ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών

Υλοποίηση φίλτρων FIR με τεχνολογία ASIC

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΓΑΛΑΝΗ ΕΛΠΙΔΑΣ-ΑΘΗΝΑΣ

Επιβλέπων: Κιαμάλ Πεκμεστζή

Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση φίλτρων FIR με τεχνολογία ASIC

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΓΑΛΑΝΗ ΕΛΠΙΔΑΣ-ΑΘΗΝΑΣ

Επιβλέπων: Κιαμάλ Πεκμεστζή

Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 28η Ιουλίου 2015.

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

Αθήνα, Ιούλιος 2015

……
ΓΑΛΑΝΗ ΕΛΠΙΔΑ-ΑΘΗΝΑ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η σχεδίαση διαφόρων τοπολογιών φίλτρων και η διερεύνησή τους ως προς τις επιδόσεις τους στην επιφάνεια, τη καθυστέρηση και την κατανάλωση σε τεχνολογία ASIC.

Υλοποιήθηκαν τρεις τοπολογίες φίλτρου βασισμένες σε διαφορετικό συνδυασμό των δομικών μονάδων που απαιτούνται. Επιπρόσθετα, κάθε μια από αυτές τις τοπολογίες υλοποιήθηκε χρησιμοποιώντας τέσσερις διαφορετικές εκδοχές της μονάδας του πολλαπλασιαστή.

Πιο συγκεκριμένα, υλοποιήθηκαν σε γλώσσα Verilog η direct, η transpose και η mixed μορφή φίλτρου FIR για μήκη λέξεων 16 bits. Παράλληλα, κάθε μια από τις παραπάνω μορφές υλοποιήθηκε χρησιμοποιώντας τέσσερα διαφορετικά κυκλώματα ενός πολλαπλασιαστή. Το πρώτο σύμφωνα με τη συμβατική υλοποίηση πολλαπλασιαστή με Modified Booth προ-κωδικοποίηση, το δεύτερο σύμφωνα με την υλοποίηση πολλαπλασιαστή με NR4SD προ-κωδικοποίηση, το τρίτο και το τέταρτο κύκλωμα αποτελούν τις περικομμένες μορφές των δύο προηγούμενων, δηλαδή πολλαπλασιαστή σταθερού μήκους με Modified Booth προ-κωδικοποίηση και πολλαπλασιαστή σταθερού μήκους με NR4SD προ-κωδικοποίηση αντίστοιχα. Τα κυκλώματα προσομοιώθηκαν με βάση μια standard cell CMOS βιβλιοθήκη 90nm της Artisan. Τα αποτελέσματα καθυστέρησης, επιφάνειας και κατανάλωσης των παραπάνω περιπτώσεων συγκρίθηκαν μεταξύ τους για να επιλεγεί η βέλτιστη υλοποίηση.

Λέξεις Κλειδιά: Φίλτρα FIR, Direct μορφή φίλτρου, Transpose μορφή φίλτρου, Mixed μορφή φίλτρου, πολλαπλασιαστής με προ-κωδικοποίηση Modified Booth, πολλαπλασιαστής με προ-κωδικοποίηση NR4SD, truncated πολλαπλασιαστής (σταθερού μήκους), Verilog, CMOS 90nm

Abstract

The purpose of this thesis is to design different filter topologies and investigate their performance on the surface, the delay and the power consumption in ASIC technology.

Three filter topologies are implemented based on different combination of the structural units that are needed. Additionally, each of these topologies is implemented using four different versions of the multiplier unit.

More specifically, a direct, a transpose and a mixed form of FIR filter are implemented in Verilog language for inputs 16 bits. Furthermore, each of the above forms is implemented using four different multiplier circuits. The first circuit is the conventional implementation of Modified Booth pre-encoding multiplier, the second is the implementation of NR4SD pre-encoding multiplier, the third and the fourth circuits are the truncated forms of the previous two, i.e. a pre-encoded Modified Booth fixed-width multiplier and a pre-encoded NR4SD fixed-width multiplier respectively. Those filter topologies were simulated based on a standard cell CMOS 90nm library of Artisan. The delay, surface and consumption results of the above cases were compared to select the optimal design.

Keywords: Filter Finite Impulse Response (FIR), Direct filter, Transpose filter, Mixed filter, pre-encoded Modified Booth multiplier, pre-encoded NR4SD multiplier, truncated multiplier (fixed- width), Verilog, CMOS 90nm

Πίνακας περιεχομένων

1.ΕΙΣΑΓΩΓΗ

1.1 Ψηφιακή σχεδίαση

Η ψηφιακή σχεδίαση έχει ως αντικείμενο τη σχεδίαση ψηφιακών ηλεκτρονικών κυκλωμάτων και παρόλο που έχει λίγη διάρκεια ζωής η εξέλιξη της είναι ραγδαία. Είναι χρήσιμη σε πάρα πολλούς τομείς και πλέον υποστηρίζεται με σύνθετα σχεδιαστικά πακέτα λογισμικού. Μέχρι πρόσφατα τα εργαλεία αυτά ήταν αποκλειστικότητα των χωρών με βιομηχανία πυριτίου, τώρα όμως παρέχεται η δυνατότητα πρόσβασης και άλλων χωρών σε αυτά δεδομένου ότι έχει διαχωριστεί με σαφήνεια το σχεδιαστικό μέρος των συστημάτων VLSI , από το κατασκευαστικό τους. Σε αυτό έχει συμβάλλει και η τυποποίηση των γλωσσών περιγραφής κυκλωμάτων (VHDL, Verilog).

Η καθυστέρηση, η επιφάνεια και η κατανάλωση των ψηφιακών κυκλωμάτων αποτελούν τις παραμέτρους που χαρακτηρίζουν την ποιότητα της υλοποίησης. Σε κάποιες περιπτώσεις δίνεται σκόπιμα σε μια από τις παραπάνω παραμέτρους περισσότερο βάση ανάλογα με τις ανάγκες της υλοποίησης, εν γένει όμως η βέλτιστη υλοποίηση είναι εκείνη που συνδυάζει τις επιθυμητές τιμές και των τριών παραμέτρων.

Η ψηφιακή επεξεργασία σήματος (Digital Signal Processing, DSP) αποτελεί ένα τομέα εφαρμογής της ψηφιακής σχεδίασης και σχετίζεται με το μαθηματικό χειρισμό ενός σήματος και τους τρόπους επεξεργασίας του. Η επεξεργασία ήχου και σημάτων φωνής, τα συστήματα ραντάρ, οι αισθητήρες, η ψηφιακή επεξεργασία εικόνας, η επεξεργασία ιατρικών σημάτων και η επεξεργασία σημάτων στις τηλεπικοινωνίες είναι κάποια από τα πεδία εφαρμογών της ψηφιακής επεξεργασίας σήματος.

Το φίλτρο αποτελεί βασικό εργαλείο για την ψηφιακή επεξεργασία σήματος και για αυτό έχει ιδιαίτερο ενδιαφέρον ο σχεδιασμός όλο και αποδοτικότερων κυκλωμάτων υλοποίησής του.

1.2 Αντικείμενο διπλωματικής

Η παρούσα διπλωματική έχει ως στόχο τη διερεύνηση τριών τοπολογιών φίλτρων, κάθε μια εκ των οποίων υλοποιείται και με τέσσερις περιπτώσεις πολλαπλασιαστών, ώστε να επιλεχθεί η βέλτιστη.

Εν συντομία, σχεδιάστηκαν οι τρεις τοπολογίες φίλτρων FIR- direct, transpose και mixed- με τέσσερις διαφορετικές παραλλαγές της δομικής μονάδας του πολλαπλασιαστή. Στο πρώτο κύκλωμα πολλαπλασιαστή χρησιμοποιήθηκε η συμβατική υλοποίηση πολλαπλασιαστή με προ-κωδικοποίηση Modified Booth, στο δεύτερο κύκλωμα πολλαπλασιαστή χρησιμοποιήθηκε μια καινοτόμα προ-κωδικοποίηση, η NR4SD. Το τρίτο και το τέταρτο κύκλωμα πολλαπλασιαστή αποτελούν μια επέκταση της μελέτης μας αφού υλοποιούν τους δύο πρώτους πολλαπλασιαστές στην «περικομμένη» τους μορφή, δηλαδή είναι πολλαπλασιαστές σταθερού μήκους και μελετώνται λόγω της βελτίωσης που παρουσιάζουν στις παραμέτρους οι οποίες επηρεάζουν την απόδοση του κυκλώματος με επίπτωση όμως τη μείωση της ακρίβειας.

Στα πλαίσια της εργασίας μας, υλοποιήθηκαν σε γλώσσα Verilog τρεις διαφορετικές τοπολογίες φίλτρων που παρουσιάστηκαν παραπάνω με εισόδους των 16 bits. Η λειτουργία των κυκλωμάτων αυτών προσομοιώθηκε και επαληθεύτηκε με το περιβάλλον ModelSim. Στη συνέχεια πραγματοποιήθηκε σύνθεση των παραπάνω κυκλωμάτων από τον Synopsys Design Compiler με χρήση μιας standard cell βιβλιοθήκης της Artisan. Τα αποτελέσματα καθυστέρησης, επιφάνειας και κατανάλωσης εξάχθηκαν από το Synopsys Design Compiler και το Synopsys PrimePower.

1.3 Συνεισφορά διπλωματικής

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Υλοποιήθηκαν τρεις τοπολογίες φίλτρου –direct, transpose, mixed- για τις περιπτώσεις πολλαπλασιαστών MB και NR4SD πλήρους ακρίβειας καθώς και των πολλαπλασιαστών MB και NR4SD περικομμένης ακρίβειας.
2. Αξιολογήθηκαν τα πλεονεκτήματα και τα μειονεκτήματα κάθε υλοποίησης.
3. Αξιολογήθηκε η απόδοση και πιο συγκεκριμένα η καθυστέρηση, η επιφάνεια που απαιτείται και η κατανάλωση κάθε κυκλώματος καθώς και η ακρίβεια των αποτελεσμάτων.
4. Παρουσιάστηκαν τα συγκριτικά αποτελέσματα των διάφορων τοπολογιών και αξιολογήθηκε η επίδοση τους ως προς την επιφάνεια και την κατανάλωση τους στα 90nm.

1.4 Οργάνωση κειμένου

Στο κεφάλαιο 2 παρουσιάζεται το απαραίτητο θεωρητικό υπόβαθρο για το αντικείμενο της διπλωματικής.

Στο κεφάλαιο 3 αναλύονται οι υλοποιήσεις που εξετάστηκαν, με έμφαση στα δομικά τους στοιχεία.

Στο κεφάλαιο 4 παρουσιάζεται μια θεωρητική ανάλυση για τον υπολογισμό της επιφάνειας και της καθυστέρησης των υλοποιήσεων.

Στο κεφάλαιο 5 παρουσιάζονται τα πειραματικά αποτελέσματα για την επιφάνεια, τη καθυστέρηση και τη κατανάλωση της κάθε υλοποίησης. Επίσης, γίνεται σύγκριση των αποτελεσμάτων και τέλος μια σύνοψη της διπλωματικής μαζί με τα συμπεράσματα και πιθανές προεκτάσεις αυτής.

Στο κεφάλαιο 6 παρατίθεται η βιβλιογραφία που χρησιμοποιήθηκε.

2.ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 Κωδικοποιήσεις Booth και Modified Booth

2.1.1 Κωδικοποίηση Booth

Ένας αριθμός X σε συμπλήρωμα ως προς 2 παρίσταται από την ακόλουθη σχέση:

$$X = -x_{n-1} + \sum_{i=0}^{n-2} 2^i \cdot x_i$$

Μπορεί όμως να γραφεί και ισοδύναμα: X=2X – X

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: $z_0=0 - x_0$

$z_1= x_0 - x_1$

$z_2= x_1 - x_2$

.
.
.

$z_{n-2}= x_{n-3} - x_{n-2}$

Από τα παραπάνω μόνο το ψηφίο z_{n-1} αξίζει να αναλυθεί περαιτέρω και προκύπτει από τη σχέση:

$$z_{n-1} = -2x_{n-1} + x_{n-2} + x_{n-1} = x_{n-2} - x_{n-1}$$

Συνεπώς, ισχύει πάντα η σχέση $z_{n-1} = x_{n-2} - x_{n-1}$ και κατά επέκταση ο αριθμός X δίνεται από τη σχέση:

$$X = \sum_{i=0}^{n-1}(z_i \cdot 2^i)$$

Ο αλγόριθμος του Booth δίνει τη δυνατότητα πολλαπλασιασμού αριθμών σε μορφή συμπληρώματος ως προς δύο, ανεξάρτητα από το αν είναι θετικοί ή αρνητικοί.

Πιο συγκεκριμένα, έστω το γινόμενο $P=X \cdot Y$ όπου $Y=y_{n-1}y_{n-2}...y_1y_0$ και $X=x_{n-1}x_{n-2}...x_1x_0$. Για το γινόμενο, σύμφωνα με τη κωδικοποίηση Booth, ισχύει η εξής σχέση:

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

Η παραπάνω σχέση υποδεικνύει ότι σε κάθε βήμα i, το 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. Στην πραγματικότητα με την κωδικοποίηση Booth ελέγχουμε τις σειρές από μονάδες που παρουσιάζονται μέσα σε κάθε δυαδικό αριθμό. Παρατηρούμε ότι ενώ η άθροιση ξεκινάει από το 0 χρησιμοποιείται και ο δείκτης -1, η διαδικασία της κωδικοποίησης δηλαδή ξεκινάει με y_{-1}=0. Στη συνέχεια, με φορά από τα δεξιά προς τα αριστερά, εξετάζουμε τα ψηφία ανά δύο σύμφωνα με τον πίνακα 2.1.1. Τα ζευγάρια των bits που εξετάζουμε δεν είναι ανεξάρτητα αλλά έχουν επικάλυψη ενός ψηφίου, έτσι σε κάθε έλεγχο μόνο ένα bit του πολλαπλασιαστή μένει απ'έξω.

Y_i	Y_{i-1}	**Κωδικοποιημένα Ψηφία ($Y_{i-1} - Y_i$)**	**Λειτουργία**
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 2.1.1 *Λειτουργίες κωδικοποίησης Booth*

Ανακεφαλαιώνοντας, η λειτουργία του αλγορίθμου συνοψίζεται στα ακόλουθα:

- Αν Y_iY_{i-1} =00 ή Y_iY_{i-1} =11, τότε ολίσθησε το τρέχον άθροισμα 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 μια θέση προς τα αριστερά.
- Αν 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.

Το γεγονός ότι η κωδικοποίηση θετικών και αρνητικών αριθμών είναι η ίδια αποτελεί βασικό πλεονέκτημα της κωδικοποίησης Booth. Επίσης, άλλο ένα πλεονέκτημα είναι ότι επειδή το κάθε ψηφίο του κωδικοποιημένου αριθμού είναι συνάρτηση μόνο των δύο bit Y_iY_{i-1} μπορεί να παραχθεί αμέσως, ανεξάρτητα από τα προηγούμενα μερικά γινόμενα. Τέλος, το σημαντικότερο πλεονέκτημα είναι το γεγονός ότι μπορούμε με την μετατροπή του δυαδικού αριθμού να μειώσουμε τον αριθμό των προσθέσεων που απαιτούνται για τον πολλαπλασιασμό. [1]

2.1.2 Κωδικοποίηση Modified Booth

Ο τροποποιημένος αλγόριθμος του Booth επεκτείνει το σύνολο των ψηφίων κωδικοποίησης και προκύπτει αλγεβρικά από τον απλό αλγόριθμο Booth.

Ένας δυαδικός αριθμός Y σε κωδικοποίηση Booth, όπως προαναφέρθηκε, δίνεται από τη σχέση:

$$Y = \sum_{i=0}^{n-1}(z_i \cdot 2^i)$$

Η σχέση αυτή μπορεί να αναλυθεί ως εξής:

$$Y = \sum_{i=0}^{n-1} z_i 2^i = \sum_{j=0}^{\frac{n}{2}-1} z_{2j}2^{2j} + z_{2j+1}2^{2j+1} = \sum_{j=0}^{\frac{n}{2}-1}(z_{2j} + 2\,z_{2j+1})2^{2j} = \sum_{j=0}^{\frac{n}{2}-1} {b_j}^{MB}4^j$$

όπου το ψηφίο z_i είναι το ψηφίο του Y σε απλή Booth κωδικοποίηση ενώ το ${b_j}^{MB}$ το ψηφίο κωδικοποιημένο σύμφωνα με τον τροποποιημένο αλγόριθμο Booth και μπορεί να παρασταθεί ως εξής:

$${b_i}^{MB} = 2z_{i+1} + z_i = y_i - 2y_{i+1} + y_{i-1}$$

Σε αυτό τον αλγόριθμο δεν κωδικοποιούμε πλέον ζευγάρια ψηφίων αλλά τρία bits μαζί. Στον πίνακα 2.1.2 φαίνεται η αντιστοίχιση των δυαδικών ψηφίων κωδικοποιημένων σε απλό Booth και κωδικοποιημένων σε Modified Booth.

y_{i+1}	y_i	y_{i-1}	***Κωδικοποίηση Booth*** z_{i+1}	z_i	***Κωδικοποίηση Modified Booth*** ${b_j}^{MB}$
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 2.1.2 *Λειτουργία κωδικοποίησης Modified Booth*

Όπως φαίνεται από τον πίνακα 2.2.1., το σύνολο της αναφοράς τώρα είναι {-2, -1, 0, +1, +2}. Για να βρεθεί ο δεκαδικός αριθμός από τα κωδικοποιημένα ψηφία του, πολλαπλασιάζονται τα ψηφία αυτά με το βάρος που έχουν, όπως φαίνεται από τη σχέση:

$$Y = \sum_{j=0}^{\frac{n}{2}-1} {b_j}^{MB}4^j$$

Κάθε κωδικοποιημένο ψηφίο καθορίζει την λειτουργία που πρόκειται να γίνει στον πολλαπλασιαστέο.

Για παράδειγμα, έστω ότι πρόκειται να πολλαπλασιάσουμε δύο αριθμούς Α και Β, όπου Β ο πολλαπλασιαστής. Κωδικοποιούμε τα ψηφία του Β σύμφωνα με τον πίνακα 2.1.2 και με βάση τον πίνακα 2.1.3 επιλέγονται οι λειτουργίες που καθορίζουν τα ψηφία αυτά.

Κωδικοποιημένα Ψηφία	**Λειτουργία**
0	Πρόσθεσε το 0 στο μερικό γινόμενο
+1	Πρόσθεσε το (A) στο μερικό γινόμενο
+2	Πρόσθεσε το (2A) στο μερικό γινόμενο
-2	Αφαίρεσε το (2A) στο μερικό γινόμενο
-1	Αφαίρεσε το (A) στο μερικό γινόμενο

Πίνακας 2.1.3 *Λειτουργίες κωδικοποίησης Modified Booth*

Πιο συγκεκριμένα, ο πολλαπλασιαστής Β μπορεί να αναπαραστεί σε MB μορφή ως εξής:

$$B = \sum_{j=0}^{k-1} b_j^{MB} 2^{2j}$$

Τα ψηφία b_j^{MB} ανήκουν στο σύνολο {-2, -1, 0, +1, +2}, 0 ≤ j ≤ k-1, και αντιστοιχίζονται σε τρία συνεχόμενα ψηφία b_{2j+1}, b_{2j} και b_{2j-1}, με ένα bit να επικαλύπτεται και b_{-1}=0. Παίρνουν τις τιμές του πίνακα 2.1.4.

Binary			***MB Digit***	***MB Encoding***		
b_{2j+1}	b_{2j}	b_{2j-1}	b_j^{MB}	$sign=s_j$	$x1=one_j$	$x2=two_j$
0	0	0	0	0	0	0
0	0	1	+1	0	1	0
0	1	0	+1	0	1	0
0	1	1	+2	0	0	1
1	0	0	-2	1	0	1
1	0	1	-1	1	1	0
1	1	0	-1	1	1	0
1	1	1	0	1	0	0

Πίνακας 2.1.4 *Κωδικοποίηση Modified Booth*

Κάθε ψηφίο b_j^{MB} αναπαρίσταται από τα τρία bits s, one, two. Το bit προσήμου s δείχνει αν το ψηφίο είναι αρνητικό (s=1) ή θετικό (s=0). Το one δείχνει αν η απόλυτη τιμή του ψηφίου είναι ίση με το 1 (one=1) ή όχι (one=0). Το two δείχνει αν η απόλυτη τιμή του ψηφίου είναι ίση με το 2 (two=1) ή όχι (two=0). Χρησιμοποιώντας αυτά τα bits, το ψηφίο b_j^{MB} υπολογίζεται από τη σχέση:

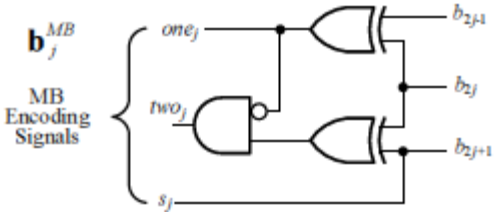
$$b_j^{MB} = (-1)^{s_j} \cdot (one_j + 2two_j)$$

Όπως φαίνεται στον πίνακα 2.1.4, το s bit ισούται με το πιο σημαντικό bit από τα τρία bits που χρησιμοποιούνται στην κωδικοποίηση. Οι ακόλουθες σχέσεις δείχνουν πως προκύπτουν τα MB κωδικοποιημένα σήματα και το σχήμα 2.1.1 δείχνει την υλοποίηση τους με πύλες.

$$one_j = b_{2j-1} \oplus b_{2j}$$

$$two_j = \left(b_{2j+1} \oplus b_{2j}\right) \Lambda \overline{one_j}$$

$$s_j = b_{2j+1}$$



Σχήμα 2.1.1 *Υλοποίηση των σημάτων MB κωδικοποίησης*

Και ο τροποποιημένος αλγόριθμος του Booth έχει το πλεονέκτημα ότι εφαρμόζεται ανεξάρτητα από το αν οι αριθμοί είναι σε απλή δυαδική ή σε μορφή συμπληρώματος ως προς 2. Επίσης, έχουμε λιγότερες προσθέσεις να εκτελέσουμε γιατί έχουμε μείωση του αριθμού των μερικών γινομένων, το οποίο συμβάλει τόσο στην αύξηση της ταχύτητας του πολλαπλασιαστή όσο και στη μείωση της επιφάνειας που καταλαμβάνει. Παρόλα αυτά, όμως, έχουμε και αύξηση της πολυπλοκότητας του κυκλώματος γιατί πρέπει να υπάρχει πρόβλεψη για τον υπολογισμό των +2A, -2A και –A. [1][2]

2.2 Κωδικοποιήσεις NR4SD

2.2.1 Non-Redundant Radix-4 Signed-Digit Algorithm Minus

Έστω το γινόμενο P=A·B όπου B=$b_{k-1}b_{k-2}…b_1b_0$ και A=$a_{k-1}a_{k-2}…a_1a_0$ σε μορφή συμπληρώματος ως προς δυο. Ο πολλαπλασιαστής B θα κωδικοποιηθεί σύμφωνα με τον NR4SD αλγόριθμο και τα ψηφία b_j^{NR} θα έχουν σύνολο αναφοράς το {-2, -1, 0, +1} αν χρησιμοποιηθεί ο NR4SD$^-$ αλγόριθμος ή το σύνολο {-1, 0, +1, +2} αν χρησιμοποιηθεί ο NR4SD$^+$. Δηλαδή σε κάθε βήμα j, το A πολλαπλασιάζεται με ένα από τα στοιχεία του συνόλου {-2, -1, 0, +1} για τον NR4SD$^-$ αλγόριθμο ή του συνόλου {-1, 0, +1, +2} για τον NR4SD$^+$, με 0 ≤ j ≤k-2. Για j=k-1 το A πολλαπλασιάζεται με ένα ψηφίο b_{k-1}^{MB} από το σύνολο {-2, -1, 0, +1, +2} της κωδικοποίησης Modified Booth ώστε να καλυφθεί το δυναμικό εύρος της αναπαράστασης σε συμπλήρωμα ως προς δύο.

Πιο συγκεκριμένα, ο αλγόριθμος NR4SD$^-$ ακολουθεί τα ακόλουθα βήματα.

Βήμα 1: Ορίζονται αρχικές τιμές j=0 και c_0=0.

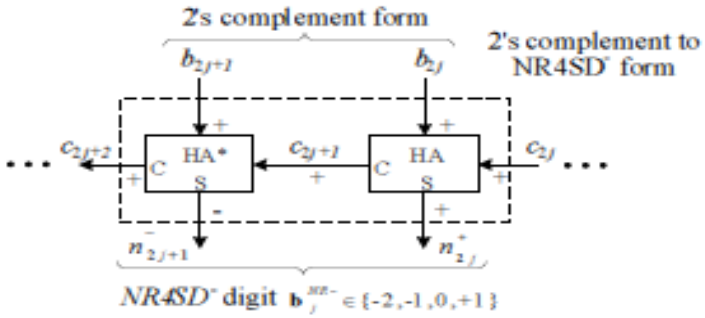
Βήμα 2: Υπολογίζονται το κρατούμενο c_{2j+1} και το άθροισμα n_{2j}^+ , που αποτελούν τις εξόδους ενός ημιαθροιστή (HA) με εισόδους το b_{2j} και c_{2j} (Σχήμα 2.2.1).

$$n_{2j}^+ = b_{2j} \oplus c_{2j}$$

$$c_{2j+1} = b_{2j} \wedge c_{2j}$$

Βήμα 3: Υπολογίζονται το κρατούμενο c_{2j+2} (θετικού βάρους) και το άθροισμα n_{2j+1}^- (αρνητικού βάρους), που αποτελούν τις εξόδους ενός ημιαθροιστή* (HA*) με εισόδους b_{2j+1} και c_{2j+1} (Σχήμα 2.2.1). Η σχέση που έχουν οι είσοδοι και οι έξοδοι ενός HA* και δείχνει ότι το κρατούμενο c_{2j+2} είναι θετικού βάρους ενώ το άθροισμα n_{2j+1}^- είναι αρνητικού βάρους, είναι η εξής:

$$2c_{2j+2} - n_{2j+1}^- = b_{2j+1} + c_{2j+1}$$



Σχήμα 2.2.1 *Σχηματικό διάγραμμα της NR4SD$^-$ κωδικοποίησης σε επίπεδο ψηφίου*

Ο HA* υλοποιεί τις ακόλουθες λογικές συναρτήσεις:

$$n_{2j+1}^{-} = b_{2j+1} \oplus c_{2j+1} = b_{2j+1} \oplus (b_{2j} \wedge c_{2j})$$

$$c_{2j+2} = b_{2j+1} \vee c_{2j+1} = b_{2j+1} \vee (b_{2j} \wedge c_{2j})$$

και έχει τον ακόλουθο πίνακα αληθείας:

Inputs		Outputs	
b_{2j+1}	c_{2j+1}	c_{2j+2}	n_{2j+1}^{-}
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

Πίνακας 2.2.1 Πίνακας αληθείας HA*

Βήμα 4: Υπολογίζεται το ψηφίο b_j^{NR-} από την σχέση:

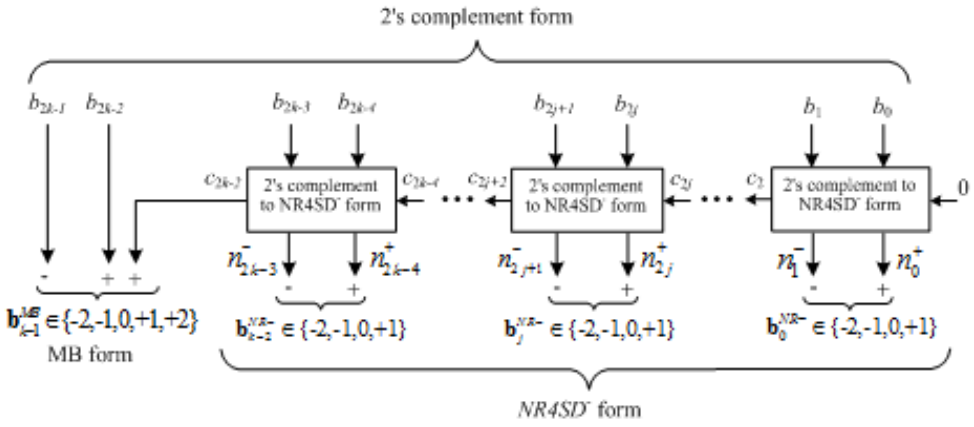
$$b_j^{NR-} = -2n_{2j+1}^{-} + n_{2j}^{+}$$

Η παραπάνω εξίσωση προκύπτει από το γεγονός ότι το n_{2j}^{+} είναι θετικού βάρους και το n_{2j+1}^{-} αρνητικού βάρους.

Βήμα 5: j=j+1.

Βήμα 6: Αν (j<k-1) τότε μετάβαση στο *Βήμα 2.*

Αν (j=k-1) τότε κωδικοποιείται το τελευταίο και πιο σημαντικό bit του πολλαπλασιαστή σύμφωνα με τον αλγόριθμο Modified Booth λαμβάνοντας υπόψη ότι τα τρία συνεχόμενα ψηφία είναι το b_{2k-1}, b_{2k-2}, c_{2k-2} (Σχήμα 2.2.2).



Σχήμα 2.2.2 *Σχηματικό διάγραμμα της NR4SD⁻ κωδικοποίησης σε επίπεδο λέξης*

Βήμα 7: Αν (j=k) τότε σταματάει ο αλγόριθμος.

Ο ακόλουθος πίνακας δείχνει πως τα NR4SD$^-$ ψηφία προκύπτουν ανάλογα με το συνδυασμό των εισόδων.

2's complement			***NR4SD$^-$ form***			***Digit***	***NR4SD$^-$ encoding***		
b_{2j+1}	b_{2j}	c_{2j}	c_{2j+2}	n_{2j+1}^-	n_{2j}^+	b_j^{NR-}	one_j^+	one_j^-	two_j^-
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	+1	1	0	0
0	1	0	0	0	1	+1	1	0	0
0	1	1	1	1	0	-2	0	0	1
1	0	0	1	1	0	-2	0	0	1
1	0	1	1	1	1	-1	0	1	0
1	1	0	1	1	1	-1	0	1	0
1	1	1	1	0	0	0	0	0	0

Πίνακας 2.2.2 *Κωδικοποίηση NR4SD$^-$*

Από τον παραπάνω πίνακα αληθείας προκύπτουν οι εξής εξισώσεις των κωδικοποιημένων σημάτων one_j^+, one_j^-, two_j^- :

$$one_j^+ = \overline{n_{2j+1}^-} \wedge n_{2j}^+$$

$$one_j^- = n_{2j+1}^- \wedge n_{2j}^+$$

$$two_j^- = n_{2j+1}^- \wedge \overline{n_{2j}^+}$$

Τα ελάχιστα και τα μέγιστα όρια του δυναμικού εύρους της NR4SD$^-$ μορφής είναι

$$-2^{n-1} - 2^{n-3} - 2^{n-5} - \cdots - 2 < -2^{n-1}$$

και

$$2^{n-1} + 2^{n-4} + 2^{n-6} + \cdots + 1 > 2^{n-1} - 1$$

αντίστοιχα. Όπως παρατηρείται από τις παραπάνω σχέσεις το δυναμικό εύρος της NR4SD$^-$ μορφής αναπαράστασης είναι μεγαλύτερο από το δυναμικό εύρος της μορφής συμπληρώματος ως προς δύο. [2]

2.2.2 Non-Redundant Radix-4 Signed-Digit Algorithm Plus

Αντίστοιχα υλοποιείται και η NR4SD$^+$ κωδικοποίηση. Αναλυτικότερα, τα βήματα του αλγορίθμου είναι τα ακόλουθα:

Βήμα 1: Ορίζονται αρχικές τιμές j=0 και c_0=0.

