
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής &Υπολογιστών


# Σχεδίαση και Υλοποίηση Φίλτρων FIR Βασισμένων στον Αλγόριθμο Karatsuba

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του


**ΕΥΑΓΓΕΛΟΥ ΚΥΡΙΤΣΗ**

**Επιβλέπων**: Κιαμάλ Πεκμεστζή,

Καθηγητής Ε.Μ.Π.



Αθήνα, Μάρτιος 2016


ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής &Υπολογιστών


# Σχεδίαση και Υλοποίηση Φίλτρων FIR Βασισμένων στον Αλγόριθμο Karatsuba

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του


**ΕΥΑΓΓΕΛΟΥ ΚΥΡΙΤΣΗ**

**Επιβλέπων**: Κιαμάλ Πεκμεστζή,

Καθηγητής Ε.Μ.Π.



Αθήνα, Μάρτιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

# Σχεδίαση και Υλοποίηση Φίλτρων FIR Βασισμένων στον Αλγόριθμο Karatsuba

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

**ΕΥΑΓΓΕΛΟΥ ΚΥΡΙΤΣΗ**

**Επιβλέπων**: Κιαμάλ Πεκμεστζή

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή επιτροπή την ……..

| | | |
|---|---|---|
| ….…… | ...…… | ...…….. |
| Κιαμάλ Πεκμεστζή | Δημήτριος Σούντρης | Γιώργος Οικονομάκος |
| Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. | Καθηγητής Ε.Μ.Π. |

Αθήνα, Μάρτιος 2016

……

ΕΥΑΓΓΕΛΟΣ ΚΥΡΙΤΣΗΣ

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

# Περίληψη

Σκοπός της παρούσας εργασίας είναι η διερεύνηση της αποδοτικής αξιοποίησης του πολλαπλασιαστικού αλγορίθμου Karatsuba στην σχεδίαση φίλτρων πεπερασμένης κρουστικής απόκρισης (Finite Impulse Response-FIR). Ο αλγόριθμος Karatsuba, είναι ένας από τους αλγορίθμους που αναπτύχθηκαν για να αυξήσουν την αποδοτικότητα και να μειώσουν το κόστος του πολλαπλασιασμού μεγάλων αριθμών, διαχωρίζοντας τους τελεστέους σε δύο τμήματα ίσου μήκους.

Σχεδιάστηκε μια νέα αρχιτεκτονική ενός φίλτρου FIR, βασισμένη στον αλγόριθμο Karatsuba. Το κύκλωμα του φίλτρου Karatsuba προκύπτει από την σύνθεση τριών υποφίλτρων μειωμένου δυναμικού εύρους που λειτουργούν παράλληλα. Η συγκεκριμένη αρχιτεκτονική υλοποιήθηκε σε τεχνολογία ASIC, σε direct, transposed και mixed μορφή. Επίσης υλοποιήθηκαν οι αντίστοιχες συμβατικές τοπολογίες με σκοπό την αξιολόγηση της απόδοσης του φίλτρου Karatsuba.

Στην σχεδίαση της mixed μορφής των φίλτρων αξιοποιήθηκε η τεχνική της συνεχούς διοχέτευσης για τον διαχωρισμό των κυκλωμάτων σε δύο στάδια, με σκοπό την μεγιστοποίηση της συχνότητας λειτουργίας. Σε όλες τις υλοποιήσεις των φίλτρων, χρησιμοποιείται ως δομική μονάδα, ένας παράλληλος, Carry-Save (CS) δενδρικός πολλαπλασιαστής Wallace με προ-κωδικοποίηση Modified Booth (MB).

Για την υλοποίηση των φίλτρων χρησιμοποιήθηκε η γλώσσα περιγραφής υλικού Verilog. Ο κώδικας που αναπτύχθηκε είναι παραμετρικός με δύο παραμέτρους, τον αριθμό των σημείων (taps) του φίλτρου και το μήκος λέξης (bits) των δεδομένων εισόδου και των συντελεστών.

Τα κυκλώματα που προέκυψαν προσομοιώθηκαν με βάση μια standard-cell CMOS βιβλιοθήκη της Artisan στα 90nm. Τέλος, έγινε σύγκριση τους ως προς την καθυστέρηση, την επιφάνεια και την κατανάλωση με βάση τα αποτελέσματα της προσομοίωσης.

**Λέξεις κλειδιά :** Ψηφιακή Επεξεργασία Σήματος, Φίλτρα FIR, Αλγόριθμος Karatsuba, Συμβατικός Πολλαπλασιαστής με Modified Booth Κωδικοποίηση, ASIC, CMOS 90nm, Verilog.

## Abstract

The scope of the present thesis is the investigation of the efficient implementation of the Karatsuba multiplication algorithm on the design of Finite Impulse Response (FIR) filters. The Karatsuba algorithm is one of the algorithms developed for increasing the efficiency and reducing the cost of the multiplication of large numbers by splitting the operands in two parts of equal length.

A new architecture of a FIR filter based on Karatsuba algorithm has been designed. The Karatsuba filter circuit is composed by three sub-filters of reduced dynamic range working in parallel. This architecture has been implemented in ASIC, in direct, transposed and mixed form. Also, the respective conventional topologies have been implemented in order to evaluate Karatsuba filter.

The pipelining technique has been used on the design of the mixed form separating the circuits into two stages in order to maximize the operating frequency. A parallel Carry-Save (CS) Wallace tree multiplier with Modified Booth (MB) pre-encoding has been used as a building block in all implementations.

Verilog HDL has been used to describe the filters with the code being parametric with two parameters, the number of taps and the bit width of both inputs and coefficients.

The filters were simulated based on a standard-cell CMOS 90nm library of Artisan. Finally, the designs were evaluated and compared in terms of delay, area and power consumption.



**Keywords:** Karatsuba Algorithm, Digital Signal Processing, FIR filters, Conventional Modified Booth Algorithm, ASIC, CMOS 90nm, Verilog.

*Στην εκλιπούσα, πολυαγαπημένη μου μητέρα Ελευθερία*

# Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή Κιαμάλ Πεκμεστζή που τα εξαιρετικά μαθήματα του που παρακολούθησα κατά την διάρκεια των σπουδών μου αποτέλεσαν το έναυσμα για να ασχοληθώ με το αντικείμενο της διπλωματικής μου. Παράλληλα θέλω να εκφράσω την βαθιά μου ευγνωμοσύνη για την υποστήριξη του, τις συμβουλές του, το κίνητρο που μου μετέδιδε και τα όσα μου έμαθε κατά τις συναντήσεις μας στα πλαίσια της επίβλεψης της παρούσας διπλωματικής. Επίσης θα ήθελα να ευχαριστήσω τους υποψήφιους διδάκτορες Κώστα Τσουμάνη και Γεώργιο Ζερβάκη για τις πολύτιμες συμβουλές τους και την τεχνική βοήθεια που μου παρείχαν όποτε και αν την χρειάστηκα. Τέλος, ευχαριστώ θερμά την οικογένεια και τους φίλους μου για την ψυχολογική και την υλική υποστήριξη που μου πρόσφεραν.

# Πίνακας περιεχομένων

# 1

# ΕΙΣΑΓΩΓΗ

## 1.1 Ψηφιακή σχεδίαση συστημάτων VLSI

Η ψηφιακή σχεδίαση ολοκληρωμένων κυκλωμάτων είναι ένα πολύ σημαντικό αντικείμενο της επιστήμης του ηλεκτρολόγου μηχανικού και μηχανικού υπολογιστών. Η ραγδαία αύξηση της τεχνολογίας δημιούργησε την ανάγκη για όσο το δυνατόν αποδοτικότερα ολοκληρωμένα κυκλώματα, αλλά ταυτόχρονα έδωσε την δυνατότητα για την σχεδίαση και την υλοποίηση ολοκληρωμένων κυκλωμάτων σε τεράστια κλίμακα ολοκλήρωσης. Σταδιακά τα αναλογικά ολοκληρωμένα κυκλώματα βρίσκουν όλο και λιγότερες εφαρμογές με αποτέλεσμα σήμερα η πλειονότητα των ολοκληρωμένων κυκλωμάτων που παράγονται να είναι ψηφιακά. Συνεπώς τα ψηφιακά ολοκληρωμένα κυκλώματα έχουν αποκτήσει ιδιαίτερο βάρος και επηρεάζουν άμεσα την εξέλιξη πολλών κλάδων της τεχνολογίας όπως της επιστήμη των υπολογιστών, των τηλεπικοινωνιών, της βιοϊατρικής τεχνολογίας κ.α.

Οι προδιαγραφές ενός ψηφιακού κυκλώματος κατά την φάση της σχεδίασης του συνήθως ορίζουν την μέγιστη καθυστέρηση, κατανάλωση και επιφάνεια που θα καταλαμβάνει το κύκλωμα. Οι τρείς τελευταίοι παράγοντες είναι οι κύριοι παράγοντες για την αξιολόγηση της επίδοσης του κυκλώματος. Από την δεκαετία του 70’ και μετά ξεκίνησε η παραγωγή κυκλωμάτων πολύ μεγάλης κλίμακας ολοκλήρωσης (*Very Large Scale Integration* ή VLSI) που σημαίνει εκατομμύρια τρανζίστορ, το καθένα με μήκος κάποιων δεκάδων nm πάνω σε ένα μόνο ολοκληρωμένο κύκλωμα. Η ανάγκη για την σχεδίαση τόσο πολύπλοκων κυκλωμάτων οδήγησε στην ανάπτυξη ειδικών γλωσσών περιγραφής υλικού όπως η VHDL και η Verilog και εργαλείων λογισμικού για την ανάλυση και αξιολόγηση τους πριν από την φάση της παραγωγής, που είναι γνωστά ως EDA (*Electronic Design Automation*). Επίσης, έχει αναπτυχθεί και ακολουθείται από τις εταιρίες που σχεδιάζουν VLSI συστήματα, μία

ροή σχεδίασης που περιλαμβάνει αρκετά στάδια σχεδίασης και επαλήθευσης του κυκλώματος πριν την τελική φάση της παραγωγής του.

## 1.2 Ψηφιακή επεξεργασία σήματος

Η ψηφιακή επεξεργασία σήματος (*Digital Signal Processing* ή DSP) ασχολείται με την αναπαράσταση, την μετατροπή και την επεξεργασία ψηφιακών δεδομένων. Τα ψηφιακά σήματα συνήθως προέρχονται από φυσικά αναλογικά σήματα όπως σήματα ήχου, εικόνας, θερμοκρασίας κτλ. Για την μετατροπή ενός αναλογικού σήματος σε ψηφιακό απαιτείται ένας μετατροπέας (*Analog to Digital Converter* ή ADC) που εκτελεί τα στάδια της δειγματοληψίας για την διακριτοποίηση του συνεχούς σήματος και στην συνέχεια της κβάντισης του με σκοπό την αντιστοίχηση των διακριτών τιμών σε συγκεκριμένες στάθμες.

Η ανάπτυξη της ψηφιακής επεξεργασίας σήματος εισήγαγε αρκετά πλεονεκτήματα σε σχέση με την αναλογική και κατέστησε δυνατή την εκτέλεση πολύπλοκων αλγορίθμων για την επεξεργασία ψηφιακών δεδομένων. Οι αλγόριθμοι αυτοί υλοποιούνται σε επίπεδο υλικού, συνήθως σε ειδικά σχεδιασμένα ολοκληρωμένα κυκλώματα (*Application Specific Integrated Circuit* ή ASIC).

Η πράξη του πολλαπλασιασμού είναι ίσως η σημαντικότερη αλλά και η απαιτητικότερη αριθμητική πράξη στα σύγχρονα συστήματα DSP. Για αυτόν τον λόγο είναι ιδιαίτερα σημαντική η διερεύνηση για την ανάπτυξη νέων αποδοτικών αλγορίθμων πολλαπλασιασμού με σκοπό την γενικότερη βελτίωση της απόδοσης των συστημάτων αυτών.

## 1.3 Αντικείμενο της διπλωματικής

Σκοπός της παρούσας διπλωματικής είναι η εφαρμογή του πολλαπλασιαστικού αλγορίθμου Karatsuba στην σχεδίαση φίλτρων FIR και η αξιολόγηση της απόδοσης τους σε σύγκριση με τις αντίστοιχες συμβατικές υλοποιήσεις.

Τα φίλτρα περιγράφηκαν σε γλώσσα Verilog και υλοποιήθηκαν σε direct, transposed και mixed μορφή. Όσον αφορά την mixed μορφή, εφαρμόστηκε η τεχνική της συνεχούς διοχέτευσης, σε συνδυασμό με μια κατάλληλη ομαδοποίηση των γινομένων του φίλτρου με σκοπό την μεγιστοποίηση της συχνότητας λειτουργίας.

Σχεδιάστηκε ένας δενδρικός πολλαπλασιαστής Wallace με προ-κωδικοποίηση MB και χρησιμοποιήθηκε ως δομική μονάδα σε όλα τα κυκλώματα των φίλτρων. Η έξοδος της συγκεκριμένης μονάδας είναι σε CS αναπαράσταση ώστε να επιταχυνθούν οι υπολογισμοί των φίλτρων. Επίσης, επιχειρήθηκε η μείωση του υλικού που χρησιμοποιείται στους διορθωτικούς όρους της μονάδας πολλαπλασιασμού. Πιο συγκεκριμένα, εφαρμόστηκε μία τεχνική για την παραγωγή ενός τελικού διορθωτικού όρου που συγχωνεύει όλους τους άσσους από την επέκταση προσήμου όλων των μερικών γινομένων. Ο τελικός διορθωτικός όρος προστίθεται στον τελικό αθροιστή των φίλτρων.

Τα κυκλώματα προσομοιώθηκαν με την βοήθεια του εργαλείου λογισμικού *ModelSim* ώστε να επαληθευτεί η ορθή λειτουργία τους. Στην συνέχεια, διενεργήθηκε η σύνθεση των κυκλωμάτων με χρήση του εργαλείου *Design Compiler* της *Synopsys* με βάση μια standard-cell CMOS βιβλιοθήκη της *Artisan* στα 90*nm*. Τα φίλτρα FIR υλοποιήθηκαν για μήκη λέξης 16 και 32 bits (δεδομένα εισόδου και συντελεστές του φίλτρου), για 16 και 32 σημεία (taps). Η ανάλυση χρονισμού των κυκλωμάτων πραγματοποιήθηκε με το εργαλείο *Synopsys PrimeTime* και η μέση κατανάλωση τους υπολογίστηκε από το *Synopsys PrimePower*. Τέλος, με βάση τα αποτελέσματα των πειραμάτων, τα φίλτρα Karatsuba συγκρίθηκαν με τα συμβατικά ως προς την καθυστέρηση, την επιφάνεια και την κατανάλωση.

### 1.3.1 Συνεισφορά

Η συνεισφορά της διπλωματικής μπορεί να συνοψιστεί ως εξής:

1. Σχεδιάστηκε μια παράλληλη αρχιτεκτονική ενός φίλτρου FIR, η οποία βασίστηκε στον πολλαπλασιαστικό αλγόριθμο Karatsuba.
2. Υλοποιήθηκαν τρείς τοπολογίες (direct, transposed και mixed) της Karatsuba καθώς και της συμβατικής αρχιτεκτονικής, οι οποίες μπορούν να παραχθούν παραμετρικά για διαφορετικό αριθμό taps (που είναι δυνάμεις του 2 π.χ. 8, 16, 32, 64 κ.τ.λ.) και για διάφορα μήκη λέξης για τους συντελεστές και τα δεδομένα εισόδου.
3. Αναπτύχθηκε μια μεθοδολογία για την εύρεση ενός τελικού διορθωτικού όρου για το αποτέλεσμα του φίλτρου. Ο όρος αυτός παράγεται παραμετρικά με συγχώνευση όλων των άσσων από την επέκταση προσήμου κάθε επιμέρους γινομένου και ολίσθηση κατά $log_2(tap)$ θέσεις αριστερά.
4. Εφαρμόστηκε η τεχνική της διοχέτευσης στην mixed μορφή των φίλτρων FIR και αξιολογήθηκε η συνεισφορά της στην βελτίωση της απόδοσης τους.

5. Παρουσιάστηκαν τα συγκριτικά αποτελέσματα των φίλτρων Karatsuba σε σχέση με τις αντίστοιχες συμβατικές υλοποιήσεις, ως προς την καθυστέρηση, την επιφάνεια και την κατανάλωση στα 90 nm.

## 1.4 Οργάνωση του κειμένου

Στο κεφάλαιο 2 παρατίθεται η θεωρία που απαιτείται να γνωρίζει ο αναγνώστης για την κατανόηση της παρούσας εργασίας. Πιο συγκεκριμένα γίνεται μια παρουσίαση των αριθμητικών συστημάτων και των κωδικοποιήσεων που χρησιμοποιούνται στη σχεδίαση των κυκλωμάτων, παρουσιάζεται ο αλγόριθμος Karatsuba για τον πολλαπλασιασμό δύο αριθμών, εξηγείται συνοπτικά η τεχνική pipelining και τέλος γίνεται μια εισαγωγή στις βασικές έννοιες και τοπολογίες των ψηφιακών φίλτρων FIR. Στο κεφάλαιο 3 αναλύεται η σχεδίαση και η υλοποίηση των φίλτρων FIR. Το κεφάλαιο 4 περιλαμβάνει μια θεωρητική ανάλυση που αφορά την κρίσιμη καθυστέρηση των φίλτρων FIR, καθώς και τις δομικές μονάδες υλικού που αυτά χρησιμοποιούν. Στο κεφάλαιο 5 παρουσιάζονται τα πειραματικά αποτελέσματα και πραγματοποιούνται οι απαραίτητες συγκρίσεις με σκοπό την αξιολόγηση των κυκλωμάτων που υλοποιήθηκαν. Στο κεφάλαιο 6 συνοψίζονται τα συμπεράσματα που προκύπτουν από την εκπόνηση της παρούσας εργασίας και παρατίθενται τυχόν μελλοντικές επεκτάσεις της.

# 2

# ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Στο παρόν κεφάλαιο γίνεται μια συνοπτική παρουσίαση των βασικών θεωρητικών εννοιών που κρίνονται απαραίτητες για την κατανόηση της παρούσας εργασίας. Αρχικά παρουσιάζεται το δυαδικό αριθμητικό σύστημα και οι κωδικοποιήσεις που χρησιμοποιούνται στα κυκλώματα που σχεδιάστηκαν. Στην συνέχεια, γίνεται μία εισαγωγή στα βασικά είδη αριθμητικών κυκλωμάτων, τους αθροιστές και τους πολλαπλασιαστές. Παρουσιάζεται ο γενικός αλγόριθμος του Karatsuba για τον πολλαπλασιασμό δύο αριθμών, του οποίου η κυκλωματική υλοποίηση αναλύεται στο κεφάλαιο 4. Επίσης, παρουσιάζεται συνοπτικά η τεχνική του pipelining. Τέλος δίνεται η βασική θεωρία των ψηφιακών φίλτρων FIR.

## 2.1 Αριθμητικά συστήματα

### 2.1.1 Δυαδικό σύστημα

Το δυαδικό σύστημα είναι το πιο ευρέως διαδεδομένο αριθμητικό σύστημα και το πιο σημαντικό για την επιστήμη των υπολογιστών. Η ιδέα πίσω από την καθιέρωση του δυαδικού συστήματος στα ψηφιακά κυκλώματα είναι ότι για την αναπαράσταση ενός αριθμού χρησιμοποιεί μόνο δύο ψηφία τα {0,1} γεγονός που το καθιστά το πλέον κατάλληλο για την αντιστοίχηση του σε δύο επίπεδα τάσεων. Το δυαδικό σύστημα έχει ως βάση το 2 όπως το δεκαδικό που έχει αντίστοιχα το 10.

Κάθε αριθμός μπορεί να παρασταθεί στο δυαδικό σύστημα σύμφωνα με την παρακάτω σχέση:

$$\alpha_{(10)} = \sum_{i=0}^{n-1} 2^i \cdot b_i = b_{n-1} \cdot b_{n-2} \dots \cdot b_{0\,(2)}$$

Όπου α η τιμή του αριθμού, n το πλήθος των δυαδικών ψηφίων (bit) του αριθμού και $b_i$ οι τιμές των bit που ανήκουν στο σύνολο {0,1} όπως προαναφέρθηκε.

Για παράδειγμα ο δεκαδικός αριθμός 12 αναπαρίσταται στο δυαδικό σύστημα ως 1100. Το κάθε ψηφίο ενός δυαδικού αριθμού έχει διαφορετικό βάρος, με το αριστερότερο ψηφίο να έχει το μεγαλύτερο (Most Significant Bit ή MSB) και το δεξιότερο το μικρότερο (Least Significant Bit ή LSB). Κάνοντας το ανάπτυγμα του δυαδικού αριθμού 1100 προκύπτει: $0 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 = 12$ όπως αναμενόταν.

Το πλήθος των bit που χρησιμοποιεί ένας δυαδικός αριθμός για την αναπαράσταση του, καθορίζει και τον μέγιστο δεκαδικό αριθμό που μπορεί να αναπαραστήσει. Ο μέγιστος δεκαδικός που μπορεί να αναπαραστήσει ένας δυαδικός αριθμός των *n* bit είναι ο $2^n$-1 διότι με *n* δυαδικά ψηφία είναι δυνατοί $2^n$ διαφορετικοί συνδυασμοί, όμως ο ένας χρησιμοποιείται για την αναπαράσταση του μηδενός.

Ως παράδειγμα δίνεται ο παρακάτω πίνακας που περιλαμβάνει όλους τους πιθανούς συνδυασμούς ενός δυαδικού αριθμού των τριών bit και τους αντίστοιχους δεκαδικούς.

| **Δυαδικός** | **Δεκαδικός** |
|---|---|
| 000 | 0 |
| 001 | 1 |
| 010 | 2 |
| 011 | 3 |
| 100 | 4 |
| 101 | 5 |
| 110 | 6 |
| 111 | 7 |

Πίνακας 2.1: Δυαδική αρίθμηση από το 0 έως το 7

Για το άθροισμα δύο δυαδικών αριθμών απαιτείται η άθροιση κάθε ψηφίου ίδιου βάρους των δύο αριθμών ξεκινώντας από το LSB. Σε περίπτωση όπου σε μία βαθμίδα αθροίζονται δύο μονάδες προκύπτει κρατούμενο το οποίο και μεταφέρεται στην αμέσως πιο σημαντική βαθμίδα. Για παράδειγμα η άθροιση του αριθμού 132 με τον αριθμό 83 δίνει αποτέλεσμα 215:

| | |
|---|---|
| 10000100 | 132 |
| +01010011 | +83 |
| 11010111 | 215 |

### 2.1.2 Αναπαράσταση σε μορφή συμπληρώματος ως προς δύο

Η αναπαράσταση ενός δυαδικού αριθμού όπως παρουσιάστηκε στην παραπάνω υποενότητα επιτρέπει μόνο την αναπαράσταση θετικών αριθμών. Στα ψηφιακά συστήματα όμως είναι απολύτως απαραίτητη και η αναπαράσταση προσημασμένων αριθμών σε δυαδική μορφή. Η ανάγκη αυτή οδήγησε στην καθιέρωση της αναπαράστασης σε μορφή συμπληρώματος ως προς δύο, η οποία δίνεται από την παρακάτω σχέση:

$$\alpha_{(10)} = -b_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} 2^i \cdot b_i = b_{n-1} \cdot b_{n-2} \dots \cdot b_0 \,_{(2)}$$

Όπως φαίνεται στην παραπάνω σχέση το MSB του αριθμού έχει αρνητικό βάρος. Από το γεγονός ότι το MSB έχει και το μεγαλύτερο βάρος, εύκολα προκύπτει ότι όλοι οι αριθμοί με το αριστερότερο bit τους να είναι άσσος είναι αρνητικοί. Με αυτόν τον τρόπο το MSB λαμβάνει την ιδιότητα του bit προσήμου. Ο αντίθετος ενός οποιουδήποτε δυαδικού αριθμού είναι το συμπλήρωμα του ως προς δύο. Για τον υπολογισμό του απαιτείται αρχικά ο υπολογισμός του συμπληρώματος ως προς ένα του αριθμού και στην συνέχεια πρέπει να προστεθεί η μονάδα. Ως παράδειγμα δίνεται η μετατροπή του αριθμού $0101_{(2)} = 5_{(10)}$ σε συμπλήρωμα ως προς δύο:

1. Αρχικά υπολογίζεται το συμπλήρωμα ως προς ένα του αριθμού αντιστρέφοντας όλα τα ψηφία του: 0101 → 1010
2. Στην συνέχεια στο συμπλήρωμα ως προς ένα προστίθεται η μονάδα:
   1010 + 1 = 1011

Ο αριθμός $1011_{(2)} = -1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -5_{(10)}$ που είναι και το ζητούμενο αποτέλεσμα. Ένας αριθμός σε μορφή συμπληρώματος ως προς δύο των n bit μπορεί να αναπαραστήσει όλους τους ακεραίους στο διάστημα $[-2^{n-2}, 2^{n-2}-1]$.

Ο αλγόριθμος για την πρόσθεση δύο αριθμών που βρίσκονται σε μορφή συμπληρώματος ως προς δύο είναι πανομοιότυπος με εκείνον για την πρόσθεση δύο θετικών δεκαδικών αριθμών. Η άθροιση γίνεται κατά τα γνωστά ψηφίο-ψηφίο ξεκινώντας από το LSB και θεωρώντας ότι το αρχικό κρατούμενο είναι μηδέν. Στην συνέχεια γίνεται σύγκριση του κρατουμένου εισόδου και εξόδου της βαθμίδας που εξάγει το πρόσημο και σε περίπτωση όπου αυτά δεν συμφωνούν γνωρίζουμε ότι έχει συμβεί υπερχείλιση. Στην περίπτωση όπου τα κρατούμενα συμφωνούν τότε συμπεραίνουμε ότι το αποτέλεσμα της πρόσθεσης είναι σωστό. Στην συνέχεια παρουσιάζονται όλες οι δυνατές περιπτώσεις κατά την πρόσθεση δύο αριθμών σε μορφή συμπληρώματος ως προς δύο.

**Πρόσθεση δύο θετικών αριθμών**

Σε αυτήν την περίπτωση το bit προσήμου δηλαδή το MSB και των δύο αριθμών είναι μηδέν. Επομένως αναμένουμε το αποτέλεσμα να έχει και αυτό το MSB του μηδέν διότι το άθροισμα δύο θετικών αριθμών είναι επίσης θετικός αριθμός. Σε περίπτωση που πράγματι αυτό προκύπτει μηδέν είμαστε βέβαιοι ότι το αποτέλεσμα είναι σωστό και πως δεν προέκυψε υπερχείλιση:

| | |
|---|---|
| 0101 | 5 |
| +0010 | +2 |
| 0111 | 7 |

Σε περίπτωση όπου το MSB του αποτελέσματος προκύψει ένα τότε έχει συμβεί υπερχείλιση και το αποτέλεσμα δεν πρέπει να θεωρηθεί σωστό:

| | |
|---|---|
| 01011 | 11 |
| +00111 | +7 |
| 10010 | -14 |

Λύση στο πρόβλημα αυτό δίνει η λεγόμενη επέκταση προσήμου που στην ουσία θεωρεί ως το MSB του αποτελέσματος το κρατούμενο εξόδου της τελευταίας βαθμίδας.

**Πρόσθεση δύο αρνητικών αριθμών**

Ομοίως αναμένουμε το MSB του αποτελέσματος να είναι μονάδα διότι αθροίζονται δύο αρνητικοί αριθμοί. Για παράδειγμα, μια πρόσθεση χωρίς υπερχείλιση θα μπορούσε να είναι η παρακάτω:

| | |
|---|---|
| 11011 | -5 |
| +10111 | -9 |
| 10010 | -14 |

Στην περίπτωση όπου το κρατούμενο εισόδου της τελευταίας βαθμίδας είναι μηδέν, τότε ο τελευταίος πλήρης αθροιστής θα έχει άθροισμα μηδέν και κρατούμενο εξόδου μονάδα. Ένα τέτοιο αποτέλεσμα δεν είναι σωστό όπως για παράδειγμα στην περίπτωση που ακολουθεί:

| | |
|---|---|
| 10111 | -9 |
| +10100 | -12 |
| 01011 | 11 |

Λύνουμε το πρόβλημα εφαρμόζοντας όπως και προηγουμένως την τεχνική της επέκτασης προσήμου, θεωρώντας δηλαδή ότι το MSB είναι το κρατούμενο εξόδου της τελευταίας βαθμίδας. Έτσι προκύπτει το σωστό αποτέλεσμα που είναι $101011_{(2)} = -21_{(10)}$.

**Πρόσθεση ενός θετικού και ενός αρνητικού αριθμού**

Σε αυτήν την περίπτωση γνωρίζουμε ότι δεν θα προκύψει υπερχείλιση διότι αθροίζονται δυο ετερόσημοι αριθμοί γεγονός που εξασφαλίζει ότι το αποτέλεσμα θα είναι σίγουρα μικρότερο από τον θετικό αριθμό και μεγαλύτερο από τον αρνητικό. Στο παρακάτω παράδειγμα το αποτέλεσμα είναι θετικό:

| | |
|---|---|
| 01011 | 11 |
| +10111 | -9 |
| 00010 | 2 |

Ενώ στο επόμενο παράδειγμα αρνητικό:

| | |
|---|---|
| 00011 | 3 |
| +10111 | -9 |
| 11010 | 6 |

Όπως παρατηρούμε στα παραπάνω παραδείγματα δεν προκύπτει υπερχείλιση οπότε και δεν απαιτείται η εφαρμογή της επέκτασης προσήμου. Αυτό συμβαίνει διότι όπως έχει ήδη αναφερθεί τα κρατούμενα εισόδου και εξόδου της τελευταίας βαθμίδας συμφωνούν, γεγονός που εξασφαλίζει την ορθότητα του αποτελέσματος.

### 2.1.3 Αναπαράσταση σε μορφή Σωσίματος-Κρατουμένου (Carry-Save ή CS)

Η αναπαράσταση σε μορφή Σωσίματος-Κρατουμένου υπάγεται στην γενικότερη κατηγορία των αριθμητικών συστημάτων με περίσσια (Redundant Arithmetic Systems). Αυτού του είδους τα αριθμητικά συστήματα δεν έχουν αμφιμονοσήμαντη αντιστοιχία με το δεκαδικό σύστημα. Συγκεκριμένα υπάρχουν περισσότερες από μία διαφορετικές μορφές που αντιστοιχούν στην ίδια δεκαδική τιμή.

Η σχέση που εκφράζει την αναπαράσταση ενός αριθμού CS είναι $x^* = x^s + x^c$. Δηλαδή η δεκαδική του τιμή προκύπτει ως άθροισμα δύο αριθμών. Το πλεονέκτημα της μορφής CS είναι ότι επιτρέπει την ταχύτατη εκτέλεση των πράξεων της πρόσθεσης και της αφαίρεσης αφού αυτές υλοποιούνται χωρίς την διάδοση κρατουμένου. Ωστόσο η αναπαράσταση ενός αριθμού σε CS μορφή απαιτεί τον διπλάσιο αριθμό bit από ότι η αναπαράσταση της ίδιας πληροφορίας σε μορφή συμπληρώματος ως προς δύο.

## 2.2 Κωδικοποιήσεις Booth και Modified Booth

### 2.2.1 Κωδικοποίηση Booth

Ο αλγόριθμός του Booth παρουσιάστηκε πριν από τέσσερις δεκαετίες και σχεδιάστηκε με σκοπό τον αποδοτικό πολλαπλασιασμό δύο προσημασμένων αριθμών. Αυτό είναι και το κύριο πλεονέκτημα του, ότι δηλαδή είναι γενικός και ανεξάρτητος του προσήμου των αριθμών. Από τότε εφαρμόζεται κατά κόρον με μικρές τροποποιήσεις σε πλειάδα ψηφιακών αριθμητικών κυκλωμάτων. Ας εξετάσουμε όμως τα βασικά σημεία του αλγορίθμου. Όπως είδαμε στην παράγραφο 2.1.2 ένας αριθμός έστω X παριστάνεται σε μορφή συμπληρώματος ως προς δύο με βάση την σχέση:

$$\alpha_{(10)} = -b_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} 2^i \cdot b_i = b_{n-1} \cdot b_{n-2} \dots \cdot b_0 \ (2)$$

Ο αριθμός X μπορεί να εκφραστεί ισοδύναμα και ως $X = 2X - X$ όπως παρουσιάζεται σχηματικά παρακάτω:

| | | | | | | |
|---|---|---|---|---|---|---|
| 2X = | $-X_{n-1}$ | $X_{n-2}$ | $X_{n-3}$ | … | $X_0$ | 0 |
| -X = | 0 | $-X_{n-1}$ | $X_{n-2}$ | … | $X_1$ | $X_0$ |
| | | $Z_{n-1}$ | $Z_{n-2}$ | … | $Z_1$ | $Z_0$ |

Όπου: $Z_0 = 0 - X_0$,

$Z_1 = X_0 - X_1$,

.

.

.

$Z_{n-2} = X_{n-3} - X_{n-2}$,

Και τέλος $Z_{n-1} = -2\,X_{n-1} + X_{n-2} + X_{n-1} = X_{n-2} - X_{n-1}$

Έτσι προκύπτει ότι σε κάθε περίπτωση ισχύει η σχέση $Z_i = X_{i-1} - X_i$. Επομένως, ο αριθμός $X$ δίνεται από τη σχέση $X = \sum_{i=0}^{n-1} Z_i 2^i$ . Σύμφωνα με τον αλγόριθμο Booth το γινόμενο $P=X \cdot Y$ όπου $X= x_{n-1}x_{n-2}\ldots x_1x_0$ και $Y=y_{n-1}y_{n-2}\ldots y_1y_0$ δίνεται από την παρακάτω σχέση:

$$P = X \cdot Y = \sum_{i=0}^{n-1} (y_{i-1} - y_i) \cdot X \cdot 2^i$$

Από την παραπάνω σχέση προκύπτει ότι σε κάθε βήμα i του αλγορίθμου, ο αριθμός $X$ πολλαπλασιάζεται με ένα από τα στοιχεία του συνόλου {-1,0,1} σύμφωνα με το αποτέλεσμα της αφαίρεσης των δύο διαδοχικών ψηφίων του πολλαπλασιαστή $Y$. Εδώ χρειάζεται να αναφέρουμε ότι για $i$=0 ισχύει $z_0=y_{-1}-y_0$ όπου θεωρούμε $y_{-1}$=0.

Με την κωδικοποίηση Booth δίνεται η δυνατότητα ελέγχου των σειρών από μονάδες μέσα σε κάθε δυαδικό αριθμό. Στον παρακάτω πίνακα φαίνεται ο έλεγχος των κωδικοποιημένων ψηφίων:

| $Y_i$ | $Y_{i+1}$ | **Κωδικοποιημένα ψηφία ($Y_{i+1}$ - $Y_i$)** | **Σχόλια** |
|---|---|---|---|
| 0 | 0 | 0 | Δεν υπάρχει σειρά από 1 |
| 0 | 1 | 1 | Τέλος σειράς από 1 |
| 1 | 0 | -1 | Αρχή σειράς από 1 |
| 1 | 1 | 0 | Μέση σειράς από 1 |

Πίνακας 2.2: Ερμηνεία κωδικοποιημένων ψηφίων Booth

Όπως έχουμε ήδη αναφέρει τα ψηφία ενός αριθμού σε κωδικοποίηση Booth ανήκουν στο σύνολο {-1,0,1}.

Ανακεφαλαιώνοντας την λειτουργία του αλγορίθμου μπορούμε να τη συνοψίσουμε ως εξής:

- Αν $Y_iY_{i-1}$=00 ή $Y_iY_{i-1}$=11 τότε ολίσθησε το τρέχον άθροισμα των μερικών γινομένων κατά ένα bit προς τα αριστερά.
- Αν $Y_iY_{i-1}$=01 τότε πρόσθεσε τον πολλαπλασιαστή $X$ στο τρέχον άθροισμα των ενδιάμεσων γινομένων και μετά ολίσθησε το αποτέλεσμα κατά μία θέση προς τα αριστερά.
- $Y_iY_{i-1}$=10 τότε αφαίρεσε τον πολλαπλασιαστή $X$ από το τρέχον άθροισμα των ενδιάμεσων γινομένων και μετά ολίσθησε το αποτέλεσμα κατά μια θέση προς τα αριστερά.

Η διαδικασία κωδικοποίησης έχει φορά από τα δεξιά προς τα αριστερά και κάθε φορά εξετάζονται τα ψηφία ανά δύο και κωδικοποιούνται σύμφωνα με τον πίνακα. Στην πρώτη επανάληψη εξετάζεται το ζεύγος $Y_0Y_{-1}$ όπου θεωρούμε $Y_{-1}$=0. Το επόμενο ζεύγος που εξετάζεται είναι το $Y_1Y_0$, μετά το $Y_2Y_1$ και ούτω καθεξής. Παρατηρούμε ότι τα ζευγάρια bit που εξετάζονται κάθε φορά έχουν επικάλυψη ενός ψηφίου. Έτσι σε κάθε έλεγχο μόνο ένα bit του πολλαπλασιαστή δεν εξετάζεται.

Έστω τώρα ο πολλαπλασιασμός των αριθμών $X=(10110)_2=(-10)_{10}$ και $Y=(0101)_2=(5)_{10}$. Αρχικά το $Y$ κωδικοποιείται σύμφωνα με τον πίνακα και προκύπτει $Y = 01\bar{1}1\bar{1}$ (με το σύμβολο $\bar{1}$ παριστάνεται το ψηφίο -1). Στην συνέχεια ο πολλαπλασιασμός πραγματοποιείται όπως παρουσιάζεται στον παρακάτω πίνακα:

| | | |
|---|---|---|
| $X$= 1 0 1 1 0<br>$Y$= 0 0 1 0 1 | | Μερικό γινόμενο = 00000 |
| 0 0 1 0 1 0 | -1 | Πρόσθεσε τον $-X$. Πρώτο άθροισμα μερικού γινομένου |
| 1 0 1 1 0 | +1 | Πρόσθεσε τον $X$. |
| 1 1 1 0 1 1 0 | Δεύτερο άθροισμα μερικών γινομένων | |
| 0 1 0 1 0 | -1 | Πρόσθεσε τον $-X$. |
| 0 0 0 1 1 1 1 0 | Τρίτο άθροισμα μερικών γινομένων | |
| 1 0 1 1 0 | +1 | Πρόσθεσε τον $X$. |
| 1 1 1 0 0 1 1 1 0 | Τέταρτο άθροισμα μερικών γινομένων | |
| 0 0 0 0 0 | 0 | Πρόσθεσε 0 |
| 1 1 1 0 0 1 1 1 0 | Τελικό Αποτέλεσμα -50 | |

Πίνακας 2.3: Πολλαπλασιασμός δύο αριθμών σύμφωνα με τον αλγόριθμο Booth

Τα αριστερότερα ψηφία που φαίνονται υπογραμμισμένα στον πίνακα, είναι τα ψηφία από την επέκταση προσήμου. Η επέκταση προσήμου είναι απαραίτητη για να γίνει σωστά η επόμενη πρόσθεση με το μερικό γινόμενο που ακολουθεί, το οποίο είναι ολισθημένο κατά μία θέση προς τα αριστερά.

Η κωδικοποίηση Booth παρουσιάζει αρκετά πλεονεκτήματα για την υλοποίηση της πράξης του πολλαπλασιασμού. Όπως έχει ήδη αναφερθεί, η κωδικοποίηση είναι ανεξάρτητη του προσήμου του αριθμού, δηλαδή είναι ίδια για θετικούς και αρνητικούς αριθμούς. Επίσης κάθε ψηφίο του κωδικοποιημένου αριθμού είναι ανεξάρτητο από τα προηγούμενα μερικά γινόμενα διότι εξαρτάται μόνο από τα bit $Y_iY_{i-1}$ με αποτέλεσμα να μπορεί να παραχθεί αμέσως. Όλα τα μερικά γινόμενα που προκύπτουν μπορούν να αθροιστούν με την χρησιμοποίηση ενός δέντρου με CSA αθροιστές για την παραγωγή του τελικού γινομένου. Το σημαντικότερο όμως πλεονέκτημα της κωδικοποίησης Booth είναι η μείωση του πλήθους των μερικών γινομένων που προκύπτουν σε σύγκριση με την συμβατική δυαδική αναπαράσταση. Πιο συγκεκριμένα, πραγματοποιείται μια πρόσθεση για κάθε ψηφίο που είναι μονάδα, ενώ όταν το ψηφίο είναι μηδενικό απαιτείται μόνο μια ολίσθηση για ένα

πολλαπλασιαστή σε δυαδική αναπαράσταση. Όμως, ένας αριθμός σε κωδικοποίηση Booth περιέχει μονάδα μόνο όταν αρχίζει ή τελειώνει μια σειρά από μονάδες, όπως φαίνεται και στον πίνακα. Επομένως με την κωδικοποίηση ενός αριθμού κατά Booth μπορούμε να επιτύχουμε την μείωση των μη μηδενικών ψηφίων με αποτέλεσμα την μείωση των προσθέσεων που απαιτούνται για τον υπολογισμό του τελικού γινομένου. Παρόλα αυτά υπάρχουν περιπτώσεις όπου με την κωδικοποίηση Booth μπορεί ο πολλαπλασιασμός να γίνει λιγότερο αποδοτικός. Και οι δύο περιπτώσεις καταδεικνύονται στον παρακάτω πίνακα.

| **Binary** | **Booth** | **Μείωση μη μηδενικών ψηφίων** |
|---|---|---|
| 110111101 | $0\bar{1}1000\bar{1}1\bar{1}$ | 2 |
| 00110101 | $010\bar{1}1\bar{1}1\bar{1}$ | -2 |

Πίνακας 2.4: Παραδείγματα μείωσης μη μηδενικών ψηφίων της κωδικοποίησης Booth

Από τον πίνακα παρατηρούμε ότι στην πρώτη περίπτωση τα εφτά μη μηδενικά ψηφία στην συμβατική δυαδική αναπαράσταση γίνονται πέντε στην κωδικοποίηση Booth. Αντιθέτως στην δεύτερη περίπτωση από τέσσερα γίνονται έξι, δηλαδή η κωδικοποίηση Booth σε αυτήν την περίπτωση λειτουργεί επιβαρυντικά.

### 2.2.2 Κωδικοποίηση Modified Booth

Η κωδικοποίηση Modified Booth είναι μια παραλλαγή ενός κωδικοποίησης Booth που παρουσιάστηκε στην προηγούμενη ενότητα. Ουσιαστικά με την Modified Booth κωδικοποίηση επεκτείνεται το σύνολο των ψηφίων κωδικοποίησης από το {-1,0,1} ενός είδαμε στην προηγούμενη ενότητα σε {-2,-1,0,+1,+2}. Πλέον τα ψηφία δεν κωδικοποιούνται ανά δύο αλλά ανά τρία. Η κωδικοποίηση Modified Booth προκύπτει αλγεβρικά από την απλή κωδικοποίηση Booth. Πιο συγκεκριμένα αναλύοντας την σχέση $Y = \sum_{i=0}^{n-1} z_i 2^i$ που δίνει την κωδικοποίηση Booth ενός δυαδικού αριθμού $Y$ προκύπτει:

$$Y = \sum_{i=0}^{n-1} z_i 2^i = \sum_{j=0}^{n/2-1} z_{2j} 2^{2j} + z_{2j+1} 2^{2j+1} = \sum_{j=0}^{n/2-1} (z_{2j} + 2z_{2j+1}) 2^{2j}$$

$$= \sum_{j=0}^{n/2-1} w_j 4^j$$

| $y_{i+1}$ | $y_i$ | $y_{i-1}$ | **Κωδικοποίηση Booth** | | **Κωδικοποίηση Modified Booth** |
|---|---|---|---|---|---|
| | | | $z_{i+1}$ | $z_i$ | $w_i = 2z_{i+1} + z_i = y_i - 2y_{i+1} + y_{i-1}$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | +1 | +1 |
| 0 | 1 | 0 | +1 | -1 | +1 |
| 0 | 1 | 1 | +1 | 0 | +2 |
| 1 | 0 | 0 | -1 | 0 | -2 |
| 1 | 0 | 1 | -1 | +1 | -1 |
| 1 | 1 | 0 | 0 | -1 | -1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

Πίνακας 2.5: Κωδικοποιημένα ψηφία Booth και Modified Booth

Κάθε κωδικοποιημένο ψηφίο καθορίζει την επεξεργασία που πρόκειται να υποστεί ο πολλαπλασιαστέος όπως ακριβώς και στον απλό αλγόριθμό με κωδικοποίηση Booth.

| **Κωδικοποιημένα ψηφία** | **Λειτουργία** |
|---|---|
| 0 | Πρόσθεσε το 0 στο μερικό γινόμενο |
| +1 | Πρόσθεσε το ($X$) στο μερικό γινόμενο |
| +2 | Πρόσθεσε το ($2X$) στο μερικό γινόμενο |
| -2 | Αφαίρεσε το ($2X$) από το μερικό γινόμενο |
| -2 | Αφαίρεσε το ($X$) από το μερικό γινόμενο |

Πίνακας 2.6: Επεξήγηση της λειτουργίας των κωδικοποιημένων ψηφίων κατά Modified Booth

Για την καλύτερη κατανόηση ολόκληρης της διαδικασίας του πολλαπλασιασμού με κωδικοποίηση Modified Booth δίνεται το επόμενο παράδειγμα.

