μονάδων τους. Οι υπολογισμοί πραγματοποιήθηκαν σύμφωνα με τις προσεγγίσεις του *μοντέλου μοναδιαίας-πύλης (Unit-Gate Model),* που παρουσιάζεται στον πίνακα 4.1. Οι μονάδες του πίνακα είναι, *Ag - επιφάνεια μοναδιαίας πύλης*, και *Tg - καθυστέρηση μοναδιαίας πύλης*. Σημειώνεται ότι τα αποτελέσματα αφορούν μόνο το κύκλωμα της διαίρεσης, σε αντίθεση με τις υλοποιήσεις που περιλαμβάνουν επιπλέον αλλά ίδιο υλικό ανά υλοποίηση.

Πίνακας 3.3.1 Μοντέλο μοναδιαίας-πύλης.

Στοιχεία	Area (Ag)	Delay (Tg)
NAND-2, NOR-2	1	1
NAND-3, NOR-3	2	2
XOR, XNOR	2	2
H.A.	3	1 (carry)
		2 (sum)
F.A.	7	3 (carry)
		4 (sum)
MUX 2-1	3	2
#	3	2
*	2	2

4.1 Εργασία [68]

Το κύκλωμα του διαιρέτη, αρχίζει με την χρήση ενός *CSA* και ενός *CLA*, για τον πιθανό πολλαπλασιασμό των ορισμάτων με το $3/4$. Ο *διαιρετέος* αναπαρίσταται με *προσημασμένα ψηφία*, και έτσι ο *CSA* αρκεί. Ο *διαιρέτης* όμως αναπαρίσταται σε *συμπλήρωμα ως προς δύο*, οπότε είναι απαραίτητη η *διάδοση κρατουμένου*, ή καλύτερα η *πρόβλεψή* του. Για τους υπολογισμούς χρησιμοποιήθηκε ο *Brent-Kung CLA* [8], και οι εκτιμήσεις της επιφάνειας και της καθυστέρησής του, φαίνονται στον πίνακα 4.1.1. Οι αντίστοιχες εκτιμήσεις για τον CSA φαίνονται στον πίνακα 4.1.2.

Πίνακας 4.1.1 Εκτίμηση καθυστέρησης, επιφάνειας CLA.

	CLAdder (N)	
	Delay (Tg)	Area (Ag)
H.A.	1	N
#	log2(N)	N log2(N)/2
XOR	1	N
	4 + 2 log2(N)	5N + 3N log2(N)/2

Πίνακας 4.1.2 Εκτίμηση καθυστέρησης, επιφάνειας CSA.

	CSAdder (N)	
	Delay (Tg)	Area (Ag)
F.A.	1	N-1
	4	7 (N-1)

Ακολουθεί μία σειρά από πολυπλέκτες 2 σε 1, που επιλέγουν μεταξύ X και $3X/4$ για τον *διαιρετέο*, και μεταξύ D και $3D/4$ για τον *διαιρέτη*.

Πίνακας 4.1.3 Εκτίμηση καθυστέρησης, επιφάνειας των συνολικών κυκλωμάτων κανονικοποίησης του διαιρετέου και του διαιρέτη.

Scale X (n)		Scale D (n)	
Delay (Tg)	Area (Ag)	Delay (Tg)	Area (Ag)
CSAdder (n-2) mux 2-1	CSAdder (n-2) n mux 2-1	CLAdder (n-2) mux 2-1	CLAdder (n-2) n mux 2-1
6	7(n-3)+3n = 10n-21	6 + 2 log2(n-2)	5(n-2) + 3(n-2) log2(n-2)/2 + 3n

Αφού γίνει βέβαιο ότι ο *διαιρέτης* βρίσκεται στο διάστημα $[0.5, 0.75)$, εκτελείται η διαίρεση. Κάθε δομική σειρά του πίνακα, περιλαμβάνει ένα *κελί S*, ένα *κελί 2* και *n-2 κελιά 1*, όπου *n* το μήκος των ορισμάτων. Ακολουθεί η ανάλυση για κάθε κελί:

Πίνακας 4.1.4 Εκτίμηση καθυστέρησης, επιφάνειας κελιού S.

	a/s	compress	q+	q-	restore'	cell S
	-	1 XNOR	2 NOR-3 1 NAND-2	2 NOR-3 1 NAND-2	- - 1 NAND-2	
Area (Ag)	-	2	5	5	1	13
Delay (Tg)	-	2	3	3	4	4

Πίνακας 4.1.5 Εκτίμηση καθυστέρησης, επιφάνειας κελιού 2.

	z1+		z1-		cell 2
	compress 1 XOR	a/s 1 XOR 1 XNOR	compress 1 XOR	tin	
	mux 2-1		mux 2-1		
Area (Ag)	2	4	2	-	14
	+3		+3		
Delay (Tg)	6	6	7	7	7

Πίνακας 4.1.6 Εκτίμηση καθυστέρησης, επιφάνειας κελιού 1.

	z+		z-		tout	cell 1
	restore'	a/s 1 XNOR F.A.(sum)	restore'	tin	a/s 1 XNOR F.A.(carry)	
	mux 2-1		mux 2-1			
Area (Ag)	-	2+7	-	-	(2+7)	15
	+3		+3			
Delay (Tg)	6	8	7	7	5	8

Παρατηρούμε ότι το κρίσιμο μονοπάτι είναι ευρύ, κατά μήκος των *κελιών 1*. Συνεπώς το κρίσιμο μονοπάτι του σχεδίου, διαμορφώνεται παραμετρικά σύμφωνα με τον πίνακα4.1.7.

Πίνακας 4.1.7 Εκτίμηση καθυστέρησης της εργασίας [68].

	Εργασία [68]						
	n	8	16	24	32	53	64
Scale D (n)	6 + 2 log2(n-2)	12	14	16	16	18	18
n cell 1	8 n	64	128	192	256	424	512
Delay (Tg)	6 + 2 log2(n-2) + 8n	76	142	208	272	442	530

Αθροίζοντας την επιφάνεια όλων των επιμέρους δομικών μονάδων, βρίσκουμε παραμετρικά την συνολική επιφάνεια:

Πίνακας 4.1.8 Εκτίμηση επιφάνειας της εργασίας [68].

	Εργασία [68]						
	n	8	16	24	32	53	64
Scale D (n)	5(n-2) + 3(n-2) log2(n-2)/2 + 3n	72	181	314	426	797	967
Scale X (n)	10n-21	59	139	219	299	509	619
n cell S	13n	104	208	312	416	689	832
n cell 2	14n	112	224	336	448	742	896
n(n-2) cell 1	15n(n-2)	720	3360	7920	14400	40545	59520
Area (Ag)	15n^2+15n-31+3(n-2) log2(n-2)/2	1067	4112	9101	15989	43282	62834

4.2 Εργασία [29]

Σε αντίθεση με την εργασία [68], εδώ η διαίρεση αρχίζει άμεσα. Κάθε δομική σειρά του πίνακα, αποτελείται από ένα *κελί ελέγχου* που αποφασίζει το *πηλίκο*, και ακολουθούν *n πλήρεις αθροιστές* που εξαρτώνται από αυτό. Το κρίσιμο μονοπάτι λοιπόν, εξαρτάται από τις *συναρτήσεις απόφασης* των συνιστωσών του πηλίκου, και κατόπιν διασπείρεται σε όλο το μήκος της δομικής σειράς. Σύμφωνα με τον πίνακα αλήθειας, και κατόπιν βελτιστοποίησης με τη χρήση λογισμικού [69], εξήχθησαν οι *συναρτήσεις απόφασης* του *πηλίκου* σε άθροισμα γινομένων. Η βελτιστοποίηση έγινε με κριτηρίο την ελάχιστη καθυστέρηση και επιφάνεια.

q+ = z1p z1m' z2p z2m' + z0p z0m z1p' z1m' z2p' z2m + z0p z0m z1p z1m z2p' z2m + z0m' z1p' z1m' z2p' z2m + z0m' z1p z1m z2p' z2m + z1p' z1m z2p z2m' + z0p z1p z1m' z2m' + z0p z1p z1m' z2p + z0m' z1p z1m' z2m' + z0m' z1p z1m' z2p + z0p z0m' z2m' + z0p z0m' z2p ;

q- = z0p' z1p' z2p' z2m + z0p' z1m z2p' z2m + z0p z0m z1p' z1m' z2p' z2m + z0p z0m z1p z1m z2p' z2m + z0m z1p' z1m + z0p' z1p' z1m + z0p' z0m ;

Το *κελί ελέγχου* μπορεί να υλοποιηθεί ως άθροισμα γινομένων, σύμφωνα με τις ανωτέρω συναρτήσεις, και τα αποτελέσματα της θεωρητικής ανάλυσης παρουσιάζονται στον πίνακα 4.2.1.

Πίνακας 4.2.1 Εκτίμηση καθυστέρησης, επιφάνειας κελιού ελέγχου.

	q+	q-	Control
Area (Ag)	51	27-10(κοινοί όροι)	68
Delay (Tg)	6	5	6

Μετά την απόφαση του *ψηφίου πηλίκου*, και για κάθε ψηφίο του *διαιρέτη*, υπολογίζεται το τρίτο όρισμα κάθε *πλήρη αθροιστή*:

Πίνακας 4.2.2 Εκτίμηση καθυστέρησης, επιφάνειας υπολογισμού κρατουμένου.

	c(q+,q-)
Area (Ag)	3
Delay (Tg)	6+2

Συνοψίζοντας για το κελί *RSD*:

Πίνακας 4.2.3 Εκτίμηση καθυστέρησης, επιφάνειας κελιού RSD.

	r*	r**	cell rsd
	Control c(q+,q-)		
	F.A.(carry)	F.A.(sum)	
Area (Ag)	10		10
Delay (Tg)	11	12	6+6

Αθροίζοντας την ανωτέρω δομική σειρά *n* φορές, προκύπτουν παραμετρικά, το κρίσιμο μονοπάτι του πίνακα και η συνολική του επιφάνεια:

Πίνακας 4.2.4 Εκτίμηση καθυστέρησης της εργασίας [29].