Βήμα 2: Υπολογίζονται το κρατούμενο c_{2j+1} (θετικού βάρους) και το άθροισμα n_{2j}^- (αρνητικού βάρους), τα οποία παράγονται από έναν HA* με εισόδους b_{2j} και c_{2j} (Σχήμα 2.3.3). Η σχέση μεταξύ των εισόδων και των εξόδων του HA* είναι:

$$2c_{2j+1} - n_{2j}^- = b_{2j} + c_{2j}$$

Και σύμφωνα με τον Πίνακα 2.3.1, προκύπτουν οι εξής συναρτήσεις:

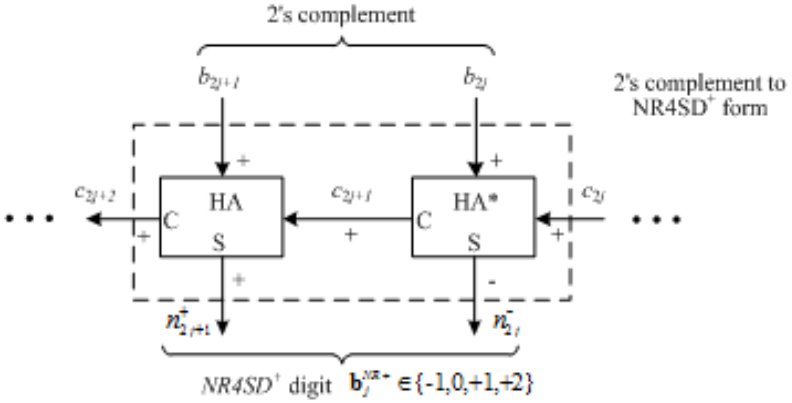
$$n_{2j}^- = b_{2j} \oplus c_{2j}$$

$$c_{2j+1} = b_{2j} \vee c_{2j}$$

Βήμα 3: Υπολογίζονται το κρατούμενο c_{2j+2} και το άθροισμα n_{2j+1}^+, τα οποία είναι οι έξοδοι ενός ημιαθροιστή HA με εισόδους b_{2j+1} και c_{2j+1} (Σχήμα 2.2.3).

$$n_{2j+1}^+ = b_{2j+1} \oplus c_{2j+1} = b_{2j+1} \oplus (b_{2j} \vee c_{2j})$$

$$c_{2j+2} = b_{2j+1} \wedge c_{2j+1} = b_{2j+1} \wedge (b_{2j} \vee c_{2j})$$



Σχήμα 2.2.3 *Σχηματικό διάγραμμα της NR4S $^+$ κωδικοποίησης σε επίπεδο ψηφίου*

Βήμα 4: Υπολογίζεται η τιμή του ψηφίου b_j^{NR+} .

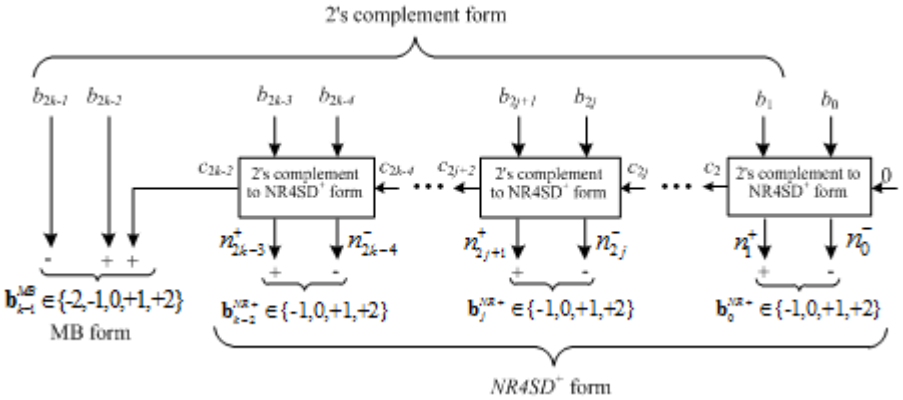
$$b_j^{NR+} = 2n_{2j+1}^+ - n_{2j}^-$$

Η παραπάνω εξίσωση προκύπτει από το γεγονός ότι το n_{2j+1}^+ είναι θετικού βάρους και το n_{2j}^- είναι αρνητικού βάρους.

Βήμα 5: j=j+1.

Βήμα 6: Αν (j<k-1) τότε μετάβαση στο *Βήμα 2.*

Αν (j=k-1) τότε κωδικοποιείται το τελευταίο και πιο σημαντικό bit του πολλαπλασιαστή σύμφωνα με τον αλγόριθμο Modified Booth λαμβάνοντας υπόψη ότι τα τρία συνεχόμενα ψηφία είναι το b_{2k-1}, b_{2k-2}, c_{2k-2} (Σχήμα 2.2.4).



Σχήμα 2.2.4 *Σχηματικό διάγραμμα της NR4SD⁺ κωδικοποίησης σε επίπεδο λέξης*

Βήμα 7: Αν (j=k) τότε σταματά ο αλγόριθμος.

Ο πίνακας 2.2.3 δείχνει πως προκύπτουν τα NR4SD⁺ ψηφία για κάθε συνδυασμό εισόδου.

2's complement			***NR4SD⁺ form***			***Digit***	***NR4SD⁺ encoding***		
b_{2j+1}	b_{2j}	c_{2j}	c_{2j+2}	n_{2j+1}^+	n_{2j}^-	b_j^{NR+}	one_j^+	one_j^-	two_j^+
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	+1	1	0	0
0	1	0	0	1	1	+1	1	0	0
0	1	1	0	1	0	+2	0	0	1
1	0	0	0	1	0	+2	0	0	1
1	0	1	1	0	1	-1	0	1	0
1	1	0	1	0	1	-1	0	1	0
1	1	1	1	0	0	0	0	0	0

Πίνακας 2.2.3 *Κωδικοποίηση NR4SD⁺*

Οι ακόλουθες εξισώσεις δείχνουν πως παράγονται τα NR4SD⁺ κωδικοποιημένα σήματα one_j^+, one_j^-, two_j^+.

$$one_j^+ = n_{2j+1}^+ \wedge n_{2j}^-$$

$$one_j^- = \overline{n_{2j+1}^+} \wedge n_{2j}^-$$

$$two_j^+ = n_{2j+1}^+ \wedge \overline{n_{2j}^-}$$

Τα ελάχιστα και μέγιστα όρια του δυναμικού εύρους της NR4SD^{+} αναπαράστασης είναι:

$$-2^{n-1} - 2^{n-4} - 2^{n-6} - \cdots - 1 < -2^{n-1}$$

και

$$2^{n-1} + 2^{n-3} + 2^{n-5} + \cdots + 2 > 2^{n-1} - 1$$

αντίστοιχα. Και εδώ παρατηρείται ότι το δυναμικό εύρος της NR4SD^{+} , όπως και της NR4SD^{-} , μορφής αναπαράστασης είναι μεγαλύτερο από το δυναμικό εύρος της μορφής συμπληρώματος ως προς δύο.

Με απώτερο σκοπό την καλύτερη κατανόηση των NR4SD^{-} και NR4SD^{+} τεχνικών κωδικοποίησης, ακολουθεί ένα παράδειγμα. Έστω ότι έχουμε ένα 4-bit αριθμό N σε μορφή συμπληρώματος ως προς δύο. Ο πίνακας 2.2.4 παρουσιάζει τον N, ο οποίος παίρνει ως τιμές τα όρια ($-2^8 = -128$, $2^8 - 1 = 127$) και άλλες δύο χαρακτηριστικές τιμές, καθώς επίσης και τα αντίστοιχα ψηφία του σε MB, NR4SD^{-}, NR4SD^{+} κωδικοποιήσεις μετά από εφαρμογή των αντίστοιχων αλγορίθμων που περιγράφηκαν παραπάνω. Οι αρνητικές τιμές έχουν μια γραμμή από πάνω για να ξεχωρίζουν από τις θετικές. [2]

2's complement	1000 0000	1001 1010	0101 1001	0111 1111
Integer	-128	-102	+89	+127
Modified Booth	$\overline{2}\,0\,0\,0$	$\overline{2}\,2\,\overline{1}\,\overline{2}$	$1\,2\,\overline{2}\,1$	$2\,0\,0\,\overline{1}$
NR4SD^{-}	$\overline{2}\,0\,0\,0$	$\overline{1}\,\overline{2}\,\overline{1}\,\overline{2}$	$2\,\overline{2}\,\overline{2}\,1$	$2\,0\,0\,\overline{1}$
NR4SD^{+}	$\overline{2}\,0\,0\,0$	$\overline{2}\,1\,2\,2$	$1\,1\,2\,1$	$2\,0\,0\,\overline{1}$

Πίνακας 2.2.4 *Αριθμητικά παραδείγματα των τεχνικών κωδικοποίησης*

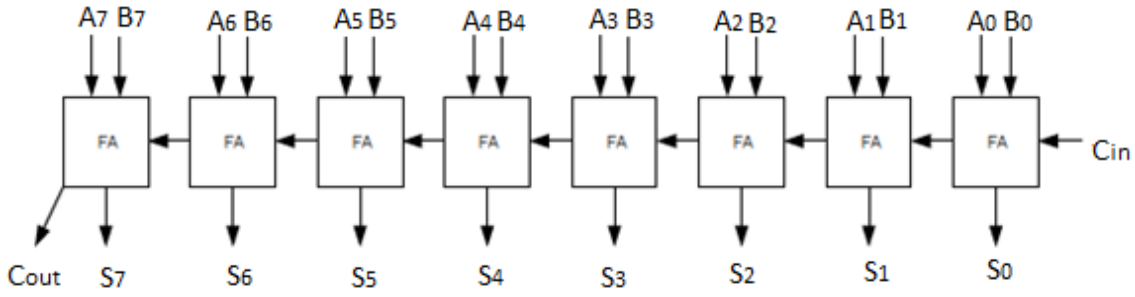
2.3 Είδη αθροιστών

2.3.1 Αθροιστής διάδοσης κρατουμένου (Carry Propagate Adder)

Ο αθροιστής διάδοσης κρατουμένου (Carry Propagate Adder) πραγματοποιεί την πράξη της πρόσθεσης με βάση τον αλγόριθμο που ξέρουμε για την πρόσθεση δεκαδικών αριθμών, δηλαδή προσθέτει δύο ψηφία ίσου βάρους και παράγει ένα ψηφίο ίσου βάρους, το οποίο είναι το άθροισμά τους, καθώς και ένα κρατούμενο με το αμέσως μεγαλύτερο βάρος.

Κυκλωματικά, αποτελείται από ένα δίκτυο πλήρων αθροιστών συνδεδεμένων σε σειρά. Κάθε πλήρης αθροιστής αντιστοιχεί σε μία βαθμίδα συγκεκριμένου βάρους. Το κρατούμενο εξόδου C_{out} του κάθε πλήρους αθροιστή συνδέεται με το κρατούμενο εισόδου C_{in} του πλήρους αθροιστή της επόμενης βαθμίδας, και το αποτέλεσμα της πράξης αποτελείται από τα save ψηφία όλων των πλήρων αθροιστών, καθώς και το κρατούμενο εξόδου της πιο σημαντικής βαθμίδας. Με τον τρόπο αυτό τα κρατούμενα διαδίδονται μέσα στη δομή του αθροιστή από τη χαμηλότερη μέχρι την υψηλότερη βαθμίδα, για το λόγο αυτό άλλωστε ονομάστηκε αθροιστής διάδοσης κρατουμένου.

Ακολουθεί το κύκλωμα αθροιστή διάδοσης κρατουμένου 8 bits (Σχήμα 2.3.1), το οποίο μπορεί να δέχεται ως είσοδο είτε δύο δυαδικούς αριθμούς είτε έναν Carry-Save αριθμό και δίνει έξοδο ένα δυαδικό αριθμό, ο οποίος αποτελείται από τα ψηφία {C_{out}, S_7, S_6, S_5, S_4, S_3, S_2, S_1, S_0}.



Σχήμα 2.3.1 *Αθροιστής διάδοσης κρατουμένου 8 bits*

Το παραπάνω κύκλωμα μπορεί να λειτουργήσει και για δυαδικούς αριθμούς σε μορφή συμπληρώματος ως προς δύο, αν προστεθεί έλεγχος για τις περιπτώσεις υπερχείλισης ή αν απλά αντιστραφούν τα ψηφία αρνητικής αξίας {A_7, B_7, C_{out}}.

Αν και ο αθροιστής διάδοσης κρατουμένου έχει πολύ απλή δομή και πολύ μικρή επιφάνεια, η διάδοση του κρατουμένου τον καθιστά απαγορευτικά αργό για τις

περισσότερες εφαρμογές καθώς η καθυστέρηση (critical path) ενός N-bit αθροιστή είναι N επίπεδα πλήρων αθροιστών. [1]

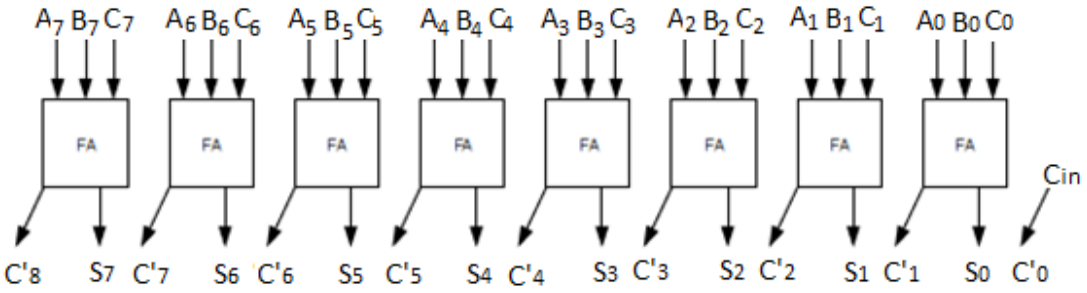
2.3.2 Αθροιστής σωσίματος κρατουμένου (Carry Save Adder)

Ο αθροιστής σωσίματος κρατουμένου (Carry Save Adder) λειτουργεί παρόμοια με τον αθροιστή διάδοσης κρατουμένου, όμως δεν παράγει έναν δυαδικό αριθμό ως αποτέλεσμα, αντί αυτού διατηρεί το αποτέλεσμα σε μορφή Carry-Save όπως προκύπτει από τις επιμέρους αθροίσεις κάθε βαθμίδας.

Η δομή του αθροιστή σωσίματος κρατουμένου είναι όμοια με τον αθροιστή διάδοσης κρατουμένου, με τη διαφορά ότι το κρατούμενο εξόδου (C_{out}) κάθε πλήρους αθροιστή δεν είναι είσοδος σε κάποιον άλλον, αλλά αποτελεί αποτέλεσμα στη συγκεκριμένη βαθμίδα, μαζί με το αντίστοιχο Save ψηφίο.

Ακολουθεί ένα κύκλωμα αθροιστή σωσίματος κρατουμένου των 8 bits (Σχήμα2.3.2), το οποίο δέχεται ως είσοδο είτε τρεις δυαδικούς αριθμούς είτε ένα δυαδικό και έναν Carry-Save αριθμό και δίνει έξοδο ένα Carry-Save αριθμό που αποτελείται από τα ψηφία:

C={C'_8, C'_7, C'_6, C'_5, C'_4, C'_3, C'_2, C'_1, C'_0} και S={ S_7, S_6, S_5, S_4, S_3, S_2, S_1, S_0}



Σχήμα 2.3.2 *Αθροιστής σωσίματος κρατουμένου 8 bits*

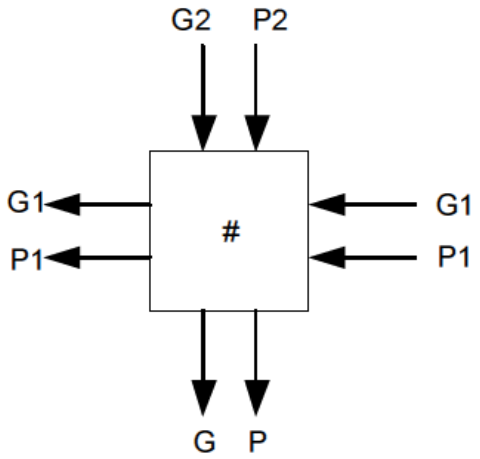
Το παραπάνω κύκλωμα μπορεί να λειτουργήσει και για δυαδικούς αριθμούς σε μορφή συμπληρώματος ως προς δύο, αν αντιστραφούν τα ψηφία αρνητικής αξίας {A_7, B_7, C_7, S_7, C'_8 }. Επίσης, ονομάζεται και ως πλήρης αθροιστής 3-2 συμπιεστής.

Ο αθροιστής σωσίματος κρατουμένου έχει απλή δομή και την ίδια επιφάνεια με τον αθροιστή διάδοσης κρατουμένου. Επίσης, μπορεί να χειριστεί τρείς αντί για δύο δυαδικούς αριθμούς ταυτόχρονα. Το μεγάλο πλεονέκτημά του όμως είναι ότι η καθυστέρηση (critical path) ενός N-bit αθροιστή είναι μόνο ένα επίπεδο πλήρους αθροιστή. Το βασικό μειονέκτημά του είναι ότι το αποτέλεσμα δεν βρίσκεται σε δυαδική μορφή, οπότε χρειάζεται περαιτέρω επεξεργασία. [1]

2.3.3 Αθροιστής πρόβλεψης κρατουμένου (Carry Look-Ahead Adder)

Ο αθροιστής πρόβλεψης κρατουμένου (Carry Look-ahead Adder) ακολουθεί μία διαφορετική φιλοσοφία άθροισης ώστε να αποφύγει την καθυστέρηση από την διάδοση κρατουμένου. Χρησιμοποιεί κυκλώματα ημιαθροιστών για τη δημιουργία των σημάτων P και G, κυκλώματα πρόβλεψης κρατουμένου (#) για την πρόβλεψη του τελικού κρατουμένου και ένα τελευταίο στάδιο που είναι η γεννήτρια του τελικού αθροίσματος, η οποία στην απλοποιημένη περίπτωση που δεν έχουμε κρατούμενο εισόδου, αποτελείται μόνο από N πύλες αποκλειστικού-OR (XOR), όπου N το πλήθος των ψηφίων του αθροιστή.

Το κύκλωμα πρόβλεψης κρατουμένου αποτελεί βασικό δομικό στοιχείο της γεννήτριας κρατούμενου, η οποία είναι η «καρδιά» του αθροιστή πρόβλεψης κρατουμένου. Δέχεται ως εισόδους τα σήματα G1, P1, G2 και P2, τα οποία είναι τα σήματα γέννησης (G, Generation) και διάδοσης (P, propagation) κρατουμένου δύο βαθμίδων διαφορετικής αξίας, ενώ δίνει ως έξοδο τα σήματα G και P, που αποτελούν δύο νέα σήματα γεννήσεως και διάδοσης κρατουμένου, στη μεγαλύτερη από τις δύο βαθμίδες. Το κύκλωμα της πρόβλεψης κρατουμένου είναι το εξής:



Σχήμα 2.3.3 *Κύκλωμα πρόβλεψης κρατουμένου*

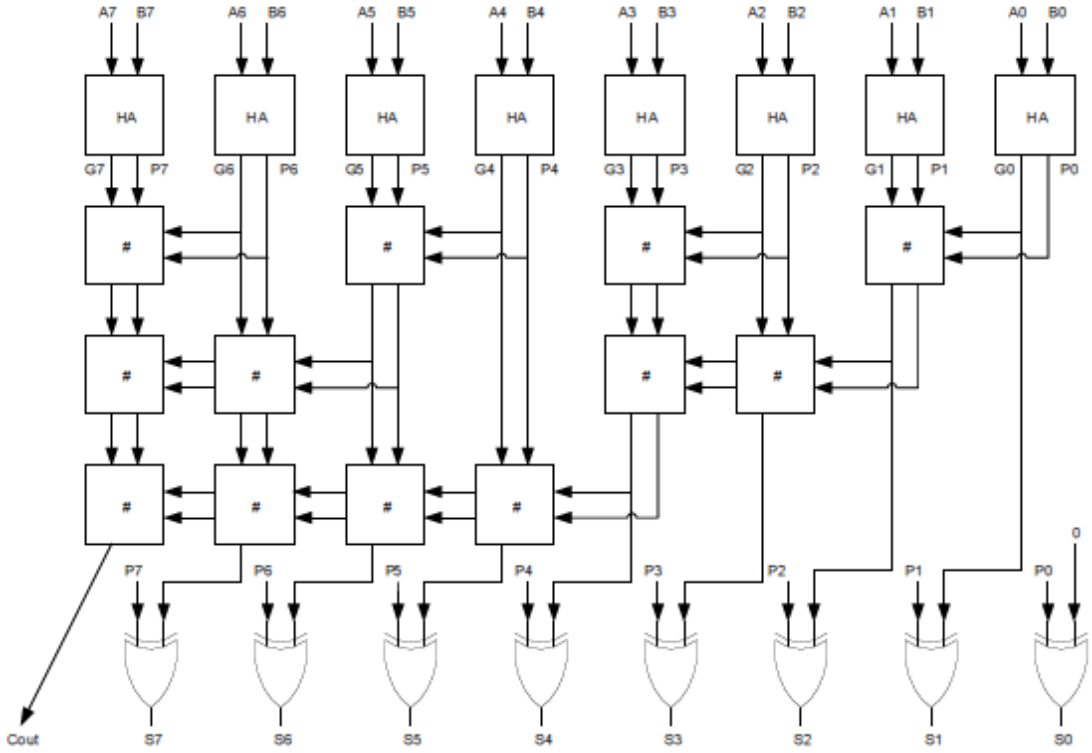
Και ικανοποιεί τις ακόλουθες σχέσεις μεταξύ εισόδων, εξόδων:

$$G = G_2 + (G_1 \cdot P_2)$$

$$P = P_1 \cdot P_2$$

Το κύκλωμα δημιουργίας κρατουμένου, εκμεταλλεύεται την προσεταιριστικότητα του τελεστή # για να υπολογίσει παράλληλα τα τελικά κρατούμενα. Αυτό επιτυγχάνεται με την διάταξη των κυκλωμάτων πρόβλεψης κρατουμένου σε μια δενδρική δομή, και επομένως τη παραγωγή των τελικών αποτελεσμάτων με καθυστέρηση logN κυκλωμάτων #.

Ακολουθεί ένα κύκλωμα αθροιστή πρόβλεψης κρατουμένου 8bits χωρίς κρατούμενο εισόδου (Σχήμα 2.3.4), το οποίο δέχεται ως είσοδο είτε δύο δυαδικούς αριθμούς είτε έναν Carry-Save αριθμό και δίνει έξοδο ένα δυαδικό αριθμό που αποτελείται από τα ψηφία {C_{out}, S_7, S_6, S_5, S_4, S_3, S_2, S_1, S_0}.



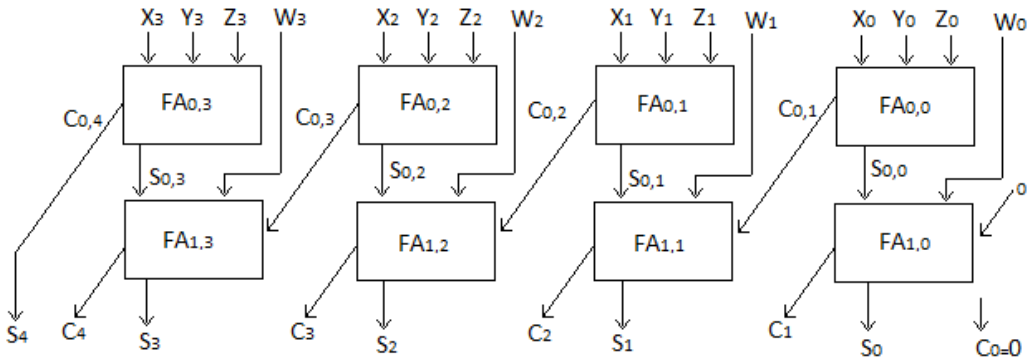
Σχήμα 2.3.4 *Αθροιστής πρόβλεψης κρατουμένου 8 bits*

Το παραπάνω κύκλωμα μπορεί να λειτουργήσει και για δυαδικούς αριθμούς σε μορφή συμπληρώματος ως προς δυο, αν αντιστραφούν τα ψηφία αρνητικής αξίας {A_7, B_7, C_{out}}.

Ο αθροιστής πρόβλεψης κρατουμένου έχει πιο πολύπλοκη δομή, αλλά είναι πιο γρήγορος από τον αθροιστή διάδοσης κρατουμένου, καθώς αποφεύγει τη διάδοση κρατουμένου και παράγει τα αποτελέσματά του με καθυστέρηση ενός HA συν logN κυκλωμάτων # συν μιας πύλης XOR. Το μειονέκτημα του αθροιστή πρόβλεψης κρατουμένου είναι η επιφάνειά του, η οποία αυξάνει πολύ σε μεγάλα μήκη λέξης. Συγκεκριμένα, ο αριθμός των κυκλωμάτων πρόβλεψης κρατουμένου (#) που απαιτούνται είναι N·log(N/2), όπου N το πλήθος των ψηφίων του αθροιστή. Σε γενικές γραμμές ωστόσο, το κέρδος σε ταχύτητα είναι πιο σημαντικό από την επιβάρυνση σε επιφάνεια. [1]

2.3.4 Αθροιστής 4:2 (4:2 Adder)

Ένας 4 σε 2 αθροιστής δέχεται τέσσερις εισόδους και δίνει δύο εξόδους για αυτό λέγεται και συμπιεστής (compressor). Όπως φαίνεται στο Σχήμα 2.3.5, αποτελείται από δύο επίπεδα πλήρων αθροιστών (FA) όπου το κάθε επίπεδο έχει n πλήρεις αθροιστές, δηλαδή όσα τα bits των εισόδων. Στο παράδειγμα αυτό ο 4:2 adder έχει τέσσερις εισόδους των 4 bits, τις X, Y, Z, W και δίνει δύο εξόδους των 4 bits, τις S και C. Στο πρώτο επίπεδο αθροιστών, κάθε αθροιστής έχει ως εισόδους και κρατούμενο εισόδου τα X_i,Y_i, Z_i αντίστοιχα και δίνει έξοδο το άθροισμα $S_{0,i}$ και το κρατούμενο εξόδου $C_{0,i+1}$, $0 \le i \le n$. Στο δεύτερο επίπεδο αθροιστών, κάθε πλήρης αθροιστής έχει ως εισόδους τη τέταρτη είσοδο W_i του 4:2 adder και το άθροισμα που προέκυψε από τον αντίστοιχο πλήρη αθροιστή του προηγούμενου επιπέδου $S_{0,i}$ και ως κρατούμενο εισόδου το κρατούμενο εξόδου που προκύπτει από τον προηγούμενο του αντίστοιχου πλήρη αθροιστή στο προηγούμενο επιπέδου $C_{0,i}$ και δίνει έξοδο το άθροισμα S_i και κρατούμενο C_i . Σημειώνεται ότι το κρατούμενο εισόδου του πρώτου πλήρη αθροιστή στο δεύτερο επίπεδο θεωρείται 0, όπως επίσης και το πρώτο κρατούμενο εξόδου C_0. Το πιο σημαντικό ψηφίο (MSB) στο άθροισμα της εξόδου ισούται με το κρατούμενο εξόδου του τελευταίου πλήρη αθροιστή του πρώτου επιπέδου, δηλαδή $S_4=C_{0,4}$.



Σχήμα 2.3.5 *Αθροιστής 4 σε 2 των 4 bits*

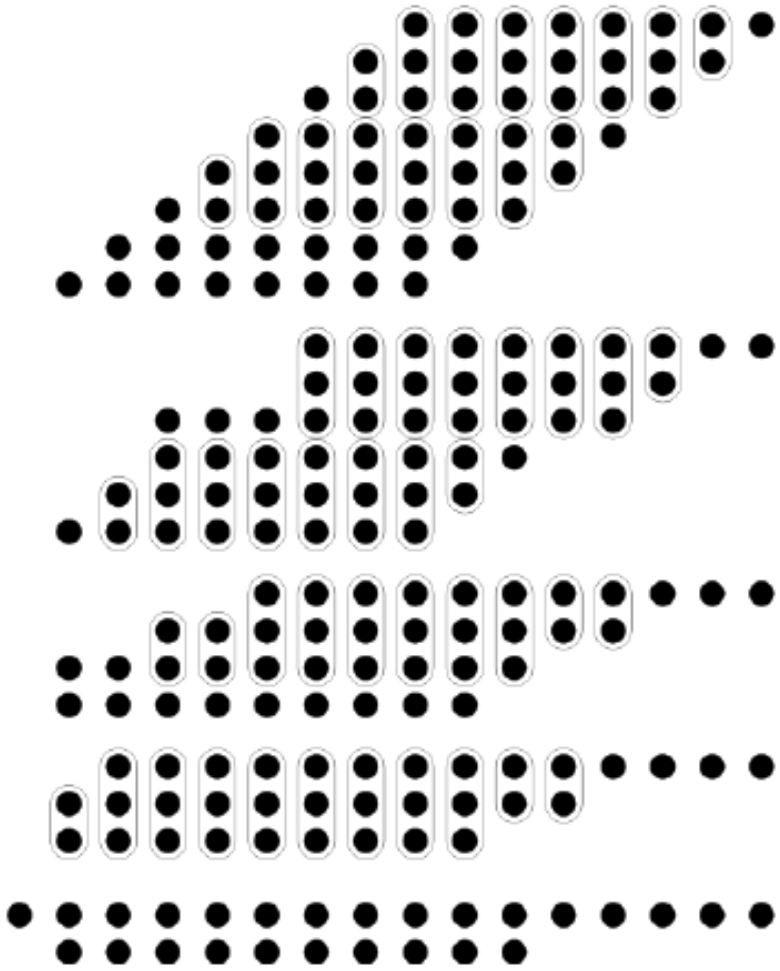
2.3.5 Δενδρικός πολλαπλασιαστής Wallace (Wallace tree)

Οι δενδρικοί πολλαπλασιαστές χρησιμοποιούν μια διαφορετική φιλοσοφία άθροισης, στοχεύοντας στη παραγωγή κυκλωμάτων πολλαπλασιασμού με την ελάχιστη καθυστέρηση. Η δημιουργία των μερικών γινομένων στους δενδρικούς πολλαπλασιαστές γίνεται ανεξάρτητα από το στάδιο της πρόσθεσης. Αφού παραχθούν όλα τα μερικά γινόμενα, οδηγούνται σε ένα δίκτυο FA και HA, από το οποίο παράγεται ένας αριθμός σε μορφή άθροισμα-κρατουμένου. Στο τελικό στάδιο, μπορεί να χρησιμοποιηθεί ένας αθροιστής διάδοσης κρατουμένου ή πρόβλεψης κρατουμένου για να παραχθεί το τελικό αποτέλεσμα σε δυαδική μορφή.

Επειδή η δημιουργία των μερικών γινομένων γίνεται ανεξάρτητα από το στάδιο της δενδρικής συμπίεσης, είναι δυνατή η κωδικοποίηση του ενός από τους δύο αριθμούς ή και των δυο με βάση κάποιο άλλο σύστημα. Με αυτόν τον τρόπο μπορεί να μειωθεί ο αριθμός των μερικών γινομένων έτσι ώστε να αυξηθεί η ταχύτητα του δενδρικού πολλαπλασιαστή.

Στόχος του δενδρικού συμπιεστή Wallace είναι να γίνει η συμπίεση των μερικών γινομένων μετά από όσο το δυνατόν λιγότερα επίπεδα FA και HA και συνεπώς σε όσο το δυνατόν μικρότερο χρονικό διάστημα. Για το σκοπό αυτό, σε κάθε επίπεδο, τα ψηφία ίδιου βάρους ομαδοποιούνται ανά τρία και εισέρχονται σαν είσοδοι σε έναν FA, εάν περισσέψουν δύο, τότε ομαδοποιούνται ανά δύο και εισέρχονται ως είσοδοι σε έναν HA, ενώ αν περισσέψει μόνο ένα, μεταφέρεται στο επόμενο επίπεδο.