Έστω $X=(10110101)_2=(-75)_{10}$ και $Y=(01110010)_2=(114)_{10}$. Και οι δύο αριθμοί είναι σε μορφή συμπληρώματος ως προς δύο. Τα ψηφία του $Y$ κωδικοποιούνται ανά τρία, με επικάλυψη ενός ψηφίου σε κάθε περίπτωση και θέτοντας $Y_{-1}=0$. Ο αριθμός $Y$ γίνεται τελικά $2\bar{1}1\bar{2}$ σε κωδικοποίηση Modified Booth. Ο κωδικοποιημένος αριθμός $Y$ υποδηλώνει ότι θα χρειαστούν οι αριθμοί $2X$=101101010, $\bar{X}$=01001011 και $2\bar{X}$=010010110. Ο πολλαπλασιασμός του $X$ με τον κωδικοποιημένο $Y$ συνοψίζεται στον πίνακα 2.7. Τα αριστερότερα bit που εμφανίζονται υπογραμμισμένα αποτελούν την επέκταση προσήμου.

Η Modified Booth κωδικοποίηση έχει τα ίδια πλεονεκτήματα με την απλή κωδικοποίηση Booth αλλά επιπλέον εξασφαλίζει την μείωση των μη μηδενικών ψηφίων του πολλαπλασιαστή και ως επακόλουθο την μείωση του πλήθους των μερικών γινομένων. Η μείωση του πλήθους των μερικών γινομένων στην πράξη μεταφράζεται σε μειωμένη καθυστέρηση και επιφάνεια. Βέβαια η ανάγκη για τον υπολογισμό των $+2A$, $-2A$ και $-A$ αυξάνει την πολυπλοκότητα του κυκλώματος.

| | | |
|---|---|---|
| $X=$ 1 0 1 1 0 1 0 1<br>$Y=$ 0 1 1 1 0 0 1 0 | | Μερικό γινόμενο = 00000 |
| $\underline{0}$ 0 1 0 0 1 0 1 1 0 | -2 | Πρόσθεσε τον $-2X$. Πρώτο άθροισμα μερικού γινομένου |
| 1 0 1 1 0 1 0 1 | +1 | Πρόσθεσε τον $X$. |
| $\underline{1}$ $\underline{1}$ 1 1 0 1 1 0 1 0 1 0 | Δεύτερο άθροισμα μερικών γινομένων | |
| 0 1 0 0 1 0 1 1 | -1 | Πρόσθεσε τον $-X$. |
| $\underline{0}$ $\underline{0}$ $\underline{0}$ 0 1 0 0 0 0 0 1 1 0 1 0 | Τρίτο άθροισμα μερικών γινομένων | |
| 1 0 1 1 0 1 1 1 | +2 | Πρόσθεσε τον $2X$. |
| 1 0 1 1 1 1 0 1 0 0 1 1 0 1 0 | Τελικό Αποτέλεσμα 8550 | |

Πίνακας 2.7: Παράδειγμα πολλαπλασιασμού δύο αριθμών κατά Modified Booth

## 2.3 Βασικοί αθροιστές, αφαιρέτες και πολλαπλασιαστές

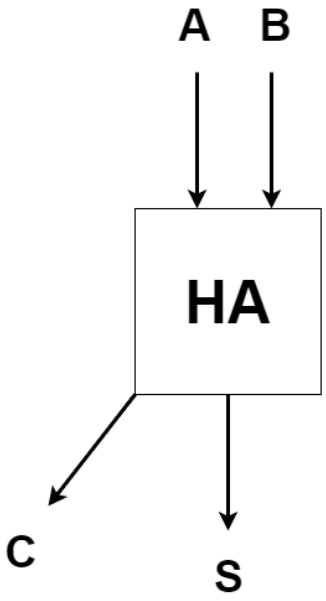
Η πρόσθεση και ο πολλαπλασιασμός είναι οι σημαντικότερες πράξεις κάθε αριθμητικού κυκλώματος. Για αυτόν τον λόγο, έχουν αναπτυχθεί διάφορα είδη κυκλωμάτων αθροιστών, αφαιρετών και πολλαπλασιαστών για την ικανοποίηση των διαφορετικών απαιτήσεων κάθε αριθμητικού κυκλώματος ανάλογα με την λειτουργία του. Στην παρούσα παράγραφο παρουσιάζονται αρχικά τα βασικότερα είδη αθροιστών που είναι ο αθροιστής διάδοσης κρατουμένου (CPA), ο αθροιστής σωσίματος κρατουμένου (CSA) και ο αθροιστής πρόβλεψης κρατουμένου (CLA). Τα αντίστοιχα κυκλώματα των αφαιρετών προκύπτουν εύκολα με κατάλληλη μετατροπή του πλήρους αθροιστή σε πλήρη αφαιρέτη. Στην συνέχεια παρουσιάζονται τα βασικότερα είδη πολλαπλασιαστών όπως ο παράλληλος πολλαπλασιαστής διάδοσης κρατουμένου, ο παράλληλος πολλαπλασιαστής σωσίματος κρατουμένου, ο σειριακός-παράλληλος πολλαπλασιαστής και ο δενδρικός πολλαπλασιαστής.

### 2.3.1 Δομικά στοιχεία

Τα δομικά στοιχεία που συνθέτουν κάθε κύκλωμα αθροιστή, και πολλαπλασιαστή είναι ο ημιαθροιστής (HA), ο πλήρης αθροιστής (FA) και η μονάδα πρόβλεψης κρατουμένου. Στην συνέχεια γίνεται μια σύντομη παρουσίαση των συγκεκριμένων μονάδων.

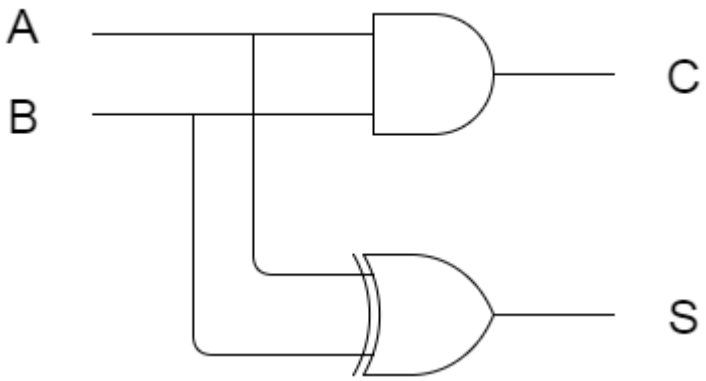
#### 2.3.1.1 Ημιαθροιστής (HA)

Ο ημιαθροιστής είναι το απλούστερο δομικό κύτταρο άθροισης. Έχει ως είσοδο δύο bit, τα οποία προσθέτει, και ως έξοδο ένα bit αθροίσματος και ένα bit κρατουμένου. Ο HA δεν υποστηρίζει την ύπαρξη κρατουμένου εισόδου.




Σχήμα 2.1: Ημιαθροιστής (HA)

Ουσιαστικά ο HA επιτελεί τις σχέσεις $S=A \oplus B$ και $C=A \cdot B$, όπου $S$ είναι το άθροισμα και έχει το ίδιο βάρος με τα bit εισόδου $A$, $B$ ενώ το $C$ είναι το κρατούμενο εξόδου και έχει το αμέσως μεγαλύτερο βάρος.



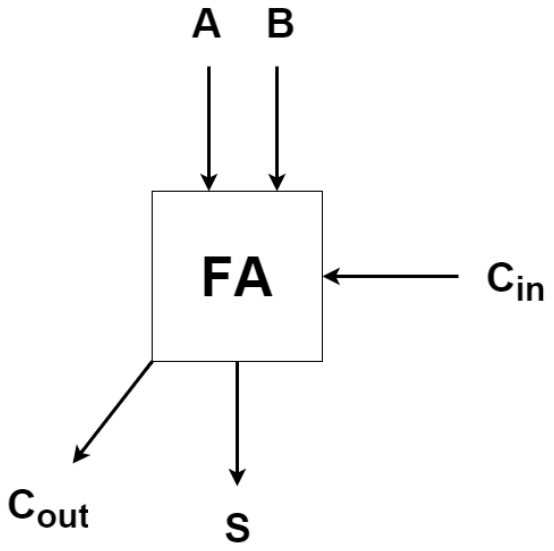

Σχήμα 2.2: Κυκλωματική υλοποίηση ημιαθροιστή

| *A* | *B* | *S* | *C* |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Πίνακας 2.8: Πίνακας αληθείας ημιαθροιστή

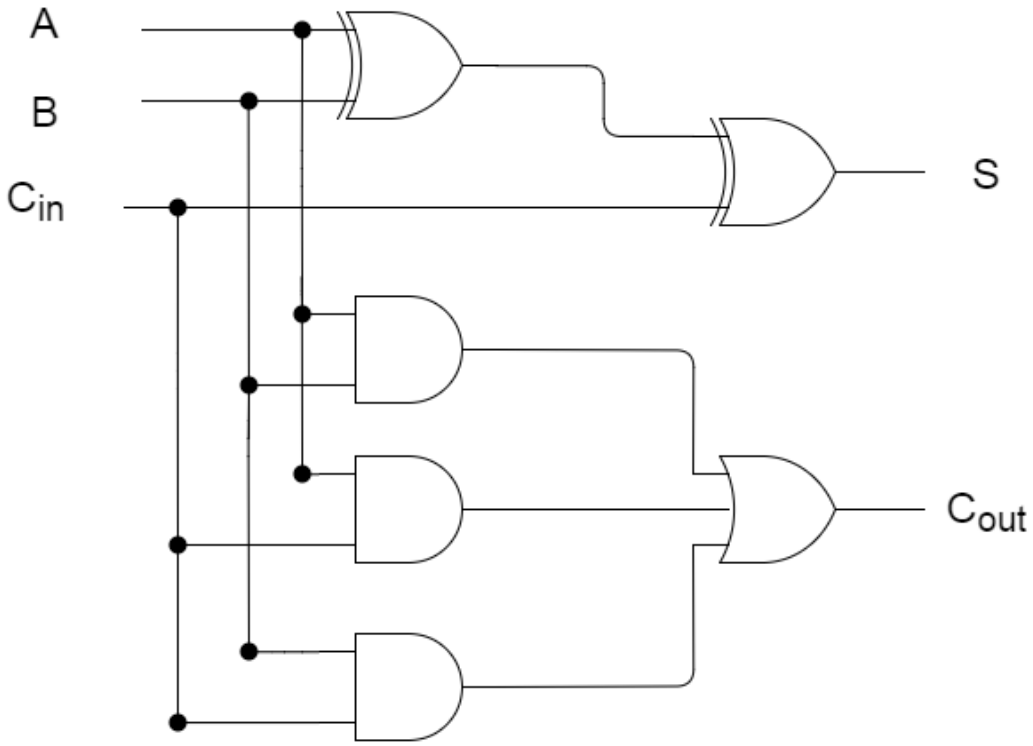
2.3.1.2 Πλήρης αθροιστής (FA)

Ο πλήρης αθροιστής είναι το βασικό δομικό κύτταρο κάθε αριθμητικού κυκλώματος. Όπως και ο HA, ο FA προσθέτει δυαδικά ψηφία ίδιας αξίας και παράγει ένα ψηφίο ίδιας αξίας *S* και ένα κρατούμενο εξόδου $C_{out}$ της αμέσως μεγαλύτερης αξίας. Σε αντίθεση με τον HA, ο FA έχει και κρατούμενο εισόδου $C_{in}$ που συνήθως προέρχεται από την αμέσως χαμηλότερης αξίας βαθμίδα σε έναν αθροιστή πολλών bit. Σύμφωνα με τα παραπάνω, ο FA δέχεται δύο σήματα από τους προστιθέμενους δυαδικούς αριθμούς και ένα κρατούμενο από τον προηγούμενο πλήρη αθροιστή, δηλαδή σύνολο τρείς εισόδους.




Σχήμα 2.3: Πλήρης αθροιστής (FA)

Ο πλήρης αθροιστής υλοποιεί τις λογικές πράξεις, $S=A \oplus B \oplus C_{in}$ και $C_{out}=(A \cdot B)+(A \cdot C_{in})+(B \cdot C_{in})$.



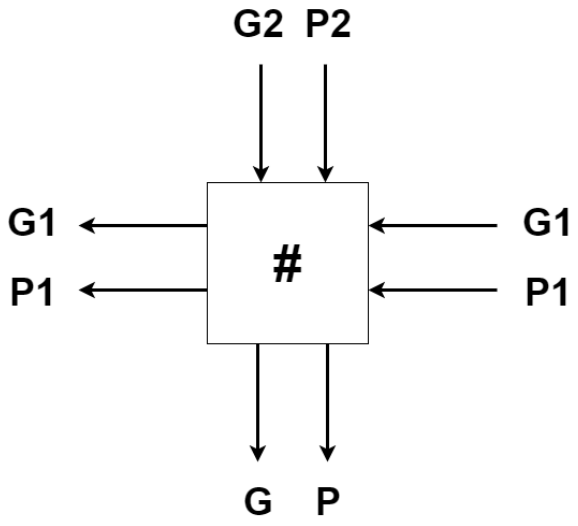

Σχήμα 2.4: Κυκλωματική υλοποίηση πλήρη αθροιστή

| ***A*** | ***B*** | $C_{in}$ | ***S*** | $C_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Πίνακας 2.9: Πίνακας αληθείας πλήρη αθροιστή

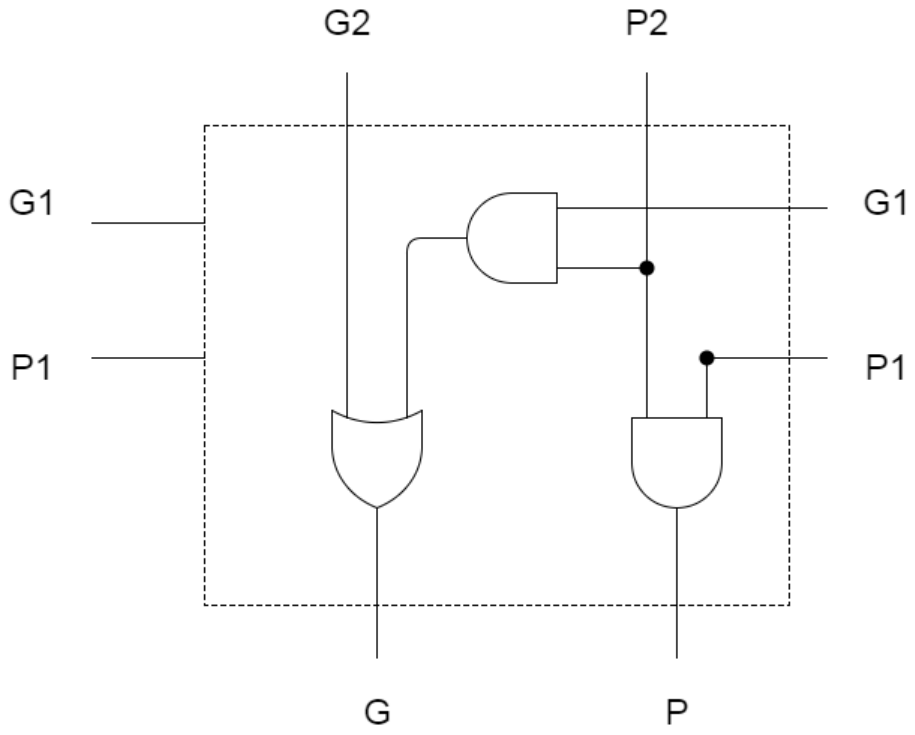
2.3.1.3 Μονάδα πρόβλεψης κρατουμένου

Η μονάδα πρόβλεψης κρατουμένου είναι το βασικό κύτταρο της γεννήτριας κρατουμένου ενός αθροιστή πρόβλεψης κρατουμένου. Έχει ως εισόδους τέσσερα σήματα, τα *G*1, *P*1, *G*2, *P*2 τα οποία είναι σήματα γέννησης (*G*, generation) και διάδοσης (*P*, propagation) κρατουμένου δυο διαφορετικής αξίας βαθμίδων της γεννήτριας κρατουμένου. Η μονάδα πρόβλεψης κρατουμένου παράγει ως έξοδο τα σήματα γεννήσεως και διάδοσης κρατουμένου της μεγαλύτερης από τις δύο βαθμίδες (*G,P*).




Σχήμα 2.5: Μονάδα πρόβλεψης κρατουμένου

Οι είσοδοι και οι έξοδοι της μονάδας πρόβλεψης κρατουμένου συνδέονται με τις επόμενες σχέσεις: $G=G2+(G1\cdot P2)$ και $P=P1\cdot P2$

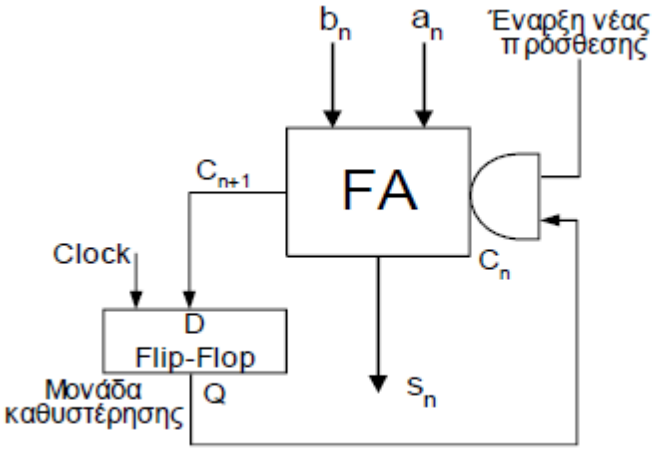



Σχήμα 2.6: Κυκλωματική υλοποίηση μονάδας πρόβλεψης κρατουμένου

### 2.3.2 Αθροιστές

2.3.2.1 Σειριακός αθροιστής

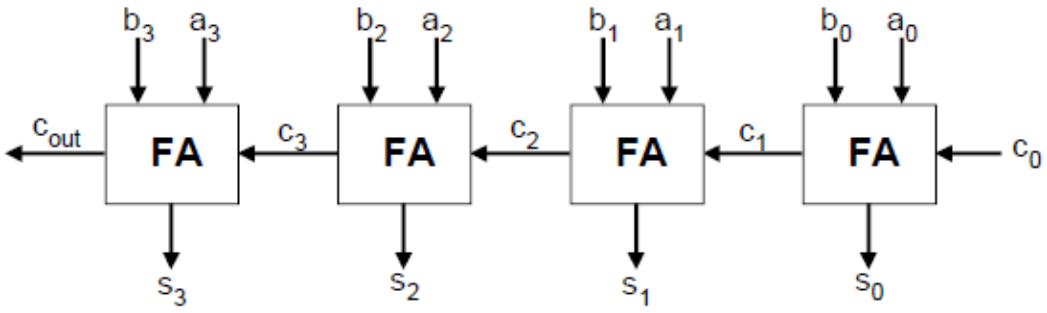
Σκοπός του σειριακού αθροιστή είναι η πρόσθεση δυο αριθμών των *n*-bit οι οποίοι εισάγονται σειριακά, δηλαδή τα bits των αριθμών εισάγονται το ένα μετά το άλλο. Το αποτέλεσμα της πρόσθεσης προκύπτει επίσης στην ίδια μορφή. Όπως φαίνεται και στο σχήμα ο σειριακός αθροιστής αποτελείται από έναν πλήρη αθροιστή και έναν μανδαλωτή. Σε κάθε κύκλο ρολογιού παράγεται ένα άθροισμα και ένα κρατούμενο. Το κρατούμενο αυτό πρέπει να προστεθεί με τα bits των αριθμών που εισάγονται στον επόμενο κύκλο ($a_{n+1}$, $b_{n+1}$) για αυτό και αποθηκεύεται στον μανδαλωτή. Για την πρόσθεση δύο αριθμών των *n*-bit απαιτούνται $n+1$ κύκλοι ρολογιού, όπου στον τελευταίο κύκλο εξάγεται το κρατούμενο $c_{n+1}$. Προφανώς στον πρώτο κύκλο πρέπει να εισαχθεί μηδενικό κρατούμενο εισόδου.




Σχήμα 2.7: Σειριακός αθροιστής

### 2.3.2.2 Αθροιστής διάδοσης κρατουμένου (CPA)

Ένας από τους πιο συνηθισμένους παράλληλους αθροιστές είναι ο αθροιστής διάδοσης κρατουμένου, ο οποίος αποτελείται από ένα δίκτυο πλήρων αθροιστών όπως φαίνεται στο σχήμα:




Σχήμα 2.8: Αθροιστής διάδοσης κρατουμένου (CPA)

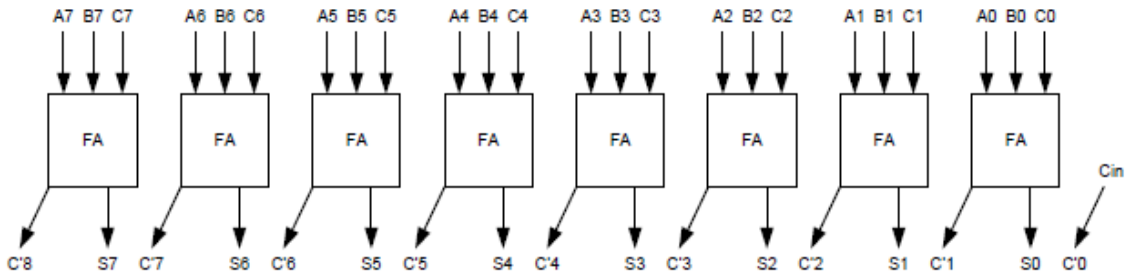
Τα $a_k$, $b_k$ για $\kappa$=0,1,2,3 είναι τα bit των αριθμών $A$ και $B$ που προστίθενται. Αντίστοιχα τα $s_k$,$c_k$ για $\kappa$=0,1,2,3 είναι τα bit αθροίσματος και κρατουμένου που προκύπτουν σε κάθε βαθμίδα. Όπως φαίνεται στο σχήμα 2.8, τα bit εισόδου αθροίζονται και το κρατούμενο που προκύπτει διαδίδεται για να αθροιστεί με τα bit της επόμενης βαθμίδας που έχουν την ίδια αξία. Για το κρατούμενο εισόδου της πρώτης βαθμίδας ισχύει $c_0$=0. Το $c_{out}$ είναι το κρατούμενο εξόδου της τελευταίας βαθμίδας και αποτελεί το MSB. Το μειονέκτημα του

CPA είναι ότι για να προκύψει το τελικό αποτέλεσμα απαιτείται προηγουμένως η διάδοση του κρατουμένου μέσα από όλες τις βαθμίδες του αθροιστή.

Ένας άλλος τρόπος να δει κανείς τον CPA είναι σαν το ξεδίπλωμα στον χρόνο του σειριακού αθροιστή που παρουσιάστηκε στην προηγούμενη παράγραφο. Πράγματι, αναλύοντας την λειτουργία του σειριακού αθροιστή σε κάθε κύκλο ρολογιού είναι ακριβώς η ίδια με την λειτουργία κάθε βαθμίδας του CPA.

### 2.3.2.3 Αθροιστής σωσίματος κρατουμένου (CSA)

Ο αθροιστής σωσίματος κρατουμένου έχει ως είσοδο τρείς δυαδικούς αριθμούς και παράγει στην έξοδο έναν αριθμό σε αναπαράσταση αθροίσματος-κρατουμένου. Για αυτόν τον λόγο ο CSA αναφέρεται ορισμένες φορές και ως 3:2 συμπιεστής.




Σχήμα 2.9: Αθροιστής σωσίματος-κρατουμένου (CSA)

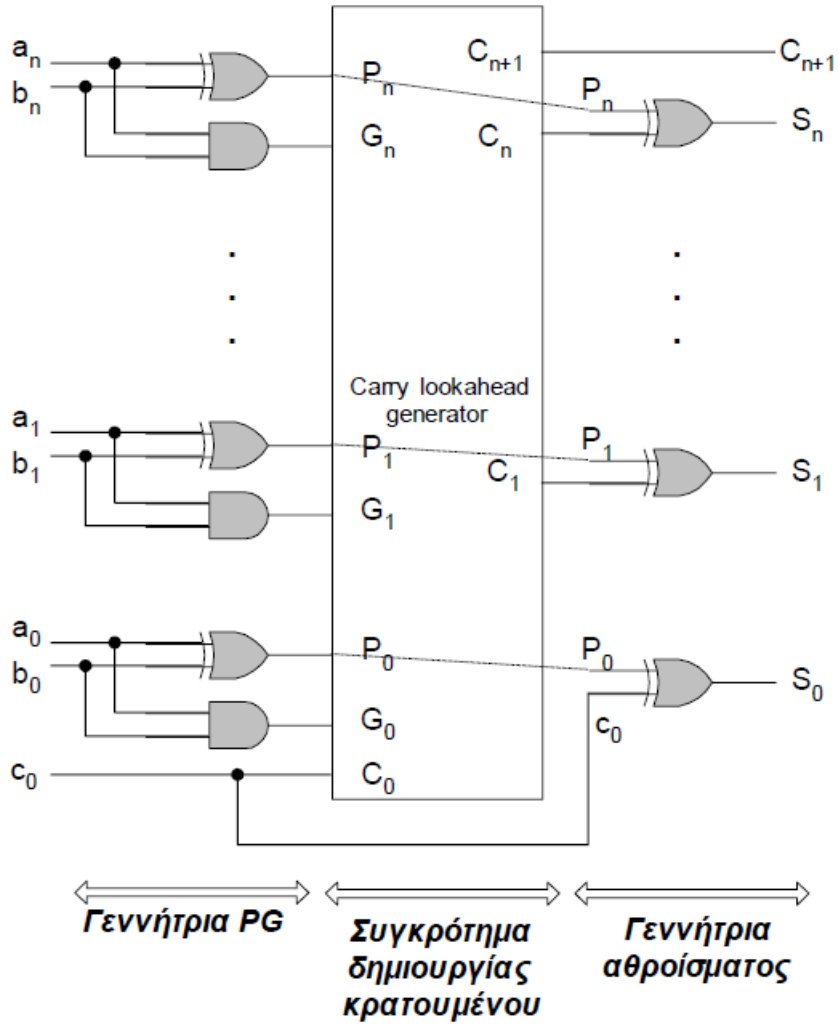
Σε αντίθεση με τον CPA, στον CSA δεν υπάρχει διάδοση κρατουμένου μεταξύ των βαθμίδων, γεγονός που τον καθιστά πολύ ταχύτερο. Πιο συγκεκριμένα, η καθυστέρηση ενός n-bit CSA είναι μόλις ένα επίπεδο FA, ανεξάρτητο από το μήκος bit των εισόδων του αθροιστή. Επίσης ο CSA έχει την ίδια επιφάνεια με τον CPA. Το μειονέκτημα του CSA είναι ότι το αποτέλεσμα που προκύπτει δεν βρίσκεται σε δυαδική αναπαράσταση όπως στον CPA αλλά σε CS αναπαράσταση.

### 2.3.2.4 Αθροιστής πρόβλεψης κρατουμένου

Όπως είδαμε προηγουμένως ο CPA δεν είναι αρκετά αποδοτικός λόγω της καθυστέρησης από την διάδοση του κρατουμένου μέσα από όλες τις βαθμίδες του. Η ιδέα πίσω από την

σχεδίαση του αθροιστή πρόβλεψης κρατουμένου είναι η αποφυγή της διάδοσης του κρατουμένου με την βοήθεια μίας γεννήτριας πρόβλεψης κρατουμένου που υπολογίζει εκ των προτέρων τα κρατούμενα όλων των βαθμίδων.

Πιο συγκεκριμένα, ένας CLA με προστιθέμενους των n-bit αποτελείται από μια σειρά με n το πλήθος HA για τον υπολογισμό των σημάτων $P_i$ και $G_i$, μια γεννήτρια πρόβλεψης κρατουμένου που αποτελείται από ένα δενδρικής μορφής δίκτυο μονάδων πρόβλεψης κρατουμένου (#) και τέλος μία σειρά από *n* το πλήθος πύλες XOR για τον υπολογισμό του τελικού αθροίσματος σε δυαδική αναπαράσταση.




Σχήμα 2.10: Αθροιστής πρόβλεψης κρατουμένου

Ας εξετάσουμε όμως πιο προσεκτικά την σημασία και τον ρόλο των σημάτων $P_i$ και $G_i$. Το σήμα $G_i = a_i \cdot b_i$ ονομάζεται σήμα γέννησης κρατουμένου διότι παίρνει την τιμή της μονάδας μόνο σε περίπτωση όπου τα bit των προσθετέων είναι και τα δύο μονάδα. Με άλλα λόγια όταν το $G_i$ είναι μονάδα, παράγεται ένα κρατούμενο εξόδου ανεξάρτητα από την τιμή του κρατουμένου εισόδου της αντίστοιχης βαθμίδας. Από την άλλη μεριά, σε περίπτωση που ένα μόνο από τα bit των προσθετέων είναι μονάδα τότε και το σήμα διάδοσης κρατουμένου $P_i = a_i \oplus b_i$ είναι μονάδα. Σε αυτήν την περίπτωση, μόνο εάν το κρατούμενο εισόδου είναι μονάδα μπορεί να προκύψει κρατούμενο εξόδου, δηλαδή να έχουμε διάδοση κρατουμένου.

Από τις παραπάνω διαπιστώσεις καταλήγουμε στην σχέση που εκφράζει το κρατούμενο εξόδου μιας βαθμίδας: $C_{i+1}=G_i+P_i \cdot C_i$, όπου $C_i$ το κρατούμενου εισόδου που προέρχεται από την προηγούμενη βαθμίδα και $C_{i+1}$ το κρατούμενο εξόδου. Επίσης το τελικό άθροισμα μπορεί να εκφραστεί ως $S_i=C_i \oplus P_i$.

Για παράδειγμα εφαρμόζοντας την σχέση που δίνει το κρατούμενο εξόδου για έναν αθροιστή των τεσσάρων bit προκύπτει:

$C_1=G_0+P_0C_0$

$C_2=G_1+P_1G_0+P_1P_0C_0$

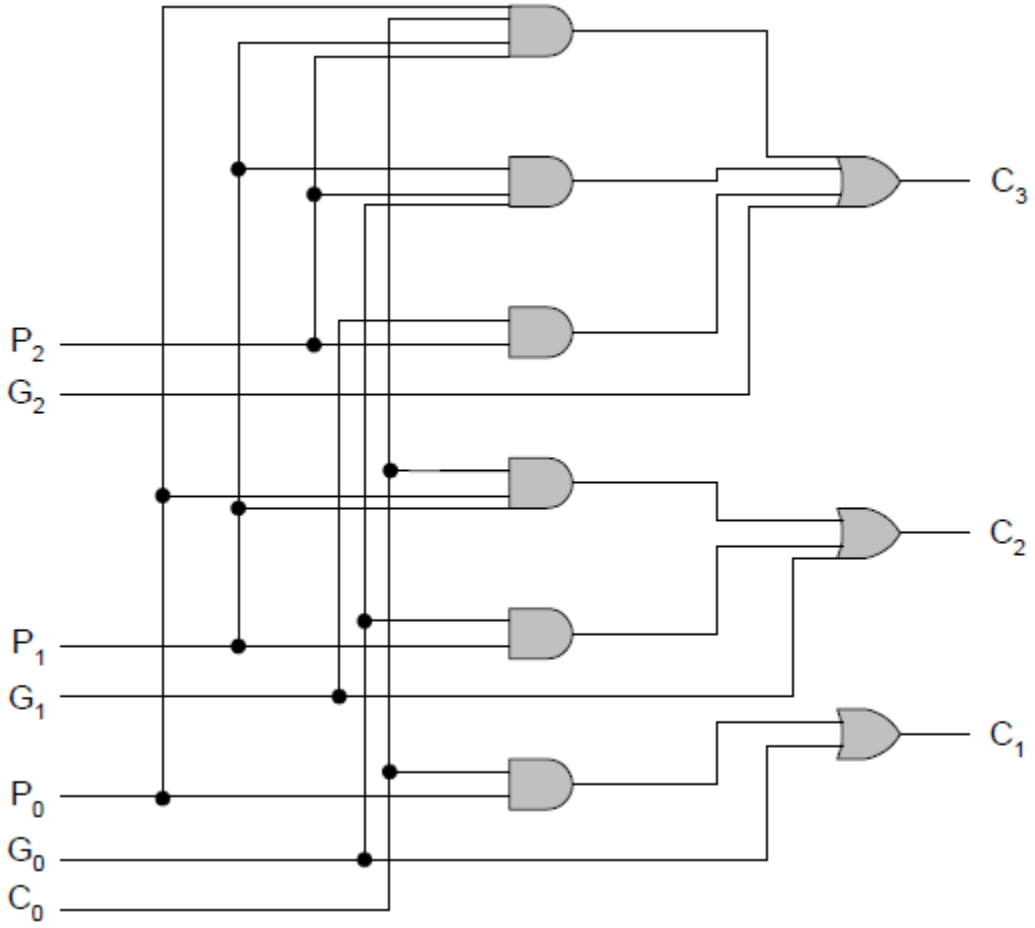
$C_3=G_2+P_2G_1+P_2P_1G_0+P_2P_1P_0C_0$

$C_4=G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1G_0+P_3P_2P_1P_0C_0$

Η γενική μορφή της σχέσης για το κρατούμενο είναι η παρακάτω:

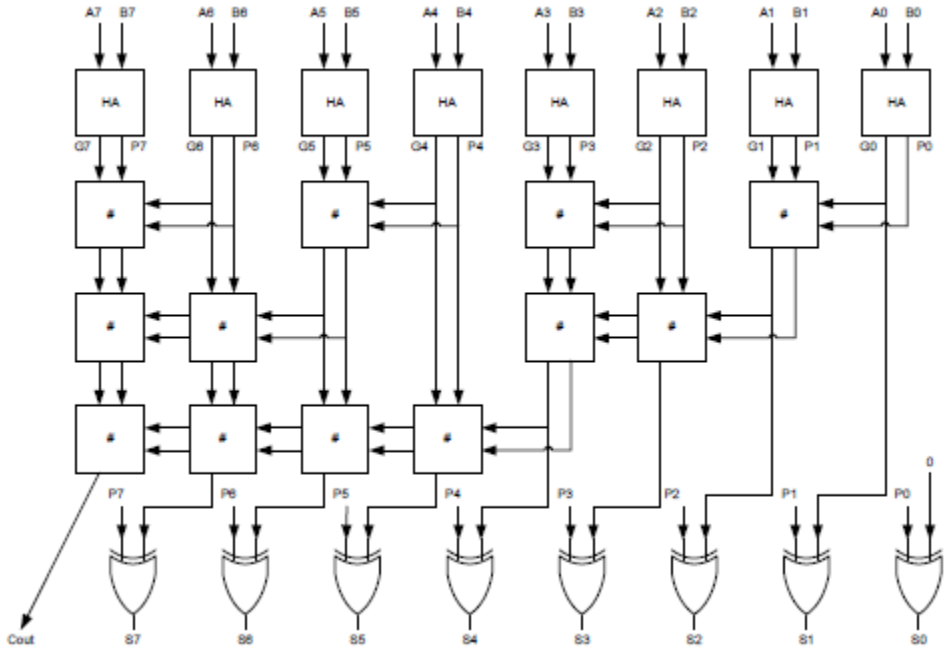
$C_{n+1} = G_n + P_n (G_{n-1} + P_{n-1} (G_{n-2} + P_{n-2}( G_{n-3} + P_{n-3} (\dots + P_2 (G_1 + P_1C_1)))))$ =

$= G_n + P_nG_{n-1} + P_nP_{n-1}G_{n-2} + P_nP_{n-1}P_{n-2}G_{n-3} + \dots + P_nP_{n-1}P_{n-2}\dots P_2G_1 + P_nP_{n-1}P_{n-2}\dots P_2P_1C_1$




Σχήμα 2.11: Κυκλωματική υλοποίηση γεννήτριας κρατουμένου

Με αυτόν τον τρόπο, το κύκλωμα δημιουργίας κρατουμένου εκμεταλλεύεται την προσεταιριστικότητα του τελεστή (#) για να υπολογίσει παράλληλα τα τελικά κρατούμενα. Τα τελικά αποτελέσματα προκύπτουν με καθυστέρηση $\log_2 N$ κυκλωμάτων (#). Στο παρακάτω σχήμα φαίνεται ένας αθροιστής πρόβλεψης κρατουμένου 8-bit.




Σχήμα 2.12: Αθροιστής πρόβλεψης κρατουμένου των 8-bit

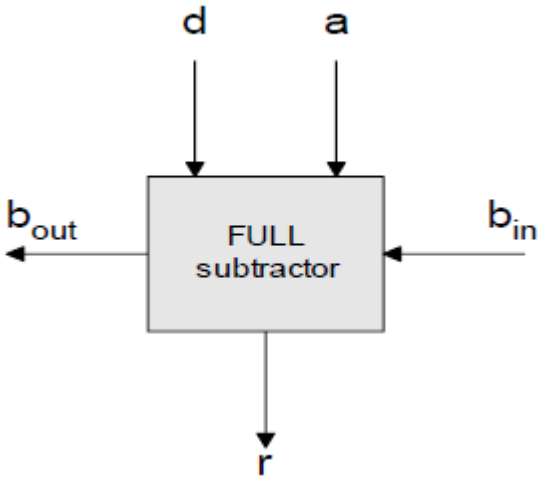
Συνολικά η καθυστέρηση του αθροιστή πρόβλεψης κρατουμένου είναι η καθυστέρηση ενός HA, συν $\log_2 N$ κυκλωμάτων (#), συν μιας πύλης XOR όπως φαίνεται και στο παραπάνω σχήμα. Η καθυστέρηση του CLA είναι σαφώς μικρότερη της καθυστέρησης του CPA καθώς ο πρώτος αποφεύγει την διάδοση κρατουμένου. Για μεγάλα μήκη λέξης, η επιφάνεια του CLA αυξάνει πολύ, καθώς απαιτούνται $N \times \log N/2$ κυκλώματα (#), όπου $N$ το πλήθος των ψηφίων των εισόδων. Το τελευταίο γεγονός αποτελεί και το κύριο μειονέκτημα του CLA.

### 2.3.3 Αφαιρέτες

Στην προηγούμενη υποενότητα παρουσιάστηκαν τα βασικά είδη αριθμητικών κυκλωμάτων αθροιστών που χρησιμοποιούνται συχνότερα στην πράξη. Αλλάζοντας μόνο την δομική μονάδα του πλήρους αθροιστή με εκείνη του πλήρους αφαιρέτη που θα παρουσιαστεί στην συνέχεια, προκύπτουν στην ίδια ακριβώς λογική τα αντίστοιχα είδη κυκλωμάτων αφαιρετών.

Το κύκλωμα του πλήρους αφαιρέτη κατά αντιστοιχία με τον πλήρη αθροιστή, έχει τρείς εισόδους δηλαδή τα ψηφία των αριθμών (αφαιρετέος $a$, αφαιρέτης $d$) και εφόσον το κύτταρο αυτό προορίζεται να χρησιμοποιηθεί και για την αφαίρεση αριθμών με περισσότερα του ενός ψηφία, θα πρέπει να δέχεται και το "δανεικό" από την αφαίρεση των ψηφίων της

μικρότερης τάξης ($b_{in}$). Οι έξοδοι του κυττάρου θα πρέπει να είναι δύο, εκ των οποίων η μία πρέπει να αποτελεί το υπόλοιπο της αφαίρεσης που διενεργείται σε αυτό ($r$), ενώ η δεύτερη θα αποτελεί το δανεικό που θα μεταφερθεί στο κύτταρο επόμενης τάξης ($b_{out}$).




Σχήμα 2.13: Πλήρης αφαιρέτης

Σε αντίθεση με την πρόσθεση, στο κύτταρο του δυαδικού αφαιρέτη οι είσοδοι δεν είναι ισοδύναμες. Στην περίπτωση του πλήρη αθροιστή, σε οποιαδήποτε από τις τρείς εισόδους και αν εφαρμόζαμε τα bits των δύο προσθετέων και του κρατουμένου, τόσο το άθροισμα όσο και το κρατούμενο εξόδου προέκυπταν σωστά. Δεν ισχύει το ίδιο όμως και για την είσοδο που προορίζεται ως ψηφίο αφαιρετέου στο δυαδικό αφαιρέτη. Ενώ οι άλλες δύο είσοδοι (δανεικό και bit αφαιρέτη) είναι ισοδύναμες μεταξύ τους, η είσοδος του αφαιρετέου δεν είναι ισοδύναμη με καμία από τις άλλες δύο. Το γεγονός αυτό προκύπτει εύκολα με ένα παράδειγμα: Θέτοντας σαν εισόδους του κυττάρου $a$=1, b=0, $b_{in}$=1, οπότε οι έξοδοι είναι $r$=0 και $b_{out}$=0, ενώ αν εναλλάξουμε τις τιμές των $a$ και $b$ οι έξοδοι γίνονται $r$=0 και $b_{out}$=1. Κατά συνέπεια, κατά την κατασκευή του κυττάρου του δυαδικού αφαιρέτη, η μία από τις δύο εισόδους του, πρέπει προορίζεται αποκλειστικά για το bit του αφαιρετέου. Από τα παραπάνω προκύπτει και ο πίνακας αληθείας του δυαδικού αφαιρέτη.

| ***A*** | ***d*** | $b_{in}$ | ***r*** | $b_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

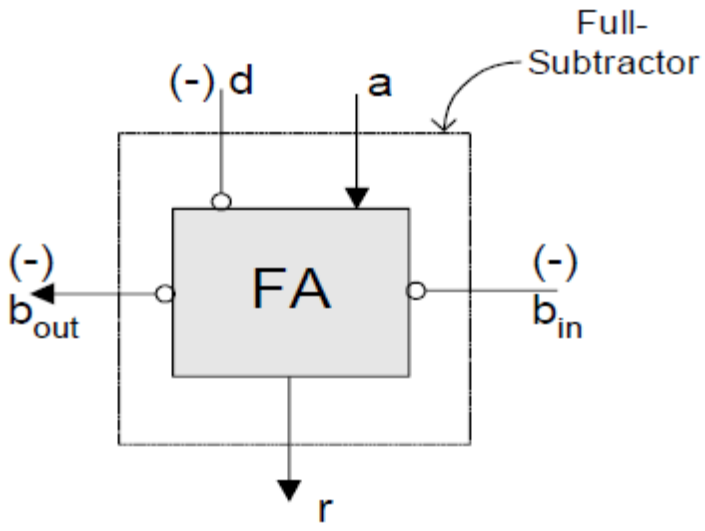
Πίνακας 2.10: Πίνακας αληθείας πλήρους αφαιρέτη

Από την πίνακα αληθείας του πλήρους αφαιρέτη παρατηρούμε ότι η έξοδος r του κυττάρου είναι περιττή συνάρτηση των εισόδων του. Άρα, το *r* προκύπτει από την λογική πράξη XOR όπως και το άθροισμα στον πλήρη αθροιστή. Όσον αφορά το $b_{\text{out}}$, αυτό προκύπτει από την έκφραση: $\overline{b_{out}} = a\,\overline{b_{in}} + \bar{d}\,\overline{b_{in}} + a\,\bar{d}$

Η μόνη διαφορά της παραπάνω έκφρασης σε σύγκριση με την έκφραση του κρατουμένου εξόδου του πλήρους αθροιστή έγκειται στην αναστροφή που υπάρχει στο *a* του $b_{\text{out}}$. Έτσι λοιπόν μπορούμε να παράγουμε το δανεικό του πλήρους αφαιρέτη από την έξοδο του κρατουμένου του πλήρους αθροιστή, προσθέτοντας μια αναστροφή σε μια από τις εισόδους του πλήρους αθροιστή και αυτή η είσοδος να παίζει το ρόλο του *a*. Όμως με την αντιστροφή του *a* επηρεάζεται και το *r*. Ωστόσο το πρόβλημα αυτό λύνεται εύκολα κάνοντας την κατάλληλη μετατροπή διότι όπως αναφέρθηκε προηγουμένως το r είναι περιττή συνάρτηση των εισόδων. Αντιστρέφοντας την τιμή της μιας από τις τρείς μεταβλητές εισόδου, η έξοδος του αθροίσματος στον πλήρη αθροιστή δίνει τώρα μονάδα όπου πριν την αλλαγή έδινε μηδενικό, και το αντίστροφο κάτι που επιβεβαιώνεται και από τις ιδιότητες της πύλης XOR. Πράγματι ισχύει $\bar{a} \oplus d \oplus b = \overline{a \oplus d \oplus b} = \bar{r}$. Επομένως, η σωστή έξοδος *r* προκύπτει από την έξοδο του αθροίσματος στον πλήρη αθροιστή με ανεστραμμένη τη μια είσοδο της, προσθέτοντας σε αυτήν την έξοδο μια αναστροφή.