	Εργασία [29]						
	n	8	16	24	32	53	64
n Control	6n	48	96	144	192	318	384
n cell rsd	6n	48	96	144	192	318	384
Delay (Tg)	12n	96	192	288	384	636	768

Πίνακας 4.2.5 Εκτίμηση επιφάνειας της εργασίας [29].

	Εργασία [29]						
	n	8	16	24	32	53	64
n Control n^2 cell rsd	68n 10n^2	544 640	1088 2560	1632 5760	2176 10240	3604 28090	4352 40960
Area (Ag)	10n^2+68n	1184	3648	7392	12416	31694	45312

4.3 Παρούσα Εργασία

Στο σχέδιο της παρούσης εργασίας, γίνεται κανονικοποίηση των ορισμάτων όπως στην εργασία [68]. Εξ αιτίας της διαφορετικής κωδικοποίησης των ψηφίων του *μερικού υπολοίπου*, ο CSA προκύπτει απλούστερος επειδή χρειάζεται ημιαθροιστές αντί για πλήρεις αθροιστές. Οι εκτιμήσεις καθυστέρησης και επιφάνειας, φαίνονται στον πίνακα 4.1.1 για τον CLA και τον πίνακα 4.3.1 για τον CSA.

Πίνακας 4.3.1 Εκτίμηση καθυστέρησης, επιφάνειας CSA2.

	CSAdder2 (N)	
	Delay (Tg)	Area (Ag)
H.A.	1	N-1
	2	3 (N-1)

Ακολουθεί η σειρά από πολυπλέκτες 2 σε 1, που επιλέγουν μεταξύ X και $3X/4$ για τον *διαιρετέο*, και μεταξύ D και $3D/4$ για τον *διαιρέτη*. Τα συνολικά κυκλώματα κανονικοποίησης απαιτούν καθυστέρηση και επιφάνεια σύμφωνα με τον πίνακα 4.3.2.

Πίνακας 4.3.2 Εκτίμηση καθυστέρησης, επιφάνειας των συνολικών κυκλωμάτων κανονικοποίησης του διαιρετέου και του διαιρέτη.

Scale X (n)		Scale D (n)	
Delay (Tg)	Area (Ag)	Delay (Tg)	Area (Ag)
CSAdder2 (n-2) mux 2-1	CSAdder2 (n-2) n mux 2-1	CLAdder (n-2) mux 2-1	CLAdder (n-2) n mux 2-1
4	3(n-3)+3n = 6n-9	6 + 2 log2(n-2)	5(n-2) + 3(n-2) log2(n-2)/2 + 3n

Το ψηφίο του *πηλίκου* αποφασίζεται πολύ απλά, σε αντίθεση με την εργασία [29]:

Πίνακας 4.3.3 Εκτίμηση καθυστέρησης, επιφάνειας κελιού ελέγχου.

	q+	q-	Control
Area (Ag)	4	3	7
Delay (Tg)	3	3	3

Έτσι και το τρίτο όρισμα Y_i που καθορίζει την πράξη που θα εκτελεστεί, μπορεί να υπολογιστεί από το *πηλίκο*:

Yi = q+ di + q- di';

ή από ένα μικρό άθροισμα γινομένων σε κάθε κελί *rsd*:

Yi = di R1 + di' R1' R0 + di R0' R-1p' R-1m + di' R1' R-1p R-1m';

Πίνακας 4.3.4 Εκτίμηση καθυστέρησης, επιφάνειας υπολογισμού κρατουμένου.

	Y_0 = q+	$Y_{-1} = \mathrm{q}^-$	$Y_i(\mathrm{q}^+, \mathrm{q}^-)$	Y_i (είσοδοι)
Area (Ag)	-	-	3	12
Delay (Tg)	3	3	5	4

Όπως φαίνεται όμως, η επιφάνεια υλοποίησης του αθροίσματος γινομένων είναι αρκετά μεγάλη για το *κελί rsd*, και δεν παρέχει οφέλη στα πλαίσια της θεωρητικής ανάλυσης, αφού δεν συμμετέχει κάθε Y_i στο κρίσιμο μονοπάτι. Θεωρούμε συνεπώς, ότι κάθε Y_i εξαρτάται από το *πηλίκο*.

Αναλύοντας το κρίσιμο μονοπάτι, στο *κελί αθροιστή 3 σε 1*, παρατηρούμε ότι εξαρτάται από το c_0. Εστιάζοντας σε αυτό, και βελτιστοποιώντας το με τη χρήση λογισμικού [69], έχουμε:

Entered:
Y-2 = R1 d-2 + R1' R0 d-2' + R0' R-1p' R-1m d-2 + R1' R-1p R-1m' d-2';
c0 = R-2m' Y-2 + R-2m' R-2p + R-2p Y-2;

Minimized:
c0 = R-2m' R-2p + R1 d-2 R-2m' + R1 d-2 R-2p + R1' d-2' R0 R-2m' + R1' d-2' R0 R-2p + d-2 R0' R-1p' R-1m R-2m' + R1' d-2' R-1p R-1m' R-2m' + d-2 R0' R-1p' R-1m R-2p + R1' d-2' R-1p R-1m' R-2p;

Υλοποιώντας την συγκεκριμένη συνάρτηση απόφασης, σχεδιάζουμε μία ταχύτερη υλοποίηση με λίγο επιπλέον κόστος. Αντί για 8 Ag και 8 Tg που απαιτούνταν για την απόφαση του c_0, απαιτούνται 35 Ag και 6 Tg. Στους ακόλουθους δύο πίνακες, παρουσιάζεται η ανάλυση ως προς καθυστέρηση και επιφάνεια, των δύο διαφοροποιήσεων του *κελιού αθροιστή 3 σε 1*:

Πίνακας 4.3.5 Εκτίμηση καθυστέρησης των δύο διαφοροποιήσεων του κελιού αθροιστή 3 σε 1.

	cell adder 3 σε 1			
	χαμηλό κόστος		χαμηλή καθυστέρηση	
	s_0	s_{-1}	s_0	s_{-1}
	Y_{-2}		c_0 (είσοδοι)	
	F.A.(carry)		* 1 XOR	1 XOR
	* 1 XOR	1 XOR		
Delay (Tg)	12	10	10	8
max Delay (Tg)	12		10	

Πίνακας 4.3.6 Εκτίμηση επιφάνειας των δύο διαφοροποιήσεων του κελιού αθροιστή 3 σε 1.

cell adder 3 σε 1			
χαμηλό κόστος		χαμηλή καθυστέρηση	
Στοιχεία	Area (Ag)	Στοιχεία	Area (Ag)
Y_{-2}	3	c_0 (είσοδοι)	35
F.A.(carry)	5	4 H.A.	12
4 H.A.	12	*	2
*	2	2 XOR	4
2 XOR	4		
Σύνολο	26	Σύνολο	53

Οι λεπτομέρειες του *κελιού rsd*, παρουσιάζονται στον πίνακα:

Πίνακας 4.3.7 Εκτίμηση καθυστέρησης, επιφάνειας του κελιού rsd.

	r^+	r^-	cell rsd
	Y_i		
	F.A.(carry)	F.A.(sum)	
Area (Ag)	10		10
Delay (Tg)	8	9	9

Ενώ για το πρώτο *κελί rsd*, που δεν υπολογίζει *κρατούμενο*, έχουμε:

Πίνακας 4.3.8 Εκτίμηση καθυστέρησης, επιφάνειας του πρώτου κελιού rsd.

	r^-	cell rsd1
	Y-2 F.A.(sum)	
Area (Ag)	6	6
Delay (Tg)	9	9

Χρησιμοποιώντας τα ανωτέρω, υπολογίζουμε την καθυστέρηση και την επιφάνεια για το σχέδιο χαμηλού κόστους:

Πίνακας 4.3.9 Εκτίμηση καθυστέρησης σχεδίου χαμηλής επιφάνειας.

	diploma cost_opt						
	n	8	16	24	32	53	64
Scale D (n)	6 + 2 log2(n-2)	12	14	16	16	18	18
n cell adder 3:1	12n	96	192	288	384	636	768
Delay (Tg)	12n	108	206	304	400	654	786

Πίνακας 4.3.10 Εκτίμηση επιφάνειας σχεδίου χαμηλής επιφάνειας.

	diploma cost_opt						
	n	8	16	24	32	53	64
Scale D (n)	5(n-2) + 3(n-2) log2(n-2)/2 + 3n	72	181	314	426	797	967
Scale X (n)	3(n-3)+3n = 6n-9	39	87	135	183	309	375
n Control	7n	56	112	168	224	371	448
n cell adder 3:1	26n	208	416	624	832	1378	1664
n cell rsd1	6n	48	96	144	192	318	384
n(n-2) cell rsd	10n(n-2)	480	2240	5280	9600	27030	39680
Area (Ag)	10n^2+19n	903	3132	6665	11457	30203	43518

Αντίστοιχα, υπολογίζουμε για το σχέδιο χαμηλής καθυστέρησης:

Πίνακας 4.3.11 Εκτίμηση καθυστέρησης σχεδίου χαμηλής καθυστέρησης.

	diploma delay_opt						
	n	8	16	24	32	53	64
Scale D (n)	6 + 2 log2(n-2)	12	14	16	16	18	18
n cell adder 3:1	10n	80	160	240	320	530	640
Delay (Tg)	10n	92	174	256	336	548	658

Πίνακας 4.3.12 Εκτίμηση επιφάνειας σχεδίου χαμηλής καθυστέρησης.

	diploma delay_opt						
	n	8	16	24	32	53	64
Scale D (n)	5(n-2) + 3(n-2) log2(n-2)/2 + 3n	72	181	314	426	797	967
Scale X (n)	3(n-3)+3n = 6n-9	39	87	135	183	309	375
n Control	7n	56	112	168	224	371	448
n cell adder 3:1	53n	424	848	1272	1696	2809	3392
n cell rsd1	6n	48	96	144	192	318	384
n(n-2) cell rsd	10n(n-2)	480	2240	5280	9600	27030	39680
Area (Ag)	10n^2+19n	1119	3564	7313	12321	31634	45246

4.4 Σύγκριση σχεδίων

Συνοψίζοντας όλα τα αποτελέσματα σε δύο πίνακες, για καθυστέρηση και επιφάνεια, μπορούμε να συγκρίνουμε τελικά τα σχέδια:

Πίνακας 4.4.1 Σύγκριση σχεδίων ως προς την καθυστέρηση.

