Τ.Ε.Ι. Δυτικής Ελλάδας
Τμήμα Μηχανικών Πληροφορικής Τ.Ε

Σχεδιασμός, ανάπτυξη και υλοποίηση τηλεπικοινωνιακών VHDL IP Cores σε FPGA

Πτυχιακή Εργασία

Σουμπερνίτσκαγια Χριστίνα (Α.Μ: 1097)

Επιβλέπων καθηγητής
Νικόλαος Βώρος

Αντίριο, 2014

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή μου, κύριο Νικόλαο Βώρο και στο Διευθύνοντα Σύμβουλο και Νόμιμο Εκπρόσωπο της εταιρείας Noesis Technologies κύριο Γιώργο Κρίκη , οι οποίοι με τη μεθοδική τους καθοδήγηση, τις συμβουλές, τις παρατηρήσεις και τις εξειδικευμένες επιστημονικές τους γνώσεις συνέβαλαν καθοριστικά στην εκπόνηση της παρούσας πτυχιακής εργασίας, που δε θα ήταν δυνατή χωρίς την πολύτιμη βοήθειά τους. Ακόμη, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στην οικογένεια μου για την έμπρακτη ηθική και υλική τους υποστήριξη καθ' όλη τη διάρκεια των σπουδών μου.

Πρόλογος

Τα σύγχρονα τηλεπικοινωνιακά συστήματα έχουν υιοθετήσει κώδικες διόρθωσης λαθών με στόχο να αυξήσουν την αξιοπιστία των συστημάτων κατά τη μετάδοση πληροφορίας. Οι Κώδικες- Χαμηλής- Πυκνότητας-Ελέγχου-Ισοτιμίας (Low-Density-Parity-Check codes) ανήκουν στην κατηγορία των γραμμικών block κωδίκων που διαθέτουν εξαιρετικές επιδόσεις. Οι LDPC κώδικες προσεγγίζουν αρκετά τη μέγιστη χωρητικότητα του διαύλου (όριο του Shannon).

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι να σχεδιαστεί ένας LDPC κωδικοποιητής ο οποίος μπορεί να πετύχει ρυθμό απόδοσης (Throughput rate) τουλάχιστον 1 Gbps. Αρχικά μελετούνται τα ιδιαίτερα χαρακτηριστικά και οι παράμετροι των κωδίκων αυτών. Στη συνέχεια παρουσιάζονται οι πιο δημοφιλείς αλγόριθμοι κωδικοποίησης και αποκωδικοποίησης των LDPC κωδίκων, με ιδιαίτερη έμφαση στον αλγόριθμο κωδικοποίησης R&U . Τέλος υλοποιείται μια προτεινόμενη αρχιτεκτονική, η οποία ικανοποίει τον αρχικό σκοπό.

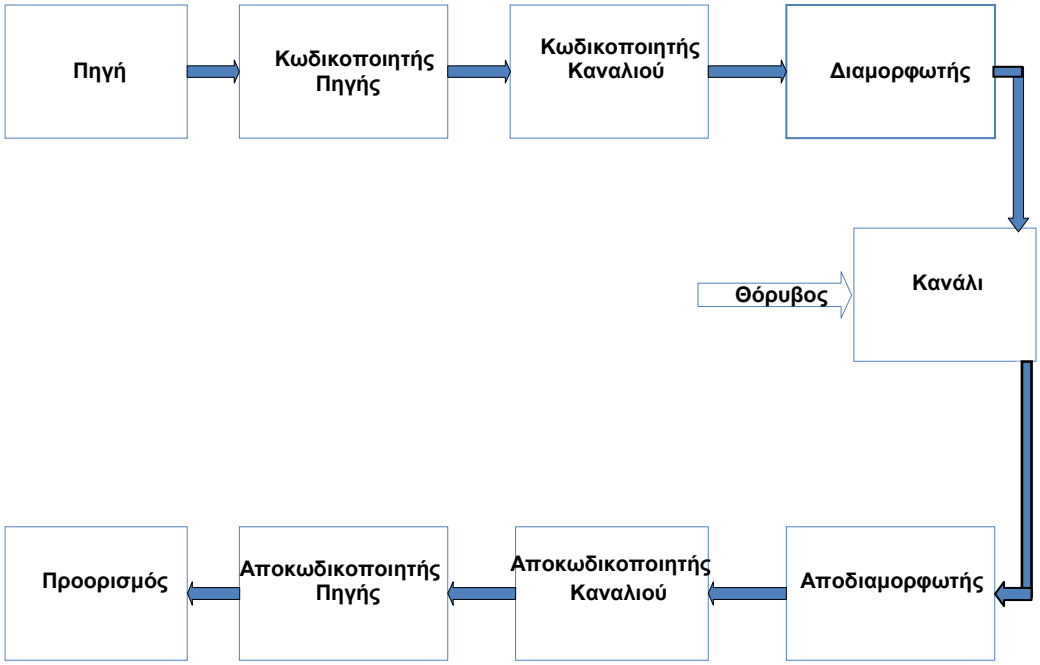
Περιεχόμενα

Κατάλογος σχημάτων

ΚΕΦΑΛΑΙΟ 1
Εισαγωγή

1.1 Βασικά μέρη ενός τηλεπικοινωνιακού συστήματος

Η πληροφορία στη φύση παρουσιάζεται αποκλειστικά σε αναλογική μορφή, καθώς ο άνθρωπος μπορεί να αντιληφθεί μόνο αναλογικά σήματα. Έτσι λοιπόν, τα αναλογικά δεδομένα πρέπει να μετατραπούν σε ακολουθίες από 0 και 1, ώστε ο δέκτης να μην είναι υποχρεωμένος να κάνει μια εκτίμηση των άπειρων τιμών ενός αναλογικού σήματος, αλλά απλά να πάρει μια απόφαση μεταξύ των δυο διακριτών τιμών για κάθε σήμα, 0 ή 1. Η διαδικασία αυτή πραγματοποιείται μέσω ενός τηλεπικοινωνιακού συστήματος.



Σχήμα 1.1: Τα βασικά στοιχεία ενός τηλεπικοινωνιακού συστήματος

- Η *πηγή πληροφορίας* παράγει την πηγαία πληροφορία που πρόκειται να μεταδοθεί μέσω του καναλιού. Η παραγόμενη πληροφορία , στην έξοδο της πηγής , μπορεί να είναι είτε σε ψηφιακή μορφή (σε μορφή διακριτών συμβόλων) είτε σε αναλογική μορφή (συνεχής κυματομορφή).
- Ο *κωδικοποιητής πηγής* μετατρέπει την έξοδο της ψηφιακής πηγής πληροφορίας σε ακολουθία από δυαδικά ψηφία η οποία ονομάζεται ακολουθία πληροφορίας. Στην περίπτωση που η πηγή πληροφορίας είναι αναλογική , ο κωδικοποιητής πηγής περιλαμβάνει και μετατροπή αναλογικού σήματος σε ψηφιακό.
- Το κανάλι προσθέτει *θόρυβο* στην μεταδιδόμενη πληροφορία με αποτέλεσμα να μειώνεται η αξιοπιστία του συστήματος. Προκειμένου να βελτιωθεί η αξιοπιστία του συστήματος και να προστατευθεί αυτό από το θόρυβο του καναλιού, προστίθεται στην προς μετάδοση πληροφορία επιπλέον (πλεονάζουσα) πληροφορία. Για την ευθύνη της προσθήκης της πληροφορίας είναι υπεύθυνος ο *κωδικοποιητής καναλιού*.

- Τα ψηφιακά δεδομένα δεν είναι κατάλληλα για μετάδοση μέσω ενός καναλιού. Η ακολουθία ψηφιακών δεδομένων που εξέρχεται από τον κωδικοποιητή καναλιού, θα πρέπει να μετατραπεί σε μία συνεχή κυματομορφή. Τα χαρακτηριστικά της απαιτούμενης κυματομορφής καθορίζονται από το είδος του καναλιού, το παρεχόμενο εύρος ζώνης και τα μεταδιδόμενα σύμβολα. Η διαδικασία της παραγωγής αυτής της κυματομορφής αποκαλείται διαμόρφωση του σήματος. Ο *διαμορφωτής* αναλαμβάνει τη διεκπεραίωση της διαμόρφωσης και παραδίδει, εν συνεχεία, στο κανάλι επικοινωνίας την προκύπτουσα κυματομορφή.
- Ο *αποδιαμορφωτής* λαμβάνει στην είσοδό του την έξοδο του καναλιού, μία κυματομορφή, δηλαδή, αντίστοιχη εκείνης την οποία παράγει ο διαμορφωτής στον πομπό. Την κυματομορφή αυτή την επεξεργάζεται και παράγει μία ακολουθία εξόδου που αποκαλείται ληφθείσα ακολουθία και ενδέχεται να είναι διακριτή ή συνεχής.
- *Ο αποκωδικοποιητή καναλιού* αναλαμβάνει να αφαιρέσει την πλεονάζουσα πληροφορία την οποία εισήγαγε ο κωδικοποιητής καναλιού και να μετατρέψει την ληφθείσα ακολουθία σε δυαδική, αξιοποιώντας την πλεονάζουσα πληροφορία. Η διαδικασία αυτή ονομάζεται αποκωδικοποίηση καναλιού. Η στρατηγική αποκωδικοποίησης βασίζεται στους κανόνες της κωδικοποίησης καναλιού καθώς και στα χαρακτηριστικά του θορύβου που προσθέτει το κανάλι.
- Η δυαδική ακολουθία που εξέρχεται από τον αποκωδικοποιητή καναλιού αποτελεί την είσοδο του *αποκωδικοποιητή πηγής*, ο οποίος, γνωρίζοντας τη μέθοδο που χρησιμοποιείται για την κωδικοποίηση της πηγής, προσπαθεί να ανακατασκευάσει όσο γίνεται πιστότερα το αρχικό αναλογικό σήμα της πηγής.

1.2 Κωδικοποίηση και βασικές έννοιες

Σύμφωνα με το μοντέλο του τηλεπικοινωνιακού συστήματος, ο όρος «κωδικοποίηση» αναφέρεται ουσιαστικά στα πρώτα τρία στάδια που εκτελεί ένα τηλεπικοινωνιακό σύστημα :

- Η κωδικοποίηση πηγής, σύμφωνα με την οποία η πληροφορία, που βρίσκεται σε αναλογική μορφή, υφίσταται συμπίεση ώστε να μετατραπεί σε μορφή κατάλληλη για μετάδοση, με τη μικρότερη δυνατή σπατάλη εύρους ζώνης και ισχύος
- Η κωδικοποίηση διαύλου, κατά την οποία προστίθενται πλεονάζοντα δυαδικά ψηφία της πληροφορίας με σκοπό να γίνει δυνατή η ανίχνευση και διόρθωση σφαλμάτων
- Η διαδικασία της διαμόρφωσης, με την οποία η προς μετάδοση, κωδικοποιημένη πλέον, δυαδική πληροφορία μετατρέπεται σε μορφή η οποία συμφωνεί με τα χαρακτηριστικά του διαύλου ώστε να μεταδοθεί διαμέσου αυτού.

Η σημασία της αρχής του ρυθμού πληροφορίας είναι ότι μας εισάγει σε ένα θεώρημα που οφείλεται στον Shannon και είναι θεμελιώδες στη θεωρία επικοινωνιών. Αυτό το θεώρημα αφορά στο ρυθμό μετάδοσης πληροφορίας μέσω καναλιού επικοινωνιών. Το θεώρημα του Shannon λέει ότι είναι δυνατό να επινοηθεί ένα μέσο δια του οποίου ένα σύστημα επικοινωνιών θα μεταδίδει πληροφορία με μια αυθαίρετα μικρή πιθανότητα σφάλματος δεδομένου ότι ο ρυθμός πληροφορίας R είναι μικρότερος ή ίσος με έναν ρυθμό C ο οποίος καλείται χωρητικότητα καναλιού. Η τεχνική που χρησιμοποιείται για να προσεγγιστεί αυτό το όριο καλείται κωδικοποίηση. Έτσι λοιπόν σύμφωνα με το θεώρημα, έστω ότι υπάρχει μια πηγή M ισοπίθανων μηνυμάτων, με $M > 1$, η οποία παράγει πληροφορία με ρυθμό R. Δίνεται κανάλι χωρητικότητας C, τότε αν $R \leq C$, υπάρχει μια τεχνική κωδικοποίησης τέτοια ώστε η έξοδος της πηγής να μπορεί να μεταδοθεί μέσω καναλιού με πιθανότητα σφάλματος στο μήνυμα του δέκτη αυθαίρετα μικρή.
Το σημαντικό χαρακτηριστικό αυτού του θεωρήματος είναι ότι τονίζει ότι για κάθε $R \leq C$ μπορεί να επιτευχθεί μετάδοση δίχως σφάλμα παρόντος του θορύβου. Ακόμα κι όταν ο θόρυβος του σήματος υπερισχύει, οδηγώντας κατά συνέπεια σε σφάλματα, το θεώρημα του Shannon λέει ότι αυτό δεν προκαλεί σφάλματα σε ένα μήνυμα.
Υπάρχει και η αρνητική διατύπωση που αντιστοιχεί στο θεώρημα του Shannon, σύμφωνα με το οποίο αν υπάρχει μια πηγή M, ισοπίθανων μηνυμάτων, με $M>1$, ή οποία παράγει πληροφορία με ρυθμό R, τότε αν $R > C$, τότε η πιθανότητα σφάλματος είναι πολύ κοντά στη μονάδα για κάθε πιθανή ομάδα Μ σημάτων του πομπού. Ουσιαστικά το αρνητικό θεώρημα δηλώνει ότι εάν ο ρυθμός πληροφορίας R υπερβαίνει μια προκαθορισμένη τιμή C, η πιθανότητα σφάλματος θα αυξάνει τείνοντας στη μονάδα, καθώς αυξάνει ο M.
Η κωδικοποίηση έχει την χρησιμότητα να μας επιτρέπει την αύξηση του ρυθμού με τον οποίον μπορεί να μεταδοθεί πληροφορία μέσω ενός καναλιού και ταυτόχρονα να διατηρείται ο ρυθμός σφάλματος σε μια καθορισμένη τιμή. Εναλλακτικά η κωδικοποίηση μας επιτρέπει να ελαττώνουμε το ρυθμό μετάδοσης εσφαλμένης πληροφορίας ενώ διατηρείται σταθερός ένας συγκεκριμένος ρυθμός μετάδοσης. Το τίμημα για την επιθυμία να φτάσουμε όσο πιο κοντά γίνεται στο όριο του Shannon, είναι η αυξημένη πολυπλοκότητα του υλικού τόσο στον πομπό, όπου γίνεται η κωδικοποίηση, όσο και στον δέκτη όπου επιτελείται η αποκωδικοποίηση.

1.2 Ανίχνευση και διόρθωση λαθών

Η απλούστερη τεχνική ανίχνευσης σφάλματος συνιστάται στην πρόσθεση ενός επιπλέον δυαδικού ψηφίου στο τέλος κάθε λέξης. Αυτό το πρόσθετο bit καλείται bit ελέγχου ισοτιμίας και επιλέγεται για να κάνει των αριθμό των «1» σε κάθε λέξη άρτιο. Το bit ισοτιμίας χρησιμοποιείται παγκόσμια στους ψηφιακούς υπολογιστές για ανίχνευση σφαλμάτων. Αυτό το bit ελέγχου είναι αποτελεσματικό αν η πιθανότητα σφάλματος σε ένα bit είναι τόσο χαμηλή που να μπορούμε να αγνοήσουμε την πιθανοφάνεια σφάλματος σε μια λέξη μήκους μεγαλύτερο από ένα bit .

Σε περίπτωση εμφάνισης σφάλματος στο δέκτη μια από τις τεχνικές που μπορεί να εφαρμοστεί για την αντιμετώπιση τους είναι τεχνική *Forward Error Correction*, κατά την οποία ο δέκτης σε περίπτωση ανίχνευσης σφάλματος εκτελεί και τη διόρθωση, σύμφωνα με τους κανόνες κωδικοποίησης. Η τεχνική FEC περιλαμβάνει δυο μεγάλες κατηγορίες κωδίκων, τους κώδικες δομής *(Block Codes)* και στους *συνελικτικούς κώδικες (Convolutional Codes).*

Το κύριο χαρακτηριστικό των κωδίκων δομής είναι ο τεμαχισμός της προς μετάδοση πληροφορίας σε μπλοκ *x* συμβόλων και η αντιστοιχία καθενός από αυτά σε ένα μπλοκ *y* συμβόλων, το οποίο ονομάζεται κωδική λέξη και όπου ισχύει (x ≥ y). Κάθε διαδικασία κωδικοποίησης εξαρτάται μόνο από την τρέχουσα πληροφορία εισόδου και όχι από τα προηγούμενα ψηφία εισόδου, πράγμα που σημαίνει ότι ο κωδικοποιητής δεν έχει μνήμη. Επίσης ένας κώδικας δομής εισάγει πλεονασμό στη πληροφορία με τέτοιο τρόπο, ώστε στο δέκτη να μπορεί να γίνει αποκωδικοποίηση με μηδενική πιθανότητα λάθους, εξασφαλίζοντας ότι ο ρυθμός μετάδοσης bit/sec δε θα υπερβεί την χωρητικότητα του καναλιού. Οι κώδικες δομής περιέχουν σταθερό μήκος των πακέτων που μεταδίδονται.

Στους συνελικτικούς κώδικες, κάθε *m-bit* σύμβολο πληροφορίας, μετατρέπεται σε ένα *n-bit* σύμβολο (n ≥ m) , με κάθε σύμβολο να εξαρτάται όχι μόνο από το ίδιο, αλλά και από τα k προηγούμενα από αυτό σύμβολα πληροφορίας.

Επομένως, η κύρια διαφορά μεταξύ των δύο κατηγοριών κωδικών είναι η ύπαρξη μνήμης στους συνελικτικούς κώδικες.

Μια από τις κατηγορίες κωδικών δομής είναι οι *γραμμικοί κώδικες δομής* στην οποία και ανήκουν και οι Κώδικες Χαμηλής –Πυκνότητας –Ελέγχου-Ισοτιμίας (LDPC). Ένας μπλοκ κώδικας είναι γραμμικός αν το άθροισμα οποιονδήποτε έγκυρων κωδικών λέξεων είναι επίσης κωδική λέξη. Στην περίπτωση δυαδικού κώδικα αυτό σημαίνει πως το αποτέλεσμα της λογικής πράξης XOR μεταξύ δύο κωδικών λέξεων, είναι επίσης κωδική λέξη. Στους γραμμικούς κώδικες ανήκει η μηδενική λέξη, καθώς το άθροισμα μιας οποιασδήποτε έγκυρης κωδικής λέξης με τον εαυτό της μας δίνει τη μηδενική λέξη και σύμφωνα με τον ορισμό των γραμμικών κωδικών, θα πρέπει και αυτή να είναι έγκυρη κωδική λέξη.

Η μετατροπή της ακολουθίας bits μιας λέξη πληροφορίας σε ακολουθία bits μιας κωδικής λέξης πραγματοποιείται με την βοήθεια ενός δυαδικού πίνακα *G*, ο οποίος ονομάζεται γεννήτορας πίνακας του κώδικα. Η κωδική λέξη (*c*) παράγεται με τον πολλαπλασιασμό της λέξης πληροφορίας (*u)* με τον γεννήτορα πίνακα (*G*):

$$c = u \oplus G$$

Για την ανίχνευση σφαλμάτων χρησιμοποιείται ο *Πίνακας Ελέγχου Ισοτιμίας (Parity Check Matrix)*, ο οποίος εξετάζει αν η λέξη που φτάνει στο δέκτη αποτελέι κωδική λέξη ή όχι. Ο έλεγχος σφαλμάτων που πραγματοποιείται από τον πίνακα ελέγχου ισοτιμίας *H*, γίνεται αν ισχύει η σχέση :

$$c * H^T = 0 (\mathrm{mod}\, 2)$$

όπου το *c* αναπαριστά την κωδική λέξη και το H^T αναπαριστά τον ανάστροφο πίνακα ελέγχου ισοτιμίας *H.*

Πρακτικά, στην περίπτωση του δυαδικού κώδικα, λόγω της modulo-2 άθροισης, κάθε μία γραμμή ελέγχει αν μεταξύ συγκεκριμένων ψηφίων της κωδικής λέξης υπάρχει άρτιο πλήθος άσων. Επομένως, ο πίνακας *H* ελέγχει την ισοτιμία των άσων της κωδικής λέξης. Για τον λόγο αυτό, ονομάζεται Πίνακας Ελέγχου Ισοτιμίας.

1.3 Έννοια της Αποκωδικοποίησης

Η διαδικασία της αποκωδικοποίησης (decoding) αναφέρεται στην παραγωγή των μηνυμάτων πληροφορίας από τις ληφθείσες κωδικές λέξεις. Επειδή όμως οι κωδικές λέξεις *c* αντιστοιχούν στα μηνυμάτα πληροφορίας *u* και αντίστροφα , αρκεί ο αποκωδικοποιητής να καθορίσει την κωδική λέξη που έχει αποσταλεί από τον πομπό, παρατηρώντας τη λαμβανόμενη ακολουθία *r.* Στη λαμβανόμενη αυτή ακολουθία *r* , υπάρχει πιθανότητα(λόγω της παρουσίας του θορύβου του καναλιού) ένα ή περισσότερα κωδικά bits να έχουν αλλάξει τιμή(σε σχέση με την κωδική λέξη που έχει αποσταλεί). Άρα αυτό που κάνει εν τέλει ο αποκωδικοποιητής , είναι να ανιχνεύει αν υπάρχουν κωδικά bits που έχουν αλλάξει τιμή(ανίχνευση λαθών) και αν είναι δυνατόν να διορθώνει την τιμή αυτών των bits. Η χρήση του βέλτιστου σχήματος αποκωδικοποίησης οδηγεί στον στόχο βέλτιστων επιδόσεων (χαμηλή πιθανότητα σφάλματος bit – *BER*), όμως το αντίτιμο είναι η υψηλή πολυπλοκότητα αποκωδικοποίησης, η οποία αυξάνει εκθετικά με το block length του κώδικα. Η σημαντική καινοτομία που εισήγαγε ο Gallager με τους LDPC κώδικες , ήταν η επαναληπτική (iterative) , αποκωδικοποίηση. Η δυνατότητα χρήσης της επαναληπτικής , message-passing αποκωδικοποίησης αποτελεί το κυριότερο συστατικό των LDPC κωδίκων καθώς μειώνει δραματικά την πολυπλοκότητα της αποκωδικοποίησης.