Συμπερασματικά το τελικό λογικό διάγραμμα του κυττάρου του πλήρους αφαιρέτη είναι όμοιο με αυτό του πλήρη αθροιστή με μόνη την προσθήκη τριών αντιστροφέων, ενός στην

είσοδο αρνητικής αξίας *d*, ενός στην είσοδο του κρατουμένου και ενός στην έξοδο του κρατουμένου, όπως φαίνεται και στο σχήμα. Το υπόλοιπο εκφράζεται από την σχέση $r = a \oplus d \oplus b_{in} = a \oplus \bar{d} \oplus \overline{b_{in}}$




Σχήμα 2.14: Μετατροπή FA σε πλήρη αφαιρέτη

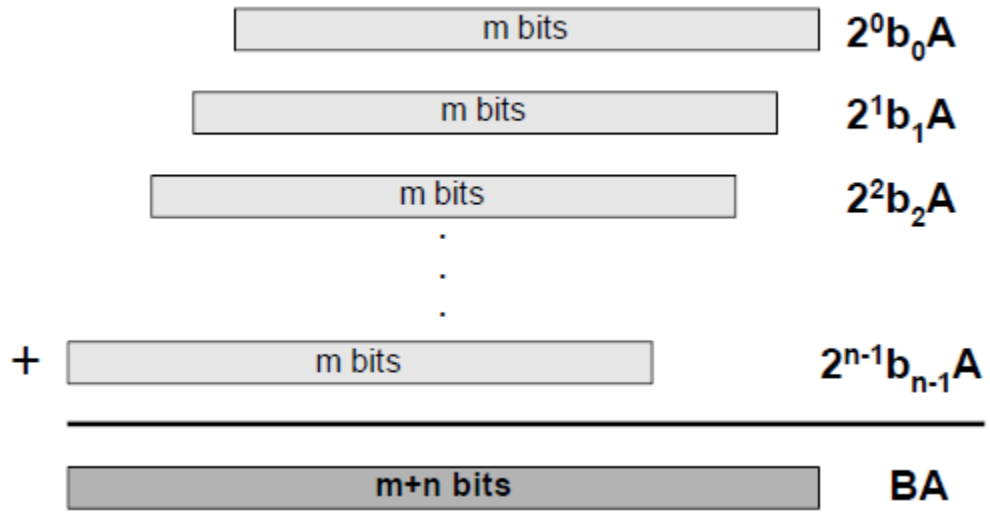
### 2.3.4 Πολλαπλασιαστές

Το κύκλωμα του πολλαπλασιαστή είναι η σημαντικότερη δομική μονάδα ενός φίλτρου FIR και η μελέτη του θα μάς απασχολήσει ιδιαίτερα στην παρούσα εργασία.

Έστω δυαδικοί αριθμοί *A* και *B* τους οποίους θέλουμε να πολλαπλασιάσουμε. Έστω ότι ο *A* αποτελείται από *m* bits ενώ ο *B* από *n* bits. Ορίζουμε τον *A* να είναι ο πολλαπλασιαστέος και ο *B* ο πολλαπλασιαστής. Η πράξη του πολλαπλασιασμού αναλύεται ως εξής:

$$A \cdot B = A \cdot \{2^0 b_0 + 2^1 b_1 + 2^2 b_2 \ldots + 2^{n-1} b_{n-1}\} = A \cdot \sum_{i=0}^{n-1} 2^i b_i$$

Ουσιαστικά συντελείται η λογική πράξη AND μεταξύ του κάθε bit του *B* με τον *A* και στην συνέχεια αθροίζονται όλα τα επιμέρους μερικά γινόμενα που προκύπτουν. Η διαδικασία της παραγωγής του τελικού γινομένου φαίνεται και στο σχήμα .




Σχήμα 2.15: Πολλαπλασιασμός δυο αριθμών (A×B)

Στην παρούσα ενότητα θα παρουσιαστούν τα βασικά είδη παράλληλων πολλαπλασιαστών όπως ο πολλαπλασιαστής διάδοσης κρατουμένου, ο πολλαπλασιαστής σωσίματος κρατουμένου και ο δενδρικός πολλαπλασιαστής Wallace ο οποίος χρησιμοποιείται κατά κόρον στην παρούσα εργασία. Όπως παρουσιάζεται ο σειριακός-παράλληλος πολλαπλασιαστής.

Ο παράλληλος πολλαπλασιαστής βασίζεται στην ιδιότητα ότι τα μερικά γινόμενα μπορούν να υπολογιστούν ανεξάρτητα. Για παράδειγμα έστω $A = \sum_{i=0}^{m-1} a_i 2^i$ $και$ $B = \sum_{j=0}^{n-1} b_j 2^j$ όπου *A* και *B*, απρόσημοι δυαδικοί αριθμοί. Το γινόμενο όπως δίνεται όπως την σχέση:

$$P = A \cdot B = \sum_{i=0}^{m-1} a_i 2^i \cdot \sum_{j=0}^{n-1} b_j 2^j = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (a_i b_j) 2^{i+j} = \sum_{k=0}^{m+n-1} p_k 2^k$$

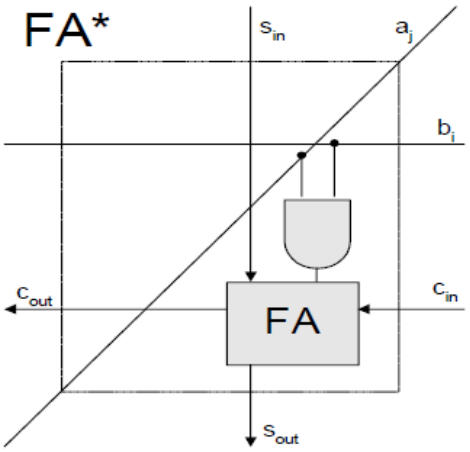
Για παράδειγμα, για αριθμούς των 4-bits, η παραπάνω σχέση αναπτύσσεται όπως φαίνεται στον πίνακα .

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | $a_3$ | $a_2$ | $a_1$ | $a_0$ | **Πολ/στέος** |
| | | | | $b_3$ | $b_2$ | $b_1$ | $b_0$ | **Πολ/στής** |
| | | | | $a_3b_0$ | $a_2b_0$ | $a_1b_0$ | $a_0b_0$ | |
| | | | $a_3b_1$ | $a_2b_1$ | $a_1b_1$ | $a_0b_1$ | | |
| | | $a_3b_2$ | $a_2b_2$ | $a_1b_2$ | $a_0b_2$ | | | |
| | $a_3b_3$ | $a_2b_3$ | $a_1b_3$ | $a_0b_3$ | | | | |
| $p_7$ | $p_6$ | $p_5$ | $p_4$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ | **Γινόμενο** |

Πίνακας 2.11: Παραγωγή μερικών γινομένων της πράξης $a \times b$

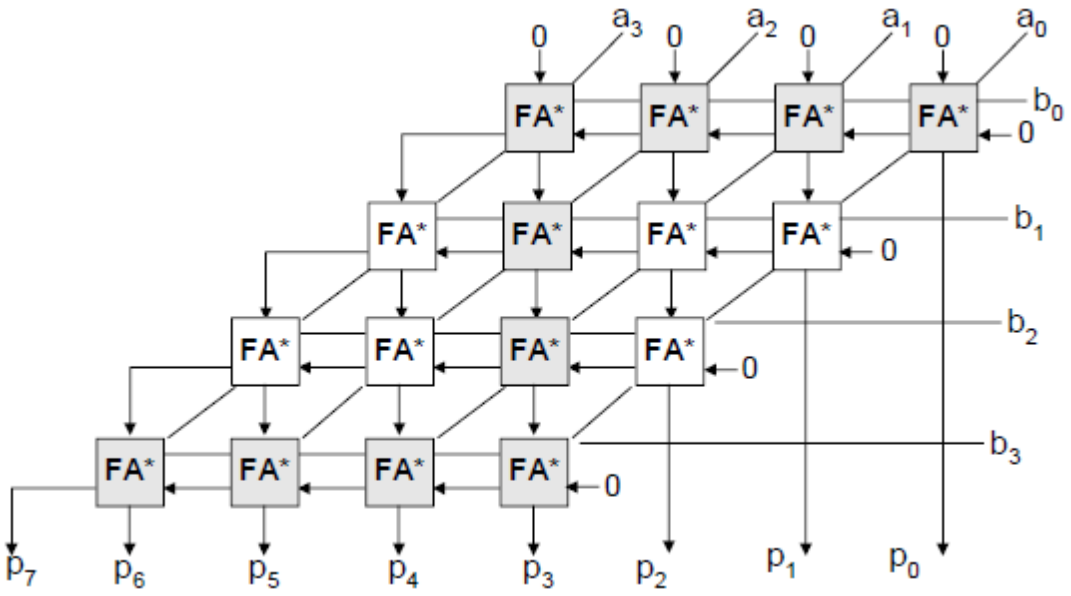
2.3.4.1 Παράλληλος πολλαπλασιαστής με άθροιση διάδοσης κρατουμένου

Γενικά ένας παράλληλος πολλαπλασιαστής, αποτελείται από ένα δίκτυο κυκλωμάτων που σκοπό έχουν την παραγωγή των μερικών γινομένων $a_ib_j$, το οποίο στην ουσία αποτελείται από λογικές πύλες AND. Επίσης συμπεριλαμβάνει αθροιστές οι οποίοι απαιτούνται για την πρόσθεση των μερικών γινομένων που προκύπτουν, που στην συγκεκριμένη περίπτωση είναι CPA. Η βασική δομική μονάδα του πολλαπλασιαστή διάδοσης κρατουμένου αποτελείται από έναν πλήρη αθροιστή και μία ενσωματωμένη πύλη AND. Η πύλη AND χρησιμεύει στην δημιουργία του ενδιάμεσου γινομένου των δύο bits των αριθμών, το οποίο εισέρχεται στον πλήρη αθροιστή για να προστεθεί με το ενδιάμεσο γινόμενο και το κρατούμενο της προηγούμενης βαθμίδας.




Σχήμα 2.16: Δομική μονάδα FA*

Ο πολλαπλασιαστής διάδοσης κρατουμένου αποτελείται από σειρές με δομικές μονάδες του σχήματος 2.16. Το κρατούμενο εξόδου κάθε δομικής μονάδας, γίνεται κρατούμενο εισόδου της επόμενης βαθμίδας, δηλαδή το κρατούμενο διαδίδεται μέσα από το δίκτυο των FA*.




Σχήμα 2.17: Παράλληλος πολλαπλασιαστής διάδοσης κρατουμένου

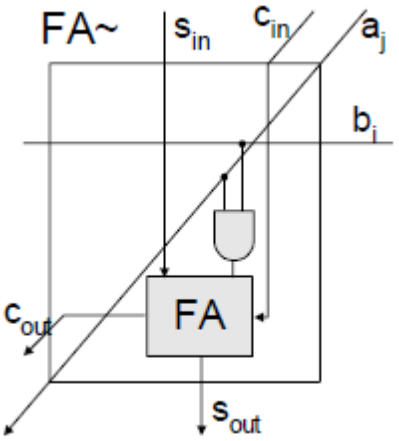
Όπως είδαμε προηγουμένως η μονάδα FA* έχει ενσωματωμένη και μια πύλη AND για τον υπολογισμό των $a_i b_j$. Όμως για τον υπολογισμό της καθυστέρησης του πολλαπλασιαστή, λαμβάνεται υπόψιν μόνο η καθυστέρηση της πύλης AND στο πάνω-δεξιά γωνιακό κύτταρο διότι τα $a_i$ και $b_j$ διαδίδονται μέσα στο δίκτυο του πολλαπλασιαστή χωρίς να παρεμβάλλεται καθυστέρηση. Επομένως, όταν τα $s_{in}$ και $c_{in}$ φτάνουν στα κύτταρα της επόμενης οριζόντιας σειράς, η λογική πράξη AND έχει ήδη εκτελεστεί.

Στο σχήμα φαίνεται γραμμοσκιασμένο ένα από τα κρίσιμα μονοπάτια. Διατρέχοντας το μονοπάτι προκύπτει ότι η πάνω οριζόντια σειρά αποτελείται από $n$-1 κύτταρα FA*, η κάθετη σειρά από m κύτταρα FA* και η κάτω οριζόντια σειρά από $n$-1 κύτταρα FA*. Σημειώνουμε ότι τα κοινά κύτταρα των οριζόντιων με την κάθετη σειρά προσμετρώνται μόνο στην κάθετη. Επομένως η συνολική καθυστέρηση δίνεται από τον τύπο $T=(m+n-1+n-1) \cdot T_{FA} \Rightarrow T=(2n+m-2) \cdot T_{FA}$, όπου $T_{FA}$ ο χρόνος καθυστέρησης του πλήρη αθροιστή, η οποία θεωρούμε ότι είναι ίδια και για τις δύο εξόδους του. Λόγω της κανονικότητας του κυκλώματός, οποιαδήποτε διαδρομή με οριζόντιες γραμμές από το πάνω-δεξιά μέχρι το κάτω-αριστερά

γωνιακό κύτταρο, η οποία αποτελεί κρίσιμο μονοπάτι του κυκλώματος, θα έχει την ίδια καθυστέρηση. Για τον πολλαπλασιαστή που εμφανίζεται στο σχήμα 2.17, ο οποίος έχει πολλαπλασιαστέο και πολλαπλασιαστή των 4-bits, δηλαδή $m=n=4$ ισχύει $T=10 \cdot T_{FA}$. Το γεγονός αυτό επιβεβαιώνεται με δοκιμή διαφορετικών μονοπατιών στο πολλαπλασιαστή του σχήματος όπου προκύπτει η ίδια καθυστέρηση των δέκα FA σε κάθε περίπτωση.

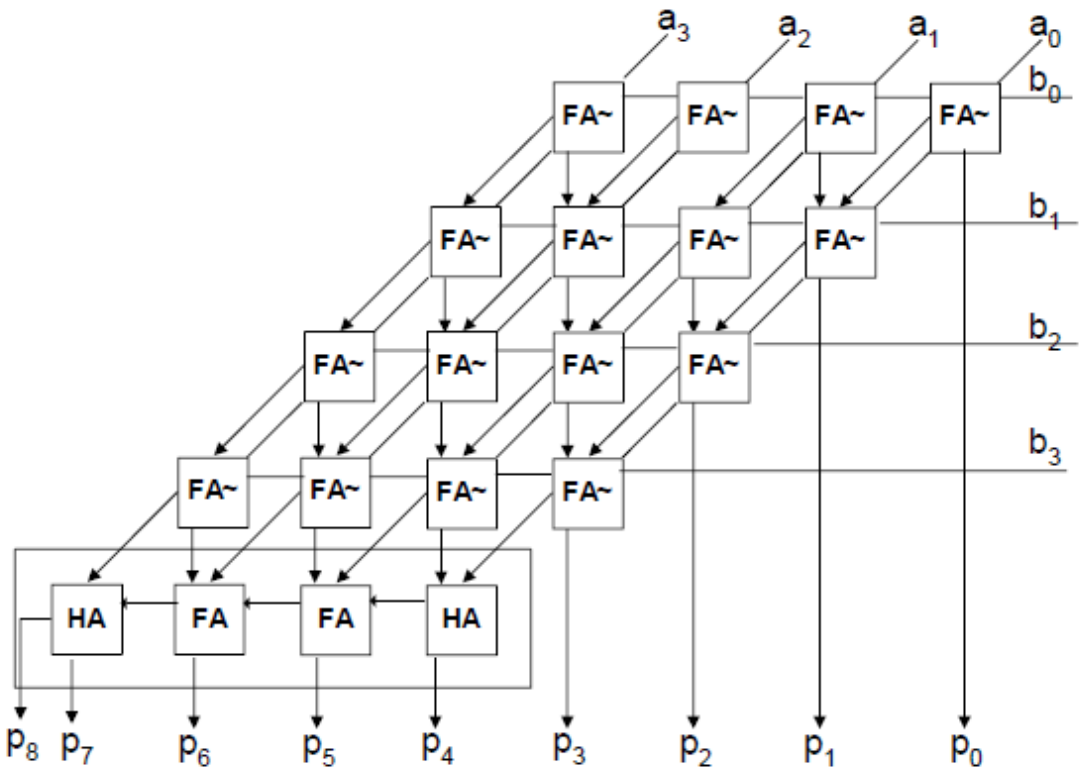
### 2.3.4.2 Παράλληλος πολλαπλασιαστής με άθροιση σωσίματος κρατουμένου

Ο παράλληλος πολλαπλασιαστής με άθροιση σωσίματος κρατουμένου, πρόκειται για έναν αρκετά διαδεδομένο τύπο πολλαπλασιαστή. Έχει παρόμοια φιλοσοφία με τον πολλαπλασιαστή διάδοσης κρατουμένου, με την διαφορά ότι τα κρατούμενα δεν διαδίδονται στο επόμενο κύτταρο του ίδιου επιπέδου, αλλά οδηγούνται στο επόμενο επίπεδο. Η βασική μονάδα του πολλαπλασιαστή αλλάζει μόνο ως προς την διάταξη των γραμμών, στην ουσία όμως είναι η ίδια.




Σχήμα 2.18: Δομική μονάδα FA~

Ο πολλαπλασιαστής αποτελείται από ένα δίκτυο δομικών μονάδων FA~ όπως φαίνεται στο επόμενο σχήμα 2.19.



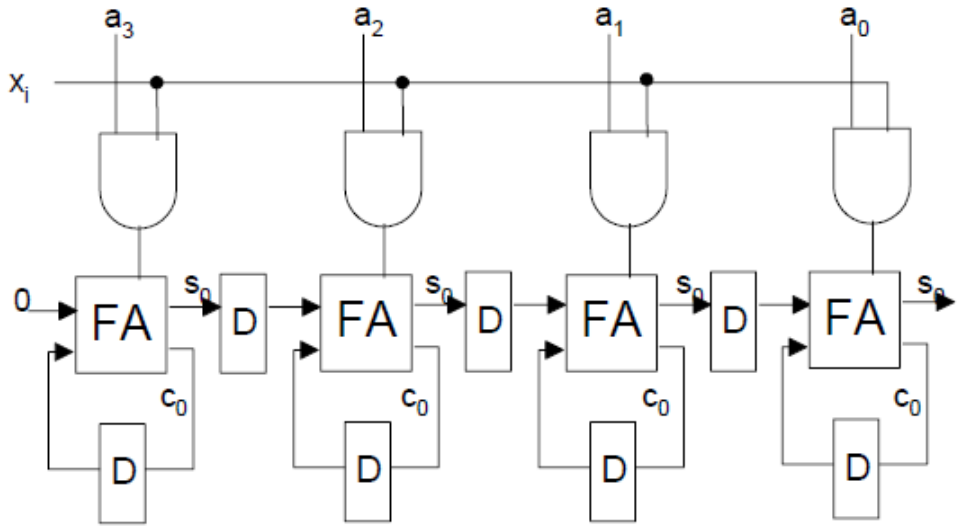

Σχήμα 2.19: Παράλληλος πολλαπλασιαστής σωσίματος κρατουμένου

Τα τέσσερα πιο σημαντικά bits του αποτελέσματος στο τέλος τις τέταρτης βαθμίδας βρίσκονται σε αναπαράσταση αθροίσματος-κρατουμένου. Το γεγονός αυτό προϋποθέτει την ύπαρξη τις αθροιστή των 4-bit για την τελική άθροιση ώστε να παραχθεί το αποτέλεσμα σε απλή δυαδική αναπαράσταση.

Ο χρόνος καθυστέρησης του πολλαπλασιαστή με σώσιμο κρατουμένου εξαρτάται άμεσα από τον τύπο του αθροιστή που θα χρησιμοποιηθεί στην τελευταία βαθμίδα. Σε περίπτωση που χρησιμοποιηθεί CPA, η καθυστέρηση δίνεται από την σχέση: $T=nT_{FA} + mT_{FA} \Rightarrow T=(n+m)T_{FA}$, όπου $nT_{FA}$ είναι η καθυστέρηση του αθροιστή τις τελευταίας βαθμίδας, ενώ $mT_{FA}$ του υπόλοιπου δικτύου. Τις αναφέρθηκε προηγουμένως, η καθυστέρηση λόγω τις πύλης AND μέσα στα κύτταρα δεν λαμβάνεται υπόψιν. Σε περίπτωση που χρησιμοποιηθεί CLA, η καθυστέρηση είναι μικρότερη: $T=(2+log_2 n)T_{FA}+mT_{FA} \Rightarrow T=(log_2 n+m+1)T_{FA}$. Από τις προηγούμενες σχέσεις παρατηρούμε ότι για $n=m=4$ οι καθυστερήσεις των δύο πολλαπλασιαστών με CPA και CLA αντίστοιχα, είναι ίσες. Επομένως η χρήση του CLA έχει νόημα για μεγάλες τιμές του $n$.

### 2.3.4.3 Σειριακός-παράλληλος πολλαπλασιαστής

Ο σειριακός-παράλληλος πολλαπλασιαστής αποτελείται από πύλες AND, πλήρεις αθροιστές και μανδαλωτές. Στο κύκλωμα αυτό ο πολλαπλασιαστέος $A$ εισέρχεται παράλληλα, ενώ ο πολλαπλασιαστής $X$ σειριακά.




Σχήμα 2.20: Σειριακός-παράλληλος πολλαπλασιαστής

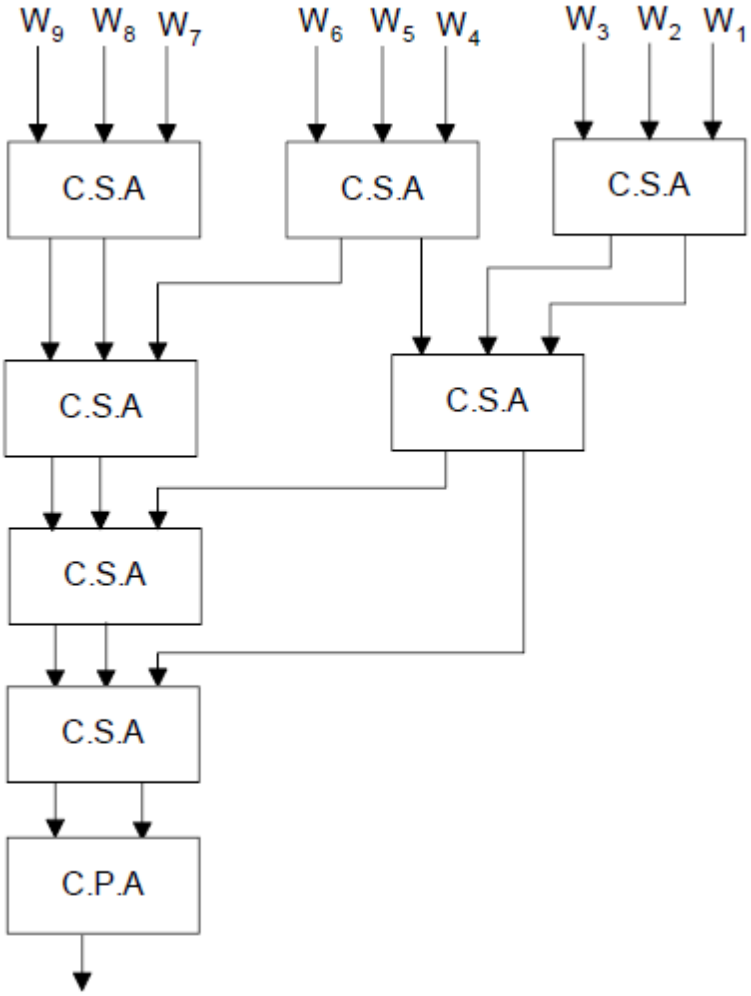
Ας αναλύσουμε την λειτουργία του πολλαπλασιαστή του σχήματος 2.20. Τα κρατούμενα κάθε πλήρη αθροιστή, αποθηκεύονται σε στοιχεία καθυστέρησης και επαναπροστίθενται με τον επόμενο κύκλο ρολογιού, στο νέο ζευγάρι $x_i a_j$. Τα ενδιάμεσα αθροίσματα διαδίδονται από τα αριστερά προς τα δεξιά περνώντας μέσα από μανδαλωτές. Τα bits του αριθμού a διατηρούνται στις εισόδους των πυλών AND ενώ με κάθε παλμό ρολογιού, εισάγονται τα bits του αριθμού X ξεκινώντας από το $x_0$. Αφού σχηματιστούν τα μερικά γινόμενα ($a_j x_i$) και προστεθούν στους αντίστοιχους πλήρεις αθροιστές με κρατούμενα και τα αθροίσματα από την προηγούμενη βαθμίδα, ολισθαίνονται κατά μια θέση προς τα δεξιά, αφήνοντας θέση για τα επόμενα μερικά γινόμενα.

Αν αναλύσουμε πιο προσεκτικά την λειτουργία των δύο πολλαπλασιαστών καταλήγουμε στο συμπέρασμα ότι υπάρχει μια χρονική αντιστοίχηση του σειριακού-παράλληλου και του παράλληλου πολλαπλασιαστή με σώσιμο κρατουμένου. Επομένως όπως και στην περίπτωση του παράλληλου αθροιστή σωσίματος κρατουμένου, μπορούμε να πούμε ότι ο παράλληλος πολλαπλασιαστής με σώσιμο κρατουμένου αποτελεί το ξεδίπλωμα στον χρόνο του σειριακού-παράλληλου πολλαπλασιαστή.

2.3.4.4 Δενδρικός πολλαπλασιαστής Wallace

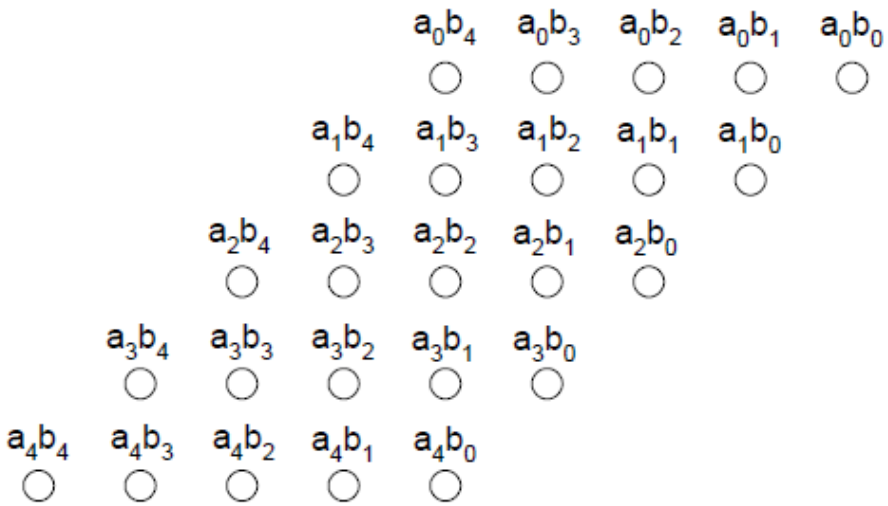
Ο δενδρικός πολλαπλασιαστής Wallace εκφράζει έναν διαφορετικό τρόπο υλοποίησης του πολλαπλασιασμού που στηρίζεται στην ιδιότητα των κυττάρων των πλήρων αθροιστών να συμπιέζουν τα τρία bits σε δύο. Έτσι, για να πραγματοποιηθεί η πρόσθεση $n$ bits, απαιτείται η ομαδοποίηση τους σε λέξεις των τριών bits και η χρησιμοποίηση μια σειράς από $n/3$ το πλήθος αθροιστές σωσίματος κρατουμένου. Το αποτέλεσμα είναι η συμπίεση των n bits σε $2n/3$ bits. Με εφαρμογή της ίδιας τακτικής σε ένα δεύτερο επίπεδο, παρατίθεται μια ακόμη σειρά από $(2n/3)\cdot(1/3)=2n/9$ πλήρεις αθροιστές, μειώνοντας ακόμη περισσότερο τον αριθμό των ψηφίων. Συνεχίζοντας με την ίδια λογική, στο τελευταίο βήμα της παραπάνω διαδικασίας προκύπτει ένας αριθμός σε αναπαράσταση αθροίσματος-κρατουμένου. Για την τελική πρόσθεση απαιτείται η χρησιμοποίηση ενός αθροιστή, όπως στο σχήμα 2.21 όπου χρησιμοποιείται ένας CPA. Η επιλογή ενός CLA ως τελικού αθροιστή είναι συνηθέστερη, καθώς εξυπηρετεί την επίτευξη υψηλότερης ταχύτητας. Η τελική μορφή του πολλαπλασιαστή σύμφωνα και με την παραπάνω περιγραφή, είναι μια δομή δένδρου η οποία αναφέρεται συνήθως ως “δέντρο Wallace”.

Ένας πολλαπλασιαστής με δέντρο Wallace είναι αρκετά γρήγορος διότι δεν υπάρχει διάδοση κρατουμένου στο κύκλωμα του. Η δομή του δένδρου στηρίζεται στην προσεταιριστική ιδιότητα της πρόσθεσης. Σε σχέση με τον πολλαπλασιαστή με σώσιμο κρατουμένου που παρουσιάστηκε στην προηγούμενη ενότητα, ο πολλαπλασιαστής με δένδρο Wallace αποτελείται από λιγότερα επίπεδα, για αυτόν τον λόγο είναι και ταχύτερος. Πιο συγκεκριμένα, απαιτεί $\log_{3/2}n=1.7\log_2 n$ επίπεδα. Αν και ταχύτερος ο δενδρικός πολλαπλασιαστής Wallace χρησιμοποιεί περισσότερο υλικό. Επίσης, οι διασυνδέσεις μεταξύ των κυττάρων CSA εμφανίζουν αυξημένη πολυπλοκότητα διότι δεν παρουσιάζουν κάποια κανονικότητα, με αποτέλεσμα να δυσκολεύουν την υλοποίηση του σε μορφή VLSI.



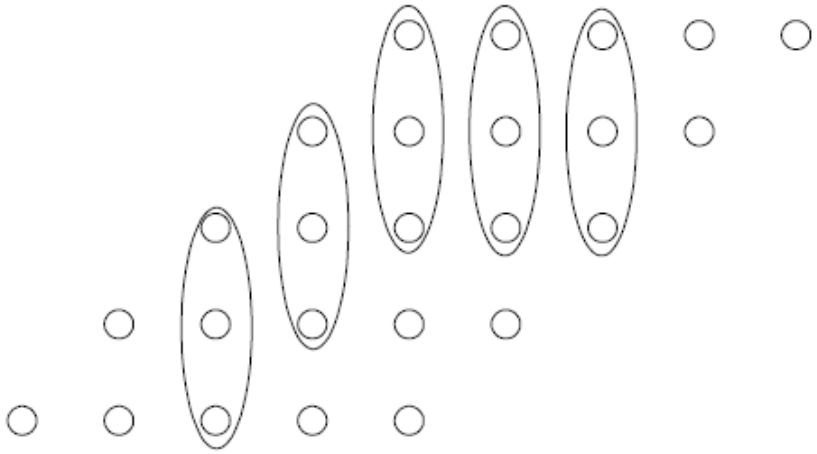

Σχήμα 2.21: Δίκτυο αθροιστών CSA

Στην συνέχεια παρουσιάζεται ο αλγόριθμος στον οποίο βασίζεται η κατασκευή ενός δένδρου Wallace, μέσα από ένα παράδειγμα πολλαπλασιασμού δύο 5-bit αριθμών $A$ και $B$. Το πρώτο βήμα είναι ο σχηματισμός όλων των μερικών γινομένων $a_i b_j$, όπου $i,j$ =0,1,2,...,5 όπως φαίνεται στο σχήμα 2.22.




Σχήμα 2.22: Παραγωγή μερικών γινομένων πολλαπλασιασμού $a \times b$

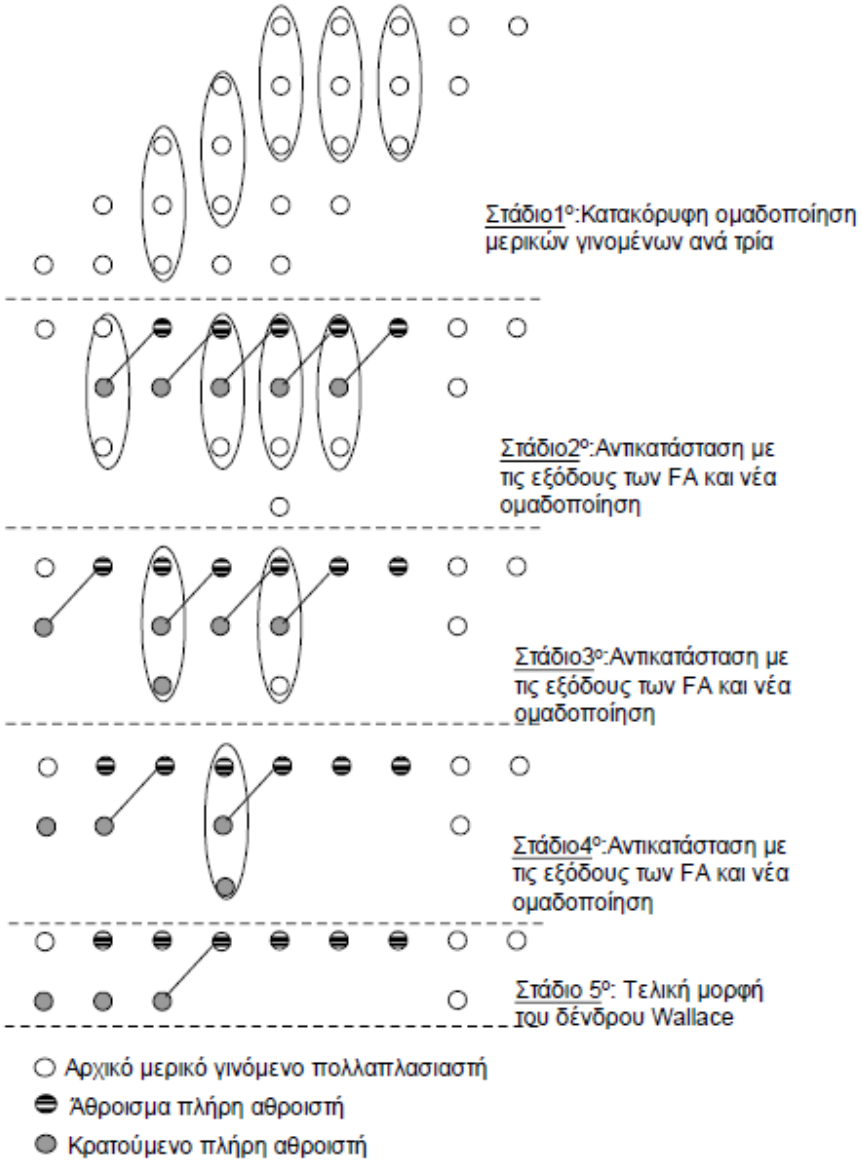
Με φορά από τα δεξιά προς τα αριστερά και ξεκινώντας από τα μικρότερης σημασίας μερικά γινόμενα του σχήματος, εξετάζουμε τις κατακόρυφες γραμμές του πίνακα και όταν εμφανίζονται τρία μερικά γινόμενα τα ομαδοποιούμε. Η διαδικασία αυτή παρουσιάζεται σχηματικά παρακάτω:



Σχήμα 2.23: Ομαδοποίηση μερικών γινομένων

Η ομαδοποίηση των μερικών γινομένων ανά τρία συνεχίζεται μέχρι της εξάλειψης όλων των τριάδων. Πιο συγκεκριμένα τα τρία μερικά γινόμενα που ομαδοποιούνται κάθε φορά εισάγονται σε έναν πλήρη αθροιστή και μετά την ολοκλήρωση της πρόσθεσης αντικαθίστανται με τις εξόδους του πλήρη αθροιστή. Από τις εξόδους, το άθροισμα,

τοποθετείται στην ίδια κατακόρυφο και το κρατούμενο τοποθετείται στην αμέσως αριστερότερη κατακόρυφο αφού έχει μεγαλύτερη αξία από το άθροισμα. Στο τελευταίο στάδιο τα ψηφία που απομένουν αποτελούν έναν αριθμό σε αναπαράσταση αθροίσματος-κρατουμένου. Για την εξαγωγή του τελικού αποτελέσματος απαιτείται η χρήση ενός αθροιστή διάδοσης ή πρόβλεψης κρατουμένου. Ολόκληρη η διαδικασία κατασκευής του δένδρου Wallace φαίνεται στο σχήμα 2.24.



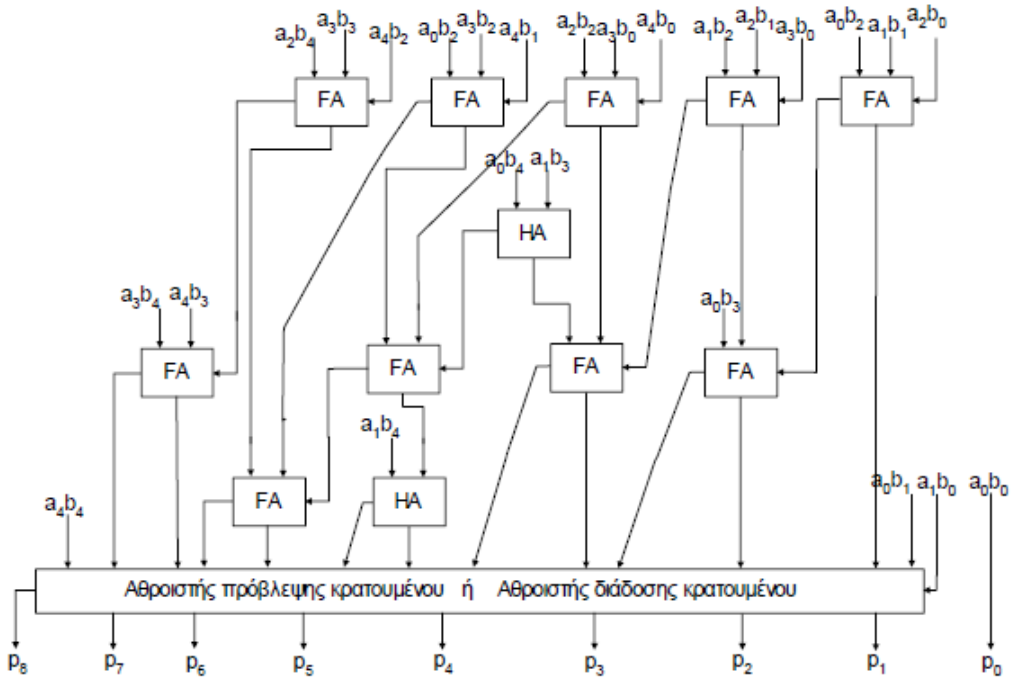

Σχήμα 2.24: Στάδια δημιουργίας δένδρου Wallace

Τα ψηφία που απομένουν στο τελευταίο επίπεδο πρέπει να προστεθούν σε έναν αθροιστή διάδοσης ή πρόβλεψης κρατουμένου για να παραχθεί το τελικό αποτέλεσμα. Για παράδειγμα αν χρησιμοποιήσουμε έναν αθροιστή διάδοσης κρατουμένου, θα χρειαστούμε

πλήρη αθροιστή όπου έχουμε δύο ψηφία στην ίδια κατακόρυφη και ημιαθροιστή όπου έχουμε ένα ψηφίο. Η δεύτερη κατακόρυφη αποτελεί εξαίρεση διότι αν και έχουμε δύο ψηφία μπορούμε να χρησιμοποιήσουμε ημιαθροιστή αφού το ψηφίο της πρώτης κατακόρυφης δεν εισάγεται στον αθροιστή διάδοσης κρατουμένου αλλά αποτελεί το MSB του γινομένου. Επομένως, απαιτούνται συνολικά 12 FA για το δέντρο Wallace, 3 FA και 5 HA για τον αθροιστή διάδοσης κρατουμένου του τελικού σταδίου.

Τα επίπεδα του δενδρικού πολλαπλασιαστή Wallace μπορούν να μειωθούν περαιτέρω με μια προσθήκη στον αλγόριθμο του σχήματος: όταν σε μια κατακόρυφη γραμμή στην οποία έχουμε κάνει ομαδοποίηση τριάδων περισσεύουν δύο μερικά γινόμενα, τα ομαδοποιούμε και αυτά και τα εισάγουμε σε έναν HA. Η υπόλοιπη διαδικασία παραμένει όπως και στο σχήμα 2.24.

Στο παρακάτω σχήμα παρουσιάζεται η οργάνωση των μερικών γινομένων δυο αριθμών των 5-bit σε επίπεδο πλήρων αθροιστών με τη χρήση της προσθήκης στον αλγόριθμο που περιγράφθηκε παραπάνω.




Σχήμα 2.25: Δένδρο Wallace με χρήση και HA

## 2.4 Αλγόριθμος Karatsuba

Ο αλγόριθμος Karatsuba ανακαλύφθηκε το 1960 από τον Anatoly Karatsuba και δημοσιεύθηκε το 1962. Ο αλγόριθμος του Karatsuba είναι ένας γρήγορος αλγόριθμος πολλαπλασιασμού. Περιορίζει τον πολλαπλασιασμό δύο αριθμών των n ψηφίων, σε $n^{log_2 3} \approx n^{1.585}$το μέγιστο μονοψήφιους πολλαπλασιασμούς, και ακριβώς $n^{log_2 3}$ όταν το n είναι δύναμη του δύο. Επομένως, είναι γρηγορότερος από τον κλασσικό αλγόριθμο, ο οποίος απαιτεί την διεξαγωγή $n^2$ μονοψήφιων πολλαπλασιασμών. Ο αλγόριθμος Karatsuba ήταν ο πρώτος αλγόριθμος πολλαπλασιασμού ασυμπτωτικά γρηγορότερος από τον κλασσικό σχολικό αλγόριθμο. Ο αλγόριθμος Toom-Cook είναι μια γρηγορότερη γενίκευση της μεθόδου Karatsuba, και ο αλγόριθμος Schonhage-Strassen είναι ακόμη γρηγορότερος για αρκετά μεγάλο *n*.

Ας εξετάσουμε όμως όλα τα βήματα του αλγορίθμου. Έστω δύο θετικοί αριθμοί *A* και *B* των *n*=2*k* bits έκαστος. Ο κάθε αριθμός μπορεί να διαιρεθεί σε δύο τμήματα των *k* bits ως εξής:
$A = A_H \cdot 2^k + A_L$ και $B = B_H \cdot 2^k + B_L$ (1)

Το γινόμενο των δύο αυτών αριθμών δίνεται από τις ακόλουθες σχέσεις:

$$P = A \cdot B = (A_H \cdot 2^k + A_L) \cdot (B_H \cdot 2^k + B_L) \Rightarrow$$

$$\Rightarrow P = A_H \cdot B_H \cdot 2^{2k} + (A_H \cdot B_L + A_L \cdot B_H) \cdot 2^k + A_L \cdot B_L \Rightarrow$$

$$\Rightarrow P = P_H \cdot 2^{2k} + P_M \cdot 2^k + P_L \quad (2)$$

Όπου $P_H = A_H \cdot B_H,\ P_M = A_H \cdot B_L + A_L \cdot B_H$ και $P_L = A_L \cdot B_L$ (3)

Υπολογίζουμε ακόμη την ποσότητα:

$dA \cdot dB = (A_H - A_L) \cdot (B_H - B_L) = A_H \cdot B_H + A_L \cdot B_L - (A_H \cdot B_L + A_L \cdot B_H)$

Η παραπάνω σχέση γράφεται και ως:

$dA \cdot dB = P_H + P_L - P_M$ (4)

Επομένως, ο όρος $P_M$δίνεται από την ακόλουθη σχέση: $P_M = P_H + P_L - dA \cdot dB$ (5)

Αντικαθιστώντας τον όρο $P_M$ όπως αυτός δίνεται από την σχέση (5) στην εξίσωση (2) που εκφράζει το γινόμενο *P* έχουμε:

$$P = P_H \cdot 2^{2k} + (P_H + P_L - dA \cdot dB) \cdot 2^k + P_L$$
$$= P_H \cdot 2^{2k} + P_H \cdot 2^k + P_L \cdot 2^k + P_L - dA \cdot dB \cdot 2^k$$

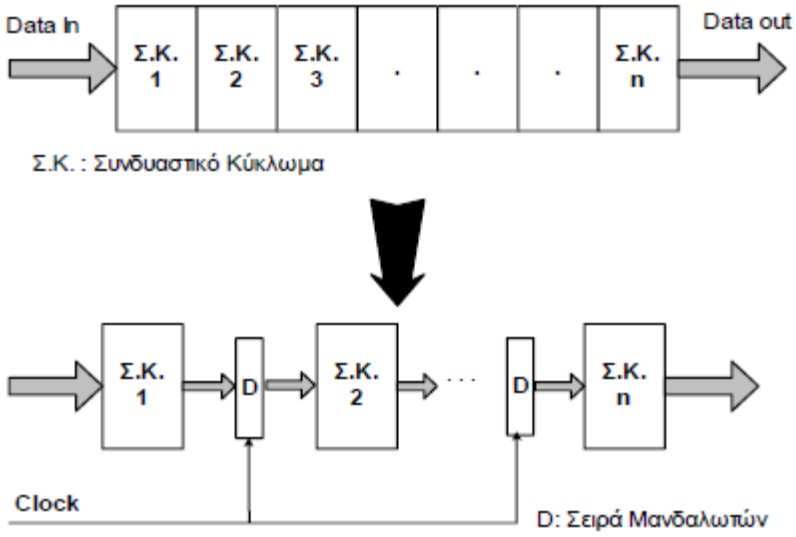
Παραγοντοποιώντας την παραπάνω έκφραση παίρνουμε τελικά:

$$P = A \cdot B = (P_H \cdot 2^k + P_L) \cdot (2^k + 1) - dA \cdot dB \cdot 2^k \qquad (6)$$

Μέσα από διαδοχικούς μετασχηματισμούς καταλήγουμε στην σχέση (6), η οποία παρατηρούμε ότι απαιτεί τρείς αντί για τέσσερις πολλαπλασιασμούς που απαιτούσε αρχικά η σχέση (2), με κόστος την αύξηση των προσθέσεων από τρείς σε έξι.

## 2.5 Τεχνική συνεχούς διοχέτευσης (Pipelining)

Η τεχνική της συνεχούς διοχέτευσης χρησιμοποιείται με σκοπό την βελτίωση του ρυθμού λειτουργίας (throughput) ενός κυκλώματος. Στο πάνω μέρος του σχήματος 2.26 φαίνεται ένα κύκλωμα που αποτελείται από *n* υποκυκλώματα. Για την μετατροπή του κυκλώματος σε συνεχούς διοχέτευσης είναι απαραίτητη η διαίρεση με παρεμβολή καθυστερήσεων μεταξύ των επιμέρους συνδυαστικών υποκυκλωμάτων όπως φαίνεται στο κάτω μέρος του σχήματος 2.26. Οι μονάδες αυτές μπορεί να είναι μανδαλωτές ή D flip-flops.