Delay (Tg)	n					
Σχέδιο	8	16	24	32	53	64
Εργασία [68]	76	142	208	272	442	530
Εργασία [29]	96	192	288	384	636	768
diploma cost_opt	108	206	304	400	654	786
diploma delay_opt	92	174	256	336	548	658

Πίνακας 4.4.2 Σύγκριση σχεδίων ως προς την επιφάνεια.

Area (Ag)	n					
Σχέδιο	8	16	24	32	53	64
Εργασία [68]	1067	4112	9101	15989	43282	62834
Εργασία [29]	1184	3648	7392	12416	31694	45312
diploma cost_opt	903	3132	6665	11457	30203	43518
diploma delay_opt	1119	3564	7313	12321	31634	45246

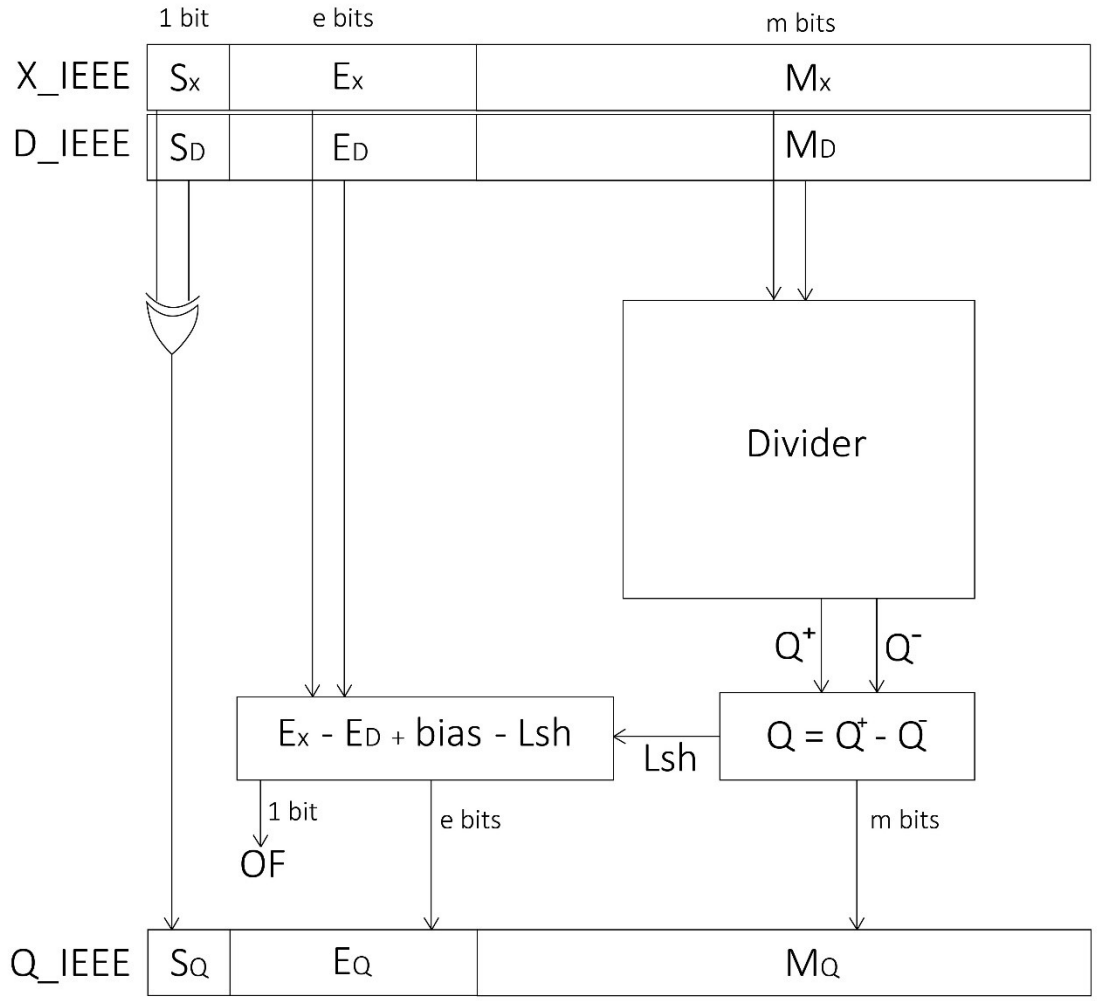
Αξίζει να σημειωθεί, ότι ενώ η εργασία [68] φαίνεται θεωρητικά αρκετά ταχύτερη, πρακτικά δεν είναι εύκολο να βελτιστοποιηθεί περαιτέρω κατά τη σύνθεση. Αυτό οφείλεται στο γεγονός, ότι το κρίσιμο μονοπάτι καταλαμβάνει όλη τη δομική σειρά του πίνακα. Αντίθετα το *κελί ελέγχου* της εργασίας [29], και το *κελί αθροιστή 3 σε 1* της παρούσης εργασίας, που ορίζουν αντίστοιχα το κρίσιμο μονοπάτι, μπορούν να βελτιστοποιηθούν σημαντικά κατά τη σύνθεση.

Όσον αφορά την επιφάνεια, το παρόν σχέδιο φαίνεται καλύτερο από τα τρία. Κατά την σύνθεση όμως, είναι πιθανό να χρησιμοποιηθεί επιπλέον υλικό για την επιτάχυνσή του.

5 ΠΕΙΡΑΜΑΤΙΚΗ ΑΝΑΛΥΣΗ ΥΛΟΠΟΙΗΣΕΩΝ

Τα σχέδια που περιγράψαμε στο κεφάλαιο 3, υλοποιήθηκαν παραμετρικά στη γλώσσα περιγραφής υλικού Verilog. Για την υλοποίηση του προτεινόμενου σχεδίου, δοκιμάστηκαν αρκετές βελτιστοποιήσεις και επιλέχθηκε η αποδοτικότερη. Συγκεκριμένα, το c_0 και το επόμενο κρατούμενο της εικόνας 3.3.5-b, εκφράσθηκαν μέσω του λογισμικού [69] σε μορφή αθροίσματος γινομένων, με εξάρτηση μόνο από τις εισόδους του βήματος j. Το κύκλωμα παραγωγής του πολλαπλάσιου του *διαιρέτη* εκφράσθηκε επίσης σε άθροισμα γινομένων όπως αναφέρθηκε κατά τη σχεδίαση.

Σε αντίθεση με την θεωρητική ανάλυση, κάθε σχέδιο, υλοποιήθηκε έτσι ώστε να δέχεται και να επιστρέφει ορίσματα κατά τα IEEE πρότυπα. Αναπτύχθηκε λοιπόν επιπλέον υλικό, επίσης παραμετρικά, που απεικονίζεται στην εικόνα 5.1. Κάθε υλοποίηση διαφέρει στο κομμάτι του Divider, ενώ το υπόλοιπο κύκλωμα είναι ταυτόσημο για κάθε υλοποίηση που εξετάσαμε. Σημειώνεται, ότι δοκιμάστηκε η υλοποίηση της μετατροπής του πηλίκου, με χρήση του αλγορίθμου on-the-fly. Παρ' όλα αυτά δεν χρησιμοποιήθηκε, καθώς προσθέτει αρκετή επιπλέον επιφάνεια, χωρίς αντίστοιχο κέρδος σε καθυστέρηση. Εάν επρόκειτο για ακολουθιακές υλοποιήσεις όμως, θα ήταν σίγουρα συμφέρουσα.



Εικόνα 5.4.4.1

Τα κυκλώματα προσομοιώθηκαν αρχικά με χρήση του λογισμικού ModelSim, για να επιβεβαιωθεί η ορθότητα των αποτελεσμάτων τους. Για τον σκοπό αυτό λήφθηκε ικανό

δείγμα δοκιμών, με την χρήση γεννήτριας τυχαίων αριθμών, ενώ χρησιμοποιήθηκαν και τιμές στα όρια λειτουργίας των αλγορίθμων.

Ακολούθησε η σύνθεση των κυκλωμάτων για μονή και διπλή ακρίβεια, μέσω του Synopsys Design Compiler, με χρήση της βιβλιοθήκης κελιών tsmc 65nm και της εντολής compile_ultra για καθολική βελτιστοποίηση. Για κάθε υλοποίηση πραγματοποιήθηκε δειγματοληψία, αρχίζοντας από την εκάστοτε ελάχιστη συχνότητα ρολογιού (πίνακας 5.1), και καταλήγοντας σε μία κοινή και ικανή απόσταση από αυτήν, για την εξαγωγή περαιτέρω συμπερασμάτων. Τέλος, αναπτύχθηκαν συγκριτικά γραφήματα επιφάνειας-καθυστέρησης, κατανάλωσης-καθυστέρησης και επιπέδων λογικής του κρίσιμου μονοπατιού, τα οποία παρουσιάζονται στην συνέχεια.