ΚΕΦΑΛΑΙΟ 2
Κώδικες Χαμηλής –Πυκνότητας –Ελέγχου-Ισοτιμίας (LDPC Codes)

2.1 Κώδικες Χαμηλής –Πυκνότητας –Ελέγχου-Ισοτιμίας(Low-Density-Parity-Check Codes)

Οι LDPC κώδικες είναι γραμμικοί block κώδικες διόρθωσης λαθών, οι οποίοι προσεγγίζουν ρυθμούς μετάδοσης δεδομένων πολύ κοντά στη χωρητικότητα καναλιού και των οποίων ο πίνακας ελέγχου ισοτιμίας Η είναι ένας αραιός πίνακας. Το κύριο χαρακτηριστικό των block κωδίκων είναι ο τεμαχισμός της προς μετάδοσης πληροφορίας σε μπλοκ των k συμβόλων και η αντιστοιχία καθενός από αυτά σε ένα μπλοκ n συμβόλων, το οποίο αποκαλείται κωδική λέξη η codeword ($n \geq k$). Λέγοντας αραιός πίνακας , εννοούμε έναν πίνακα ο οποίος περιέχει έναν πολύ μικρό αριθμό μη – μηδενικών στοιχείων. Η πυκνότητα ενός πίνακα μπορεί να οριστεί ως εξής:

$$\text{πυκνότητα πίνακα} = \frac{\text{αριθμός μη μηδενικών στοιχείων του πίνακα}}{\text{συνολικός αριθμός στοιχείων του πίνακα}}$$

Συγκεκριμένα, ο πίνακας ελέγχου ισοτιμίας, ο οποίος παρουσιάζεται στο σχήμα , έχει *n* στήλες, όπου κάθε στήλη περιέχει ένα μικρό αριθμό *j* άσσων και κάθε γραμμή περιέχει ένα μικρό αριθμό *i* άσσων.

$$H = \left[\begin{array}{cccccccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
\hline
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
\hline
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1
\end{array}\right]$$

Σχήμα 2.1 : Παράδειγμα ενός πίνακας ελέγχου ισοτιμίας με n=20, j=3, και i=4

Η αραιότητα του πίνακα Η είναι βασική , γιατί διευκολύνει την εφαρμογή της κωδικοποίησης. Η πυκνότητα του πίνακα Η αρκεί να είναι ικανοποιητικά μικρή έτσι ώστε να επιτρέπει τη χρήση της επαναληπτικής αποκωδικοποίησης. Η χρήση της επαναληπτικής αποκωδικοποίησης μειώνει σημαντικά την πολυπλοκότητα της αποκωδικοποίησης και συνεπώς επιτρέπει τη χρήση μεγάλων τιμών για το block length n.

2.2 Ιστορική αναδρομή

Το 1962 ο Robert Gallager παρουσίασε για πρώτη φορά τα οφέλη από τη χρήση γραμμικών , block κωδίκων οι οποίοι περιγράφονται από έναν αραιό(σε ότι αφορά την πυκνότητα των μη μηδενικών στοιχείων) πίνακα ελέγχου ισοτιμίας ο οποίος είχε μεγάλο μέγεθος(περίπου 500 στήλες). Οι κώδικες αυτοί ονομάστηκαν χαμηλής πυκνότητας κώδικες ελέγχου ισοτιμίας(Low Density Parity Check codes – LDPC). Η μεγάλη καινοτομία την οποία εισήγαγε όμως ο Gallager δεν ήταν απλά η χρήση ενός αραιού , μεγάλου πίνακα ελέγχου ισοτιμίας, αλλά η αραιότητα του πίνακα ελέγχου ισοτιμίας καθιστά τους LDPC κώδικες επιδεκτικούς σε διάφορους αλγόριθμους επαναληπτικής αποκωδικοποίησης(iterative decoding). Η αλματώδης ανάπτυξη όμως των LDPC κωδίκων δεν ήρθε ταυτόχρονα με την εισαγωγή τους. Αντίθετα , η δουλειά του Gallager έμεινε στην αφάνεια για τα επόμενα σχεδόν 20 χρόνια. Ο κύριος λόγος που συνέβη αυτό ήταν τεχνολογικός. Η υπολογιστική δύναμη των ηλεκτρονικών υπολογιστών της εποχής ήταν πολύ περιορισμένη. Επομένως αδυνατώντας να προσομοιώσει μεγάλου μήκους κώδικες , δεν μπορούσε να δείξει τις σημαντικές επιδόσεις των LDPC κωδίκων.Η επανεμφάνιση των LDPC κωδίκων έγινε το 1981 από τον Michael Tanner. Ο Tanner εισήγαγε μια νέα ερμηνεία των LDPC κωδίκων , από μια άλλη σκοπιά με τη χρήση της θεωρίας διμερών γραφημάτων. Ο Tanner λοιπόν, περιγράφει ένα μεγάλο κώδικα ως σύνθεση απλούστερων, μικρών κωδίκων ενώ χρησιμοποιεί την επαναληπτικότητα – αναδρομικότητα(συνεργασία μεταξύ των επιμέρους κωδίκων) για τη μείωση της πολυπλοκότητας. Για την περιγραφή της ανάλυσής του ο Tanner χρησιμοποιεί τα λεγόμενα διμερή γραφήματα. Αλλά και η δουλειά του Tanner αγνοήθηκε και έμεινε και αυτή στην αφάνεια(προσωρινά). Η αναγέννηση των LDPC κωδίκων επήλθε από το 1995 και μετά. Συγκεκριμένα το 1995 οι D. MacKay και R. Neal σημείωσαν τα σημαντικά οφέλη της χρήσης αραιών πινάκων ελέγχου ισοτιμίας για δυαδικούς block κώδικες. Στη μελέτη που δημοσίευσαν παρουσίασαν για πρώτη φορά(μέσω προσομοιώσεων με υπολογιστή) την ικανότητα των LDPC κωδίκων να λειτουργούν κοντά στο όριο Shannon για το Binary Symmetric Channel(BSC). Επίσης το 1996 οι N. Alon και M. Luby αλλά και το 1998 οι J. Byers , M. Luby , M. Mitzenmacher και D. Spielman παρατήρησαν τα πλεονεκτήματα της χρήσης ενός πίνακα ελέγχου ισοτιμίας με χαμηλή πυκνότητα. Όλοι αυτοί οι ερευνητές οδήγησαν όχι μόνο στην επανεισαγωγή των LDPC κωδίκων αλλά και στη γενίκευσή τους. Έτσι λοιπόν κατασκευάστηκαν μεγάλου μήκους LDPC κώδικες οι οποίοι με τη χρήση της επαναληπτικής αποκωδικοποίησης κατάφεραν να επιτύχουν επιδόσεις πολύ κοντά στο όριο Shannon.

2.3 Ομαλοί και Ανώμαλοι κώδικες LDPC

Οι LDPC κώδικες ταξινομούνται σε δύο κατηγορίες, τους ομαλούς (Regular) και τους ανώμαλους (Irregular) κώδικες.

Ένας ομαλός LDPC κώδικας χαρακτηρίζεται από την ύπαρξη ενός πίνακα ελέγχου ισοτιμίας, Η, ο οποίος περιέχει ακριβώς *wc*(βαθμός στήλης) μη μηδενικά στοιχεία σε κάθε του στήλη αλλά και *wr*(βαθμός γραμμής) $= wc*(\mathrm{n}/\mathrm{m})$ μη μηδενικά στοιχεία σε κάθε γραμμή του. Δηλαδή, οι αριθμοί των μονάδων των στηλών και των γραμμών του πίνακα Η είναι σταθεροί και συγκεκριμένα ονομάζονται βαθμός στήλης (column degree) και βαθμός γραμμής (row degree) αντίστοιχα.

Παράδειγμα **:** Θεωρούμε τον παρακάτω πίνακα ελέγχου ισοτιμίας **:**

$$H = \begin{bmatrix} 01011001 \\ 11100100 \\ 00100111 \\ 10011010 \end{bmatrix}$$

LDPC κώδικας είναι Regular, με βαθμό στήλης *wc* = 2 και βαθμό γραμμής

$$wr = 2*(8/4) = 4$$

Αντίθετα, αν το πλήθος των μη μηδενικών στοιχείων των γραμμών ή των στηλών του πίνακα δεν είναι σταθερό, τότε ο πίνακας ονομάζεται ανώμαλος (Irregular).

$$H = \begin{bmatrix} 11110001 \\ 11110110 \\ 00100111 \\ 00001010 \end{bmatrix}$$

LDPC κώδικας είναι Irregular.

2.4 Τρόποι αναπαράστασης των LDPC κωδίκων

Οι κώδικες Χαμηλής –Πυκνότητας –Ελέγχου-Ισοτιμίας (LDPC Codes) μπορούν να αναπαρασταθούν με δυο τρόπους. Όπως το σύνολο των κωδίκων δομής, δύνανται να περιγραφούν μέσω πινάκων. Υπάρχει όμως και μία εναλλακτική μέθοδος απεικόνισης, η οποία χρησιμοποιεί γράφους Tanner. Οι δύο αυτοί τρόποι αναπαράστασης είναι απολύτως ισοδύναμοι και χρησιμοποιούνται εξίσου.

Αναπαράσταση των κωδικών LDPC με βάση του πίνακα ελέγχου ισοτιμίας

Ως γραμμικοί block κώδικες , οι LDPC κώδικες μπορούν να αναπαρασταθούν πλήρως με τη χρήση του πίνακα ελέγχου ισοτιμίας τους. Συγκεκριμένα ένας LDPC κώδικας μπορεί να θεωρηθεί ως ο μηδενικός(null) ή δυικός(dual) χώρος ενός *mxn* πίνακα ελέγχου ισοτιμίας Η ο οποίος έχει χαμηλή πυκνότητα από άσους.

Ο τρόπος αυτός αναπαράστασης εισήχθη από τον ίδιο το Gallager , όταν εισήγαγε και περιέγραφε τους LDPC κώδικές του.

Σημαντικές παράμετροι κατά την αναπαράσταση ενός LDPC κώδικα με τη χρήση του πίνακα Η είναι οι εξής **:**

- Το πλήθος των γραμμών του πίνακα ελέγχου ισοτιμίας. Η παράμετρος *m* είναι ουσιαστικά το πλήθος των εξισώσεων ελέγχου ισοτιμίας και κατά κανόνα η αύξηση της τιμής της ενισχύει την ικανότητα διόρθωσης λαθών του κώδικα(περισσότερες εξισώσεις ελέγχουν τις τιμές των bits).
- Το πλήθος των στηλών του πίνακα ελέγχου ισοτιμίας. Η παράμετρος *n* είναι ουσιαστικά το *block length* του κώδικα και η αύξηση της τιμής της βελτιώνει τις επιδόσεις του κώδικα.
- Ο βαθμός(*rank*) του πίνακα *H* ο οποίος ισούται με το πλήθος των γραμμικά ανεξάρτητων γραμμών του πίνακα ελέγχου ισοτιμίας. Ο βαθμός του πίνακα *H* καθορίζει το ρυθμό του κώδικα. Έτσι λοιπόν ρυθμός σχεδιασμού *r (design rate*) ισούται:

$$r = \frac{n - m}{n}$$

και στον οποίο υποθέτουμε ότι ο πίνακας *H* δεν έχει γραμμικά εξαρτημένες γραμμές.

Στην πραγματικότητα όμως ο πίνακας *H* ενδέχεται να έχει γραμμικά εξαρτημένες γραμμές , οπότε ο πραγματικός ρυθμός *R*(*actual rate*) του κώδικα θα είναι:

$$R = \frac{n - rankH}{n}$$

όπου rankH είναι ο βαθμός του πίνακα *H*.

- Η κατανομή βαρών(weight distribution) των γραμμών και των στηλών του πίνακα *H*. Ως βάρος μιας γραμμής ή μιας στήλης του πίνακα *H* εννοούμε το πλήθος των μη-μηδενικών στοιχείων αυτής.

Παρακάτω δίνεται ένα παράδειγμα όπου δείχνει τον υπολογισμό της κατανομής βαρών των γραμμών και των στηλών ενός πίνακα Η.

Παράδειγμα **:** Θεωρούμε τον παρακάτω πίνακα ελέγχου ισοτιμίας **:**

$$H = \begin{bmatrix} 1\,1\,0\,1\,1\,0\,0\,1\,0\,0 \\ 0\,1\,1\,0\,1\,1\,1\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,0\,1\,0 \\ 0\,0\,1\,0\,0\,1\,0\,1\,0\,1 \end{bmatrix}$$

Ο πίνακας αυτός έχει *n* = 10 στήλες από τις οποίες οι 7 έχουν βάρος 2 και οι 3 έχουν βάρος 3. Άρα η κατανομή βαρών των στηλών του πίνακα *H* θα περιγράφεται από το διάνυσμα *v* = (0 , 0.7 , 0.3).
Επίσης ο πίνακας *H* έχει *m* = 5 γραμμές από τις οποίες οι 2 έχουν βάρος 4 και οι 3 έχουν βάρος 5. Άρα η κατανομή βαρών των γραμμών του πίνακα *H* θα περιγράφεται από το διάνυσμα *h* =(0 , 0 , 0 , 0.4 , 0.6).

Ο πίνακας ελέγχου ισοτιμίας *H* ενός LDPC κώδικα , σύμφωνα με το Gallager , θα πρέπει επίσης να έχει τις εξής ιδιότητες:

- Κάθε γραμμή του *H* αποτελείται από σταθερό αριθμό 1
- Κάθε στήλη του *H* αποτελείται από σταθερό αριθμό 1
- Ο αριθμός των 1 που έχουν σε κοινή θέση δύο οποιεσδήποτε γραμμές(ή δύο οποιεσδήποτε στήλες) δε θα πρέπει να είναι μεγαλύτερος από 1. Ο περιορισμός αυτός ονομάζεται περιορισμός γραμμών – στηλών(Row – Column Constraint)

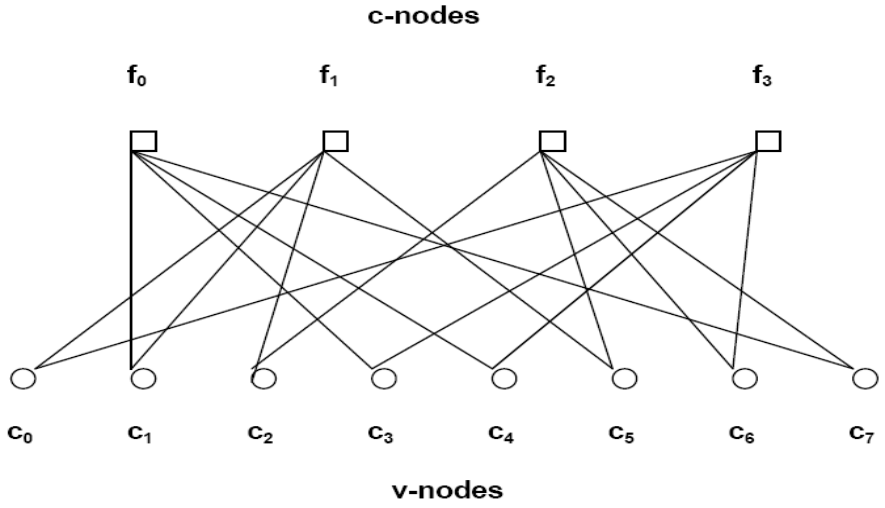
Γραφική αναπαράσταση

Ο δεύτερος τρόπος αναπαράστασης των LDPC κωδίκων προτάθηκε από τον Tanner, ο οποίος εισήγαγε τους γράφους Tanner. Το 1981 , ο Tanner αναπτύσσει την αναδρομική προσέγγιση(recursive approach) της κωδικοποίησης ώστε να πετύχει την κατασκευή μεγάλου μήκους , ισχυρών κωδίκων. Λέγοντας αναδρομικότητα εννοούμε τη σύνθεση μεγάλου μήκους , ισχυρών κωδίκων από απλούστερους , μικρού μήκους κώδικες. Η τεχνική αυτή βέβαια ήταν ήδη γνωστή από το 1965 με τους αλυσιδωτούς κώδικες του D. Forney. Η σημαντική προσθήκη του Tanner , αφορά την εισαγωγή της θεωρίας γραφημάτων για την περιγραφή τέτοιων κωδίκων. Έτσι ο Tanner επεκτείνει περαιτέρω την ήδη γνωστή αναδρομική προσέγγιση για την κατασκευή κωδίκων , αναλύοντας τους κώδικες αυτούς με γραφήματα.

Το κύριο πλεονέκτημα της χρήσης της αναδρομικότητας για την κατασκευή ενός κώδικα είναι η σημαντική μείωση της πολυπλοκότητας που απαιτείται για την υλοποίηση του κώδικα αυτού. Από την άλλη το κύριο μειονέκτημα της αναδρομικότητας είναι ότι δεν επιτρέπει στους κώδικες να αποδίδουν το βέλτιστο των δυνατοτήτων τους και συνεπώς οι επιδόσεις τέτοιων κωδίκων είναι υποβέλτιστες. Αυτό συμβαίνει επειδή δε βλέπουμε το συνολικό κώδικα ως έναν ενιαίο , μεγάλου μήκους κώδικα(και επομένως δεν εκμεταλλευτούμε όλη την «ισχύ» του) αλλά τον βλέπουμε ως κομμάτια πολλών μικρών κωδίκων(μικρής «ισχύος»).Οι γράφοι Tanner ανήκουν στην κατηγορία των *διμερών γράφων (bipartite graphs).* Ένας γράφος ονομάζεται «διμερής» όταν οι κόμβοι του χωρίζονται σε δύο ομάδες και μόνο κόμβοι διαφορετικών ομάδων μπορούν να ενωθούν μεταξύ τους. Οι δύο τύποι κόμβων που υπάρχουν σε ένα γράφο Tanner ονομάζονται κόμβοι μεταβλητών (*variable nodes),* οι οποίοι συνήθως αναφέρονται ως v-nodes και κόμβοι ελέγχου (*check nodes),* οι οποίοι συνήθως αναφέρονται ως c-nodes. Ο γράφος Tanner ενός κώδικα σχεδιάζεται ακολουθώντας το εξής κανόνα:

«Ο i-οστός κόμβος ελέγχου συνδέεται με τον j-οστό κόμβο μεταβλητών μόνο όταν το στοιχείο Hij του πίνακα ελέγχου ισοτιμίας, Η, είναι ίσο με 1».

Στο παράδειγμα που ακολουθεί παρουσιάζεται ο γράφος του κώδικα που αντιστοιχεί στον πίνακα ελέγχου ισοτιμίας *H*

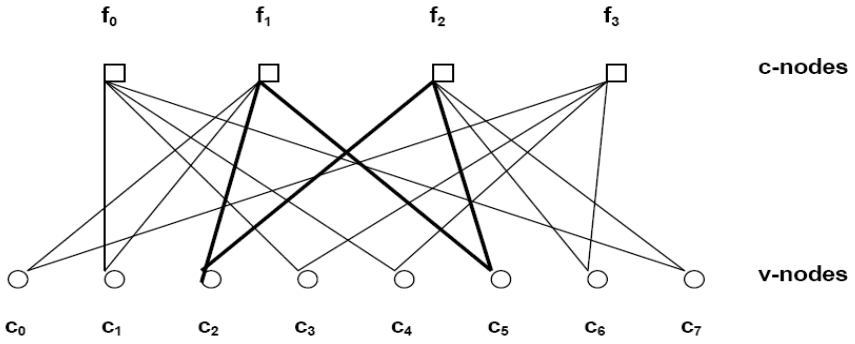


$$H = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Σχήμα 2.2 : Γράφος Tanner και ο αντίστοιχος πίνακας ελέγχου ισοτιμίας

Παρατηρούμε ότι ο κόμβος ελέγχου f_0 συνδέεται με τους κόμβους μεταβλητών c_1, c_3, c_4 και c_7 αφού τα στοιχεία h_{01}, h_{03}, h_{04} και h_{07} του πίνακα H έχουν τιμή «1». Με τον ίδιο τρόπο, ο κόμβος f_1 συνδέεται με τους c_0, c_1, c_2 και c_5, αφού $h_{10} = h_{11} = h_{12} = h_{15} = 1$. Ομοίως ο c-node f_2 συνδέεται με τους v-nodes c_2, c_5, c_6 και c_7, διότι $h_{20} = h_{25} = h_{26} = h_{27} = 1$ και τέλος, ο κόμβος f_3 συνδέεται με τους κόμβους c_0, c_3, c_4 και c_6 δηλώνοντας ότι τα στοιχεία h_{30}, h_{33}, h_{34} και h_{36} έχουν τιμή «1».

Γνωρίζοντας ότι σε κάθε γραμμικό κώδικα δομής πρέπει να ισχύει η σχέση $c^T * \mathrm{H} = 0$, ώστε να είναι δυνατή η ανίχνευση και διόρθωση σφαλμάτων, παρατηρούμε ότι οι τιμές των ψηφίων της πληροφορίας (*variable nodes*), που συνδέονται στον ίδιο κόμβο ελέγχου (*check node*), πρέπει να δίνουν μηδενικό άθροισμα.
Επίσης στο παράδειγμα που περιγράφεται, ο LDPC κώδικας είναι ομαλός, με βαθμό στήλης *wc* = 2 και βαθμό γραμμή *wr*=2 · (8/4)=4.
Πληροφορίες για την ομαλότητα του κώδικα μπορούμε να πάρουμε και από τον γράφο Tanner. Ο κώδικας είναι ομαλός όταν ο αριθμός των εισερχόμενων γραμμών είναι ίδιος για όλους τους κόμβους μεταβλητών και επίσης για όλους τους κόμβους ελέγχου. Από το παραδείγμα, παρατηρούμε ότι σε κάθε v-node εισέρχονται 2 γραμμές σύνδεσης, ενώ σε κάθε *c-node* εισέρχονται σταθερά 4 γραμμές σύνδεσης.
Ένα ακόμα μέγεθος που χαρακτηρίζει τους LDPC κώδικες προκύπτει από τη γραφική αναπαράσταση των κωδίκων και είναι ο κύκλος (*cycle*) ή βρόχος (*loop*) μήκους *i* , ο οποίος αποτελεί ένα μονοπάτι με συνολικά *i* ακμές, του οποίου η αφετηρία και το τέλος συμπίπτουν. Σύμφωνα με το παράδειγμα που εξετάσαμε παρατηρούμε ο γράφος Tanner που προέκυψε έχει κύκλο μήκους *4*, ο οποίος παρουσιάζεται με έντονες γραμμές στο σχήμα που ακολουθεί.