Σχήμα 2.26: Εφαρμογή της τεχνικής συνεχούς διοχέτευσης

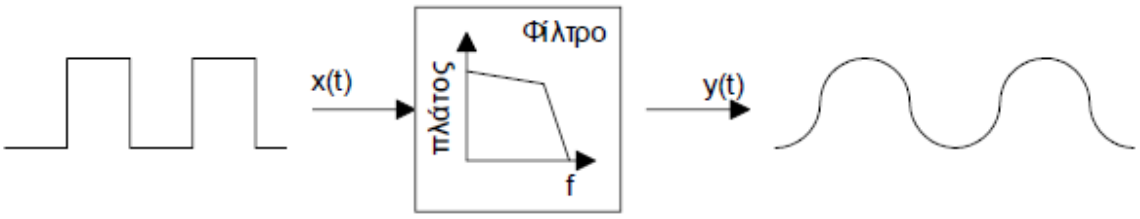
Με την εισαγωγή των μονάδων καθυστέρησης επιτυγχάνεται η αύξηση του ρυθμού λειτουργίας, διότι κάθε υποσύστημα ενεργεί πλέον ανεξάρτητα από τα υπόλοιπα με αποτέλεσμα να μην μένει ποτέ άεργο όσο αναμένει την εισαγωγή δεδομένων. Φυσικά, εξετάζοντας συνολικά το κύκλωμα, από την είσοδο μέχρι την έξοδο, η συνολική καθυστέρηση αυξάνεται λόγω της εισαγωγής των μονάδων καθυστέρησης. Έστω ότι τα *n* συνδυαστικά υποκυκλώματα είναι ισοδύναμα ως προς την καθυστέρηση και ότι το συνολικό κύκλωμα έχει καθυστέρηση *T*. Τότε το κάθε ένα από τα υποκυκλώματα θα έχει καθυστέρηση *T/n*. Για το απλό συνδυαστικό κύκλωμα, ισχύει ότι το κάθε υποκύκλωμα λειτουργεί για ένα χρονικό διάστημα ίσο με *T/n*, δηλαδή από την στιγμή που λαμβάνει τα δεδομένα εισόδου μέχρι να εξάγει τα αποτελέσματα του, και παραμένει ανενεργό για όλο το υπόλοιπο διάστημα λειτουργίας του συνολικού κυκλώματος δηλαδή για χρόνο (*n*-1)*T/n*. Αυτό συμβαίνει διότι τα δεδομένα εισόδου στην αρχή του κυκλώματος εισάγονται σε κάθε παλμό ρολογιού, και ο παλμός του ρολογιού πρέπει να διαρκεί τουλάχιστον όσο η καθυστέρηση του συνολικού κυκλώματος. Με την εισαγωγή των μονάδων καθυστέρησης ο παλμός του ρολογιού που απαιτείται είναι πλέον τουλάχιστον ίσος με την καθυστέρηση ενός μόνο από τα *n* υποκυκλώματα. Με την τεχνική της συνεχούς διοχέτευσης λοιπόν, αρκεί ένα χρονικό διάστημα αρχικοποίησης (που βέβαια είναι μεγαλύτερου από την απλή καθυστέρηση του συνολικού συνδυαστικού κυκλώματος) μέχρι να εξαχθεί το πρώτο αποτέλεσμα και στην συνέχεια τα αποτελέσματα κάθε υποκυκλώματος θα είναι διαθέσιμα σε κάθε παλμό ρολογιού.

## 2.6 Ψηφιακά φίλτρα πεπερασμένης κρουστικής απόκρισης (FIR)

### 2.6.1 Εισαγωγή

Ως φίλτρο ορίζεται ένα σύστημα επεξεργασίας σήματος, του οποίου η λειτουργία είναι να μεταβάλλει κατάλληλα το φασματικό περιεχόμενο ενός σήματος. Πιο συγκεκριμένα, ένα φίλτρο ενισχύει ή εξασθενεί επιθυμητές περιοχές συχνοτήτων του σήματος που καθορίζονται από την χαρακτηριστική εξίσωση του φίλτρου.

Τα φίλτρα μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες: σε αναλογικά και σε ψηφιακά. Τα αναλογικά φίλτρα επεξεργάζονται αναλογικά σήματα, δηλαδή συνεχή ηλεκτρικά μεγέθη και υλοποιούνται με την κατάλληλη διασύνδεση παθητικών (αντιστάσεων, πυκνωτών, πηνίων) και ενεργών (ενισχυτές) στοιχείων.




Σχήμα 2.27: Γενική μορφή αναλογικού φίλτρου

Με την βοήθεια ενός μετατροπέα A/D (Analog to Digital) το αναλογικό σήμα μετά από δειγματοληψία και ψηφιοποίηση μπορεί να μετατραπεί σε ψηφιακό. Με αυτόν τον τρόπο δίνεται η δυνατότητα η επεξεργασία του σήματος να γίνει αριθμητικά με ένα σύνολο αριθμητικών πράξεων που ουσιαστικά προσομοιώνουν την λειτουργία του αναλογικού φίλτρου. Το σύνολο αυτών των πράξεων αποτελούν το ψηφιακό φίλτρο.


Σχήμα 2.28: Ψηφιακό φίλτρο

Η χαρακτηριστική εξίσωση που περιγράφει την λειτουργία ενός ψηφιακού φίλτρου είναι:

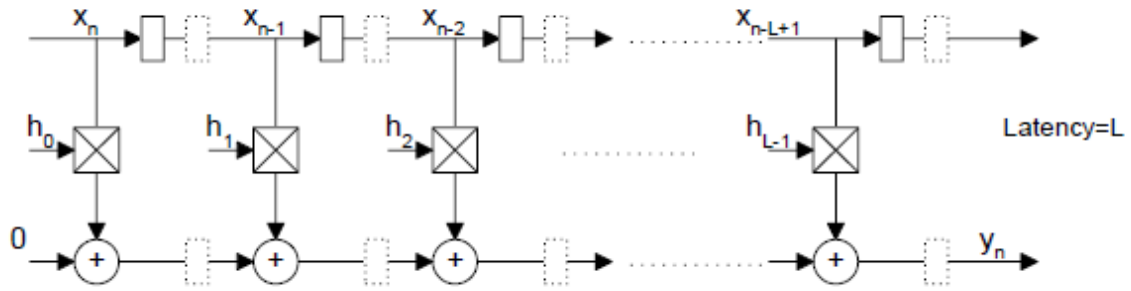
$$y_n = \sum_{i=0}^{L-1} x_{n-i} h_i$$

Από την παραπάνω σχέση προκύπτει ότι στο πεδίο του χρόνου, η έξοδος $y_n$ δίνεται από την συνέλιξη της εισόδου με τους συντελεστές (σταθερές) $h_i$, όπου οι συντελεστές αποτελούν την κρουστική απόκριση του φίλτρου.

Θεωρητικά, το ψηφιακό φίλτρο περιγράφεται από μία άπειρη σειρά, δηλαδή η κρουστική απόκριση ($h_i$) έχει άπειρη διάρκεια. Στην πράξη όμως, προσεγγίζεται ικανοποιητικά με έναν πεπερασμένο αριθμό συντελεστών ($h_0$, $h_1$,…, $h_{L-1}$). Για τις συνήθεις εφαρμογές ένα $L$ με τιμή από 16 έως 32 είναι ικανοποιητικό. Στην μορφή αυτή το ψηφιακό φίλτρο ονομάζεται

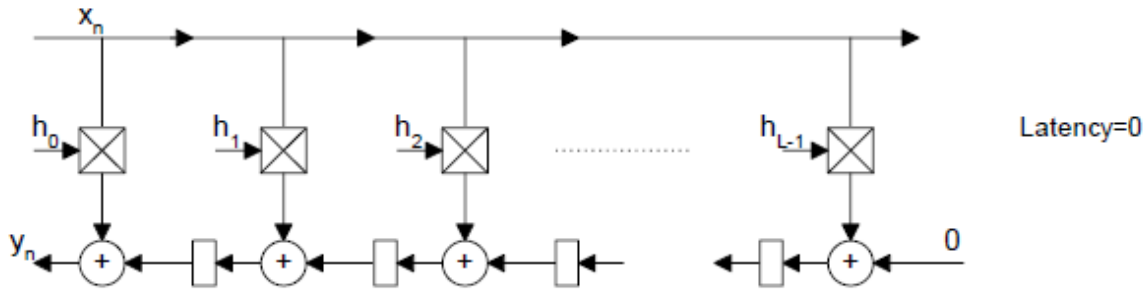
πεπερασμένης κρουστικής απόκρισης, και αναφέρεται συνήθως με την συντομογραφία FIR (Finite Impulse Response).

Οι δύο βασικές αρχιτεκτονικές που επιτρέπουν την υλοποίηση φίλτρων FIR είναι η direct και η transpose μορφή. Με απευθείας υλοποίηση της σχέσης που παρουσιάστηκε παραπάνω και περιγράφει την λειτουργία του φίλτρου, προκύπτει η direct υλοποίηση που δίνεται στο παρακάτω σχήμα.




Σχήμα 2.29: Απ' ευθείας (direct) υλοποίηση ενός φίλτρου FIR

Οι καθυστερήσεις με τις διακεκομμένες γραμμές είναι προαιρετικές και εισάγονται σε περίπτωση που επιθυμείται η εφαρμογή της διοχέτευσης των προσθέσεων στον κάτω κλάδο. Το μειονέκτημα της direct μορφής είναι ο μεγάλος αριθμός των μονάδων καθυστέρησης που πρέπει να εισαχθούν καθώς και η μεγάλη καθυστέρηση (latency), ίση με τον αριθμό των συντελεστών που παρουσιάζονται.



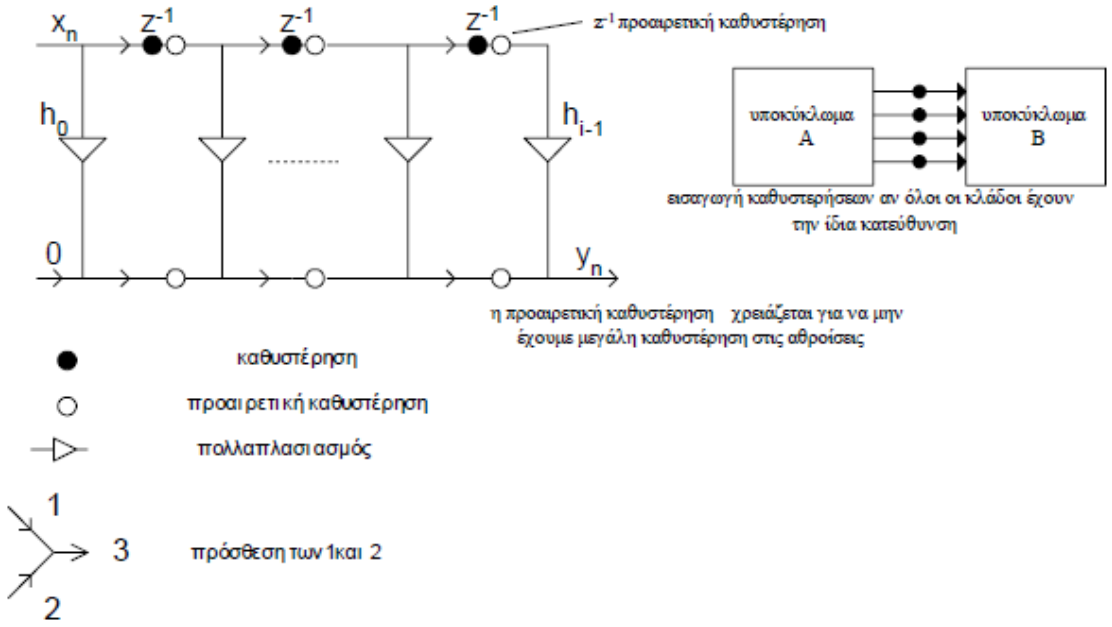

Σχήμα 2.30: Transposed μορφή ενός φίλτρου FIR

Στο παραπάνω σχήμα παρουσιάζεται η transposed μορφή. Σε αντίθεση με την direct, η transposed μορφή έχει μικρό αριθμό καθυστερήσεων και είναι άμεσης απόκρισης δηλαδή έχει μηδενικό latency. Το μειονέκτημα της transposed μορφής είναι η πιθανότητα εμφάνισης

χρονικής καθυστέρησης στην γραμμή του σήματος $x_n$ λόγω μήκους αλλά και επειδή τροφοδοτεί πολλές εισόδους.

### 2.6.2 Τεχνικές μετασχηματισμού γράφου σημάτων

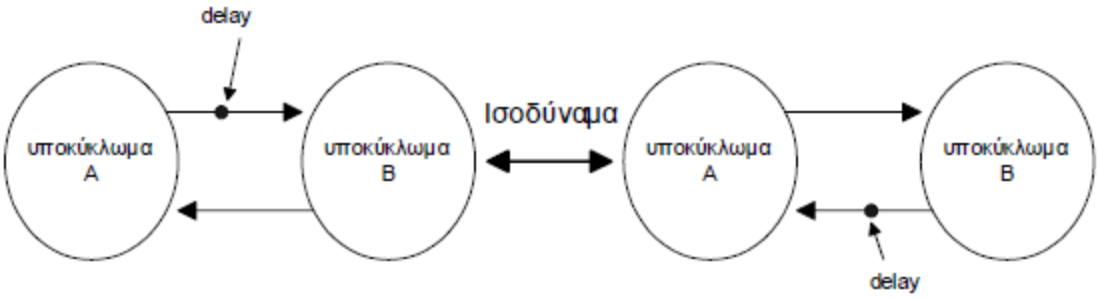
Με την εφαρμογή κατάλληλων μεθόδων είναι δυνατή η υλοποίηση των φίλτρων FIR σε συστολική μορφή. Στην παρούσα παράγραφο παρουσιάζονται τεχνικές μετασχηματισμού γράφων που εφαρμόζονται για τον σκοπό αυτό. Στο σχήμα που ακολουθεί παρουσιάζονται οι βασικοί συμβολισμοί που θα φανούν χρήσιμοι στην παρουσίαση των υπόλοιπων σχημάτων της παραγράφου.




Σχήμα 2.31: Το φίλτρο FIR σε συμβολισμό γράφου

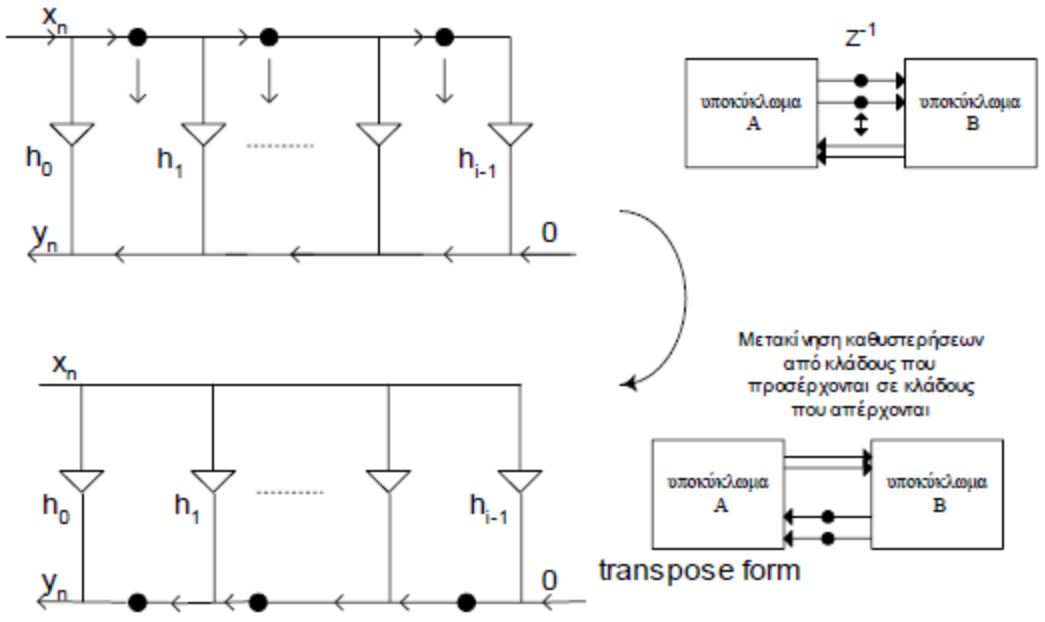
Μια βασική τεχνική που χρησιμοποιείται είναι η μετακίνηση καθυστερήσεων από έναν κόμβο σε έναν διαφορετικό, χωρίς να αλλάζει η λειτουργία του συνολικού κυκλώματος. Όπως φαίνεται στο σχήμα 2.32, παρά την μετακίνηση της καθυστέρησης σε διαφορετικό σημείο, προκύπτει ένα ισοδύναμο κύκλωμα. Προφανώς δεν αλλάζει η λειτουργία των υποκυκλωμάτων με την μετακίνηση της καθυστέρησης γιατί αντί η καθυστέρηση να είναι

στην είσοδο, βρίσκεται πλέον στην έξοδο (υποκύκλωμα *A*) και αντίστροφα για το άλλο (υποκύκλωμα *B*).



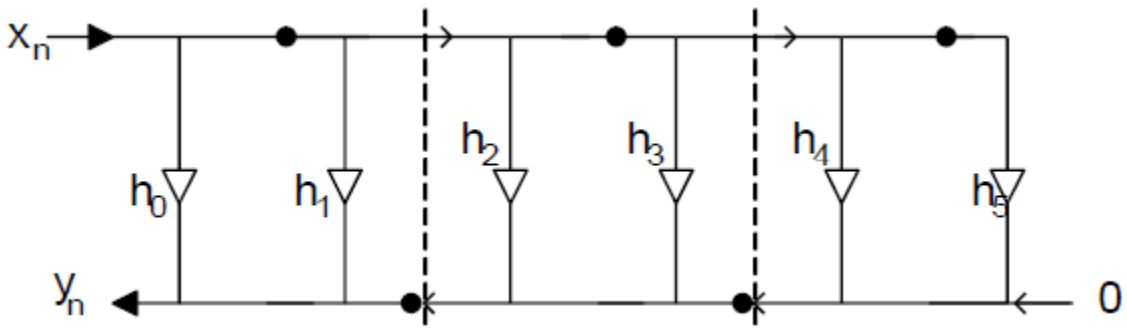

Σχήμα 2.32: Τεχνική μετακίνησης καθυστερήσεων

Ένα χαρακτηριστικό παράδειγμα εφαρμογής της μετακίνησης καθυστερήσεων σε ένα φίλτρο FIR παρουσιάζεται στο σχήμα 2.33. Είναι εμφανές ότι η transposed μορφή του φίλτρου προκύπτει από την direct μορφή με την μετακίνηση όλων των καθυστερήσεων από των άνω στον κάτω κλάδο του φίλτρου.



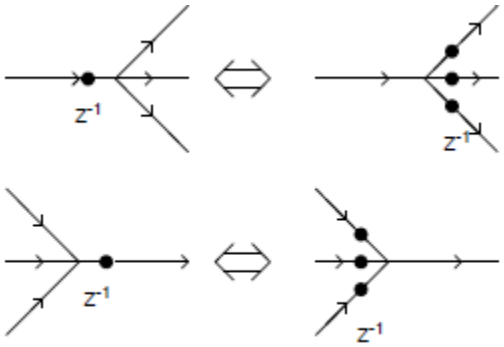

Σχήμα 2.33: Μετακίνηση καθυστερήσεων στον γράφο ενός φίλτρου FIR

Στην περίπτωση όπου δεν μετακινηθούν όλες οι καθυστερήσεις της direct μορφής στον κάτω κλάδο αλλά μόνο ορισμένες από αυτές, προκύπτει μια αρχιτεκτονική που είναι κάτι ενδιάμεσο μεταξύ της direct και της transposed υλοποίησης και αναφέρεται ως mixed μορφή.




Σχήμα 2.34: Mixed μορφή ενός φίλτρου FIR

Τέλος, μια καθυστέρηση μπορεί να μετακινηθεί μετά από έναν κόμβο ή πριν από έναν κόμβο όπως φαίνεται στο σχήμα 2.35.




Σχήμα 2.35: Μετακίνηση καθυστερήσεων πριν και μετά από τους κόμβους

# 3

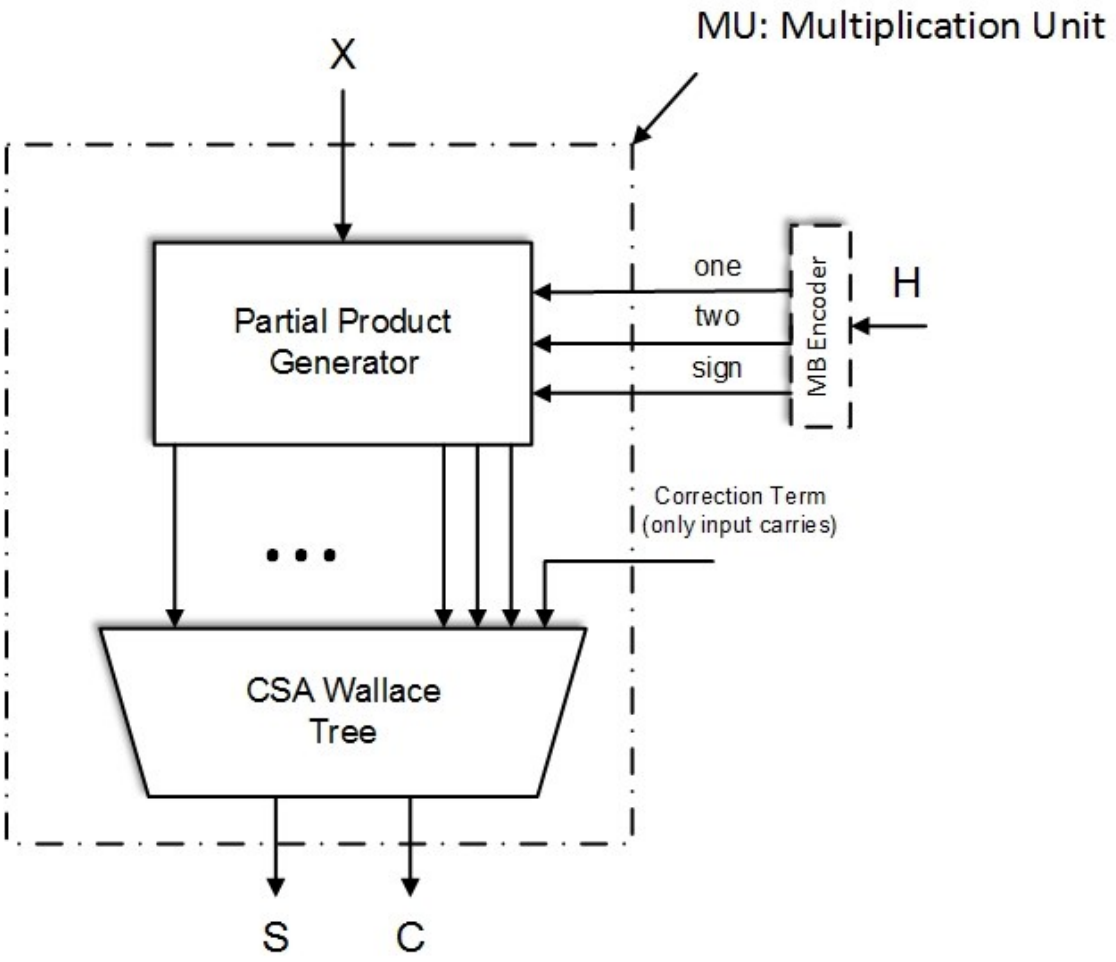
# ΣΧΕΔΙΑΣΜΟΣ ΚΥΚΛΩΜΑΤΟΣ ΦΙΛΤΡΩΝ

## 3.1 Παράλληλος πολλαπλασιαστής Wallace

Στο προηγούμενο κεφάλαιο, παρουσιάστηκε η βασική θεωρία για έναν γενικό παράλληλο πολλαπλασιαστή Wallace. Στην παρούσα ενότητα, γίνεται η παρουσίαση του παράλληλου πολλαπλασιαστή Wallace που υλοποιήθηκε και χρησιμοποιήθηκε ως δομική μονάδα σε όλες τις υλοποιήσεις των φίλτρων FIR. Όπως είδαμε, η μονάδα του πολλαπλασιαστή είναι η βασικότερη δομική μονάδα ενός φίλτρου FIR και η απόδοση του είναι καθοριστικής σημασίας για την συνολική απόδοση του κυκλώματος του φίλτρου.

Στο σχήμα 3.1, φαίνεται ο πολλαπλασιαστής που σχεδιάστηκε με σκοπό την χρησιμοποίηση του ως δομική μονάδα των φίλτρων. Παρατηρούμε ότι ο πολλαπλασιαστέος συμβολίζεται με *X* (δεδομένα εισόδου) και ο πολλαπλασιαστής με το γράμμα *H* (σταθεροί συντελεστές). Οι δομικές μονάδες που συνθέτουν τον πολλαπλασιαστή είναι μία μονάδα παραγωγής μερικών γινομένων (Partial Product Generator-PPG), και ένας αθροιστής CSA Wallace Tree για τον οποίο χρησιμοποιήθηκε το IP block *DW02_tree* της *Synopsys DesignWare*. Στο σχήμα 3.1 διακρίνεται με διακεκομμένη γραμμή και μία μονάδα που αναφέρεται ως MB encoder. Η μονάδα αυτή χρησιμοποιείται συνήθως με σκοπό την μετατροπή του πολλαπλασιαστή (*H*) από την μορφή συμπληρώματος ως προς δύο, σε κωδικοποίηση MB. Στην συγκεκριμένη υλοποίηση, η μονάδα αυτή δεν χρησιμοποιείται διότι ο πολλαπλασιαστής είναι προενταμιευμένος σε MB κωδικοποίηση (*one, two, sign*).

Τα μερικά γινόμενα προκύπτουν από την μονάδα PPG με κατάλληλη επεξεργασία των ψηφίων του αριθμού *X*, ανάλογα με τα MB ψηφία του αριθμού *H*. Στην συνέχεια, όλα τα μερικά γινόμενα που προκύπτουν από την μονάδα PPG εισάγονται σε έναν αθροιστή Wallace με σκοπό την πρόσθεση τους.




Σχήμα 3.1: Παράλληλος δενδρικός πολλαπλασιαστής Wallace με προενταμιευμένο σε MB κωδικοποίηση τον πολλαπλασιαστή (*H*) και αποτέλεσμα (γινόμενο) σε CS κωδικοποίηση.

Όπως θα δούμε παρακάτω, για να προκύψει ορθό αποτέλεσμα απαιτείται και η πρόσθεση ενός διορθωτικού όρου που περιέχει όλα τα κρατούμενα εισόδου και τους άσσους από την επέκταση προσήμου των μερικών γινομένων. Στην συγκεκριμένη υλοποίηση του πολλαπλασιαστή, ο διορθωτικός όρος περιέχει μόνο τα κρατούμενα εισόδου διότι όλοι οι άσσοι από την επέκταση προσήμου συγχωνεύονται με αυτούς των υπόλοιπων πολλαπλασιαστών του φίλτρου για να σχηματίσουν έναν τελικό διορθωτικό όρο (άθροισμα όλων των άσσων επέκτασης προσήμου για κάθε ενδιάμεσο γινόμενο του φίλτρου). Οι λεπτομέρειες για τους διορθωτικούς όρους θα αναλυθούν σε ξεχωριστή ενότητα στο παρόν κεφάλαιο. Το τελικό αποτέλεσμα που προκύπτει είναι σε CS αναπαράσταση.

Ο πολλαπλασιαστής του σχήματος 3.1 χρησιμοποιείται σε όλες τις υλοποιήσεις των φίλτρων FIR της παρούσας εργασίας. Η μόνη διαφορά σε κάθε περίπτωση, είναι οι είσοδοι του

πολλαπλασιαστή, δηλαδή ο πολλαπλασιαστέος και ο πολλαπλασιαστής. Πιο συγκεκριμένα, στην περίπτωση των συμβατικών υλοποιήσεων των φίλτρων, ο πολλαπλασιαστέος προέρχεται από τα δεδομένα εισόδου ($X = x_{2k-1}\, x_{2k-2}\, \ldots\, x_1\, x_0$) ενώ ο πολλαπλασιαστής από τους σταθερούς συντελεστές του φίλτρου ($H = h_{2k-1}\, h_{2k-2}\, \ldots\, h_1\, h_0$). Οι υλοποιήσεις των φίλτρων στις οποίες εφαρμόζεται ο αλγόριθμος Karatsuba, προκύπτουν από την σύνθεση τριών υποφίλτρων που λειτουργούν παράλληλα σύμφωνα με τις εξισώσεις του αλγορίθμου όπως θα δούμε σε επόμενη ενότητα. Έτσι λοιπόν ο πολλαπλασιαστέος και ο πολλαπλασιαστής είναι αντίστοιχα $X_H$ και $H_H$ ή $X_L$ και $H_L$ ή $-dX$ και $dH$.

Όπου: $X_H = x_{2k-1}\, x_{2k-2}\, \ldots\, x_{k+1}\, x_k$

$H_H = h_{2k-1}\, h_{2k-2}\, \ldots\, h_{k+1}\, h_k$

$X_L = x_{k-1}\, x_{k-2}\, \ldots\, x_1\, x_0$

$H_L = h_{k-1}\, h_{k-2}\, \ldots\, h_1\, h_0$

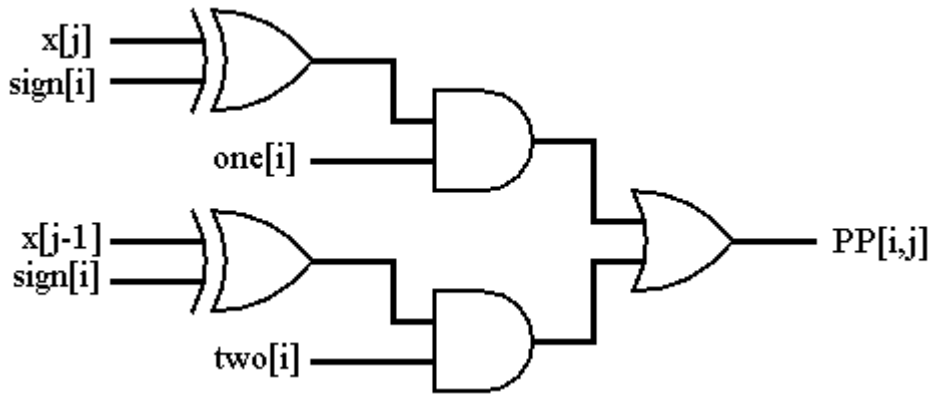
$dX = X_H - X_L$

$dH = H_H - H_L$

### 3.1.1 Μονάδα παραγωγής μερικών γινομένων (PPG)

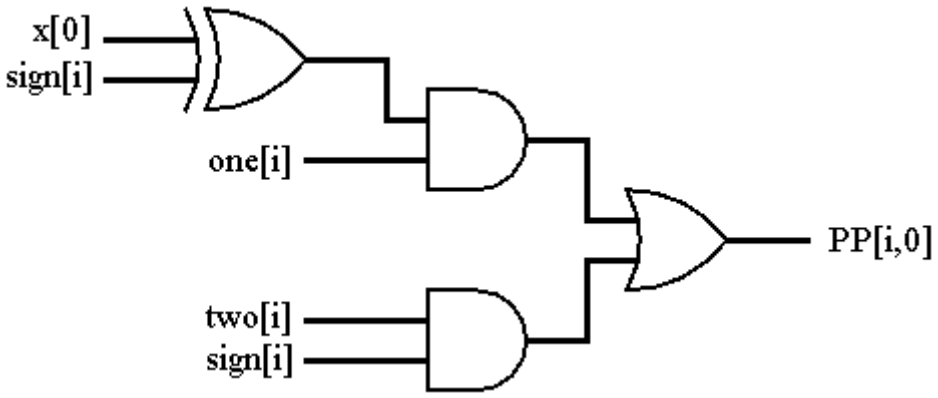
Η μονάδα παραγωγής μερικών γινομένων έχει ως εισόδους τον πολλαπλασιαστέο που βρίσκεται σε 2's complement αναπαράσταση όπως και τα προενταμιευμένα MB ψηφία του πολλαπλασιαστή. Έστω $X$ ο πολλαπλασιαστέος και $H$ ο πολλαπλασιαστής αμφότεροι αριθμοί με $2k$ ψηφία. Όπως είδαμε στο προηγούμενο κεφάλαιο, οι MB μεταβλητές (one_$H$, two_$H$ και sign_$H$) αποτελούνται από $k$ bits η κάθε μια. Σύμφωνα με τα προηγούμενα, από την μονάδα PPG δημιουργούνται $k$ το πλήθος μερικά γινόμενα, καθένα από τα οποία αποτελείται από $2k$+1 bits.

Στο σχήμα 3.2 απεικονίζεται το κύκλωμα για την δημιουργία των ψηφίων κάθε μερικού γινομένου με $j$ = 1,2,…,$2k$-1 όπου ο δείκτης $j$ συμβολίζει την θέση bit σε κάθε μερικό γινόμενο. Το $PP$[i,j] αναφέρεται στο $j$-οστό ψηφίο *του* $i$-οστού μερικού γινομένου. Παρατηρούμε ότι τα εν λόγω ψηφία προκύπτουν ως εξής: όταν το ψηφίο του *one* είναι μονάδα τότε είναι ακριβώς τα ψηφία του $X$, ενώ όταν το ψηφίο *two* είναι μονάδα τότε αυτά προκύπτουν με αριστερή ολίσθηση κατά μια θέση των ψηφίων του $X$. Σε περίπτωση όπου και το *one* και το *two* είναι μηδενικά τότε όλα τα ψηφία είναι επίσης μηδέν.



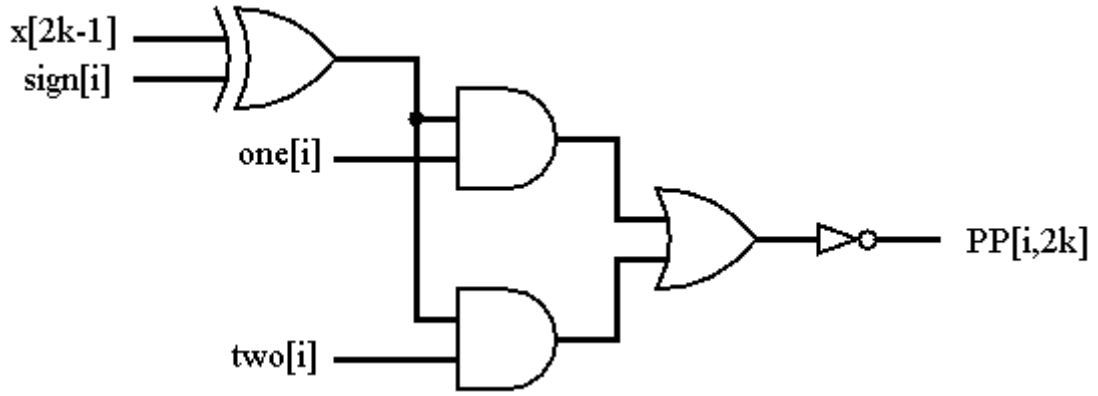

Σχήμα 3.2: Κύκλωμα παραγωγής των ψηφίων με j = 1,2,…,2k-1 για κάθε μερικό γινόμενο

Όπως αναφέρθηκε στην ενότητα 2.2.2, στην Modified Booth κωδικοποίηση τα ψηφία του αριθμού που αρχικά βρίσκεται σε 2's complement αναπαράσταση, κωδικοποιούνται ανά τρία. Για αυτόν τον λόγο είναι απαραίτητη και η θεώρηση ενός ψηφίου $h_{-1}$ με σκοπό την κωδικοποίηση της τριάδας $\{h_1, h_0, h_{-1}\}$. Ισχύει πάντα ότι $h_{-1} = 0$ και εκμεταλλευόμαστε το γεγονός αυτό για να απλοποιήσουμε το κύκλωμα για την παραγωγή του LSB κάθε μερικού γινομένου όπως φαίνεται στο σχήμα 3.3.




Σχήμα 3.3: Κύκλωμα παραγωγής του LSB κάθε μερικού γινομένου ($j$=0)

Το πιο σημαντικό bit του μερικού γινομένου προκύπτει με αντιστροφή του bit προσήμου όπως φαίνεται στο σχήμα 3.4. Το γεγονός αυτό οφείλεται στην επέκταση προσήμου των μερικών γινομένων όπως θα δούμε στην επόμενη ενότητα.

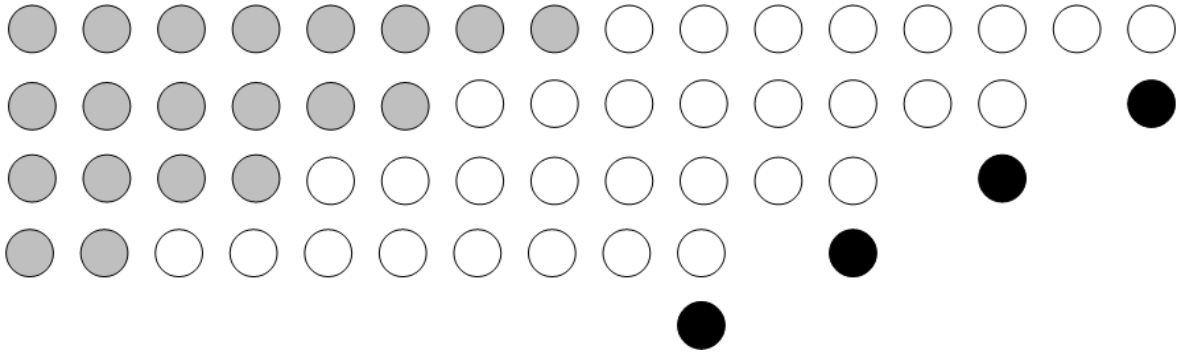



Σχήμα 3.4: Κύκλωμα παραγωγής του MSB κάθε μερικού γινομένου (j=2k)

### 3.1.2 Διορθωτικοί όροι μερικών γινομένων

Στην προηγούμενη ενότητα, παρουσιάστηκε η μονάδα παραγωγής των μερικών γινομένων η οποία αποτελεί δομική μονάδα του παράλληλου πολλαπλασιαστή Wallace (MU). Το αποτέλεσμα του πολλαπλασιασμού δύο αριθμών με μήκος $n=2k$ bits, είναι ένας αριθμός με μήκος $4k$ bits. Όμως, τα μερικά γινόμενα που παράγονται από την μονάδα PPG έχουν μήκος $2k+1$ bits και για αυτόν τον λόγο απαιτείται η εφαρμογή της μεθόδου επέκτασης προσήμου έτσι ώστε κάθε μερικό γινόμενο να έχει μήκος $4k$ bits. Ακόμη, όταν το Modified Booth ψηφίο *sign* του πολλαπλασιαστή είναι μονάδα, τότε τα ψηφία του πολλαπλασιαστέου αντιστρέφονται, παράγοντας το συμπλήρωμα ως προς ένα του αριθμού. Επομένως, για την μετατροπή του αριθμού στην επιθυμητή μορφή που είναι το συμπλήρωμα ως προς δύο του αριθμού, απαιτείται να προστεθεί μια μονάδα.

Στο σχήμα 3.5 δίνεται ένα παράδειγμα στο οποίο απεικονίζονται τα μερικά γινόμενα και οι διορθωτικοί όροι για έναν πολλαπλασιασμό 8×8 bit. Με λευκές κουκίδες απεικονίζονται τα bits των μερικών γινομένων, με γκρι κουκίδες τα bits της επέκτασης προσήμου και με μαύρες κουκίδες τα κρατούμενα εισόδου δηλαδή τα ψηφία που χρειάζεται να προστεθούν για την μετατροπή κάθε μερικού γινομένου σε μορφή συμπληρώματος ως προς δύο.



Σχήμα 3.5: Μερικά γινόμενα και διορθωτικοί όροι για έναν πολλαπλασιασμό 8bitx8bit.

Το αποτέλεσμα ενός πολλαπλασιασμού 8bit × 8bit είναι ένας αριθμός που έχει 16 ψηφία. Για αυτόν τον λόγο είναι απαραίτητη η επέκταση κάθε μερικού γινομένου μέχρι την βαθμίδα βάρους 15 (ένας 16bit αριθμός έχει βαθμίδες 15-0) και αυτό πραγματοποιείται με την συμπλήρωση του τελευταίου ψηφίου (ψηφίου προσήμου) του κάθε μερικού γινομένου. Όπως θα δούμε στην συνέχεια, με κατάλληλους μετασχηματισμούς η επέκταση προσήμου μπορεί να γίνει πολύ αποδοτικότερη ώστε να αποφευχθεί η χρησιμοποίηση μέρους του υλικού για την πρόσθεση ενός αριθμού άσσων και μηδενικών τα οποία λαμβάνουν λίγες προκαθορισμένες μορφές.

Στο παράδειγμα μας υπάρχουν τέσσερα μερικά γινόμενα, στα οποία εφαρμόζεται επέκταση προσήμου. Το γεγονός αυτό περιγράφεται με την παρακάτω σχέση:

$$ct = \sum_{k=0}^{3} s_k \cdot \sum_{i=8+2k}^{15} 2^i$$

Η σχέση αυτή όμως αφορά το παράδειγμα μας, δηλαδή έναν πολλαπλασιασμό 8bit × 8bit. Η σχέση που εκφράζει την γενική περίπτωση ενός nbit × nbit πολλαπλασιασμό είναι:

$$ct = \sum_{k=0}^{n/2} s_k \cdot \sum_{i=n+2k}^{2n} 2^i$$

Με χρήση της σχέσης:

$$\sum_{q=j}^{k} 2^q = 2^{k+1} - 2^j$$

Η σχέση που περιγράφει το παράδειγμα μας γίνεται:

$$ct = \sum_{k=0}^{3} s_k \cdot \{2^{16} - 2^{8+2k}\} = \sum_{k=0}^{3} -s_k \cdot 2^{8+2k}$$

Και στην γενική περίπτωση:

$$ct = \sum_{k=0}^{n/2} s_k \cdot \{2^{2n} - 2^{n+2k}\} = \sum_{k=0}^{n/2} -s_k \cdot 2^{n+2k}$$

Στην συνέχεια, χρησιμοποιώντας την σχέση:

$$-s_k = \overline{s}_k - 1$$

Για το παράδειγμα μας προκύπτει ότι:

$$ct = \sum_{k=0}^{3} \overline{s}_k \cdot 2^{8+2k} - \{2^{14} + 2^{12} + 2^{10} + 2^8\}$$

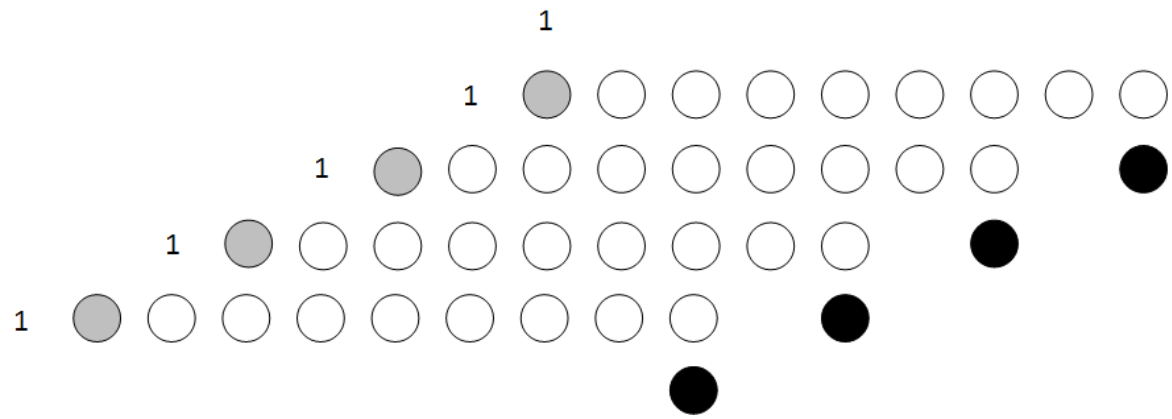
Τέλος, μετατρέποντας τον δεύτερο όρο της παραπάνω σχέσης σε συμπλήρωμα ως προς δύο, λαμβάνουμε την ισοδύναμη σχέση:

$$ct = \sum_{k=0}^{3} \overline{s}_k \cdot 2^{8+2k} + \{2^{15} + 2^{13} + 2^{11} + 2^9 + 2^8\}$$

Στην γενική περίπτωση, η παραπάνω παίρνει την εξής μορφή:

$$ct = \sum_{k=0}^{n/2} \overline{s}_k \cdot 2^{n+2k} + \{2^{2n-1} + \cdots + 2^{n+3} + 2^{n+1} + 2^n\}$$

Η τελευταία σχέση καθορίζει την σύνθεση των διορθωτικών όρων ως εξής: πρέπει να προστεθεί το συμπλήρωμα του αντίστοιχου bit προσήμου σε κάθε αντίστοιχο άρτιο βάρος που είναι μεγαλύτερο ή ίσο του μήκους λέξης *n*, και μια μονάδα στο βάρος *n* και σε κάθε περιττό βάρος μεγαλύτερο του *n*. Πλέον είναι ξεκάθαρος και ο λόγος που το κύκλωμα της μονάδας PPG για την παραγωγή του MSB των μερικών γινομένων έχει την μορφή που είδαμε στο σχήμα 3.4 της προηγούμενης ενότητας.