Στη συνέχεια παρουσιάζεται το τμήμα συμπίεσης των μερικών γινομένων ενός 8×8 bit πολλαπλασιαστή Wallace. Κάθε κουκίδα συμβολίζει ένα bit του αντίστοιχου μερικού γινομένου. Όπου υπάρχει ομαδοποίηση τριών bits, τα τρία αυτά bits εισέρχονται σαν είσοδοι σε έναν FA και όπου υπάρχει ομαδοποίηση δύο bits, τα δύο αυτά bits εισέρχονται σαν είσοδοι σε έναν HA. Προφανώς, κάθε FA και HA παράγουν ως αποτέλεσμα ένα bit ίδιου βάρους (Save) και ένα bit με το αμέσως μεγαλύτερο βάρος (Carry). [1]



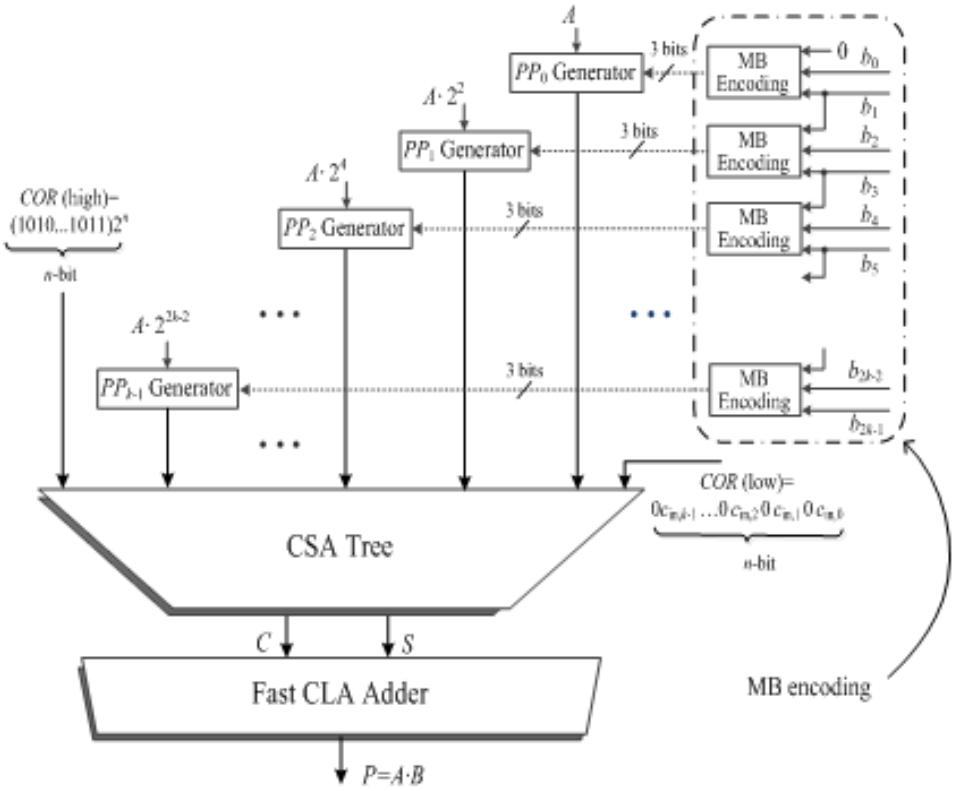
Σχήμα 2.3.6 *Ομαδοποίηση μερικών γινομένων δύο 8-bit αριθμών*

2.4 Πολλαπλασιαστές

2.4.1 Modified Booth Πολλαπλασιαστής

Η μία από τις δύο εισόδους ενός Modified Booth πολλαπλασιαστή είναι προ-κωδικοποιημένη σε Modified Booth αναπαράσταση. Θεωρούμε ότι η είσοδος αυτή προέρχεται από ένα σύνολο προκαθορισμένων συντελεστών, οι οποίοι κωδικοποιούνται από πριν σύμφωνα με τον Modified Booth αλγόριθμο.

Το σχηματικό διάγραμμα (Σχήμα 2.4.1) απεικονίζει την αρχιτεκτονική ενός συστήματος που αποτελείται από έναν Modified Booth πολλαπλασιαστή.



Σχήμα 2.4.1 *Αρχιτεκτονική συμβατικού MB πολλαπλασιαστή*

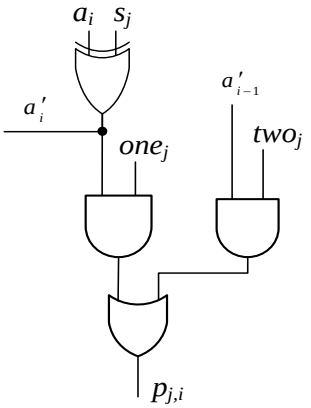
Έστω το γινόμενο P=A·B, ο συντελεστής B= $< b_{n-1}b_{n-2}...b_1b_0>_{2's}$ αποτελείται από n=2k bits και οδηγείται στα MB κωδικοποιημένα blocks. Κωδικοποιείται σύμφωνα με τον Modified Booth αλγόριθμο (βλέπε ενότητα 2.1.2) και έπειτα πολλαπλασιάζεται με το A= $< a_{n-1}a_{n-2}...a_1a_0>_{2's}$, το οποίο είναι σε μορφή συμπληρώματος ως προς δύο.

Η παραγωγή των k μερικών γινομένων (partial products) δίνεται από την ακόλουθη εξίσωση:

$$PP_j = A \cdot b_j^{MB} = \overline{p_{j,n}}\, 2^n + \sum_{i=0}^{n-1} p_{j,i} 2^i$$

Η παραγωγή του i-στού bit $p_{j,i}$ του μερικού γινομένου PP_j βασίζεται στην ακόλουθη εξίσωση, η οποία απεικονίζεται και σε μορφή λογικών πυλών στο Σχήμα 2.4.2.

$$p_{j,i} = \Big(\big(a_i \oplus s_j\big) \Lambda one_j \Big) \vee \big(\big(a_{i-1} \oplus s_j\big) \Lambda two_j \big)$$



Σχήμα 2.4.2 *Παραγωγή του i-ου bit $p_{j,I}$ του μερικού γινομένου PPj για τον MB πολλαπλασιαστή*

Για τον υπολογισμό του λιγότερου (LSB) και περισσότερου (MSB) σημαντικού bit των μερικών γινομένων υποθέτουμε ότι $a_{-1}=0$ και $a_n=a_{n-1}$ αντίστοιχα.

Αφού παραχθούν τα μερικά γινόμενα, προστίθενται, με σωστά υπολογισμένα τα βάρη τους, μέσω ενός Wallace Carry Save Adder (CSA) tree μαζί με τους διορθωτικούς όρους (COR), οι οποίοι δίνονται από τις ακόλουθες δύο εξισώσεις:

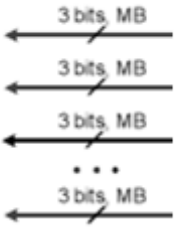
$$P = A \cdot B = COR + \sum_{j=0}^{k-1} PP_j 2^{2j}$$

$$COR = COR(low) + COR(high) = \sum_{j=0}^{k-1} c_{in,j} 2^{2j} + 2^n \Big(1 + \sum_{j=0}^{k-1} 2^{2j+1}\Big)$$

,όπου $c_{in,j}= (one_j + two_j)\cdot s_j$.

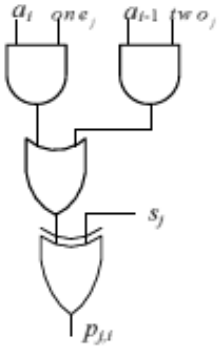
Τέλος, η έξοδος κρατούμενο-γινόμενο (carry-save) του Wallace CSA tree οδηγείται σε έναν Carry Look Ahead (CLA) αθροιστή ώστε να διαμορφωθεί το τελικό αποτέλεσμα του γινομένου P=A·B.

Σε αυτό το σημείο, θα αναλυθεί λίγο περισσότερο το κομμάτι της προ-κωδικοποίησης πολλαπλασιαστή. Ο συντελεστής B κωδικοποιείται σύμφωνα με τον MB αλγόριθμο. Το κυκλωμένο μέρος του σχήματος 2.4.1, το οποίο περιέχει το κύκλωμα κωδικοποίησης σε MB του πολλαπλασιαστή, αντικαθίσταται από το Σχήμα 2.4.3, δηλαδή στον πολλαπλασιαστή που θα χρησιμοποιηθεί τα bits εισόδου του b έχουν ήδη προ-κωδικοποιηθεί και εισάγονται στο κύκλωμα.



Σχήμα 2.4.3 *Προ-κωδικοποιημένη είσοδος του MB πολλαπλασιαστή*

Οι γεννήτριες παραγωγής των μερικών γινομένων (PP_j Generators- PPG) τροφοδοτούνται σε κάθε κύκλο ρολογιού. Το σχηματικό διάγραμμα των PPG φαίνεται στο σχήμα 2.4.4.



Σχήμα 2.4.4 *Παραγωγή του i-ου bit $p_{j,i}$ του μερικού γινομένου PPj για τον προ-κωδικοποιημένο MB πολλαπλασιαστή*

Στον προ-κωδικοποιημένο MB πολλαπλασιαστή, η τιμή ‘1’ του s_j (Πίνακας2.1.1) αντικαθίσταται από ‘0’, το οποίο μειώνει τη μεταγωγική δραστηριότητα. Η νέα εξίσωσή του είναι:

$$s_j = b_{2j+1} \oplus (b_{2j+1} \wedge b_{2j} \wedge b_{2j-1})$$

Έτσι, το Σχήμα 2.4.2 αντικαθίσταται από το 2.4.4, το οποίο είναι πιο αποδοτικό σε ισχύ(power).

Η προηγούμενη σχέση απαιτεί πιο περίπλοκο κύκλωμα, παρόλα αυτά, λόγω της χρήσης της προ-κωδικοποίησης, δεν επηρεάζεται η area και το delay του κυκλώματος.

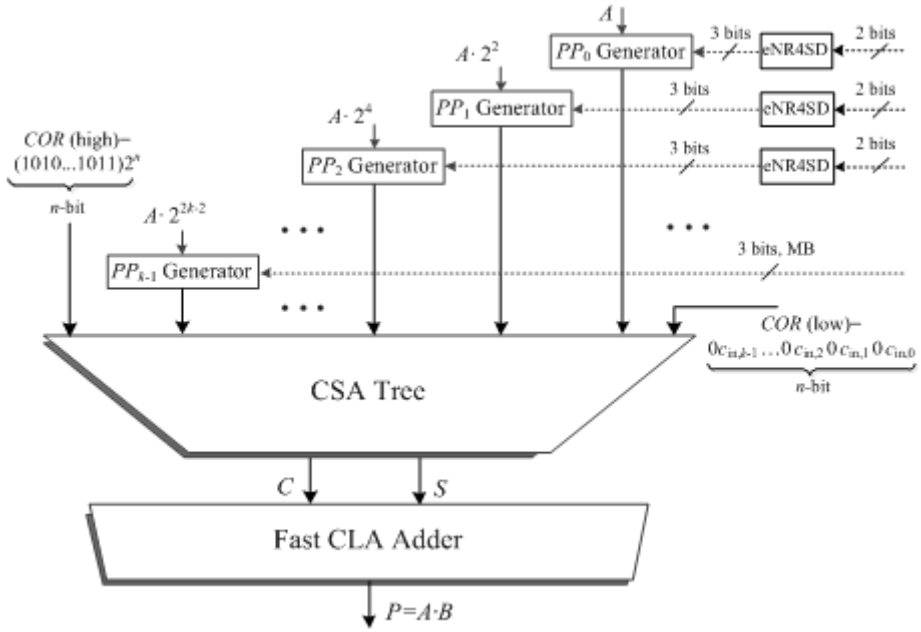
Τα μερικά γινόμενα, με σωστά βάρη, και με τον διορθωτικό όρο (COR), που αναφέρθηκε παραπάνω, εισάγονται σε ένα Wallace CSA δέντρο με κρατούμενο εισόδου $c_{in,j}$ που δίνεται από τη σχέση:

$$c_{in,j} = s_j$$

Η έξοδος, σε μορφή carry-save, του Wallace CSA δέντρου συγχωνεύεται στο τελικό αποτέλεσμα μέσω ενός CLA αθροιστή. [2]

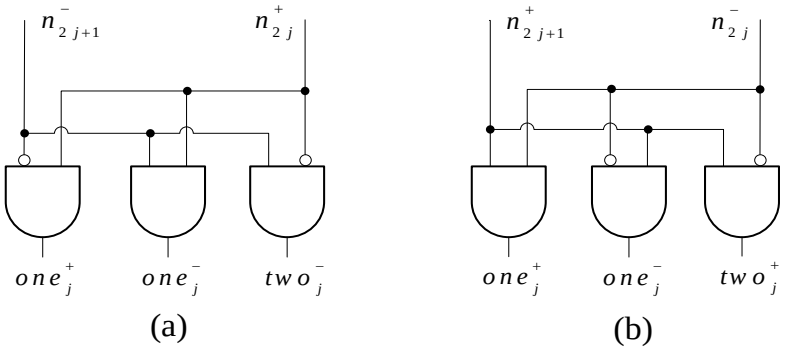
2.4.2 NR4SD Πολλαπλασιαστής

Η αρχιτεκτονική για τον προ-κωδικοποιημένο NR4SD πολλαπλασιαστή παρουσιάζεται στο Σχήμα 2.4.5.



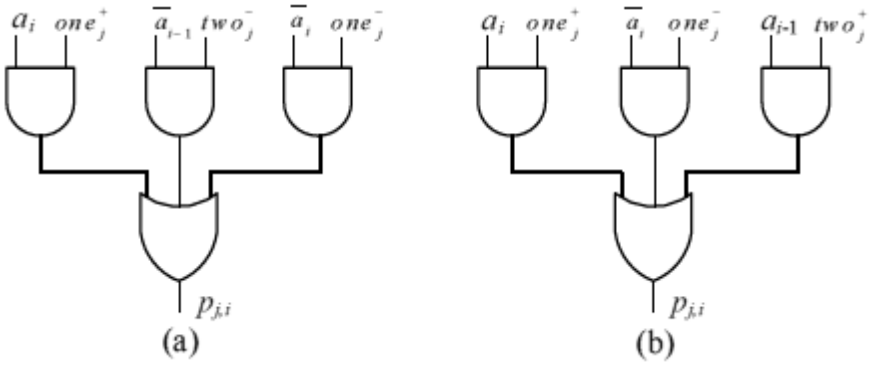
Σχήμα 2.4.5 *Αρχιτεκτονική για NR4SD πολλαπλασιαστές*

Σε αυτό τον πολλαπλασιαστή χρησιμοποιούνται δύο μόνο bits για τη κωδικοποίηση, είτε τα n_{2j+1}^{-} και n_{2j}^{+} (Πίνακας 2.2.2) στην περίπτωση της NR4SD$^-$ αναπαράστασης είτε τα n_{2j+1}^{+} και n_{2j}^{-} (Πίνακας 2.2.3) στην περίπτωση της NR4SD$^+$. Ο προ-κωδικοποιημένος NR4SD πολλαπλασιαστής χρειάζεται κάποιο επιπλέον hardware για την παραγωγή των κωδικοποιημένων σημάτων one$_j^+$, one$_j^-$, two$_j^-$ για την NR4SD$^-$ μορφή και one$_j^+$, one$_j^-$, two$_j^+$ για την NR4SD$^+$. Τα NR4SD blocks κωδικοποίησης του Σχήματος 2.4.6 υλοποιούνται από το κύκλωμα του Σχήματος 2.4.6, (a) για NR4SD$^-$ και (b) για NR4SD$^+$.



Σχήμα 2.4.6 *Επιπλέον κύκλωμα που απαιτείται της NR4SD πολλαπλασιαστές για την ολοκλήρωση της (a) NR4SD⁻ και (b) NR4SD⁺ κωδικοποίησης*

Το κύκλωμα υλοποίησης της γεννήτριας παραγωγής μερικών γινομένων (PPG) για τον NR4SD- και NR4SD+ πολλαπλασιαστή αντίστοιχα παρουσιάζονται στο Σχήμα 2.4.7 (a) και (b) αντίστοιχα.



Σχήμα 2.4.7 *Παραγωγή του i-ου bit $p_{j,I}$ του μερικού γινομένου PPj για τον (a) NR4SD⁻ και (b) NR4SD⁺ προ-κωδικοποιημένο πολλαπλασιαστή*

Κάθε μερικό γινόμενο (partial product) από τον προ-κωδικοποιημένο NR4SD- και NR4SD+ πολλαπλασιαστή υλοποιείται σύμφωνα με το Σχήμα 2.4.7 (α) και 2.4.7 (b) αντίστοιχα, εκτός από το μερικό γινόμενο PP_{k-1} που αντιστοιχεί στο πιο σημαντικό ψηφίο. Καθώς αυτό το ψηφίο είναι σε MB μορφή, υλοποιείται με τη γεννήτρια παραγωγής μερικών γινομένων του Σχήματος 2.4.4 που παρουσιάστηκε στην παραπάνω ενότητα.

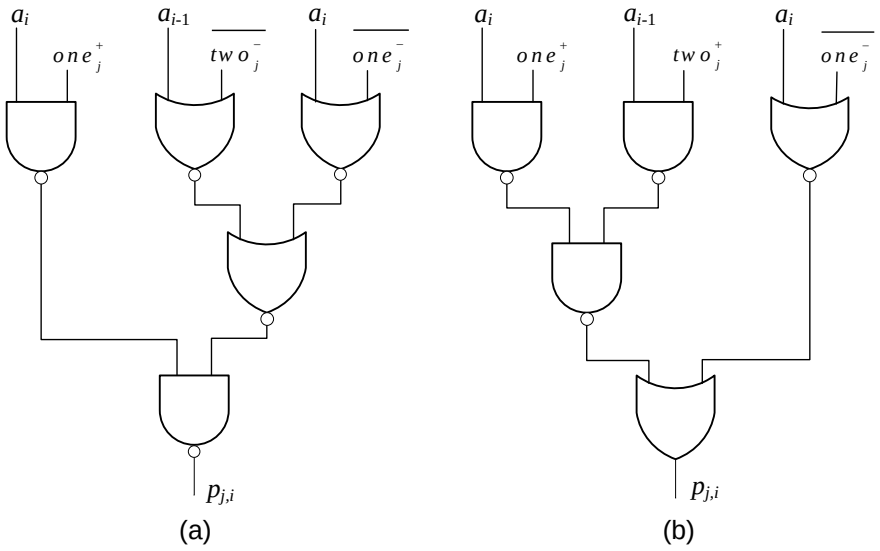
Τα μερικά γινόμενα, με σωστά βάρη, και ο διορθωτικός όρος (COR), που αναλύθηκε στην ενότητα 2.4.1, εισάγονται σε ένα Wallace CSA δένδρο. Το κρατούμενο εισόδου $c_{in,j}$ για τους προ-κωδικοποιημένους NR4SD- και NR4SD+ πολλαπλασιαστές υπολογίζεται από τις σχέσεις:

NR4SD-: $c_{in,j} = two_j^- \vee one_j^-$

NR4SD⁺: $c_{in,j} = one_j^-$

Η έξοδος σε μορφή αθροίσματος-κρατουμένου (carry-save), που προκύπτει από το Wallace tree, μετατρέπεται στο τελικό αποτέλεσμα μέσω ενός CLA αθροιστή.

Οι γεννήτριες μερικών παραγώγων για τους NR4SD⁻ και NR4SD⁺ πολλαπλασιαστές περιέχουν ένα σημαντικό αριθμό από αντιστροφείς καθώς χρειάζονται να αντιστρέψουν όλα τα bits του A στη περίπτωση ενός αρνητικού bit (b_j^{NR+}= -1 και b_j^{NR-}= -2 ή -1). Προκειμένου να αποφευχθούν όλες αυτές οι αντιστροφές και κατά επέκταση να μειωθεί η area και το delay διάδοσης, τα κυκλώματα των γεννητριών παραγωγής μερικών γινομένων των NR4SD⁻ και NR4SD⁺ προ-κωδικοποιημένων πολλαπλασιαστών (Σχήμα 2.4.7) αντικαθίστανται από αυτό του Σχήματος 2.4.8, (a) για NR4SD⁻ και (b) για NR4SD⁺ αντίστοιχα, χρησιμοποιώντας πύλες NAND και NOR. Όπως παρατηρείται στο Σχήμα 2.4.8, τα σήματα one⁻, two⁻ για την NR4SD⁻ μορφή και one⁻ για την NR4SD⁺ έχουν αντιστραφεί. [2]



Σχήμα 2.4.8 *Νέα μερικά γινόμενα για τον (a) NR4SD⁻ και (b) NR4SD⁺ προ-κωδικοποιημένο πολλαπλασιαστή*

2.4.3 Truncated Πολλαπλασιαστής

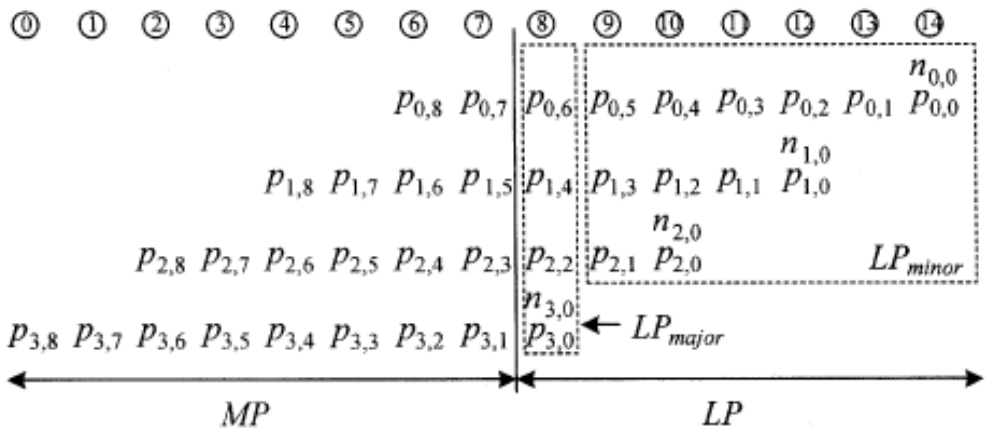
I. Πολλαπλασιαστής σταθερού μήκους

Τα μερικά γινόμενα (partial products) του MB πολλαπλασιαστή μπορούν να χωριστούν σε δύο μέρη, MP και LP, όπως φαίνεται στο Σχήμα 2.4.9 για πολλαπλασιασμό δύο αριθμών X και Y μήκους W=8 bits. Όπου ο X είναι σε μορφή σε συμπλήρωμα ως προς δυο και ο Y σε MB κωδικοποίηση. Στον Πίνακα 2.4.1 φαίνονται τα μερικά γινόμενα που αντιστοιχούν σε κάθε κωδικοποιημένο y_i'. Για να

δημιουργηθεί πιο αποτελεσματικό σφάλμα αντιστάθμισης πόλωσης, το LP μέρος μπορεί να διαχωριστεί επιμέρους σε LP_{major} και LP_{minor}.

y_{2i+1}	y_{2i}	y_{2i-1}	PP/00	PP/01	PP/10	PP/11
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	1	1	0	0
1	1	0	1	1	0	0
1	1	1	1	1	1	1

Πίνακας 2.4.1 *Ψηφία μερικών γινομένων για όλες τις πιθανές περιπτώσεις [όπου PP/αβ αντιστοιχεί σε PP για x_j x_{j-1}=αβ]*



Σχήμα 2.4.9 *MP και LP για MB πολλαπλασιαστή μήκους W=8*

Έτσι, μπορεί να εκφραστεί το (2W-1)-bit ιδανικό γινόμενο (ideal product) P_I ως εξής:

$$P_I = S_{MP} + S_{LP}$$

όπου τα S_{MP} και S_{LP} αναπαριστούν τα αθροίσματα των στοιχείων των MP και LP αντίστοιχα και δίνονται από τις σχέσεις:

$$S_{MP} = \sum_{2w-2\geq 2i+j\geq W-1} p_{i,j} 2^{-(2W-2-2i-j)}$$

και

$$S_{LP} = \sum_{W-1>2i+j\geq 0} p_{i,j} 2^{-(2W-2-2i-j)} + \sum_{i=0}^{\frac{W}{2}-1} n_{i,0} 2^{-(2W-2-2i)}$$

Στους τυπικούς σταθερού μήκους πολλαπλασιαστές, τα κύτταρα της πρόσθεσης που απαιτούνται για το S_{LP} μέρος παραλείπονται και κατάλληλες πολώσεις εισάγονται στα διατηρημένα αθροιστικά κύτταρα βασιζόμενες σε πιθανολογικές εκτιμήσεις. Έτσι, μπορεί να εκφραστεί το W-bit μήκους quantized γινόμενο (quantized product) P_Q ως:

$$P_Q = S_{MP} + \sigma \times 2^{-(W-1)}$$

όπου το σ είναι το σφάλμα αντιστάθμισης πόλωσης. Σημειώνεται ότι το σ προσεγγίζει τα σήματα κρατουμένου που διαδίδονται από το LP μέρος στο MP.

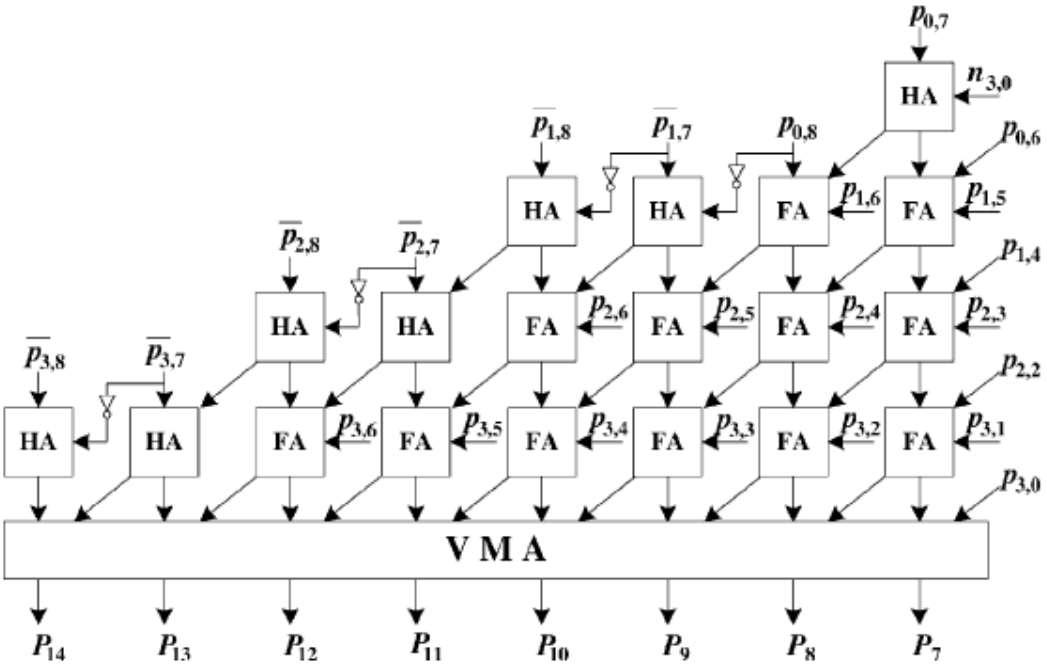
Το άθροισμα των κρατουμένων που παράγονται από το LP μέρος δίνονται από τη σχέση:

$$\sigma_{[8]} = [\frac{1}{2}\beta + \lambda]$$

όπου β είναι το άθροισμα των στοιχείων του LP_{major} και λ είναι το άθροισμα των στοιχείων του LP_{minor}. Επίσης, [t] είναι το κατώτερο όριο του t, δηλαδή είναι η μέγιστη δυνατή τιμή που μπορεί να πάρει και είναι μικρότερη ή ίση του t. Για γνωστό β, η τιμή του $\sigma_{[8]}$, χρησιμοποιώντας στατιστική ανάλυση, προκύπτει ότι ως καλύτερη επιλογή είναι να είναι ίσο με β. Για παράδειγμα, για MB πολλαπλασιαστή με μήκος W=8, το $\sigma_{[8]}$ υπολογίζεται ως:

$$\sigma_{[8]} = p_{0,6} + p_{1,4} + p_{2,2} + p_{3,0} + n_{3,0}$$

Ακολουθεί το Σχήμα 2.4.10 όπου παρουσιάζει την αρχιτεκτονική ενός πολλαπλασιαστή σταθερού μήκους, με μήκος W=8 bits.



Σχήμα 2.4.10 *Αρχιτεκτονική πολλαπλασιαστή σταθερού μήκους για W=8*

II. Μέθοδος αντιστάθμισης λάθους

Από το Σχήμα 2.4.9 παρατηρείται ότι το LP_{major} έχει κυρίαρχη επίδραση στα σήματα κρατουμένων που παράγονται από το LP μέρος, καθώς έχει μεγαλύτερη βαρύτητα. Επίσης, τα παραγόμενα μερικά γινόμενα εξαρτώνται από τις εξόδους Booth κωδικοποίησης. Συνεπώς, με βάση αυτές τις παρατηρήσεις, μια μέθοδος αντιστάθμισης λάθους παρουσιάζεται στη συνέχεια.

Από το Σχήμα 2.4.10, το S_{LP} μπορεί να γραφτεί ως:

$$S_{LP} = S_{LP,major} + S_{LP,minor}$$

Εάν ορισθεί S_{LP}' ως:

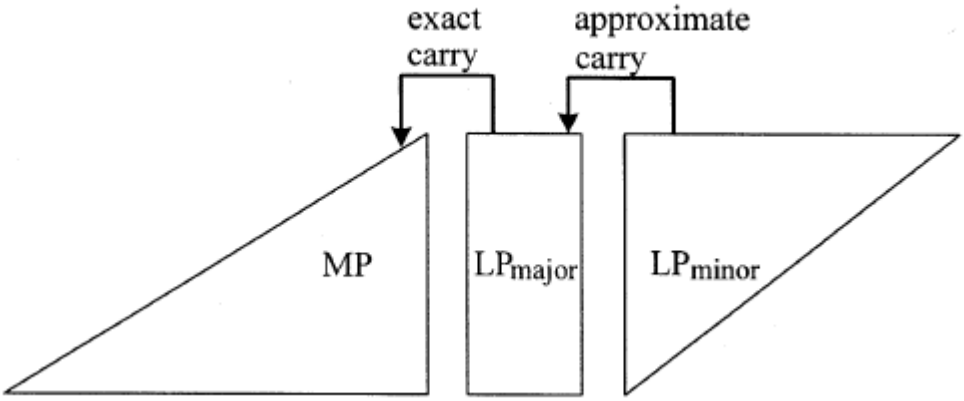
$$S'_{LP} = S_{LP} \times 2^{W}$$

τότε τα $S_{LP,major}$' και $S_{LP,minor}$' ορίζονται ως:

$$S'_{LP,major} = p_{0,6} + p_{1,4} + p_{2,2} + p_{3,0} + n_{3,0}$$

$$\begin{aligned} S'_{LP,minor} = {} & 2^{-1}(p_{0,5} + p_{1,3} + p_{2,1}) + 2^{-2}(p_{0,4} + p_{1,2} + p_{2,0} + n_{2,0}) \\ & + 2^{-3}(p_{0,3} + p_{1,1}) + 2^{-4}(p_{0,2} + p_{1,0} + n_{1,0}) + 2^{-5}(p_{0,1}) \\ & + 2^{-6}(p_{0,0} + n_{0,0}) \end{aligned}$$

Το Σχήμα 2.4.11 δείχνει τη δομή του πολλαπλασιαστή σταθερού μήκους.