5.1 Σύγκριση καθυστέρησης υλοποιήσεων

Αρχικά στον πίνακα 5.1.1, παρουσιάζεται η ελάχιστη καθυστέρηση που υλοποιήθηκε κάθε κύκλωμα, για μονή και διπλή ακρίβεια. Οι αριθμοί κινητής-υποδιαστολής είναι 32 bits και 64 bits αντίστοιχα, αλλά το κρίσιμο μονοπάτι ορίζεται ουσιαστικά από το κύκλωμα του διαιρέτη, που εξαρτάται από τα σημαντικά μέρη τους με $n = 24$ και $n = 53$ αντίστοιχα. Ακολουθεί ανατύπωση τμήματος της θεωρητικής σύγκρισης στον πίνακα 5.1.2. Συγκρίνοντας τα πειραματικά αποτελέσματα, παρατηρούμε ότι όντως η εργασία [68], στο εξής 1, είναι ταχύτερη όπως βρέθηκε και θεωρητικά. Η διαφορά όμως με την προτεινόμενη εργασία είναι ελάχιστη σε σχέση με τη θεωρητική εκτίμηση.

Η εργασία 1 είναι μεν ταχύτερη, αλλά το κρίσιμο μονοπάτι της διασπείρεται σε όλο το μήκος των κελιών 1, σε αντίθεση με την προτεινόμενη εργασία που το κρίσιμο μονοπάτι συγκεντρώνεται στη διάδοση κρατουμένου. Επίσης, στο σχέδιο της εργασίας 1 χρησιμοποιούνται πύλες έως τριών εισόδων, σε αντίθεση με την προτεινόμενη εργασία που χρησιμοποιούνται αρκετά μεγάλα αθροίσματα γινομένων. Η εργασία [29], στο εξής 2, συνδυάζει τα ανωτέρω χαρακτηριστικά, καθώς το κρίσιμο μονοπάτι της εξαρτάται αρχικά από ένα συγκεντρωμένο και μεγάλο άθροισμα γινομένων, και στη συνέχεια διασπείρεται σε όλο το μήκος των κελιών rsd.

Σύμφωνα με τα ανωτέρω χαρακτηριστικά, συμπεραίνουμε ότι ένα συγκεντρωμένο κρίσιμο μονοπάτι μπορεί να βελτιστοποιηθεί σημαντικά κατά τη σύνθεση. Ειδικά στην περίπτωση ενός μεγάλου αθροίσματος γινομένων, η θεωρητική ανάλυση θα το διασπάσει σε πολλά επίπεδα. Στην πράξη όμως, το άθροισμα γινομένων είναι δύο επίπεδα πυλών πολλών εισόδων, που ενδέχεται με χρήση πιο ακριβού και γρήγορου υλικού, να επιταχυνθεί.

Πίνακας 5.1.1 Πειραματικές μετρήσεις καθυστέρησης.

Σχέδιο	ακρίβεια			
	μονή		διπλή	
	Delay (ns)	freq(%)	Delay (ns)	freq(%)
Εργασία 1	3,64	100,0	7,97	100,0
Εργασία 2	4,58	79,5	10,47	76,1
Προτεινόμενη Εργασία	3,7	98,4	8,1	98,4

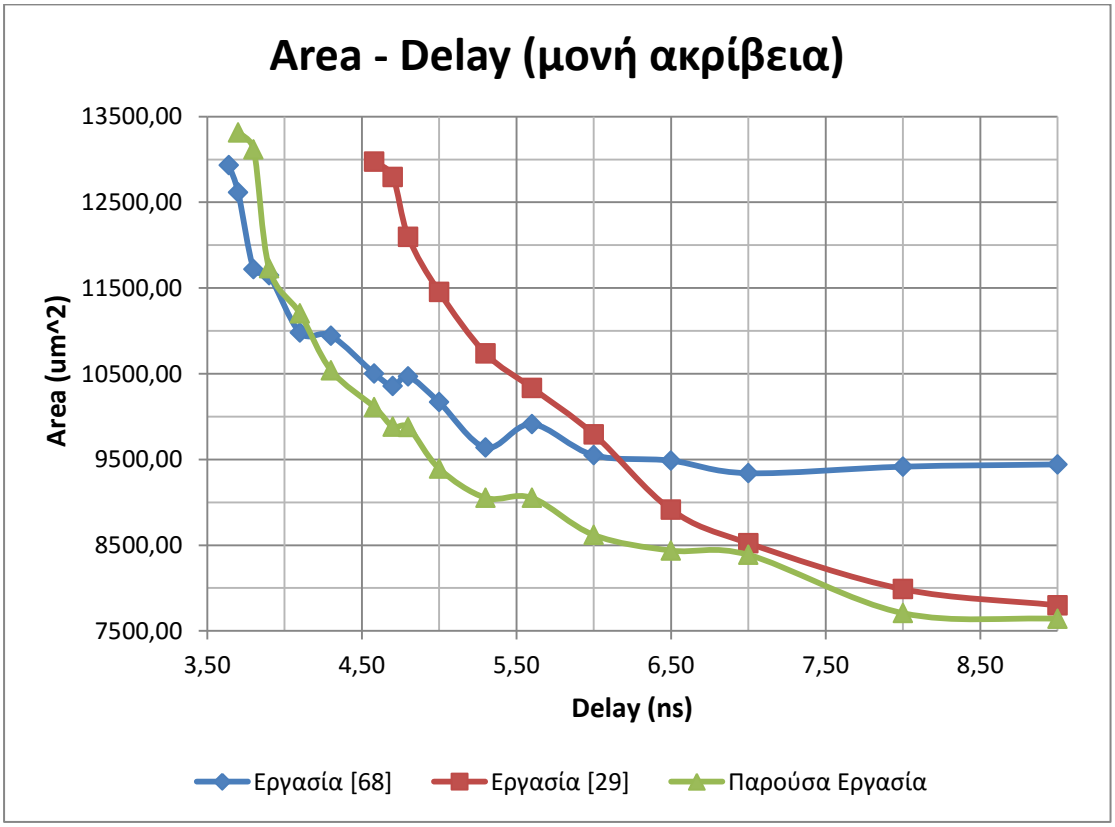
Πίνακας 5.1.2 Θεωρητικές εκτιμήσεις καθυστέρησης.

Σχέδιο	n			
	24		53	
	Delay (Tg)	freq(%)	Delay (Tg)	freq(%)
Εργασία 1	208	100,0	442	100,0
Εργασία 2	288	72,2	636	69,5
diploma cost_opt	304	68,4	654	67,6
diploma delay_opt	256	81,3	548	80,7

5.2 Σύγκριση υλοποιήσεων μονής ακρίβειας

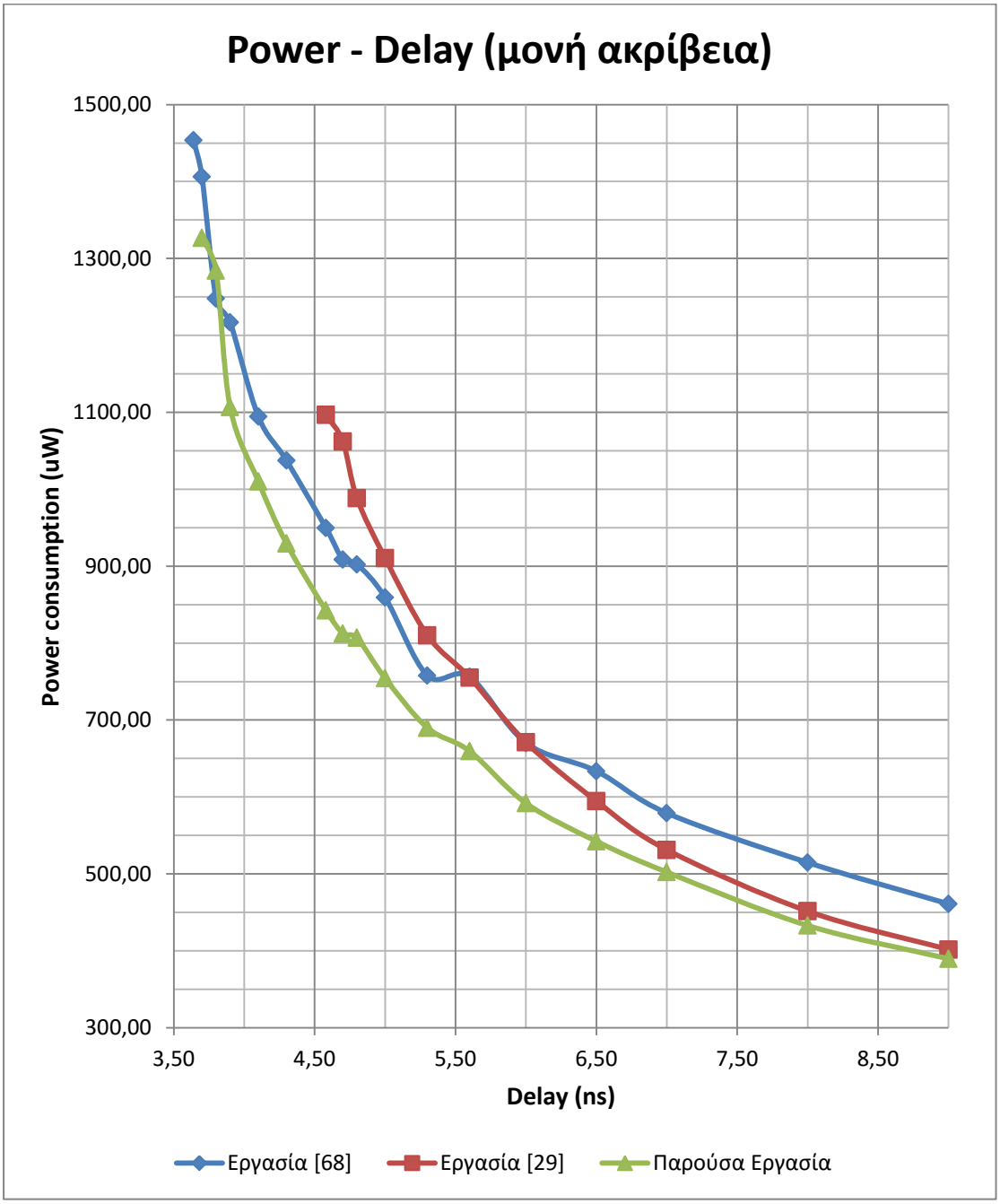
Στην εικόνα 5.2.1, παρουσιάζεται η σύγκριση της επιφάνειας για διαφορετικές τιμές καθυστέρησης, σε αριθμούς μονής ακρίβειας. Παρατηρούμε ότι σε γενικές γραμμές, οι θεωρητικές εκτιμήσεις επιβεβαιώνονται. Η προτεινόμενη υλοποίηση καταλαμβάνει λιγότερη επιφάνεια, εκτός από την περιοχή κοντά στο κρίσιμο μονοπάτι της. Αυτό εξηγείται, από την επιβάρυνση του κρίσιμου μονοπατιού με επιπλέον ταχύτερο υλικό, όπως υποτέθηκε.