Σχήμα 3.6: Μερικά γινόμενα και απλοποιημένοι διορθωτικοί όροι για έναν πολλαπλασιασμό 8bit × 8bit

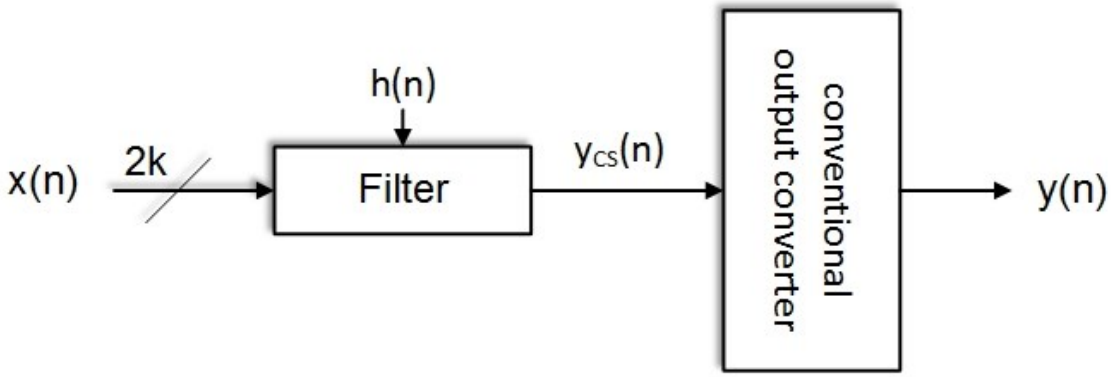
Στο σχήμα 3.6 παρουσιάζεται με παραστατικό τρόπο η απλοποιημένη μορφή των διορθωτικών όρων η οποία είναι σαφώς πολύ αποδοτικότερη από εκείνην του σχήματος 3.5. Όπως παρατηρούμε στο σχήμα 3.6, οι άσσοι από την επέκταση προσήμου έχουν μια προκαθορισμένη μορφή, σε αντίθεση με τα κρατούμενα εισόδου που εξαρτώνται από την τιμή του αντίστοιχου Modified Booth ψηφίου *sign*.

Στην παρούσα εργασία, εκμεταλλευόμαστε το γεγονός αυτό για την περαιτέρω απλοποίηση των διορθωτικών όρων σε επίπεδο φίλτρων FIR. Πιο συγκεκριμένα, το πλήθος των μονάδων πολλαπλασιαστή ενός φίλτρου FIR είναι ανάλογο της τάξης του φίλτρου (πλήθος σημείων του φίλτρου). Το πλήθος των σημείων του φίλτρου το συμβολίζουμε με *t* από τον αγγλικό όρο taps. Επομένως σε κάθε φίλτρο FIR υπάρχουν *t* ενδιάμεσα γινόμενα των δεδομένων εισόδου με τους συντελεστές του φίλτρου τα οποία πρέπει να προστεθούν.

Κάθε ενδιάμεσο γινόμενο απαιτεί την δημιουργία ενός διορθωτικού όρου. Όμως, το τμήμα κάθε διορθωτικού όρου που σχετίζεται με τους άσσους από την επέκταση προσήμου είναι πανομοιότυπο για κάθε ενδιάμεσο γινόμενο. Δηλαδή ο ίδιος όρος πρέπει να προστεθεί *t* φορές σε ένα φίλτρο FIR, αυτό όμως είναι ισοδύναμο με την ολίσθηση αριστερά κατά $log_2(t)$ θέσεις. Επομένως, απαιτείται ένας μόνο προκαθορισμένος όρος, που συμπεριλαμβάνει όλους τους άσσους από την επέκταση προσήμου κάθε ενδιάμεσου γινομένου και προστίθεται (hardwired) στον τελικό αθροιστή Wallace του φίλτρου. Ο όρος αυτός προκύπτει από τον όρο με τους άσσους των ενδιάμεσων γινομένων (1010101100000000 για 8bit × 8bit πολλαπλασιασμό) με αριστερή ολίσθηση κατά $log_2(t)$ θέσεις. Στην παρούσα εργασία υλοποιούνται φίλτρα με $t$=16 και $t$=32, τα οποία απαιτούν αριστερή ολίσθηση του προηγούμενου όρου κατά τέσσερις και πέντε θέσεις αντίστοιχα.

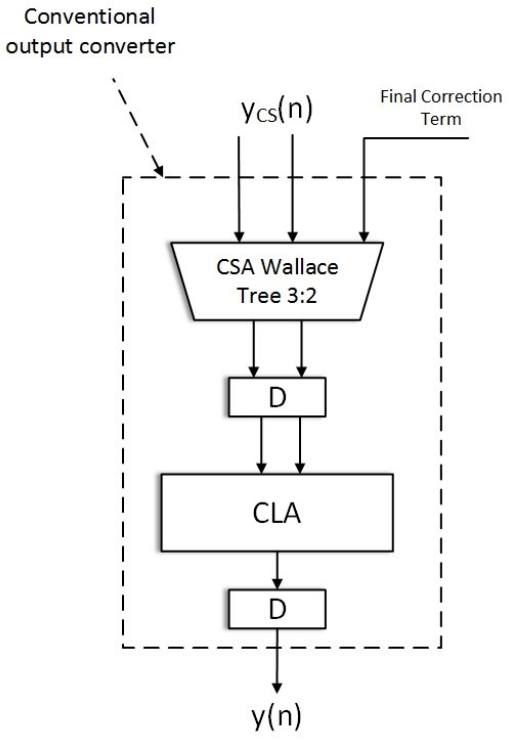
## 3.2 Υλοποιήσεις συμβατικών φίλτρων

Στην παρούσα ενότητα, παρουσιάζονται οι υλοποιήσεις των φίλτρων FIR στην κλασσική τους μορφή που εδώ αναφέρονται ως συμβατικά φίλτρα. Τα συμβατικά φίλτρα FIR υλοποιήθηκαν με σκοπό να αποτελέσουν μέτρο σύγκρισης για την αξιολόγηση της απόδοσης της νέας αρχιτεκτονικής Karatsuba που σχεδιάσαμε. Η αρχιτεκτονική του συμβατικού φίλτρου FIR απεικονίζεται στο σχήμα 3.7.




Σχήμα 3.7: Αρχιτεκτονική συμβατικού φίλτρου FIR

Η μονάδα Filter που απεικονίζεται στο σχήμα 3.7, αποτελεί το κυρίως κύκλωμα του φίλτρου το οποίο υλοποιήθηκε σε direct, transposed και mixed μορφή. Οι συγκεκριμένες μορφές παρουσιάζονται αναλυτικά στην συνέχεια. Το αποτέλεσμα της μονάδας Filter ($y_{CS}$) προκύπτει σε CS αναπαράσταση σε κάθε περίπτωση. Όμως, επιθυμούμε το αποτέλεσμα του φίλτρου να είναι σε μορφή συμπληρώματος ως προς δύο στην τελική του μορφή. Τον σκοπό αυτό εξυπηρετεί το υποκύκλωμα που απεικονίζεται στο σχήμα 3.8 (conventional output converter).



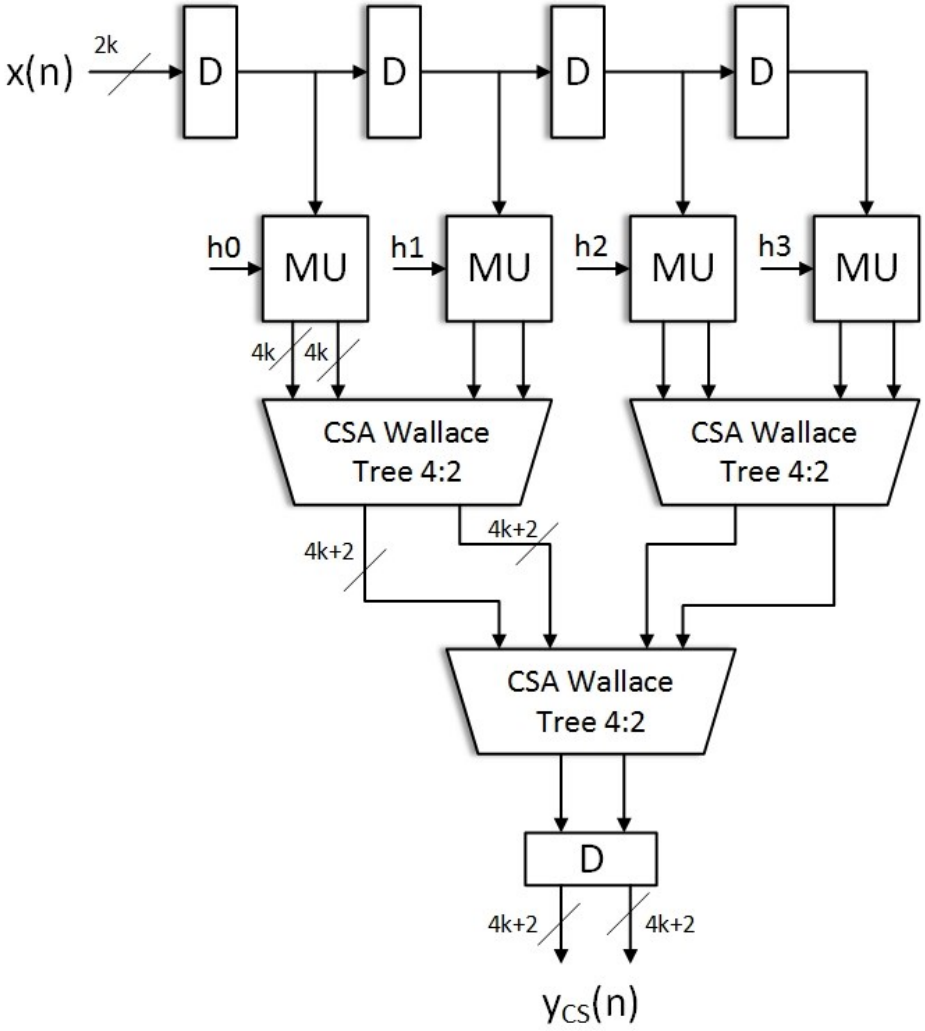

Σχήμα 3.8: Μετατροπέας εξόδου συμβατικού φίλτρου FIR

Στην πραγματικότητα, το υποκύκλωμα του μετατροπέα εξόδου εξυπηρετεί έναν διπλό σκοπό. Πρώτον, προσθέτει στο ενδιάμεσο αποτέλεσμα $y_{CS}(n)$ τον τελικό διορθωτικό όρο που περιέχει όλους τους άσσους από την επέκταση προσήμου (Final Correction Term). Ο τρόπος που προκύπτει ο τελικός διορθωτικός όρος παρουσιάστηκε στην ενότητα 3.1.2. Δεύτερον, μετατρέπει το CS διορθωμένο αποτέλεσμα σε αναπαράσταση συμπληρώματος ως προς δύο. Για τον σκοπό αυτό χρησιμοποιείται ένας γρήγορος αθροιστής πρόβλεψης κρατουμένου (CLA), όπως μπορούμε να δούμε στο σχήμα 3.8. Για την υλοποίηση της μονάδας CLA χρησιμοποιήθηκε το IP block *DW01_add* της *Synopsys DesignWare*.

Στην συνέχεια παρουσιάζονται οι τρείς μορφές (direct, transposed και mixed) στις οποίες υλοποιήθηκε η μονάδα Filter του σχήματος 3.7.

### 3.2.1 Direct μορφή

Η direct μορφή είναι η πιο απλή από τις τρείς μορφές των φίλτρων FIR που υλοποιήθηκαν στην παρούσα εργασία. Το κύκλωμα του φίλτρου σε direct μορφή, όπως και σε όλες τις άλλες μορφές περιγράφθηκε σε γλώσσα Verilog, με παραμετρικό κώδικα. Η παραμετρικότητα του κώδικα επέτρεψε την εύκολη υλοποίηση του φίλτρου για μήκος λέξης (δεδομένα εισόδου και συντελεστές φίλτρου) 16 και 32 bits, για 16 και 32 taps αντίστοιχα. Η αρχιτεκτονική της direct μορφής παρουσιάζεται στο σχήμα 3.9. Στο σχήμα διακρίνεται ένα φίλτρο που για ευκολία έχει σχεδιαστεί με $t$=4, ώστε να διακρίνονται και καλύτερα οι λεπτομέρειες της σχεδίασης του. Προφανώς, η ίδια ακριβώς αρχιτεκτονική επεκτείνεται και για 16 και 32 taps που αφορούν τις πραγματικές υλοποιήσεις του φίλτρου. Παρατηρούμε ότι τα αποτελέσματα των CSA Wallace tree έχουν μήκος λέξης $4k$+2 bits. Αυτό συμβαίνει διότι έχουμε λάβει υπόψιν την χειρότερη περίπτωση για το δυναμικό εύρος του φίλτρου, δηλαδή το μήκος λέξης των αποτελεσμάτων των πολλαπλασιαστών MU (ενδιάμεσα γινόμενα) αυξάνεται κατά $log_2(t)$ bits. Για αυτό και για το φίλτρο του σχήματος 3.9 με $t$=4, τα CSA Wallace tree έχουν όρους των $4k+log_2(4)=4k+2$ bits.

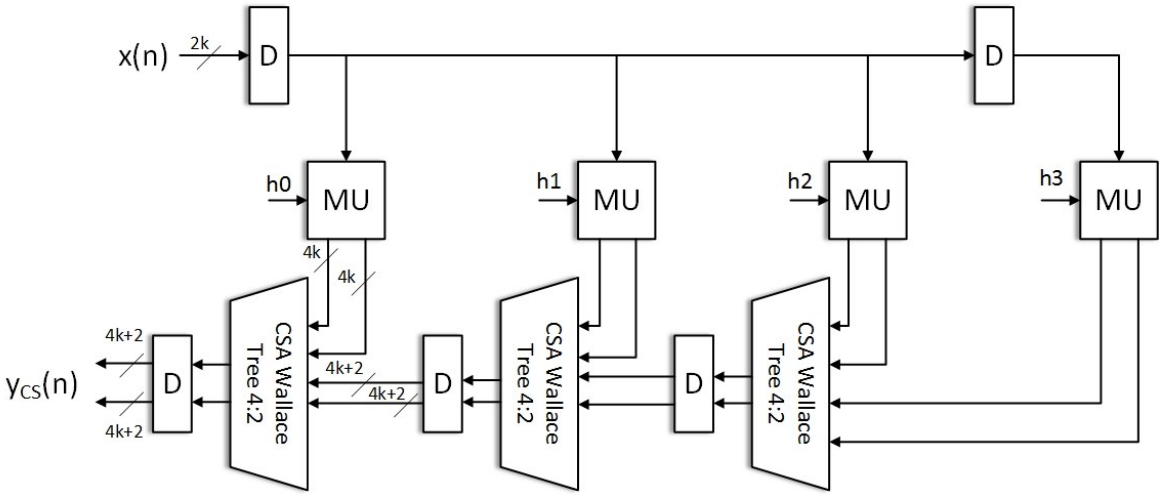



Σχήμα 3.9: Συμβατική υλοποίηση φίλτρου FIR σε direct μορφή με 4 taps

Στο κύκλωμα του σχήματος 3.9, χρησιμοποιείται ως δομική μονάδα ο πολλαπλασιαστής (MU) που παρουσιάστηκε στην ενότητα 3.1. Οι μονάδες που αναφέρονται με το γράμμα D αποτελούν τις μονάδες καθυστέρησης (D flip-flop). Θυμίζουμε ότι για τους δενδρικούς αθροιστές Wallace ή συμπιεστές Wallace χρησιμοποιήθηκε το IP block *DW02_tree* της *Synopsys DesignWare*.

### 3.2.2 Transposed μορφή

Στο σχήμα 3.10 παρουσιάζεται η transposed μορφή της συμβατικής υλοποίησης του φίλτρου FIR. Η αρχιτεκτονική του κυκλώματος δίνεται για ευκολία, όπως και προηγουμένως για $t$=4.

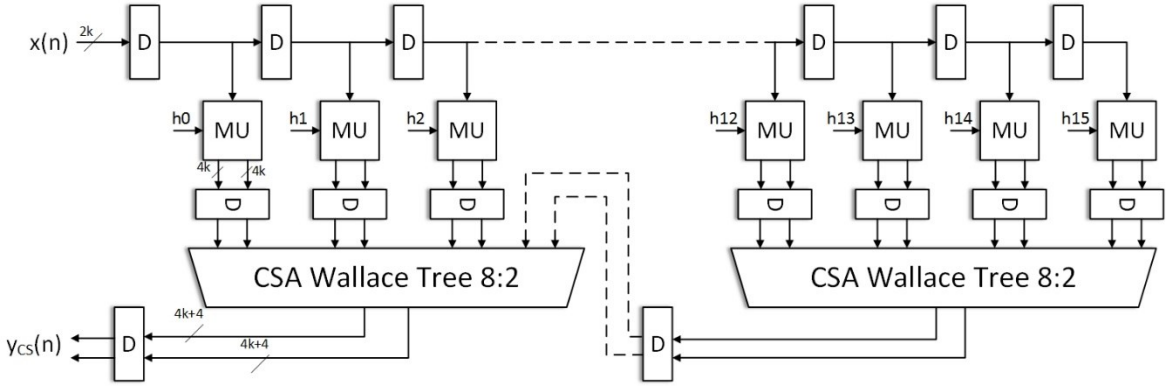



Σχήμα 3.10: Συμβατική υλοποίηση σε transposed μορφή με 4 taps

Οι μονάδες που χρησιμοποιούνται στο κύκλωμα του σχήματος 3.10 είναι ίδιες με τις αντίστοιχες μονάδες που χρησιμοποιούνται στην direct μορφή και αναφέρθηκαν στην προηγούμενη ενότητα.

### 3.2.3 Mixed μορφή

Η mixed μορφή του φίλτρου FIR αποτελεί μια ενδιάμεση μορφή μεταξύ της direct και της transposed μορφής, όπως είδαμε και στην θεωρία του πρώτου κεφαλαίου. Η συμβατική υλοποίηση του φίλτρου σε mixed μορφή παρουσιάζεται στο σχήμα 3.11.




Σχήμα 3.11: Συμβατική υλοποίηση φίλτρου FIR σε mixed μορφή για 16bits x 16taps

Στην παρούσα εργασία επιχειρήθηκε η αξιοποίηση της mixed μορφής ώστε να προκύψει ένα σχήμα με όσο το δυνατόν μικρότερη κρίσιμη καθυστέρηση. Πιο συγκεκριμένα, όπως βλέπουμε στο σχήμα 3.11, έχουν προστεθεί μονάδες καθυστέρησης στην έξοδο των μονάδων πολλαπλασιασμού. Με αυτόν τον τρόπο εφαρμόζεται η τεχνική της διοχέτευσης σε ένα κάθετο επίπεδο, διαχωρίζοντας το κύκλωμα σε δύο στάδια: το στάδιο πολλαπλασιασμού και το στάδιο πρόσθεσης. Η συγκεκριμένη επιλογή επιτρέπει την αύξηση της συχνότητας λειτουργίας του κυκλώματος, με κόστος την αύξηση της επιφάνειας λόγω των τοποθετούμενων μονάδων καθυστέρησης.

Η λειτουργία του κυκλώματος έχει ως εξής: στον ίδιο παλμό ρολογιού, υπολογίζονται τα ενδιάμεσα γινόμενα από τις μονάδες MU και αποθηκεύονται στις μονάδες D στην έξοδο τους, ενώ ταυτόχρονα αθροίζονται τα ενδιάμεσα γινόμενα που ήταν αποθηκευμένα από την προηγούμενη περίοδο. Η ομαδοποίηση των ενδιάμεσων γινομένων γίνεται με τρόπο ώστε οι καθυστερήσεις στα στάδια πολλαπλασιασμού και πρόσθεσης να είναι περίπου ίσες. Πιο συγκεκριμένα, το πλήθος των ενδιάμεσων γινομένων που αθροίζονται είναι αντίστοιχο του πλήθους των μερικών γινομένων μιας μονάδας πολλαπλασιαστή MU. Για μήκος λέξης $n=2k$ bits, μια μονάδα MU έχει $k$ μερικά γινόμενα όπως και έναν διορθωτικό όρο που περιλαμβάνει τα κρατούμενα εισόδου των μερικών γινομένων, δηλαδή συνολικά $k+1$ όρους. Στον πίνακα 3.1 δίνεται το μέγεθος των αθροιστών Wallace σε συνάρτηση με το μήκος λέξης των δεδομένων.

| **#bits (2*k*)** | **#PP (*k*+1)** | **Wallace Tree** |
|---|---|---|
| 16 | 9 | 8:2 |
| 32 | 17 | 16:2 |

Πίνακας 3.1: Μέγεθος των CSA Wallace Tree συναρτήσει του μήκους λέξης

Ένας συμπιεστής Wallace $N$:2 έχει ως εισόδους $N$ το πλήθος όρους και στην έξοδο του παράγει το άθροισμα τους σε CS αναπαράσταση. Όσον αφορά τους συμπιεστές Wallace της συγκεκριμένης υλοποίησης του φίλτρου, επιβάλλεται το πλήθος $N$ να είναι άρτιος αριθμός διότι τα αποτελέσματα των μονάδων MU προκύπτουν σε αναπαράσταση CS, δηλαδή αποτελούν δυάδες αριθμών. Επομένως το $N$ επιλέγεται ως ο αμέσως μικρότερος η ίσος του

πλήθους των μερικών γινομένων άρτιος αριθμός. Με αυτόν τον τρόπο, οι μονάδες MU και οι συμπιεστές Wallace διαχειρίζονται σχεδόν τον ίδιο αριθμό όρων με αποτέλεσμα να αποτελούνται και από τον ίδιο αριθμό επιπέδων από FA. Το γεγονός αυτό φαίνεται και στον πίνακα 3.2 που δίνει τον αριθμό των επιπέδων από FA που αντιστοιχεί στο πλήθος των όρων για ένα CSA Wallace tree.

| **Αριθμός προσθετέων όρων** | **Αριθμός επιπέδων** |
|---|---|
| 1,2,3 | 1 |
| 4 | 2 |
| 5,6 | 3 |
| 7,8,9 | 4 |
| 10-13 | 5 |
| 14-19 | 6 |

Πίνακας 3.2: Επίπεδα από FA για έναν δενδρικό αθροιστή Wallace

## 3.3 Υλοποιήσεις φίλτρων Karatsuba

Όπως είδαμε στο πρώτο κεφάλαιο, ο αλγόριθμος Karatsuba είναι ένας γρήγορος αλγόριθμος πολλαπλασιασμού. Ο αλγόριθμος Karatsuba, απλοποιεί έναν πολλαπλασιασμό $2k \times 2k$ σε μια σειρά από $(k+1) \times (k+1)$ και $k \times k$ πολλαπλασιασμούς και προσθέσεις, διαχωρίζοντας σε δύο ίσα τμήματα τους τελεστέους των $2k$ bits και παράγοντας ορισμένες βοηθητικές μεταβλητές.

Στο κεφάλαιο 2 είδαμε ότι εφαρμόζοντας τον αλγόριθμο Karatsuba καταλήγουμε στην ακόλουθη σχέση που εκφράζει έναν $2k \times 2k$ πολλαπλασιασμό:

$$P = x \cdot h = (P_H \cdot 2^k + P_L) \cdot (2^k + 1) - dP \cdot 2^k =$$
$$= (x_H \cdot h_H \cdot 2^k + x_L \cdot h_L) \cdot (2^k + 1) - dx \cdot dh \cdot 2^k \qquad (1)$$

Όπου:

$x = x_{2k-1}\, x_{2k-2}\, \ldots\, x_1\, x_0$

$x_H = x_{2k-1}\, x_{2k-2}\, \ldots\, x_{k+1}\, x_k$

$x_L = x_{k-1}\, x_{k-2}\, \ldots\, x_1\, x_0$

$dx = x_H - x_L$

Ομοίως και για τον πολλαπλασιαστή *h*.

Η έκφραση που περιγράφει την λειτουργία του φίλτρου είναι:

$$y(n) = \sum_{l=0}^{t-1} h(l) \cdot x(n-l) \quad (2)$$

Όπου με *t* συμβολίζεται το πλήθος των σημείων του φίλτρου, με *x* τα δεδομένα εισόδου και με *h* οι συντελεστές του φίλτρου. Εφαρμόζοντας την λογική της σχέσης (1) στην υλοποίηση ενός φίλτρου FIR, μπορούμε να διαχωρίσουμε το συνολικό κύκλωμα του φίλτρου σε τρία υποφίλτρα μικρότερου δυναμικού εύρους, τα οποία λειτουργούν παράλληλα. Έτσι, το τελικό αποτέλεσμα της σχέσης (2), δύναται να προκύψει με την σύνθεση των αποτελεσμάτων των τριών επιμέρους υποφίλτρων με την εφαρμογή λιγοστών πράξεων ολίσθησης και πρόσθεσης. Τα τρία υποφίλτρα περιγράφονται από τις ακόλουθες σχέσεις:

$$y_H(n) = \sum_{l=0}^{t-1} h_H(l) \cdot x_H(n-l) \quad (3)$$

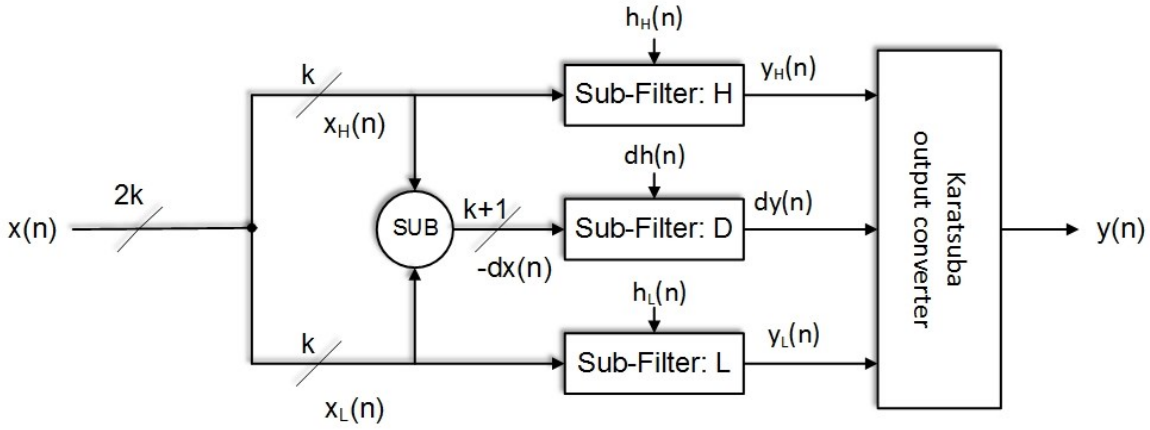
$$y_L(n) = \sum_{l=0}^{t-1} h_L(l) \cdot x_L(n-l) \quad (4)$$

$$dy(n) = \sum_{l=0}^{t-1} dh(l) \cdot dx(n-l) \quad (5)$$

Σύμφωνα με την σχέση (1) το τελικό αποτέλεσμα του φίλτρου μπορεί να προκύψει ως εξής:

$$y(n) = (y_H(n) \cdot 2^k + y_L(n)\,) \cdot (2^k + 1) - dy(n) \cdot 2^k \qquad (6)$$

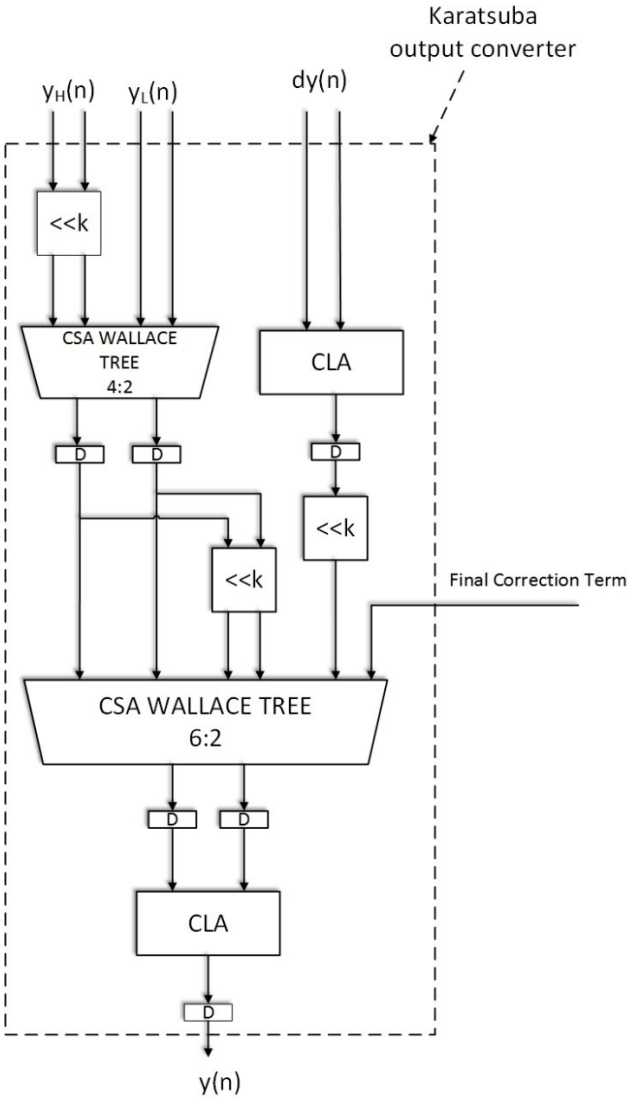
Στο σχήμα 3.12 παρουσιάζεται η αρχιτεκτονική των φίλτρου FIR που προκύπτει με την εφαρμογή του αλγορίθμου Karatsuba και βασίζεται στην σχέση (6). Παρατηρούμε, ότι τα υποφίλτρα *H, D* και *L* έχουν ως δεδομένα εισόδου τα $x_H$, *-dx* και $x_L$ αντίστοιχα. Για την παραγωγή του όρου $–dx = x_L\text{-}x_H$, απαιτείται ένας αφαιρέτης (SUB), που στην υλοποίηση μας χρησιμοποιήθηκε το IP block *DW01_sub* της *Synopsys DesignWare*. Τα τρία υποφίλτρα παράγουν τα αποτελέσματα τους ($y_H$, *dy* και $y_L$) σε CS αναπαράσταση.




Σχήμα 3.12: Αρχιτεκτονική φίλτρου FIR με εφαρμογή του αλγορίθμου Karatsuba

Ο μετατροπέας εξόδου (Karatsuba output converter) είναι μια μονάδα που χρησιμοποιείται με σκοπό την παραγωγή του τελικού αποτελέσματος σε μορφή συμπληρώματος ως προς δύο. Συγκεκριμένα εφαρμόζει μια σειρά ολισθήσεων και προσθέσεων στα αποτελέσματα των τριών υποφίλτρων όπως καθορίζεται από την σχέση (6). Το υποκύκλωμα του

μετατροπέα εξόδου φαίνεται στο σχήμα 3.13. Ομοίως με την περίπτωση του συμβατικού φίλτρου, ο τελικός διορθωτικός όρος περιλαμβάνει όλους τους άσσους από την επέκταση προσήμου των μερικών γινομένων. Στο σχήμα παρατηρούμε ότι το κύκλωμα αποτελείται από τρείς μονάδες ολίσθησης (hardwired shifters), δύο CLA αθροιστές, δύο CSA Wallace tree (4:2 και 6:2) και τις απαραίτητες μονάδες καθυστέρησης (D).




Σχήμα 3.13: Μετατροπέας εξόδου για την παραγωγή του τελικού αποτελέσματος του φίλτρου Karatsuba

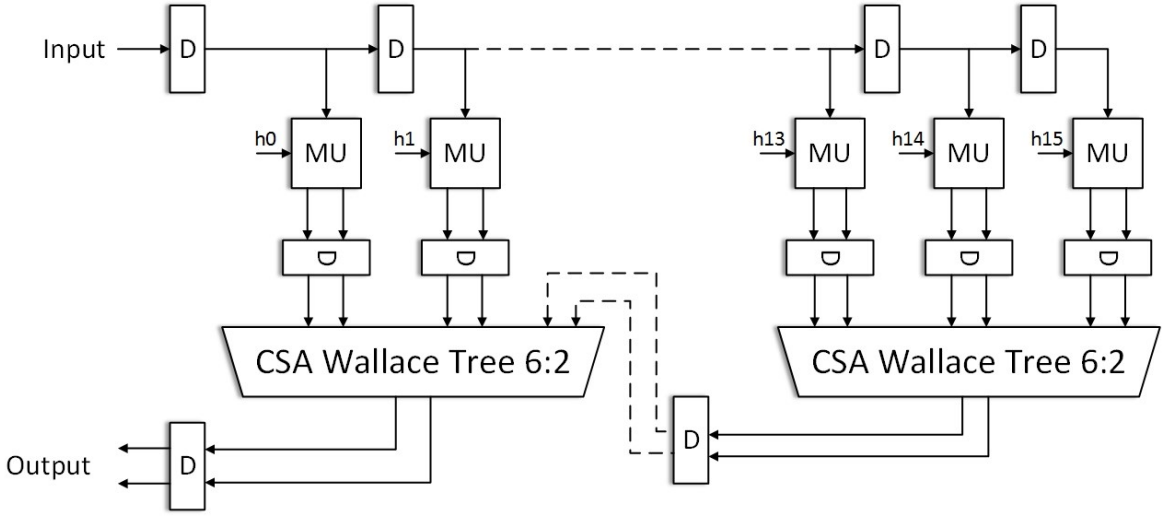
Η αρχιτεκτονική Karatsuba του σχήματος 3.12, υλοποιήθηκε σε τρείς μορφές: direct, transposed και mixed. Τα κυκλώματα των υποφίλτρων *H, D* και *L* είναι σε κάθε περίπτωση παρόμοια με τα κυκλώματα των φίλτρων που παρουσιάστηκαν στην προηγούμενη ενότητα. Πιο συγκεκριμένα, όσον αφορά την direct και την transposed μορφή, τα τρία υποφίλτρα έχουν ακριβώς την αρχιτεκτονική που φαίνεται στα σχήματα 3.9 και 3.10 αντίστοιχα.

Η mixed μορφή του φίλτρου Karatsuba, υλοποιήθηκε με την ίδια λογική όπως η αντίστοιχη συμβατική υλοποίηση. Όμως, στην περίπτωση του φίλτρου Karatsuba η ομαδοποίηση των ενδιάμεσων γινομένων είναι διαφορετική, διότι τα υποφίλτρα είναι μικρότερου δυναμικού εύρους. Συγκεκριμένα, το μέγιστο πλήθος των μερικών γινομένων είναι *k*/2+2 και αντιστοιχεί στο υποφίλτρο *D*. Το μέγεθος των αθροιστών Wallace δίνεται στον παρακάτω πίνακα.

| **#bits (2*k*)** | **#PP (*k*/2+2)** | **Wallace Tree** |
|---|---|---|
| 16 | 6 | 6:2 |
| 32 | 10 | 10:2 |

Πίνακας 3.3: Μέγεθος CSA Wallace Tree συναρτήσει του μήκους λέξης για την mixed μορφή του φίλτρου Karatsuba

Η αρχιτεκτονική των υποφίλτρων σε mixed μορφή, η οποία είναι παρόμοια με εκείνη του σχήματος 3.11 δίνεται παρακάτω:




Σχήμα 3.14: Αρχιτεκτονική της mixed μορφής των υποφίλτρων που απαρτίζουν το φίλτρο Karatsuba για $n$=16 bits και $t$=16 taps

# 4

# ΘΕΩΡΗΤΙΚΗ ΑΝΑΛΥΣΗ ΕΠΙΦΑΝΕΙΑΣ ΚΑΙ ΚΑΘΥΣΤΕΡΗΣΗΣ ΦΙΛΤΡΩΝ FIR

## 4.1 Εισαγωγή

Σκοπός του παρόντος κεφαλαίου, είναι ο θεωρητικός προσδιορισμός της κρίσιμης καθυστέρησης των φίλτρων FIR, καθώς και η παρουσίαση των υλικών πόρων που χρησιμοποιούν.

Τα κυκλώματα των φίλτρων σχεδιάστηκαν με τρόπο ώστε το κρίσιμο μονοπάτι να βρίσκεται σε κάθε περίπτωση στο κυρίως κύκλωμα του φίλτρου και όχι στους μετατροπείς εξόδου. Όσον αφορά την αρχιτεκτονική Karatsuba, είναι προφανές ότι σε κάθε περίπτωση το most critical block είναι το υποφίλτρο *D*, καθώς έχει το μεγαλύτερο δυναμικό εύρος μεταξύ των τριών υποφίλτρων που συνθέτουν το συνολικό φίλτρο.

Η αρχιτεκτονική Karatsuba συγκρίνεται με την συμβατική αρχιτεκτονική ως προς τους υλικούς πόρους που χρησιμοποιούν στην direct, την transposed και την mixed μορφή. Οι μετατροπείς εξόδου παραμένουν αμετάβλητοι σε κάθε μορφή, για αυτό οι μονάδες υλικού που χρησιμοποιούν παρουσιάζονται σε ξεχωριστή ενότητα.

Επίσης, η αρχιτεκτονική Karatsuba χρειάζεται μια μονάδα αφαιρέτη $SUB[k]$ για την παραγωγή του όρου $-dx = x_L - x_H$ όπως έχουμε ήδη αναφέρει, καθώς και μια μονάδα καθυστέρησης $D[k+1]$ για την αποθήκευση του.

Από τα παραπάνω συμπεραίνουμε ότι την μεγαλύτερη συμβολή στην απόδοση των δύο αρχιτεκτονικών ως προς την επιφάνεια έχει σε κάθε περίπτωση το κυρίως κύκλωμα των φίλτρων. Δηλαδή, η μονάδα filter για την συμβατική αρχιτεκτονική (σχήμα 3.7), και οι τρείς μονάδες sub-filters για την αρχιτεκτονική Karatsuba (σχήμα 3.12). Όπως θα δούμε, η αρχιτεκτονική Karatsuba χρειάζεται τους τριπλάσιους υλικούς πόρους σε σχέση με την

συμβατική αρχιτεκτονική. Παρόλα αυτά, οι μονάδες που χρησιμοποιεί η αρχιτεκτονική Karatsuba διαχειρίζονται δεδομένα σχεδόν του μισού μήκους λέξης. Στην συνέχεια παρουσιάζονται αναλυτικά οι μονάδες που χρησιμοποιούν οι δύο αρχιτεκτονικές σε κάθε μια από τις τρείς μορφές που υλοποιήθηκαν.

## 4.2 Μετατροπείς εξόδου

Όπως είδαμε στο προηγούμενο κεφάλαιο, οι μετατροπείς εξόδου είναι ειδικά υποκυκλώματα των φίλτρων που σκοπό έχουν την διόρθωση των ενδιάμεσων αποτελεσμάτων, καθώς και την παραγωγή του τελικού αποτελέσματος σε μορφή συμπληρώματος ως προς δύο. Ο μετατροπέας εξόδου της συμβατικής αρχιτεκτονικής παρουσιάστηκε στο σχήμα 3.8 ενώ της αρχιτεκτονικής Karatsuba στο σχήμα 3.13. Οι μετατροπείς εξόδου επιτελούν την ίδια ακριβώς λειτουργία σε όλες τις μορφές των φίλτρων και για αυτό παραμένουν τελείως αμετάβλητοι στην direct, την transposed και την mixed μορφή.

Από το σχήμα 3.8, παρατηρούμε ότι ο μετατροπέας εξόδου της συμβατικής αρχιτεκτονικής χρειάζεται $3 \cdot D[4k + log_2 t]$ καταχωρητές για την αποθήκευση των αποτελεσμάτων. Δύο από αυτούς απαιτούνται για την αποθήκευση του CS διορθωμένου αποτελέσματος και ένας για την αποθήκευση του τελικού αποτελέσματος σε μορφή συμπληρώματος ως προς δύο. Επίσης, χρησιμοποιεί έναν αθροιστή $CSA_{3:2}[4k + log_2 t]$ για την πρόσθεση του τελικού διορθωτικού όρου στο CS ενδιάμεσο αποτέλεσμα. Τέλος, απαιτείται η χρήση ενός γρήγορου αθροιστή πρόβλεψης κρατουμένου $CLA[4k + log_2 t]$ για την μετατροπή του διορθωμένου CS αποτελέσματος σε μορφή συμπληρώματος ως προς δύο.

Ομοίως από το σχήμα 3.13, προκύπτει ότι ο μετατροπέας εξόδου της αρχιτεκτονικής Karatsuba, χρειάζεται $2 \cdot D[3k + log_2 t] + D[2k + 2 + log_2 t] + 3 \cdot D[4k + log_2 t]$ συνολικά καταχωρητές. Επίσης, χρησιμοποίει τους εξής αθροιστές CSA Wallace tree: $CSA_{4:2}[3k + log_2 t] + CSA_{6:2}[4k + log_2 t]$. Ακόμη, απαιτείται η χρησιμοποίηση δύο αθροιστών πρόβλεψης κρατουμένου: $CLA[2k + 2 + log_2 t] + CLA[4k + log_2 t]$.

Στον πίνακα 4.1 παρουσιάζονται οι υλικοί πόροι που χρησιμοποιούν οι μετατροπείς εξόδου των δύο αρχιτεκτονικών.

| Hardware Resources | |
|---|---|
| **Registers (D)** | |
| **CONV** | $3 \cdot D[4k + log_2 t]$ |
| **KA** | $2 \cdot D[3k + log_2 t] + D[2k + 2 + log_2 t] + 3 \cdot D[4k + log_2 t]$ |
| **CSA Wallace Trees** | |
| **CONV** | $CSA_{3:2}[4k + log_2 t]$ |
| **KA** | $CSA_{4:2}[3k + log_2 t] + CSA_{6:2}[4k + log_2 t]$ |
| **Carry Lookahead adders** | |
| **CONV** | $CLA[4k + log_2 t]$ |
| **KA** | $CLA[2k + 2 + log_2 t] + CLA[4k + log_2 t]$ |

Πίνακας 4.1: Υλικοί πόροι των μετατροπέων εξόδου

## 4.3 Direct μορφή

### 4.3.1 Κρίσιμη καθυστέρηση

Στο κεφάλαιο 3, παρουσιάστηκε η direct μορφή της μονάδας filter (σχήμα 3.9). Το συγκεκριμένο κύκλωμα αποτελεί το κυρίως σώμα της συμβατικής αρχιτεκτονικής του φίλτρου FIR (σχήμα 3.7). Η αρχιτεκτονική του φίλτρου Karatsuba σε direct μορφή (σχήμα 3.12) αποτελείται από τρία υποφίλτρα μειωμένου δυναμικού εύρους, που έχουν ακριβώς την μορφή του σχήματος 3.9.

Στην τοπολογία του φίλτρου direct μορφής που αναφέρθηκε παραπάνω, παρατηρούμε ότι αρχικά παράγονται τα ενδιάμεσα γινόμενα από τις μονάδες MU. Στην συνέχεια, τα ενδιάμεσα γινόμενα εισάγονται σε ένα δίκτυο που αποτελείται από $log_2 t$ επίπεδα CSA Wallace tree 4:2 αθροιστών, με σκοπό την συμπίεση των *2t* το πλήθος όρων σε δύο τελικούς όρους (CS αναπαράσταση).

Εύκολα συμπεραίνουμε, ότι η κρίσιμη καθυστέρηση του συμβατικού φίλτρου είναι:

$$T_{CONV} = T_{MU}[2k] + log_2 t \cdot T_{CSA-4:2}[4k + log_2 t]$$

Όσον αφορά το φίλτρο Karatsuba, το υποφίλτρο $D$ αποτελεί το most critical block του. Επομένως, η κρίσιμη καθυστέρηση της direct μορφής του φίλτρου Karatsuba είναι:

$$T_{KA} = T_{MU}[k+1] + log_2 t \cdot T_{CSA-4:2}[2k+2+log_2 t]$$

Είναι φανερό ότι το φίλτρο Karatsuba έχει σαφώς μικρότερη κρίσιμη καθυστέρηση από το αντίστοιχο συμβατικό φίλτρο διότι τόσο οι μονάδες πολλαπλασιασμού MU, όσο και τα CSA Wallace tree διαχειρίζονται όρους με μικρότερο μήκος λέξης.

### 4.3.2 Επιφάνεια

Στην συνέχεια συγκρίνεται η direct μορφή του φίλτρου Karatsuba με την αντίστοιχη συμβατική υλοποίηση ως προς τους υλικούς πόρους που χρησιμοποιούν. Οι υλικοί πόροι που χρειάζονται οι μετατροπείς εξόδου παρουσιάστηκαν στην ενότητα 4.2.

Επίσης, όπως έχουμε ήδη αναφέρει η αρχιτεκτονική Karatsuba χρειάζεται επιπλέον έναν αφαιρέτη $SUB[k]$ και μια μονάδα καθυστέρησης $D[k+1]$ για την παραγωγή και την αποθήκευση του όρου $-dx = x_L - x_H$.