Σχήμα 2.3 : Κύκλος μήκους 4

Το ελάχιστο δυνατό μήκος ενός διμερούς γράφου είναι ο κύκλος μήκους 4. Κύκλοι αυτής της μορφής, δηλαδή μήκους 4, δηλώνουν την παρουσία τους στον πίνακα ελέγχου ισοτιμίας *H*, σαν τέσσερις μονάδες στις γωνίες κάποιου υποπίνακα του *H*. Οι μικροί κύκλοι, όπως και ο κύκλος του παραδείγματος, προτείνεται να αποφεύγονται καθώς υποβαθμίζουν την ποιότητα της αποκωδικοποίησης. Ολοκληρώνοντας λοιπόν την ενότητα αυτή , μπορούμε να πούμε ότι οι δύο τρόποι αναπαράστασης των LDPC κωδίκων είναι απολύτως ισοδύναμοι.

2.5 Πλεονεκτήματα και Μειονεκτήματα των LDPC κωδίκων

Ο πίνακας ελέγχου ισοτιμίας καθορίζει το κύκλωμα και κατά συνέπεια την πολυπλοκότητα του κωδικοποιητή. Πιο συγκεκριμένα, η πολυπλοκότητα είναι ανάλογη των μηδενικών στοιχείων του πίνακα ισοτιμίας. Όσο λιγότεροι, επομένως είναι οι άσσοι του πίνακα ισοτιμίας, τόσο απλούστερος ο κωδικοποιητής, γεγονός που φανερώνει το σημαντικό πλεονέκτημα των κωδικών LDPC, ως προς την υλοποίηση, έναντι άλλων μπλοκ κωδικών ίδιου μεγέθους.

Άλλο ένα σημαντικό πλεονέκτημα των κωδίκων LDPC, είναι το γεγονός πως οι χρησιμοποιούμενοι αλγόριθμοι αποκωδικοποίησης, πέρα από τη διόρθωση σφαλμάτων προσφέρουν σαφή και έγκυρη ανίχνευση των περιπτώσεων εκείνων που η αποκωδικοποίηση αποτυγχάνει.

Ένα εξαιρετικής σημασίας πλεονέκτημα των κωδικών LDPC είναι οι πολύ καλές επιδόσεις τους. Ενώ μέχρι πριν λίγα χρόνια οι κώδικες turbo ήταν εκείνοι οι οποίοι κατείχαν τα πρωτεία όσον αφορά τις επιδόσεις, πλέον οι κώδικες LDPC φαίνεται να κατέχουν αυτή τη θέση. Υποστηρίζουν ρυθμούς μετάδοσης, οι οποίοι πλησιάζουν τη χωρητικότητα καναλιού (όριο Shannon).

Το βασικό μειονέκτημα είναι ότι για να επιτευχθεί αυτός ο ρυθμός μετάδοσης πρέπει να χρησιμοποιηθεί κώδικας πολύ μεγάλου μήκους. Αυτό σημαίνει πως απαιτείται πολύ μεγάλος πίνακας ελέγχου ισοτιμίας. Όσο αραιός και να είναι ένας τόσο μεγάλος πίνακας, ο αριθμός των μη μηδενικών του στοιχείων είναι εξαιρετικά μεγάλος, με αποτέλεσμα η διαδικασία κωδικοποίησης αλλά και αποκωδικοποίησης να είναι εξαιρετικά απαιτητική από άποψη πολυπλοκότητας.

Τα κύρια πλεονεκτήματα των LDPC κωδίκων σε σχέση με τους Turbo κώδικες είναι τα εξής:

- Οι LDPC κώδικες δεν απαιτούν τη χρήση αναδιατάκτη (*interleaver),* ώστε να επιτύχουν καλές επιδόσεις. Αντίθετα στους Turbo κώδικες η χρήση αναδιατάκτη είναι απαραίτητη.
- Οι LDPC κώδικες(σε αντίθεση με τους Turbo κώδικες) δεν είναι κατοχυρωμένοι σε κάποιο πρόσωπο ή οργανισμό και συνεπώς αποτελούν μια οικονομικά και τεχνικά ελκυστική επιλογή για χρήση σε εμπορικά προϊόντα.
- LDPC κώδικες επιτυγχάνουν καλύτερες επιδόσεις από τους Turbo κώδικες σε ότι αφορά τα λάθη(*block error performance*).
- Στους LDPC κώδικες , το επίπεδο του error floor παρατηρείται σε πολύ χαμηλότερα BER σε σχέση με τους Turbo κώδικες
- Ο τρόπος που γίνεται η αποκωδικοποίηση των LDPC κωδίκων την καθιστά επιδεκτική στην εφαρμογή παράλληλης αρχιτεκτονικής – επεξεργασίας . Οι Check Nodes και οι Variable Nodes μπορούν να θεωρηθούν ως επεξεργαστές (κατά την αποκωδικοποίηση) οι οποίοι λειτουργούν ταυτόχρονα , γεγονός που επιτρέπει την εφαρμογή παράλληλης υλοποίησης.

ΚΕΦΑΛΑΙΟ 3
Αλγόριθμοι Κωδικοποίησης

3.1 Κωδικοποίηση μέσω του πίνακα G

Οι LDPC κώδικες , όπως προαναφέραμε , είναι γραμμικοί , block κώδικες που περιγράφονται από έναν αραιό πίνακα ελέγχου ισοτιμίας *H* , διαστάσεων *mxn*. Αν μπορούσαμε να βρούμε το γεννήτορα πίνακα *G*(διαστάσεων *kxn*) που αντιστοιχεί στον πίνακα *H* που περιγράφει τον LDPC κώδικα , τότε η διαδικασία της κωδικοποίησης θα μπορούσε να πραγματοποιηθεί πολύ απλά μέσω της έκφρασης:

$$c = u * G \tag{3.1}$$

όπου *u* είναι το μήνυμα πληροφορίας που πρόκειται να κωδικοποιηθεί και *c* είναι η κωδική λέξη που παράγεται. Ο πίνακας *g* αποτελείται από 0 και 1 που καθορίζουν αν ένα bit πληροφορίας θα μετέχει στην παραγωγή ενός κωδικού bit ή όχι. Αν *g i,j* = 1 τότε το i-στο bit της ακολουθίας πληροφορίας θα μετέχει στην παραγωγή του *j*-στου bit της κωδικής λέξης ενώ αν *g i,j* = 0 τότε το *i*-στο bit της ακολουθίας πληροφορίας δε θα μετέχει στην παραγωγή του *j*-στου bit της κωδικής λέξης. Ουσιαστικά η τιμή των *g i,j* καθορίζει αν θα έχουμε σύνδεση της *i*-στης εισόδου στην *j*-στη έξοδο.
Πιο συγκεκριμένα η σχέση (3.1) γράφετε:

$$\begin{aligned}
u_1 &= g_{1,1} \cdot u_1 \oplus g_{2,1} \cdot u_2 \oplus \ldots \oplus g_{k,1} \cdot u_k \\
u_2 &= g_{1,2} \cdot u_1 \oplus g_{2,2} \cdot u_2 \oplus \ldots \oplus g_{k,2} \cdot u_k \\
&\cdot \\
&\cdot \\
&\cdot \\
u_n &= g_{1,\mathrm{n}} \cdot u_1 \oplus g_{2,\mathrm{n}} \cdot u_2 \oplus \ldots \oplus g_{k,\mathrm{n}} \cdot u_k
\end{aligned} \tag{3.2}$$

Η παραγωγή του γεννήτορα πίνακα *G* πραγματοποιείται με τα εξής βήματα **:**

- Αρχικά έχουμε έναν αραιό πίνακα *H* ο οποίος περιγράφει τον LDPC κώδικα και αποτελείται από *m* γραμμές και *n* στήλες. Οι *m* γραμμές του πίνακα *H* δεν είναι απαραίτητο να είναι όλες γραμμικά ανεξάρτητες μεταξύ τους.

- Με την εφαρμογή της απαλοιφής Gauss(Gauss elimination) μετατρέπουμε τον πίνακα *H* σε συστηματική μορφή **:**

$$H' = [\mathrm{P} \mid \mathrm{I}_{\mathrm{n-k}}] \tag{3.3}$$

Όπου μετά την απαλοιφή Gauss ο πίνακας *H'* έχει rank*(H) = n-k ≤ m* γραμμές(οι οποίες είναι και γραμμικά ανεξάρτητες) και *n* στήλες, *rank(H)* είναι ο βαθμός του πίνακα *H* , I_{n-k} είναι ο μοναδιαίος πίνακας με *n-k* γραμμές και *P* είναι ένας *(n-k)*k* πίνακας. Τα στοιχεία του πίνακα *P* καθορίζουν τον τρόπο με τον οποίο παράγονται τα bits ελέγχου ισοτιμίας από τα bits πληροφορίας.

- Ο πίνακας *G* θα δίνεται από τη σχέση **:**

$$G = [\mathrm{I}_{\mathrm{k}} \mid \mathrm{P}^{\mathrm{T}}] \tag{3.4}$$

και θα αποτελείται από *k* γραμμές και *n* στήλες. Όπου I_k είναι ο μοναδιαίος πίνακας με *k* γραμμές και P^T είναι ο ανάστροφος πίνακας του *P*.

Η παραπάνω διαδικασία χρησιμοποιήθηκε στα αρχικά στάδια της υλοποίησης των LDPC κωδίκων αλλά είχε ένα πολύ βασικό μειονέκτημα**:** ο πίνακας *G* που προέκυψε από την παραπάνω διαδικασία ήταν μη αραιός(με μεγάλη πιθανότητα) παρ' ότι ο πίνακας *H* ήταν αραιός . Αυτό σημαίνει ότι η πολυπλοκότητα της κωδικοποίησης είναι τετραγωνικής τάξης ή τετραγωνικού χρόνου(*quadratic time*). Έτσι διαπιστώνουμε ότι δεν εκμεταλλευόμαστε (στη διαδικασία της κωδικοποίησης) την ιδιότητα της αραιότητας που έχει ο πίνακας *H* των LDPC κωδίκων , αφού η πολυπλοκότητα της κωδικοποίησης με την παραπάνω μέθοδο είναι ίδια με αυτή των κοινών , γραμμικών (ελέγχου ισοτιμίας – parity check) κωδίκων.

3.2 Κωδικοποίηση μέσω του πίνακα H

Οι LDPC κώδικες περιγράφονται μέσω του πίνακα ελέγχου ισοτιμίας τους *H*(ο οποίος είναι και αραιός). Θα πρέπει λοιπόν να πραγματοποιήσουμε την κωδικοποίηση μέσω του πίνακα *H*(αντί του πίνακα *G*) ώστε να εκμεταλλευτούμε την αραιότητα του πίνακα *H*. Θεωρούμε λοιπόν ότι έχουμε έναν LDPC κώδικα που περιγράφεται από έναν πίνακα ελέγχου ισοτιμίας *H* διαστάσεων *mxn* και μια κωδική λέξη αυτού του κώδικα θα είναι μία n–άδα *c* και περιγράφετε από την σχέση:

$$c^T * H = 0 \tag{3.5}$$

Το στάδιο της προ-επεξεργασίας περιλαμβάνει τη μετατροπή του πίνακα *H* σε συστηματική μορφή με την εφαρμογή της απαλοιφής Gauss . Έτσι ο πίνακας *H* μετατρέπεται στον ισοδύναμο πίνακα *H'* **:**

$$H' = [\mathrm{P} \mid \mathrm{I}_{\mathrm{n-k}}] \tag{3.6}$$

Μετά την απαλοιφή Gauss ο πίνακας *H'* θα έχει *rank(H) = n-k ≤ m* γραμμές(οι οποίες είναι και γραμμικά ανεξάρτητες) και *n* στήλες. I_{n-k} είναι ο μοναδιαίος πίνακας με *n-k* γραμμές και *P* είναι ένας *(n-k)xk* πίνακας. Η εξίσωση (3.6) μπορεί να γραφεί ως εξής:

$$
\begin{aligned}
u_{k+1} &= u_{1\cdot p1,1} \oplus u_2 \cdot p_{1,2} \oplus \ldots \oplus u_k \cdot p_{1,k} \\
u_{k+2} &= u_{1\cdot p2,1} \oplus u_2 \cdot p_{2,2} \oplus \ldots \oplus u_k \cdot p_{2,k} \\
&\cdot \\
&\cdot \\
&\cdot \\
u_n &= u_1 \cdot p_{n-k,1} \oplus u_2 \cdot p_{\mathrm{n-k},2} \oplus \ldots \oplus u_k \cdot p_{\mathrm{n-k},k}
\end{aligned}
\qquad (3.7)
$$

Μια κωδική λέξη *c* ενός κώδικα σε συστηματική μορφή θα έχει τη μορφή:
c = (u_1, u_2, …, u_k, $υ_{k+1}$, $υ_{k+2}$, …, $υ_n$) όπου *u = (u_1, u_2, … , u_k)* είναι η ακολουθία πληροφορίας**.** Η κωδικοποίηση μπορεί να γίνει σε δύο απλά βήματα:

- Τα πρώτα *k* bits($υ_1$, $υ_2$, …, $υ_k$) της κωδικής λέξης είναι ίδια με τα *k* bits της ακολουθίας πληροφορίας *u.*
- Τα υπόλοιπα *n-k* bits*($υ_{k+1}$, $υ_{k+2}$, … , $υ_n$)* της κωδικής λέξης αποτελούν τα bits ελέγχου ισοτιμίας(parity check bits).

Όμως μετά την εφαρμογή της απαλοιφής Gauss (στάδιο της προ επεξεργασίας) ο πίνακας *P* δε θα είναι πλέον αραιός και η πολυπλοκότητα της κωδικοποίησης είναι και πάλι τετραγωνικού χρόνου.

3.3 Αλγόριθμός Κωδικοποίησης- R&U LDPC

Το 2001 οι T. Richardson και R. Urbanke παρουσίασαν μια πιο αποδοτική διαδικασία κωδικοποίησης η οποία μπορεί να προσεγγίσει την πολυπλοκότητα γραμμικού χρόνου. Ο R&U αλγόριθμος είναι από τους πρώτους αλγόριθμους κωδικοποίησης που είχαν ως σκοπό, την επίλυση της πολυπλοκότητας του κωδικοποιητή και την εύρεση μιας αποδοτικής μεθόδου κωδικοποίησης. Η υψηλή αποδοτικότητα του κωδικοποιητή επιτυγχάνεται από την σωστή εκμετάλλευση της αραιότητας του πίνακα ελέγχου ισοτιμίας *H*, και από τον αλγόριθμο που θα μπορούσε να εφαρμοστεί πάνω σε οποιοδήποτε «αραιό» πίνακα *H.*

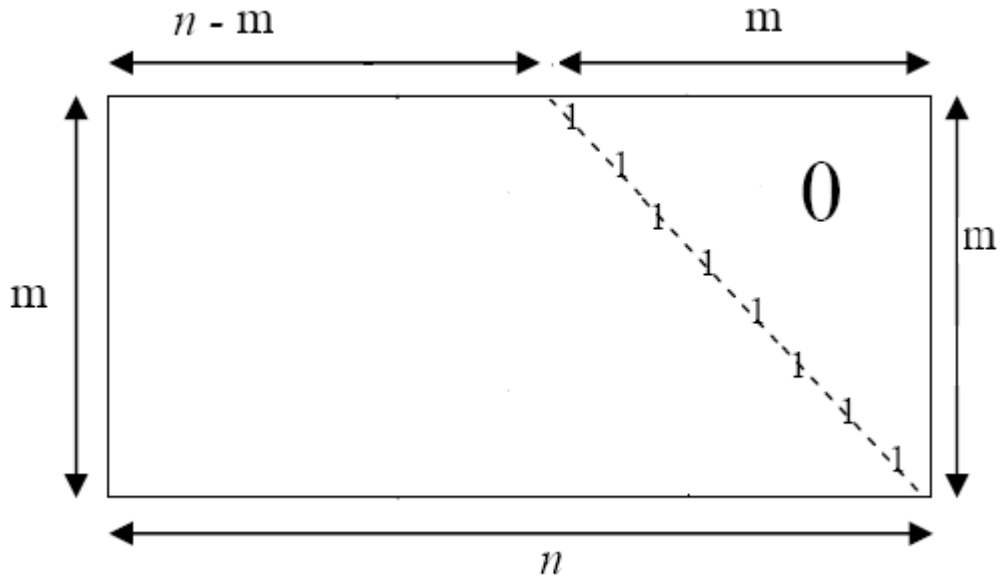
Οι κώδικες LDPC είναι γραμμικοί κώδικες δομής και η κωδικοποίηση τέτοιων κωδίκων ακολουθεί την σχέση:

$$c^T * H = 0 \tag{3.8}$$

όπου *c* είναι κωδική λέξη και *H* είναι ο πίνακας ελέγχου ισοτιμίας.

Μια απευθείας μέθοδος κωδικοποίησης απαιτεί τρία βασικά βήματα:

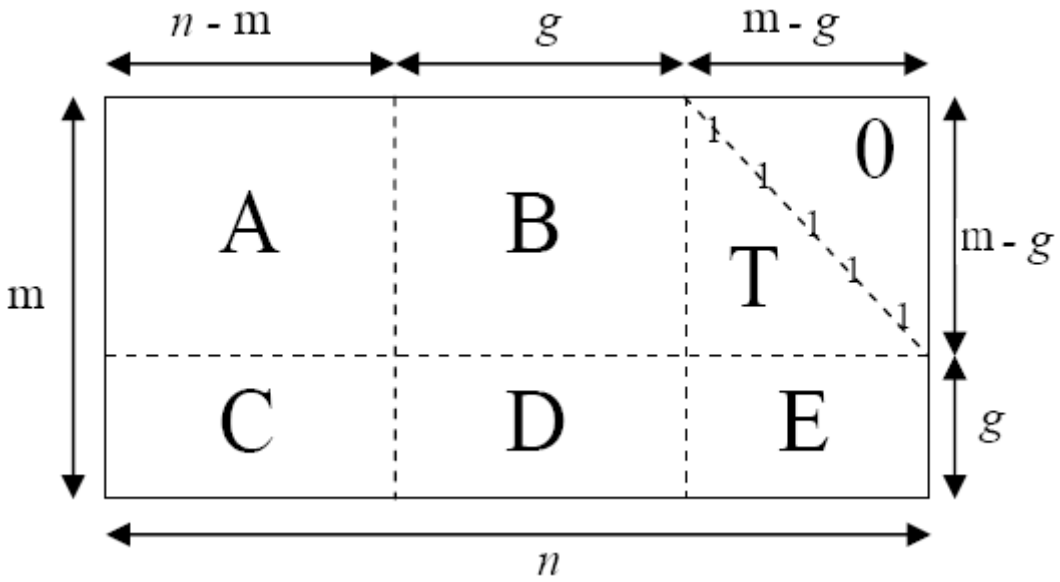
- Στο στάδιο της προ-επεξεργασίας έχουμε τον πίνακα *H* στον οποίον πραγματοποιούμε αντιμεταθέσεις γραμμών και στηλών ώστε να φέρουμε τον πίνακα ελέγχου ισοτιμίας *H* στη προσεγγιστική κάτω τριγωνική(*approximate lower triangular*) μορφή. Να τονίσουμε ότι στην προσεγγιστική κάτω τριγωνική μορφή του, ο πίνακας *H* παραμένει αραιός.



Σχήμα 3.1: Προσεγγιστική κάτω τριγωνική μορφή ενός πίνακα ελέγχου ισοτιμίας

- Διαχωρισμό του διανύσματος *c* σε bits πληροφορίας και parity bits, *c=[s, p_1, p_2],* όπου *s* είναι τα bits πληροφορίας και τα p_1 και p_2 αποτελούν τα parity bits.
- Τελευταίο βήμα είναι η επίλυση της εξίσωσης $c^T * H = 0$

Υποθέτουμε ότι έχουμε πετύχει την μετατροπή του πίνακα *H* και έχουμε φέρει τον πίνακα ελέγχου ισοτιμίας *H* σε προσεγγιστική κάτω τριγωνική μορφή, δηλαδή στο πάνω δεξιά μέρος έχουμε μόνο μηδενικά όπως φαίνεται και στο παρακάτω εικόνα:



Σχήμα 3.2: Η καινούργια μορφή του πίνακα ελέγχου ισοτιμίας, παραμένει η μορφή του κάτω τριγωνικού πίνακα

Πιο συγκεκριμένα, ο ισοδύναμος πίνακας Η θα έχει την εξής μορφή:

$$H' = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} \tag{3.9}$$

Όπου:
A είναι ένας (m – g)x(n – m) πίνακας ,
B είναι ένας (m – g)x g πίνακας ,
T είναι ένας (m – g)x(m – g) πίνακας ,
C είναι ένας g x(n – m) πίνακας ,
D είναι ένας g x g πίνακας,
E είναι ένας g x(m – g) πίνακας
Ο πίνακας *T* είναι σε κάτω τριγωνική μορφή και έχει μόνο άσσους κατά μήκος του διαγωνίου του ενώ όλοι οι πίνακες είναι αραιοί.
Η παράμετρος g ονομάζεται *κενό*(*gap*) γιατί δείχνει κατά πόσο ο πίνακας *H*′ αποκλίνει από την κάτω τριγωνική μορφή. Για *g* = 0 ο *H*′ είναι κάτω τριγωνικός. Όταν κάνουμε τις αντιμεταθέσεις γραμμών και στηλών για να φέρουμε τον *H* στη κάτω τριγωνική μορφή φροντίζουμε να πετύχουμε όσο το δυνατό μικρότερο *g*.

Πολλαπλασιάζοντας τον παραπάνω πίνακα *H'* (3.9) από αριστερά με τον πίνακα:

$$\begin{bmatrix} \mathrm{I} & 0 \\ -\mathrm{E}\mathrm{T}^{-1} & \mathrm{I} \end{bmatrix} \tag{3.10}$$

Έχουμε:

$$\begin{bmatrix} \mathrm{A} & \mathrm{B} & \mathrm{T} \\ -\mathrm{E}\Gamma^{-1} & \mathrm{A}+C & -ET^{-1}B+D & 0 \end{bmatrix} \tag{3.11}$$

Αν η κωδική λέξη c=*(s, p_1, p_2)* όπου *s* είναι τα bits της πληροφορίας και p_1 και p_2 είναι τα parity-chek bits, τότε p_1 έχει μήκος g και p_2 έχει μήκος m-g.