Είναι λογικό, οι υλοποιήσεις να επιβαρύνονται με επιπλέον επιφάνεια κοντά στο κρίσιμο μονοπάτι τους, έτσι η εργασία 2 καταλαμβάνει περισσότερη επιφάνεια στα 4,58 ns, ενώ μετά τα 6 ns, απαιτεί αρκετά λιγότερη από την εργασία 1. Οι δύο πολυπλέκτες σε κάθε κελί της εργασίας 1, θέτουν το κατώτερο όριο επιφάνειας, αρκετά ψηλότερα από τις άλλες δύο εργασίες.



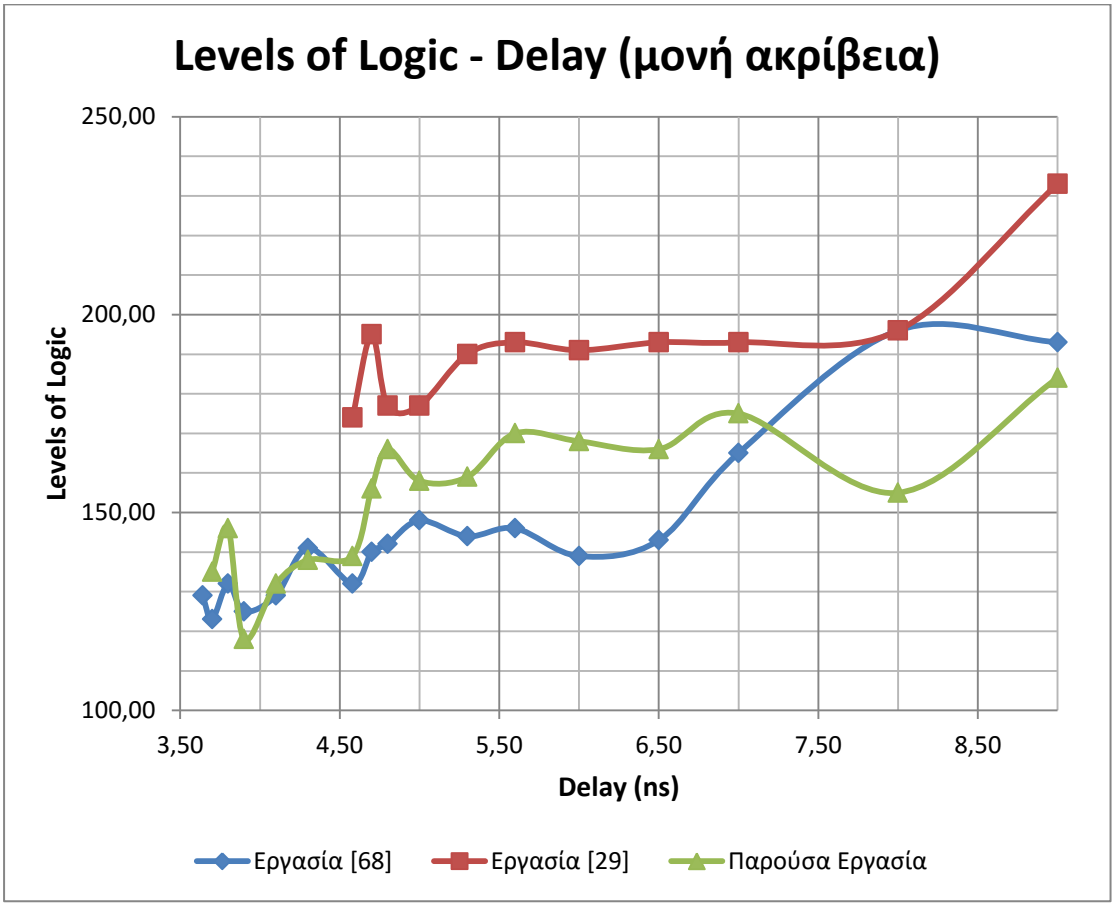
Εικόνα 5.2.1 Σύγκριση επιφάνειας ως προς την καθυστέρηση για μονή ακρίβεια.

Όπως και στην περίπτωση της επιφάνειας, η κατανάλωση αυξάνεται εκθετικά κοντά στο κρίσιμο μονοπάτι των υλοποιήσεων (εικόνα 5.2.2). Σε κάθε περίπτωση, διακρίνουμε μία κοντινή σχέση μεταξύ επιφάνειας-κατανάλωσης, με τις καμπύλες για κάθε υλοποίηση να είναι όμοιες μεταξύ τους. Ενώ όμως οι τρεις υλοποιήσεις χρησιμοποιούν περίπου την ίδια επιφάνεια κοντά στο κρίσιμο μονοπάτι τους, δεν ισχύει το ίδιο για την κατανάλωση. Μάλιστα για καθυστέρηση 3.7 ns, η εργασία 1 καταναλώνει περισσότερη ισχύ από την προτεινόμενη εργασία, καταλαμβάνοντας λιγότερη επιφάνεια. Ακόμη, φαίνεται να υπάρχει όριο στη μείωση της επιφάνειας, ενώ η κατανάλωση συνεχίζει να μειώνεται με την αύξηση της καθυστέρησης.



Εικόνα 5.2.2 Σύγκριση κατανάλωσης ισχύος ως προς την καθυστέρηση για μονή ακρίβεια.

Στην εικόνα 5.2.3, παρουσιάζονται τα επίπεδα λογικής του κρίσιμου μονοπατιού για τις διάφορες τιμές καθυστέρησης. Πρόκειται για μέτρηση, άμεσα συγκρίσιμη με την καθυστέρηση σε T_g που εκτιμήθηκε θεωρητικά. Κατά την θεωρητική ανάλυση, μετρήσαμε τα επίπεδα λογικής του σχεδίου, θεωρώντας πύλες δύο εισόδων. Κατά την πειραματική ανάλυση, βλέπουμε ότι στην πράξη τα επίπεδα λογικής μειώνονται, χρησιμοποιώντας πύλες περισσότερων εισόδων. Ενδεικτικά, παρουσιάζονται οι εκτιμήσεις και οι αντίστοιχες ελάχιστες μετρήσεις των υλοποιήσεων, στον πίνακα 5.2.1. Βλέπουμε ότι τα επίπεδα λογικής μειώνονται μέχρι και 53,9% για την προτεινόμενη εργασία, φτάνοντας πιο χαμηλά από την εργασία 1, που υποδεικνύει και τον τρόπο επιτάχυνσής της. Τέλος, παρατηρούμε αρκετές διακυμάνσεις όσο πλησιάζει κάθε καμπύλη στο κρίσιμο μονοπάτι της. Αυτό υποδεικνύει την προσπάθεια εύρεσης ισορροπίας, μεταξύ επιφάνειας, κατανάλωσης και αριθμού εισόδων των πυλών.



Εικόνα 5.2.3 Σύγκριση επιπέδων λογικής ως προς την καθυστέρηση για μονή ακρίβεια.

Πίνακας 5.2.1 Εκτιμήσεις-μετρήσεις επιπέδων λογικής.