**Registers (D)**

Το συμβατικό φίλτρο έχει δεδομένα εισόδου με μήκος λέξης $2k$ bits. Μια υλοποίηση ενός φίλτρου FIR τάξης $t$, χρησιμοποιεί για το μονοπάτι των δεδομένων εισόδου $t \cdot D[2k]$ καταχωρητές. Επίσης, οι συντελεστές του φίλτρου είναι προενταμιευμένοι σε MB κωδικοποίηση (*one_h, two_h, sign_h*). Για την αποθήκευση τους απαιτούνται $3t \cdot D[k]$ καταχωρητές. Τέλος, όπως φαίνεται στο σχήμα 3.9, χρησιμοποιούνται ορισμένοι καταχωρητές στο κύκλωμα για την αποθήκευση του ενδιάμεσου αποτελέσματος $y_{CS}(n)$. Συγκεκριμένα, οι καταχωρητές που χρησιμοποιούνται είναι $2 \cdot D[4k + log_2 t]$, καθώς το ενδιάμεσο αποτέλεσμα προκύπτει σε CS αναπαράσταση. Συνολικά, η συμβατική υλοποίηση του φίλτρου σε direct μορφή χρειάζεται τους εξής καταχωρητές:

$$t \cdot D[2k] + 3t \cdot D[k] + 2 \cdot D[4k + log_2 t]$$

Το αντίστοιχο φίλτρο Karatsuba στην direct μορφή, συντίθεται από τρία υποφίλτρα ίδιας μορφής αλλά μικρότερου δυναμικού εύρους. Επομένως το πλήθος των καταχωρητών που χρησιμοποιεί για τα δεδομένα εισόδου και τους συντελεστές τριπλασιάζεται. Βέβαια, οι

καταχωρητές τώρα έχουν μικρότερο μήκος λέξης. Πιο συγκεκριμένα, τα υποφίλτρα $H$ και $L$ χρειάζονται για τα δεδομένα εισόδου και τους συντελεστές $t \cdot \left(D[k] + 3 \cdot D\left[\frac{k}{2}\right]\right)$ καταχωρητές το καθένα. Αντίστοιχα, το υποφίλτρο $D$ χρησιμοποιεί $t \cdot \left(D[k+1] + 3 \cdot D\left[\frac{k}{2} + 1\right]\right)$.

Ακόμη, είναι απαραίτητη η χρησιμοποίηση ορισμένων καταχωρητών για την αποθήκευση των επιμέρους αποτελεσμάτων των υποφίλτρων που προκύπτουν σε CS αναπαράσταση. Τα υποφίλτρα $H$ και $L$ έχουν $2 \cdot D[2k + log_2 t]$ καταχωρητές το καθένα, ενώ το υποφίλτρο $D$ χρησιμοποιεί $2 \cdot D[2k + 2 + log_2 t]$.

Επομένως το φίλτρο Karatsuba χρησιμοποιεί συνολικά τους εξής καταχωρητές:

$$t \cdot \left(2 \cdot D[k] + 6 \cdot D\left[\frac{k}{2}\right] + D[k+1] + 3 \cdot D\left[\frac{k}{2} + 1\right]\right) + 4 \cdot D[2k + log_2 t] + 2 \cdot D[2k + 2 + log_2 t]$$

**Multipliers (MU)**

Όπως έχει ήδη αναφερθεί, όλες οι υλοποιήσεις των φίλτρων χρησιμοποιούν ως δομική μονάδα πολλαπλασιασμού, τον παράλληλο πολλαπλασιαστή Wallace (MU) του σχήματος 3.1. Γενικά, για ένα φίλτρο FIR τάξης $t$, είναι απαραίτητη η χρησιμοποίηση $t$ το πλήθος πολλαπλασιαστών για την παραγωγή των ενδιάμεσων γινομένων. Στην προκείμενη περίπτωση, η συμβατική υλοποίηση χρησιμοποιεί συνολικά $t \cdot MU[2k]$.

Όσον αφορά το φίλτρο Karatsuba, τα υποφίλτρα $H$ και $L$ χρησιμοποιούν $t \cdot MU[k]$ το καθένα, ενώ το υποφίλτρο $D$ χρησιμοποιεί $t \cdot MU[k+1]$. Επομένως, το φίλτρο Karatsuba χρησιμοποιεί συνολικά $2t \cdot MU[k] + t \cdot MU[k+1]$.

**CSA Wallace trees**

Οι αθροιστές CSA Wallace χρησιμοποιούνται στις υλοποιήσεις των φίλτρων με σκοπό την πρόσθεση των ενδιάμεσων γινομένων που παράγονται από τις μονάδες πολλαπλασιασμού MU. Στην direct μορφή του φίλτρου FIR, τα ενδιάμεσα γινόμενα αθροίζονται ανά τέσσερα για αυτό και χρησιμοποιούνται 4:2 αθροιστές. Όπως είδαμε παραπάνω, για την πρόσθεση όλων των ενδιάμεσων γινομένων ενός direct φίλτρου απαιτούνται $log_2 t$ το πλήθος επίπεδα

από CSA Wallace tree 4:2. Αρχικά παράγονται από τους πολλαπλασιαστές (MU) 2*t* όροι, οι οποίοι υποδιπλασιάζονται στο κάθε επίπεδο από CSA Wallace tree 4:2 αθροιστές.

Έτσι, το συνολικό πλήθος των CSA Wallace tree που χρησιμοποιεί η συμβατική υλοποίηση του φίλτρου FIR στην direct μορφή εκφράζεται από την σειρά:

$$\sum_{n=0}^{log_2(t)} 2^n \cdot CSA_{4:2}[4k + log_2 t] = (2^{log_2 t+1} - 1) \cdot CSA_{4:2}[4k + log_2 t]$$

Το φίλτρο Karatsuba, αποτελείται από τρία υποφίλτρα ακριβώς ίδιας αρχιτεκτονικής με το συμβατικό φίλτρο. Έτσι, το πλήθος των μονάδων CSA Wallace tree του κάθε υποφίλτρου εκφράζεται από μια σειρά παρόμοια με αυτήν που είδαμε παραπάνω, με την διαφορά να είναι το μήκος λέξης των εισόδων των αθροιστών. Συγκεκριμένα οι μονάδες CSA Wallace tree των υποφίλτρων H και L διαχειρίζονται όρους των $2k + log_2 t$ bits, ενώ του υποφίλτρου *D* όρους των $2k + 2 + log_2 t$ bits.

Επομένως η υλοποίηση του φίλτρου Karatsuba σε direct μορφή χρησιμοποιεί συνολικά τις παρακάτω μονάδες αθροιστών Wallace:

$$(2^{log_2 t+1} - 1) \cdot (2 \cdot CSA_{4:2}[2k + log_2 t] + CSA_{4:2}[2k + 2 + log_2 t])$$

Στον πίνακα 4.2 παρουσιάζονται συγκεντρωτικά οι υλικοί πόροι που χρησιμοποιούνται από τις υλοποιήσεις των φίλτρων FIR σε direct μορφή.

| Hardware Resources | |
|---|---|
| **Registers (D)** | |
| **CONV** | $t \cdot D[2k] + 3t \cdot D[k] + 2 \cdot D[4k + log_2 t]$ |
| **KA** | $t \cdot \left(2 \cdot D[k] + 6 \cdot D\left[\frac{k}{2}\right] + D[k+1] + 3 \cdot D\left[\frac{k}{2} + 1\right]\right) + 4 \cdot D[2k + log_2 t] + 2 \cdot D[2k + 2 + log_2 t]$ |
| **Multiplication Units (MU)** | |
| **CONV** | $t \cdot MU[2k]$ |
| **KA** | $2t \cdot MU[k] + t \cdot MU[k+1]$ |
| **CSA Wallace Trees** | |
| **CONV** | $(2^{log_2 t+1} - 1) \cdot CSA_{4:2}[4k + log_2 t]$ |
| **KA** | $(2^{log_2 t+1} - 1) \cdot (2 \cdot CSA_{4:2}[2k + log_2 t] + CSA_{4:2}[2k + 2 + log_2 t])$ |

Πίνακας 4.2: Υλικοί πόροι των φίλτρων FIR σε direct μορφή

## 4.4 Transposed μορφή

### 4.4.1 Κρίσιμη καθυστέρηση

Η αρχιτεκτονική της transposed μορφής της μονάδας filter παρουσιάστηκε στο σχήμα 3.10. Από το σχήμα αυτό, εύκολα προκύπτει ότι η κρίσιμη καθυστέρηση της συμβατικής υλοποίησης είναι:

$$T_{CONV} = T_{MU}[2k] + T_{CSA-4:2}[4k + log_2 t]$$

Η αρχιτεκτονική του φίλτρου Karatsuba (σχήμα 3.12) στην transposed μορφή αποτελείται από τρία υποφίλτρα με την αρχιτεκτονική του σχήματος 3.10. Και σε αυτήν την περίπτωση το most critical block είναι το υποφίλτρο *D* με κρίσιμη καθυστέρηση:

$$T_{KA} = T_{MU}[k+1] + T_{CSA-4:2}[2k + 2 + log_2 t]$$

Είναι εμφανές το πλεονέκτημα του φίλτρου Karatsuba ως προς την καθυστέρηση, καθώς το κρίσιμο μονοπάτι του περιλαμβάνει τις ίδιες δομικές μονάδες με εκείνο της συμβατικής υλοποίησης, με την διαφορά ότι στην περίπτωση της αρχιτεκτονικής Karatsuba οι δομικές μονάδες διαχειρίζονται δεδομένα μικρότερου μήκους λέξης.

### 4.4.2 Επιφάνεια

Στην παρούσα υποενότητα, παρουσιάζονται οι υλικοί πόροι που χρησιμοποιούνται το φίλτρο Karatsuba στην transposed μορφή και συγκρίνονται με τους αντίστοιχους πόρους που χρησιμοποιεί το συμβατικό φίλτρο.

Οι υλικοί πόροι που χρειάζονται οι μετατροπείς εξόδου παρουσιάστηκαν στην ενότητα 4.2 και παραμένουν ίδιοι σε όλες τις μορφές. Επίσης, όπως έχουμε ήδη αναφέρει η αρχιτεκτονική Karatsuba χρειάζεται επιπλέον έναν αφαιρέτη $SUB[k]$ και μια μονάδα καθυστέρησης $D[k+1]$ για την παραγωγή και την αποθήκευση του όρου $-dx = x_L - x_H$.

**Registers (D)**

Η συμβατική υλοποίηση του φίλτρου χρησιμοποιεί δύο καταχωρητές για τα δεδομένα εισόδου, δηλαδή $2 \cdot D[2k]$ και $3t \cdot D[k]$ για τους συντελεστές. Επίσης, στην γραμμή των CSA Wallace tree 4:2 υπάρχουν $2(t-1) \cdot D[4k + log_2 t]$ μονάδες. Συνολικά, η συμβατική υλοποίηση του φίλτρου FIR σε transposed μορφή χρησιμοποιεί του εξής καταχωρητές:

$$2 \cdot (t-1) \cdot D[4k + log_2 t] + 2 \cdot D[2k] + 3t \cdot D[k]$$

Με παρόμοιο τρόπο προκύπτει και ο συνολικός αριθμός των καταχωρητών που χρησιμοποιεί η υλοποίηση του φίλτρου Karatsuba σε transposed μορφή που είναι:

$$2(t-1) \cdot (2 \cdot D[2k + log_2 t] + D[2k + 2 + log_2 t]) +$$

$$+2 \cdot \left(2 \cdot D[k] + 6t \cdot D\left[\frac{k}{2}\right] + D[k+1] + 3t \cdot D[\frac{k}{2} + 1]\right)$$

Όπου, οι μονάδες $D[2k + log_2 t]$, $D[k]$ και $D\left[\frac{k}{2}\right]$ αντιστοιχούν στα υποφίλτρα $H$ και $L$, ενώ οι μονάδες $D[2k + 2 + log_2 t]$, $D[k+1]$ και $D[\frac{k}{2} + 1]$ αντιστοιχούν στο υποφίλτρο $D$.

**Multipliers (MU)**

Όσον αφορά τις χρησιμοποιούμενες μονάδες πολλαπλασιασμού των υλοποιήσεων στην transposed μορφή, αυτές είναι ακριβώς ίδιες όπως και στην περίπτωση της direct μορφής. Δηλαδή, η συμβατική υλοποίηση έχει $t \cdot MU[2k]$, ενώ η υλοποίηση Karatsuba $2t \cdot MU[k] + t \cdot MU[k+1]$.

**CSA Wallace trees**

Στην transposed μορφή ενός φίλτρου FIR, τα ενδιάμεσα γινόμενα που παράγονται από τους πολλαπλασιαστές (MU), αθροίζονται σειριακά ανά τέσσερα από μονάδες 4:2 συμπιεστών.

Η συμβατική υλοποίηση του φίλτρου σε transposed μορφή χρησιμοποιεί $(t-1) \cdot CSA_{4:2}[4k + log_2 t]$ όπως μπορούμε να δούμε στο σχήμα 3.10.

Αντίστοιχα η υλοποίηση του φίλτρου Karatsuba χρειάζεται $(t-1) \cdot CSA_{4:2}[2k + log_2 t]$ για το κάθε ένα από τα υποφίλτρα *H* και *L*, ενώ χρειάζεται $(t-1) \cdot CSA_{4:2}[2k + 2 + log_2 t]$ για το υποφίλτρο *D*.

Στον πίνακα 4.3 παρατίθενται οι υλικοί πόροι που χρησιμοποιεί τόσο η συμβατική, όσο και η Karatsuba υλοποίηση στην transposed μορφή τους.

| Hardware Resources | |
|---|---|
| **Registers (D)** | |
| **CONV** | $2 \cdot (t-1) \cdot D[4k + log_2 t] + 2 \cdot D[2k] + 3t \cdot D[k]$ |
| **KA** | $2(t-1) \cdot (2 \cdot D[2k + log_2 t] + D[2k + 2 + log_2 t])$ $+2 \cdot \left(2 \cdot D[k] + 6t \cdot D\left[\frac{k}{2}\right] + D[k+1] + 3t \cdot D[\frac{k}{2} + 1]\right)$ |
| **Multiplication Units (MU)** | |
| **CONV** | $t \cdot MU[2k]$ |
| **KA** | $2t \cdot MU[k] + t \cdot MU[k+1]$ |
| **CSA Wallace Trees** | |
| **CONV** | $(t-1) \cdot CSA_{4:2}[4k + log_2 t]$ |
| **KA** | $(t-1) \cdot (2 \cdot CSA_{4:2}[2k + log_2 t] + CSA_{4:2}[2k + 2 + log_2 t])$ |

Πίνακας 4.3: Υλικοί πόροι των φίλτρων FIR σε transposed μορφή

## 4.5 Mixed μορφή

### 4.5.1 Κρίσιμη καθυστέρηση

Η mixed μορφή της μονάδας filter παρουσιάστηκε στο σχήμα 3.11. Η αρχιτεκτονική Karatsuba του σχήματος 3.12 αποτελείται από τρία υποφίλτρα σε mixed μορφή όπως αυτό του σχήματος 3.14. Για να μεγιστοποιήσουμε την συχνότητα λειτουργίας των φίλτρων FIR, τοποθετήσαμε μονάδες καθυστέρησης αμέσως μετά τους πολλαπλασιαστές (MU).

Εύκολα συμπεραίνουμε ότι το κρίσιμο μονοπάτι των υλοποιήσεων σε mixed μορφή βρίσκεται στο κύκλωμα των πολλαπλασιαστών. Συγκεκριμένα, η κρίσιμη καθυστέρηση του συμβατικού φίλτρου είναι $T_{CONV} = T_{MU}[2k]$. Ως γνωστόν, το most critical block του φίλτρου Karatsuba είναι το υποφίλτρο *D* με κρίσιμη καθυστέρηση $T_{KA} = T_{MU}[k+1]$.

### 4.5.2 Επιφάνεια

Στην συνέχεια γίνεται σύγκριση της mixed μορφής του φίλτρου Karatsuba με την αντίστοιχη συμβατική υλοποίηση του φίλτρου, ως προς τους υλικούς πόρους που αυτές χρησιμοποιούν.

Όπως και στις άλλες δύο μορφές των φίλτρων που αναλύθηκαν παραπάνω, οι μετατροπείς εξόδου χρειάζονται ακριβώς τους ίδιους υλικούς πόρους όπως παρουσιάστηκαν στην ενότητα 4.2.

Επίσης, όπως έχουμε ήδη αναφέρει η αρχιτεκτονική Karatsuba χρειάζεται επιπλέον έναν αφαιρέτη $SUB[k]$ και μια μονάδα καθυστέρησης $D[k+1]$ για την παραγωγή και την αποθήκευση του όρου $-dx = x_L - x_H$.

**Registers (D)**

Έχουμε ήδη αναφέρει, ότι τοποθετήσαμε μονάδες καθυστέρησης στην έξοδο των πολλαπλασιαστών για να μειώσουμε την κρίσιμη καθυστέρηση των φίλτρων. Για αυτόν τον λόγο, το συμβατικό φίλτρο χρησιμοποιεί $2t \cdot D[4k]$ μονάδες για την αποθήκευση των ενδιάμεσων γινομένων. Ως συνήθως, απαιτούνται $3t \cdot D[k]$ για τους συντελεστές του φίλτρου. Επίσης, στο μονοπάτι των δεδομένων εισόδου έχει $\left(t - \frac{2(t-1)}{k-2}\right) \cdot D[2k]$ μονάδες, ενώ στο μονοπάτι άθροισης των ενδιάμεσων γινομένων έχει $\frac{2(t-1)}{k-2} \cdot D[4k + log_2 t]$. Το προηγούμενο γεγονός, οφείλεται στον τρόπο που ομαδοποιούνται τα ενδιάμεσα γινόμενα για να προστεθούν και θα διαφανεί καλύτερα στην ανάλυση των μονάδων CSA Wallace tree.

Τα τρία υποφίλτρα που απαρτίζουν το φίλτρο Karatsuba σε mixed μορφή, έχουν αντίστοιχο πλήθος καταχωρητών, διαφορετικού μήκους λέξης φυσικά. Πιο συγκεκριμένα, τα υποφίλτρα *H* και *L,* έχουν $2t \cdot D[2k] + \left(t - \frac{2(t-1)}{\frac{k}{2}-2}\right) \cdot D[k] + 3t \cdot D[\frac{k}{2}] + \frac{2(t-1)}{\frac{k}{2}-2} \cdot D[2k + log_2 t]$ καταχωρητές συνολικά το καθένα. Ομοίως, το υποφίλτρο *D* του φίλτρου Karastuba έχει τις εξής μονάδες: $2t \cdot D[2k+2] + \left(t - \frac{2(t-1)}{\frac{k}{2}-2}\right) \cdot D[k+1] + 3t \cdot D[\frac{k}{2}+1] + \frac{2(t-1)}{\frac{k}{2}-2} \cdot D[2k+2+log_2 t]$.

**Multipliers (MU)**

Το πλήθος των μονάδων πολλαπλασιασμού παραμένει ίδιο σε όλες τις μορφές των υλοποιήσεων.

**CSA Wallace trees**

Όπως είδαμε, στις υλοποιήσεις mixed μορφής τοποθετήσαμε μονάδες καθυστέρησης στην έξοδο των πολλαπλασιαστών (MU) με σκοπό να μεγιστοποιήσουμε την συχνότητα λειτουργίας των φίλτρων. Με αυτόν τον τρόπο, στην ίδια περίοδο ρολογιού υπολογίζονται τα καινούργια ενδιάμεσα γινόμενα ενώ ταυτόχρονα αθροίζονται τα ενδιάμεσα γινόμενα που παρήχθησαν την αμέσως προηγούμενη περίοδο.

Για να εξισορροπηθούν οι καθυστερήσεις των δύο σταδίων του κυκλώματος, δηλαδή του σταδίου πολλαπλασιασμού και του σταδίου πρόσθεσης, τα ενδιάμεσα γινόμενα ομαδοποιούνται με κατάλληλο τρόπο. Πιο συγκεκριμένα, σε κάθε μονάδα πολλαπλασιασμού (MU) της συμβατικής υλοποίησης παράγονται $k$+1 το πλήθος μερικά γινόμενα τα οποία στην συνέχεια αθροίζονται στην δομική μονάδα CSA Wallace tree του πολλαπλασιαστή (MU). Η κάθε μονάδα MU λοιπόν παράγει ένα ενδιάμεσο γινόμενο που είναι το άθροισμα όλων των μερικών γινομένων του.

Στην συνέχεια όλα τα ενδιάμεσα γινόμενα πρέπει να προστεθούν για να προκύψει το τελικό αποτέλεσμα του φίλτρου. Για την πρόσθεση των ενδιάμεσων γινομένων επιλέγουμε την χρησιμοποίηση αθροιστών CSA Wallace tree $k$:2. Έτσι, τα CSA Wallace tree $k$:2 αποτελούνται από το ίδιο πλήθος επιπέδων από Full Adders με εκείνο των δομικών μονάδων CSA Wallace tree $k$+1:0 που αποτελούν τμήμα των πολλαπλασιαστών MU.

Σε κάθε φίλτρο παράγονται συνολικά $t$ το πλήθος ενδιάμεσα γινόμενα σε CS αναπαράσταση, δηλαδή συνολικά 2$t$ όροι. Όπως φαίνεται στο σχήμα 3.11 στο οποίο απεικονίζεται η συμβατική υλοποίηση του φίλτρου σε mixed μορφή, χρησιμοποιείται ένας $CSA_{k:2}[4k + log_2 t]$ για την πρόσθεση των k πρώτων όρων ξεκινώντας από την τελευταία χρονική βαθμίδα. Άρα, απομένουν 2$t$-$k$ όροι ακόμη να προστεθούν. Οι υπόλοιποι αθροιστές CSA Wallace tree $k$:2 προσθέτουν σε κάθε περίπτωση $k$-2 όρους των ενδιάμεσων γινόμενων και τους δύο όρους που προέρχονται από τον αθροιστή CSA της προηγούμενης χρονικής βαθμίδας. Επομένως, είναι απαραίτητη η χρησιμοποίηση $\left(\frac{2t-k}{k-2}\right) \cdot CSA_{k:2}[4k + log_2 t]$ μονάδων ακόμη. Άρα, η συμβατική υλοποίηση χρειάζεται συνολικά:

$$\left(\frac{2t-k}{k-2}+1\right)\cdot CSA_{k:2}[4k+log_2 t] = \left(\frac{2(t-1)}{k-2}\right)\cdot CSA_{k:2}[4k+log_2 t]$$

Όσον αφορά το φίλτρο Karatsuba, αυτό αποτελείται από τρία υποφίλτρα (*H, L* και *D*) της mixed μορφής του σχήματος 3.14. Το most critical block του φίλτρου Karatsuba αποτελεί το υποφίλτρο του *D*, και είναι αυτό που καθορίζει τις μονάδες αθροιστών CSA Wallace tree που θα χρησιμοποιηθούν για την πρόσθεση των ενδιάμεσων γινομένων. Οι μονάδες πολλαπλασιαστών του υποφίλτρου *D* παράγουν τώρα *k*/2+1 μερικά γινόμενα. Για αυτόν τον λόγο και στα τρία υποφίλτρα χρησιμοποιούνται CSA Wallace tree *k*/2:2 για την πρόσθεση των ενδιάμεσων γινομένων τους. Με παρόμοιο τρόπο, όπως και για την συμβατική υλοποίηση προκύπτει ότι τα υποφίλτρα *H* και *L* έχουν το καθένα:

$$\left(\frac{2(t-1)}{k/_2-2}\right)\cdot CSA_{\frac{k}{2}:2}[2k+log_2 t]$$

Αντίστοιχα, για το υποφίλτρο D προκύπτει ότι χρησιμοποιεί:

$$\left(\frac{2(t-1)}{k/_2-2}\right)\cdot CSA_{\frac{k}{2}:2}[2k+2+log_2 t]$$

Είναι πλέον ξεκάθαρος ο λόγος που οι καταχωρητές διαμοιράστηκαν μεταξύ μονοπατιού δεδομένων και αθροιστών CSA με τον τρόπο που είδαμε παραπάνω.

Στον πίνακα 4.4 δίνονται τα συγκεντρωτικά στοιχεία για τις μονάδες υλικού που χρησιμοποιούν οι υλοποιήσεις των φίλτρων FIR σε mixed μορφή.

| Hardware Resources | |
|---|---|
| **Registers (D)** | |
| **CONV** | $2t \cdot D[4k] + \left(t - \frac{2(t-1)}{k-2}\right) \cdot D[2k] + 3t \cdot D[k] + \frac{2(t-1)}{k-2} \cdot D[4k + log_2 t]$ |
| **KA** | $t \cdot \left(4 \cdot D[2k] + 6 \cdot D\left[\frac{k}{2}\right] + 2 \cdot D[2k+2] + 3 \cdot D\left[\frac{k}{2} + 1\right]\right)$ $+ \left(t - \frac{2(t-1)}{\frac{k}{2} - 2}\right) \cdot (2 \cdot D[k] + D[k+1])$ $+ \left(\frac{2(t-1)}{\frac{k}{2} - 2}\right) \cdot (2 \cdot D[2k + log_2 t] + D[2k + 2 + log_2 t])$ |
| **Multiplication Units (MU)** | |
| **CONV** | $t \cdot MU[2k]$ |
| **KA** | $2t \cdot MU[k] + t \cdot MU[k+1]$ |
| **CSA Wallace Trees** | |
| **CONV** | $\left(\frac{2(t-1)}{k-2}\right) \cdot CSA_{k:2}[4k + log_2 t]$ |
| **KA** | $\left(\frac{2(t-1)}{k/2-2}\right) \cdot (2 \cdot CSA_{\frac{k}{2}:2}[2k + log_2 t] + CSA_{\frac{k}{2}:2}[2k + 2 + log_2 t])$ |

Πίνακας 4.4: Υλικοί πόροι των φίλτρων FIR σε mixed μορφή

## 4.6 Συμπεράσματα

Με βάση την θεωρητική ανάλυση που προηγήθηκε στις προηγούμενες υποενότητες του παρόντος κεφαλαίου, μπορούμε να συναγάγουμε τα εξής συμπεράσματα:

- Το φίλτρο Karatsuba έχει μικρότερη κρίσιμη καθυστέρηση από το συμβατικό φίλτρο για κάθε μορφή (direct, transposed και mixed).
- Οι τρείς μορφές των φίλτρων ταξινομούνται με σειρά από την ταχύτερη προς την βραδύτερη ως εξής: mixed, transposed, direct.
- Το πλήθος των μονάδων πολλαπλασιασμού (MU) παραμένει σταθερό μεταξύ των διαφορετικών μορφών, τόσο για τις υλοποιήσεις Karatsuba όσο και για τις συμβατικές υλοποιήσεις. Επομένως αυτό που καθορίζει τις διαφορές ως προς την επιφάνεια είναι οι αθροιστές CSA Wallace tree και οι καταχωρητές σε κάθε περίπτωση.
- Η direct μορφή παρουσιάζει το μεγαλύτερο πλήθος αθροιστών CSA Wallace tree με την transposed μορφή να ακολουθεί και την mixed μορφή να έρχεται τελευταία. Βέβαια, η mixed μορφή αποτελείται από CSA Wallace tree με μεγαλύτερο πλήθος όρων.
- Η mixed μορφή των φίλτρων FIR υποφέρει από την χρησιμοποίηση μεγάλου αριθμού καταχωρητών εξαιτίας των επιπρόσθετων μονάδων καθυστέρησης που βρίσκονται στην έξοδο των πολλαπλασιαστών (MU). Η direct και η transposed μορφή χρησιμοποιούν παρόμοιο πλήθος καταχωρητών. Όμως, οι καταχωρητές που χρησιμοποιούνται στην transposed μορφή έχουν μεγαλύτερο μήκος λέξης καθώς βρίσκονται στο μονοπάτι άθροισης των CSA Wallace tree, ενώ οι καταχωρητές της direct μορφής στο μονοπάτι των δεδομένων εισόδου.

# 5

# ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

## 5.1 Οργάνωση πειραμάτων

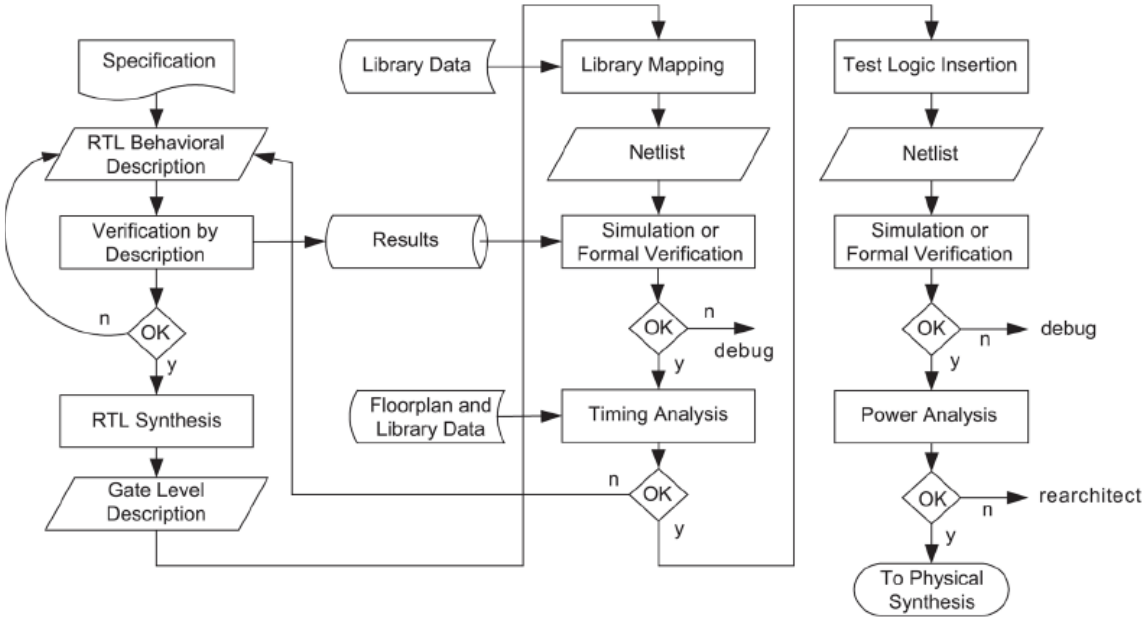
Η διαδικασία που ακολουθήθηκε στην παρούσα εργασία μέχρι να καταλήξουμε στην σύνθεση των κυκλωμάτων και την εξαγωγή των αποτελεσμάτων είναι η βασική ροή σχεδίασης συστημάτων ASIC. Αρχικά, περιγράφηκαν σε Verilog HDL όλα τα κυκλώματα με παραμετρικό κώδικα. Ο κώδικας είχε δύο παραμέτρους, τον πλήθος των taps και το πλήθος των bits των δεδομένων εισόδου και των συντελεστών. Έτσι κατέστη δυνατή η παραγωγή μίας οικογένειας φίλτρων FIR για διαφορετικό αριθμό taps και bits.

Στην συνέχεια πραγματοποιήθηκε προσομοίωση της ορθής λειτουργίας όλων των κυκλωμάτων (functional simulation) στο εργαλείο *ModelSim* για 20480 τυχαίες τιμές των δεδομένων εισόδου και σταθερές τιμές για τους συντελεστές του φίλτρου.

Αφού επαληθεύτηκε η ορθή λειτουργία όλων των κυκλωμάτων, στο επόμενο βήμα πραγματοποιήθηκε η σύνθεση των κυκλωμάτων. Η σύνθεση έγινε με το εργαλείο *Design Compiler* της *Synopsys*, με χρήση της βιβλιοθήκης κελιών Faraday 90nm της *Artisan*. Η σύνθεση πραγματοποιήθηκε με χρήση της εντολής –compile_ultra σε κάθε περίπτωση για βέλτιστα αποτελέσματα.

Η ανάλυση χρονισμού (Static Time Analysis-STA) των κυκλωμάτων πραγματοποιήθηκε με το εργαλείο *Synopsys PrimeTime*. Σε κάθε περίπτωση προσδιορίστηκε η ελάχιστη περίοδος ρολογιού που ήταν δυνατόν να προσομοιωθούν τα κυκλώματα, μεταβάλλοντας την τιμή της περιόδου κατά 0.01 *ns*. Από τον *Design Compiler* προέκυψαν τα αποτελέσματα για την επιφάνεια στην αντίστοιχη περίοδο ρολογιού. Μετά την σύνθεση απαιτείται ξανά προσομοίωση των κυκλωμάτων ώστε να επαληθευτεί ότι η σύνθεση πραγματοποιήθηκε σωστά (post-synthesis simulation). Η προσομοίωση έγινε ξανά στο *ModelSim* με τις ίδιες

τυχαίες τιμές δεδομένων εισόδου και συντελεστών. Από την προσομοίωση προκύπτει ένα αρχείο (επέκτασης .vcd) το οποίο χρησιμοποιεί το εργαλείο *PrimePower* της *Synopsys* για την εξαγωγή της μέσης δυναμικής κατανάλωσης των κυκλωμάτων.




Σχήμα 5.1: Ροή σχεδίασης συστημάτων σε τεχνολογία ASIC.

## 5.2 Συγκριτικά αποτελέσματα

Στην ενότητα αυτή, παρουσιάζονται τα αποτελέσματα από την σύνθεση των κυκλωμάτων. Τα κυκλώματα των φίλτρων FIR υλοποιήθηκαν για 16 και 32 taps, με δεδομένα εισόδου και συντελεστές των 16 και 32 bits.

Όλες οι υλοποιήσεις συγκρίθηκαν ως προς την ελάχιστη καθυστέρηση, την επιφάνεια και την κατανάλωση τους. Επίσης συγκρίθηκε η επίδοση τους με βάση τα κριτήρια *area·delay* ($A \cdot T$), *area·delay*$^2$ ($A \cdot T^2$) και *power·delay* ($P \cdot T$). Όλες οι μετρήσεις πραγματοποιήθηκαν στην high performance συχνότητα δηλαδή στην μέγιστη συχνότητα που ήταν δυνατό να προσομοιωθεί το εκάστοτε κύκλωμα.

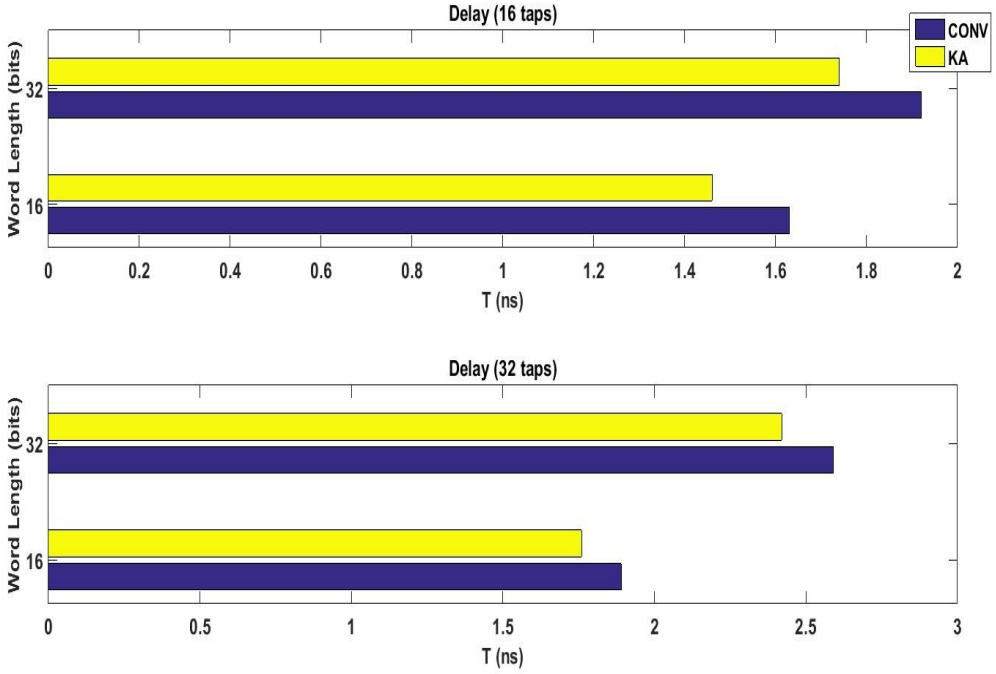
Για να καταστεί δυνατή η απευθείας σύγκριση των υλοποιήσεων Karatsuba με τις συμβατικές υλοποιήσεις ως προς την επιφάνεια και την μέση δυναμική κατανάλωση στην ίδια συχνότητα λειτουργίας, οι υλοποιήσεις Karatsuba προσομοιώθηκαν επίσης στην ελάχιστη περίοδο ρολογιού των συμβατικών υλοποιήσεων.

### 5.2.1 Direct μορφή

Στον πίνακα 5.1 και το σχήμα 5.2 παρουσιάζονται τα αποτελέσματα για την ελάχιστη καθυστέρηση των υλοποιήσεων των φίλτρων FIR σε direct μορφή. Όπως αναμενόταν, το φίλτρο Karatsuba έχει μικρότερη καθυστέρηση σε κάθε περίπτωση.

| ***Delay* Conventional (nsec)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 1.63 | 1.89 |
| **32** | 1.92 | 2.59 |
| ***Delay* Karatsuba (nsec)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 1.46 | 1.76 |
| **32** | 1.74 | 2.42 |

Πίνακας 5.1: Αποτελέσματα ελάχιστης καθυστέρησης για direct FIR φίλτρα

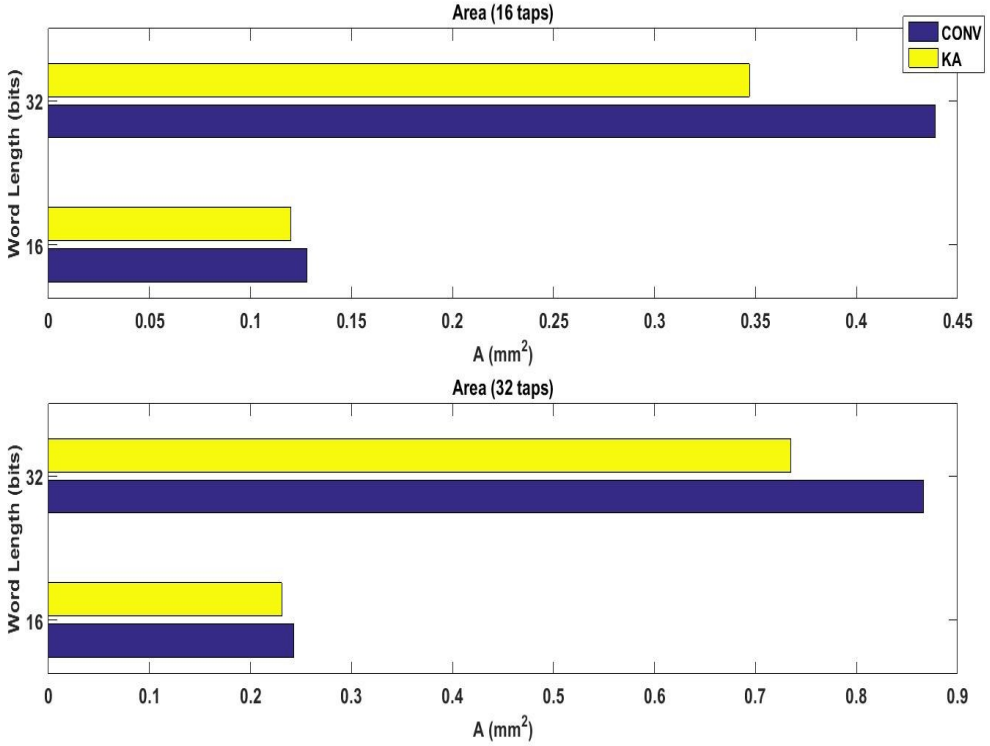



Σχήμα 5.2: Ελάχιστη καθυστέρηση υλοποιήσεων direct μορφής

Στον πίνακα 5.2 και το σχήμα 5.3 παρουσιάζονται τα αποτελέσματα για την επιφάνεια των υλοποιήσεων των φίλτρων FIR σε direct μορφή. Τα συγκεκριμένα αποτελέσματα για την επιφάνεια, προέκυψαν από την σύνθεση των κυκλωμάτων για περίοδο ρολογιού ίση με την ελάχιστη καθυστέρηση της συμβατικής υλοποίησης ($T$=$T_{\text{HP-CONV}}$). Όπως είδαμε στον πίνακα 5.1, η συμβατική υλοποίηση σε direct μορφή, είναι πιο αργή από την αντίστοιχη υλοποίηση Karatsuba και για αυτό δεν θα ήταν δυνατόν το συμβατικό φίλτρο FIR να προσομοιωθεί για περίοδο ρολογιού ίση με την ελάχιστη καθυστέρηση της Karatsuba υλοποίησης. Αντίθετα, το φίλτρο Karatsuba μπορεί να προσομοιωθεί σε μια χαλαρότερη περίοδο ρολογιού όπως η ελάχιστη καθυστέρηση της συμβατικής υλοποίησης. Με αυτόν τον τρόπο κατέστη δυνατή η απευθείας σύγκριση των δύο υλοποιήσεων ως προς την επιφάνεια στην ίδια συχνότητα λειτουργίας.

| ***Area* Conventional ($mm^2$)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.128 | 0.243 |
| **32** | 0.439 | 0.866 |
| ***Area* Karatsuba ($mm^2$)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.120 | 0.231 |
| **32** | 0.347 | 0.735 |

Πίνακας 5.2: Αποτελέσματα επιφάνειας για $T$=$T_{\text{HP-CONV}}$, για direct FIR φίλτρα

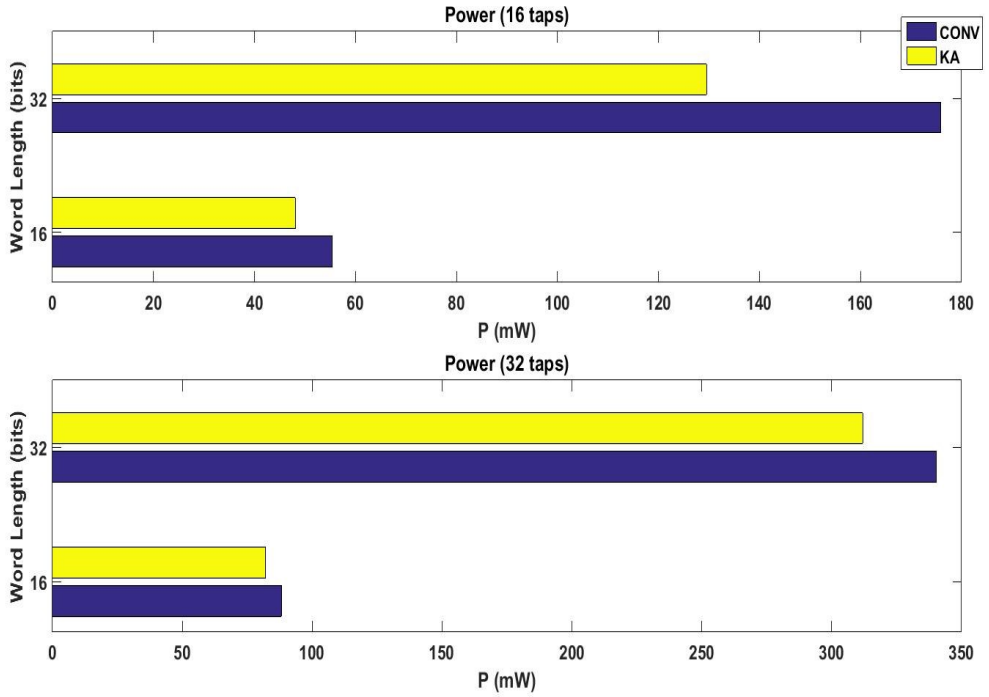



Σχήμα 5.3: Επιφάνεια υλοποιήσεων direct μορφής για $T=T_{HP\text{-}CONV}$

Τα αποτελέσματα για την μέση κατανάλωση των υλοποιήσεων των φίλτρων FIR σε direct μορφή δίνονται στον πίνακα 5.3 και το σχήμα 5.4. Όπως και τα αποτελέσματα για την επιφάνεια, τα αποτελέσματα για την κατανάλωση των φίλτρων αντιστοιχούν σε λειτουργία για περίοδο ρολογιού ίση με την ελάχιστη καθυστέρηση της συμβατικής υλοποίησης.

| ***Power* Conventional (mW)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 55.4 | 88.1 |
| **32** | 175.8 | 340.4 |
| ***Power* Karatsuba (mW)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 48.1 | 82.1 |
| **32** | 129.5 | 311.9 |

Πίνακας 5.3: Αποτελέσματα μέσης κατανάλωσης για $T=T_{HP\text{-}CONV}$, για direct FIR φίλτρα



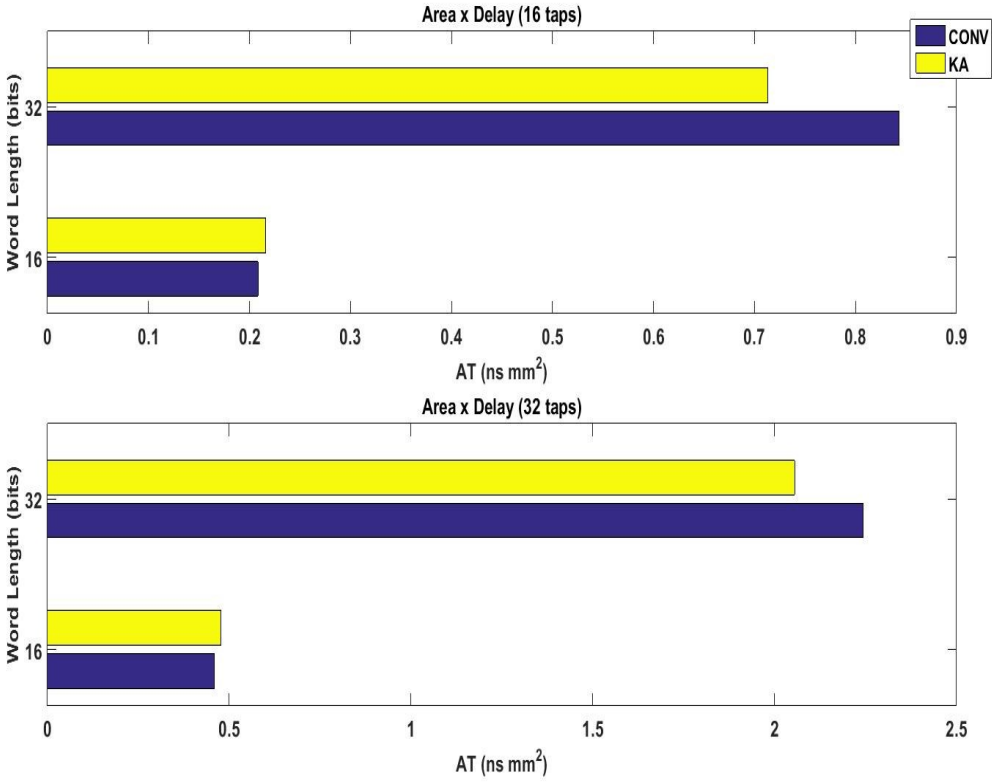

Σχήμα 5.4: Μέση κατανάλωση υλοποιήσεων direct μορφής για $T=T_{\text{HP-CONV}}$

Στους πίνακες 5.4, 5.5 και 5.6 δίνονται τα συγκριτικά αποτελέσματα για τις μετρικές *Area×Delay, Area×Delay*$^2$ *και Power×Delay* των φίλτρων FIR direct μορφής. Τα αποτελέσματα αυτά απεικονίζονται γραφικά στα σχήματα 5.5, 5.6 και 5.7 αντίστοιχα.