Σχήμα 2.4.11 *Δομή προτεινόμενου MB πολλαπλασιαστή σταθερού μήκους*

Το προτεινόμενο λάθος αντιστάθμισης πόλωσης ορίζεται ως εξής:

$$\sigma_{prop} = C_E\left[S'_{LP,major} + C_A\left[S'_{LP,minor}\right]\right]$$

όπου το $C_E[t]$ αναπαριστά την ακριβή τιμή κρατουμένου του t και το $C_A[t]$ αναπαριστά την προσεγγιστική τιμή κρατουμένου του t. Σημειώνεται ότι το $C_A\left[S'_{LP,minor}\right]$ υπολογίζει τη κατά προσέγγιση τιμή κρατουμένου από το LP_{minor} στο LP_{major}.

Η παραπάνω διαδικασία συνοψίζεται στα ακόλουθα:

1. Διαχωρισμός LP μέρους σε LP_{major} και LP_{minor}.
2. Υπολογισμός της προσεγγιστικής τιμής κρατουμένου από το LP_{minor}.
3. Πρόσθεση της προσεγγιστικής τιμής στο LP_{major}.
4. Η τιμή κρατουμένου που προκύπτει από την πρόσθεση στο βήμα 3 είναι το επιθυμητό λάθος αντιστάθμισης πόλωσης.

III. Διαδικασίες παραγωγής προσεγγιστικού κρατούμενου

Ορίζεται το y_i'' ως:

$$y_i'' = \begin{cases} 1, & \alpha\nu\ y_i' \neq 0 \\ 0, & \alpha\nu\ y_i' = 0 \end{cases}$$

Από τον Πίνακα 2.1.4 της ενότητας 2.1.2 φαίνεται ότι το y_i'' μπορεί να υπολογιστεί ως εξής:

$$y_i'' = X_{i\_sel} \vee 2X_{i\_sel}$$

όπου X_{i_sel} αντιστοιχεί στο one_i του Πίνακα 2.1.4, το $2X_{i\_sel}$ αντιστοιχεί στο two_i και το NEG_i αντιστοιχεί στο s_i.

Τα προσεγγιστικά σήματα κρατουμένων προστίθενται στο LP_{major} μέρος. Έτσι, τα τελικά κρατούμενα από το LP_{major} προστίθενται στο MP ως λάθος αντιστάθμισης πόλωσης.

a) Πρώτη διαδικασία παραγωγής προσεγγιστικού κρατουμένου (Approximate Carry Generation procedure - ACGP I)

Η διαδικασία αυτή έχει ως εξής:

1. Για δεδομένο W, ο αριθμός των σημάτων προσεγγιστικών κρατουμένων ορίζεται ως $N_{AC} = [W/4]$.
2. Τα σήματα των προσεγγιστικών κρατουμένων συμβολίζονται ως a_carry_0,a_carry_1,…,a_carry_(N_{AC}-1).
3. Η τιμή κάθε σήματος a_carry_i ορίζεται ως 1 εάν έστω (2i+1) από τα $y''_{w/2-2}, y''_{w/2-3}, \dots, y''_1$ και y''_0 είναι 1.
4. Χρησιμοποιώντας τη μέθοδο χαρτών Karnaugh (ή κάποιας άλλης) προκύπτουν οι συναρτήσεις του βήματος 3 και κατά επέκταση το κύκλωμα παραγωγής προσεγγιστικών κρατουμένων.

Για μεγάλο W, η προσέγγιση μέσω χαρτών Karnaugh γίνεται ιδιαίτερα περίπλοκη. Επιπρόσθετα, ο χρόνος κρίσιμου μονοπατιού (critical path) αυξάνεται όσο αυξάνεται το W. Σημειώνεται ότι:

$$\sum_{i=0}^{N_{AC}-1} a\_carry\_i = \left\{ \sum_{i=0}^{\frac{W}{2}-2} \frac{y_i''}{2} \right\}_r$$

Έτσι, για απαλοιφή των μειονεκτημάτων αυτών προτείνεται η δεύτερη διαδικασία παραγωγής προσεγγιστικών κρατουμένων.

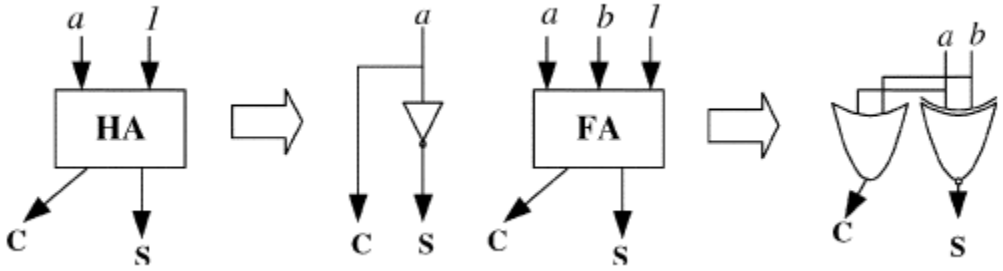
b) Δεύτερη διαδικασία παραγωγής προσεγγιστικού κρατουμένου (Approximate Carry Generation procedure - ACGP II)

Η διαδικασία αυτή ακολουθεί τα εξής βήματα:

1. Τα σήματα του συνόλου $\{y''_{w/2-2} y''_{w/2-3} \dots y''_0\}$ χωρίζονται σε ομάδες των τριών σημάτων. Εάν ο αριθμός των σημάτων του συνόλου είναι 3N+l (l=1,2), τότε η τελευταία ομάδα περιέχει μόνο l σήματα. Τα 3N σήματα προστίθενται χρησιμοποιώντας N πλήρους αθροιστές (FA). Για l=2, τα δύο σήματα της τελευταίας ομάδας προστίθενται με έναν ημιαθροιστή (HA). Για l=1, το σήμα της τελευταίας ομάδας περνάει στο επόμενο στάδιο. Τα N (ή N+1 για l=2) κρατούμενα από κάθε αθροιστή είναι προσεγγιστικά σήματα κρατουμένων.

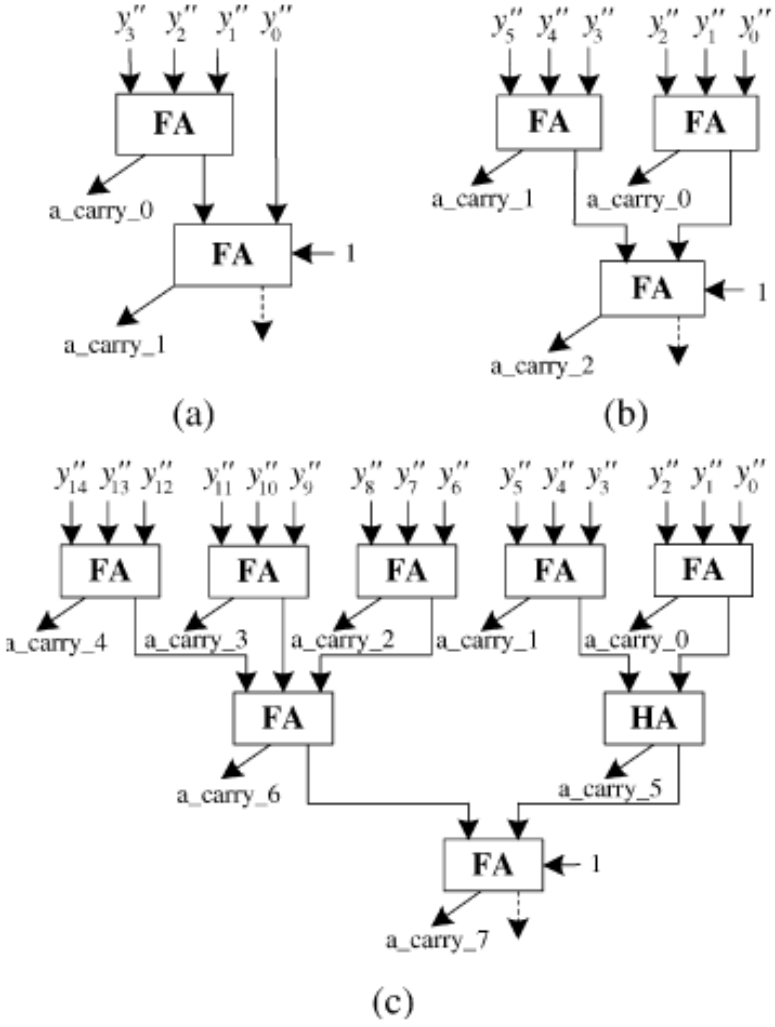
2. Τα σήματα αθροισμάτων που παράχθηκαν στο βήμα 1 προστίθενται εκ νέου χρησιμοποιώντας την ίδια αρχή με το βήμα 1. Τα κρατούμενα από κάθε αθροιστή είναι προσεγγιστικά κρατούμενα και τα νέα αθροίσματα περνάνε στο επόμενο στάδιο.
3. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να μείνει ένα μόνο άθροισμα.
4. Προστίθεται '1' στον τελευταίο αθροιστή.

Ο συνολικός αριθμός αθροιστών που χρησιμοποιούνται στη διαδικασία ACGP II είναι N_{AC} και κατά συνέπεια ο συνολικός αριθμός των προσεγγιστικών κρατουμένων είναι N_{AC}. Για W=4m, χρησιμοποιούνται (N_{AC} -1) πλήρεις αθροιστές-FA και ένας ημιαθροιστής- HA. Αλλιώς, για περιττό W χρησιμοποιούνται N_{AC} πλήρεις αθροιστές –FA. Η πρόσθεση '1' στο βήμα 4, η οποία αντιστοιχεί στη διαδικασία της στρογγυλοποίησης, μπορεί να γίνει χρησιμοποιώντας το Σχήμα 2.4.12.



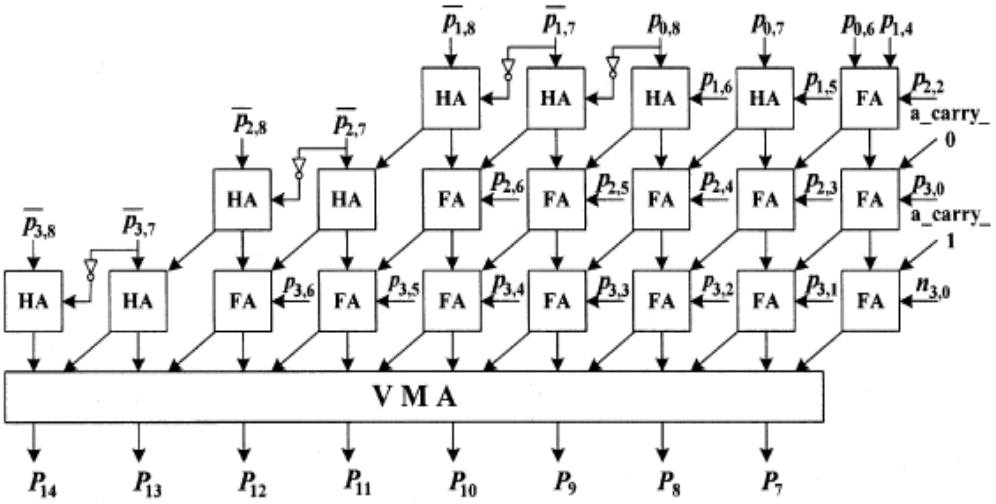
Σχήμα 2.4.12 *Απλοποιημένοι Αθροιστές*

Το Σχήμα 2.4.14 δείχνει κυκλώματα παραγωγής προσεγγιστικών κρατουμένων μέσω της ACGP II για W=10, W=14, W=32 στα (a),(b), (c) αντίστοιχα.

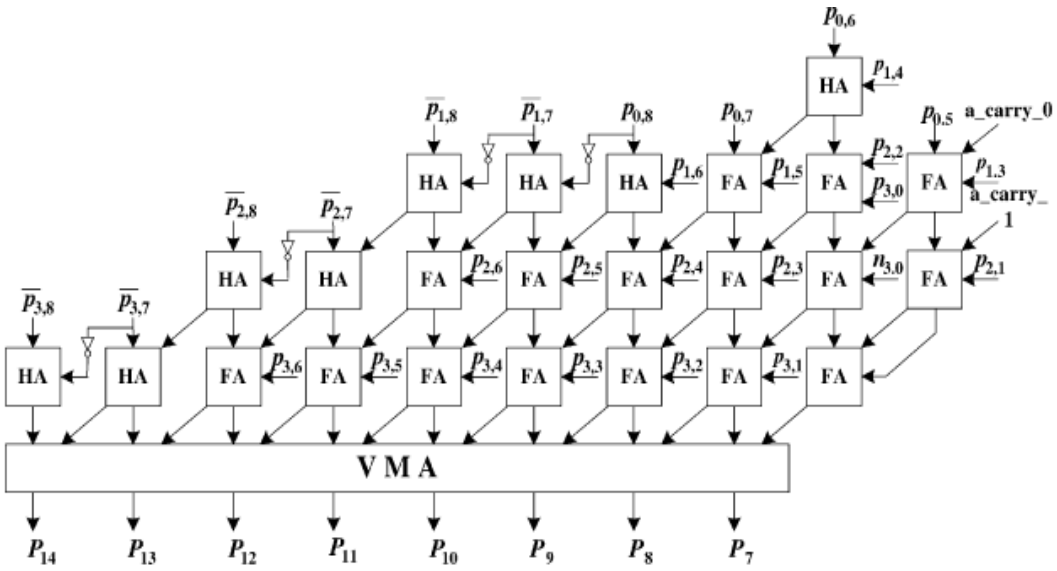


Σχήμα 2.4.13 *Κυκλώματα γεννήτριας προσεγγιστικών κρατουμένων με βάση τη διαδικασία ACGP II. (a) W=10, (b) W=14, (c) W=32*

Όπως φαίνεται στο Σχήμα 2.4.2 υπάρχει μόνο μια στήλη LP_{major}. Ωστόσο, το LP_{major} μπορεί να επεκταθεί σε δύο ή περισσότερες στήλες. Το Σχήμα 2.4.14 και το Σχήμα 2.4.15 δείχνουν έναν πολλαπλασιαστή σταθερού μήκους W=8, κρατώντας μια και δύο στήλες για το LP_{major} αντίστοιχα.



Σχήμα 2.4.14 *Πολλαπλασιαστής σταθερού μήκους κρατώντας μια στήλη στο LP_{major} για W=8*



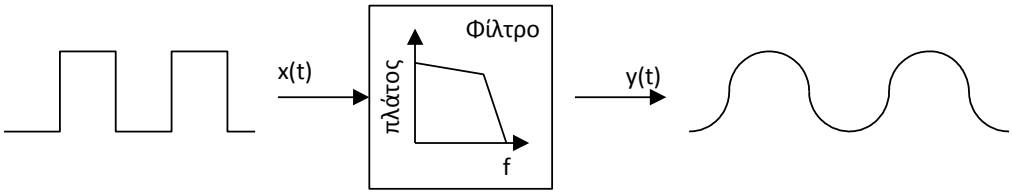
Σχήμα 2.4.15 *Πολλαπλασιαστής σταθερού μήκους κρατώντας μια στήλη στο LP_{major} για W=8*

Κρατώντας περισσότερες στήλες στο LP_{major} , μειώνεται το μέσο σφάλμα με συνέπεια όμως τη χρήση περισσότερου υλικού. [3]

**Όμοια με τον truncated MB πολλαπλασιαστή, υλοποιείται και ο truncated NR4SD πολλαπλασιαστής. Η μόνη διαφορά είναι η προ-κωδικοποίηση του Y, που ακολουθεί τη κωδικοποίηση NR4SD.

2.5 Φίλτρα FIR

Γενικότερα, φίλτρο μπορούμε να ονομάσουμε το σύστημα εκείνο που ενισχύει ή και εξασθενεί επιθυμητές περιοχές συχνοτήτων ενός σήματος. Ένα φίλτρο λοιπόν, όταν δέχεται στην είσοδό του ένα σήμα, παρέχει στην έξοδό του το σήμα αυτό με ενισχυμένο ή εξασθενημένο το φασματικό του περιεχόμενο, στις συχνότητες που καθορίζονται από την χαρακτηριστική του φίλτρου.



Σχήμα 2.5.1 *Γενική μορφή αναλογικού φίλτρου*

Τα αναλογικά φίλτρα επεξεργάζονται συνεχή ηλεκτρικά μεγέθη και υλοποιούνται με τη κατάλληλη διασύνδεση πυκνωτών, αντιστάσεων, ενισχυτών, διακοπτών καθώς και πηνίων. Είναι δυνατόν όμως μετά από δειγματοληψία του σήματος και ψηφιοποίηση του (μετατροπέας A/D) να γίνει αριθμητικά η παραπάνω επεξεργασία. Το αποτέλεσμα της επεξεργασίας θα αποτελεί την ψηφιοποιημένη μορφή του επιθυμητού (φιλτραρισμένου) αναλογικού σήματος. Το σύνολο των αριθμητικών πράξεων μέσω των οποίων προσομοιώνουμε την λειτουργία ενός αναλογικού φίλτρου είναι αυτό που ονομάζουμε ψηφιακό φίλτρο.

Ένα ψηφιακό φίλτρο μπορεί να ενσωματωθεί σε αναλογικά συστήματα αν περιληφθούν στην είσοδο και στην έξοδο ένας A/D και ένας D/A μετατροπέας αντίστοιχα.

Σχήμα 2.5.2 *Χρήση ψηφιακών φίλτρων σε αναλογικά σήματα*

Η σχέση μεταξύ εισόδου και εξόδου σε ένα ψηφιακό φίλτρο δίνεται από τη σχέση:

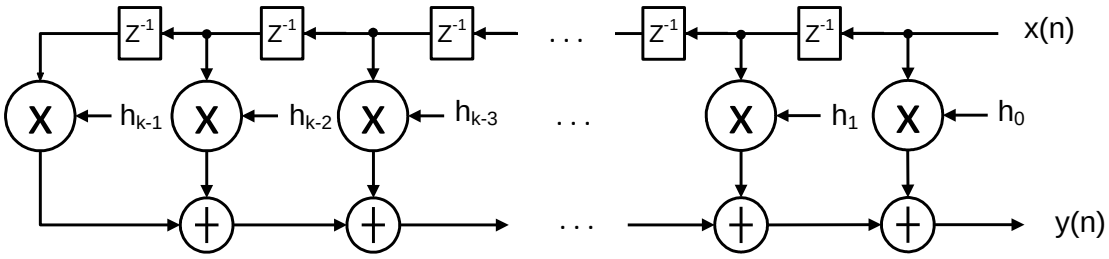
$$y_n = \sum_{i=0}^{L-1} x_{n-i} h_i$$

Στο πεδίο του χρόνου η έξοδος y_n δίνεται από τη συνέλιξη της εισόδου με τους συντελεστές (σταθερές) h_i, οι οποίοι αποτελούν την κρουστική απόκριση του φίλτρου.

Στην πραγματικότητα, η απόκριση αυτή έχει άπειρη διάρκεια. Προσεγγίζεται όμως ικανοποιητικά με ένα πεπερασμένο αριθμό συντελεστών (h_0, h1,…, h_{L-1}). Για τις συνήθεις εφαρμογές ένα L με τιμή από 16 έως 32 είναι ικανοποιητικό. Στη μορφή

αυτή το ψηφιακό φίλτρο λέμε ότι είναι πεπερασμένης κρουστικής απόκρισης, τύπου FIR (Finite Impulse Response). Για Low-Pass φίλτρα έχουμε συμμετρικές τιμές στα h_i ενώ σε Band-Pass φίλτρα οι τιμές είναι αντισυμμετρικές.

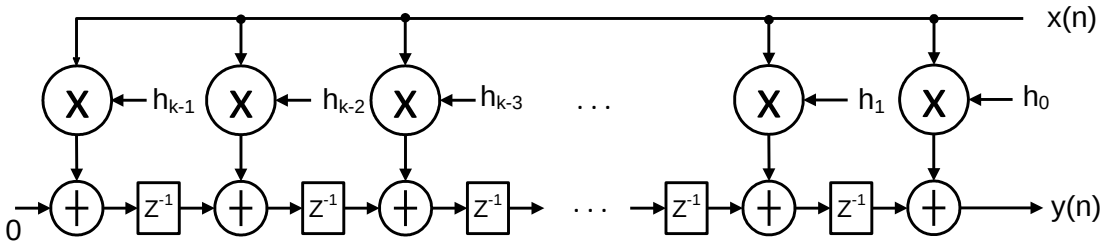
Στη συνέχεια αναλύονται οι αρχιτεκτονικές που επιτρέπουν την υλοποίηση φίλτρων FIR με τη χρήση ειδικών κυκλωμάτων. Μια απ'ευθείας (direct) υλοποίηση της σχέσης που περιγράφηκε παραπάνω μεταξύ εισόδου και εξόδου του φίλτρου φαίνεται στο ακόλουθο σχήμα:



Σχήμα 2.5.3 *Direct υλοποίηση FIR φίλτρου με k σταθερούς όρους*

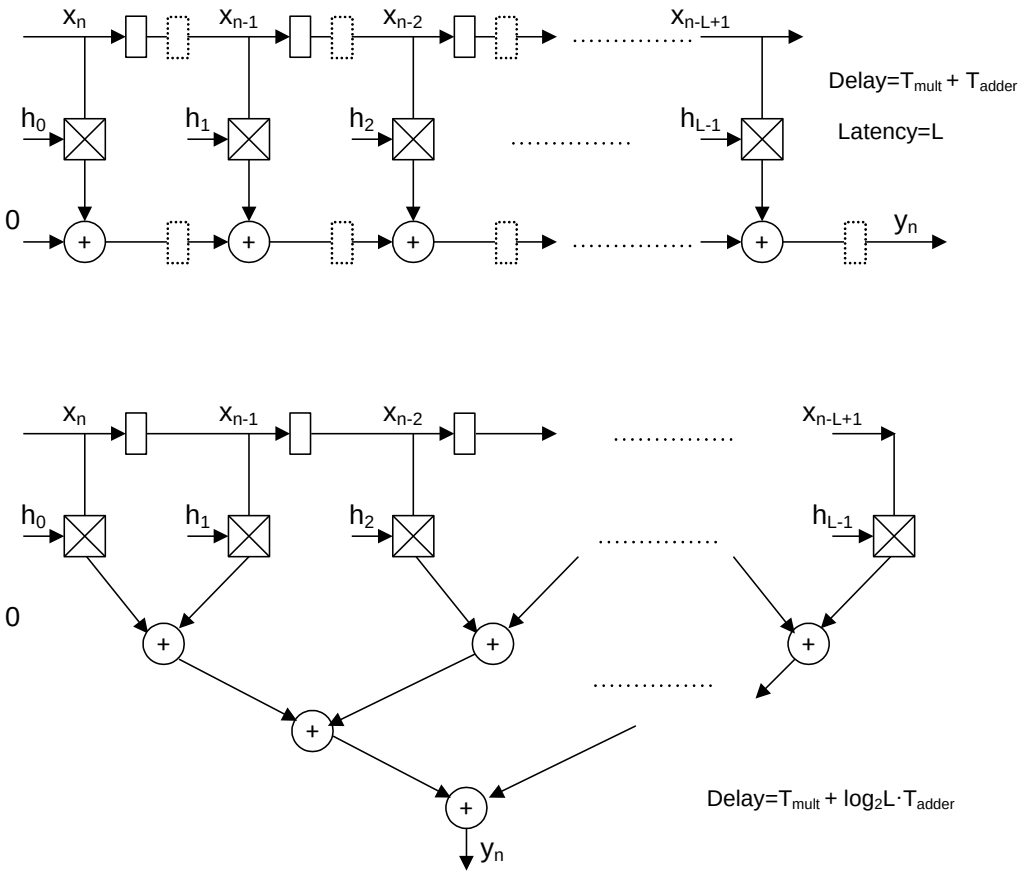
Οι καθυστερήσεις με τις διακεκομμένες γραμμές είναι προαιρετικές. Επιτρέπουν όμως την λειτουργία διοχέτευσης των προσθέσεων του κάτω κλάδου. Το μειονέκτημα εδώ είναι ο μεγάλος αριθμός των μονάδων καθυστέρησης που πρέπει να εισαχθούν καθώς και η μεγάλη καθυστέρηση (latency), ίση με τον αριθμό των συντελεστών που παρουσιάζονται.

Αντίθετα η transpose μορφή (Σχήμα 2.5.4) έχει μικρό αριθμό καθυστερήσεων και είναι άμεσης απόκρισης, δηλαδή έχει Latency=0. Στο σχήμα αυτό πρόβλημα παρουσιάζεται με τη διάδοση του x_n που λόγω μήκους αλλά και επειδή τροφοδοτεί πολλές εισόδους μπορεί να επιφέρει χρονική καθυστέρηση στη γραμμή του σήματος x_n. Με κατάλληλους μετασχηματισμούς στο γράφο μπορεί να επιτευχθεί και συστολικότητα αλλά και μικρή καθυστέρηση (Latency).



Σχήμα 2.5.4 *Transpose μορφή FIR φίλτρου με k σταθερούς όρους*

Ακολουθεί το Σχήμα 2.5.5 που παρουσιάζει αρχιτεκτονική που επιτρέπει την direct υλοποίηση φίλτρων FIR με τη χρήση ειδικών κυκλωμάτων. [1]



Σχήμα 2.5.5 *Απ'ευθείας υλοποίηση φίλτρου FIR*

3. ΣΧΕΔΙΑΣΜΟΣ ΚΥΚΛΩΜΑΤΟΣ ΦΙΛΤΡΟΥ

Σε αυτή την ενότητα παρουσιάζονται αναλυτικά οι υλοποιήσεις των φίλτρων που εξετάστηκαν, με έμφαση στα δομικά στοιχεία που χρησιμοποιήθηκαν και τις παραλλαγές που εξετάστηκαν στις υλοποιήσεις.

Αρχικά, παρουσιάζονται οι δομικές μονάδες που χρειάστηκαν για κάθε υλοποίηση.

Στη συνέχεια παρουσιάζεται η υλοποίηση φίλτρου με 16 σταθερούς όρους χρησιμοποιώντας τον συμβατικό πολλαπλασιαστή MB, έχοντας εισόδους των 16 bits και έξοδο των 36 bits, για τρεις δυνατές περιπτώσεις σχεδιασμού.

Έπειτα, παρουσιάζεται η υλοποίηση του ίδιου φίλτρου για τις ίδιες τρεις περιπτώσεις σχεδιασμού με χρήση τώρα πολλαπλασιαστή NR4SD προ-κωδικοποίησης.

Τέλος, παρουσιάζονται οι παραπάνω υλοποιήσεις φίλτρων στην truncated μορφή τους, με τις εισόδους φίλτρων τώρα 16 bits και έξοδο των 20 bits, δηλαδή χρησιμοποιώντας στην υλοποίηση truncated πολλαπλασιαστές για προ-κωδικοποίηση σε MB ή NR4SD μορφή.

3.1 Δομικά Στοιχεία

Για την υλοποίηση του φίλτρου 16 σταθερών όρων, απαιτούνται κάποια υποκυκλώματα για να πραγματοποιηθεί η πράξη της συνέλιξης που υλοποιεί κάθε φίλτρο. Έτσι, χρειάζεται δομική μονάδα D flip flop,δομική μονάδα αθροιστή 4 σε 2, δομική μονάδα CLA adder, δομική μονάδα αθροιστή με Wallace tree και δομική μονάδα του εκάστοτε πολλαπλασιαστή για τις περιπτώσεις που εξετάστηκαν. Με όλες τις προαναφερθείσες μονάδες προσαρμοσμένες στον αριθμό των bits που απαιτούνται. Πιο συγκεκριμένα:

- Η δομική μονάδα του 4 σε 2 αθροιστή έχει παρουσιαστεί αναλυτικά στην Ενότητα 2.3.4 και υλοποιείται σύμφωνα με το κύκλωμα που παρουσιάζεται στο Σχήμα 2.3.5.
- Η δομική μονάδα του CLA αθροιστή, παρουσιάζεται στην Ενότητα 2.3.3 και στην υλοποίηση μας χρησιμοποιείται έτοιμη από την DesignWare με την ονομασία DW01_add. [7]
- Η δομική μονάδα άθροισης με χρήση Wallace tree , Ενότητα 2.4.4, χρησιμοποιείται και αυτή έτοιμη από την DesignWare με την ονομασία DW02_tree. [8]

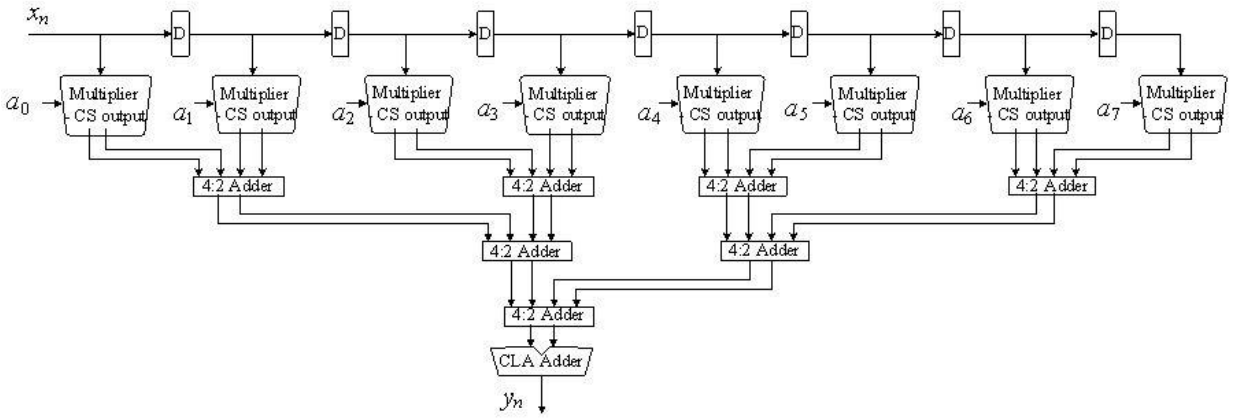
- Η δομική μονάδα του πολλαπλασιαστή με κωδικοποίηση MB, παρουσιάστηκε στην Ενότητα 2.4.1 και υλοποιείται σύμφωνα με το Σχήμα 2.4.1. Η μονάδα του πολλαπλασιαστή με κωδικοποίηση NR4SD παρουσιάστηκε στην Ενότητα 2.4.2 και υλοποιείται σύμφωνα με το Σχήμα 2.4.6. Η μονάδα του truncated πολλαπλασιαστή , παρουσιάζεται στην Ενότητα 2.4.3 και υλοποιείται σύμφωνα με το Σχήμα 2.4.14 με τα μερικά γινόμενα να παράγονται είτε σύμφωνα με την κωδικοποίηση MB είτε με τη κωδικοποίηση NR4SD που αναλύθηκαν παραπάνω.
- Η δομική μονάδα του D flip flop στο κύκλωμα του φίλτρου για δημιουργία καθυστέρησης ενός κύκλου υλοποιείται με την μετάθεση της τιμής του σήματος σε νέο καταχωρητή.