Συνδέοντας την εξίσωση $c^T * H = 0$, έχουμε:

$$\begin{bmatrix} \mathrm{I} & 0 \\ -\mathrm{E}\mathrm{T}^{-1} & \mathrm{I} \end{bmatrix} \mathrm{H}c^{\mathrm{T}} \tag{3.12}$$

$$\begin{bmatrix} \mathrm{A} & \mathrm{B} & \mathrm{T} \\ -\mathrm{E}\Gamma^{-1} & \mathrm{A}+C & -ET^{-1}B+D & 0 \end{bmatrix} \begin{bmatrix} s \\ p1 \\ p2 \end{bmatrix} = 0 \tag{3.13}$$

Και στη συνέχεια πολλαπλασιάζοντας ουσιαστικά τον πίνακα επί διάνυσμα έχουμε τις εξής δυο βασικές εξισώσεις:

$$[As^T] = \mathrm{T}\mathrm{y}^{\mathrm{T}} \tag{3.14}$$

$$(-ET^{-1}A+C)s^T + (-ET^{-1}\mathrm{B}+\mathrm{D})\mathrm{p}_1^{\ T} = 0 \tag{3.15}$$

Θέτουμε $\Phi := -ET^{-1}B + D$ και στη συνέχεια απλοποιείται και η σχέση (3.15) σε:

$$p_1^T = -\Phi^{-1}(-ET^{-1}A + C)\mathrm{s}^{\mathrm{T}} \qquad (3.16)$$

Εφόσον ο πίνακας $-\Phi^{-1}(-ET^{-1}A + C)$ έχει διαστάσεις *gx(n-m)* και έχει προϋπολογιστεί off-line, έτσι η πολυπλοκότητα για τον υπολογισμό του p_1, εκτιμάται με *O(g*(n-m)).*

Η πολυπλοκότητα αυτή μπορεί να μειωθεί περαιτέρω , με τη διαδικασία που φαίνεται στον πίνακα παρακάτω. Αντί να προ-υπολογίσουμε τον πίνακα: $-\Phi^{-1}(-ET^{-1}A + C)$ και στη συνέχεια να εκτελέσουμε τον πολλαπλασιασμό με το s^T , θα μπορούσε το p_1 να υπολογιστεί, σπάζοντας τον υπολογισμό σε αρκετά μικρότερα βήματα, καθένα από τα οποία μπορεί να υπολογιστεί με πολύ χαμηλή πολυπλοκότητα.
Αρχικά προσδιορίζουμε το As^T με πολυπλοκότητα O(n) , αφού ο πίνακας *A* είναι αραιός. Έπειτα πολλαπλασιάζουμε το αποτέλεσμα με T^{-1}. Ισχύ ότι το $T^{-1}[\mathrm{As}^{\mathrm{T}}] = y^T$ είναι ισοδύναμο με το σύστημα $[As^T] = \mathrm{Ty}^{\mathrm{T}}$, άρα η διαδικασία αυτή μπορεί να πραγματοποιηθεί με πολυπλοκότητα O(n). Τα υπόλοιπα βήματα μπορούν να πραγματοποιηθούν απ' ευθείας με πολυπλοκότητα O(n) εκτός από το τελευταίο βήμα που πραγματοποιείται με πολυπλοκότητα $O(g^2)$. Άρα η συνολική πολυπλοκότητα, για τον υπολογισμό του p_1 θα είναι $O(n+g^2)$.

Αποδοτικός υπολογισμός του ***p_1***

- $A*s^T$:*Πολλαπλασιασμός με έναν αραιό πίνακα*
- $\mathrm{T}^{-1} * [\mathrm{A} * s^T]$: $\mathrm{T}^{-1} * [\mathrm{A} * s^T] = y^T \leftrightarrow [\mathrm{A}*\mathrm{s}^{\mathrm{T}}] = \mathrm{T}*\mathrm{y}^{\mathrm{T}}$
- $-E*[\mathrm{T}^{-1}*\mathrm{A}*\mathrm{s}^{\mathrm{T}}]$:*Πολλαπλασιασμός με έναν αραιό πίνακα*
- $C*\mathrm{s}^{\mathrm{T}}$: *Πολλαπλασιασμός με έναν αραιό πίνακά*
- $-E*[\mathrm{T}^{-1}*\mathrm{A}*\mathrm{s}^{\mathrm{T}}] + [\mathrm{C}*\mathrm{s}^{\mathrm{T}}]$: *Πρόσθεση*
- $-\varphi*[-\mathrm{E}*\mathrm{T}^{-1}*\mathrm{A}*\mathrm{s}^{\mathrm{T}} + C*s^T]$: *Πολλαπλασιασμός με έναν πυκνό gxg πίνακα*

Επίσης από την εξίσωση (3.13) θα έχουμε:

$$p_2^T = -T^{-1} * [A * s^T + B * p_1^T]$$

Με παρόμοιο τρόπο (όπως για το p_1), ο υπολογισμός του p_2 γίνεται σε μικρά σταδιακά βήματα. Η πολυπλοκότητα που απιστείτε για τον υπολογισμό του p_2 είναι O(n).

Αποδοτικός υπολογισμός του ***p_2***

- $A * s^T$: *Πολλαπλασιασμός με έναν αραιό πίνακα*
- $B * p_1^T$:*Πολλαπλασιασμός με έναν αραιό πίνακα*
- $[A * s^T] + [B * P_1^T]$: *Πρόσθεση*
- $-T^{-1}[A * s^T + B * p_1^T]$: $T^{-1}[A * s^T + B * p_1^T] \leftrightarrow [A * s^T + B * p_1^T] = T * y^T$

Περιγραφή της συνολικής διαδικασίας κωδικοποίησης

Στο στάδιο της *προ-επεξεργασίας***:** η είσοδος είναι πίνακας ελέγχου ισοτιμίας *H*, όπου η ορίζουσα του δεν είναι μηδέν. Η έξοδος είναι ο ίδιος ο πίνακας *H* στη μορφή (3.11), τέτοιος ώστε και στον πίνακα: $-ET^{-1}B + D$ η ορίζουσα επίσης δεν είναι μηδέν.

- *Τριγωνοποίηση* **:** Πραγματοποιούμε αντιμεταθέσεις γραμμών και στηλών ώστε να φέρουμε τον πίνακα *H* σε προσεγγιστική , κάτω τριγωνική μορφή, με όσο το δυνατό μικρότερο *g*.
- *Έλεγχος του rank* **:** Με τη χρήση της απαλοιφής *Gauss* , φέρουμε τον προσεγγιστικά , κάτω τριγωνικό πίνακα στη μορφή της σχέσης (3.11), ώστε να ελέγξουμε αν ο πίνακας $-ET^{-1}B + D$ είναι *full rank*. Όταν λέμε *full rank* εννοούμε ότι η ορίζουσα του πίνακα δεν είναι μηδέν. Αν δεν είναι *full rank* , εκτελούμε επιπλέον αντιμεταθέσεις στηλών ώστε να πετύχουμε αυτή την ιδιότητα.

Στο στάδιο της *κωδικοποίησης:* η είσοδός είναι ο πίνακας ελέγχου ισοτιμίας *H*, η ακολουθία πληροφορίας *s*, κωδική λέξη *c=[s, p_1, p_2],* τέτοια ώστε να ύσχει: $c^T * H = 0$.

- Υπολογισμός του p_1
- Υπολογισμός του p_2

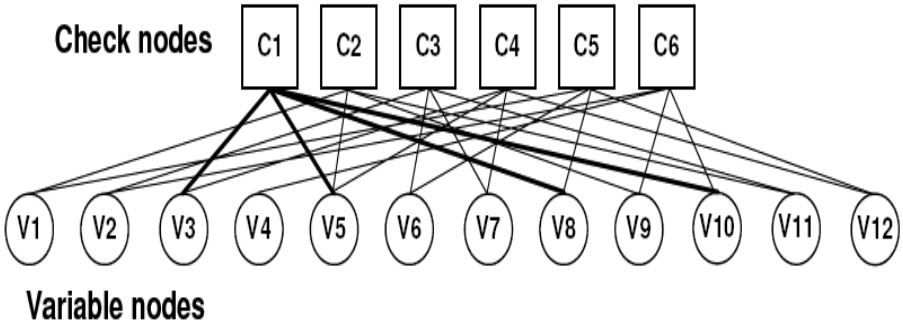
ΚΕΦΑΛΑΙΟ 4
Αλγόριθμοι Αποκωδικοποίησης

4.1 Αλγόριθμοι Message-Passing

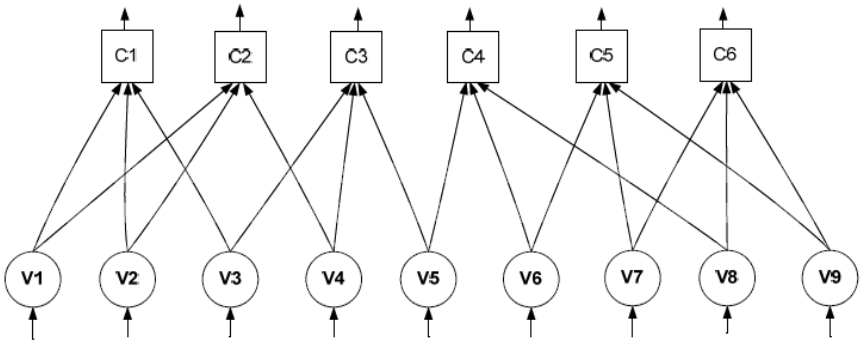
Μια οικογένεια αλγορίθμων που χρησιμοποιείται για να αποκωδικοποιήσει τους LDPC κώδικες είναι οι αλγόριθμοι ανταλλαγής μηνυμάτων (*message – passing).* Η λειτουργία τους πραγματοποιείται με διαδοχική ανταλλαγή μηνυμάτων μεταξύ των δύο ειδών κόμβων του αποκωδικοποιητή, των κόμβων ελέγχου(*Cn*) και των κόμβων μεταβλητών(*Vn*). Με αυτό τον τρόπο ο αποκωδικοποιητής, έχοντας μια αρχική εκτίμηση από το κανάλι για τα σύμβολα του σήματος που λαμβάνονται και εκτελώντας κάποιους υπολογισμούς σε αυτούς τους κόμβους, βελτιώνει την εκτίμησή του με κριτήριο να ικανοποιηθούν όλες οι εξισώσεις ελέγχου ισοτιμίας. Συνεπώς, οι αλγόριθμοι αποκωδικοποίησης LDPC κωδίκων είναι επαναληπτικοί αλγόριθμοι οι οποίοι εκτελώντας διαδοχικά κάποιες πράξεις στους κόμβους προσπαθούν να ανακαλύψουν την απεσταλμένη λέξη, δηλαδή να συγκλίνουν μετά από έναν αριθμό επαναλήψεων σε έγκυρες κωδικές λέξεις. Η message-passing αποκωδικοποίηση δεν αφορά συγκεκριμένα μόνο τους LDPC κώδικες, αλλά οποιονδήποτε κώδικα που προκύπτει ως αποτέλεσμα αλυσιδωτής σύνδεσης επιμέρους υποκωδίκων.

Αν υποθέσουμε ότι κάποια από τα ψηφία-σύμβολα μιας κωδικής λέξης, καθώς αυτή διαδίδεται μέσω του καναλιού επικοινωνίας, έχουν υποστεί αλλοίωση, εξαιτίας της παρουσίας θορύβου στο κανάλι. Αφού φτάσει η αλλοιωμένη κωδική λέξη στον αποκωδικοποιητή, κάθε σύμβολό της μπαίνει ως είσοδος στον αντίστοιχο κόμβο μεταβλητής. Ο κάθε κόμβος μεταβλητής προκειμένου να πάρει μια απόφαση σχετικά με το αν το bit που έλαβε είναι σωστό ή λάθος, «ρωτά» όλους τους γειτονικούς του κόμβους ελέγχου, ποια είναι η «γνώμη» τους για την τιμή του συγκεκριμένου bit. Καθένας από αυτούς τους κόμβους ελέγχου ρωτά, στη συνέχεια, τους υπόλοιπους γειτονικούς του κόμβους μεταβλητής ποια είναι η τρέχουσα τιμή-εκτίμησή τους για τα δικά τους bits (Φάση Α) και στέλνει στον αρχικό κόμβο μεταβλητή μια απάντηση, η οποία είναι συνάρτηση των τιμών αυτών (Φάση Β). Στην ουσία, με αυτό τον τρόπο ελέγχεται η άρτια ή περιττή ισοτιμία των άσσων της ελεγχόμενης ομάδας ψηφίων. Επειδή, όμως, κάθε κόμβος μεταβλητής έχει περισσότερους από έναν γειτονικούς κόμβους ελέγχου, έχει και περισσότερες από μία γνώμες για το αν είναι σωστό το δικό του bit. Πρέπει με κάποιο τρόπο να επεξεργαστεί αυτές τις γνώμες και να βγάλει κάποιο συμπέρασμα για την τιμή του συγκεκριμένου bit. Για παράδειγμα, θα μπορούσε να «πιστέψει» την πλειοψηφία. Η παραπάνω διαδικασία αποτελεί μία επανάληψη του message passing αλγορίθμου. Με κάθε επιπλέον επανάληψη που γίνεται, αυξάνεται η πιθανότητα να είναι σωστή η εκτίμηση των κόμβων μεταβλητής για τις τιμές των ψηφίων της κωδικής λέξης.

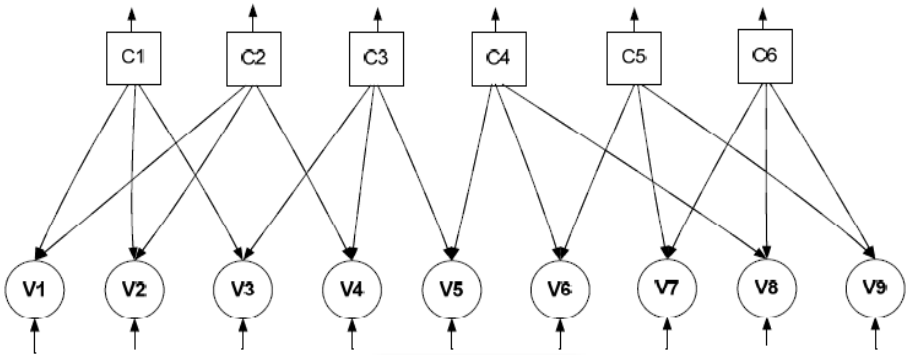
$$H = \begin{array}{c} \begin{matrix} V1 & V2 & V3 & V4 & V5 & V6 & V7 & V8 & V9 & V10 & V11 & V12 \end{matrix} \\ \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} C1 \\ C2 \\ C3 \\ C4 \\ C5 \\ C6 \end{matrix}$$



Φάση Α **:** το στάδιο αυτό υλοποιείται στους *check nodes*. Κάθε *check node* απαιτείται να γνωρίζει μόνο τις πιθανότητες *Pij* που αφορούν τα bits που συμμετέχουν σε αυτόν ώστε να κάνει μια εκτίμηση για την τιμή τους με δεδομένο ότι ικανοποιείται η εξίσωση ελέγχου ισοτιμίας την οποία αυτός εκφράζει.



Φάση Β **:** Το στάδιο αυτό υλοποιείται στους *variable nodes.* Ο κάθε κόμβος(*variable node*) χρειάζεται να γνωρίζει μόνο τις πιθανότητες οι οποίες αφορούν τους *check nodes* με τους οποίους αυτούς συνδέεται στο διάγραμμα Tanner και οι οποίες υπολογίζονται από αυτούς στη *Φάση Α*. Οι πιθανότητες αυτές δεν είναι παρά οι εκτιμήσεις των *check nodes* για την τιμή *bj* του bit *j* στο οποίο αντιστοιχεί ο συγκεκριμένος *variable node.* Βάση αυτών και ο κάθε *variable node* μπορεί να κάνει μια συνολικότερη εκτίμηση της τιμής *bj*.



Σχήμα 4.1 : Αποκωδικοποίηση με χρήση του Message-Passing αλγορίθμου δύο φάσεων

Κατά τη μετάδοση δεδομένων, κάποια από τα λαμβανόμενα σύμβολα διαφέρουν από εκείνα που εισήχθησαν στο κανάλι. Στην περίπτωση που έχουμε έναν δυαδικό κώδικα και διαμόρφωση BPSK τα μεταδιδόμενα σύμβολα είναι το +1 (λογικό 1) και το -1 (λογικό 0). Εξαιτίας του θορύβου στο κανάλι επικοινωνίας, το σήμα που λαμβάνει ο αποδιαμορφωτής κατά τη δειγματοληψία δεν είναι ίδιο με το αρχικό, αλλά παίρνει ενδιάμεσες τιμές ή ακόμα και ανεστραμμένες, αν υπάρχει πολύς θόρυβος. Υπάρχουν δύο προσεγγίσεις όσον αφορά τον τρόπο έκφρασης της τιμής κάθε λαμβανομένου συμβόλου, καθώς και των ενδιάμεσων εκτιμήσεων, κατά τη διάρκεια της αποκωδικοποίησης.

Η προφανής, η οποία ονομάζεται *hard decision*, είναι να θεωρήσουμε πως το πρόσημο του λαμβανομένου συμβόλου είναι αρκετό για να κάνουμε μία πρώτη εκτίμηση για το μεταδιδόμενο σύμβολο. Η αποκωδικοποίηση αυτή είναι χαμηλής πολυπλοκότητας, οδηγεί όμως σε σημαντική απώλεια της ληφθείσας πληροφορίας.

Η δεύτερη προσέγγιση, η οποία ονομάζεται *soft decision* χρησιμοποιεί τα περισσότερα από τα δυαδικά ψηφία, τα λεγόμενα soft bits για την αναπαράσταση των λαμβανόμενων συμβόλων και των μηνυμάτων που ανταλλάσσονται μεταξύ των κόμβων ελέγχου και της μεταβλητής. Η ληφθείσα πληροφορία, η οποία προκύπτει από την ακριβή τιμή του λαμβανόμενου

συμβόλου, διατηρείται καθ 'όλη τη διάρκεια της αποκωδικοποίησης και αξιοποιείται κατά τον καλύτερο δυνατό τρόπο. Η αποκωδικοποίηση αυτή επιτυγχάνει αποτελεσματικότερη αποκωδικοποίηση. Το κόστος της, από άποψη υπολογιστικής πολυπλοκότητας, είναι σαφώς μεγαλύτερο από αυτό της hard decision εκτίμησης. Οι αλγόριθμοι *Message-Passing* βασίζονται κυρίως σε soft decision αποκωδικοποίηση.
Μια πολύ σημαντική πλευρά της πληροφορίας που ανταλλάσσεται μεταξύ των κόμβων του γραφήματος Tanner έχει να κάνει με τη λεγόμενη εξωγενή πληροφορία(*extrinsic information*). Η ιδέα της εξωγενούς πληροφορίας βασίζεται στο γεγονός ότι ένας τοπικός message-passing αποκωδικοποιητής *X*, δεν περνά σε ένα γειτονικό του τοπικό αποκωδικοποιητή *Y* πληροφορία την οποία ο *Y* ήδη κατέχει.
Αυτό σημαίνει ότι μόνο εξωγενής πληροφορία περνά από τον *X* στον *Y*, δηλαδή πληροφορία που αφορά τον *Y* , αλλά προέρχεται έξω από αυτόν. Επομένως ο αποκωδικοποιητής *Y* λαμβάνει από τον αποκωδικοποιητή *X* το άθροισμα όλων των μηνυμάτων που φθάνουν στον *X*(από τους γείτονές του) μείον την πληροφορία την οποία έχει ήδη ο *Y*(και την οποία έχει στείλει στον *X*).