Όσον αφορά τις τιμές της επιφάνειας και της κατανάλωσης, αυτές αντιστοιχούν σε λειτουργία στην ελάχιστη περίοδο ρολογιού που μπορούσε να προσομοιωθεί η κάθε υλοποίηση (πίνακας 5.1).

| ***Area×Delay* Conventional (nsec·mm$^2$)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.209 | 0.459 |
| **32** | 0.843 | 2.243 |
| ***Area×Delay* Karatsuba (nsec·mm$^2$)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.216 | 0.477 |
| **32** | 0.713 | 2.055 |

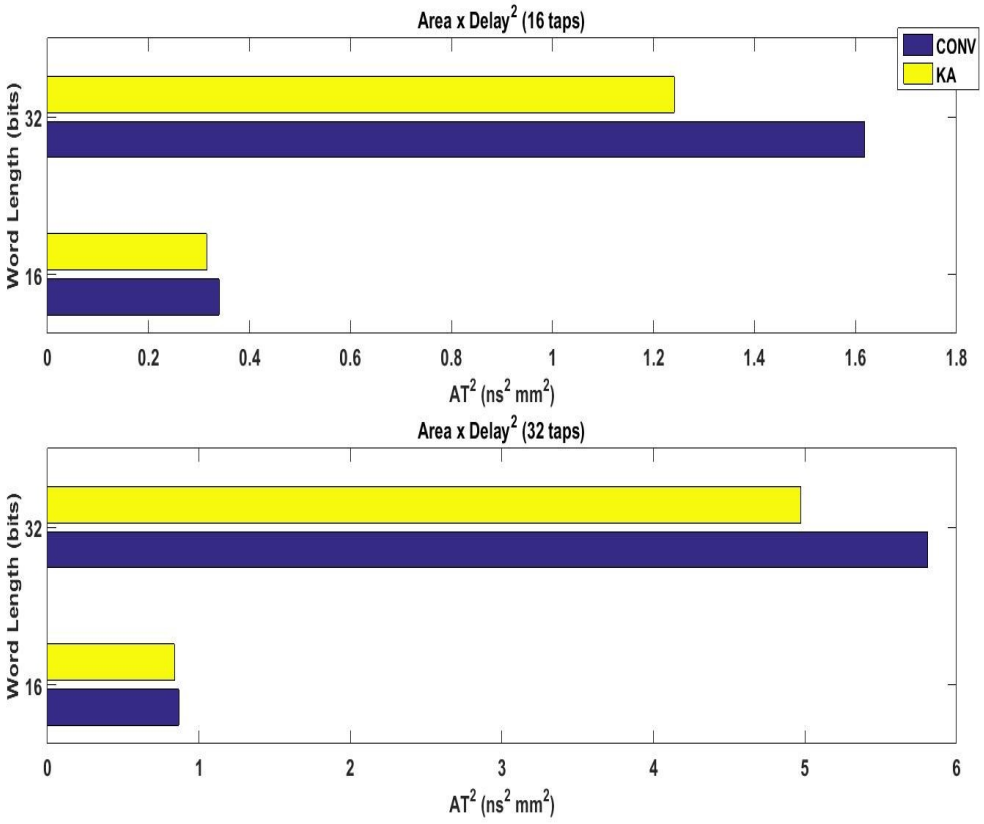
Πίνακας 5.4: Αποτελέσματα μετρικής *Area×Delay* για direct FIR φίλτρα




Σχήμα 5.5: *Area×Delay* υλοποιήσεων direct μορφής

| ***Area×Delay*² Conventional (nsec²·mm²)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.340 | 0.868 |
| **32** | 1.618 | 5.809 |
| ***Area×Delay*² Karatsuba (nsec²·mm²)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.316 | 0.839 |
| **32** | 1.241 | 4.972 |

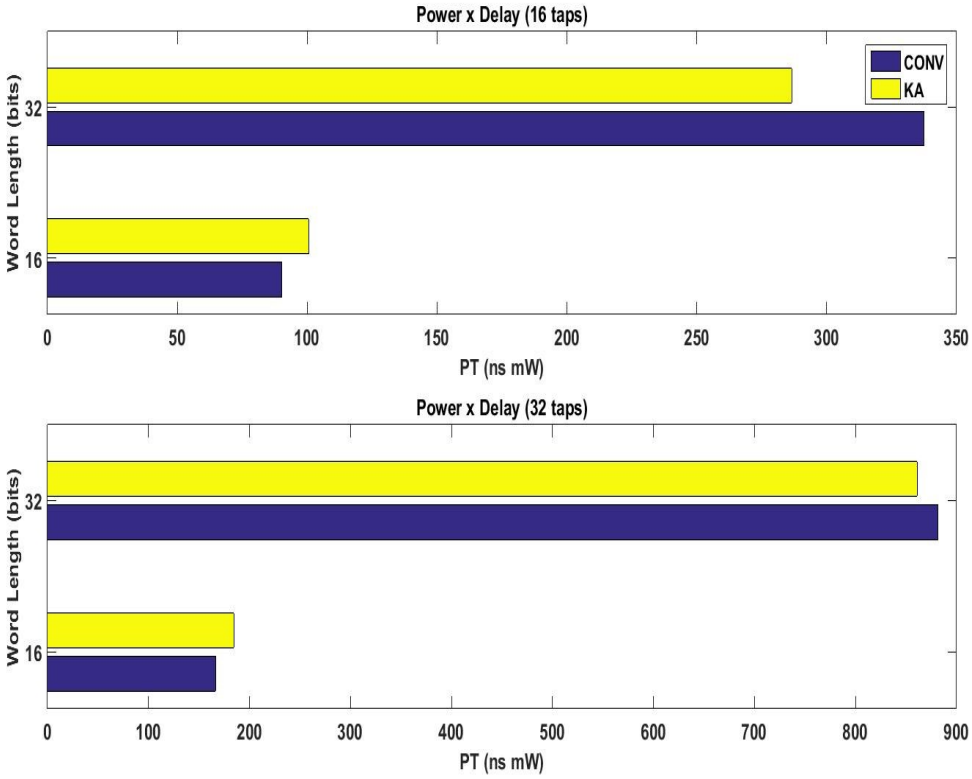
Πίνακας 5.5: Αποτελέσματα μετρικής *Area×Delay*² για direct FIR φίλτρα




Σχήμα 5.6: *Area*×*Delay*$^2$ υλοποιήσεων direct μορφής

| ***Power*×*Delay* Conventional (nsec·mW)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 90.30 | 166.51 |
| **32** | 337.54 | 881.64 |
| ***Power*×*Delay* Karatsuba (nsec·mW)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 100.59 | 184.45 |
| **32** | 286.58 | 861.04 |

Πίνακας 5.6: Αποτελέσματα μετρικής *Power*×*Delay* για direct FIR φίλτρα




Σχήμα 5.7: *Power×Delay* υλοποιήσεων direct μορφής

Στην συνέχεια τα δεδομένα των αποτελεσμάτων που παρουσιάστηκαν παραπάνω δίνονται σε ποσοστιαία μορφή, με σκοπό την καλύτερη σύγκριση των δυο υλοποιήσεων. Συγκεκριμένα, στους πίνακες 5.7 έως 5.12 δίνονται οι ποσοστιαίες διαφορές των αποτελεσμάτων.

| ***Delay* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -10.43 | -6.88 |
| **32** | -9.38 | -6.56 |

Πίνακας 5.7: Ποσοστιαία διαφορά ελάχιστης καθυστέρησης για direct FIR φίλτρα

Στον πίνακα 5.7, βλέπουμε ότι υπάρχει σημαντική μείωση της καθυστέρησης στην Karatsuba αρχιτεκτονική για όλα τα μήκη λέξης. Παρατηρούμε ότι το κέρδος είναι μεγαλύτερο για $t$=16.

| ***Area* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -6.25 | -4.94 |
| **32** | -20.96 | -15.13 |

Πίνακας 5.8: Ποσοστιαία διαφορά επιφάνειας για direct FIR φίλτρα, στην ελάχιστη περίοδο ρολογιού των συμβατικών υλοποιήσεων ($T$=$T_{\text{HP-CONV}}$)

Στον πίνακα 5.8, παρατηρούμε ότι το κέρδος της αρχιτεκτονικής Karatsuba σε επιφάνεια αυξάνεται κατακόρυφα από τα 16 στα 32 bits. Ο λόγος είναι, όπως έχουμε ήδη αναφέρει ότι η αποδοτικότητα του αλγορίθμου Karatsuba αυξάνεται με την αύξηση του μήκους λέξης των τελεστέων.

Επίσης, παρατηρείται μια τάση μείωσης του κέρδους με την αύξηση του αριθμού των taps του φίλτρου καθώς το κέρδος για $t$=32 είναι μικρότερο από ότι για $t$=16. Αυτό οφείλεται στο ότι η αρχιτεκτονική Karatsuba υποφέρει από την χρησιμοποίηση μεγάλου αριθμού καταχωρητών, γεγονός που επιδεινώνεται με την αύξηση των σημείων του φίλτρου.

| ***Power* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -13.18 | -6.81 |
| **32** | -26.34 | -8.37 |

Πίνακας 5.9: Ποσοστιαία διαφορά μέσης κατανάλωσης για direct FIR φίλτρα, στην ελάχιστη περίοδο ρολογιού των συμβατικών υλοποιήσεων (T=$T_{\text{HP-CONV}}$)

Σε συμφωνία με τα αποτελέσματα για την επιφάνεια, στον πίνακα 5.9 παρατηρούμε ότι το κέρδος σε κατανάλωση που προσφέρει η αρχιτεκτονική Karatsuba αυξάνεται με την αύξηση του μήκους λέξης ενώ μειώνεται με την αύξηση του πλήθους των taps.

| ***Area*×*Delay* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | +3.57 | +3.85 |
| **32** | -15.36 | -8.40 |

Πίνακας 5.10: Ποσοστιαία διαφορά μετρικής A·T για direct FIR φίλτρα

Ο πίνακας 5.10 αναφέρεται στην μετρική *Area×Delay*. Βλέπουμε ότι για *n*=16 bits η διαφορά είναι θετική, δηλαδή το φίλτρο Karatsuba είναι λιγότερο αποδοτικό από ότι το συμβατικό. Το γεγονός αυτό φαίνεται αντιφατικό αν θυμηθούμε τα αποτελέσματα του πίνακα 5.2 για την επιφάνεια, από τα οποία προέκυψε ότι το φίλτρο Karatsuba έχει μικρότερη επιφάνεια σε σχέση με το συμβατικό. Η διαφορά έγκειται στο γεγονός ότι τα αποτελέσματα του πίνακα 5.2 αναφέρονται στην ίδια περίοδο ρολογιού ($T_{\text{HP-CONV}}$) ενώ τα αποτελέσματα του πίνακα 5.10 στην high performance περίοδο της εκάστοτε υλοποίησης. Προφανώς η επιφάνεια δεν αυξάνεται γραμμικά με την αύξηση της συχνότητας λειτουργίας και για αυτό στην high performance περίοδο του φίλτρου Karatsuba που είναι αρκετά μεγαλύτερη υπάρχει μια δυσανάλογη επιβάρυνση σε επιφάνεια.

Για *n*=32 bits η κατάσταση αντιστρέφεται, και το φίλτρο Karatsuba έχει μικρότερο *Area×Delay*.

| ***Area×Delay*$^2$ [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -7.24 | -3.29 |
| **32** | -23.30 | -14.41 |

Πίνακας 5.11: Ποσοστιαία διαφορά μετρικής $A \cdot T^2$ για direct FIR φίλτρα

Τα αποτελέσματα για την μετρική *Area×Delay*$^2$ περιλαμβάνονται στον πίνακα 5.11. Σε αντίθεση με το *Area×Delay*, παρατηρούμε ότι το φίλτρο Karatsuba παρουσιάζει καλύτερα αποτελέσματα για όλα τα μήκη λέξης. Η καθυστέρηση έχει μεγαλύτερη βαρύτητα στην μετρική *Area×Delay*$^2$. Συγκεκριμένα, εξαρτάται από το τετράγωνο της καθυστέρησης. Το φίλτρο Karatsuba υπερτερεί σημαντικά έναντι του συμβατικού ως προς την καθυστέρηση, γεγονός που το καθιστά αποδοτικότερο ως προς την μετρική *Area×Delay*$^2$.

| ***Power×Delay* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | +11.40 | +10.77 |
| **32** | -15.10 | -2.34 |

Πίνακας 5.12: Ποσοστιαία διαφορά μετρικής $P \cdot T$ για direct FIR φίλτρα

Όσον αφορά τα αποτελέσματα για την μετρική *Power×Delay*, αυτά περιλαμβάνονται στον πίνακα 5.12. Όπως αναφέρθηκε και για τα αποτελέσματα της επιφάνειας, έτσι και η

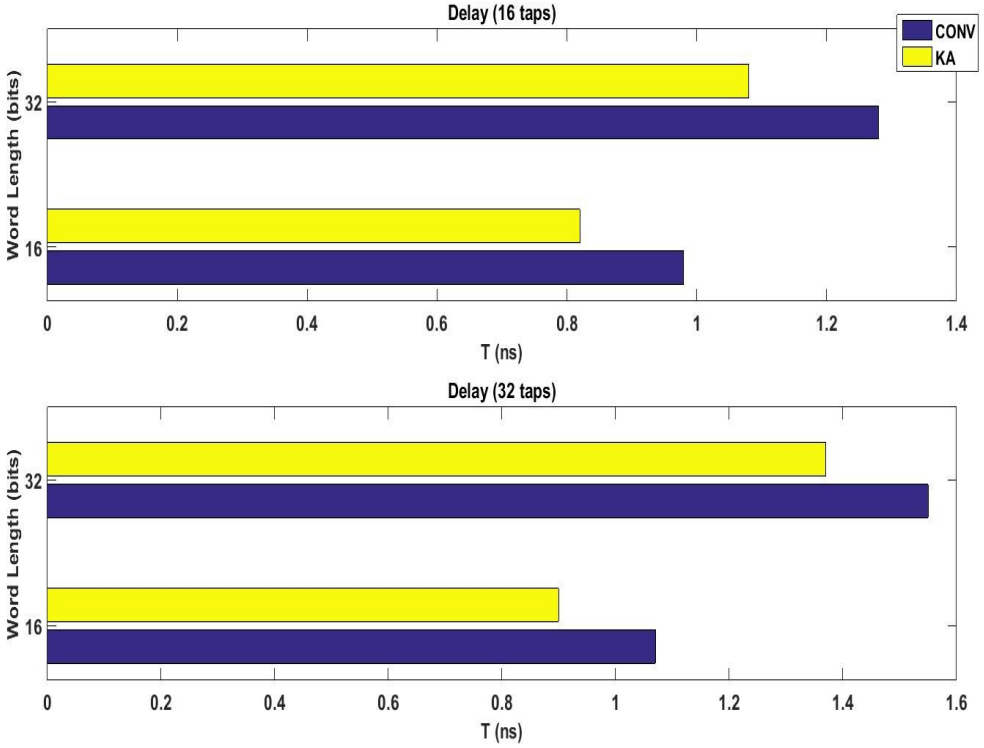
κατανάλωση δεν αυξάνεται γραμμικά με την αύξηση της συχνότητας λειτουργίας των φίλτρων. Τα αποτελέσματα των προσομοιώσεων έδειξαν ότι το φίλτρο Karatsuba στις υψηλές συχνότητες που λειτουργεί, παρουσιάζει ιδιαίτερα υψηλή κατανάλωση που οφείλεται κυρίως στον μεγάλο αριθμό καταχωρητών που χρησιμοποιεί. Το γεγονός αυτό, αποτυπώνεται στα αποτελέσματα του πίνακα 5.12 για $n$=16 bits, όπου το φίλτρο Karatsuba παρουσιάζει μεγαλύτερο *Power*×*Delay* από ότι το συμβατικό φίλτρο. Τα αποτελέσματα αντιστρέφονται για $n$=32 bits, καθώς όπως έχουμε αναφέρει η αρχιτεκτονική Karatsuba είναι αποδοτικότερη για μεγαλύτερα μήκη λέξης.

### 5.2.2 Transposed μορφή

Στην ενότητα αυτή παρουσιάζονται τα αποτελέσματα από την σύνθεση των υλοποιήσεων των φίλτρων FIR σε transposed μορφή. Στον πίνακα 5.13 δίνονται τα αποτελέσματα για την ελάχιστη καθυστέρηση. Τα αποτελέσματα αυτά απεικονίζονται γραφικά στο σχήμα 5.8. Όπως αναμέναμε, το φίλτρο Karatsuba παρουσιάζει μικρότερη ελάχιστη καθυστέρηση σε κάθε περίπτωση.

| ***Delay* Conventional (nsec)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.98 | 1.07 |
| **32** | 1.28 | 1.55 |
| ***Delay* Karatsuba (nsec)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.82 | 0.90 |
| **32** | 1.08 | 1.37 |

Πίνακας 5.13: Αποτελέσματα ελάχιστης καθυστέρησης για transposed FIR φίλτρα

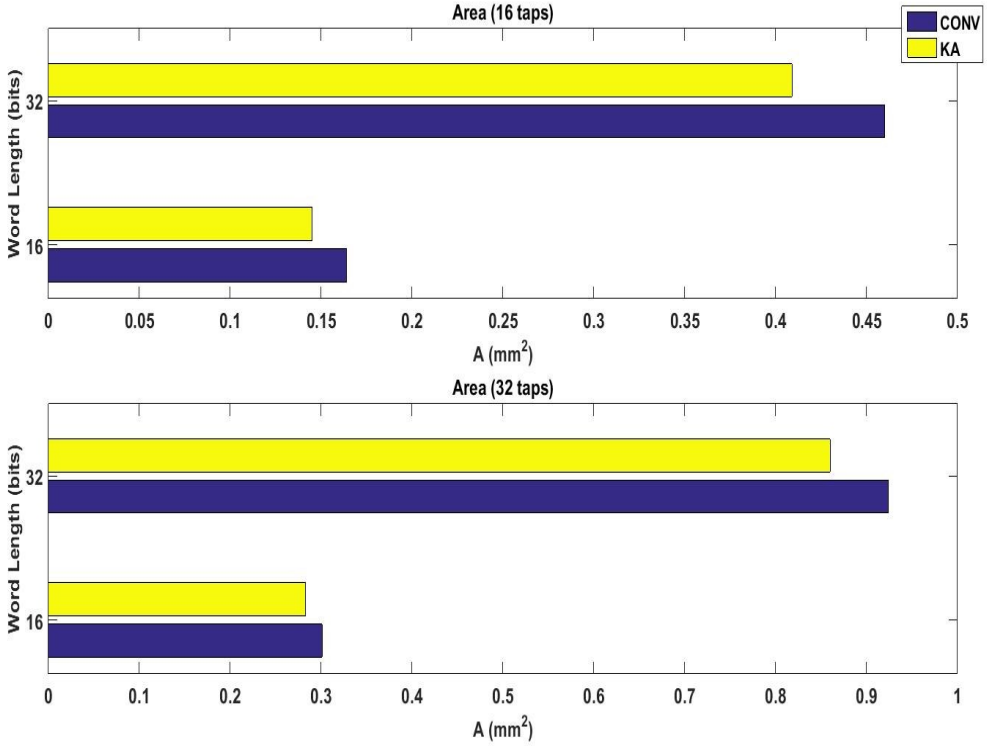



Σχήμα 5.8: Ελάχιστη καθυστέρηση υλοποιήσεων transposed μορφής

Τα αποτελέσματα για την επιφάνεια και την μέση κατανάλωση των υλοποιήσεων σε transposed μορφή παρουσιάζονται στους πίνακες 5.14 και 5.15 αντίστοιχα. Τα συγκεκριμένα αποτελέσματα απεικονίζονται στα σχήματα 5.9 και 5.10 αντίστοιχα. Όπως έγινε και στην περίπτωση των υλοποιήσεων σε direct μορφή, οι υλοποιήσεις συγκρίνονται ως προς την επιφάνεια και την κατανάλωση για την ίδια περίοδο ρολογιού $T_{\text{HP-CONV}}$.

| ***Area* Conventional (mm$^2$)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.164 | 0.301 |
| **32** | 0.460 | 0.924 |
| ***Area* Karatsuba (mm$^2$)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.145 | 0.283 |
| **32** | 0.409 | 0.860 |

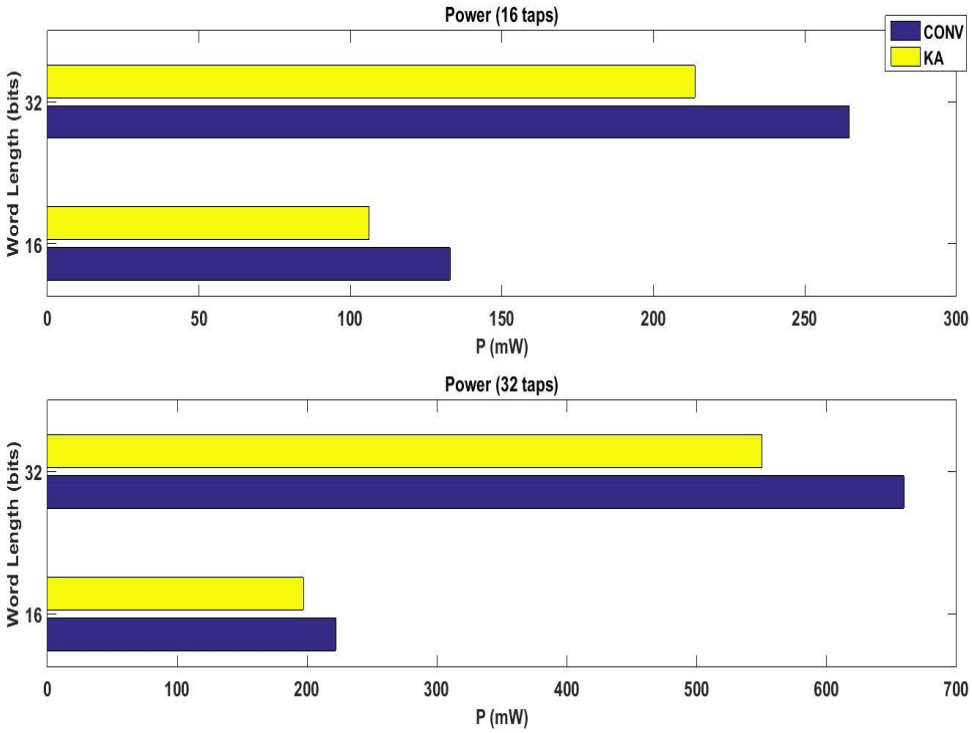
Πίνακας 5.14: Αποτελέσματα επιφάνειας για $T=T_{\text{HP-CONV}}$, για transposed FIR φίλτρα




Σχήμα 5.9: Επιφάνεια υλοποιήσεων transposed μορφής για $T=T_{\text{HP-CONV}}$

| ***Power* Conventional (mW)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 132.9 | 222.1 |
| **32** | 264.6 | 659.6 |
| ***Power* Karatsuba (mW)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 106.2 | 197.1 |
| **32** | 213.7 | 550.2 |

Πίνακας 5.15: Αποτελέσματα μέσης κατανάλωσης για $T=T_{\text{HP-CONV}}$, για transposed FIR φίλτρα

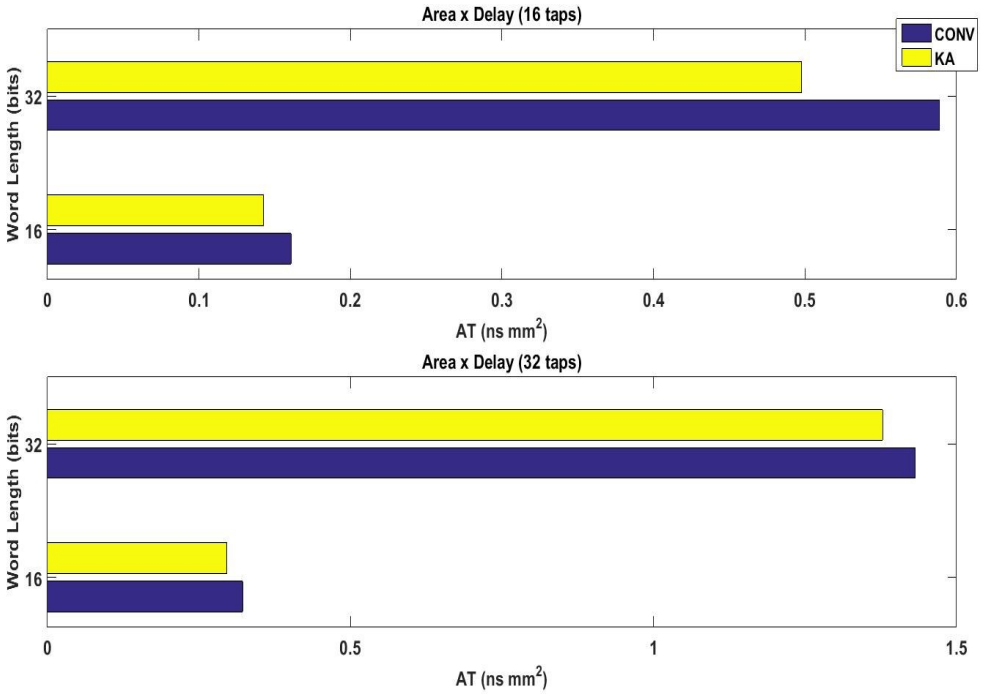



Σχήμα 5.10: Κατανάλωση των υλοποιήσεων σε transposed μορφή για $T=T_{\text{HP-CONV}}$

Τα αποτελέσματα για τις ποσότητες *Area×Delay*, *Area×Delay*$^2$ και *Power×Delay* δίνονται στους πίνακες 5.16, 5.17 και 5.18 αντίστοιχα. Τα αποτελέσματα αυτά απεικονίζονται στα σχήματα 5.11, 5.12 και 5.13 αντίστοιχα.

| ***Area×Delay* Conventional (nsec·mm$^2$)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.161 | 0.322 |
| **32** | 0.589 | 1.432 |
| ***Area×Delay* Karatsuba (nsec·mm$^2$)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.143 | 0.296 |
| **32** | 0.498 | 1.378 |

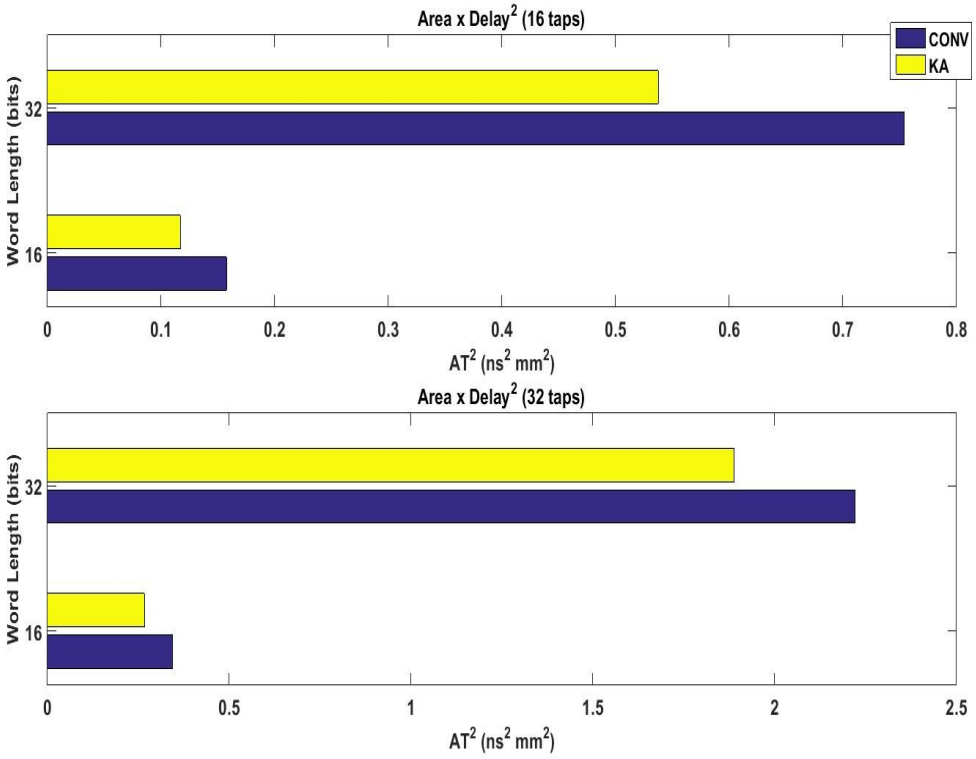
Πίνακας 5.16: Αποτελέσματα μετρικής *Area x Delay* για transposed FIR φίλτρα




Σχήμα 5.11: *Area×Delay* των υλοποιήσεων σε transposed μορφή

| ***Area×Delay*² Conventional (nsec²·mm²)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.158 | 0.345 |
| **32** | 0.754 | 2.220 |
| ***Area×Delay*² Karatsuba (nsec²·mm²)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.117 | 0.267 |
| **32** | 0.538 | 1.888 |

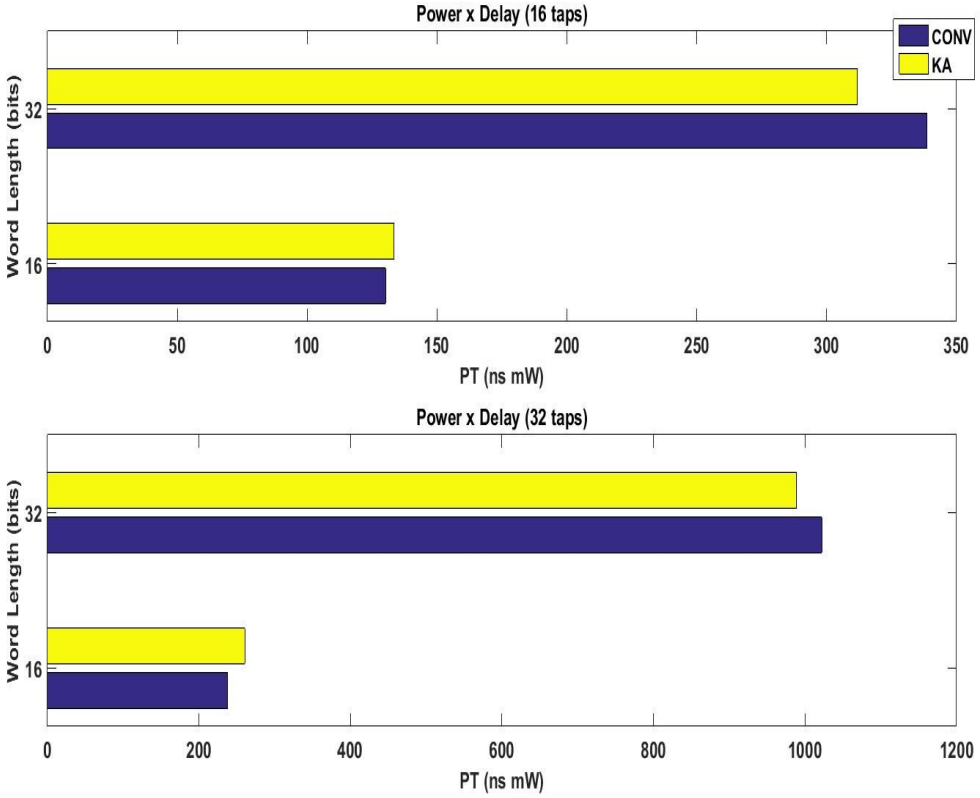
Πίνακας 5.17: Αποτελέσματα μετρικής *Area×Delay*² για transposed FIR φίλτρα




Σχήμα 5.12: *Area*×*Delay*$^2$ των υλοποιήσεων σε transposed μορφή

| ***Power*×*Delay* Conventional (nsec·mW)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 130.24 | 237.65 |
| **32** | 338.69 | 1022.40 |
| ***Power*×*Delay* Karatsuba (nsec·mW)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 133.50 | 260.46 |
| **32** | 311.80 | 988.87 |

Πίνακας 5.18: Αποτελέσματα μετρικής *Power*×*Delay* για transposed FIR φίλτρα




Σχήμα 5.13: *Power×Delay* των υλοποιήσεων σε transposed μορφή

Στο υπόλοιπο της παρούσας ενότητας, ακολουθούν τα αποτελέσματα που παρουσιάστηκαν παραπάνω, σε μορφή ποσοστών. Από τα δεδομένα προκύπτει ότι το φίλτρο Karatsuba είναι αρκετά αποδοτικότερο ως προς όλα τα κριτήρια για όλα τα μήκη λέξης και το πλήθος taps.

Μόνο για $n$=16 bits, το φίλτρο Karatsuba παρουσιάζει ελαφρώς υψηλότερες τιμές *Power×Delay* σε σχέση με το αντίστοιχο συμβατικό. Αντίθετα, για $n$=32 bits η κατάσταση ανατρέπεται και το φίλτρο Karatsuba παρουσιάζει ως συνήθως μικρότερες τιμές *Power×Delay*.

| ***Delay* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -16.33 | -15.89 |
| **32** | -15.63 | -11.61 |

Πίνακας 5.19: Ποσοστιαία διαφορά ελάχιστης καθυστέρησης για transposed FIR φίλτρα

| ***Area* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -11.59 | -5.98 |
| **32** | -11.09 | -6.93 |

Πίνακας 5.20: Ποσοστιαία διαφορά επιφάνειας για transposed FIR φίλτρα, στην ελάχιστη περίοδο ρολογιού των συμβατικών υλοποιήσεων ($T$=$T_{\text{HP-CONV}}$)

| ***Power* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -20.09 | -11.26 |
| **32** | -19.24 | -16.59 |

Πίνακας 5.21: Ποσοστιαία διαφορά μέσης κατανάλωσης για transposed FIR φίλτρα, στην ελάχιστη περίοδο ρολογιού των συμβατικών υλοποιήσεων ($T$=$T_{\text{HP-CONV}}$)

| ***Area*×*Delay* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -11.22 | -8.06 |
| **32** | -15.44 | -3.77 |

Πίνακας 5.22: Ποσοστιαία διαφορά μετρικής $A \cdot T$ για transposed FIR φίλτρα

| ***Area*×*Delay*$^2$ [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -25.72 | -22.67 |
| **32** | -28.65 | -14.94 |

Πίνακας 5.23: Ποσοστιαία διαφορά μετρικής $A \cdot T^2$ για transposed FIR φίλτρα

| ***Power*×*Delay* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | +2.50 | +9.60 |
| **32** | -7.94 | -3.28 |

Πίνακας 5.24: Ποσοστιαία διαφορά μετρικής P·T για transposed FIR φίλτρα

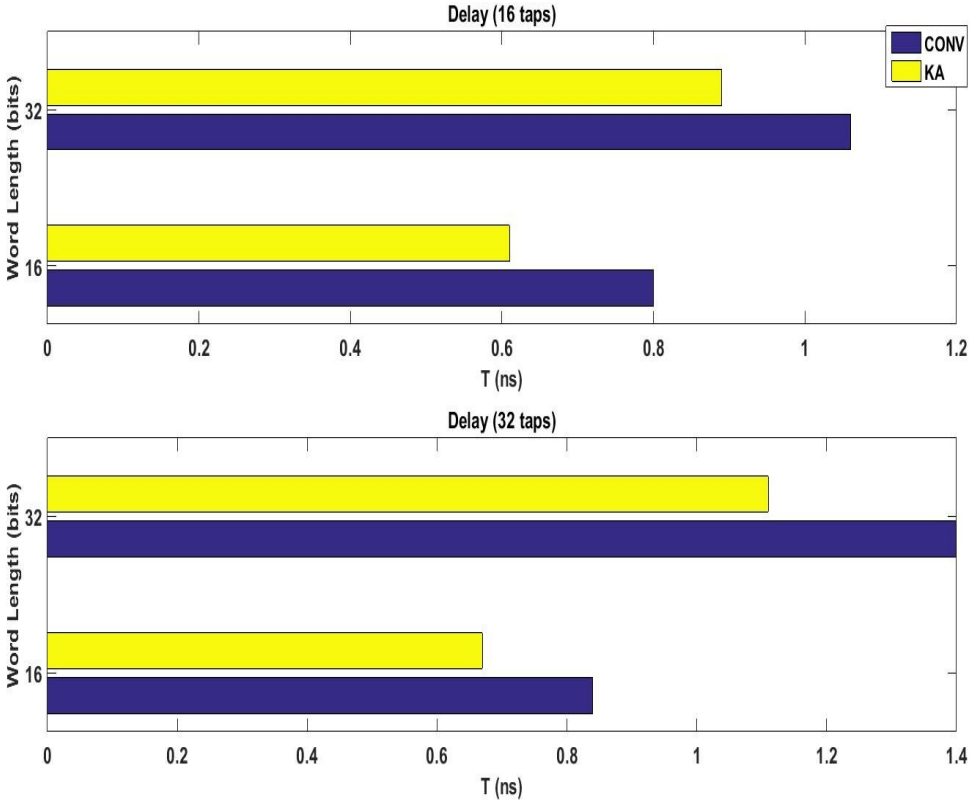
### 5.2.3 Mixed μορφή

Όπως και στις δύο προηγούμενες ενότητες ακολουθούν τα αποτελέσματα για την mixed μορφή των φίλτρων FIR. Στους πίνακες 5.25, 5.26 και 5.27 δίνονται τα αποτελέσματα για την καθυστέρηση, την επιφάνεια και την μέση κατανάλωση αντίστοιχα. Τα ίδια αποτελέσματα απεικονίζονται γραφικά στα σχήματα 5.14, 5.15 και 5.16 αντίστοιχα.