3.2 Υλοποιήσεις FIR Φίλτρου

3.2.1 Συμβατική υλοποίηση φίλτρου FIR με MB πολλαπλασιαστή

- Direct μορφή

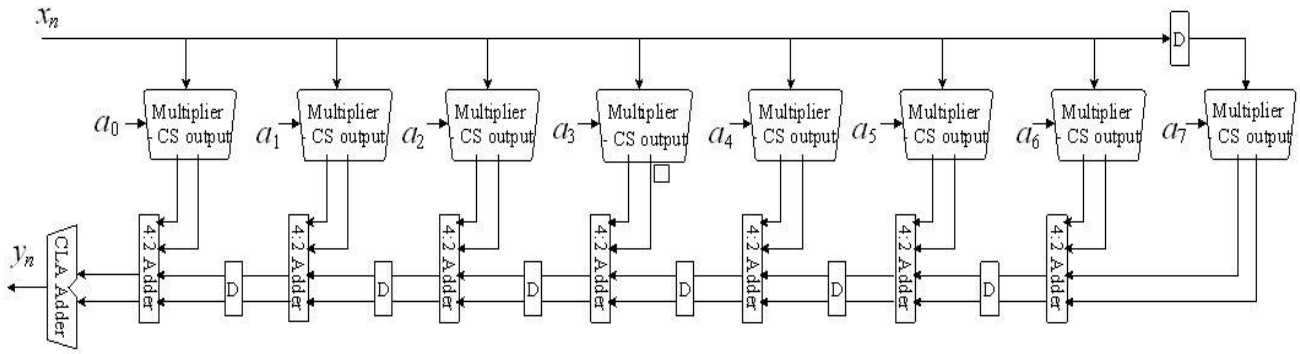
Σε αυτή τη περίπτωση υλοποιείται το Σχήμα 3.2.1 με τη διαφορά ότι το σχήμα είναι με 8 σταθερούς όρους, ενώ η υλοποίηση μας με 16. Για καλύτερη ευκρίνεια παρουσιάζεται το μικρότερο κύκλωμα. Η υλοποίηση που χρησιμοποιήθηκε, στην ουσία, είναι το σχήμα αυτό με επέκταση άλλο τόσο και τη διαφορά ότι έχει τέσσερα επίπεδα 4:2 Adder για να υπολογιστεί σωστά το τελικό αποτέλεσμα σε 36 bits. Θεωρητικά η έξοδος του φίλτρου θα ήταν 32 bits, καθώς τόσο είναι οι έξοδοι των πολλαπλασιαστών. Όμως, χρειάζονται 36 bits για να εκφραστεί σωστά το τελικό αποτέλεσμα αφού υπάρχουν υπερχειλίσεις στις προσθέσεις.



Σχήμα 3.2.1 *Direct υλοποίηση FIR φίλτρου με 8 σταθερούς όρους*

- Transpose μορφή

Εδώ υλοποιείται το Σχήμα 3.2.2, το οποίο πάλι για λόγους ευκρίνειας είναι 8 σταθερών όρων αντί για 16. Στην υλοποίηση μας χρησιμοποιήθηκε αυτό σαν βάση, επεκταμένο κατά άλλους 8 σταθερούς όρους. Το σχήμα δείχνει όλες τις αθροίσεις να υλοποιούνται με 4 σε 2 αθροιστές. Ο αθροιστής 4 σε 2 που χρησιμοποιήθηκε όμως, δίνει έξοδο αυξημένη κατά 1 bit, γεγονός που δεν θέλουμε να συμβαίνει σε κάθε πρόσθεση αλλά μόνο όπου είναι απαραίτητο ώστε να βγει το τελικό αποτέλεσμα κατά 4 bits αυξημένο σε σχέση με τα αποτελέσματα των πολλαπλασιαστών και όχι αυξημένο κατά 15 bits, όσες είναι οι προσθέσεις δηλαδή. Έτσι, χρησιμοποιούνται μόνο τέσσερις 4 σε 2 αθροιστές και οι υπόλοιπες προσθέσεις γίνονται με τη χρήση του αθροιστή με Wallace tree compressor της DesignWare, DW02_tree. [8]



Σχήμα 3.2.2 *Transpose υλοποίηση FIR φίλτρου με 8 σταθερούς όρους*

Για να ελεγχθεί σε ποια σημεία απαιτείται αύξηση bit γίνεται μελέτη της χειρότερης περίπτωσης, δηλαδή αν όλα τα ψηφία είναι 1, για να γίνει εμφανές σε ποια σημεία χρειάζεται να αυξηθούν τα bits.

Έστω ότι η έξοδος του κάθε πολλαπλασιαστή είναι c_i=FFFF hex και s_i=FFFF hex, όπου i=0,…,15.

Στον αθροιστή 14 έχουμε ως είσοδο c15, s15, c14, s14. Έτσι πραγματοποιείται η πράξη *FFFF+FFFF = **1**FFFE* , συνεπώς απαιτείται αύξηση κατά ένα bit.

Στον αθροιστή 13 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c13, s13. Έτσι πραγματοποιείται η πράξη *1FFFE+FFFF=**2**FFFD* , συνεπώς απαιτείται αύξηση κατά ένα bit.

Στον αθροιστή 12 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c12, s12. Έτσι πραγματοποιείται η πράξη *2FFFD+FFFF=3FFFC* , δεν χρειάζεται αύξηση των bits.

Στον αθροιστή 11 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c11, s11. Έτσι πραγματοποιείται η πράξη *3FFFC+FFFF=**4**FFFB* , συνεπώς απαιτείται αύξηση κατά ένα bit.

Στον αθροιστή 10 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c10, s10. Έτσι πραγματοποιείται η πράξη *4FFFB+FFFF=5FFFA* , δεν χρειάζεται αύξηση των bits.

Στον αθροιστή 9 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c9, s9. Έτσι πραγματοποιείται η πράξη *5FFFA+FFFF=6FFF9* , δεν χρειάζεται αύξηση των bits.

Στον αθροιστή 8 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c8, s8. Έτσι πραγματοποιείται η πράξη *6FFF9+FFFF=7FFF8* , δεν χρειάζεται αύξηση των bits.

Στον αθροιστή 7 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c7, s7. Έτσι πραγματοποιείται η πράξη *7FFF8+FFFF=**8**FFF7*, συνεπώς απαιτείται αύξηση κατά ένα bit.

Στον αθροιστή 6 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c6, s6. Έτσι πραγματοποιείται η πράξη *8FFF7+FFFF=9FFF6* , δεν χρειάζεται αύξηση των bits.

Στον αθροιστή 5 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c5, s5. Έτσι πραγματοποιείται η πράξη *9FFF6+FFFF=AFFF5* , δεν χρειάζεται αύξηση των bits.

Στον αθροιστή 4 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c4, s4. Έτσι πραγματοποιείται η πράξη *AFFF5+FFFF=BFFF4* , δεν χρειάζεται αύξηση των bits.

Στον αθροιστή 3 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c3, s3. Έτσι πραγματοποιείται η πράξη *BFFF4+FFFF=CFFF3* , δεν χρειάζεται αύξηση των bits.

Στον αθροιστή 2 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c2, s2. Έτσι πραγματοποιείται η πράξη *CFFF3+FFFF=DFFF2* , δεν χρειάζεται αύξηση των bits.

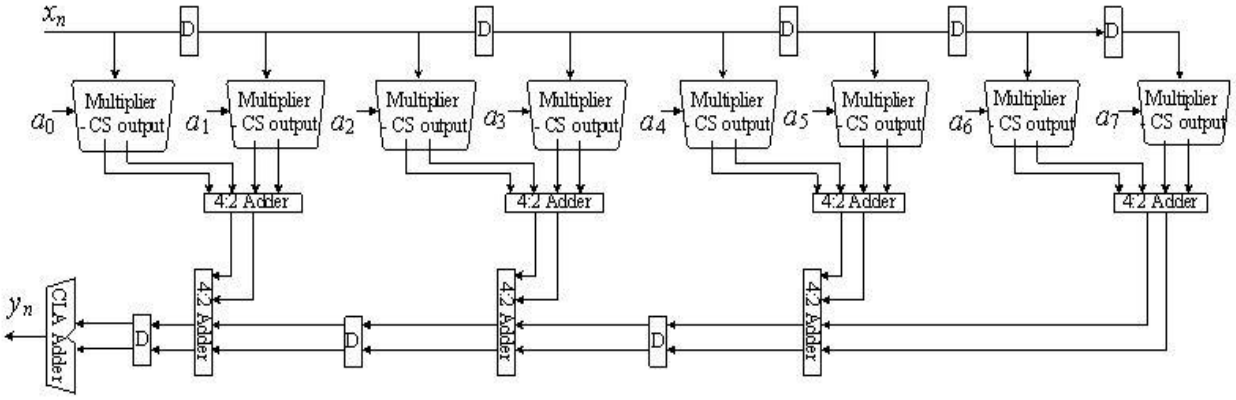
Στον αθροιστή 1 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c1, s1. Έτσι πραγματοποιείται η πράξη *DFFF2+FFFF=EFFF1* , δεν χρειάζεται αύξηση των bits.

Στον αθροιστή 0 έχουμε ως είσοδο την έξοδο του προηγούμενου και τα c0, s0. Έτσι πραγματοποιείται η πράξη *EFFF1+FFFF=FFFF0* , δεν χρειάζεται αύξηση των bits.

Επομένως, σύμφωνα με τα παραπάνω, στους αθροιστές 14, 13, 11 και 7 (ξεκινώντας την αρίθμηση από το 0 και με αύξηση τιμής από αριστερά προς τα δεξιά) θα χρησιμοποιήσουμε τον 4 σε 2 adder και στους υπόλοιπους τον αθροιστή DW02_tree.

- Mixed μορφή

Η Mixed μορφή αποτελεί ένα συνδυασμό των δύο προηγούμενων και παρουσιάζεται στο Σχήμα 3.2.3, πάλι με 8 σταθερούς όρους αντί 16, οπότε το υλοποιημένο κύκλωμα είναι επεκταμένο. Αντίστοιχα με την transpose μορφή, δεν γίνονται όλες οι προσθέσεις με 4 σε 2 αθροιστές όπως φαίνεται στο Σχήμα 3.2.3. Σε αυτή τη περίπτωση αφήνεται το πρώτο στάδιο άθροισης που είναι όμοιο με της direct μορφής και δίνει αύξηση κατά 1 bit και τροποποιείται το επόμενο κομμάτι του σχήματος που είναι όμοιο με της transpose. Επαναλαμβάνοντας έλεγχο όπως παραπάνω, προκύπτει ότι 4 σε 2 αθροιστές χρειάζονται στο οριζόντιο επίπεδο κάτω από τους πολλαπλασιαστές και στους αθροιστές 0, 4 και 6 (ξεκινώντας την αρίθμηση από το 0 και με αύξηση τιμής από αριστερά προς τα δεξιά). Οι υπόλοιπες προσθέσεις γίνονται με τον Wallace tree compressor DW02_tree. [8]



Σχήμα 3.2.3 *Mixed υλοποίηση FIR φίλτρου με 8 σταθερούς όρους*

3.2.2 Υλοποίηση φίλτρου FIR με NR4SD πολλαπλασιαστή

Σε αυτή την υλοποίηση επαναλαμβάνονται οι τρεις προηγούμενες περιπτώσεις –direct, transpose, mixed- με τη διαφορά ότι αντί για τον MB πολλαπλασιαστή, χρησιμοποιείται ο πολλαπλασιαστής με NR4SD προ-κωδικοποίηση που παρουσιάστηκε στην Ενότητα 2.4.2.

3.2.3 Υλοποίηση φίλτρου FIR με truncated MB πολλαπλασιαστή

Εδώ έχουμε υλοποίηση των περιπτώσεων direct, transpose, mixed με MB πολλαπλασιαστή σταθερού μήκους, βλέπε Ενότητα 2.4.3. Η έξοδος τώρα είναι των 20 bits, καθώς οι είσοδοι είναι 16 bits οπότε κάθε πολλαπλασιασμός δίνει αποτέλεσμα 16 bits και το κύκλωμα προβλέπει μια αύξηση 4 bits με τους 4 σε 2 αθροιστές.

3.2.3 Υλοποίηση φίλτρου FIR με truncated NR4SD πολλαπλασιαστή

Αντίστοιχα με την υλοποίηση με truncated πολλαπλασιαστή MB, με τη διαφορά όμως ότι χρησιμοποιείται προ-κωδικοποίηση NR4SD.

4. ΘΕΩΡΗΤΙΚΗ ΑΝΑΛΥΣΗ AREA ΚΑΙ DELAY ΦΙΛΤΡΟΥ FIR

4.1 Συμβατικές υλοποιήσεις φίλτρου με MB πολλαπλασιαστή

Σε αυτή την ενότητα υπολογίζονται θεωρητικά το area και το delay για κάθε μια από τις υλοποιήσεις φίλτρου –direct, transpose, mixed- που παρουσιάστηκαν στην ενότητα 3.2.1.

- DIRECT FILTER

Το φίλτρο αποτελείται από k=16 σταθερούς όρους και οι είσοδοι είναι των n=16bits.

- Η εξίσωση υπολογισμού του area του direct φίλτρου είναι η ακόλουθη:

$$\mathbf{A_{DFIR}=k\cdot A_{MBWM}+ A_{Adder}+ (k-1)\cdot n\cdot A_{DFF}+ A_{CLA} + (2n+4)\cdot A_{DFF}} \quad \mathbf{(1)}$$

Όπου:

- $A_{MBWM}=A_{MBE}+A_{PPG}+A_{TREE}=5\cdot\frac{n}{2}+5(n+1)\cdot(\frac{n}{2})+(7\cdot n+17)(\frac{n}{2})-11$

  → $A_{MBWM}=(12\cdot n+27)(\frac{n}{2})-11$ (2)

- $A_{Adder}=8\cdot A_{Adder\_0}+4\cdot A_{Adder\_1}+2\cdot A_{Adder\_2}+A_{Adder\_3}$

Ο Adder_0 είναι ένας 4:2 adder με εισόδους των n'=32bits, αντίστοιχα ο Adder_1 είναι ένας 4:2 adder των n'+1=33bits, ο Adder_2 των n'+2=34bits και ο Adder_3 των n'+3=35bits. Ο γενικός υπολογισμός του area ενός 4:2 adder είναι $A_{Adder\_42}=2\cdot n\cdot A_{FA}$, όπου $A_{FA}=7$ όπως προκύπτει από τον Πίνακα 4.1.

Components	***Area (unit gates)***	***Delay***
NAND-2, NOR-2	1	1
XOR, XNOR	2	2
Half Adder	3	2
Full Adder	7	4
2 to 1 multiplexer (MUX)	3	2
Circuit of prefix adder (p,g)	3	2
Register	6	2

Πίνακας 4.1 *Μετρικές για Area Delay για μονάδα πύλης*

Συνεπώς,

$A_{Adder}=8\cdot 2\cdot n'\cdot A_{FA}+4\cdot 2\cdot(n'+1)\cdot A_{FA}+2\cdot 2\cdot(n'+2)\cdot A_{FA}+2\cdot(n'+3)\cdot A_{FA}$

→ $A_{Adder}=8\cdot 14\cdot 2\cdot n+4\cdot 14\cdot(2\cdot n+1)+2\cdot 14\cdot(2\cdot n+2)+14\cdot(2\cdot n+3)$

→ $A_{Adder}=420\cdot n+154$ (3)

- $A_{CLA}=(A_{CLA})_{32bits}+(A_{CLA})_{4bits}$

Ο CLA είναι 36 bits και καθώς δεν είναι δύναμη του 2 ο υπολογισμός του area γίνεται τμηματικά. Αρχικά υπολογίζουμε το area για cla των 32bits ($\log_2 32=5$) και προσθέτουμε 4 bits στο κύκλωμα που αντιστοιχούν σε 8 μονάδες PG, 4 μονάδες HA και 4 πύλες XOR.

$A_{CLA}= (A_{CLA})_{32bits} + 8A_{PG}+4A_{HA}+4A_{XOR}$

$\rightarrow A_{CLA} = [5 \cdot (2 \cdot n) - 2 + 3\frac{2 \cdot n}{2} log_2(2 \cdot n)] + (8 \cdot 3+4 \cdot 3+4 \cdot 2)$

$\rightarrow A_{CLA} =10 \cdot n+3 \cdot n \cdot \log_2(2 \cdot n)+42$ (4)

- $A_{DFF}=6$ (5)

Επομένως με βάση τις σχέσεις (2), (3), (4),(5) υπολογίζεται η (1) και προκύπτει:

$$A_{DFIR} = k\left[(12n + 27)\left(\frac{n}{2}\right) - 11\right] + 420n + 154 + 6(k - 1)n + 10n + 3n\log_2(2n) + \\ + 42 + (2n + 4) \cdot 6$$

$$\rightarrow \mathbf{A_{DFIR} = k\left[(12n + 27)\left(\frac{n}{2}\right) - 11\right] + 436n + 220 + 6kn + 3n\log_2(2n)} \quad \mathbf{(6)}$$

Αντικαθιστώντας για τις τιμές των k και n, έχουμε:

$$\mathbf{A_{DFIR}=36828} \quad \mathbf{(7)}$$

- Η εξίσωση υπολογισμού του delay του direct φίλτρου είναι η ακόλουθη:

$$\mathbf{T_{DFIR}= T_{MBWM}+ T_{Adder}+ T_{DFF}+ T_{CLA}} \quad \mathbf{(8)}$$

Όπου:

- $T_{MBWM} =T_{MBE\&PPG} + T_{TREE} = 5 + 4D(\frac{n}{2} + 2)$

  $T_{MBWM} = 5 + 4D(\frac{n}{2} + 2)$ (9)

- $T_{Adder}=T_{Adder\_0} + T_{Adder\_1} + T_{Adder\_2} + T_{Adder\_3}$

Ο γενικός υπολογισμός του delay ενός 4:2 adder είναι $T_{Adder\_42}=2 \cdot T_{FA}$,όπου $T_{FA}=4$. Δηλαδή η καθυστέρηση είναι ανεξάρτητη από τον αριθμό των bits. Όμως, η υλοποίηση μπορεί να γίνει σε δενδρική μορφή με αποτέλεσμα να προκύπτει ότι $T_{Adder\_42}=6$. Συνεπώς,

$T_{Adder\_0} = T_{Adder\_1} = T_{Adder\_2} = T_{Adder\_3} =6$ (10)

$\rightarrow T_{Adder} = 24$

- $T_{CLA}=(T_{CLA})_{32bits} + (T_{CLA})_{4bits}$

Το delay ενός CLA ισούται με $T_{CLA}=T_{HA} + \log_2 n \cdot T_{PG}+T_{XOR}$. Στην περίπτωση μας όμως η είσοδος του CLA είναι 36 bits και καθώς δεν είναι δύναμη του 2, ο υπολογισμός του delay γίνεται σε δύο φάσεις. Υπολογίζεται η καθυστέρηση για 32 bits, τα οποία είναι δύναμη του 2, και σε αυτή τη καθυστέρηση προστίθεται η καθυστέρηση που προκύπτει αν προστεθούν στο γνωστό κύκλωμα άλλα 4 bits. Η τελευταία καθυστέρηση ισούται με την προσθήκη ενός επιπλέον επιπέδου PG, συνεπώς ισούται με T_{PG}.

$T_{CLA}=T_{HA} + [\log_2 (2n)+1]\cdot T_{PG}+T_{XOR}$, όπου $T_{HA}=2$, $T_{PG}=2$ και $T_{XOR}=2$

→ $T_{CLA}=2 \log_2 (2n)+6$ (11)

- $T_{DFF}=2$ (12)

Επομένως με βάση τις σχέσεις (9), (10), (11),(12) υπολογίζεται η (8) και προκύπτει:

$T_{DFIR}= 5 + 4D(\frac{n}{2} + 2) + 24 + 2 + 2 \log2 (2n) + 6$

→ $\mathbf{T_{DFIR}= 37+4D(\frac{n}{2} + 2) + 2\log_2 2n}$ **(13)**

Depth of Wallace Tree							
h	4	5-6	7-9	10-13	14-19	20-28	29-42
D(h)	2	3	4	5	6	7	8

Πίνακας 4.2 Βάθος Wallace tree

Αντικαθιστώντας για n=16bits είσοδο και, σύμφωνα με τον Πίνακα 4.2, D(10)=5, έχουμε:

$T_{DFIR}= 67$ **(14)**

- TRANSPOSE FILTER

Το φίλτρο αποτελείται από k=16 σταθερούς όρους και οι είσοδοι είναι n=16bits.

- Η εξίσωση υπολογισμού του area του transpose φίλτρου είναι η ακόλουθη:

$A_{TFIR}=k\cdot A_{MBWM}+ A_{Adder} + A_{DFF\_s} + A_{CLA}$ **(15)**

Όπου:

- $A_{Adder}=A_{Adder\_0} + A_{Adder\_1} + 2\cdot A_{Adder\_2} + 4\cdot A_{Adder\_3}+ 7\cdot A_{Adder\_4}$

Ο Adder_0 είναι ένας 4:2 adder με εισόδους των 2n bits, αντίστοιχα ο Adder_1 είναι ένας 4:2 adder των 2n+1bits, ο Adder_2 των 2n+2 bits ,ο Adder_3 των 2n+3 bits και ο Adder_4 των 2n+4 bits. Ο γενικός υπολογισμός του area ενός 4:2 adder είναι $A_{Adder\_42}=2\cdot n'\cdot A_{FA}=14\cdot n'$, όπου $A_{FA}=7$ και n' ο αριθμός των bits της εισόδου.

Συνεπώς,

$$A_{Adder}=14\cdot 2\cdot n + 14\cdot (2\cdot n+1) + 2\cdot 14\cdot (2\cdot n+2) + 4\cdot 14\cdot (2\cdot n+3) + 7\cdot 14\cdot (2\cdot n+4)$$

$$\rightarrow A_{Adder}= 420\cdot n + 630 \qquad (16)$$

- $A_{DFF\_s}= n\cdot A_{DFF} + (2n+1)\cdot A_{DFF} + 2\cdot(2n+2)\cdot A_{DFF} + 4\cdot(2n+3)\cdot A_{DFF} + 8\cdot(2n+4)\cdot A_{DFF}$

Όπου $A_{DFF}=6$, άρα:

$$A_{DFF\_s}= 186n +294 \qquad (17)$$

Επομένως με βάση τις σχέσεις (2), (4), (16), (17) υπολογίζεται η (15) και προκύπτει:

$$A_{TFIR}= k\left[(12n+27)\left(\frac{n}{2}\right)-11\right] + 420n + 630 + 186n + 294 + 10n + +3n\log_2(2n) + 42$$

$$\rightarrow \mathbf{A_{TFIR}= k\left[(12n+27)\left(\frac{n}{2}\right)-11\right] + 3n\log_2(2n) + 616n + 966} \qquad \mathbf{(18)}$$

Αντικαθιστώντας για τις τιμές των k και n, έχουμε:

$$\mathbf{A_{TFIR}=38918} \qquad \mathbf{(19)}$$

- Η εξίσωση υπολογισμού του delay του transpose φίλτρου είναι η ακόλουθη:

$$\mathbf{T_{TFIR}= T_{MBWM}+ T_{Adder\_3}+ T_{DFF}} \qquad \mathbf{(20)}$$

Όπου:

- $T_{Adder\_3}=6$ (21)

Επομένως με βάση τις σχέσεις (9), (12), (21) υπολογίζεται η (20) και προκύπτει:

$$T_{TFIR} = 5 + 4D\left(\frac{n}{2}+2\right) + 6 + 2$$

$$\rightarrow \mathbf{T_{TFIR} = 4D\left(\frac{n}{2}+2\right) + 13} \qquad \mathbf{(22)}$$

Αντικαθιστώντας για n=16bits είσοδο και σύμφωνα με τον Πίνακα 4.2, έχουμε:

$$\mathbf{T_{TFIR}=33} \qquad \mathbf{(23)}$$

***Στην transpose μορφή παραλείπεται το delay του CLA καθώς υπάρχει καθυστέρηση (D flip flop) πριν τον τελικό αθροιστή και η συνολική καθυστέρηση πριν τον CLA είναι μεγαλύτερη από την καθυστέρηση του ίδιου.*

Επίσης, μπορούμε να απλοποιήσουμε το κύκλωμα του Modified booth- Wallace tree multiplier και η υλοποίηση του Wallace tree να πραγματοποιείται εκτός του multiplier μαζί με το tree του αθροιστή, για καλύτερη ορθότητα με αποτέλεσμα το delay να μειώνεται. Πιο συγκεκριμένα:

$$\mathbf{T_{TFIR} = T_{MBM} + T_{TREE} + T_{DFF}} \quad \mathbf{(24)}$$

Όπου:

- $T_{MBM}=5$ (25)
- $T_{TREE} = 4D(\frac{n}{2}+2)$ (26)

Συνεπώς, με βάση τις σχέσεις (12),(25),(26) καταλήγουμε στο αποτέλεσμα:

$$T_{TFIR} = 5 + 4D\left(\frac{n}{2}+2\right) + 2$$

$$\mathbf{\rightarrow T_{TFIR} = 4D\left(\frac{n}{2}+2\right) + 7} \quad \mathbf{(27)}$$

Αντικαθιστώντας για n=16bits είσοδο και σύμφωνα με τον Πίνακα 4.2, έχουμε:

$$\mathbf{T_{TFIR}=27} \quad \mathbf{(28)}$$

Διαπιστώνουμε, λοιπόν, ότι με τη χρήση της παραπάνω υλοποίησης στην transpose μορφή επιτυγχάνεται μείωση του χρόνου καθυστέρησης.

- MIXED FILTER

Το φίλτρο αποτελείται από k=16 σταθερούς όρους και οι είσοδοι είναι n=16bits.

- Η εξίσωση υπολογισμού του area του mixed φίλτρου είναι η ακόλουθη:

$$\mathbf{A_{mixFIR}=k\cdot A_{MBWM} + A_{Adder} + A_{DFF\_s} + A_{CLA}} \quad \mathbf{(29)}$$

Όπου:

- $A_{Adder} = 8\cdot A_{Adder\_0} + A_{Adder\_1} + 2\cdot A_{Adder\_2} + 4\cdot A_{Adder\_3}$

$A_{Adder}=8\cdot 14\cdot 2\cdot n + 14\cdot (2\cdot n+1) + 2\cdot 14\cdot (2\cdot n+2) + 4\cdot 14\cdot (2\cdot n+3)$

→ $A_{Adder}= 420\cdot n + 238$ (30)

- $A_{DFF\_s}= 9\cdot n\cdot A_{DFF} + 2\cdot(2n+2)\cdot A_{DFF} + 4\cdot(2n+3)\cdot A_{DFF} + (2n+4)\cdot A_{DFF}$

Όπου $A_{DFF}=6$, άρα:

$$A_{DFF\_s}= 138n +120 \qquad (31)$$

Επομένως με βάση τις σχέσεις (2), (4), (30), (31) υπολογίζεται η (29) και προκύπτει:

$$A_{mixFIR}= k\left[(12n + 27)\left(\frac{n}{2}\right) - 11\right] + 420n + 238 + 138n + 120 + 10n + \\ +3n\log_2(2n) + 42$$

$$\rightarrow \mathbf{A_{mixFIR}= k\left[(12n + 27)\left(\frac{n}{2}\right) - 11\right] + 3n\log_2(2n) + 568n + 400} \qquad \mathbf{(32)}$$

Αντικαθιστώντας για τις τιμές των k και n, έχουμε:

$$\mathbf{A_{mixFIR}= 37584} \qquad \mathbf{(33)}$$

- Η εξίσωση υπολογισμού του delay του mixed φίλτρου είναι η ακόλουθη:

$$\mathbf{T_{mixFIR}= T_{MBWM}+ T_{Adder\_0}+ T_{Adder\_3} + T_{DFF}} \qquad \mathbf{(34)}$$

Επομένως με βάση τις σχέσεις (9),(10), (11), (12) υπολογίζεται η (34) και προκύπτει:

$$T_{mixFIR} = 5 + 4D(\frac{n}{2} + 2) + 6 + 6 + 2$$

$$\rightarrow \mathbf{T_{mixFIR} = 4D(\frac{n}{2} + 2) + 19} \qquad \mathbf{(35)}$$

Αντικαθιστώντας για n=16bits είσοδο και, σύμφωνα με τον Πίνακα 4.2 το D(18)=6, έχουμε:

$$\mathbf{T_{mixFIR}=39} \qquad \mathbf{(36)}$$

Και στην περίπτωση του mixed filter μπορούμε να απλοποιήσουμε το κύκλωμα για βέλτιστο υπολογισμό του delay. Πιο συγκεκριμένα μπορούμε να βάλουμε σε ένα Wallace tree τους όρους από δύο πολλαπλασιαστές και τις δύο αθροιστικές μονάδες μαζί. Έτσι έχουμε:

$$\mathbf{T_{mixFIR}= T_{MBM}+ T_{TREE} + T_{DFF}} \qquad \mathbf{(37)}$$

Όπου:

- $T_{TREE} = 4D(n + 4 + 2)$ (38)

Συνεπώς, με βάση τις σχέσεις (12),(25),(38) καταλήγουμε στο αποτέλεσμα:

$T_{TFIR} = 5 + 4D(n+6) + 2$

$\rightarrow \mathbf{T_{TFIR} = 4D(n+6) + 7}$ **(39)**

Αντικαθιστώντας για n=16bits είσοδο και σύμφωνα με πίνακα D(22)=7, έχουμε:

$\mathbf{T_{mixFIR}=35}$ **(40)**

ΣΥΜΠΕΡΑΣΜΑ

Σύμφωνα με τα παραπάνω, η απλοποιημένη transpose μορφή φίλτρου παρουσιάζει μικρότερο delay από την direct αλλά και από την mixed μορφή, οπότε προτιμάται στην περίπτωση που μας ενδιαφέρει η ελάχιστη δυνατή τιμή του delay. Αντίθετα, στην περίπτωση του area η direct μορφή είναι καλύτερη καθώς παρουσιάζει μικρότερη τιμή.

Σε γενικότερη περίπτωση, η επιλογή της χρήσης μιας εκ των παραπάνω μορφών φίλτρου εξαρτάται από την μετρική A·T·cc, δηλαδή του γινομένου area επί delay επί αριθμό των κύκλων που απαιτείται ώστε να υπολογιστεί το πρώτο αποτέλεσμα, επιθυμώντας την ελάχιστη τιμή του γινομένου αυτού. Όμως, στην προκειμένη μπορούμε να υπολογίσουμε μόνο το γινόμενο area επί delay καθώς απλοποιείται ο αριθμός των κύκλων αφού και στις τρεις περιπτώσεις χρειάζονται 16 κύκλοι. Έτσι:

$(A\cdot T\cdot)_{direct} = 2467476$ (41)

$(A\cdot T)_{transpose} = 1050785$ (42)

$(A\cdot T)_{mixed} = 1315440$ (43)

Συνεπώς, η transpose μορφή είναι καλύτερη καθώς παρουσιάζει βέλτιστη τιμή της μετρικής A·T, έπειτα προτιμάται η mixed και τελευταία η direct.

Τα παραπάνω αποτελέσματα συνοψίζονται στον ακόλουθο πίκα:

	Area	***Delay***	***Area × Delay***
Direct	*36828*	*67*	*2467476*
Transpose	*38918*	*27*	*1050785*
Mixed	*37584*	*35*	*1315440*

Πίνακας 4.3 *Σύγκριση θεωρητικών αποτελεσμάτων φίλτρου FIR πλήρους ακρίβειας*

4.2 Υλοποιήσεις φίλτρου με πολλαπλασιαστή NR4SD προ-κωδικοποίησης

Σε αυτή την ενότητα υπολογίζονται θεωρητικά το area και το delay για κάθε μια από τις υλοποιήσεις φίλτρου –direct, transpose, mixed- που παρουσιάστηκαν στην ενότητα 3.2.2. Οι θεωρητικοί υπολογισμοί είναι αντίστοιχοι με πριν, με μόνη διαφορά ότι χρησιμοποιείται διαφορετικός πολλαπλασιαστής και κατά επέκταση τροποποιείται το area και το delay του.

- DIRECT FILTER

Το φίλτρο αποτελείται από k=16 σταθερούς όρους και οι είσοδοι είναι των n=16bits.