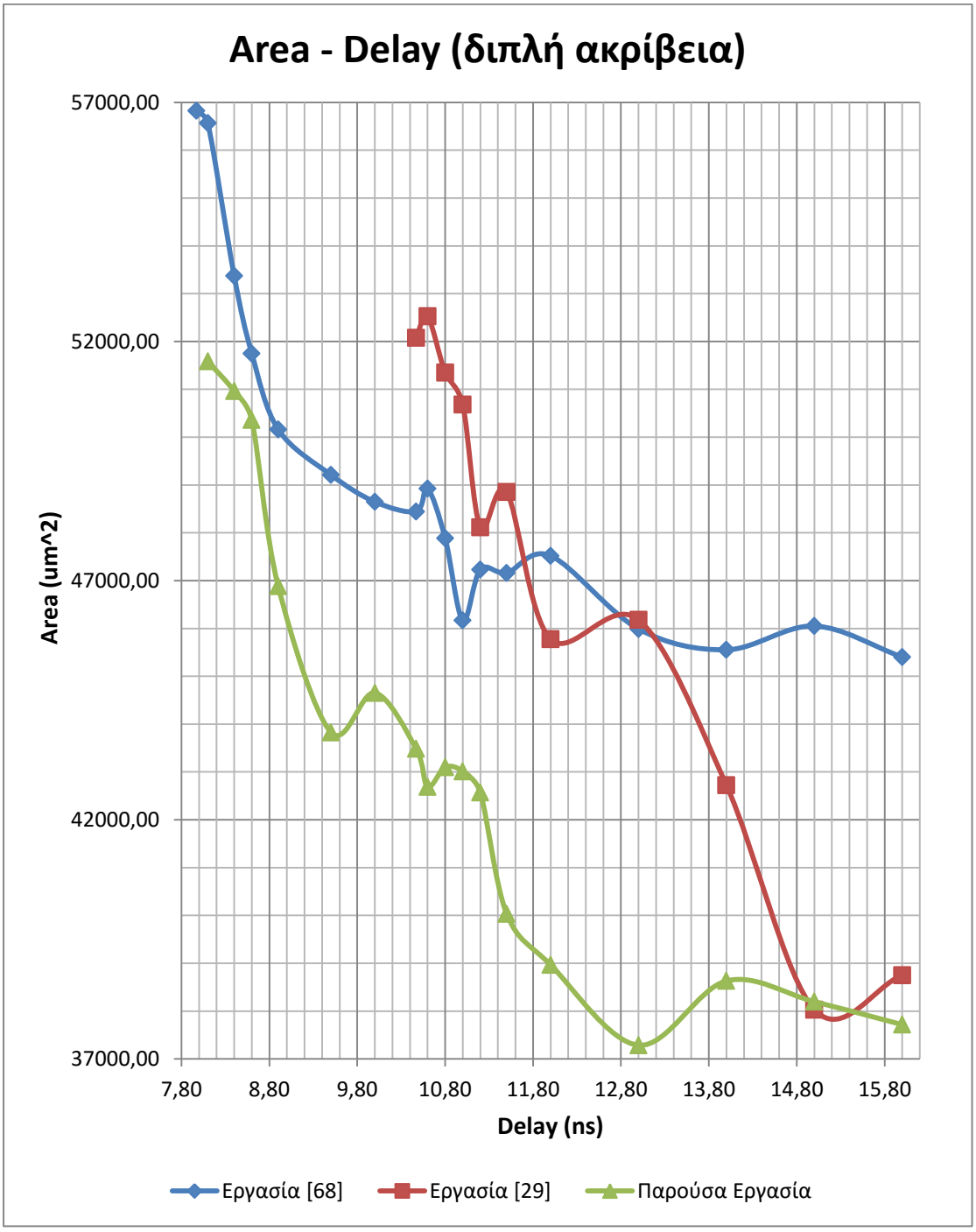
	Delay (Tg)	Levels of Logic	μείωση(%)
Εργασία 1	208	123	40,9
Εργασία 2	288	174	39,6
Προτεινόμενη Εργασία	256	118	53,9

5.3 Σύγκριση υλοποιήσεων διπλής ακρίβειας

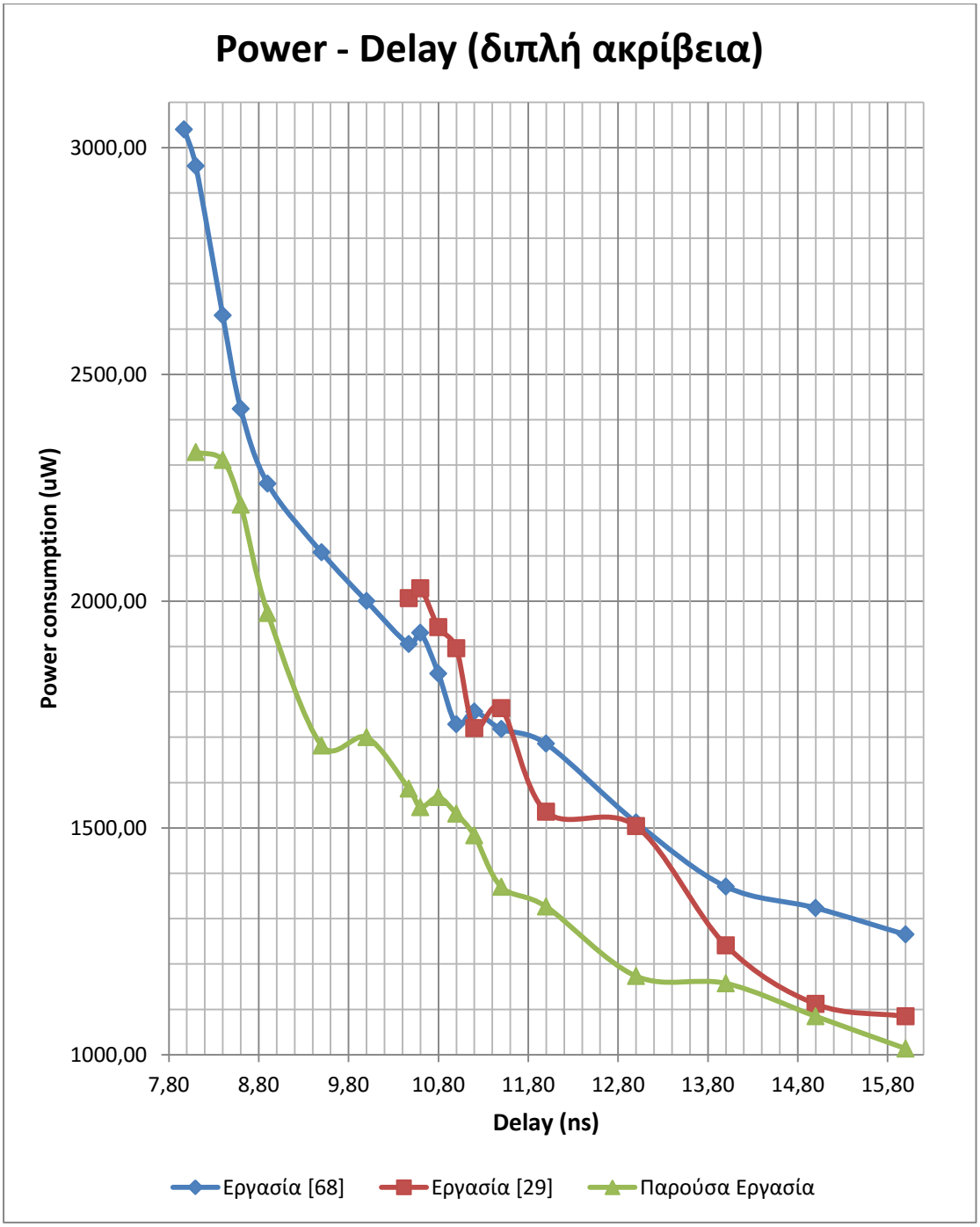
Στις εικόνες 5.3.1 και 5.3.2, παρουσιάζεται η σύγκριση της επιφάνειας και της κατανάλωσης αντίστοιχα για διαφορετικές τιμές καθυστέρησης, σε αριθμούς διπλής

ακρίβειας. Και στα δύο γραφήματα, παρατηρούμε αρκετές διακυμάνσεις των καμπυλών. Αυτό οφείλεται ίσως στη διαδικασία της σύνθεσης, και ενισχύεται από το μεγάλο μέγεθος των κυκλωμάτων.

Σε αντίθεση με την μονή ακρίβεια, η προτεινόμενη εργασία καταλαμβάνει γενικά την λιγότερη επιφάνεια, ακόμα και στο κρίσιμο μονοπάτι της. Το ίδιο ισχύει και με την κατανάλωση, με την εργασία 1 να αποκλίνει αρκετά κοντά στο κρίσιμο μονοπάτι. Η εργασία 2 υπερτερεί της 1, μόνο για χαμηλές ταχύτητες, σε επιφάνεια και κατανάλωση.