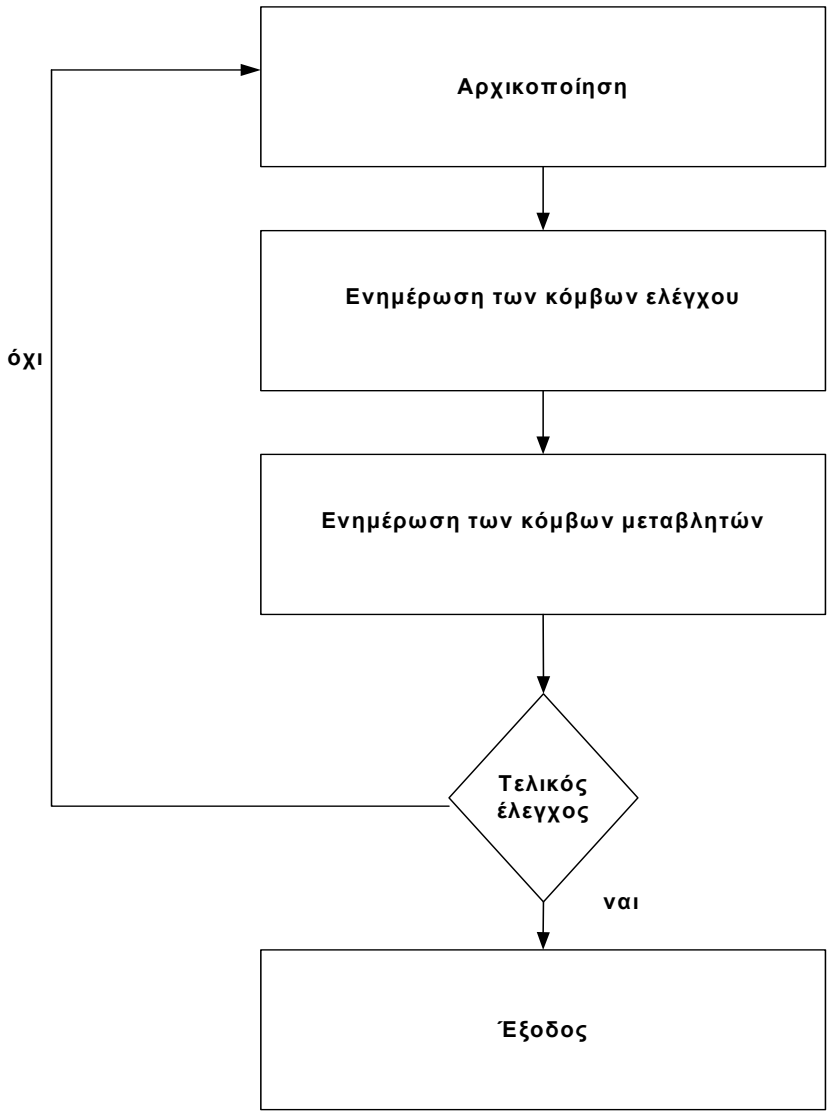
Παρακάτω ακολουθούν τα γενικά βήματα της διαδικασίας της επαναληπτικής, message-passing αποκωδικοποίησης για έναν LDPC κώδικα

1. Αρχικοποίηση των κόμβων μεταβλητών*(Vn*) με τα μηνύματα του καναλιού
2. Επεξεργασία των εισερχόμενων μηνυμάτων από κόμβους μεταβλητών και αποστολή εξωγενούς πληροφορίας στους γειτονικούς κόμβους ελέγχου
3. Επεξεργασία των εισερχόμενων μηνυμάτων από τους κόμβους ελέγχου και αποστολή της εξωγενούς πληροφορίας στους γειτονικούς κόμβους μεταβλητών
4. Υπολογισμός της συνολικής πληροφορίας που φθάνει σε κάθε κόμβο μεταβλητών και εκτίμηση της κωδικής λέξης που έχει αποσταλεί
5. Εξέταση του κριτηρίου ολοκλήρωσης. Αν αυτό δεν ικανοποιείται τότε επιστρέφουμε στο βήμα 2. Διαφορετικά έχουμε ολοκλήρωση της διαδικασίας

Με κάθε επιπλέον επανάληψη που πραγματοποιείται, αυξάνεται η πιθανότητα να είναι σωστή η εκτίμηση των bit nodes για τις τιμές των bits της κωδικής λέξης που έφτασε στον αποκωδικοποιητή. Μετά από την εκτέλεση ενός προκαθορισμένου μέγιστου αριθμού επαναλήψεων, ή μετά την επαλήθευση κάποιου κριτηρίου τερματισμού, ο αποκωδικοποιητής υπολογίζει την εκ των προτέρων πιθανότητα (a posteriori probability, *APP*), τον λόγο πιθανοφάνειας (likelihood ratio, *LR*) ή τον λογάριθμο του λόγου πιθανοφάνειας (loglikelihood ratio, *LLR*) με τη βοήθεια των οποίων λαμβάνονται οι αποφάσεις για τις τιμές των δυαδικών ψηφίων.

Τα πιο συνηθισμένα κριτήρια ολοκλήρωσης στην επαναληπτική αποκωδικοποίηση είναι **:**

- Η ικανοποίηση όλων των εξισώσεων ελέγχου ισοτιμίας $c^T * H = 0$
- Το πέρας του μέγιστου αριθμού επαναλήψεων της επαναληπτικής διαδικασίας αποκωδικοποίησης. Επομένως η διαδικασία αποκωδικοποίησης σταματά αμέσως μόλις βρεθεί μια έγκυρη κωδική λέξη($c^T * H = 0$ **).** Διαφορετικά θα περνάμε στην επόμενη επανάληψη έως ότου ολοκληρωθεί ο μέγιστος αριθμός επαναλήψεων. Μόλις ολοκληρωθεί αυτός ο μέγιστος αριθμός επαναλήψεων , χωρίς να έχει βρεθεί μια έγκυρη κωδική λέξη τότε λέμε ότι ο αλγόριθμος δε συγκλίνει.



Σχήμα 4.2: Διάγραμμα ροής ενός message-passing αποκωδικοποιητή

Οι αλγόριθμοι ανταλλαγής μηνυμάτων ποικίλουν, ανάλογα με τις διεργασίες που εκτελούνται σε κάθε κόμβο και τη δομή των μηνυμάτων που αποστέλλονται ανάμεσα σε αυτούς. Ανάλογα με τις απαιτήσεις της εφαρμογής επιλέγεται συνήθως ο κατάλληλος αλγόριθμος, σε σχέση με τις απαιτήσεις της υλοποίησης του, την πολυπλοκότητα που επιβάλλει στις πράξεις σε σχέση με την απόδοση που επιτυγχάνει. Παρακάτω παρουσιάζονται δυο βασικοί εκπρόσωποι των Message-Passing αλγορίθμων, ο Sum – Product (SP) και ο Min – Sum (MS) αλγόριθμος.

4.1.1 Αλγόριθμος Sum-Product

Ο αλγόριθμος Sum – Product (SPA) εισήχθη από τον ίδιο το Gallager , μαζί με την εργασία του με την οποία εισήγαγε τους LDPC κώδικες το 1962. Ο αλγόριθμος Sum – Product είναι ένας soft decision αλγόριθμος , που σημαίνει ότι ο αποκωδικοποιητής χρησιμοποιεί απ' ευθείας τη λαμβανόμενη ακολουθία bits από το κανάλι χωρίς να μεσολαβεί ανιχνευτής μεταξύ του αποδιαμορφωτή και του αποκωδικοποιητή. Τα μηνύματα που διακινούνται μεταξύ των κόμβων του γραφήματος Tanner είναι πιθανότητες ή λόγοι πιθανοτήτων (γνωστά ως LLRs- log–Likelihood Ratio) σχετικά με τις τιμές που λαμβάνει τα bits κάθε κόμβος μεταβλητών. Να σημειώσουμε ότι ο αλγόριθμος Sum – Product είναι γνωστός και ως αλγόριθμος Belief – Propagation (BPA).

Ακολουθούν τα βήματα της διαδικασίας του αλγορίθμου Sum-Product:

1. Αρχικοποίηση

Αρχικά υπολογίζονται τα μηνύματα στους κόμβους μεταβλητών και αποστέλλονται στους κόμβους ελέγχου

$$Q_{vc}^{0} = \log \frac{p(\mathrm{x}=0 \mid \mathrm{y_n})}{p(\mathrm{x}=1 \mid \mathrm{y_n})} = LLR_v \qquad (4.1)$$

Όπου η τιμή y_n είναι η τιμή του λαμβανόμενου συμβόλου.

2. Ενημέρωση των κόμβων ελέγχου

Έπειτα υπολογίζονται τα μηνύματα Rncv στους κόμβους ελέγχου και αποστέλλονται στους κόμβους μεταβλητής, κατά τη n-στη επανάληψη του αλγορίθμου

$$R_{cv}^{n} = \prod_{v' \in S_v^c(:\mathrm{v})} \mathrm{sign}(\mathrm{Q}_{v'c}^{n-1}) \cdot \Phi\Big(\sum_{v' \in S_v^c(:\mathrm{v})} \Phi(\mathrm{Q}_{v'c}^{n-1}) \Big) \qquad (4.2)$$

Όπου ο όρος $v' \in S_v^c(:\mathrm{v})$ συμβολίζει το σύνολο των κόμβων μεταβλητών, εκτός του v, με τους οποίους συνδέεται ο κόμβος ελέγχου c.
Η συνάρτηση Φ(x) είναι μια σύνθετη συνάρτηση η οποία δίνεται από τη σχέση:

$$\Phi(\mathrm{x}) = -\log\left(\tanh\left(\frac{|\,\mathrm{x}\,|}{2}\right)\right) \tag{4.3}$$

3. Ενημέρωση των κόμβων μεταβλητών

Στη συνέχεια, κατά την n-στη επανάληψη, υπολογίζονται τα μηνύματα Q_{vc}^n στους κόμβους μεταβλητών n και αποστέλλονται στους κόμβους ελέγχου c

$$Q_{vc}^n = LLR_v + \sum_{c'\in S_c^v(:\mathrm{c})} R_{c'v}^n$$

Ο όρος $c' \in S_c^v(:\mathrm{c})$ συμβολίζει το σύνολο των κόμβων ελέγχου, εκτός του c με τους οποίους συνδέεται ο κόμβος μεταβλητών v.

4. Εκτίμηση της πιο πιθανής τιμής

Τέλος, ο κάθε κόμβος μεταβλητών, ο οποίος αντιστοιχεί σε ένα bit του σήματος εισόδου στον αποκωδικοποιητή, εξάγει μια εκτίμηση σύμφωνα με τη σχέση:

$$Q_v = \sum_{c'\in S_c^v} R_{c'v}^n + LLR_v \tag{4.4}$$

Βάσει της τιμής της ποσότητας Qv παίρνεται μία απόφαση:

$$b_v = 1, \mathrm{Q_v} \le 0$$

ή

$$b_v = 0, \mathrm{Q_v} \ge 0$$

Αντίστοιχη απόφαση λαμβάνεται για όλα τα bits της κωδικής λέξης. Αν η κωδική λέξη που προκύπτει είναι έγκυρη ο αλγόριθμος τερματίζει και η συγκεκριμένη κωδική λέξη αποτελεί την αποκωδικοποιημένη πληροφορία.

Διαφορετικά, επαναλαμβάνεται η διαδικασία από το στάδιο του υπολογισμού των μηνυμάτων των κόμβων ελέγχου.

Ένα πρόβλημα που παρουσιάζει ο αλγόριθμος Sum – Product έχει να κάνει με την παρουσία της υπερβολικής , της αντίστροφης υπερβολικής εφαπτομένης και του γινομένου που εμφανίζονται στην εξίσωση ενημέρωσης των κόμβων ελέγχου. Η παρουσία αυτών των δύο συναρτήσεων και του γινομένου δημιουργεί κύριο προβλήμα αύξησης της πολυπλοκότητας αφού η υλοποίηση αυτών των δύο συναρτήσεων είναι σχετικά δύσκολη κυρίως με τη χρήση υλικού(hardware) αλλά και λογισμικού(software).

4.1.2 Αλγόριθμος Min-Sum

Ο αλγόριθμος Min-Sum, είναι βασικά ο Sum-Product αλγόριθμος, αλλά με μια βασική διαφορά: μειώνει την υπολογιστική πολυπλοκότητα του συγκεκριμένου αλγορίθμου έχοντας αντικαταστήσει την πολύπλοκη εξίσωση του αλγορίθμου

$$\Phi(\mathrm{x}) = -\log\left(\tanh\left(\frac{|\mathrm{x}|}{2}\right)\right) \tag{4.5}$$

με άλλες ευκολότερες στην υλοποίηση. Ουσιαστικά δηλαδή ο αλγόριθμος αυτός διαφέρει μόνο στο στάδιο της ενημέρωσης των κόμβων ελέγχου, δηλαδή :

$$\begin{aligned} R_{cv}^{n} &= \prod_{v' \in S_v^c(:\mathrm{v})} \mathrm{sign}(\mathrm{Q}_{v'c}^{n-1}) \cdot \Phi\Big(\sum_{v' \in S_v^c(:\mathrm{v})} \Phi(\mathrm{Q}_{v'c}^{n-1})\Big) = \\ &= \prod_{v' \in S_v^c(:\mathrm{v})} \mathrm{sign}(\mathrm{Q}_{v'c}^{n-1}) \cdot \min_{v' \in S_v^c(:\mathrm{v})}(|\mathrm{Q}_{v'c}^{n-1}|) \end{aligned} \tag{4.6}$$

και τα υπόλοιπα στάδια των δυο αλγορίθμων είναι ίδια.

Μπορούμε να πούμε πως ο αλγόριθμος MS μειώνει σημαντικά την πολυπλοκότητα της αποκωδικοποίησης, λόγω της απουσίας της συνάρτησης Φ(). Παρότι ο Sum-Product είναι θεωρητικά βέλτιστος, ο Min-Sum είναι αρκετά απλούστερος και έχει μεγαλύτερο πρακτικό ενδιαφέρον στα πλαίσια μιας ρεαλιστικής εφαρμογής.

ΚΕΦΑΛΑΙΟ 5
Προτεινόμενος Κωδικοποιητής LDPC

5.1. Προδιαγραφές και λειτουργικές απαιτήσεις του standard

Η παρόν εργασία έχει βασιστεί πάνω σε ένα συγκεκριμένο standard στο οποίο περιγράφονται αναλυτικά οι προδιαγραφές και οι απαιτήσεις που θα πρέπει να τηρηθούν. Σύμφωνα με το standard λοιπόν, ο στόχος είναι η υλοποίηση ενός κωδικοποιητή για Low-Density-Parity-Check Codes που μπορεί να υποστηρίζει ρυθμό απόδοσης (*Throughput rate) 1 Gbps,* δεδομένου ότι :

- Το code rate που θα πρέπει να υποστηρίζει ο κώδικας είναι: *R*=1/2, *R*=2/3, *R*=5/6
- Η μορφές των πινάκων βάσεις (compact matrices)είναι συγκεκριμένες και δίνονται παρακάτω.

Οι παράμετροι είναι:

- Τα εισερχόμενα bits πληροφορίας, *s*
- Αριθμός των bits της κωδικής λέξης, *c*
- Αριθμός των parity-check bits*, p, c-s*
- Ρυθμός κώδικα, *R, s/c*
- Πίνακας ελέγχου ισοτιμίας *H,* έχει διάσταση *r x t* όπου (*r* γραμμές και *t* στήλες) και αποτελείται από δεξιά κυκλικά ολισθημένο, κατά τον αντίστοιχο συντελεστή ολίσθησης, μοναδιαίο υπο-πίνακα μεγέθους *z x z*
- Διάσταση του υπο-πίνακα είναι *z x z***,** ορίζεται : *z =c/t*
- Στην περίπτωση που ο συντελεστής ολίσθησης είναι αρνητικός, τότε ο υπο-πίνακας-αντικαταστάτης είναι ένας μηδενικός τετραγωνικός πίνακας μεγέθους z
- Στην περίπτωση που ο συντελεστής ολίσθησης είναι 0, τότε ο υπο-πίνακας-αντικαταστάτης είναι ένας μοναδιαίος τετραγωνικός πίνακας μεγέθους z
- Στην περίπτωση που ο συντελεστής ολίσθησης είναι κάποιος θετικός αριθμός, τότε ο υπο-πίνακας-αντικαταστάτης είναι μοναδιαίος, δεξιά κυκλικά ολισθημένος πίνακας, κατά τον αντίστοιχο συντελεστή ολίσθησης πίνακα μεγέθους z
- Η προκύπτουσα κωδική λέξη θα πρέπει να ικανοποιεί την εξής εξίσωση: $c^T * H = 0$

Αν παρατηρήσουμε προσεκτικά τις μορφές των πινάκων βάσης που μας δίνει το standard, θα δούμε ότι οι πίνακες ελέγχου ισοτιμίας *H* που θα προκύψουν σε κάθε περίπτωση στη συνέχεια από την αντικατάσταση με τους τετραγωνικούς υπο-πίνακες *z*, σύμφωνα πάντα με τους κανόνες που περιγράφονται από standard, βρίσκονται ήδη σε κάτω τριγωνική μορφή – μια ιδιότητα που είναι απαραίτητη για τον πίνακα ελέγχου ισοτιμίας, ώστε να γίνει η σωστή διαδικασία κωδικοποίησης.

Αυτό φαίνεται ξεκάθαρα ακόμα και πριν γίνει η αντικατάσταση, ότι στην πάνω δεξιά γωνία υπάρχουν μια σειρά από μηδενικά διαγώνια. Αυτά τα μηδενικά λοιπόν, σύμφωνα με τους κανόνες αντικατάστασης θα αντικατασταθούν με μοναδιαίους τετραγωνικούς υπο-πίνακες διάστασης *z*, που σημαίνει ότι ακριβώς σε αυτό το σημείο θα προκύψει μια σειρά από άσους διαγώνια, καθώς και οι αρνητικοί αριθμοί που βρίσκονται στη πάνω δεξιά γωνία, θα αντικατασταθούν με μηδενικούς πίνακες επίσης διάστασης *z*, δηλαδή θα προκύψει η κάτω τριγωνική μορφή.

Case-1

Ο πρώτος compact πίνακας αποτελείται από 12 γραμμές (*r*), 24 στήλες (*t*) και έχει code rate *R*=1/2. Η κωδική λέξη που θα προκύψει από τη διαδικασία κωδικοποίησης θα πρέπει να αποτελείτε από 336 bits. Ο υπο-πίνακας-αντικαταστατής (*z*) έχει διάσταση: 14x14.

$$\begin{bmatrix}
-1 & -1 & -1 & 6 & -1 & -1 & 9 & 6 & -1 & -1 & 2 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 \\
9 & & & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & & & & 0 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & & & & & 0 \; 0 \\
1 & 0 & 11 & -1 & -1 & -1 & 3 & -1 & -1 & 0 & -1 & -1 & -1 & 4 & 8 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Case-2

Ο δεύτερος compact πίνακας αποτελείται από 12 γραμμές (*r*), 24 στήλες (*t*) και έχει code rate *R*=1/2. Η κωδική λέξη που θα προκύψει από τη διαδικασία κωδικοποίησης θα πρέπει να αποτελείτε από 1920 bits. Ο υπο-πίνακας-αντικαταστατής (*z*) έχει διάσταση: 80x80.

$$\begin{bmatrix}
27 & -1 & -1 & -1 & 55 & 19 & -1 & 30 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
16 & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & & & & 0 & -1 \\
35 & & & & & & & & & & & & & & & & & & & & & & & 0 \; 0 \\
-1 & 46 & 28 & -1 & -1 & -1 & 38 & -1 & -1 & -1 & 8 & -1 & 10 & 58 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Case-3

Ο τρίτος compact πίνακας αποτελείται από 12 γραμμές (*r*), 24 στήλες (*t*) και έχει code rate *R*=1/2. Η κωδική λέξη που θα προκύψει από τη διαδικασία κωδικοποίησης θα πρέπει να αποτελείτε από 8640 bits. Ο υπο-πίνακας-αντικαταστάτης (*z*) έχει διάσταση: 360x360

$$\begin{bmatrix}
-1 & 34 & -1 & 95 & -1 & 279 & -1 & -1 & -1 & -1 & 248 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
51 & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
163 & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 \\
46 & & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & & & & 0 & 0 \\
-1 & -1 & 312 & -1 & -1 & -1 & 100 & -1 & -1 & 144 & -1 & 307 & 33 & 166 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Case-4

Ο τέταρτος compact πίνακας αποτελείται από 8 γραμμές (*r*), 24 στήλες (*t*) και έχει code rate *R*=2/3. Η κωδική λέξη που θα προκύψει από τη διαδικασία κωδικοποίησης θα πρέπει να αποτελείτε από 1440 bits. Ο υπο-πίνακας-αντικαταστάτης (*z*) έχει διάσταση: 60x60.

$$\begin{bmatrix}
49 & -1 & -1 & 21 & 31 & -1 & 57 & -1 & -1 & 19 & -1 & 29 & 2 & -1 & 19 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 \\
53 & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 \\
55 & & & & & & & & & & & & & & & & & & & & & 0 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & & & & & 0 & -1 \\
48 & & & & & & & & & & & & & & & & & & & & & & & 0 \; 0 \\
-1 & 24 & 16 & -1 & -1 & 0 & 53 & -1 & -1 & 41 & -1 & 38 & 51 & -1 & 58 & -1 & 59 & 8 & -1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Case-5

Ο πέμπτος compact πίνακας αποτελείται από 8 γραμμές (*r*), 24 στήλες (*t*) και έχει code rate *R*=2/3. Η κωδική λέξη που θα προκύψει από τη διαδικασία κωδικοποίησης θα πρέπει να αποτελείτε από 6480 bits. Ο υπο-πίνακας-αντικαταστατής (*z*) έχει διάσταση: 270x270.

$$\begin{bmatrix}
78 & -1 & -1 & 167 & 237 & -1 & 3 & -1 & 266 & -1 & -1 & 102 & 153 & -1 & -1 & 212 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & \\
92 & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & & \\
144 & & & & & & & & & & & & & & & & & & 0 & -1 & -1 & & & \\
-1 & & & & & & & & & & & & & & & & & & 0 & -1 & & & & \\
-1 & & & & & & & & & & & & & & & & & & 0 & 0 & & & & \\
196 & -1 & 187 & -1 & 73 & -1 & 80 & -1 & 139 & -1 & 57 & -1 & -1 & 236 & 267 & -1 & 62 & 256 & -1 & -1 & -1 & -1 & -1 & 0
\end{bmatrix}$$

Case-6

Ο τέταρτος compact πίνακας αποτελείται από 4 γραμμές (*r*), 24 στήλες (*t*) και έχει code rate *R*=5/6. Η κωδική λέξη που θα προκύψει από τη διαδικασία κωδικοποίησης θα πρέπει να αποτελείτε από 1152 bits. Ο υπο-πίνακας-αντικαταστατής (*z*) έχει διάσταση: 48x48.

$$\begin{bmatrix}
-1 & 13 & 32 & 47 & 41 & 24 & -1 & 25 & 22 & 40 & 1 & 31 & 8 & 15 & 20 & 15 & 42 & 30 & 13 & 3 & -1 & 0 & -1 & -1 \\
25 & & & & & & & & & & & & & & & & & & & & & & 0 & -1 \\
35 & & & & & & & & & & & & & & & & & & & & & & 0 & 0 \\
9 & 32 & 6 & 22 & 26 & 31 & 9 & 8 & 22 & 32 & 40 & 4 & 18 & 40 & 36 & -1 & -1 & 23 & 31 & 41 & 39 & 20 & -1 & 0
\end{bmatrix}$$

Case-7

Ο έβδομος compact πίνακας αποτελείται από 4 γραμμές (*r*), 24 στήλες (*t*) και έχει code rate *R*=5/6. Η κωδική λέξη που θα προκύψει από τη διαδικασία κωδικοποίησης θα πρέπει να αποτελείτε από 5184 bits. Ο υπο-πίνακας-αντικαταστατής (*z*) έχει διάσταση: 216x216.

$$\begin{bmatrix}
-1 & 47 & 146 & 203 & 184 & 112 & -1 & 116 & 103 & 181 & 3 & 140 & 38 & 68 & 91 & 70 & 191 & 138 & 62 & 14 & -1 & 0 & -1 & -1 \\
117 & & & & & & & & & & & & & & & & & & & & & & 0 & -1 \\
153 & & & & & & & & & & & & & & & & & & & & & & 0 & 0 \\
44 & 147 & 27 & 83 & 118 & 130 & 41 & 38 & 100 & 146 & 183 & 19 & 85 & 180 & 163 & -1 & -1 & 106 & 140 & 185 & 177 & 94 & -1 & 0 \\
 & & & & & & & & & & & & & & & & & & & & & & &
\end{bmatrix}$$

5.2 Περιγραφή του προτεινόμενου αλγόριθμου κωδικοποίησης

Ένα σημαντικό χαρακτηριστικό των κωδικών LDPC είναι ότι ο αλγόριθμος κωδικοποίησης, και κατά συνέπεια και το κύκλωμα που τον υλοποιεί, εξαρτώνται άμεσα από τη δομή του πίνακα ελέγχου ισοτιμίας. Καθώς οι πίνακες βάσεις για τον συγκεκριμένο κωδικοποιητή μας δίνονται έτοιμες από το standard, ο στόχος είναι ο αλγόριθμος και στη συνέχεια η υλοποίηση του να αξιοποιεί και να εκμεταλλεύεται τόσο τη δομή του πίνακα ελέγχου ισοτιμίας *H*, όσο και το πιο σημαντικό πλεονέκτημα του, το οποίο είναι η αραιότητα καθώς ο πίνακας έχει πολλά μηδενικά στοιχεία.