- Η εξίσωση υπολογισμού του area του direct φίλτρου είναι η ακόλουθη:

$$\mathbf{A_{DFIR}=k\cdot A_{NR4SDWM}+ A_{Adder}+ (k-1)\cdot n\cdot A_{DFF}+ A_{CLA} + (2n+4)\cdot A_{DFF}} \quad \mathbf{(44)}$$

Όπου:

- $A_{NR4SDWM}=A_{NR4SD\ E}+A_{PPG}+A_{TREE}= 3\cdot\frac{n}{2}+5(n+1)\cdot(\frac{n}{2})+(7\cdot n+17)(\frac{n}{2})-11$

→ $A_{NR4SDWM}=(12\cdot n+25)(\frac{n}{2})-11$ (45)

Επομένως με βάση τις σχέσεις (3), (4), (5),(45) υπολογίζεται η (44) και προκύπτει:

$$\mathrm{A_{DFIR}} = k\left[(12n+25)\left(\frac{n}{2}\right)-11\right]+420\mathrm{n}+154+6(\mathrm{k}-1)\mathrm{n}+10\mathrm{n}+3\mathrm{n}\log_2(2\mathrm{n})+ \\ +42+\ (2\mathrm{n}+4)\cdot 6$$

$$\rightarrow \mathbf{A_{DFIR}} = \boldsymbol{k}\left[(\mathbf{12}\boldsymbol{n}+\mathbf{25})\left(\frac{\boldsymbol{n}}{\mathbf{2}}\right)-\mathbf{11}\right]+\mathbf{436n}+\mathbf{220}+\mathbf{6kn}+\mathbf{3n}\log_{\mathbf{2}}(\mathbf{2n}) \quad \mathbf{(46)}$$

Αντικαθιστώντας για τις τιμές των k και n, έχουμε:

$$\mathbf{A_{DFIR}=36572} \quad \mathbf{(47)}$$

- Η εξίσωση υπολογισμού του delay του direct φίλτρου είναι η ακόλουθη:

$$\mathbf{T_{DFIR}= T_{NR4SDWM}+ T_{Adder}+ T_{DFF}+ T_{CLA}} \quad \mathbf{(48)}$$

Όπου:

- $T_{NR4SDWM} =T_{NR4SDE\&PPG} + T_{TREE} = 4+4D(\frac{n}{2}+2)$

→ $T_{NR4SDWM} = 4+4D(\frac{n}{2}+2)$ (49)

Επομένως με βάση τις σχέσεις (10), (11), (12), (49) υπολογίζεται η (48) και προκύπτει:

$T_{DFIR}= 4 + 4D(\frac{n}{2} + 2) + 24 + 2 + 2\ \log_2 n + 6$

→ $T_{DFIR}= 36+4D(\frac{n}{2} + 2) + 2\log_2 n$ (50)

Αντικαθιστώντας για n=16bits είσοδο και, σύμφωνα με τον Πίνακα 4.2, D(10)=5, έχουμε:

$T_{DFIR}= 64$ (51)

- TRANSPOSE FILTER

Το φίλτρο αποτελείται από k=16 σταθερούς όρους και οι είσοδοι είναι n=16bits.

- Η εξίσωση υπολογισμού του area του transpose φίλτρου είναι η ακόλουθη:

$A_{TFIR}=k\cdot A_{NR4SDWM}+ A_{Adder} + A_{DFF\_s} + A_{CLA}$ (52)

Επομένως με βάση τις σχέσεις (4), (16), (17), (45) υπολογίζεται η (52) και προκύπτει:

$A_{TFIR}= k\left[(12n + 25)\left(\frac{n}{2}\right) - 11\right] + 420n + 630 + 186n + 294 + 10n +$
$+3n\log_2(2n) + 42$

→$A_{TFIR}= k\left[(12n + 25)\left(\frac{n}{2}\right) - 11\right] + 3n\log_2(2n) + 616n + 966$ (53)

Αντικαθιστώντας για τις τιμές των k και n, έχουμε:

$A_{TFIR}=38662$ (54)

- Η εξίσωση υπολογισμού του delay του transpose φίλτρου είναι η ακόλουθη:

$T_{TFIR}= T_{NR4SDWM}+ T_{Adder\_3}+ T_{DFF}$ (55)

Επομένως με βάση τις σχέσεις (12), (21), (49) υπολογίζεται η (55) και προκύπτει:

$T_{TFIR} = 4 + 4D\left(\frac{n}{2} + 2\right) + 6 + 2$

→$T_{TFIR} = 4D(\frac{n}{2} + 2) + 12$ (56)

Αντικαθιστώντας για n=16bits είσοδο και σύμφωνα με τον Πίνακα 4.2, έχουμε:

$T_{TFIR}=32$ (57)

***Στην transpose μορφή παραλείπεται το delay του CLA καθώς υπάρχει καθυστέρηση (D flip flop) πριν τον τελικό αθροιστή και η συνολική καθυστέρηση πριν τον CLA είναι μεγαλύτερη από την καθυστέρηση του ίδιου.*

Επίσης, μπορούμε να απλοποιήσουμε το κύκλωμα του Modified booth- Wallace tree multiplier και η υλοποίηση του Wallace tree να πραγματοποιείται εκτός του multiplier μαζί με το tree του αθροιστή, για καλύτερη ορθότητα με αποτέλεσμα το delay να μειώνεται. Πιο συγκεκριμένα:

$$\mathbf{T_{TFIR} = T_{NR4SDBM} + T_{TREE} + T_{DFF}} \qquad \mathbf{(58)}$$

Όπου:

- $T_{NR4SDM}=4$ (59)

Συνεπώς, με βάση τις σχέσεις (12),(26),(59) καταλήγουμε στο αποτέλεσμα:

$$T_{TFIR} = 4+4D\left(\frac{n}{2}+2\right)+2$$

$$\mathbf{\rightarrow T_{TFIR} = 4D\left(\frac{n}{2}+2\right)+6} \qquad \mathbf{(60)}$$

Αντικαθιστώντας για n=16 bits είσοδο και σύμφωνα με τον Πίνακα 4.2, έχουμε:

$$\mathbf{T_{TFIR}=26} \qquad \mathbf{(61)}$$

Διαπιστώνουμε, λοιπόν, ότι με τη χρήση της παραπάνω υλοποίησης στην transpose μορφή επιτυγχάνεται μείωση του χρόνου καθυστέρησης.

- MIXED FILTER

Το φίλτρο αποτελείται από k=16 σταθερούς όρους και οι είσοδοι είναι n=16bits.

- Η εξίσωση υπολογισμού του area του mixed φίλτρου είναι η ακόλουθη:

$$\mathbf{A_{mixFIR}=k\cdot A_{NR4SDWM} + A_{Adder} + A_{DFF\_s} + A_{CLA}} \qquad \mathbf{(62)}$$

Επομένως με βάση τις σχέσεις (4), (30), (31), (45) υπολογίζεται η (62) και προκύπτει:

$$A_{mixFIR}= k\left[(12n+25)\left(\frac{n}{2}\right)-11\right]++420\mathrm{n}+238+138\mathrm{n}+120+10\mathrm{n}+ \\ +3\mathrm{n}\log_2(2\mathrm{n})+42$$

$$\rightarrow A_{mixFIR} = \boldsymbol{k}\left[(\mathbf{12}\boldsymbol{n} + \mathbf{25})\left(\frac{\boldsymbol{n}}{\mathbf{2}}\right) - \mathbf{11}\right] + \mathbf{3nlog_2(2n)} + \mathbf{568n} + \mathbf{400} \qquad \textbf{(63)}$$

Αντικαθιστώντας για τις τιμές των k και n, έχουμε:

$$\mathbf{A_{mixFIR} = 37328} \qquad \textbf{(64)}$$

- Η εξίσωση υπολογισμού του delay του mixed φίλτρου είναι η ακόλουθη:

$$\mathbf{T_{mixFIR} = T_{NR4SDWM} + T_{Adder\_0} + T_{Adder\_3} + T_{DFF}} \qquad \textbf{(65)}$$

Επομένως με βάση τις σχέσεις (10), (11), (12), (49) υπολογίζεται η (65) και προκύπτει:

$$T_{mixFIR} = 4 + 4D(\frac{n}{2} + 2) + 6 + 6 + 2$$

$$\rightarrow \mathbf{T_{mixFIR} = 4D(\frac{n}{2} + 2) + 18} \qquad \textbf{(66)}$$

Αντικαθιστώντας για n=16bits είσοδο και, σύμφωνα με τον Πίνακα 4.2 το D(18)=6, έχουμε:

$$\mathbf{T_{mixFIR} = 38} \qquad \textbf{(67)}$$

Και στην περίπτωση του mixed filter μπορούμε να απλοποιήσουμε το κύκλωμα για βέλτιστο υπολογισμό του delay. Πιο συγκεκριμένα μπορούμε να βάλουμε σε ένα Wallace tree τους όρους από δύο πολλαπλασιαστές και τις δύο αθροιστικές μονάδες μαζί. Έτσι έχουμε:

$$\mathbf{T_{mixFIR} = T_{NR4SDBM} + T_{TREE} + T_{DFF}} \qquad \textbf{(68)}$$

Όπου:

Συνεπώς, με βάση τις σχέσεις (12),(26),(59) καταλήγουμε στο αποτέλεσμα:

$$T_{TFIR} = 4 + 4D(n + 6) + 2$$

$$\rightarrow \mathbf{T_{TFIR} = 4D(n + 6) + 6} \qquad \textbf{(69)}$$

Αντικαθιστώντας για n=16bits είσοδο και σύμφωνα με πίνακα D(22)=7, έχουμε:

$$\mathbf{T_{mixFIR} = 35} \qquad \textbf{(70)}$$

ΣΥΜΠΕΡΑΣΜΑ

Αντίστοιχα με τη περίπτωση του MB πολλαπλασιαστή, η απλοποιημένη transpose μορφή φίλτρου παρουσιάζει μικρότερο delay από την direct αλλά και από την mixed μορφή, οπότε προτιμάται στην περίπτωση που μας ενδιαφέρει η ελάχιστη δυνατή τιμή του delay. Αντίθετα, στην περίπτωση του area η direct μορφή είναι καλύτερη καθώς παρουσιάζει μικρότερη τιμή.

Γενικότερο κριτήριο επιλογής είναι η μετρική A·T·cc, επιθυμώντας την ελάχιστη τιμή του γινομένου αυτού. Μπορούμε να υπολογίσουμε μόνο το γινόμενο area επί delay καθώς απλοποιείται ο αριθμός των κύκλων αφού και στις τρεις περιπτώσεις χρειάζονται 16 κύκλοι. Έτσι:

$$(A\cdot T\cdot)_{direct} = 2340608 \qquad (71)$$

$$(A\cdot T)_{transpose} = 1005212 \qquad (72)$$

$$(A\cdot T)_{mixed} = 1306480 \qquad (73)$$

Συνεπώς, και με τον NR4SD πολλαπλασιαστή η transpose μορφή είναι καλύτερη καθώς παρουσιάζει βέλτιστη τιμή της μετρικής A·T, έπειτα προτιμάται η mixed και τελευταία η direct.

Τα παραπάνω αποτελέσματα συνοψίζονται στον ακόλουθο πίκα:

	Area	***Delay***	***Area × Delay***
Direct	*36572*	*64*	*2340608*
Transpose	*38662*	*26*	*1005212*
Mixed	*37328*	*35*	*1306480*

Πίνακας 4.3 *Σύγκριση θεωρητικών αποτελεσμάτων φίλτρου FIR περικομμένης ακρίβειας*

4.3 Θεωρητική σύγκριση υλοποίησης φίλτρου FIR μεταξύ MB και NR4SD πολλαπλασιαστή

Όπως φαίνεται από τα παραπάνω αποτελέσματα, η κάθε περίπτωση υλοποίησης φίλτρου με πολλαπλασιαστή προ-κωδικοποίησης NR4SD δίνει καλύτερα αποτελέσματα της μετρικής A·T·cc σε σχέση την αντίστοιχη υλοποίηση φίλτρου με πολλαπλασιαστή σε MB προ-κωδικοποίηση. Η καλύτερη όλων των περιπτώσεων είναι η transpose υλοποίηση με NR4SD πολλαπλασιαστή.

5. ΠΕΙΡΑΜΑΤΙΚΗ ΑΝΑΛΥΣΗ

5.1 Οργάνωση πειραμάτων

Οι τοπολογίες που παρουσιάστηκαν στην προηγούμενη ενότητα, περιγράφηκαν σε γλώσσα υλικού Verilog και η υλοποίηση τους σε ASIC πραγματοποιήθηκε από το Synopsys Design Compiler, κάνοντας χρήση της βιβλιοθήκης κελιών Faraday 90nm και της εντολής –compile_ultra για βέλτιστα αποτελέσματα. Τα κυκλώματα που παρήχθησαν, προσομοιώθηκαν από το περιβάλλον ModelSim για επιβεβαίωση της ορθής λειτουργίας τους.

Από το περιβάλλον του Design Compiler, εξήχθησαν οι τιμές καθυστέρησης και επιφάνειας κάθε κυκλώματος, ενώ από το περιβάλλον του Synopsys PrimePower (με βάση το αρχείο .vcd που προέκυπτε από κάθε προσομοίωση του ModelSim) υπολογίστηκε η μέση κατανάλωση κάθε κυκλώματος.

Για την προσομοίωση της λειτουργίας κάθε κυκλώματος χρησιμοποιήθηκαν ως είσοδοι 20480 τυχαίες τιμές για το x και σταθερές τιμές για τους 16 σταθερούς όρους του φίλτρου. Σε όλα τα κυκλώματα τα μήκη λέξης ήταν n=16 bits.

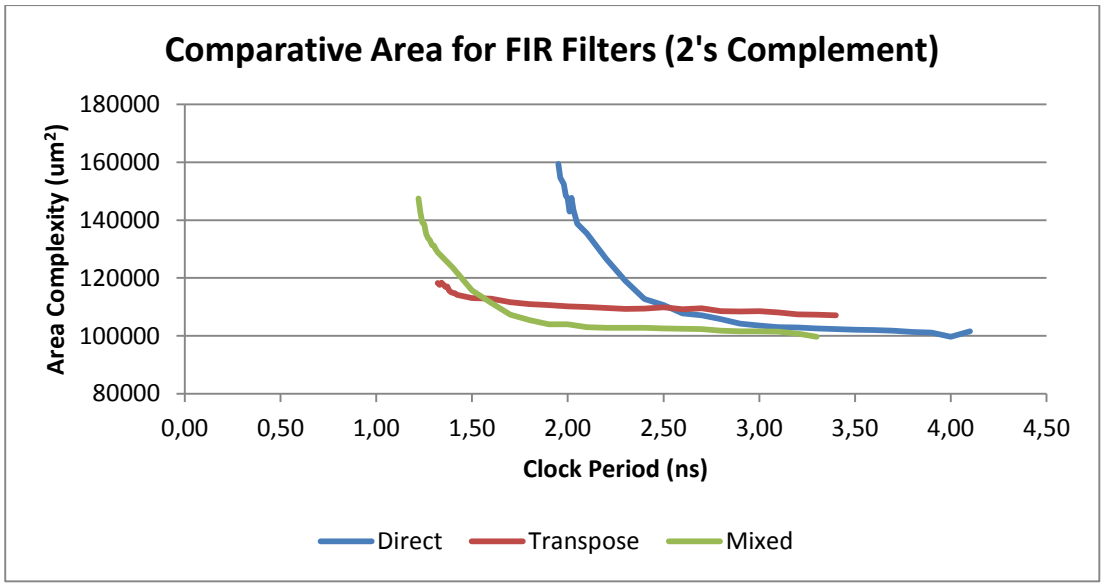
Για όλα τα κυκλώματα, λήφθηκαν μετρήσεις στην χαμηλότερη δυνατή συχνότητα ρολογιού που μπορούσε να προσομοιωθεί το εκάστοτε κύκλωμα. Η περίοδος του ρολογιού για αυτό το σκοπό μεταβαλλόταν με βήμα 0.01 ns για 20 επαναλήψεις. Στη συνέχεια πάρθηκαν 10 μετρήσεις από τον αμέσως μεγαλύτερο αριθμό κατά ένα δέκατο, της τελευταίας μέτρησης, με βήμα 0.1 ns για να εξεταστεί το κύκλωμα σε συνθήκες χαλαρότερων περιορισμών ρολογιού.

Τέλος, τα κυκλώματα συγκρίθηκαν ως προς την καθυστέρηση, την επιφάνεια και την ισχύ που καταναλώνουν.

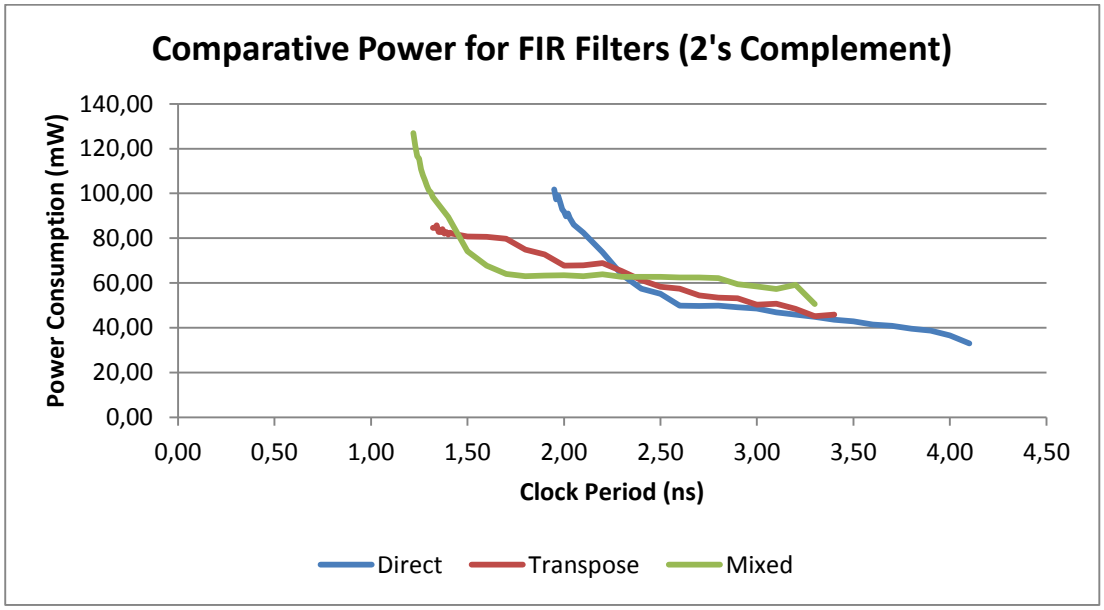
5.2 Πειραματική ανάλυση Area και Delay

5.2.1 Υλοποιήσεις φίλτρου FIR πλήρους ακρίβειας

i. Σύγκριση direct, transpose και mixed μορφών φίλτρου για συμβατική υλοποίηση με MB πολλαπλασιαστή



Σχήμα 5.1 *Σύγκριση επιφάνειας για FIR φίλτρα με MB πολλαπλασιαστή*

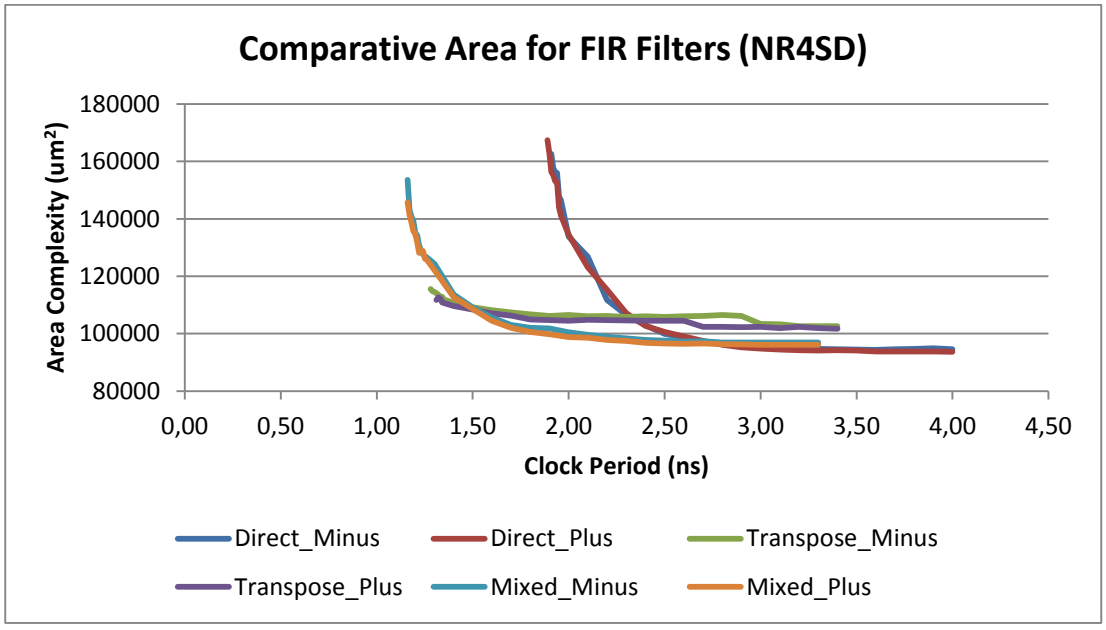


Σχήμα 5.2 *Σύγκριση κατανάλωσης για FIR φίλτρα με MB πολλαπλασιαστή*

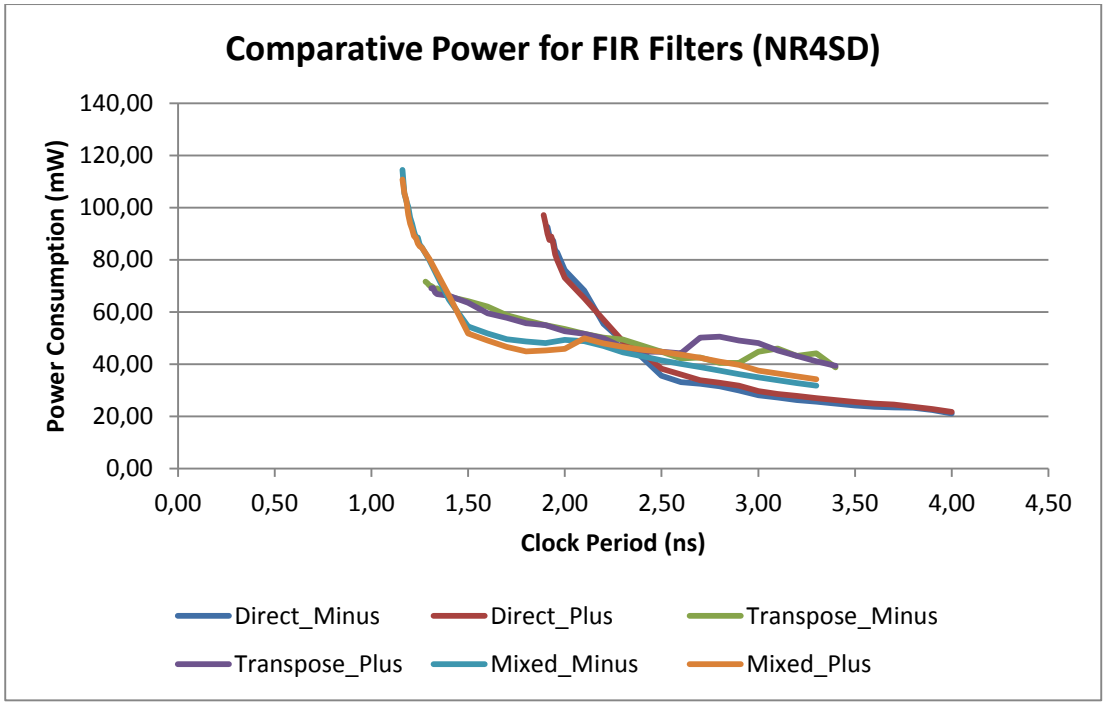
Παρατηρούμε ότι η direct υλοποίηση παρουσιάζει μεγαλύτερη κρίσιμη καθυστέρηση σε σχέση με τη transpose και τη mixed, καθώς ξεκινάει στο 1,95ns ενώ οι άλλες στο 1,32ns και 1,22ns αντίστοιχα. Στο Σχήμα 5.1, η transpose υλοποίηση ξεκινάει με μικρότερο area σε σχέση με τις άλλες δυο αλλά μετά τα 1,60ns η mixed υλοποίηση βελτιώνει την επιφάνεια που απαιτεί και τέλος από τα 2,60ns και έπειτα ακολουθεί και η direct την mixed. Στον Σχήμα 5.2 παρόλο που η mixed ξεκινάει με καλύτερες απαιτήσεις κατανάλωσης ισχύος από τα 2,40ns οι άλλες δύο υλοποιήσεις καταναλώνουν λιγότερο, με καλύτερη επιλογή την transpose εφόσον μας ενδιαφέρει και η κρίσιμη καθυστέρηση. Πιθανότατα να συμβαίνει αυτή η ιδιαιτερότητα στην καμπύλη της mixed γιατί έχει επέλθει κορεσμός και ο design compiler δεν μπορεί να βελτιστοποιήσει άλλο τη σύνθεση της.

Καλύτερο συνδυασμό κρίσιμης καθυστέρησης, επιφάνειας και κατανάλωσης παρουσιάζει η transpose υλοποίηση. Σε περίπτωση όμως που δεν μας ενδιαφέρει ο συνδυασμός και των τριών παραγόντων απόδοσης του κυκλώματος αλλά ένας ή δύο από αυτούς, μπορεί να επιλεχθεί κάποια από τις άλλες δυο υλοποιήσεις ανάλογα με τις απαιτήσεις.

ii. Σύγκριση direct, transpose και mixed μορφών φίλτρου για συμβατική υλοποίηση με NR4SD πολλαπλασιαστή



Σχήμα 5.3 *Σύγκριση επιφάνειας για FIR φίλτρα με NR4SD πολλαπλασιαστή*



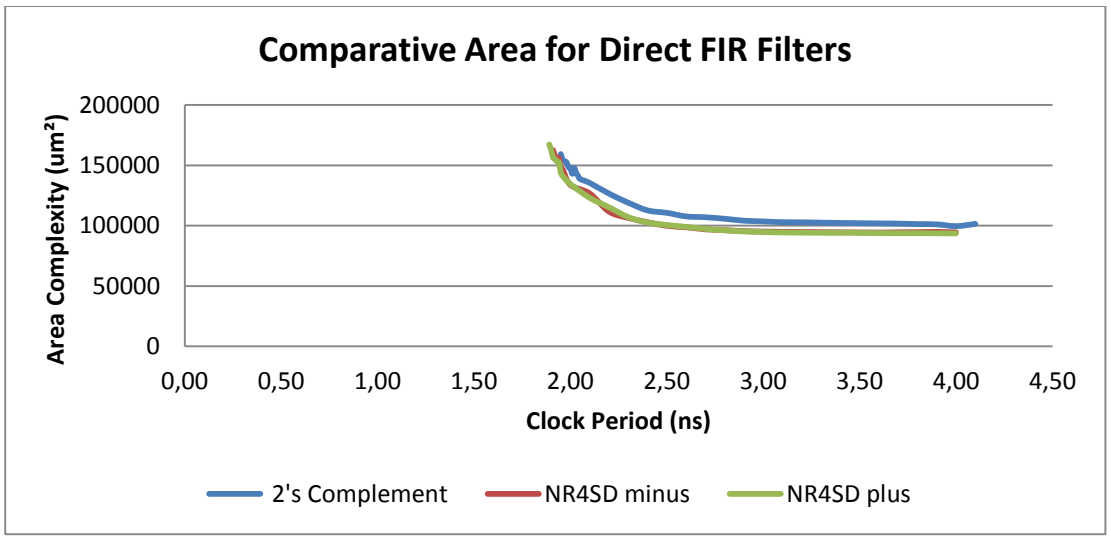
Σχήμα 5.4 *Σύγκριση κατανάλωσης για FIR φίλτρα με NR4SD πολλαπλασιαστή*

Παρατηρούμε ότι οι υλοποιήσεις NR4SD minus και plus σχεδόν ταυτίζονται για κάθε τοπολογία φίλτρου. Και σε αυτή τη περίπτωση η direct παρουσιάζει μεγαλύτερη κρίσιμη καθυστέρηση ξεκινώντας στα 1,90 ns ενώ η transpose στα 1,30ns και η mixed στα 1,16ns. Από άποψη επιφάνειας, η transpose ξεκινάει με χαμηλότερη απαίτηση area αλλά στη συνέχεια η mixed, η οποία παρουσιάζει ελάχιστη διαφορά στη κρίσιμη καθυστέρηση από την transpose, επιτυγχάνει χαμηλότερες τιμές area με τη direct να την ακολουθεί από τα 2,70ns και έπειτα. Από άποψη ισχύος, η transpose ξεκινάει με λιγότερη κατανάλωση, με την mixed να συμβαδίζει από τα 2,10ns και την direct να εμφανίζει την ελάχιστη κατανάλωση από τα 2,50ns και έπειτα. Το γεγονός ότι η mixed παρουσιάζει αυτή την ιδιομορφία στα διαγράμματα οφείλεται στο ότι αποτελεί συνδυασμό των άλλων δυο υλοποιήσεων και εμφανίζει στην ουσία ένα συνδυαστικό τους διάγραμμα.

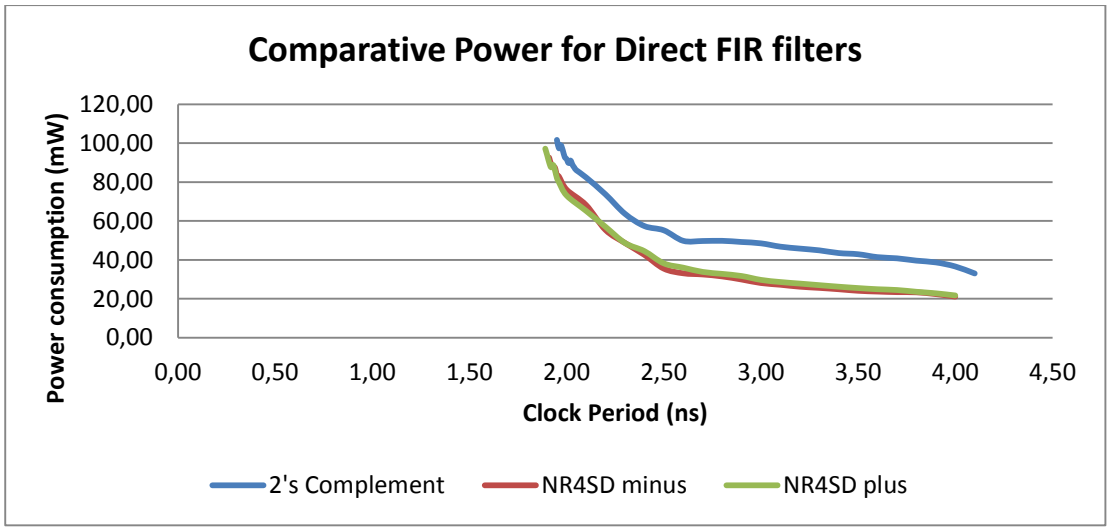
Καλύτερο συνδυασμό κρίσιμης καθυστέρησης, επιφάνειας και κατανάλωσης ισχύος παρουσιάζει η transpose μορφή. Σε περίπτωση που δεν μας ενδιαφέρει όμως, για παράδειγμα, η κρίσιμη καθυστέρηση αλλά κύριος παράγοντας είναι η χαμηλή κατανάλωση ισχύος μπορεί να επιλεγεί και η direct ή αν μας ενδιαφέρει κυρίως η κρίσιμη καθυστέρηση μπορεί να επιλεγεί η mixed.

iii. Σύγκριση υλοποιήσεων κάθε είδους φίλτρου με πολλαπλασιαστή κωδικοποίησης MB και πολλαπλασιαστή NR4SD κωδικοποίησης.