Όπως και στις άλλες δύο μορφές που παρουσιάστηκαν προηγουμένως, το φίλτρο Karatsuba σε mixed μορφή έχει σε κάθε περίπτωση μικρότερη ελάχιστη καθυστέρηση από το αντίστοιχο συμβατικό φίλτρο. Από τον πίνακα 5.25 προκύπτει ότι η mixed μορφή τόσο της συμβατικής υλοποίησης, όσο και της υλοποίησης Karatsuba έχουν μικρότερη ελάχιστη καθυστέρηση σε σχέση με τις αντίστοιχες υλοποιήσεις των άλλων δύο μορφών. Το γεγονός αυτό καθιστά την mixed μορφή, την ταχύτερη μεταξύ των τριών μορφών που υλοποιήθηκαν.

| ***Delay* Conventional (nsec)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.80 | 0.84 |
| **32** | 1.06 | 1.40 |
| ***Delay* Karatsuba (nsec)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.61 | 0.67 |
| **32** | 0.89 | 1.11 |

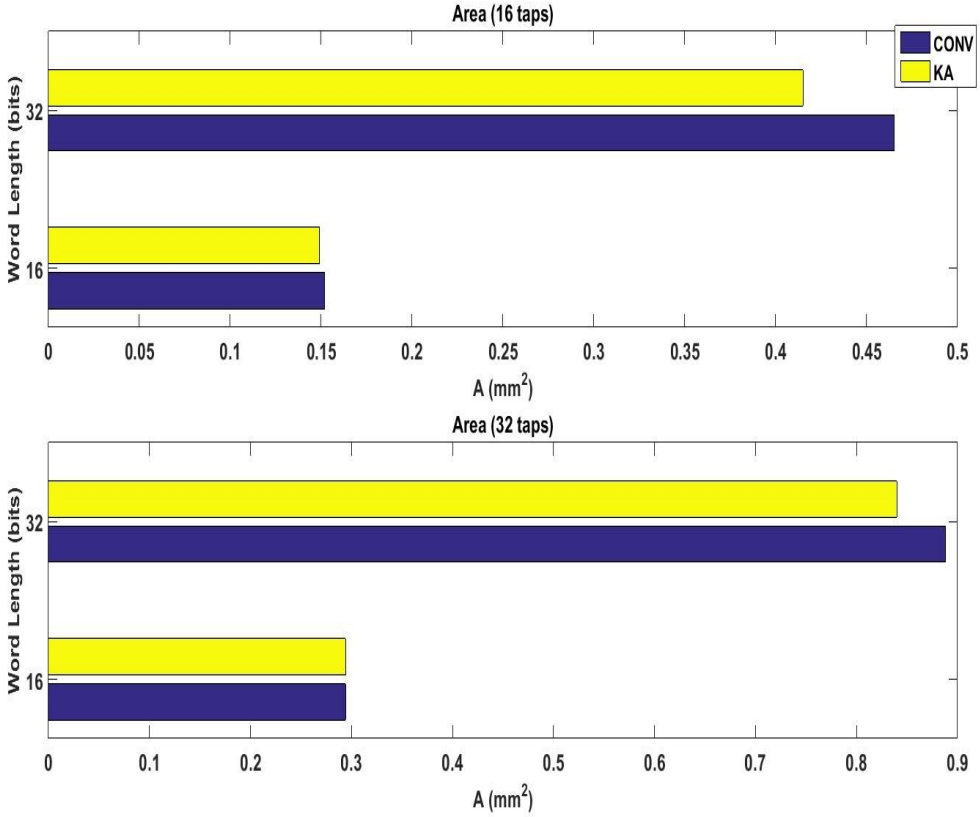
Πίνακας 5.25: Αποτελέσματα ελάχιστης καθυστέρησης για mixed FIR φίλτρα




Σχήμα 5.14: Ελάχιστη καθυστέρηση των υλοποιήσεων σε mixed μορφή

| ***Area* Conventional ($mm^2$)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.152 | 0.294 |
| **32** | 0.465 | 0.888 |
| ***Area* Karatsuba ($mm^2$)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.149 | 0.294 |
| **32** | 0.415 | 0.840 |

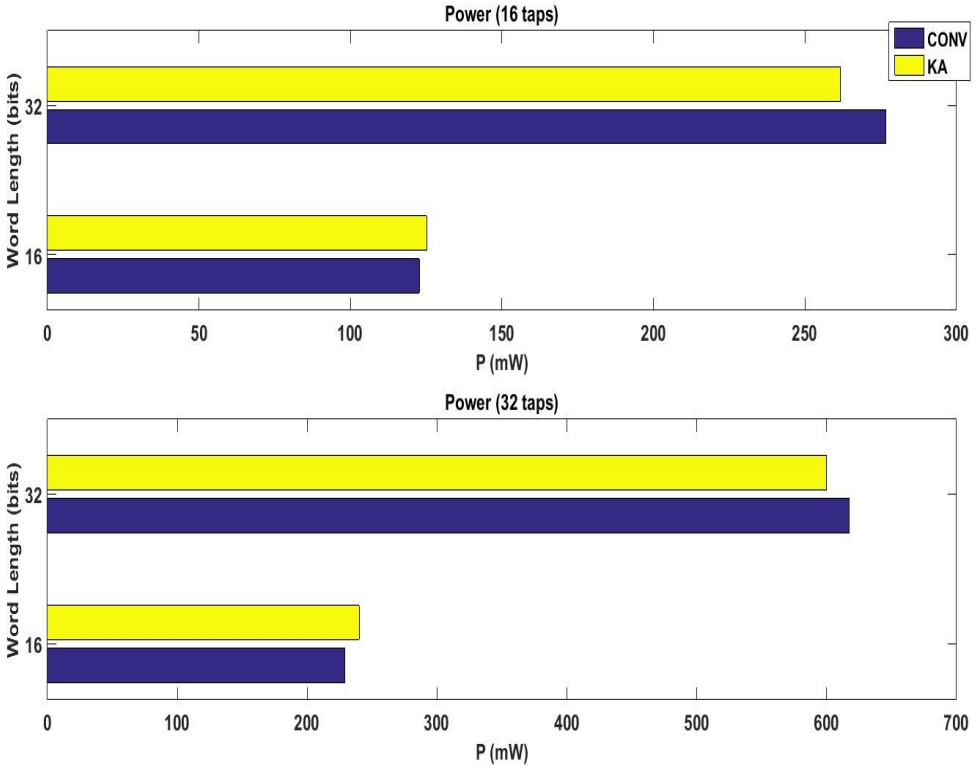
Πίνακας 5.26: Αποτελέσματα επιφάνειας για $T=T_{\text{HP-CONV}}$, για mixed FIR φίλτρα




Σχήμα 5.15: Επιφάνεια των υλοποιήσεων σε mixed μορφή για $T=T_{HP\text{-}CONV}$

| ***Power* Conventional (mW)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 122.7 | 229.0 |
| **32** | 276.6 | 617.4 |
| ***Power* Karatsuba (mW)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 125.1 | 240.2 |
| **32** | 261.7 | 600.0 |

Πίνακας 5.27: Αποτελέσματα μέσης κατανάλωσης για $T=T_{HP\text{-}CONV}$, για mixed FIR φίλτρα

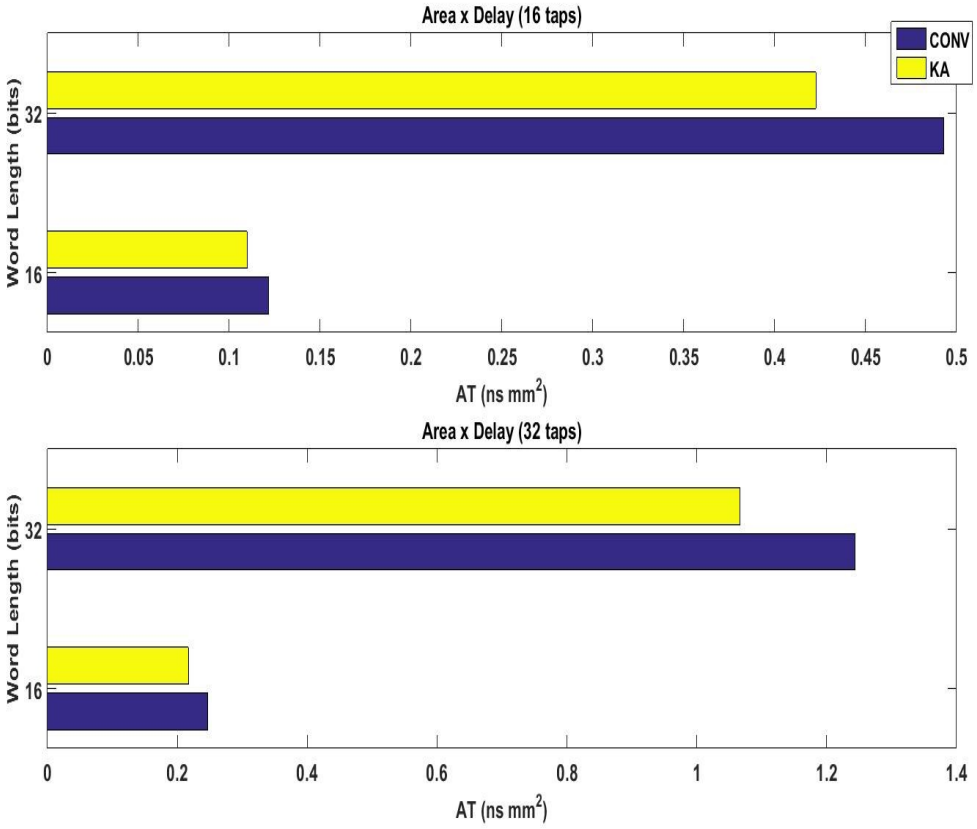



Σχήμα 5.16: Κατανάλωση των υλοποιήσεων σε mixed μορφή για $T=T_{\text{HP-CONV}}$

Στους πίνακες 5.28, 5.29 και 5.30 δίνονται τα αποτελέσματα για τις ποσότητες *Area*×*Delay*, *Area*×*Delay*$^2$ και *Power*×*Delay*. Τα ίδια αποτελέσματα απεικονίζονται γραφικά στα σχήματα 5.17, 5.18 και 5.19 αντίστοιχα.

| ***Area*×*Delay* Conventional (nsec·mm$^2$)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.122 | 0.247 |
| **32** | 0.493 | 1.243 |
| ***Area*×*Delay* Karatsuba (nsec·mm$^2$)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.110 | 0.217 |
| **32** | 0.423 | 1.067 |

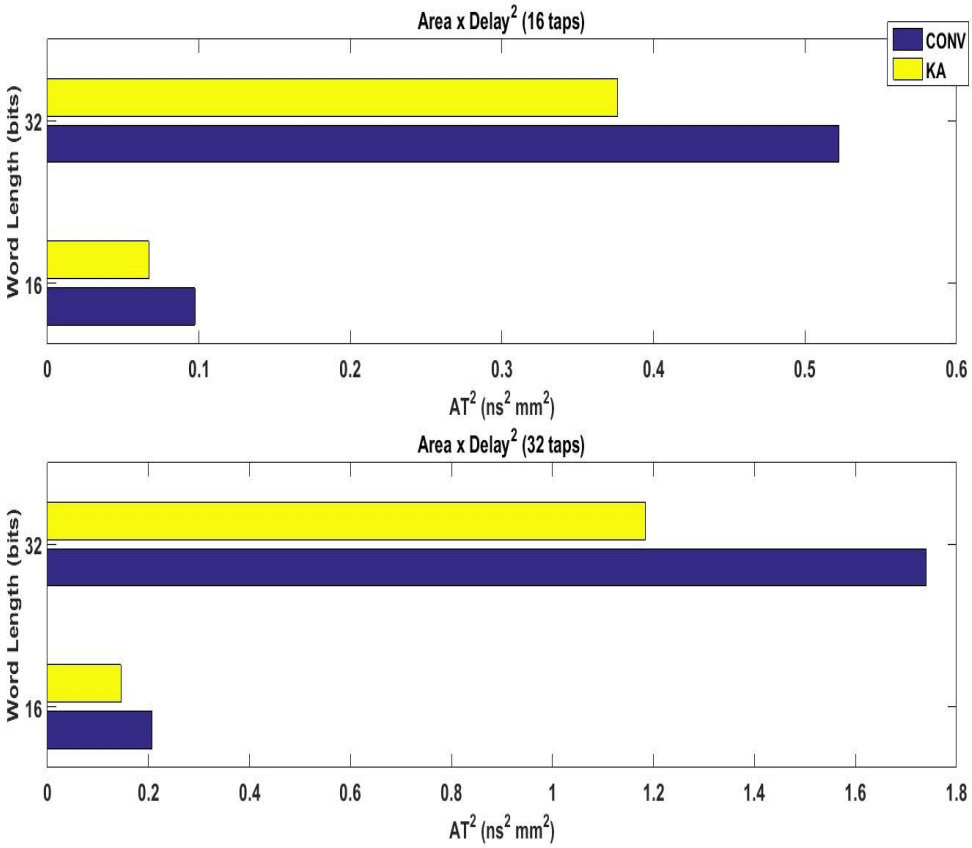
Πίνακας 5.28: Αποτελέσματα μετρικής *Area*×*Delay* για mixed FIR φίλτρα




Σχήμα 5.17: *Area×Delay* των υλοποιήσεων σε mixed μορφή

| ***Area×Delay*² Conventional (nsec²·mm²)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.097 | 0.207 |
| **32** | 0.523 | 1.741 |
| ***Area×Delay*² Karatsuba (nsec²·mm²)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 0.067 | 0.145 |
| **32** | 0.376 | 1.184 |

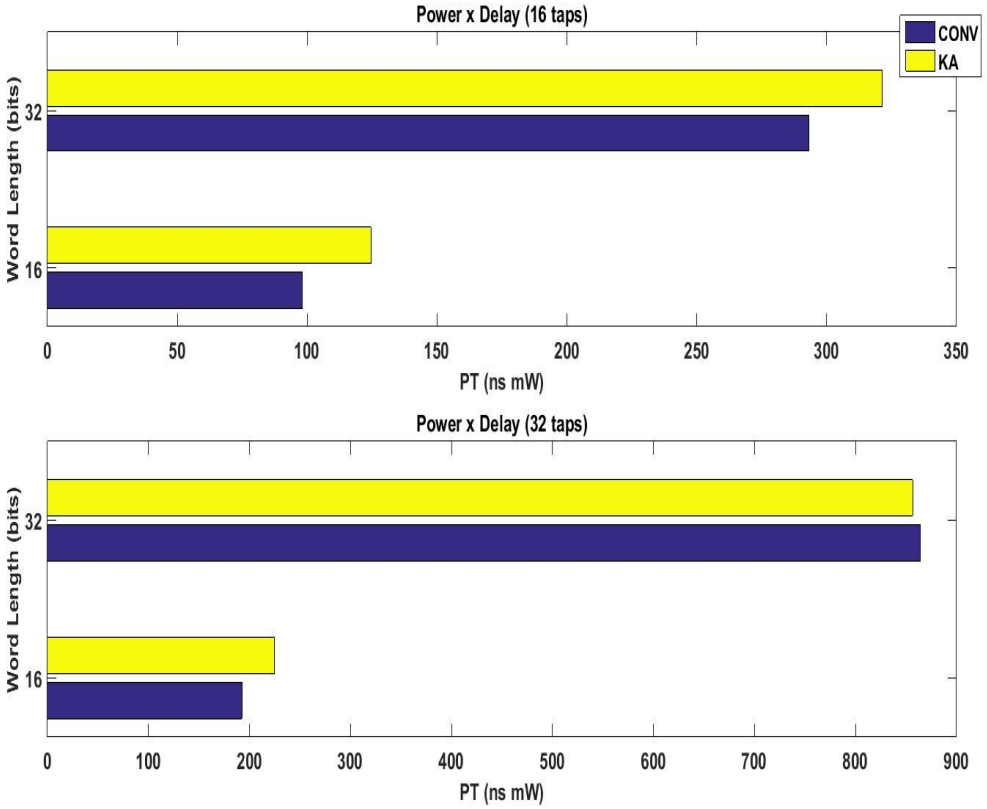
Πίνακας 5.29: Αποτελέσματα μετρικής *Area×Delay*² για mixed FIR φίλτρα




Σχήμα 5.18: *Area×Delay* $^2$ των υλοποιήσεων σε mixed μορφή

| ***Power×Delay* Conventional (nsec·mW)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 98.16 | 192.36 |
| **32** | 293.20 | 864.36 |
| ***Power×Delay* Karatsuba (nsec·mW)** | | |
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | 124.50 | 224.65 |
| **32** | 321.47 | 856.81 |

Πίνακας 5.30: Αποτελέσματα μετρικής *Power×Delay* για mixed FIR φίλτρα




Σχήμα 5.19*: Power×Delay* των υλοποιήσεων σε mixed μορφή

Στους πίνακες 5.31 έως 5.36 δίνονται τα αποτελέσματα σε μορφή ποσοστών. Στον πίνακα 5.31 βλέπουμε ότι το φίλτρο Karatsuba εμφανίζει σημαντικά μειωμένη ελάχιστη καθυστέρηση σε σύγκριση με το συμβατικό φίλτρο. Συγκρίνοντας τα αποτελέσματα του πίνακα 5.31 με τα αντίστοιχα αποτελέσματα των άλλων δύο μορφών(πίνακες 5.7 και 5.19 ) συμπεραίνουμε ότι το φίλτρο Karatsuba σε mixed μορφή εμφανίζει την μεγαλύτερη ποσοστιαία μείωση της ελάχιστης καθυστέρησης. Το γεγονός αυτό αναμενόταν διότι στη mixed μορφή των υλοποιήσεων εστιάσαμε στην ελαχιστοποίηση της καθυστέρησης των φίλτρων. Για να το πετύχουμε αυτό εφαρμόσαμε την τεχνική της συνεχούς διοχέτευσης ώστε να περιορίσουμε το κρίσιμο μονοπάτι των φίλτρων στο κύκλωμα των μονάδων πολλαπλασιασμού (MU).

| ***Delay* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -23.75 | -20.24 |
| **32** | -16.04 | -20.71 |

Πίνακας 5.31: Ποσοστιαία διαφορά ελάχιστης καθυστέρησης για mixed FIR φίλτρα

| ***Area* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -1.97 | 0.00 |
| **32** | -10.75 | -5.41 |

Πίνακας 5.32: Ποσοστιαία διαφορά επιφάνειας για mixed FIR φίλτρα, στην ελάχιστη περίοδο ρολογιού των συμβατικών υλοποιήσεων ($T=T_{\text{HP-CONV}}$)

| ***Power* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | +1.96 | +4.89 |
| **32** | -5.39 | -2.82 |

Πίνακας 5.33: Ποσοστιαία διαφορά μέσης κατανάλωσης για mixed FIR φίλτρα, στην ελάχιστη περίοδο ρολογιού των συμβατικών υλοποιήσεων ($T=T_{\text{HP-CONV}}$)

Το φίλτρο Karatsuba σε mixed μορφή είναι αποδοτικότερο από το αντίστοιχο συμβατικό ως προς την επιφάνεια για όλα τα μήκη λέξης και αριθμό taps, εκτός από $n$=16 και $t$=32 όπου οι δύο υλοποιήσεις εμφανίζουν ίδια επιφάνεια (πίνακας 5.32).

Από την άλλη μεριά, το φίλτρο Karatsuba εμφανίζει μικρότερη κατανάλωση μόνο για $n$=32 bits (πίνακας 5.33). Όπως ξέρουμε στην mixed μορφή έχουν προστεθεί μονάδες καθυστέρησης στην έξοδο των μονάδων πολλαπλασιασμού με σκοπό την ελαχιστοποίηση της καθυστέρησης. Όμως, οι επιπρόσθετες μονάδες καθυστέρησης εισάγουν σημαντική κατανάλωση, ειδικά στις πολύ υψηλές συχνότητες λειτουργίας του φίλτρου Karatsuba.

| ***Area*×*Delay* [KA/CONV-1] (%)** | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -9.70 | -12.10 |
| **32** | -14.23 | -14.20 |

Πίνακας 5.34: Ποσοστιαία διαφορά μετρικής $A \cdot T$ για mixed FIR φίλτρα

| *Area×Delay*$^2$ [KA/CONV-1] (%) | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | -31.15 | -29.89 |
| **32** | -27.99 | -31.97 |

Πίνακας 5.35: Ποσοστιαία διαφορά μετρικής $A \cdot T^2$ για mixed FIR φίλτρα

Από τους πίνακες 5.34 και 5.35, εξάγουμε το συμπέρασμα ότι το φίλτρο Karatsuba σε mixed μορφή εμφανίζει σημαντικά μικρότερες τιμές *Area×Delay* και *Area×Delay*$^2$ σε σύγκριση με το αντίστοιχο συμβατικό φίλτρο. Το γεγονός αυτό ενισχύεται από την μεγάλη μείωση της καθυστέρησης που προσφέρει η αρχιτεκτονική Karatsuba στην mixed μορφή της.

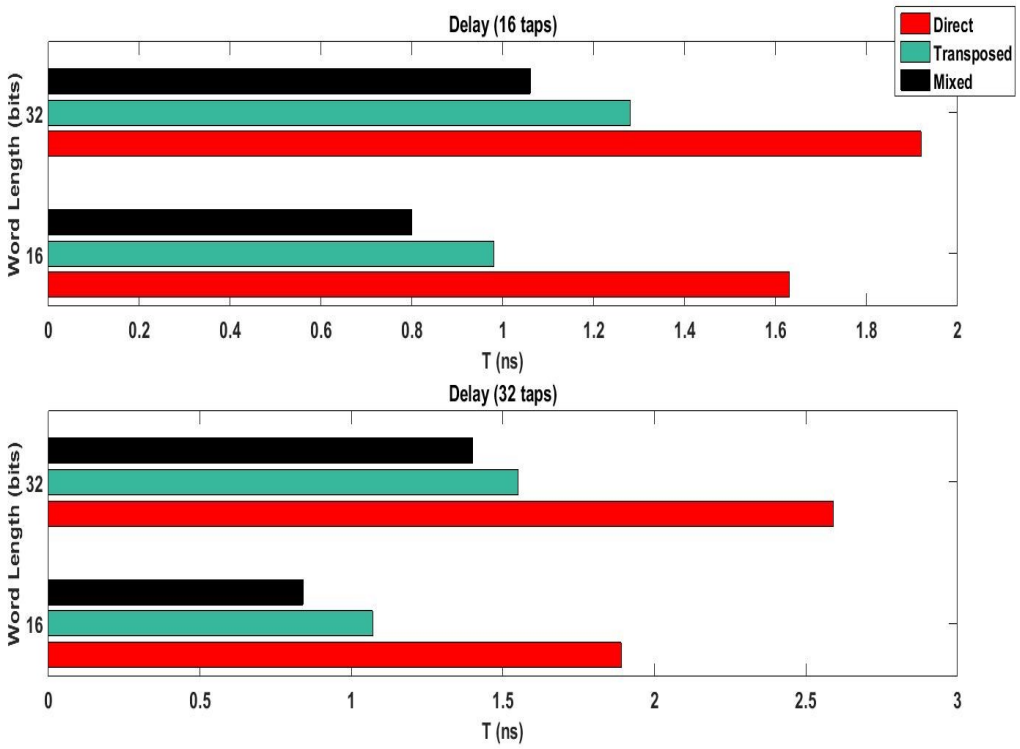
Αντίθετα, λόγω της υψηλής κατανάλωσης του φίλτρου Karatsuba σε mixed μορφή, βλέπουμε ότι εμφανίζει μεγαλύτερες τιμές *Power×Delay* σε σχέση με το συμβατικό φίλτρο (πίνακας 5.36). Μόνο για $n$=32 bits και $t$=32 taps, εμφανίζει ελάχιστα μικρότερη τιμή.

| *Power×Delay* [KA/CONV-1] (%) | | |
|---|---|---|
| **#bits** | **#taps** | |
| | **16** | **32** |
| **16** | +26.83 | +16.79 |
| **32** | +9.64 | -0.87 |

Πίνακας 5.36: Ποσοστιαία διαφορά μετρικής $P \cdot T$ για mixed FIR φίλτρα

### 5.2.4 Υλοποιήσεις συμβατικών φίλτρων

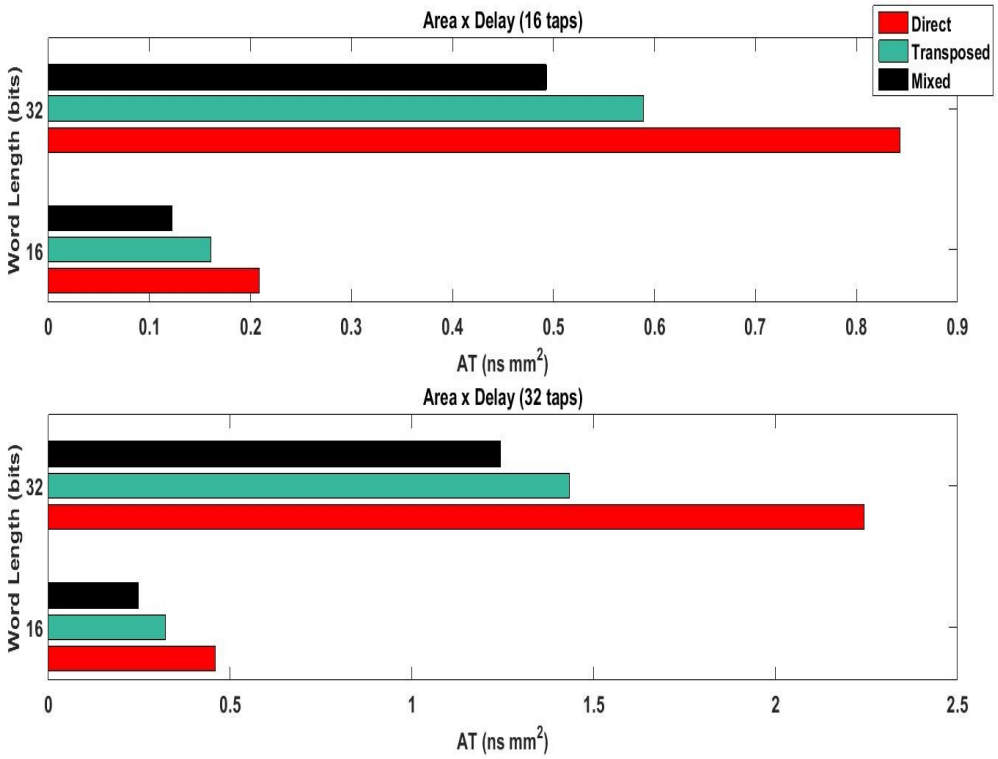
Τα συμβατικά φίλτρα FIR υλοποιήθηκαν σε τρείς διαφορετικές μορφές: direct, transposed και mixed. Στην ενότητα αυτή παρουσιάζονται τα συγκριτικά αποτελέσματα των τριών διαφορετικών τοπολογιών της συμβατικής αρχιτεκτονικής του φίλτρου FIR.



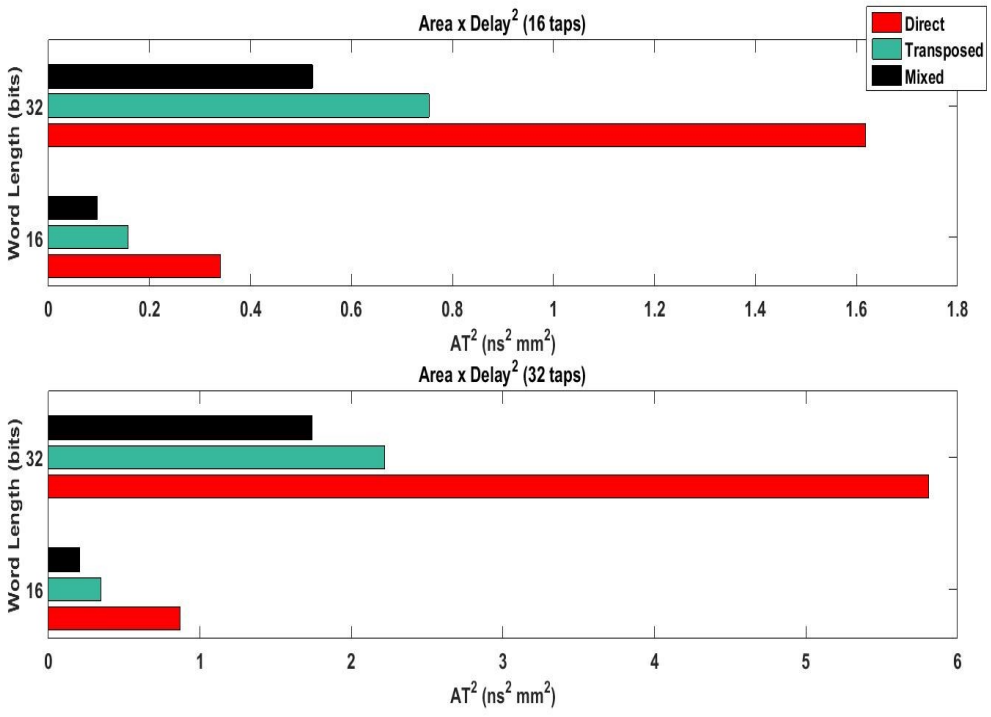

Σχήμα 5.20: Ελάχιστη καθυστέρηση των συμβατικών φίλτρων FIR

Στο σχήμα 5.20 παρουσιάζονται τα αποτελέσματα για την ελάχιστη καθυστέρηση των συμβατικών φίλτρων FIR. Όπως αναμενόταν, η mixed μορφή είναι η ταχύτερη, με την transposed και την direct μορφή να ακολουθούν.

Στα σχήματα 5.21, 5.22 και 5.23 απεικονίζονται τα αποτελέσματα για τις μετρικές *Area*×*Delay*, *Area*×*Delay*$^2$ και *Power*×*Delay* αντίστοιχα. Βλέπουμε ότι ως προς τις μετρικές *Area*×*Delay* και *Area*×*Delay*$^2$, οι υλοποιήσεις ακολουθούν την σειρά κατάταξης της ελάχιστης καθυστέρησης. Δηλαδή οι τρείς μορφές κατατάσσονται από την αποδοτικότερη προς την λιγότερη αποδοτική ως εξής: mixed, transposed και direct.

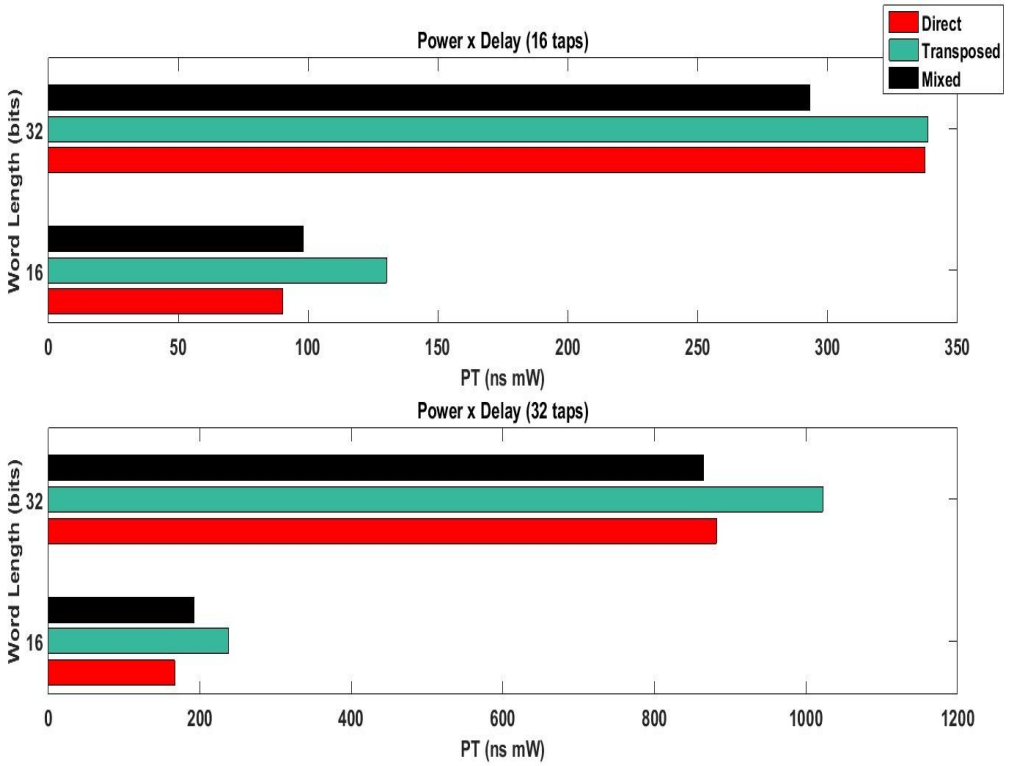



Σχήμα 5.21: Αποτελέσματα *Area×Delay* των συμβατικών φίλτρων FIR




Σχήμα 5.22: Αποτελέσματα *Area×Delay*$^2$ των συμβατικών φίλτρων FIR

Στο σχήμα 5.23, παρατηρούμε ότι η transposed μορφή εμφανίζει αυξημένο *Power×Delay* σε σχέση με τις άλλες δύο μορφές. Η direct και mixed μορφή, έχουν παρόμοιες τιμές *Power×Delay* εκτός από την περίπτωση του φίλτρου με *t*=16 και *n*=32 όπου η direct μορφή παρουσιάζει αρκετά μεγαλύτερη τιμή.

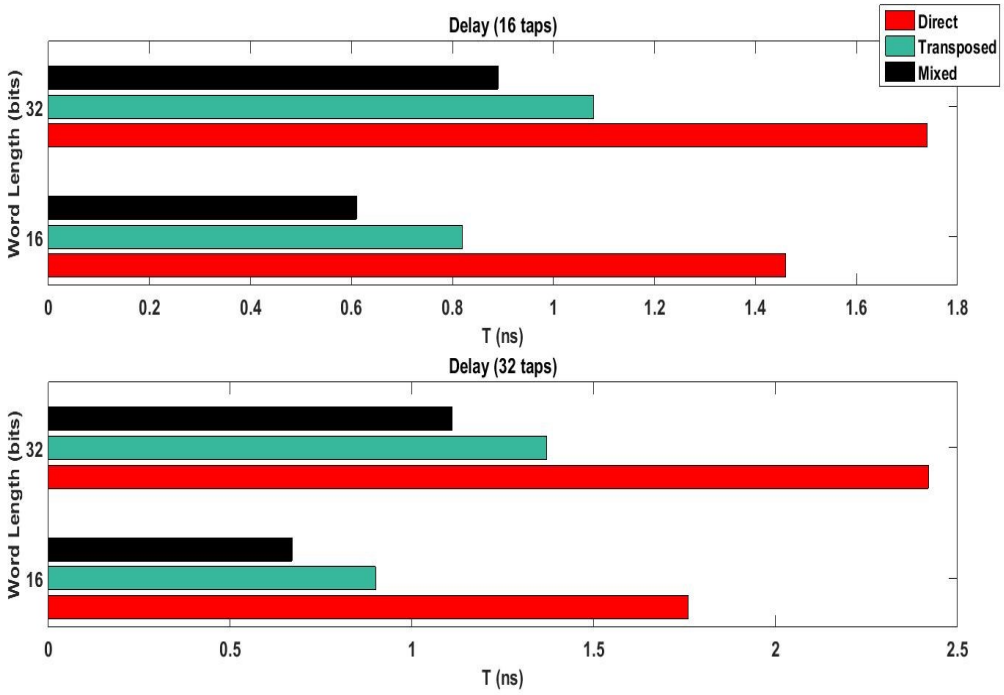



Σχήμα 5.23: Αποτελέσματα *Power×Delay* των συμβατικών φίλτρων FIR

### 5.2.5 Υλοποιήσεις φίλτρων Karatsuba

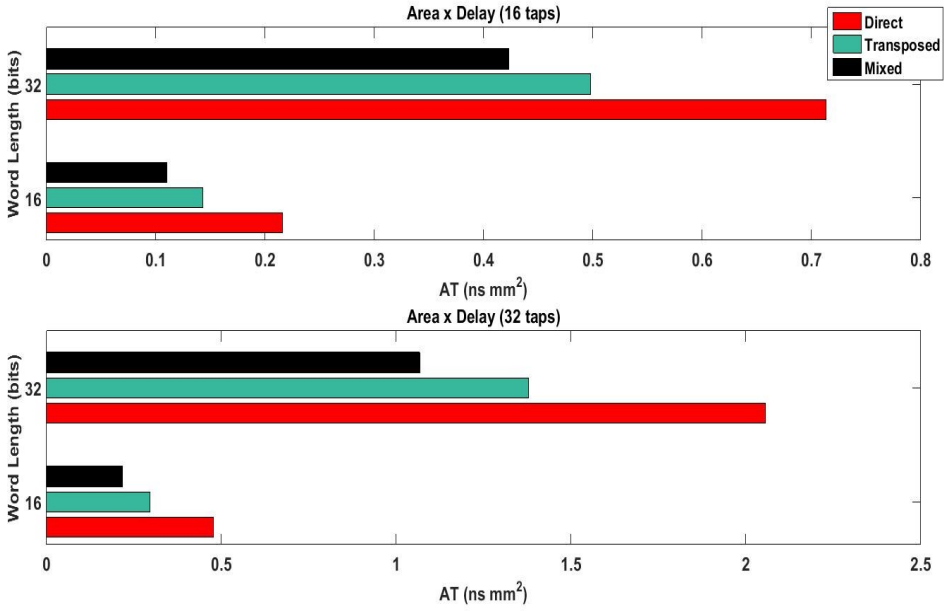
Στην παρούσα ενότητα παρουσιάζονται τα συγκριτικά αποτελέσματα των τριών διαφορετικών τοπολογιών που υλοποιήθηκε η αρχιτεκτονική Karatsuba του φίλτρου FIR. Στο σχήμα 5.24 απεικονίζονται τα αποτελέσματα για την ελάχιστη καθυστέρηση της direct, της transposed και της mixed μορφής του φίλτρου Karatsuba.

Όσον αφορά την ελάχιστη καθυστέρηση, ισχύει ότι και για τις συμβατικές υλοποιήσεις, δηλαδή η mixed μορφή είναι η ταχύτερη, με την transposed μορφή να ακολουθεί ενώ η direct αποτελεί την πιο αργή από τις τρείς μορφές.

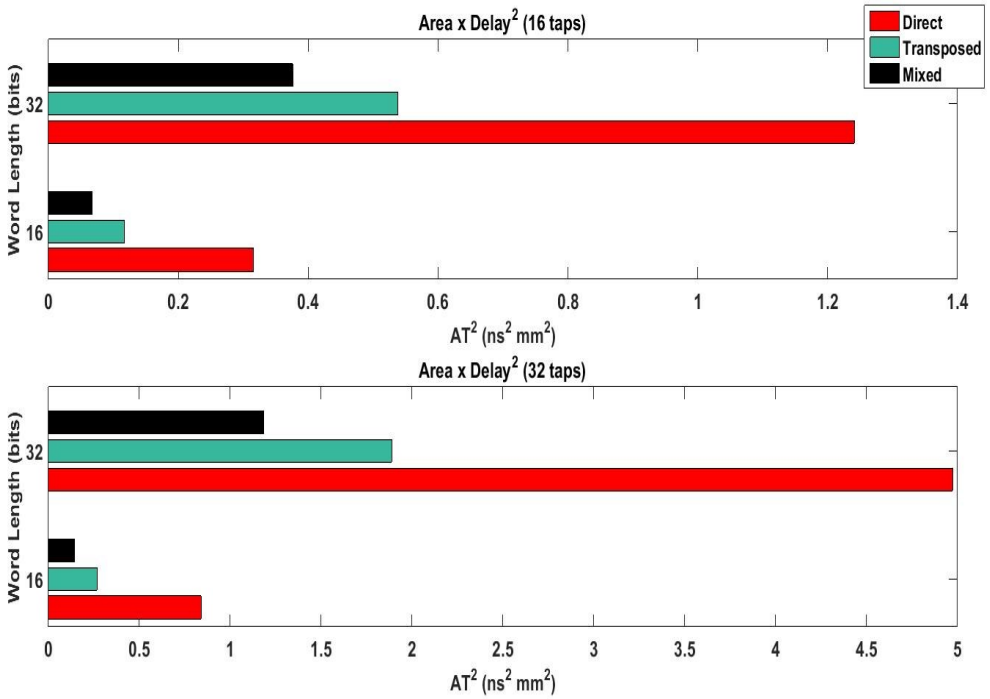



Σχήμα 5.24: Ελάχιστη καθυστέρηση των φίλτρων Karatsuba

Τα αποτελέσματα για τις μετρικές *Area×Delay*, *Area×Delay*$^2$ και *Power×Delay* απεικονίζονται στα σχήματα 5.25, 5.26 και 5.27 αντίστοιχα.



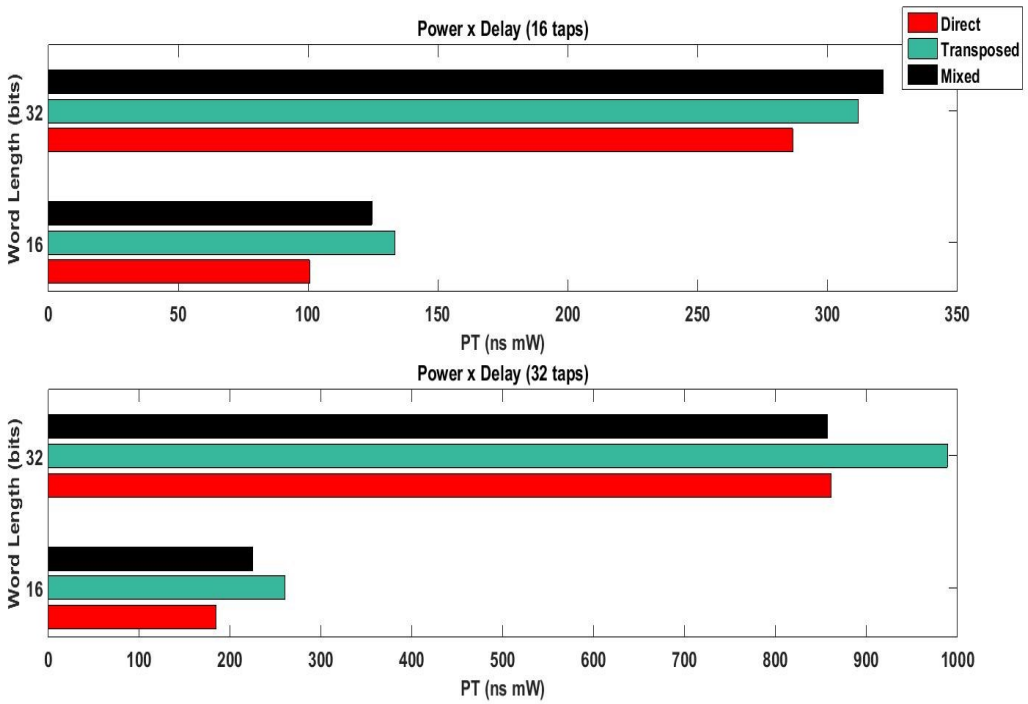

Σχήμα 5.25: Αποτελέσματα *Area×Delay* των φίλτρων Karatsuba




Σχήμα 5.26: Αποτελέσματα *Area×Delay* $^2$ των φίλτρων Karatsuba

Η απόδοση των τριών διαφορετικών μορφών ως προς την ελάχιστη καθυστέρηση, καθορίζει και την απόδοση τους ως προς τα κριτήρια *Area×Delay* και *Area×Delay*$^2$. Όπως μπορούμε να δούμε στα σχήματα 5.25 και 5.26, η mixed μορφή παρουσιάζει τις μικρότερες τιμές, με την transposed και την direct μορφή να ακολουθούν.

Στο σχήμα 5.27 παρατηρούμε ότι η transposed μορφή παρουσιάζει τις υψηλότερες τιμές *Power×Delay* σε σχέση με τις άλλες δύο μορφές. Η ίδια τάση παρουσιάστηκε και στην περίπτωση των συμβατικών υλοποιήσεων των φίλτρων όπως είδαμε στην προηγούμενη ενότητα. Εδώ, εξαίρεση αποτελεί η υλοποίηση με $t$=16 και $n$=32 όπου η mixed μορφή παρουσιάζει την μεγαλύτερη τιμή *Power×Delay*.




Σχήμα 5.27: Αποτελέσματα *Power×Delay* των φίλτρων Karatsuba

# 6

# ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΕΠΕΚΤΑΣΕΙΣ

## 6.1 Σύνοψη και συμπεράσματα

Συνοψίζοντας, στην παρούσα εργασία διερευνήσαμε την απόδοση μιας νέας αρχιτεκτονικής ενός φίλτρου FIR βασισμένης στον πολλαπλασιαστικό αλγόριθμο Karatsuba, συγκρίνοντάς την με την συμβατική αρχιτεκτονική. Οι δύο αρχιτεκτονικές υλοποιήθηκαν και συγκρίθηκαν σε τρεις διαφορετικές τοπολογίες: direct, transposed και mixed. Από τα πειραματικά αποτελέσματα του κεφαλαίου 5 εξάγουμε τα παρακάτω συμπεράσματα:

- Το φίλτρο Karatsuba παρουσιάζει σημαντικά μικρότερη καθυστέρηση από το συμβατικό φίλτρο και για τις τρείς διαφορετικές μορφές που υλοποιήθηκαν (direct, transposed και mixed).
- Η μεγαλύτερη ποσοστιαία μείωση της καθυστέρησης που προσφέρει η αρχιτεκτονική Karatsuba έναντι της συμβατικής εμφανίζεται στην mixed μορφή, δεύτερη έρχεται η transposed μορφή, με την direct μορφή να έρχεται τελευταία.
- Όσον αφορά την επιφάνεια, το φίλτρο Karatsuba παρουσιάζει ξανά καλύτερη απόδοση έναντι του συμβατικού φίλτρου, για όλες τις μορφές.
- Το μεγαλύτερο κέρδος σε επιφάνεια εμφανίζεται στην direct μορφή για μήκος λέξης $n$=32 bits. Όμως, κατά μέσο όρο η καλύτερη επίδοση της Karatsuba υλοποίησης έναντι της συμβατικής ως προς την επιφάνεια παρουσιάζεται στην transposed μορφή. Το φίλτρο Karatsuba σε mixed μορφή παρουσιάζει τα μικρότερα κέρδη σε επιφάνεια, κυρίως λόγω των αυξημένων αναγκών του σε μονάδες καθυστέρησης.
- Από τα αποτελέσματα της σύνθεσης των φίλτρων FIR προκύπτει ότι η ανωτερότητα του φίλτρου Karatsuba ως προς την επιφάνεια οφείλεται κυρίως στα συνδυαστικά κυκλώματα. Στην πραγματικότητα το φίλτρο Karatsuba χρησιμοποιεί περισσότερη

επιφάνεια σε καταχωρητές σε σχέση με το συμβατικό φίλτρο, όμως η καλύτερη απόδοση του ως προς τα συνδυαστικά κυκλώματα υπερκαλύπτει την διαφορά.

- Το φίλτρο Karatsuba σε direct και σε transposed μορφή, εμφανίζει σημαντικά μικρότερη κατανάλωση από το αντίστοιχο συμβατικό φίλτρο. Όσον αφορά την mixed μορφή, το φίλτρο Karatsuba είναι αποδοτικότερο ως προς την κατανάλωση μόνο για μήκος λέξης $n$=32 bits. Το γεγονός αυτό οφείλεται στην υψηλή κατανάλωση που έχει το φίλτρο Karatsuba σε mixed μορφή, κυρίως λόγω του αυξημένου αριθμού καταχωρητών που χρησιμοποιεί.
- Συγκρίνοντας τις τρείς μορφές των υλοποιήσεων ως προς την καθυστέρηση συμπεραίνουμε ότι οι υλοποιήσεις mixed μορφής παρουσιάζουν την μικρότερη καθυστέρηση, οι υλοποιήσεις transposed μορφής την δεύτερη μικρότερη, ενώ οι υλοποιήσεις direct μορφής την μεγαλύτερη καθυστέρηση.
- Όσον αφορά τις μετρικές *Area×Delay* και *Area×Delay*$^2$, οι υλοποιήσεις mixed μορφής αποδείχθηκαν ως οι αποδοτικότερες με τις υλοποιήσεις transposed και direct μορφής να ακολουθούν.
- Οι υλοποιήσεις transposed μορφής παρουσίασαν ως επί το πλείστον τις μεγαλύτερες τιμές *Power×Delay*, ενώ για τις υλοποιήσεις mixed και direct μορφής προέκυψαν παρόμοιες τιμές.
- Με μία γενικότερη ματιά επί των αποτελεσμάτων, μπορούμε να ισχυριστούμε ότι παρατηρούμε δύο κύριες τάσεις: Πρώτον, την αύξηση της αποδοτικότητας της αρχιτεκτονικής Karatsuba με την αύξηση του μήκους λέξης των δεδομένων εισόδου και των συντελεστών (bits). Δεύτερον, την μείωση της αποδοτικότητας της αρχιτεκτονικής Karatsuba με την αύξηση της τάξης του φίλτρου (taps), κυρίως λόγω της χρησιμοποίησης πολλών καταχωρητών.

## 6.2 Μελλοντικές επεκτάσεις

- Η υλοποίηση των κυκλωμάτων των φίλτρων FIR μπορεί να επεκταθεί και για περισσότερα μήκη λέξης για τα δεδομένα εισόδου και τους συντελεστές (bits), καθώς και για μεγαλύτερο αριθμό σημείων (taps).
- Η μελέτη των φίλτρων FIR θα μπορούσε να επεκταθεί σε ένα διάστημα συχνοτήτων λειτουργίας. Με αυτόν τον τρόπο θα δινόταν η δυνατότητα για την αξιολόγηση της

απόδοσης τους ως προς την επιφάνεια και την κατανάλωση σε ένα μεγαλύτερο εύρος τιμών της περιόδου του ρολογιού και όχι μόνο στην high performance περίοδο.

- Στην παρούσα εργασία, οι αρχιτεκτονικές των φίλτρων FIR υλοποιήθηκαν με βάση μια standard-cell CMOS βιβλιοθήκη της *Artisan* σε τεχνολογία 90*nm*. Η υλοποίηση των φίλτρων FIR θα μπορούσε να επεκταθεί και για άλλες τεχνολογίες (π.χ. 65*nm*).
- Θα μπορούσε να διερευνηθεί η υλοποίηση της αρχιτεκτονικής του φίλτρου FIR που βασίστηκε στον πολλαπλασιαστικό αλγόριθμο Karatsuba και σε τεχνολογία FPGA.

# 7

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] Kiamal Pekmestzi, “DIGITAL VLSI SYSTEMS”, NTUA Lectures Notes, Athens 2014.

[2] A. Karatsuba and Y. Ofman, “Multiplication of Multidigit Numbers on Automata,” Doklady Akademii Nauk SSSR, vol. 145, no. 2, pp. 293-294, 1962.

[3] P. Albicocco, G. C. Cardarilli, S. Pontarelli, M. Re, “Karatsuba Implementation of FIR filters,” in ASILOMAR, Pacific Groove, CA, 2012, pp.1111-1114

[4] C. Eyupoglu, “Performance Analysis of Karatsuba Multiplication Algorithm for Different Bit Lenghts,” Procedia – Social and Behavioral Sciences, vol. 195, pp. 1860-1864, Jul. 2015.

[5] W.-C. Yeh and C.-W. Jen, “High-speed Booth Encoded Parallel Multiplier Design,” IEEE Trans. Comput., vol. 49, no. 7, pp. 692-701, Jul. 2000.

[6] Neil H. Weste, Kamran Eshraghian, “Principles of CMOS VLSI Design: A Systems Perspective,” Fourth Edition.

[7] C.S. Wallace, “A Suggestion for a Fast Multiplier,” IEEE Trans. Comput., vol. EC-13, pp. 14-17, Feb. 1964.

[8] DesignWare Building Blocks Datasheet, available at: www.synopsys.com/dw/doc.php/doc/dwf/datasheets/dw02 mult.pdf