Επίσης, θα πρέπει να έχουμε πάντα υπόψη μας ότι ο στόχος είναι να υλοποιήσουμε έναν LDPC κωδικοποιητή με *Throughput* τουλάχιστον *1 Gbps*, και για τον σκοπό αυτό ακολούθησαν τα εξής βήματα:

- Έγινε μελέτη για τον κατάλληλο αλγόριθμο κωδικοποίησης για τους κώδικες χαμηλής –πυκνότητας –ελέγχου-ισοτιμίας, με σκοπό να μειωθεί η πολυπλοκότητα κωδικοποίησης
- Πραγματοποιήθηκε περιγραφή του αλγορίθμου σε Matlab
- Σύμφωνα με το μοντέλο του Matlab ακολουθήθηκε μια πλήρως παραμετροποιήσιμη ανάπτυξη του κώδικα σε γλώσσα vhdl
- Υλοποιήθηκε μια πλήρως παραλληλοποιήσιμη αρχιτεκτονική για των κωδικοποιητή, ώστε να πετύχουμε τον σκοπό μας- ρυθμό απόδοσης 1 Gbps.

Η διαδικασία της κωδικοποίησης(encoding) αναφέρεται στην παραγωγή των κωδικών λέξεων του κώδικα , από τα μηνύματα πληροφορίας(information messages).

Σε όλους τους τρόπους κωδικοποίησης , η διαδικασία της κωδικοποίησης πραγματοποιείται γενικά σε δύο στάδια**:**

- Ένα στάδιο προ-επεξεργασίας (preprocessing step) το οποίο πραγματοποιείται μόνο μια φορά και περιλαμβάνει offline υπολογισμούς ώστε να φέρουμε τους πίνακες στη μορφή που θέλουμε. Το στάδιο αυτό δεν έχει καμία συσχέτιση και «επαφή» με τα δεδομένα που θα κωδικοποιηθούν
- Το πραγματικό στάδιο της κωδικοποίησης το οποίο περιλαμβάνει online υπολογισμούς για την παραγωγή των κωδικών λέξεων. Το στάδιο αυτό έχει άμεση «επαφή» με τα δεδομένα που πρόκειται να κωδικοποιηθούν.

Η υλοποίηση πραγματοποιήθηκε σύμφωνα με τον διάσημο αλγόριθμο των T. Richardson και R. Urbanke, ο οποίος μας δίνει την πιο αποδοτική διαδικασία κωδικοποίησης η οποία μπορεί να προσεγγίσει πολύ χαμηλή πολυπλοκότητα.

Προ-επεξεργασία του πίνακα ***H*** **–**offline, υπολογισμοί

Ας πάρουμε ως παράδειγμα το case-1, το οποίο μας δίνει το standard. Σε αυτή τη περίπτωση έχουμε τον εξής πίνακα βάσης:

$$\begin{bmatrix}
-1 & -1 & -16 & -1 & -196 & -1 & -12 & -1 & -10 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & 0 & -1 & -1 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & 0 & -1 & -1 & -1 \\
9 & & & & & & & & & & & & & & & & 0 & -1 & -1 \\
-1 & & & & & & & & & & & & & & & & & 0 & -1 \\
-1 & & & & & & & & & & & & & & & & & & 0\ 0 \\
1 & 0\ 11 & -1 & -1 & -13 & -1 & -10 & -1 & -1 & -148 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -10
\end{bmatrix}$$

- Ο συγκεκριμένος compact πίνακας αποτελείτε από 12 γραμμές, *r*, 24 στήλες, *t.*
- Έχει code rate *R=1/2.* Πράγματι αν κάνουμε την πράξη *r/t***,** βγαίνει το code rate 1/2.
- Η κωδική λέξη, *c*, που θα προκύψει από τη διαδικασία κωδικοποίησης θα πρέπει να αποτελείται από 336 ψηφία.
- Ο υπο-πίνακας αντικαταστάτης (z) έχει διάσταση: 14x14, το οποίο προέκυψε από την πράξη *c/t*, δηλαδή αριθμός της κωδικής λέξης/αριθμός των στηλών, 336/24=14
- Γνωρίζοντας την διάσταση του υπο-πίνακα, μπορούμε εύκολα να υπολογίσουμε την διάσταση του πίνακα ελέγχου ισοτιμίας *H,* πολλαπλασιάζοντας τη διάσταση γραμμών επί την διάσταση του υπο-πίνακα, και την διάσταση στηλών επίσης επί την διάσταση του υπο-πίνακα. Έτσι ο expanded πίνακας ελέγχου ισοτιμίας Η, θα έχει διαστάσεις *r*z x t*z*, δηλαδή θα έχει 168 γραμμές και 336 στήλες *($H_{168x336}$).*
- Γνωρίζοντας επίσης ότι το *code rate* είναι 1/2, μπορούμε να ξέρουμε τον αριθμό των bits της εισερχόμενης πληροφορίας. Δηλαδή είναι ο αριθμός των γραμμών του πίνακα ισοτιμίας, 168 * 1, άρα information bits, *s* =168. Στη περίπτωση που έχουμε code rate 2/3, θα πολλαπλασιάζαμε το πλήθος των γραμμών*2, και αντίστοιχα αν είχαμε code rate 5/6, γίνεται πολλαπλασιασμός του πλήθους των γραμμών του πίνακα ισοτιμίας * 5.
- Από τη στιγμή που ξέρουμε τον αριθμό των information bits, πλέον είναι εύκολο να υπολογίσουμε και τα parity bits, αφαιρώντας από τον αριθμό των ψηφίων της κωδικής λέξης *c,* τα ψηφία πληροφορίας, s. Αρά έχουμε 336-168=168 είναι τα parity bits *p.*
- Ένα άλλο βασικό στοιχείο είναι να ορίσουμε την παράμετρο *gap*, *g*, το οποίο δείχνει κατά πόσο ο πίνακας *H* αποκλίνει από την κάτω τριγωνική μορφή. Σύμφωνα με τον αλγόριθμο, το ***g*** ισούται με τη διάσταση του υπο-πίνακα- αντικαταστάτη (*z*), και στη προκειμένη περίπτωση είναι 14 bits.

- Σύμφωνα με τους κανόνες κωδικοποίησης, ο πίνακας ελέγχου ισοτιμίας *H* προκύπτει ως εξής:

Κάθε θετικός αριθμός στη θέση (i,j) του πίνακα βάσης αντικαθιστάται με ένα zxz μοναδιαίο δεξιά κυκλικά ολισθημένο πίνακα, κατά τον αντίστοιχο συντελεστή ολίσθησης.
Κάθε αρνητικός αριθμός στη θέση (i,j) του πίνακα βάσης αντικαθιστάται με ένα zxz μηδενικό τετραγωνικό πίνακα.
Κάθε «0» στη θέση (i,j) του πίνακα βάσης αντικαθιστάται με ένα zxz μοναδιαίο τετραγωνικό πίνακα

Παράδειγμα:
Αν υποθέσουμε ότι το *z=2,* τότε η αντικατάσταση θα γίνει με έναν τετραγωνικό πίνακα 2x2 ,

Πίνακας βάσης Πίνακας ελέγχου ισοτιμίας

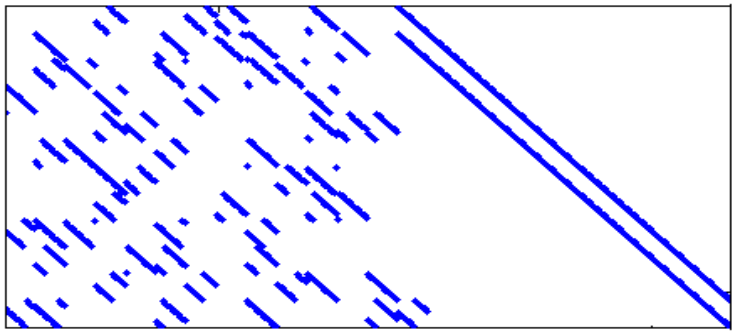
$$\begin{bmatrix} -1 & 0 & -1 & -1 \\ 2 & -1 & 0 & -1 \\ 1 & 3 & -1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0\,0 & 1\,0 & 0\,0 & 0\,0 \\ 0\,0 & 0\,1 & 0\,0 & 0\,0 \\ 1\,0 & 0\,0 & 1\,0 & 0\,0 \\ 0\,1 & 0\,0 & 0\,1 & 0\,0 \\ 0\,1 & 0\,1 & 0\,0 & 1\,0 \\ 1\,0 & 1\,0 & 0\,0 & 0\,1 \end{bmatrix}$$

Ο συντελεστής 3 αντικαθιστάτε με τον πίνακα : $\begin{bmatrix} 0\,1 \\ 1\,0 \end{bmatrix}$

για τον λόγο ότι πραγματοποιείτε δεξιά κυκλική ολίσθηση του μοναδιαίου

πίνακα : $\begin{bmatrix} 1\,0 \\ 0\,1 \end{bmatrix} \rightarrow 1 \rightarrow \begin{bmatrix} 0\,1 \\ 1\,0 \end{bmatrix} \rightarrow 2 \rightarrow \begin{bmatrix} 1\,0 \\ 0\,1 \end{bmatrix} \rightarrow 3 \rightarrow \begin{bmatrix} 0\,1 \\ 1\,0 \end{bmatrix}$

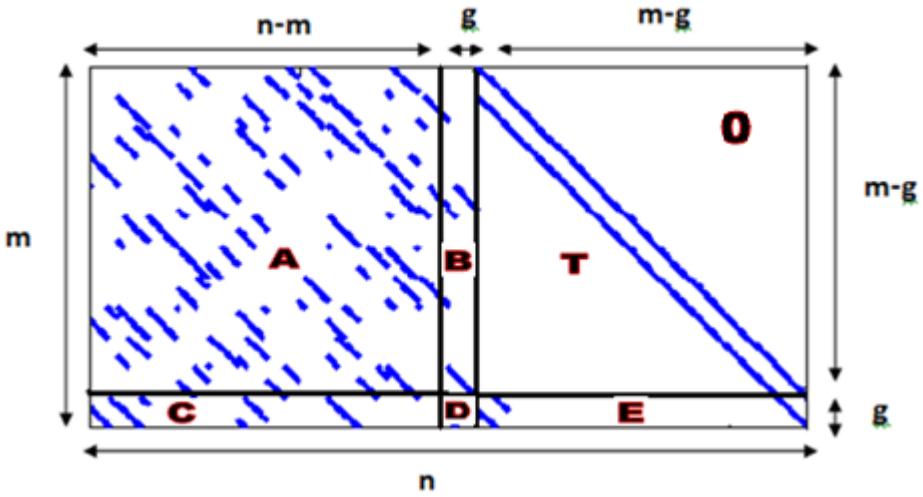
Με αυτόν τον τρόπο λοιπόν ο πίνακας ελέγχου ισοτιμίας *H* , θα έχει διάσταση 168x336 και θα έχει την εξής μορφή:



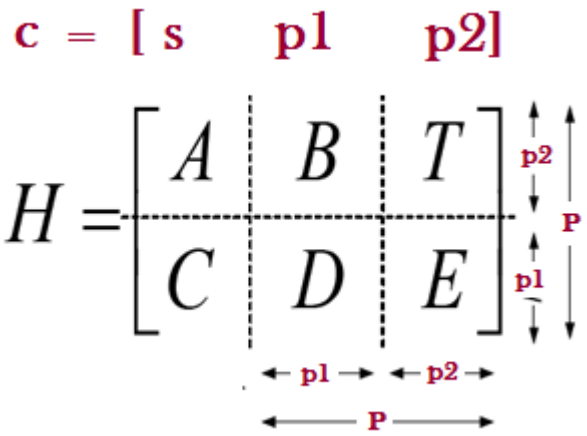
Σχήμα 5.1: Πίνακας ελέγχου ισοτιμίας $H_{168x336}$ όπου με μπλε γραμμές αναπαριστάνονται οι άσοι και στα κενά είναι τα μηδενικά

Με την βοήθεια του σχήματος που μας εκτύπωσε το matlab μπορούμε να παρατηρήσουμε το πλεονέκτημα που παρουσιάζει ο πίνακας ελέγχου ισοτιμίας *H* έχοντας πολύ μεγάλο αριθμό μηδενικών στοιχείων. Η συγκεκριμένη ιδιότητα θα μας βοηθήσει πολύ στη διαδικασία γρήγορης και αποδοτικής κωδικοποίησης. Επίσης φαίνεται ξεκάθαρα και η κάτω τριγωνική μορφή του πίνακα *H*. Αυτό σημαίνει ότι βρισκόμαστε σε πολύ καλό στάδιο και δεν χρειάζεται να γίνουν αντιμεταθέσεις γραμμών ή στηλών ώστε να πετύχουμε την συγκεκριμένη μορφή. Καθώς και η παράμετρος *gap* είναι πολύ μικρή, κάτι που συμβαίνει και στον αλγόριθμο R&U.

Υποθέτοντας ότι το *m* είναι γραμμές του πίνακα ελέγχου ισοτιμίας *H* και *n* είναι οι στήλες έχουμε:



Ακλουθώντας τον αλγόριθμο R&U και έχοντας έναν έτοιμο πίνακα ελέγχου ισοτιμίας *H* μπορούμε κάλλιστα να υπολογίσουμε τα parity bits:



Σχήμα 5.2: Δομή του πίνακα H βάση τον αλγόριθμο R&U

Το πρώτο διάνυσμα των parity bits *p1* προκύπτει σπάζοντας τις πράξεις σε μικρότερα σταδιακά βήματα για τη μείωση της πολυπλοκότητας του αλγορίθμου :

- Βρίσκουμε τον αντίστροφο πίνακα του πίνακα *T* => προκύπτει ο πίνακας *invT*
- Πολλαπλασιάζουμε τον πίνακα *invT* επί τον πίνακα *E* => προκύπτει ο πίνακας *ET1*
- Πολλαπλασιάζουμε *ET1* επί τον πίνακα *B* και προσθέτουμε τον πίνακα *D* => προκύπτει ο πίνακας *phi*

Σε αυτό το σημείο θα πρέπει να υπολογίσουμε την ορίζουσα του πίνακα *phi* και να σιγουρευτούμε ότι η ορίζουσα του είναι διάφορη του μηδενός. Είναι μια σημαντική και απαραίτητη ιδιότητα που ορίζει ο αλγόριθμος R&U, για να μπορεί να πραγματοποιηθεί η συνέχεια της διαδικασίας κωδικοποίησης.

- Πολλαπλασιάζουμε τον πίνακα *ET1* επί τον πίνακα *A* και προσθέτουμε τον πίνακα *C* => προκύπτει ο πίνακας *extra*
- Πολλαπλασιάζουμε τους πίνακες *phi* επί *extra* => προκύπτει ο πίνακας *x*
- Τέλος βρίσκουμε το *p1*, πολλαπλασιάζοντας το διάνυσμα των information bits επί των ανάστροφο πίνακα *x* => προκύπτει το διάνυσμα *p1*
- Μετονομάζουμε τον πίνακα *A* σε *y*
- Μετονομάζουμε τον πίνακα *B* σε *z*

Αντίστοιχα προκύπτει και το διάνυσμα *p2*, δεδομένου ότι βρέθηκαν τα parity bits *p1,* έτσι έχουμε:

- Πολλαπλασιάζουμε τον πίνακα *y* επί το ανάστροφο διάνυσμα των information bits=> προκύπτει το διάνυσμα *e*
- Πολλαπλασιάζουμε τον πίνακα *z* επί το ανάστροφο διάνυσμα των *p1*, που υπολογίσαμε προηγούμενος => προκύπτει το διάνυσμα *f*

- Προσθέτουμε τα δυο διανύσματα *e* και *f* => προκύπτει το διάνυσμα *h*
- Τέλος βρίσκουμε το *p2* πολλαπλασιάζοντας τον πίνακα *invT* επί τον πίνακα *h* => προκύπτει το διάνυσμα *p2*

Έχουμε βρει το *p1* και το *p2* , το μόνο που μένει είναι να συνθέσουμε την κωδική λέξη και να ελέγξουμε αν είναι έγκυρη.

codeword =[information bits parity1 parity2]

c=[s p1 p2]

Όπως αναφέρθηκε και στα προηγούμενα κεφάλαια, οι κώδικες LDPC είναι γραμμικοί κώδικες δομής και η κωδικοποίηση τέτοιων κωδίκων ακολουθεί την σχέση:

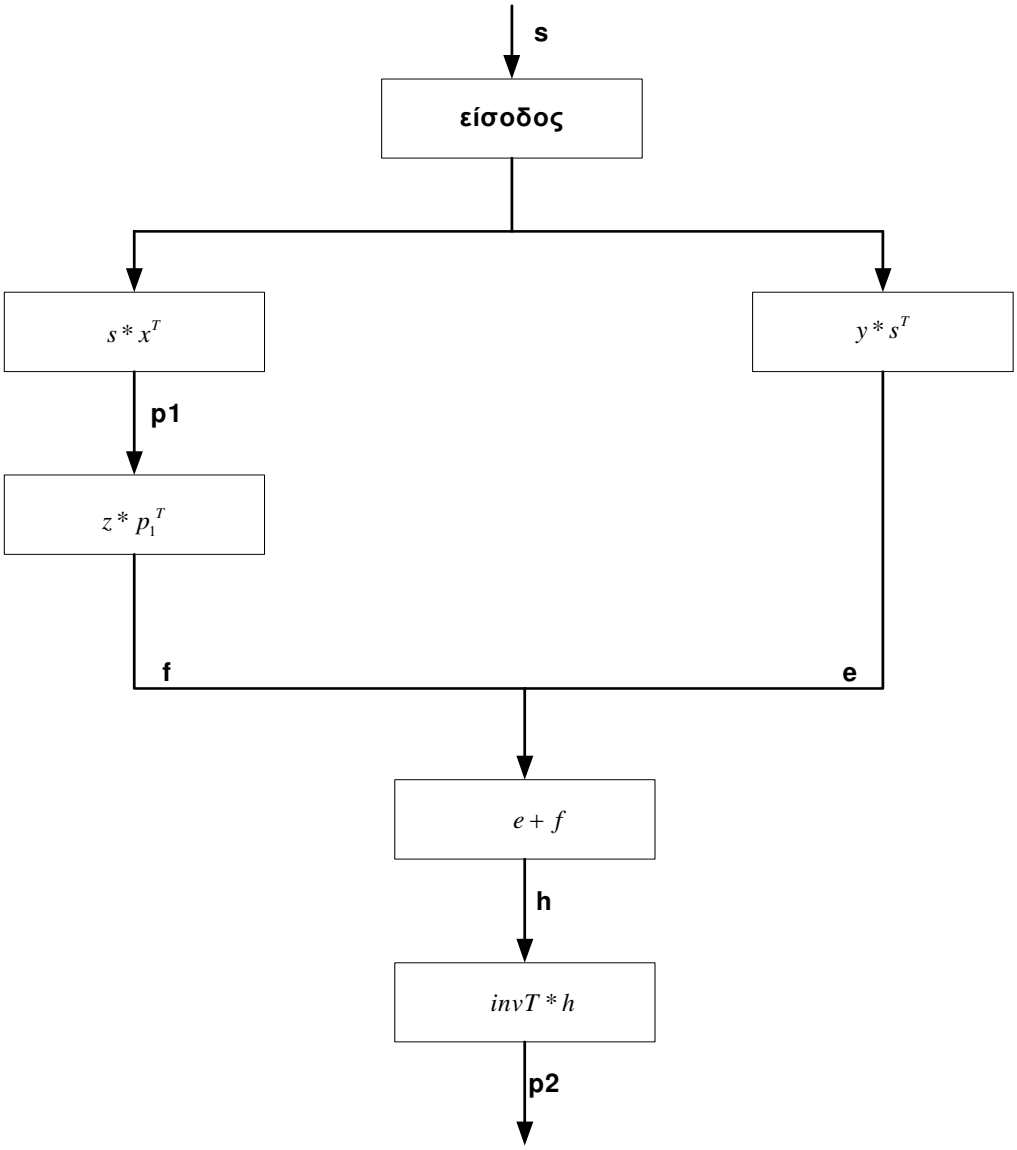
$$c^T * H = 0$$

όπου *H,* είναι πίνακας ελέγχου ισοτιμίας (parity check matrix) και *c* είναι η κωδική λέξη.
Η κωδική λέξη που βρέθηκε είναι έγκυρη μόνο και μόνο αν πολλαπλασιάζοντας την κωδική λέξη επί τον ανάστροφο πίνακα ελέγχου ισοτιμίας το αποτέλεσμα είναι μηδέν.