- Direct

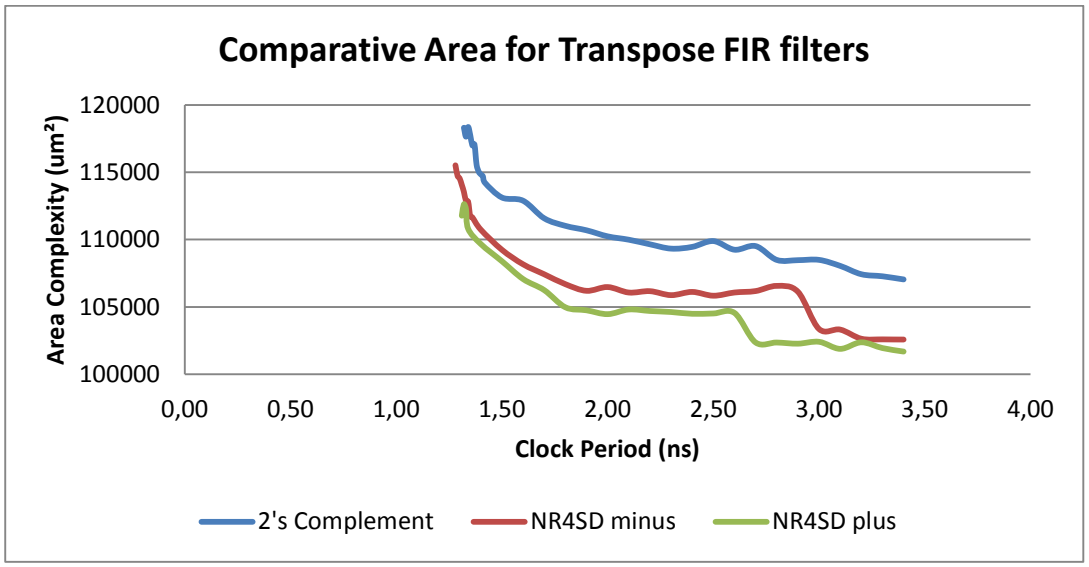


Σχήμα 5.5 *Σύγκριση επιφάνειας για direct FIR φίλτρα με MB και NR4SD πολλαπλασιαστή*

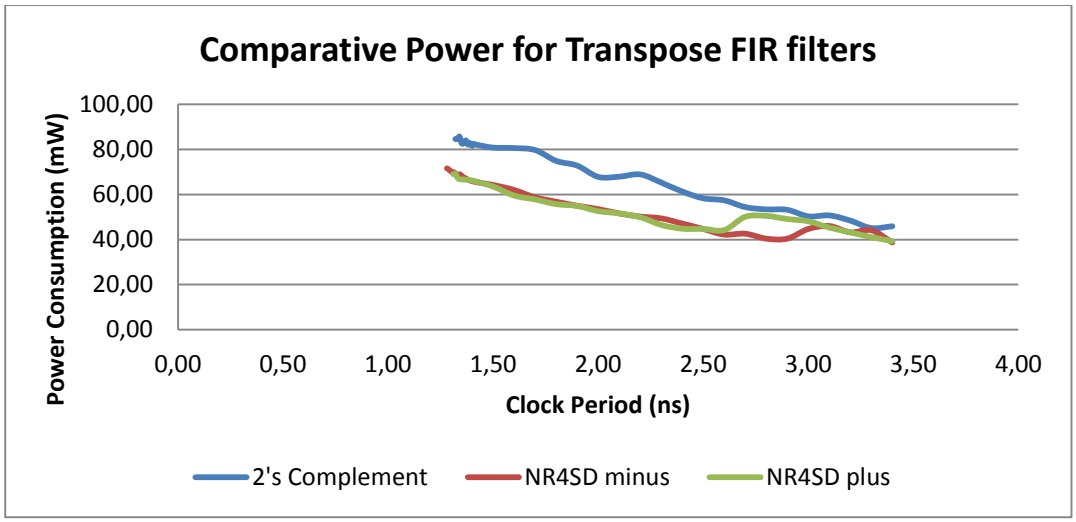


Σχήμα 5.6 *Σύγκριση κατανάλωσης για direct FIR φίλτρα με MB και NR4SD πολλαπλασιαστή*

- Transpose

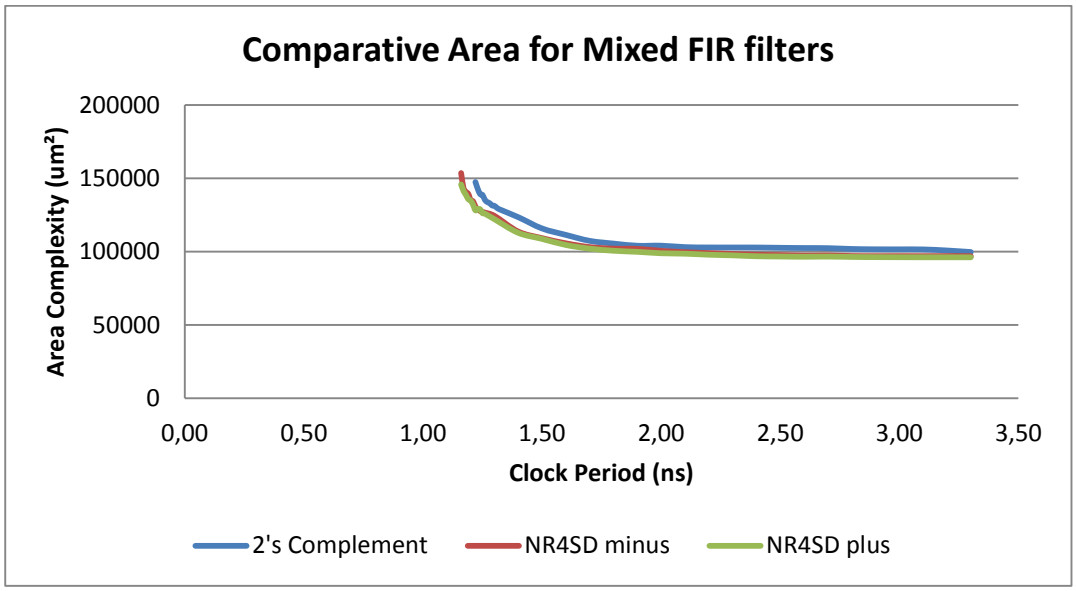


Σχήμα 5.7 *Σύγκριση επιφάνειας για transpose FIR φίλτρα με MB και NR4SD πολλαπλασιαστή*

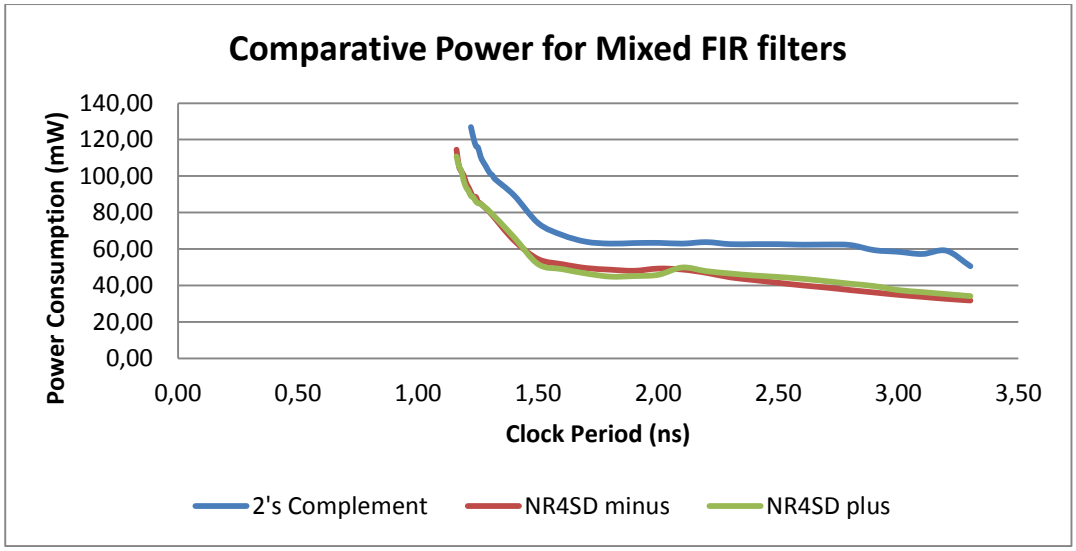


Σχήμα 5.8 *Σύγκριση κατανάλωσης για transpose FIR φίλτρα με MB και NR4SD πολλαπλασιαστή*

- Mixed



Σχήμα 5.9 *Σύγκριση επιφάνειας για mixed FIR φίλτρα με MB και NR4SD πολλαπλασιαστή*

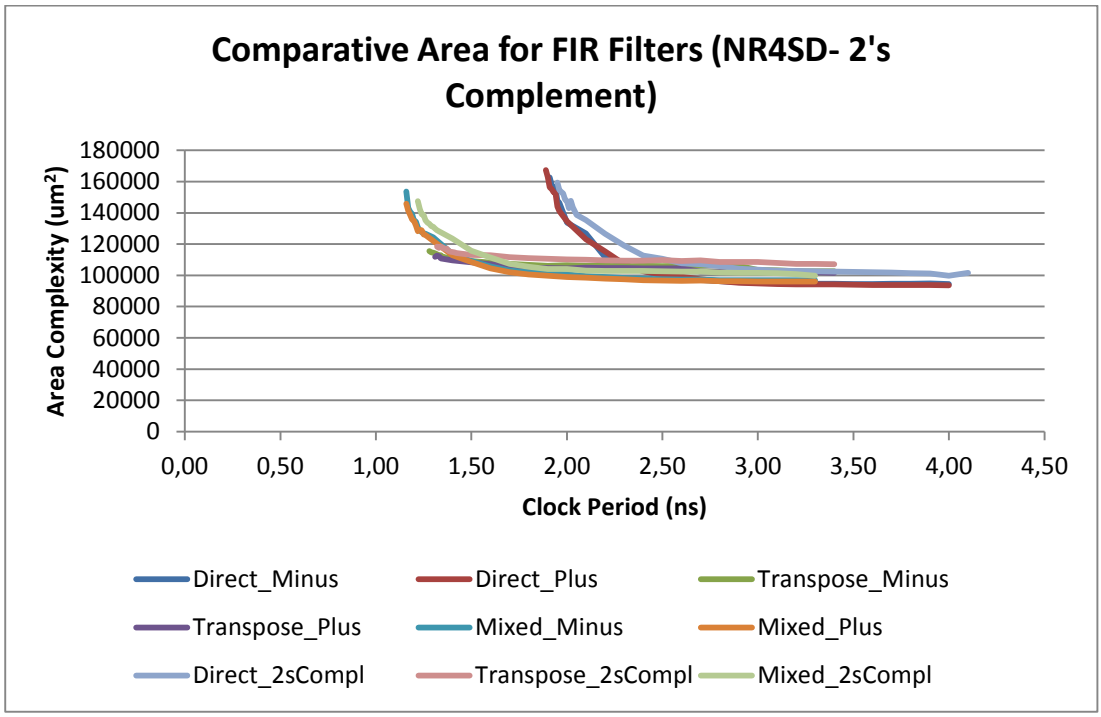


Σχήμα 5.10 *Σύγκριση κατανάλωσης για mixed FIR φίλτρα με MB και NR4SD πολλαπλασιαστή*

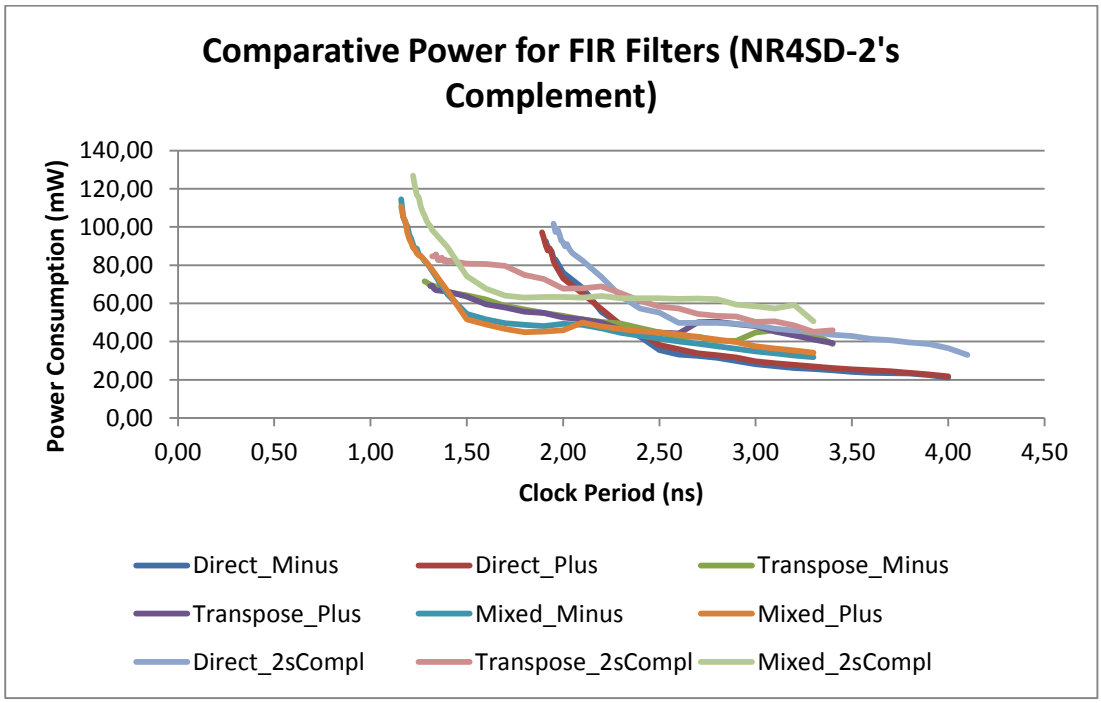
Τα Σχήματα 5.5 - 5.10 παρουσιάζουν τα διαγράμματα επιφάνειας και κατανάλωσης για τις τρεις υλοποιήσεις, κωδικοποίηση σε MB, NR4SD minus και NR4SD plus για κάθε μορφή φίλτρου –direct, transpose,mixed. Είναι εμφανές πως σε κάθε μορφή φίλτρου η υλοποίηση με τη προ-κωδικοποίηση NR4SD είναι καλύτερη σε σχέση με τη συμβατική σε MB, καθώς απαιτεί λιγότερη επιφάνεια και κατανάλωση ισχύος, γεγονός που οφείλεται στο ότι χρησιμοποιεί λιγότερο σύνθετα κυκλώματα μερικών γινομένων σε σχέση με τη συμβατική υλοποίηση σε MB.

iv. Σύγκριση MB και NR4SD κωδικοποίησης για όλες τις μορφές φίλτρων

Τέλος, παρουσιάζονται δύο τελικά σχήματα για επιφάνεια και κατανάλωση περιέχοντας όλες τις παραπάνω περιπτώσεις.



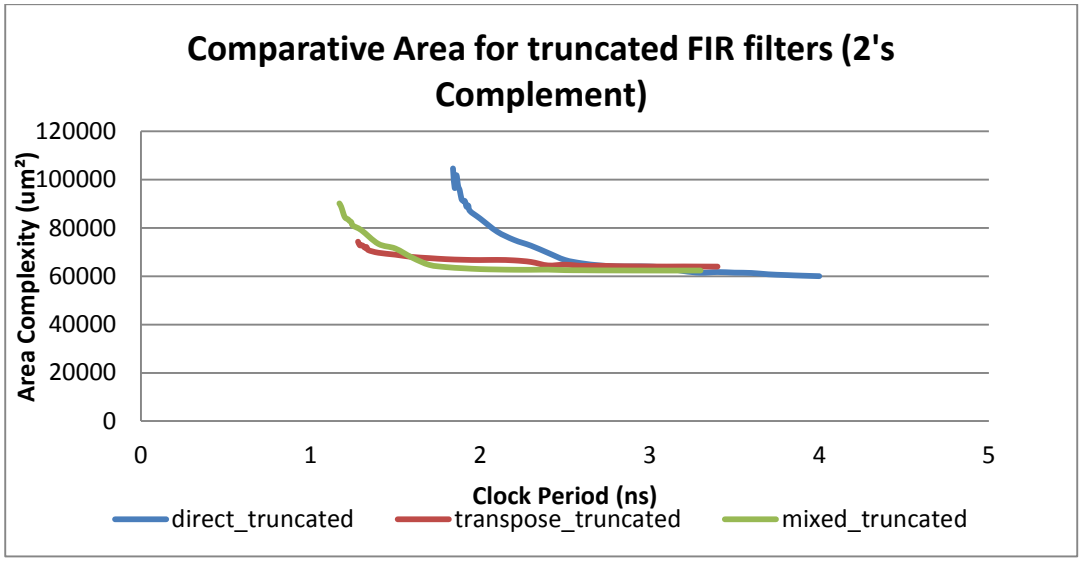
Σχήμα 5.11 *Σύγκριση επιφάνειας για FIR φίλτρα με MB και NR4SD πολλαπλασιαστή*



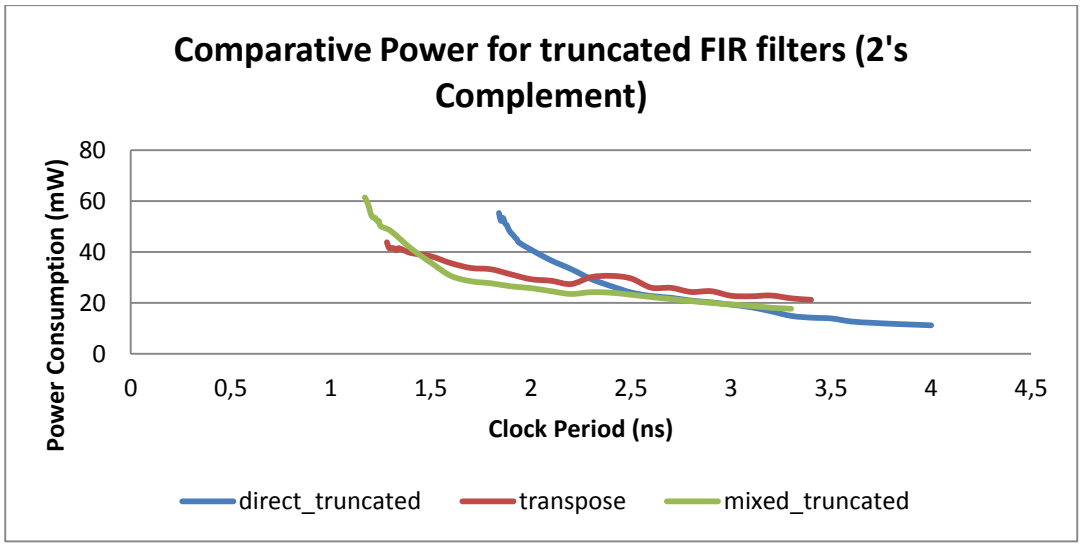
Σχήμα 5.12 *Σύγκριση κατανάλωσης για FIR φίλτρα με MB και NR4SD πολλαπλασιαστή*

5.2.2 Υλοποιήσεις φίλτρου FIR περικομμένης ακρίβειας

i. Σύγκριση direct, transpose και mixed μορφών φίλτρου για συμβατική υλοποίηση με truncated MB πολλαπλασιαστή



Σχήμα 5.13 *Σύγκριση επιφάνειας για FIR φίλτρα με MB πολλαπλασιαστή σταθερού μήκους*

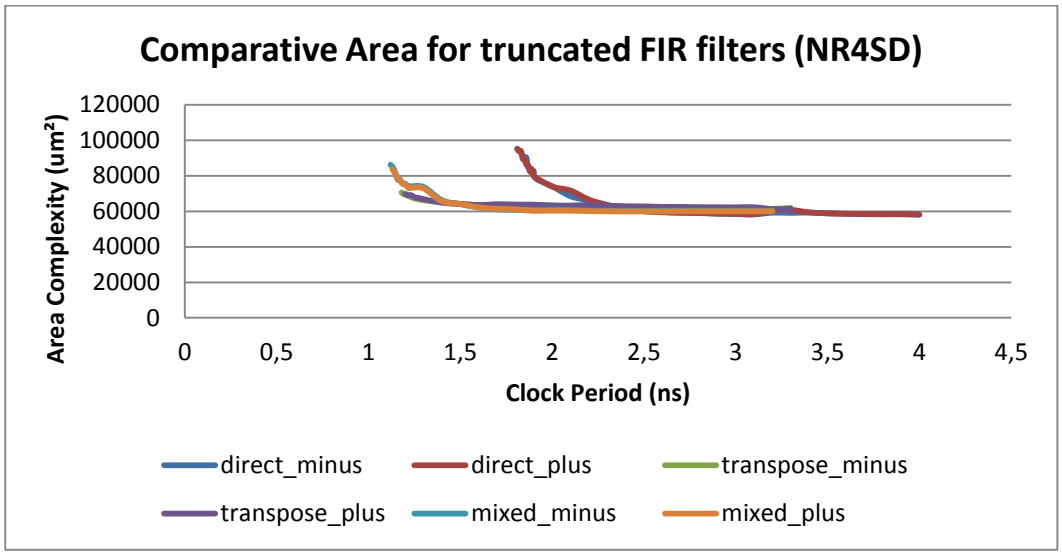


Σχήμα 5.14 *Σύγκριση κατανάλωσης για FIR φίλτρα με MB πολλαπλασιαστή σταθερού μήκους*

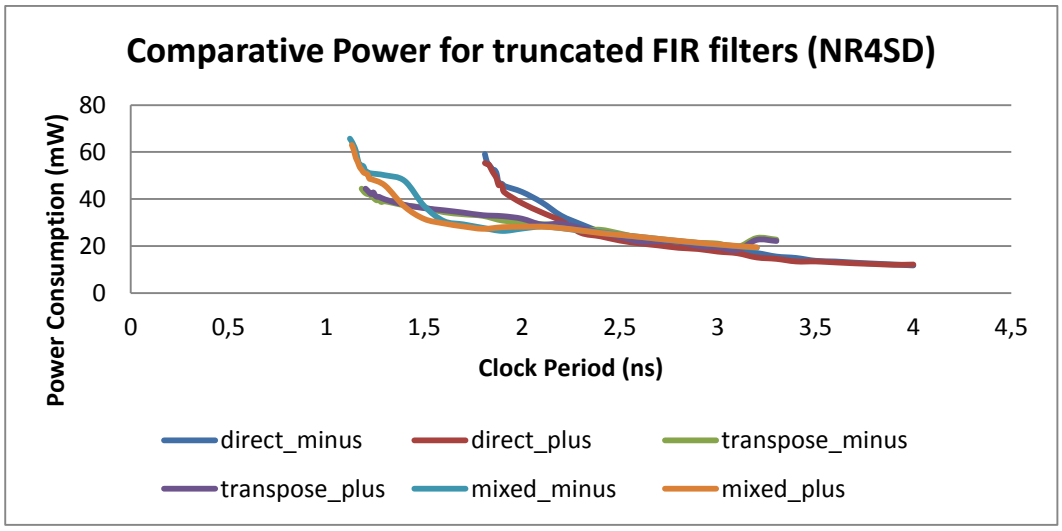
Στα Σχήματα 5.13 και 5.14 παρατηρούμε αρκετές ομοιότητες με τα αντίστοιχα Σχήματα 5.1 και 5.2 ως προς τις καμπύλες, με τη διαφορά ότι παρουσιάζονται κάποιες μικρές βελτιώσεις ως προς τη κρίσιμη καθυστέρηση κάθε τοπολογίας αλλά

και μεγάλη βελτίωση στην επιφάνεια και στη κατανάλωση ισχύος που απαιτεί κάθε υλοποίηση, γεγονός αναμενόμενο από τη στιγμή που έχει επέλθει περικοπή στο hardware και μειώθηκε σημαντικά ο αριθμός των μερικών γινομένων που υπολογίζονται. Βέβαια τα κέρδη αυτά στην απόδοση έχουν αρνητική συνέπεια τη μείωση της ακρίβειας. Οπότε σε περιπτώσεις όπου υπάρχει περιθώριο μείωσης της ακρίβειας, μπορεί να επιλεχθούν αυτές οι υλοποιήσεις για χαμηλότερες απαιτήσεις σε επιφάνεια και ισχύ.

ii. Σύγκριση direct, transpose και mixed μορφών φίλτρου για συμβατική υλοποίηση με truncated NR4SD πολλαπλασιαστή



Σχήμα 5.15 *Σύγκριση επιφάνειας για FIR φίλτρα με NR4SD πολλαπλασιαστή σταθερού μήκους*

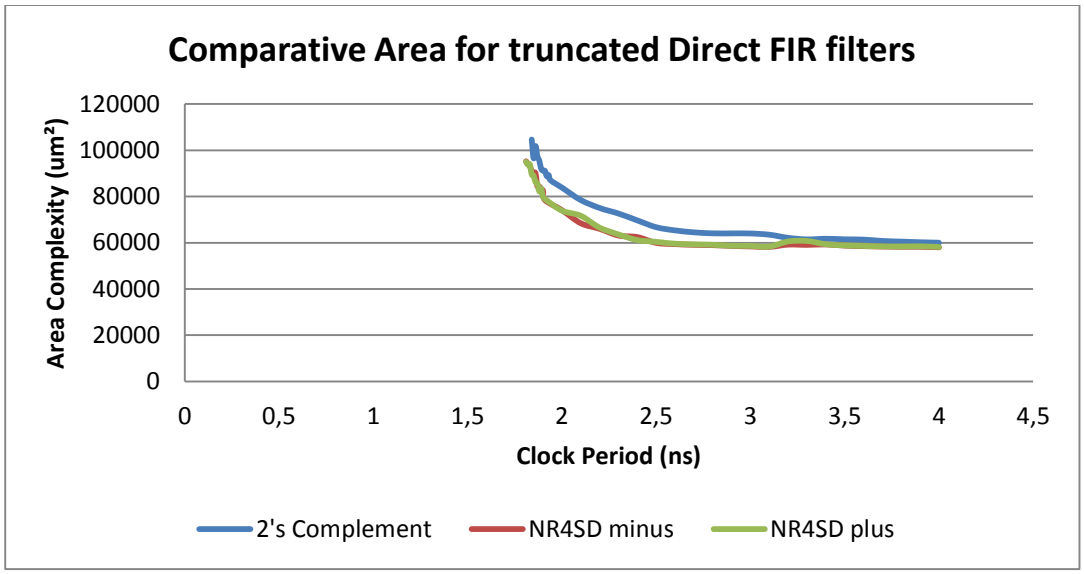


Σχήμα 5.16 *Σύγκριση κατανάλωσης για FIR φίλτρα με NR4SD πολλαπλασιαστή σταθερού μήκους*

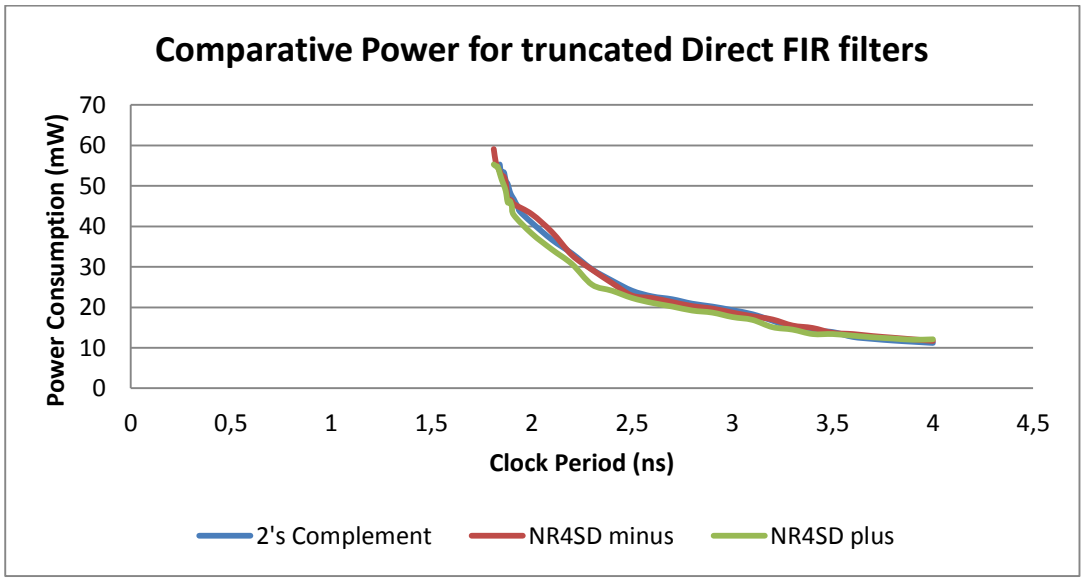
Αντίστοιχα με πριν, τα Σχήματα 5.15 και 5.16 παρουσιάζουν ομοιότητα στις καμπύλες με τα Σχήματα 5.3 και 5.4 με μικρή βελτίωση της κρίσιμης καθυστέρησης και μεγάλη βελτίωση στις απαιτήσεις επιφάνειας και κατανάλωσης ισχύος.

iii. Σύγκριση υλοποιήσεων κάθε είδους φίλτρου με πολλαπλασιαστή κωδικοποίησης MB και πολλαπλασιαστή NR4SD κωδικοποίησης.