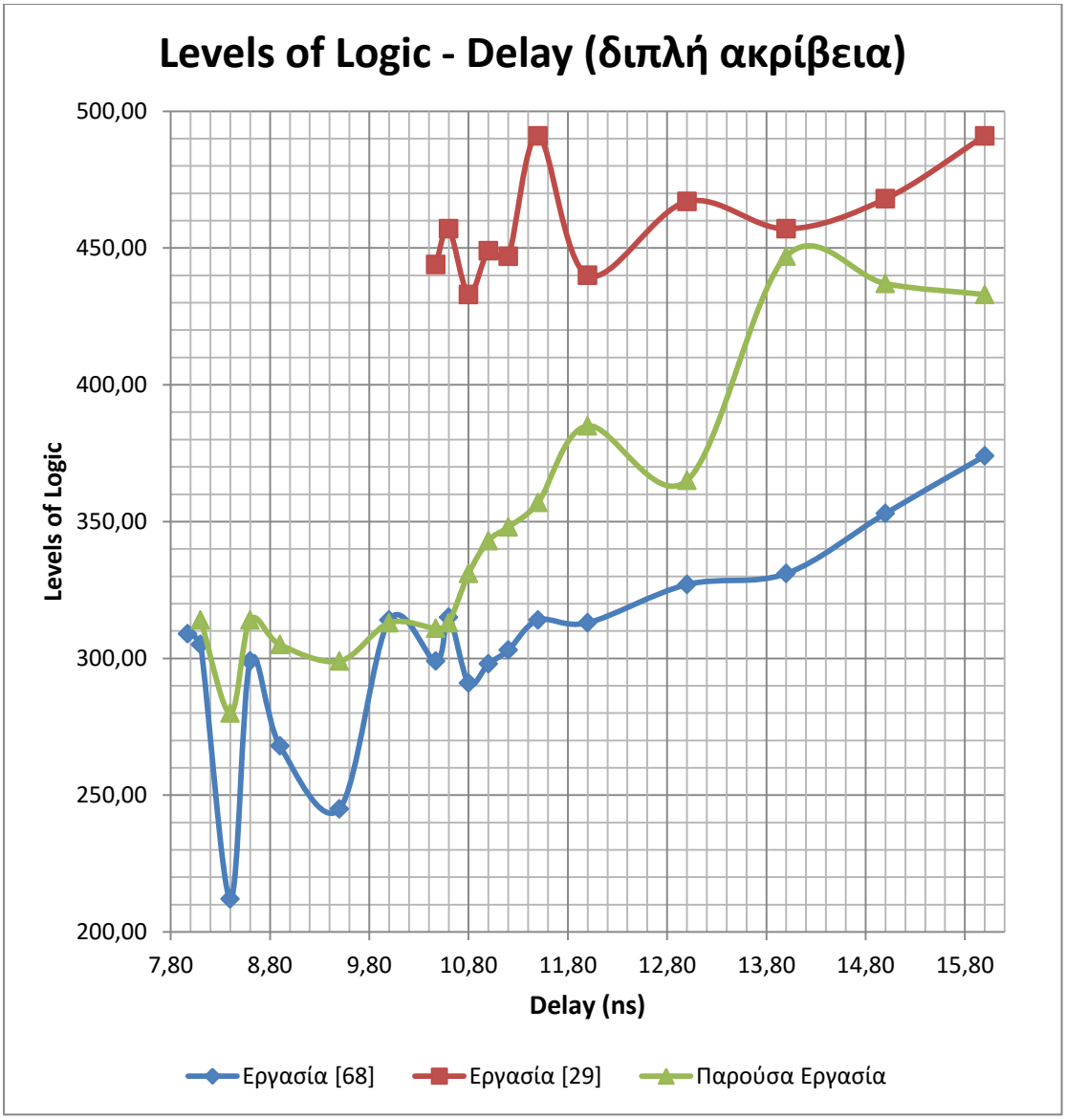


Εικόνα 5.3.1 Σύγκριση επιφάνειας ως προς την καθυστέρηση για διπλή ακρίβεια.



Εικόνα 5.3.2 Σύγκριση κατανάλωσης ισχύος ως προς την καθυστέρηση για διπλή ακρίβεια.

Τέλος, τα επίπεδα λογικής της προτεινόμενης υλοποίησης, βρίσκονται κοντά σε αυτά της εργασίας 2 για χαμηλές ταχύτητες, ενώ παρατηρείται σημαντική απόκλιση όσο μειώνεται η καθυστέρηση. Αναμενόμενα, τα επίπεδα λογικής της εργασίας 2 βρίσκονται αρκετά χαμηλότερα για χαμηλές ταχύτητες, αλλά συγκλίνουν με της προτεινόμενης για υψηλές ταχύτητες. Παρατηρούνται επίσης μεγάλες διακυμάνσεις κοντά στο κρίσιμο μονοπάτι, το οποίο τελικά επιτυγχάνεται με υλοποίηση περισσότερων επιπέδων λογικής.



Εικόνα 5.3.3 Σύγκριση επιπέδων λογικής ως προς την καθυστέρηση για διπλή ακρίβεια.

6 ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΕΠΕΚΤΑΣΕΙΣ

Στο παρόν κεφάλαιο συνοψίζεται η παρουσίαση της διπλωματικής εργασίας.

6.1 Σύνοψη

Συνοψίζοντας, στα πλαίσια της παρούσας διπλωματικής εργασίας, διερευνήσαμε την ανάλυση και βελτιστοποίηση των αλγορίθμων SRT. Περιγράψαμε δύο παραλλαγές της κατηγορίας από τη βιβλιογραφία, και αναπτύξαμε μία τρίτη παραλλαγή. Τα ανωτέρω σχέδια αναλύθηκαν και συγκρίθηκαν θεωρητικά, και στη συνέχεια υλοποιήθηκαν και συγκρίθηκαν πειραματικά.

6.2 Συμπεράσματα

Κατόπιν της πειραματικής ανάλυσης, εξήχθησαν συμπεράσματα για την αποδοτικότητα κάθε υλοποίησης.

Στα πλαίσια της καθυστέρησης, πλεονεκτεί ελάχιστα το σχέδιο της εργασίας 1, σε αντίθεση με τις εκτιμήσεις της θεωρητικής ανάλυσης για μεγάλη διαφορά, από την προτεινόμενη υλοποίηση και την εργασία 2. Ακολουθεί με μικρή διαφορά η παρούσα εργασία και με αρκετή διαφορά η εργασία 2.

Όσον αφορά την επιφάνεια, η προτεινόμενη υλοποίηση υπερτερεί των άλλων εργασιών για τις διάφορες τιμές κύκλου ρολογιού στους αριθμούς διπλής ακρίβειας. Μειονεκτεί μόνο κοντά στο κρίσιμο μονοπάτι σε σχέση με την εργασία 1, για αριθμούς μονής ακρίβειας.

Κοντά στο κρίσιμο μονοπάτι της εργασίας 2, αυτή μειονεκτεί στις παραμέτρους της επιφάνειας και της κατανάλωσης σε σχέση με την εργασία 1. Με την ελάττωση όμως της καθυστέρησης, καθίσταται περισσότερο αποδοτική. Και σε ότι αφορά την κατανάλωση, η προτεινόμενη υλοποίηση υπερτερεί για κάθε περίπτωση.

Τέλος, μπορούμε να χαρακτηρίσουμε συνολικά αποδοτικότερη την προτεινόμενη υλοποίηση, καθώς επιτυγχάνει υψηλή ταχύτητα λειτουργίας με σχετικά χαμηλή επιφάνεια και κατανάλωση.

6.3 Μελλοντικές επεκτάσεις

Η συγκεκριμένη εργασία, αποτέλεσε μία πρώτη προσπάθεια αποδοτικής υλοποίησης αλγορίθμων διαίρεσης. Εφαρμόζοντας συνδυασμό από τεχνικές επιτάχυνσης που αναφέρθηκαν στην ενότητα 2.4, θα μπορούσε να επεκταθεί ο προτεινόμενος αλγόριθμος. Σε αυτό μπορεί να λειτουργήσει θετικά, η επεκτασιμότητα του αλγορίθμου, καθώς η υβριδική του αναπαράσταση παρεχεί αρκετά οφέλη.

7 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] *ModelSim SE-64 6.5c,* Mentor Graphics Corporation.

[2] «Synopys Design Compiler».

[3] M. D. Ercegovac και T. Lang, «Digital Arithmetic,» 2004. [Ηλεκτρονικό]. [Πρόσβαση 2016].

[4] A. Omondi και B. Premkumar, Residue number systems : theory and implementation, London: Imperial College Press, 2007.

[5] H. L. Garner, «The residue number system,» σε *western joint computer conference*, New York, 1959.

[6] U. Meyer-Baese, «Computer Arithmetic,» σε *Digital Signal Processing with Field Programmable Gate Arrays*, Berlin, Springer-Verlag, 2014, pp. 57-178.

[7] I. Koren, Computer Arithmetic Algorithms, Natick, Massachusetts: A K Peters, 2002.

[8] K. Pekmestzi, DIGITAL VLSI SYSTEMS, Athens: NTUA Lectures Notes, 2003.

[9] K. K. Parhi, «Redundant Arithmetic,» σε *VLSI digital signal processing systems : design and implementation*, New York, John Wiley, 1999, pp. 529-558.

[10] «IEEE standard for floating-point arithmetic,» Institute of Electrical and Electronics Engineers, New York, 2008.

[11] T. Jebelean, «Practical Integer Division with Karatsuba Complexity,» σε *ISSAC 97*, Maui, Hawaii, 1997.

[12] S. F. Oberman και M. J. Flynn, «AN ANALYSIS OF DIVISION ALGORITHMS AND IMPLEMENTATIONS,» Stanford, California, 1995.

[13] «en.wikipedia.org,» [Ηλεκτρονικό]. Available: en.wikipedia.org/wiki/Division_(mathematics). [Πρόσβαση 2016].

[14] D. G. Bailey, «Space Efficient Division on FPGAs,» σε *Electronics New Zealand Conference*, 2006.

[15] P. Fenwick, «High-radix Division with Approximate Quotient-digit Estimation,» *Journal of Universal Computer Science,* τόμ. I, αρ. 1, pp. 2-22, 1995.

[16] Y. Tsunekawa, M. Hinosugi και M. Miura, «Design and VLSI evaluation of a high-speed cellular array divider with a selection function,» *Electrical Engineering in Japan,* τόμ.

124, αρ. 4, p. 47–55, 1998.

[17] D. L. Harris, S. F. Oberman και M. A. Horowitz, «Srt Division Architectures and Implementations,» σε *Computer Arithmetic, 13th IEEE Symposium on*, Asilomar, CA, 1997.

[18] J. E. Robertson, «A New Class of Digital Division Methods,» *IRE Transactions on Electronic Computers,* Τόμ. %1 από %2EC-7, αρ. 3, pp. 218-222, 1958.

[19] M. D. Ercegovac και T. Lang, «On-the-Fly Conversion of Redundant into Conventional Representations,» *IEEE Transactions on Computers,* τόμ. 36, αρ. 7, pp. 895-897, 1987.

[20] A. E. Bashagha, «Area-efficient nonrestoring radix-2^k division,» *Digital Signal Processing,* τόμ. 15, αρ. 4, pp. Pages 367-381 , 2005.