Online υπολογισμοί

Από την περιγραφή του αλγορίθμου και την off-line προ-επεξεργασία , οδηγούμαστε στο συμπέρασμα ότι για να γίνει κωδικοποίηση βάση των υπολογισμών που πραγματοποιηθήκαν θα σημειώσουμε τους on-line υπολογισμούς που χρειάζονται για την εύρεση των parity bits. Δηλαδή τις συγκεκριμένες πράξεις που στη συνέχεια θα υλοποιηθούν στη ανάπτυξή του κώδικα σε vhdl:

- $p_1 = s * x^T$
- $e = y * s^T$
- $f = z * p_1^T$
- $h = e + f$
- $p_2 = invT * h$



Σχήμα 5.3 : Εκτέλεση online υπολογισμών για την εύρεση τον parity bits

Παρατηρώντας τις παραπάνω πράξεις, μπορούμε να βγάλουμε συμπέρασμα ότι καταρχάς το στάδιο της προ-επεξεργασίας είναι ασύλληπτα χρήσιμο όσον αφορά την πολυπλοκότητα του αλγορίθμου. Πλέον έχοντας τα συγκεκριμένα βήματα είναι πάρα πολύ εύκολο να υπολογίσουμε τα ψηφία ισοτιμίας καθώς και να υλοποιήσουμε έναν κωδικοποιητή σε hardware, αφού ουσιαστικά έχουμε να κάνουμε με υλοποίηση πολλαπλασιαστών. Συγκεκριμένα εκτελούνται πολλαπλασιασμοί πίνακα επί διάνυσμα στις τέσσερις περιπτώσεις και μια πρόσθεση μεταξύ των δυο διανυσμάτων, για αυτόν τον λόγο στην υλοποίηση χρησιμοποιούνται μόνο λογικές πράξεις (AND και XOR). Ένα άλλο πλεονέκτημα είναι ότι οι πίνακες που χρησιμοποιούνται είναι αραιοί, κάτι που διευκολύνει ακόμα περισσότερα την εκτέλεση των πολλαπλασιαστών και την υλοποίηση του encoder. Επίσης, αν προσέξουμε την σειρά με την οποία εκτελούνται οι τελικές πράξεις για την εύρεση της κωδικής λέξης , θα δούμε ότι κάποιες από αυτές δεν εξαρτούνται μια από την άλλη, κάτι το οποίο μας δίνει την δυνατότητα της παράλληλης υλοποίησης τους.

5.3 Προτεινόμενη παράλληλη Αρχιτεκτονική

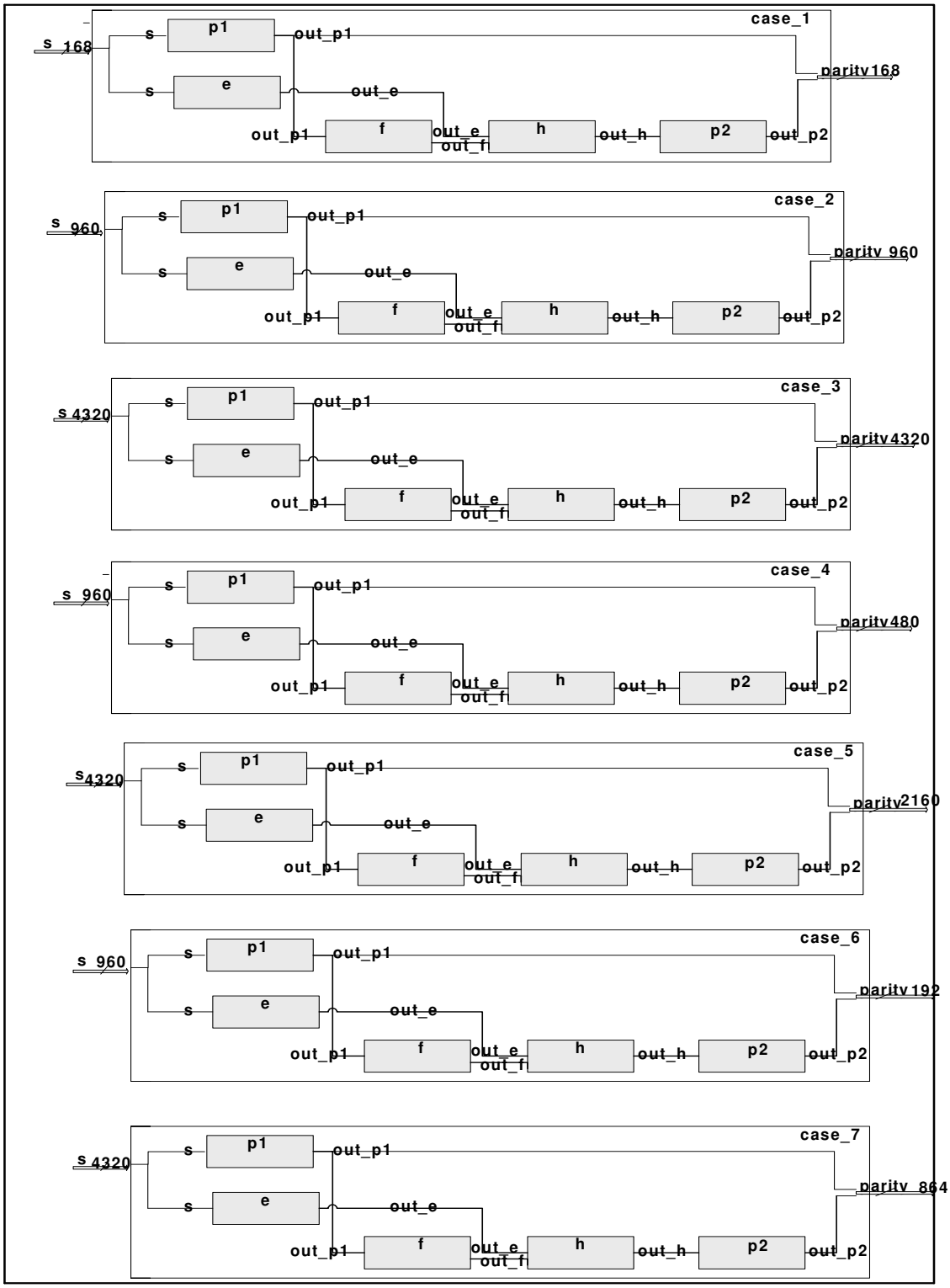
Εκτελώντας τα παραπάνω βήματα, δηλαδή το στάδιο της προ-επεξεργασίας και των on-line υπολογισμών για κάθε compact πίνακα που προτείνει το standard παρατηρούμε ότι τα οι περιπτώσεις 3 , 5 , 7 έχουν ιδιαίτερα αυξημένες απαιτήσεις και ειδικά *το case 3.* Αν το αναλύσουμε, όπως κάναμε και στο παράδειγμα του *case 1,* θα δούμε ότι ο expanded πίνακας, δηλαδή ο πίνακας ελέγχου ισοτιμίας έχει διαστάσεις *4320x8640* bits. Η εισερχόμενη πληροφορία (information bits) αποτελείται από το διάνυσμα των *4320* ψηφίων. Οι διαστάσεις αυτές είναι υπερβολικά μεγάλες για την υλοποίηση ενός κωδικοποιητή με πολύ χαμηλή πολυπλοκότητα τόσο στο λογισμικό, όσο και στο υλικό, ακόμα κι αν η υπολογιστική δύναμη έχει αναπτυχθεί πάρα πολύ. Παρόλα αυτά στις απαιτήσεις του standard, ο στόχος είναι η υλοποίηση ενός κωδικοποιητή για τους κώδικες χαμηλής- πυκνότητας –ελέγχου-ισοτιμίας, ο οποίος επιτυγχάνει ρυθμό απόδοσης *1 Gbps*. Για τον σκοπό αυτό έχει σχεδιαστεί πλήρως παράλληλη αρχιτεκτονική με τεχνική pipelining - τεχνική με την οποία μπορούν να παραλληλιστούν κάποιες επαναληπτικές διαδικασίες με στόχο την επιτάχυνση της εκτέλεσής τους. Η αρχιτεκτονική που ακολουθεί είναι σχεδιασμένη σύμφωνα με τις προδιαγραφές της τρίτης περίπτωσης , για τον προφανή λόγο ότι αν η συγκεκριμένη περίπτωση είναι μέσα στα πλαίσια του αρχικού μας στόχου, αυτό σημαίνει ότι και οι υπόλοιπες περιπτώσεις μπορούν να πετύχουν throughput rate 1 Gbps.

Έχει γίνει υπολογισμός ότι για να πετύχουμε *throughput rate* τουλάχιστον *1 Gbps* με συχνότητα λειτουργίας ρολογιού 100 *MHz*, ο κωδικοποιητής θα κωδικοποιεί 32 bits ανά δείγμα (*block length*).

Η υλοποίηση της βασικής αρχιτεκτονικής του κωδικοποιητή βασίζεται πάνω στη σχεδίαση ανεξάρτητων, κατά τμήμα, μονάδων επεξεργασίας όπου η κάθε μια αναλαμβάνει την υλοποίηση μιας συγκεκριμένης λειτουργίας, αποφεύγοντας έτσι την πολυπλοκότητα σχεδιασμού, έχοντας τη δυνατότητα αλλαγής οποιασδήποτε μονάδας για οποιαδήποτε παράμετρο λειτουργίας.

5.3.1 Data Path Unit

Η μονάδα χειρισμού ή διαδρομής δεδομένων (Data Path Unit) η οποία θεωρείται η «καρδιά» του κωδικοποιητή , στην περίπτωση μας αποτελείται από πολλαπλασιαστές και αθροιστές.



Σχήμα 5.4: Data Path Unit του κωδικοποιητή LDPC

Όπως φαίνεται και στο παραπάνω σχήμα, το Block_DPU αποτελείται από εφτά ανεξάρτητες υπο-μονάδες, όπου η κάθε μια αντιστοιχεί και σε κάθε περίπτωση που μας δίνεται από το standard. Κάθε υπο-μονάδα αποτελείται από τέσσερεις πολλαπλασιαστές (*p1, e , f, p2*) και έναν αθροιστή (*h*). Σαν είσοδο δέχεται ένα διάνυσμα (*s*), το οποίο είναι τα bits πληροφορίας και η έξοδος είναι τα parity bits, όπου και η είσοδος και η έξοδος έχουν συγκεκριμένο μήκος αντίστοιχα για την κάθε περίπτωση. Κάθε πολλαπλασιαστής αποτελείται από λογικές πύλες *AND* και *XOR* και εκτελεί τον παράλληλο πολλαπλασιασμό του πίνακα ελέγχου ισοτιμίας με το διάνυσμα των bits πληροφορίας. Ο κώδικας που έχει γραφτεί σε γλώσσα *vhdl* και περιγράφει τις συγκεκριμένες λειτουργίες είναι πλήρως παραμετρικός, κάτι που δίνει την δυνατότητα σε οποιονδήποτε χρήστη με την βοήθεια των generics που χρησιμοποιούνται, να τον τροποποιήσει κατάλληλα, εισάγοντας τις επιθυμητές τιμές. Ακόμα και έναν δικό του πίνακα ελέγχου ισοτιμίας, αρκεί να πληροί τις κατάλληλες διαστάσεις, καθώς ο κάθε πίνακας ελέγχου ισοτιμίας είναι ενσωματωμένος σε επτά διαφορετικά *packages,* τα οποία χρησιμοποιεί ο αντίστοιχος κώδικας και υλοποιεί την κάθε περίπτωση. Ένα άλλο πλεονέκτημα που μας δίνει ο συγκεκριμένος τρόπος υλοποίησης, είναι ότι δεν χρειάζεται να έχουμε άπειρο αριθμό μνημών, ώστε να αποθηκεύουμε τους πίνακες ελέγχου ισοτιμίας, οι οποίες έχουν τεράστιες διαστάσεις. Με αυτόν τον τρόπο όχι μόνο εξοικονομούμε τον χώρο σε ένα FPGA κατά την υλοποίηση του κωδικοποιητή, αλλά και μειώνουμε την πολυπλοκότητα και το κόστος. Επίσης στο διάγραμμά φαίνετε ξεκάθαρα η παράλληλη εκτέλεση των πολλαπλασιαστών *p1* και *f* , στη συνέχεια το αποτέλεσμα του *p1* γίνεται είσοδος για τον πολλαπλασιαστή *f*. Ο αθροιστής h δέχεται ως είσοδο τα αποτελέσματα του *f* και *e*, τα οποία είναι δυο διανύσματα. Τέλος το αποτέλεσμα του αθροιστή h αποτελεί την είσοδο για τον πολλαπλασιαστή *p2.* Τα συνολικά parity bits είναι η ένωση των αποτελεσμάτων από τους πολλαπλασιαστές *p1* και *p2.* Για την εκτέλεση των συγκεκριμένων υπολογισμών απαιτείται ένας κύκλος ρολογιού (1 *Clock Cycle*).

Παρακάτω δίνεται λεπτομερές σχήμα υλοποίησης της μονάδας DPU και συγκριμένα στον τρόπο πολλαπλασιασμού ενός πίνακα ελέγχου ισοτιμίας με ένα διάνυσμα ψηφίων πληροφορίας. Μετά από πολλές μελέτες και εξομοιώσεις διαπιστώθηκε ότι μια από τις αποδοτικότερες μεθόδους πολλαπλασιασμού ενός πίνακα πολύ μεγάλων διαστάσεων επί ένα διάνυσμα είναι ο τεμαχισμός του πίνακα σε μικρότερους υπο-πίνακες. Αυτός ο τρόπος όχι μόνο διευκολύνει την λειτουργία του πολλαπλασιαστή, αλλά μειώνει σημαντικά την ταχύτητα και τον χρόνο που απαιτείται για την παραγωγή του αποτελέσματος.

Ορίζουμε τις παραμέτρους:

c_all : αριθμός των στηλών του πίνακα

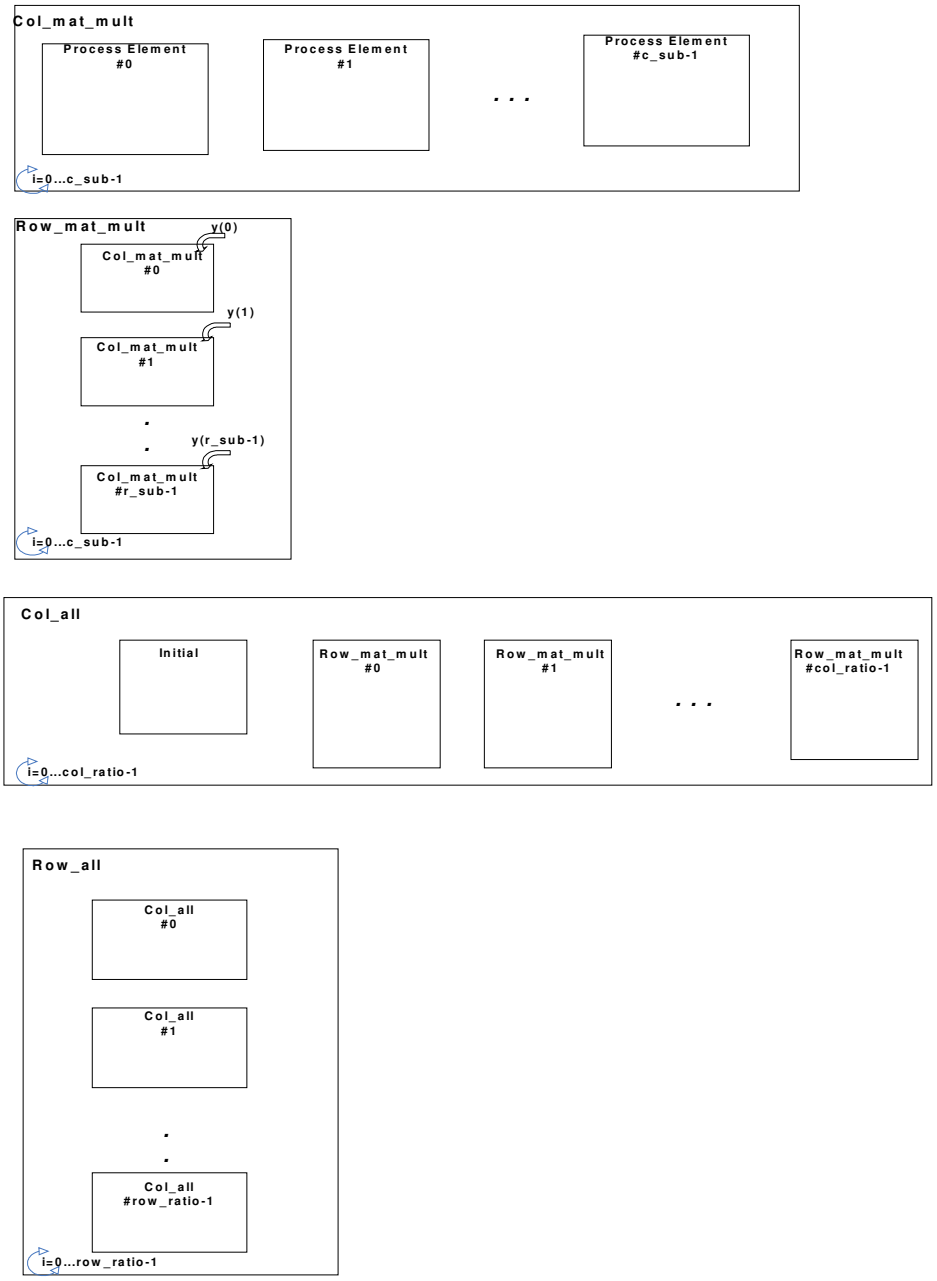
r_all : αριθμός των γραμμών του πίνακα

c_sub : *c_all /PF(Parallelization Factor),* αριθμός των στηλών του υπο-πίνακα

r_sub : *r_all /PF(Parallelization Factor),* αριθμός των γραμμών του υπο-πίνακα

col_ratio : *c_all /c_sub*

row_ratio : *r_all /r_sub*



Σχήμα 5.5 : Αποδοτικός πολλαπλασιασμός πίνακα ελέγχου ισοτιμίας επί διάνυσμα ψηφίων πληροφορίας

5.3.2 Μνήμη Dual Port RAM

Όπως αναφέρθηκε και πιο πάνω, με τον τρόπο λειτουργίας της μονάδας DPU, όπου οι πίνακες ελέγχου ισοτιμίας δεν αποθηκεύονται στις μνήμες αλλά χρησιμοποιούνται απευθείας από τα packages δεν υπάρχει ανάγκη για την ύπαρξη πολλών και μεγάλου μεγέθους μνημών. Η μόνη μνήμη που χρησιμοποιείται είναι η μνήμη BRAM (memory Block Ram) η οποία χρειάζεται για την αποθήκευση των εισερχόμενων bits πληροφορίας. Οι μνήμες BRAM χωρίζονται σε οικογένειες όπως Single Port Ram, Dual Port Ram, FIFO κλπ. Για το συγκεκριμένο project έχει υλοποιηθεί η Dual Port Ram με την βοήθεια του λογισμικού CORE Generator του εργαλείου σχεδίασης ISE της εταιρίας Xilinx. Ο λόγος της επιλογής της συγκεκριμένης μνήμης είναι ότι επιτρέπει ταυτόχρονα την εγγραφή σε μια θέση μνήμης και την ανάγνωση από μια άλλη θέση στον ίδιο παλμό ρολογιού. Αξιοποιούμε αυτό το πλεονέκτημα ώστε να πετύχουμε μια παράλληλη αρχιτεκτονική κατά τη σχεδίαση του κωδικοποιητή LDPC.

Σύμφωνα λοιπόν με τα δεδομένα, οι περιπτώσεις 3, 5 και 7 δέχονται ως είσοδο 4320 bits πληροφορίας. Αυτό σημαίνει ότι για να αποθηκεύσουμε αυτά τα ψηφία θα χρειαστούμε μια Dual Port Ram μνήμη των 512 βαθμίδων, όπου η κάθε μια χωράει 32 bits. Όπως απεικονίζεται και στο διάγραμμα, η μνήμη «χωρίζεται» σε δυο ίσα τμήματα, δηλαδή από την διεύθυνση 0 έως 255 θεωρείτε bank-0 και από 256 έως 511 είναι το bank-1. Η τεχνική που υλοποιείται σε αυτό το σημείο ονομάζετε τεχνική “ping-pong”. Αρχικά από 0 έως 255 είναι οι θέσεις στις οποίες πραγματοποιείται η εγγραφή της εισερχόμενης πληροφορίας των block των 32 bits και από την διεύθυνση 256 έως 511 πραγματοποιείται η ανάγνωση. Στη συνέχεια, στον επόμενο κύκλο ρολογιού από την διεύθυνση 0 έως 255 πραγματοποιείται η ανάγνωση της πληροφορίας και στη διεύθυνση 256 έως 511 πραγματοποιείται η εγγραφή των νέων block των information bits. Ο συγκεκριμένος τρόπος σχεδιασμού της μνήμης βοηθάει στην παράλληλη εγγραφή και ανάγνωση της μνήμης, άρα και στην επιτάχυνση της λειτουργίας του κωδικοποιητή. Έτσι τα εισερχόμενα σήματα χωρίζονται ανά ζεύγη:

- *dina* και *dinb* είναι ζευγάρι σημάτων που φέρνουν τα ψηφία πληροφορίας των 32 bits

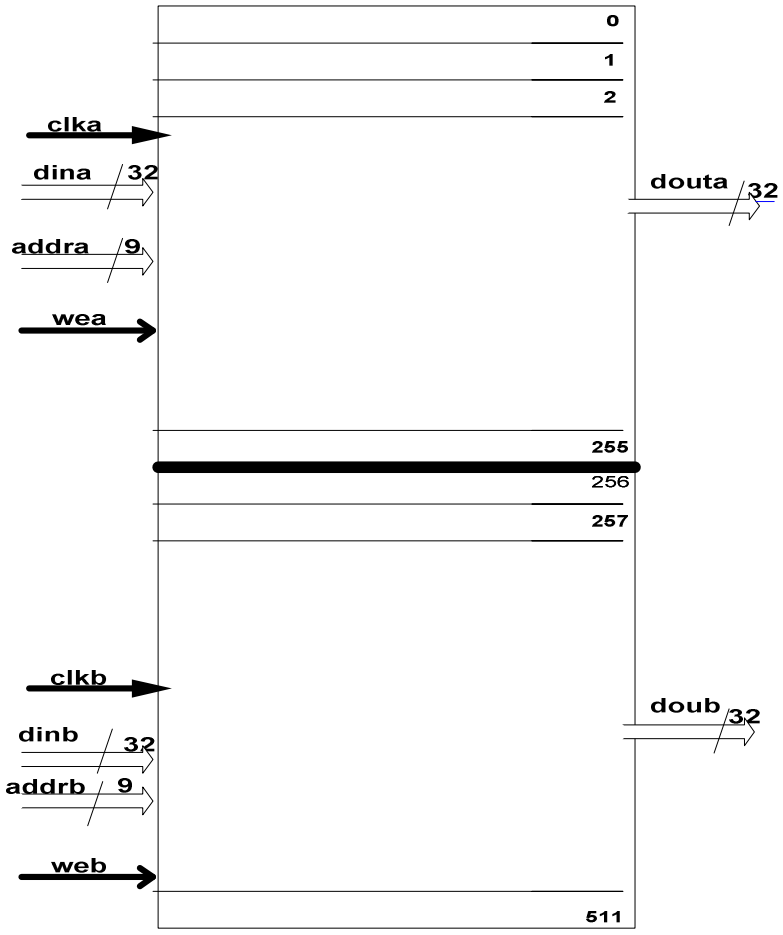
- *addra* και *addrb* είναι ζευγάρι σημάτων που δείχνουν την διεύθυνση, δηλαδή τις συγκεκριμένες θέσεις μνήμης και αποτελείται από 9 bits, λόγω του ότι ισχύει 2^9 =512 θέσεις

- *wea* και *web* είναι ζευγάρι σημάτων για την ενεργοποίηση είτε της λειτουργίας εγγραφής είτε της ανάγνωσης

- *clka* και *clkb* είναι τα ρολόγια της μνήμης, τα οποία συνδέονται σε ένα εξωτερικό ρολόι και ελέγχεται από τη μονάδα ελέγχου

Τα σήματα εξόδου της μνήμης είναι επίσης ένα ζευγάρι σημάτων *douta* και *doutb* των 32 bits το κάθε σήμα.