- Direct

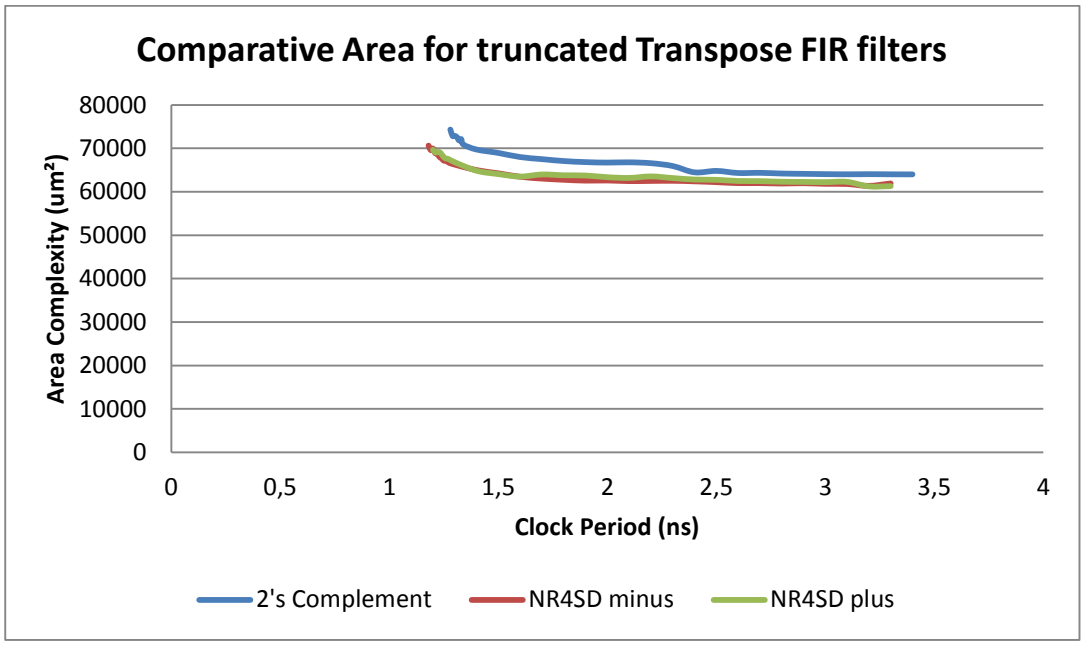


Σχήμα 5.17 *Σύγκριση επιφάνειας για direct FIR φίλτρα με MB και NR4SD πολλαπλασιαστή σταθερού μήκους*

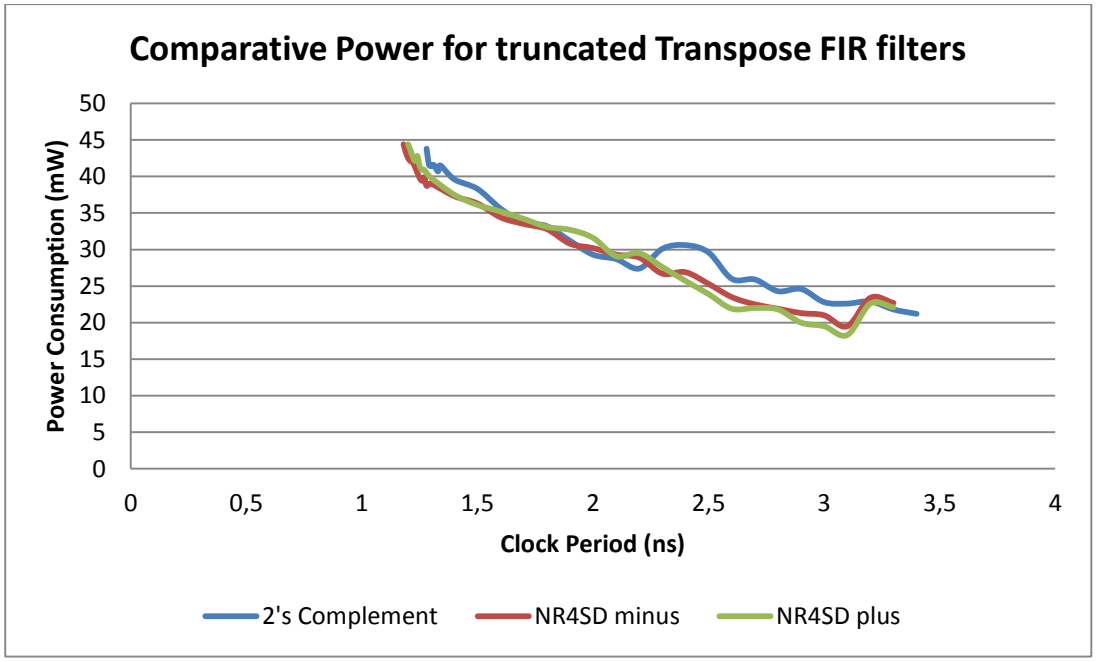


Σχήμα 5.18 *Σύγκριση κατανάλωσης για direct FIR φίλτρα με MB και NR4SD πολλαπλασιαστή σταθερού μήκους*

- Transpose

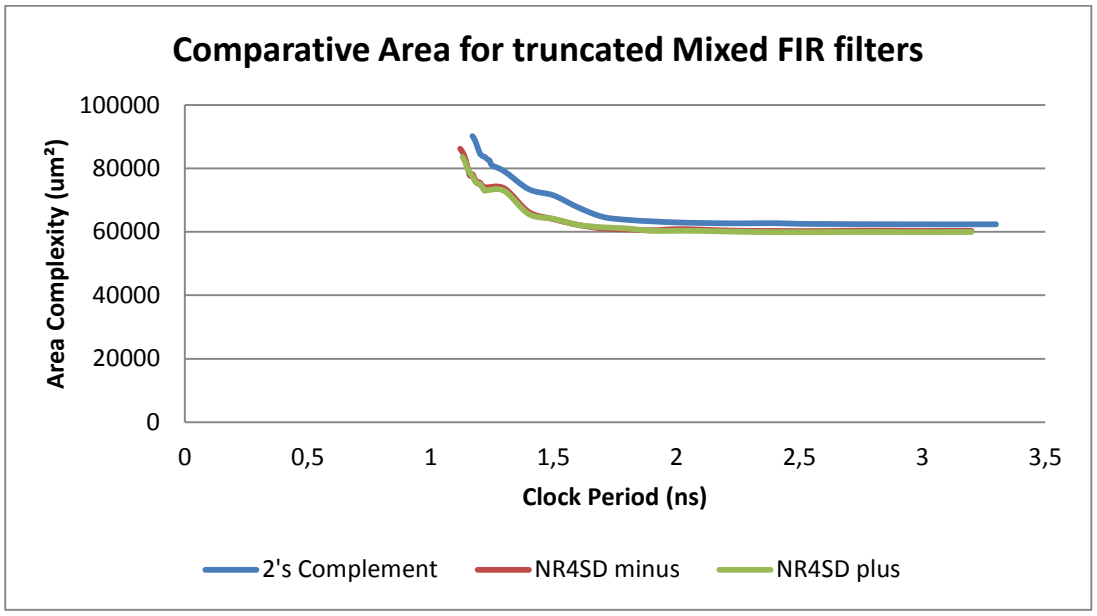


Σχήμα 5.19 *Σύγκριση επιφάνειας για transpose FIR φίλτρα με MB και NR4SD πολλαπλασιαστή σταθερού μήκους*

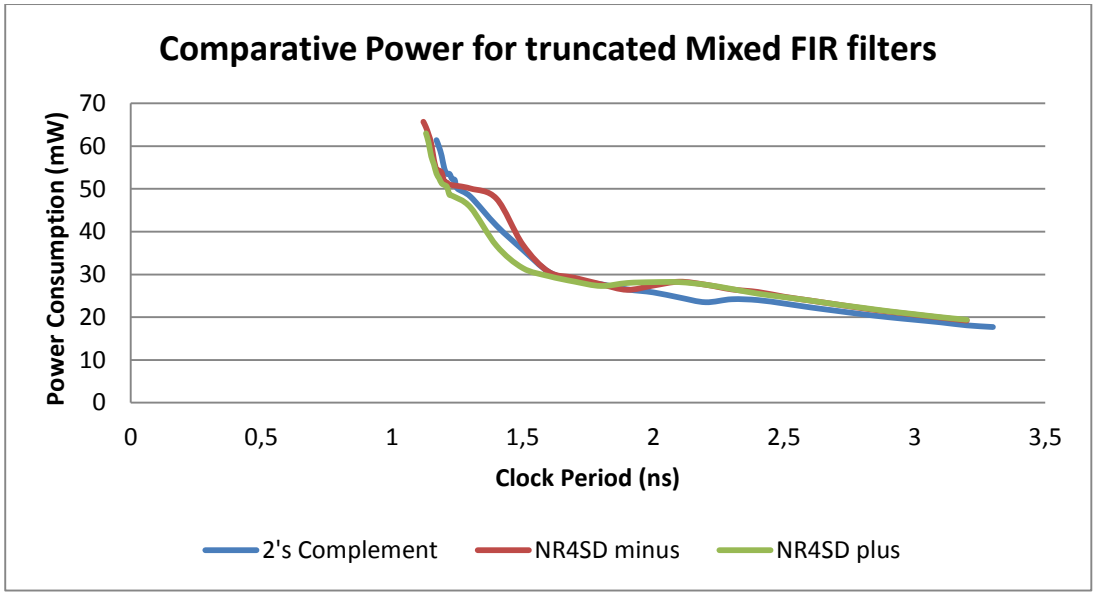


Σχήμα 5.20 *Σύγκριση κατανάλωσης για transpose FIR φίλτρα με MB και NR4SD πολλαπλασιαστή σταθερού μήκους*

- Mixed



Σχήμα 5.21 *Σύγκριση επιφάνειας για mixed FIR φίλτρα με MB και NR4SD πολλαπλασιαστή σταθερού μήκους*

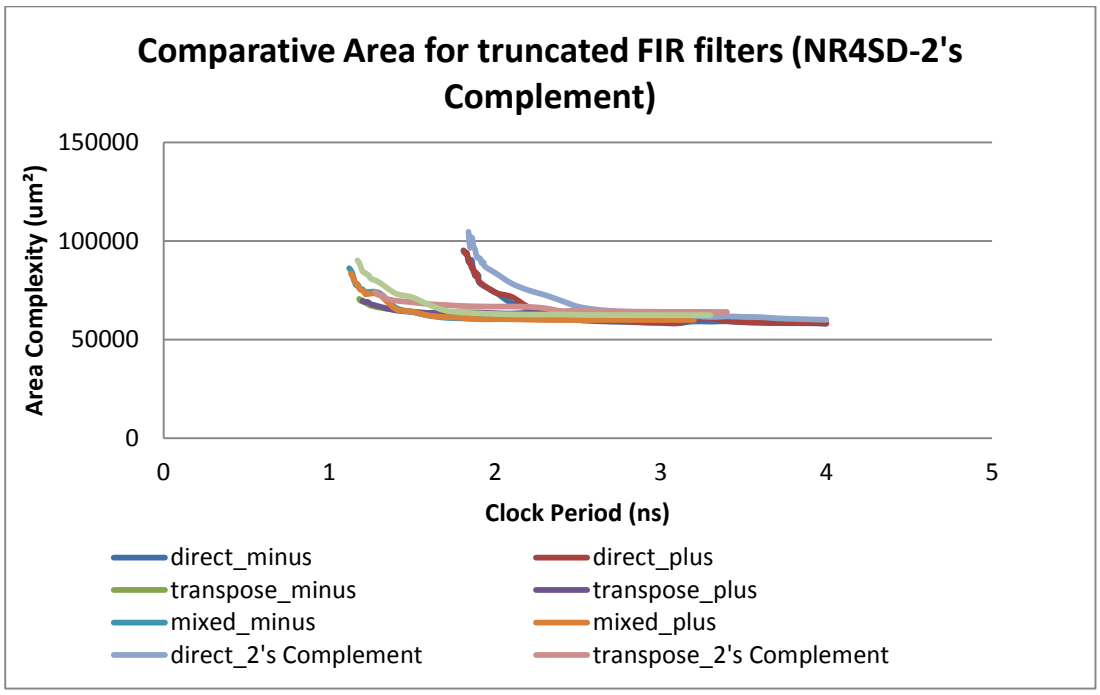


Σχήμα 5.22 *Σύγκριση κατανάλωσης για mixed FIR φίλτρα με MB και NR4SD πολλαπλασιαστή σταθερού μήκους*

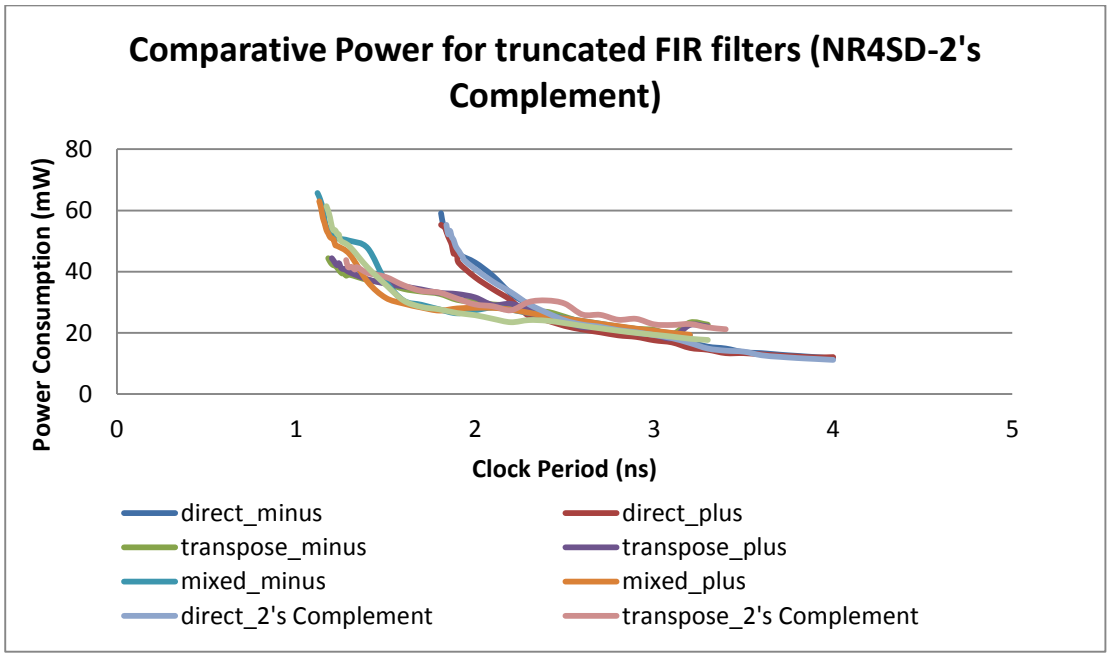
Όπως είναι εμφανές από τα παραπάνω διαγράμματα, οι υλοποιήσεις με NR4SD κωδικοποίηση παρουσιάζουν καλύτερη απόδοση σε σχέση με τη συμβατική με MB.

iv. Σύγκριση MB και NR4SD κωδικοποίησης για όλες τις μορφές φίλτρων

Τέλος, παρουσιάζονται δύο τελικά σχήματα για επιφάνεια και κατανάλωση περιέχοντας όλες τις παραπάνω περιπτώσεις.



Σχήμα 5.23 *Σύγκριση επιφάνειας για FIR φίλτρα με MB και NR4SD πολλαπλασιαστή σταθερού μήκους*

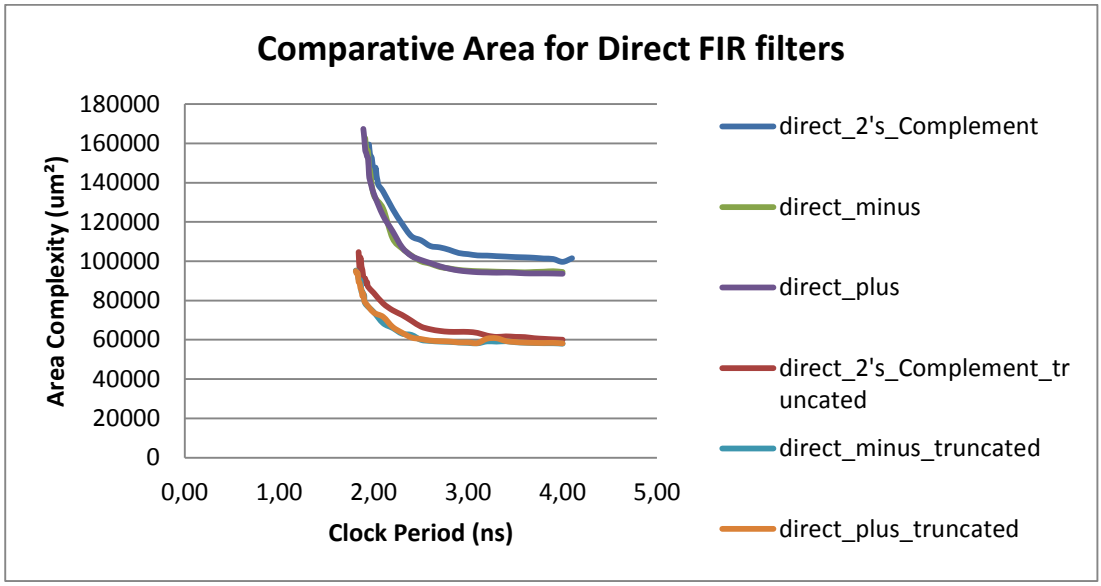


Σχήμα 5.24 *Σύγκριση κατανάλωσης για FIR φίλτρα με MB και NR4SD πολλαπλασιαστή σταθερού μήκους*

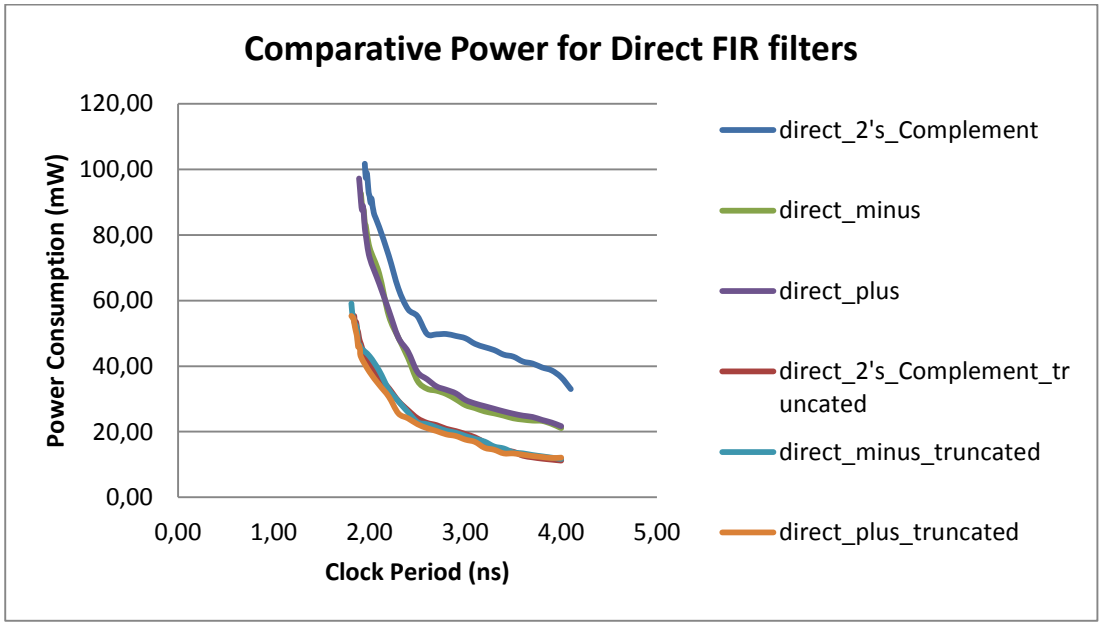
5.3 Σύγκριση φίλτρου FIR πλήρους και περικομμένης ακρίβειας

Σε αυτή την ενότητα παρουσιάζονται τα διαγράμματα για κάθε τοπολογία φίλτρου για τη περίπτωση κάθε πολλαπλασιαστή στην πλήρη και τη περικομμένη μορφή του.

- Direct

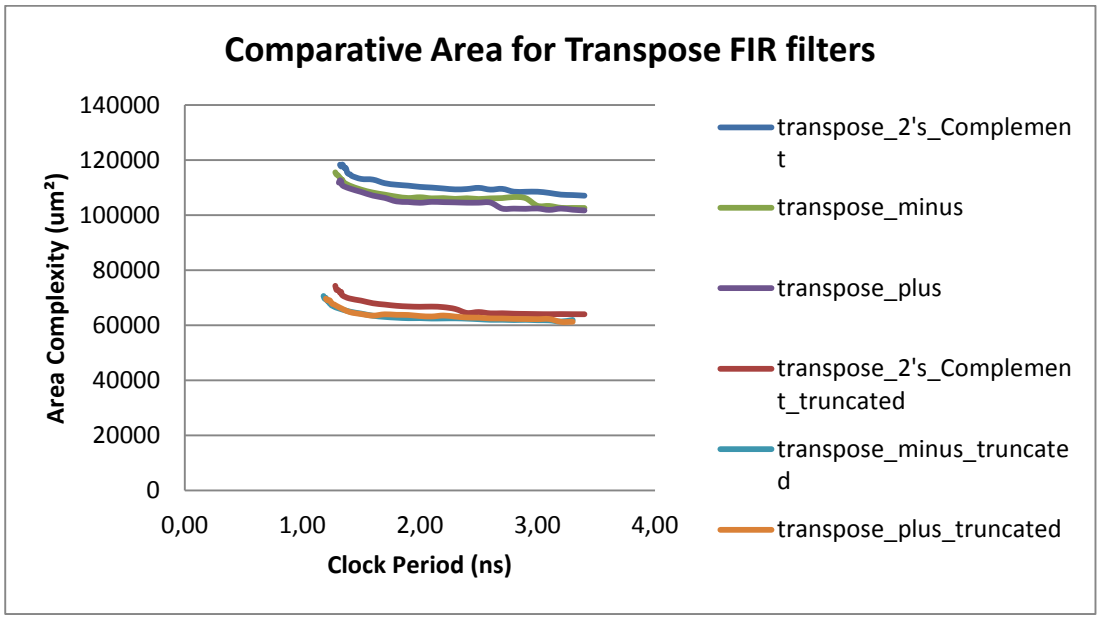


Σχήμα 5.25 *Σύγκριση επιφάνειας για direct FIR φίλτρα με MB και NR4SD πολλαπλασιαστή για πλήρη και περικομμένη ακρίβεια*

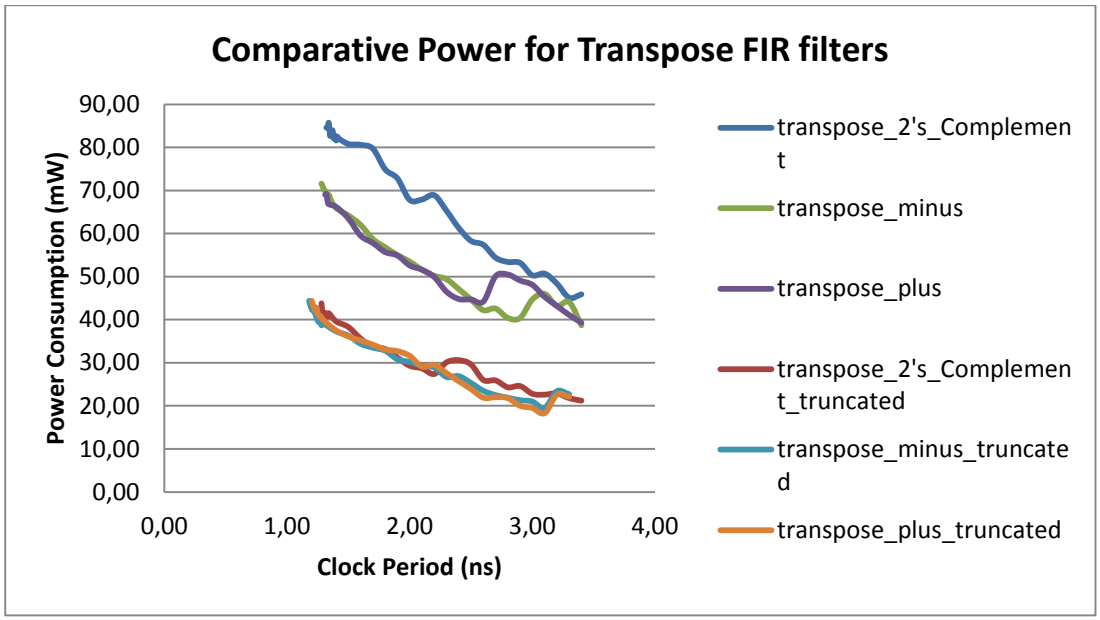


Σχήμα 5.26 *Σύγκριση κατανάλωσης για direct FIR φίλτρα με MB και NR4SD πολλαπλασιαστή για πλήρη και περικομμένη ακρίβεια*

- Transpose

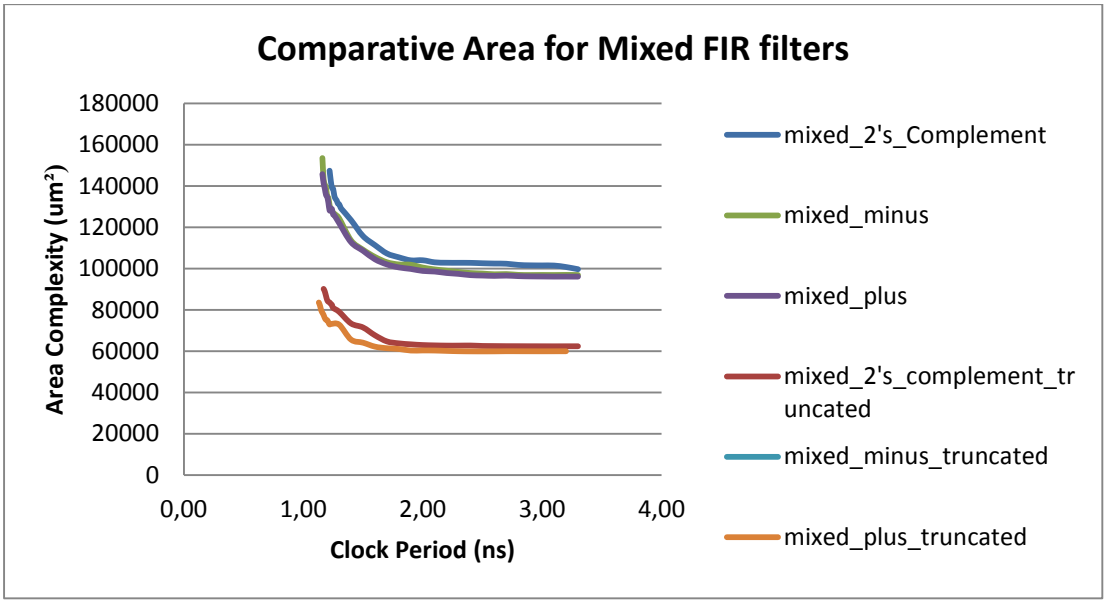


Σχήμα 5.27 *Σύγκριση επιφάνειας για transpose FIR φίλτρα με MB και NR4SD πολλαπλασιαστή για πλήρη και περικομμένη ακρίβεια*

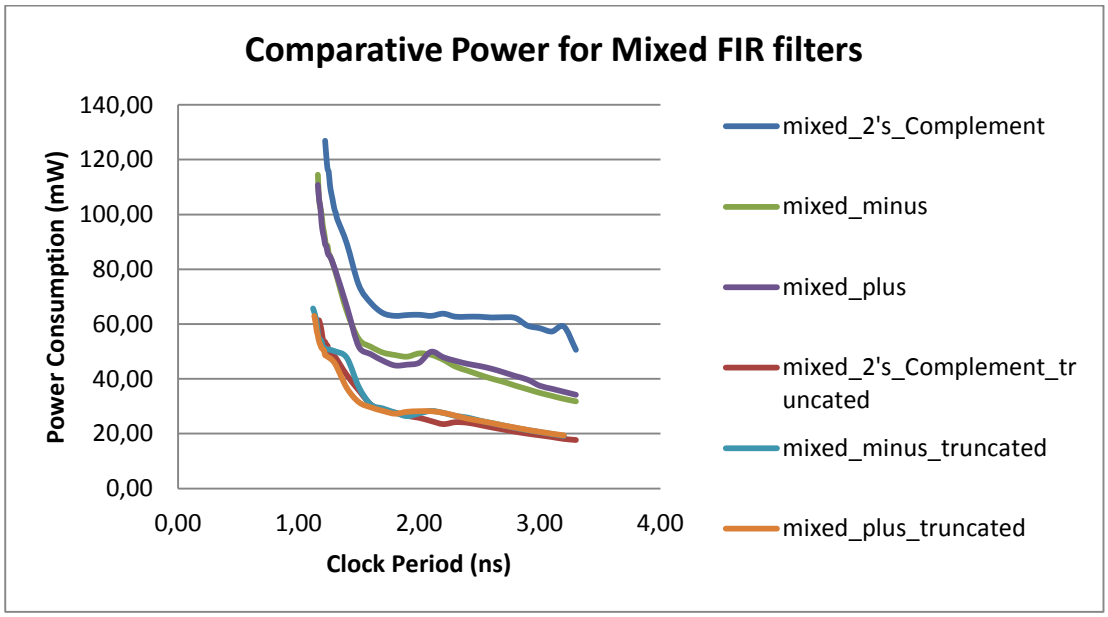


Σχήμα 5.28 *Σύγκριση κατανάλωσης για transpose FIR φίλτρα με MB και NR4SD πολλαπλασιαστή για πλήρη και περικομμένη ακρίβεια*

- Mixed



Σχήμα 5.29 *Σύγκριση επιφάνειας για mixed FIR φίλτρα με MB και NR4SD πολλαπλασιαστή για πλήρη και περικομμένη ακρίβεια*



Σχήμα 5.30 *Σύγκριση κατανάλωσης για mixed FIR φίλτρα με MB και NR4SD πολλαπλασιαστή για πλήρη και περικομμένη ακρίβεια*

Από τα Σχήματα 5.25 – 5.30 είναι ιδιαίτερα εμφανές ότι παρουσιάζεται σημαντική βελτίωση στην απόδοση του φίλτρου με τη χρησιμοποίηση της truncated μορφής, καθώς μειώνεται η επιφάνεια και κατανάλωση ισχύος που απαιτούνται.

5.4 Μέσο τετραγωνικό σφάλμα ακρίβειας

Σε αυτή την ενότητα υπολογίζεται το μέσο τετραγωνικό σφάλμα που προκύπτει στην ακρίβεια όταν γίνει περικοπή στο hardware σε σχέση με την περικοπή στο τελικό αποτέλεσμα του φίλτρου πλήρους ακρίβειας. Πιο συγκεκριμένα, υπολογίζεται το μέσο τετραγωνικό σφάλμα για 20480 αποτελέσματα που προκύπτουν από τον πολλαπλασιαστή σταθερού μήκους με MB προ-κωδικοποίηση σε σχέση με τα αποτελέσματα που προκύπτουν από τον συμβατικό πολλαπλασιαστή με MB προ-κωδικοποίηση, από τα οποία έχουν διατηρηθεί μόνο τα 20 πιο σημαντικά (MSB) bits. Υπολογίζεται δηλαδή η εξής σχέση:

$$\sigma = \frac{\sum_{i=0}^{20480}(y - y_i)^2}{\sum_{i=0}^{20480} y^2}$$

,όπου y είναι τα αποτελέσματα που προκύπτουν με περικοπή από τα αποτελέσματα της πλήρους ακρίβειας ενώ y_i είναι τα αποτελέσματα της περικοπής του hardware. Αντικαθιστώντας για τις τιμές που εξήχθησαν από την προσομοίωση, προκύπτει η εξής τιμή για μέσο τετραγωνικό σφάλμα:

$$\sigma=2.189 \times 10^{-7}$$

Η τιμή του σφάλματος είναι ιδιαίτερα μικρή, γεγονός που σημαίνει ότι οι δύο περιπτώσεις αποτελεσμάτων συγκλίνουν σε μεγάλο βαθμό και συνεπώς δεν χάνεται σημαντικό ποσοστό ακρίβειας σε περίπτωση χρησιμοποίησης των truncated υλοποιήσεων για μείωση της καθυστέρησης, της επιφάνειας και της κατανάλωσης του κυκλώματος.

5.5 Συμπεράσματα

Με βάση τα αποτελέσματα των προσομοιώσεων που παρουσιάστηκαν στις ενότητες 5.2 - 5.4 για υλοποιήσεις με πλήρη και περικομμένη ακρίβεια αντίστοιχα, καταλήγουμε στο συμπέρασμα ότι οι τοπολογίες που χρησιμοποιούσαν τον προ-κωδικοποιημένο NR4SD πολλαπλασιαστή εμφανίζουν καλύτερες αποδόσεις σε κρίσιμη καθυστέρηση, επιφάνεια και κατανάλωση ισχύος σε σχέση με τις αντίστοιχες τοπολογίες με τον συμβατικό MB πολλαπλασιαστή. Επίσης, από τις τρεις μορφές υλοποίησης φίλτρου –direct, transpose, mixed- η transpose εμφανίζει καλύτερη απόδοση συνδυάζοντας χαμηλότερες τιμές για κρίσιμη καθυστέρηση, επιφάνεια και κατανάλωση. Η επιλογή βέβαια της κατάλληλης μορφής φίλτρου εξαρτάται από τις απαιτήσεις του κατασκευαστή, συνεπώς ανάλογα με τη παράμετρο απόδοσης που δίνεται βάρος επιλέγεται η αντίστοιχη μορφή φίλτρου με βάση τα σχήματα που παρουσιάστηκαν παραπάνω. Τέλος, παρατηρούμε ότι οι υλοποιήσεις με πολλαπλασιαστή σταθερού μήκους παρουσιάζουν σημαντική βελτίωση της απόδοσης με απώλεια ένα ποσοστό της ακρίβειας των αποτελεσμάτων.

6. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Kiamal Pekmestzi (2003). Digital VLSI systems. NTUA Lectures Notes.

[2] Tsoumanis, K., Axelos, N., Moschopoulos, N., Zervakis, G. and Pekmestzi, K. (2015). Pre-Encoded Multipliers Based on Non-Redundant Radix-4 Signed-Digit Encoding. *IEEE Trans. Comput.*, pp.1-1.

[3] Kyung-Ju Cho, Kwang-Chul Lee, Jin-Gyun Chung, and Parhi, K. (2004). Design of low-error fixed-width modified booth multiplier. *IEEE Trans. VLSI Syst.*, 12(5), pp.522-531.

[4] Synopsys Design Compiler, www.synopsys.com.

[5] ModelSim Corporation, www.model.com.

[6] Synopsys Primetime PX, www.synopsys.som.

[7] DW01_add , www.synopsys.com/dw/ipdir.php?c=DW01_add.

[8] DW02_tree, www.synopsys.com/dw/ipdir.php?c=DW02_tree.