[21] O. L. Macsorley, «High-Speed Arithmetic in Binary Computers,» *Proceedings of the IRE,* pp. 67 - 91, Jan. 1961.

[22] K. D. Tocher, «TECHNIQUES OF MULTIPLICATION AND DIVISION FOR AUTOMATIC BINARY COMPUTERS,» *The Quarterly Journal of Mechanics and Applied Mathematics,* τόμ. 11, αρ. 3Pp, pp. 364-384, 1958.

[23] D. E. Atkins, «Higher-Radix Division Using Estimates of the Divisor and Partial Remainders,» *IEEE Transactions on Computers,* Τόμ. %1 από %2C-17, αρ. 10, pp. 925-934, 1968.

[24] K. G. Tan, «The theory and implementation of high radix division,» σε *IEEE 4th Symposium on Computer Arithmetic (ARITH)*, Santa Monica, California, 1978.

[25] M. D. Ercegovac και T. Lang, Division and Square Root: Digit-Recurrence Algorithms and Implementations, Norwell, MA: Kluwer Academic, 1994.

[26] W.-K. Chen, The VLSI Handbook, Chicago, Illinois: CRC Press, 2000.

[27] S. F. Oberman, «DESIGN ISSUES IN HIGH PERFORMANCE FLOATING POINT ARITHMETIC UNITS,» Stanford, California, 1996.

[28] A. Avizienis, «Signed-Digit Numbe Representations for Fast Parallel Arithmetic,» *IRE Transactions on Electronic Computers,* Τόμ. %1 από %2EC-10, αρ. 3, pp. 389 - 400, 1961.

[29] E. V. T. D. P. J. A. Vandemeulebroecke, «A New Carry-Free Division Algorithm and Its Application to a Single-Chip 1024-b RSA Processor,» *IEEE Journal of Solid-State Circuits,* τόμ. 25, αρ. 3, pp. 748-756, 1990.

[30] T. N. H. E. T. T. N. T. S. Kuninobu, «Design of high speed MOS multiplier and divider using redundant binary representation,» σε *Computer Arithmetic (ARITH), IEEE 8th*

Symposium on, Como, 1987.

[31] M. H. Ted E. Williams, «SRT division diagrams and their usage in designing intergrated circuits for division,» Stanford, CA, 1986.

[32] G. Cornetta, «Division with Speculation of the Quotient Digits,» Barcelona, 1998.

[33] D. E. Atkins, «Design of the Arithmetic Units of ILLIAC III: Use of Redundancy and Higher Radix Methods,» *IEEE Transactions on Computers,* Τόμ. %1 από %2C-19, αρ. 8, pp. 720-733, 1970.

[34] G. S. Taylor, «Radix 16 SRT dividers with overlapped quotient selection stages,» σε *Computer Arithmetic (ARITH), IEEE 7th Symposium on*, Urbana, IL, 1985.

[35] T. L. M. D. Ercegovac, «Simple radix-4 division with operands scaling,» *IEEE Transactions on Computers,* τόμ. 39, αρ. 9, pp. 1204-1208, 1990.

[36] A. Svoboda, «An algorithm for division,» *Information Processing Machines,* τόμ. 9, pp. 29-34, 1963.

[37] C. Tung, «A Division Algorithm for Signed-Digit Arithmetic,» *IEEE Transactions on Computers,* Τόμ. %1 από %2C-17, αρ. 9, pp. 887-889, 1968.

[38] P. Montuschi και L. Ciminiera, «Over-redundant digit sets and the design of digit-by-digit division units,» *IEEE Transactions on Computers,* τόμ. 43, αρ. 3, pp. 269-277, 1994.

[39] H. R. Srinivas και K. K. Parhi, «A fast radix-4 division algorithm and its architecture,» *IEEE Transactions on Computers,* τόμ. 44, αρ. 6, pp. 826-831, 1995.

[40] S. F. Oberman και M. J. Flynn, «Minimizing the complexity of SRT tables,» *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* τόμ. 6, αρ. 1, pp. 141-149, 1998.

[41] N. Quach και M. Flynn, «A radix-64 floating-point divider,» Computer Systems Laboratory, Stanford, 1992.

[42] C. S. Wallace, «A Suggestion for a Fast Multiplier,» *IEEE Transactions on Electronic Computers,* Τόμ. %1 από %2EC-13, αρ. 1, pp. 14-17, 1964.

[43] R. Alverson, «Integer division using reciprocals,» σε *Computer Arithmetic, 10th IEEE Symposium on*, Grenoble, 1991.

[44] D. D. Sarma και D. W. Matula, «Measuring the accuracy of ROM reciprocal tables,» σε *Computer Arithmetic, 11th IEEE Symposium on*, Windsor, Ont., 1993.

[45] D. D. Sarma και D. W. Matula, «Faithful bipartite ROM reciprocal tables,» σε *Computer*

Arithmetic, 12th Symposium on, Bath, 1995.

[46] D. Ferrari, «A Division Method Using a Parallel Multiplier,» *IEEE Transactions on Electronic Computers,* Τόμ. %1 από %2EC-16, αρ. 2, pp. 224-226, 1967.

[47] M. J. Flynn, «On Division by Functional Iteration,» *IEEE Transactions on Computers,* Τόμ. %1 από %2C-19 , αρ. 8, pp. 702-706, 1970.

[48] D. L. Fowler και J. E. Smith, «An accurate, high speed implementation of division by reciprocal approximation,» σε *Computer Arithmetic, 9th Symposium on*, Santa Monica, CA, 1989.

[49] M. Ito, N. Takagi και S. Yajima, «Efficient initial approximation and fast converging methods for division and square root,» σε *Computer Arithmetic, 12th Symposium on*, Bath, 1995.

[50] M. Ito, N. Takagi και S. Yajima, «Efficient initial approximation for multiplicative division and square root by a multiplication with operand modification,» *IEEE Transactions on Computers,* τόμ. 46, αρ. 4, pp. 495-498, 1997.

[51] J. Phillips και S. Vassiliadis, «High Performance Dividers with Multiply-Add,» σε *2nd International Conference on Massively Parallel Computing Systems*, Ischia, 1996.

[52] M. J. Schulte, J. Omar και E. E. Swartzlander, «Optimal initial approximations for the Newton-Raphson division algorithm,» *Computing,* τόμ. 53, αρ. 3-4, pp. 233-242, 1994.

[53] E. M. Schwarz και M. J. Flynn, «Hardware starting approximation method and its application to the square root operation,» *IEEE Transactions on Computers,* τόμ. 45, αρ. 12, pp. 1356-1369, 1996.

[54] D. Knuth και A. Wesley, The Art of Computer Programming, 1969.

[55] «http://web.stanford.edu/,» [Ηλεκτρονικό]. Available: http://web.stanford.edu/class/ee486/doc/chap5.pdf. [Πρόσβαση 2015].

[56] S. F. Anderson, E. J. G., G. R. E. και P. D. M., «The IBM system/360 model 91: floating-point execution unit,» *IBM Journal of Research and Development,* τόμ. 11, αρ. 1, pp. 34-53, 1967 .

[57] E. M. Schwarz και M. J. Flynn, «Using a floating-point multiplier's internals for high-radix division and square root,» Computer Systems Laboratory, Stanford, California, 1993.

[58] D. Wong και M. Flynn, «Fast division using accurate quotient approximations to reduce the number of iterations,» *IEEE Transactions on Computers,* τόμ. 41, αρ. 8, pp. 981-995, 1992.

[59] W. S. Briggs και D. W. Matula, «A 17 × 69 bit multiply and add unit with redundant binary feedback and single cycle latency,» σε *Computer Arithmetic, 11th Symposium on*, Windsor, Ont., 1993.

[60] M. D. Ercegovac, T. Lang και P. Montuschi, «Very high radix division with selection by rounding and prescaling,» σε *Computer Arithmetic, 11th Symposium on*, Windsor, Ont., 1993.

[61] M. D. Ercegovac, T. Lang και P. Montuschi, «Very-high radix division with prescaling and selection by rounding,» *IEEE Transactions on Computers,* τόμ. 43, αρ. 8, pp. 909-918, 1994.

[62] T. E. Williams και M. A. Horowitz, «A zero-overhead self-timed 160-ns 54-b CMOS divider,» *IEEE Journal of Solid-State Circuits,* τόμ. 26, αρ. 11, pp. 1651-1661, 1991.

[63] J. Cortadella και T. Lang, «High-radix division and square-root with speculation,» *IEEE Transactions on Computers,* τόμ. 43, αρ. 8, pp. 919-931, 1994.

[64] G. Cornetta και J. Cortadella, «A multi-radix approach to asynchronous division,» σε *Asynchronus Circuits and Systems, Seventh International Symposium on*, Salt Lake City, UT, 2001.

[65] G. Cornetta και J. Cortadella, «A radix-16 SRT division unit with speculation of the quotient digits,» σε *VLSI. Ninth Great Lakes Symposium on*, Ypsilanti, MI, 1999.

[66] C. L. Wey και C. P. Wang, «Design of a fast radix-4 SRT divider and its VLSI implementation,» *IEE Proceedings - Computers and Digital Techniques,* τόμ. 146, αρ. 4, pp. 205-210, 1999.

[67] P. Montuschi και L. Ciminiera, «Quotient prediction without prescaling,» *IEE Proceedings - Computers and Digital Techniques,* τόμ. 142, αρ. 1, pp. 15-22, 1995.

[68] S. E. McQuillan, J. V. McCanny και R. Hamill, «New algorithms and VLSI architectures for SRT division and square root,» σε *Computer Arithmetic. 11th Symposium on*, Windsor, Ont., 1993.

[69] S. Rickman, *Logic Friday v. 1.1.4. Espresso, misII.*