Σύμφωνα με τον αλγόριθμο κωδικοποίησης μια κωδική λέξη αποτελείται από τα information bits και parity bits. Αυτό σημαίνει ότι η εισερχόμενη πληροφορία θα πρέπει όχι μόνο να κωδικοποιηθεί, αλλά και να εμφανιστεί ανεπηρέαστη στην έξοδο του κωδικοποιητή ώστε να σχηματιστεί η κωδική λέξη. Ένας έξυπνος τρόπος για πετύχουμε αυτό είναι η υλοποίηση ενός ζεύγος πολυπλέκτων, οι οποίοι τοποθετούνται ακριβώς στην έξοδο της μνήμης και με την βοήθεια τους πραγματοποιείται η επιλογή του block των information bits, ο οποίος θα πρέπει να οδηγηθεί απευθείας στην έξοδο του κωδικοποιητή, χωρίς να υποστεί επαναληπτική, προσωρινή αποθήκευση του στους καταχωρητές του συστήματος. Αυτή η μέθοδος βοηθάει να υλοποιήσουμε ακόμα πιο αποδοτική την αρχιτεκτονική του encoder.



*Σχήμα 5.6 : Μνήμη Dual Port RAM 512 Words*32 bits/Word*

5.3.3 Register File

Το Register File είναι ουσιαστικά ένα αρχείο καταχωρητών, το οποίο χρησιμοποιείται για την προσωρινή αποθήκευση των εσωτερικών δεδομένων κατά την γρήγορη εκτέλεση των εντολών του συστήματος. Στην προκειμένη περίπτωση, το αρχείο καταχωρητών που υλοποιείται, αποτελείται από 135 καταχωρητές κάθε ένα των 32 bits. Χρησιμοποιούμε 135 καταχωρητές λόγω του ότι θα πρέπει να αποθηκευτούν το πολύ 4320 bits τα οποία αποτελούν τα ψηφία πληροφορίας. Κατά την λειτουργία του κωδικοποιητή, το Register File αποθηκεύει όχι μόνο τα εισερχόμενα από την μνήμη τα ψηφία πληροφορίας που πρόκειται να κωδικοποιηθούν, αλλά και τα parity bits που εξέρχονται από την μονάδα DPU. Ανάλογα με την περίπτωση, είτε είναι τα ψηφία πληροφορίας είτε τα parity bits, πρόκειται για διανύσματα διαφορετικού μήκους, για αυτό τον λόγο χρησιμοποιείται αντίστοιχο σήμα για την ενεργοποίηση εγγραφής πληροφορίας μόνο σε τόσους καταχωρητές, όσοι χρειάζονται. Το αρχείο καταχωρητών είναι απαραίτητο στοιχείο στη συγκεκριμένη αρχιτεκτονική, επειδή βοηθά στη πραγματοποίηση του παράλληλου σχεδιασμού.
Τα σήματα εισόδου του Register File είναι:

- σήμα ρολογιού που καθορίζει πότε θα ενημερωθεί η αποθηκευμένη τιμή
- σήμα για την ενεργοποίηση εγγραφής πληροφορίας
- σήμα για εκκαθάριση του αρχείου καταχωρητών
- σήμα για την εισερχόμενη πληροφορία, το οποίο υλοποιείται ως ένας πίνακας 135x32 bits

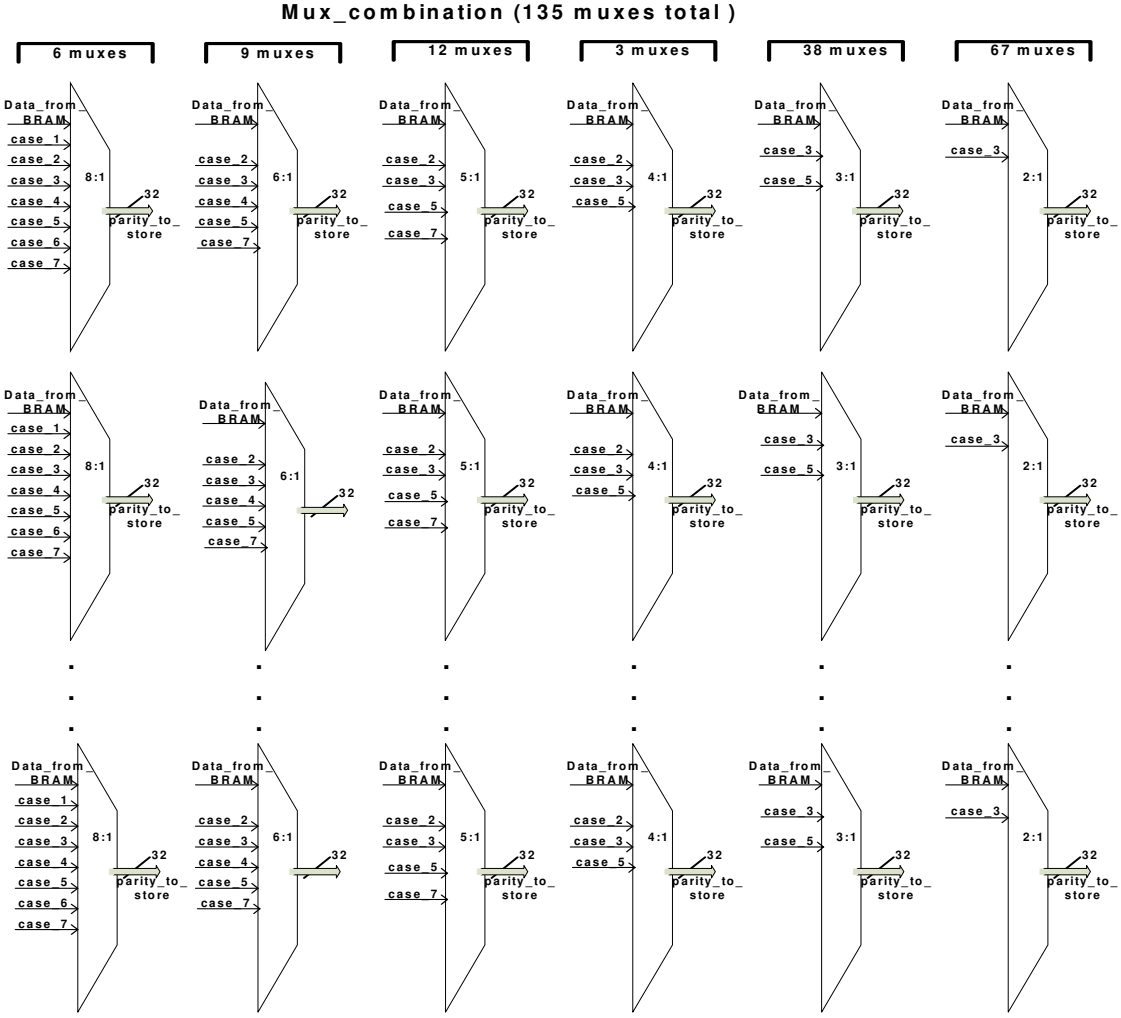
Η έξοδος είναι επίσης ένα σήμα σε μορφή ενός πίνακα 135x32 bits και προσαρμόζεται αντίστοιχα για κάθε περίπτωση που υλοποιείται.

Όπως φαίνεται και στο διάγραμμα παρακάτω που παρουσιάζει συνολικό block του Register File, υπάρχει ένα επίσης σημαντικό σημείο που απαιτεί τη προσοχή κατά την σχεδίαση της αρχιτεκτονικής – είναι οι πολυπλέκτες (*muxes*). Η κύρια εφαρμογή του πολυπλέκτη είναι η επιλογή μίας από τις πολλές πληροφορίες που εφαρμόζονται στις εισόδους του και η μεταφορά της στην έξοδό του. Στη περίπτωση μας υλοποιούνται συνολικά 135 πολυπλέκτες, όπου η κύρια λειτουργία τους είναι η επιλογή μεταξύ των information bits που εξέρχονται από την μνήμη και parity bits που εξέρχονται από την μονάδα DPU, ώστε να οδηγηθούν στο αρχείο καταχωρητών, και για αυτό είναι τοποθετημένοι στην είσοδο του Register File. Για την αποτελεσματική χρήση των πολυπλεκτών, υπολογίσαμε ότι δεν είναι απαραίτητο όλοι οι 135 πολυπλέκτες να είναι είδους 8:1, κάτι το οποίο βαρύνει αρκετά το κύκλωμα. Όταν λέμε πολυπλέκτης είδους 8:1, εννοούμε ότι έχει 8 εισόδους των 32 bits, δηλαδή τα 7 σήματα που εξέρχονται από την μονάδα DPU και αντιπροσωπεύουν τις εφτά περιπτώσεις που υλοποίει και 1 σήμα που εξέρχεται από την μνήμη και φέρνει την 32-αδα των ψηφίων πληροφορίας.

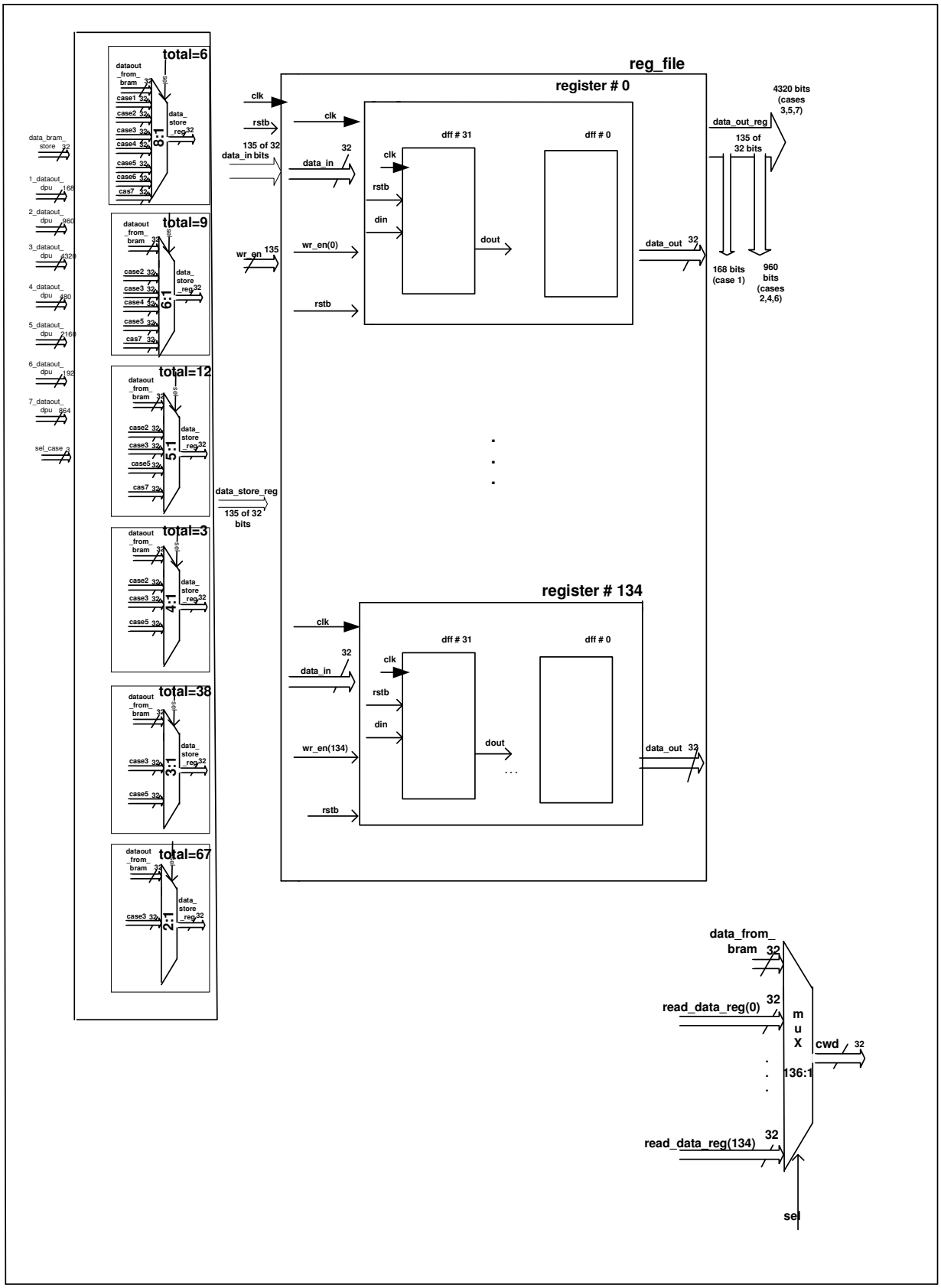
Στο σχήμα που δίνεται πιο κάτω, στην κάτω δεξιά γωνία υπάρχει επίσης ένας πολυπλέκτης του είδους 136:1 ο οποίος δέχεται σαν είσοδο έναν πίνακα 135x32 bits που αντιπροσωπεύουν τα parity bits που εξέρχονται από το αρχείο καταχωρητών και ένα σήμα των 32 bits που αντιπροσωπεύουν τα ψηφία πληροφορίας που εξέρχονται από την μνήμη. Ο συγκεκριμένος πολυπλέκτης αναλαμβάνει την λειτουργία του στο σημείο όταν έχουν βρεθεί τα parity bits και εξάγει με τη σωστή σειρά την συνολική κωδική λέξη: [s p1 p2].

Έχοντας υπόψη τον αριθμό των parity bits και information bits της κάθε περίπτωσης χρειαζόμαστε :

- 6 πολυπλέκτες του είδους 8:1
- 9 πολυπλέκτες του είδους 6:1
- 12 πολυπλέκτες του είδους 5:1
- 3 πολυπλέκτες του είδους 4:1
- 38 πολυπλέκτες του είδους 3:1
- 67 πολυπλέκτες του είδους 2:1



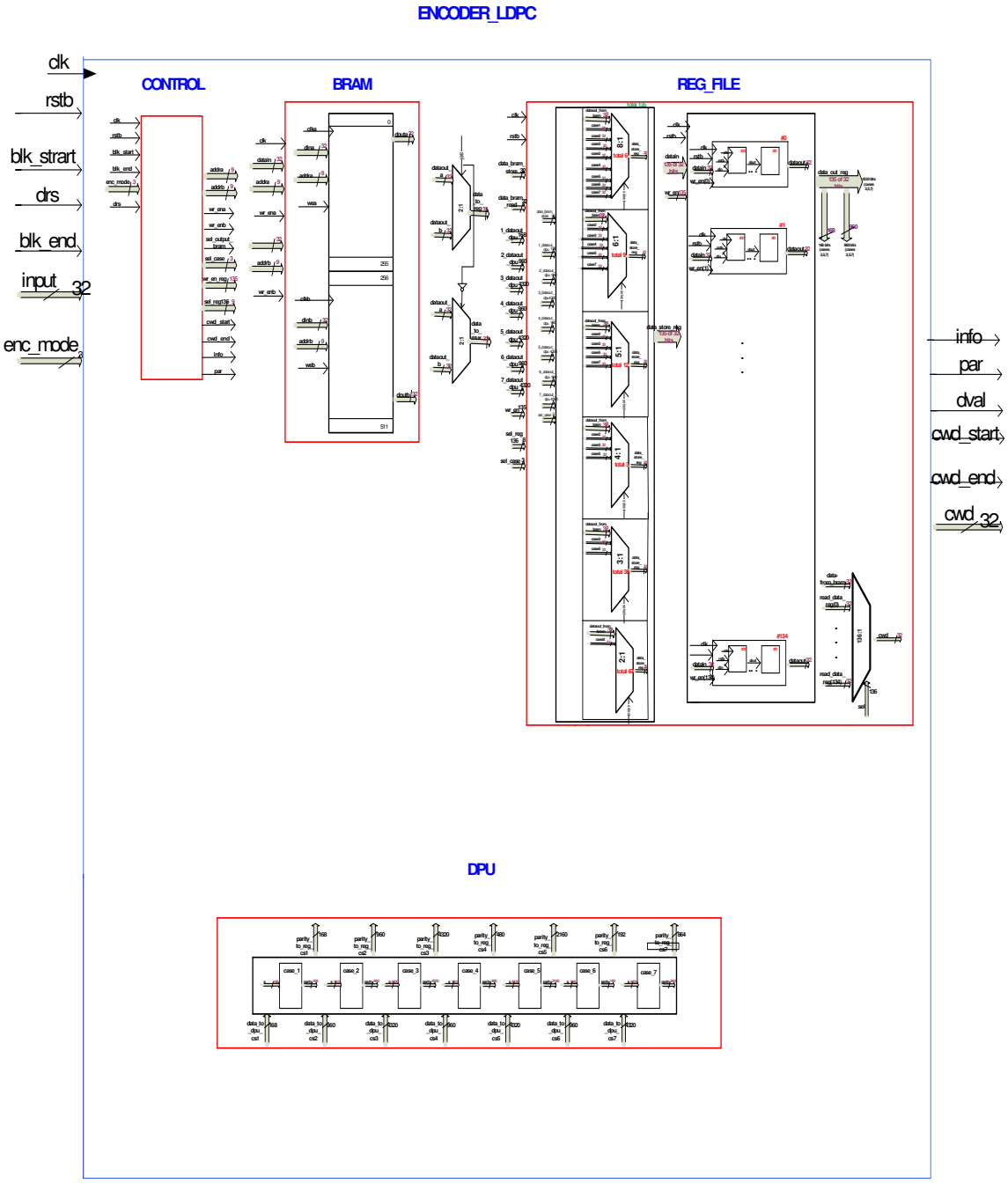
Σχήμα 5.7 : Συνδυασμός των πολυπλεκτών που οδηγούν τα δεδομένα στον Register File



Σχήμα 5.8: Συνολικό Block του Register File

5.3.4 Control Unit

Η Μονάδα Ελέγχου είναι από τα πιο σημαντικά τμήματα της αρχιτεκτονικής, καθώς αναλαμβάνει τον έλεγχο της κάθε μονάδας του συστήματος, και παράγει όλα εκείνα τα σήματα που είναι απαραίτητα για να διαβαστεί και να εκτελεστεί κάθε εντολή. Θα μπορούσαμε να τη χαρακτηρίσουμε ως το «μυαλό» του κυκλώματος μας. Παρακάτω ακολουθεί συνολικό διάγραμμα της προτεινόμενης αρχιτεκτονικής του κωδικοποιητή LDPC.



Σχήμα 5.9 : Αρχιτεκτονική LDPC Encoder

Όπως φαίνεται και στο διάγραμμα, το κύκλωμα του κωδικοποιητή αποτελείται από τέσσερις μονάδες ή τμήματα, δηλαδή τη μνήμη, το αρχείο καταχωρητών, τη μονάδα διαδρομής δεδομένων και την μονάδα ελέγχου. Η μονάδα ελέγχου χειρίζεται τα απαραίτητα σήματα που απαιτούνται όχι μόνο για την σωστή λειτουργία του κωδικοποιητή αλλά και για την αποδοτική του λειτουργία, δηλαδή έχει υλοποιηθεί με τέτοιο τρόπο ώστε σε συνδυασμό με τις υπόλοιπες μονάδες πετυχαίνεται το Throughput του κωδικοποιητή 1 Gbps.

Τα σήματα εισόδου του κωδικοποιητή είναι:

- *clk* **:** ρολόι του συστήματος
- *rstb***:** ασύγχρονο reset
- *blk_start***:** σήμα το οποίο βρίσκεται σε υψηλό επίπεδο τάσης, δηλαδή «1», όταν παρουσιάζεται το πρώτο bit της 32-αδας των ψηφίων πληροφορίας, δηλαδή μας δείχνει την έναρξη του block των information bits
- *blk_end:* σήμα το οποίο βρίσκεται σε υψηλό επίπεδο τάσης, δηλαδή «1», όταν παρουσιάζεται το τελευταίο bit της 32-αδας των ψηφίων πληροφορίας, δηλαδή μας δείχνει το τέλος του block των information bits
- *input:* σήμα το οποίο φέρνει τα 32 bits των ψηφίων πληροφορίας
- *enc_mode:* σήμα το οποίο ορίζει ποια περίπτωση από τις επτά παρούσες πρόκειται να υλοποιηθεί
- *info_new:* σήμα το οποίο βρίσκεται σε υψηλό επίπεδο τάσης, δηλαδή «1», όταν εισέρχεται μια καινούργια 32-αδα των ψηφίων πληροφορίας

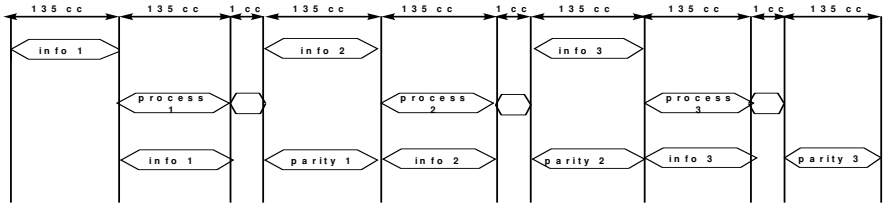
Τα σήματα εξόδου του κωδικοποιητή είναι:

- *info:* σήμα το οποίο βρίσκεται σε υψηλό επίπεδο τάσης, δηλαδή «1» σε όλοι την διάρκεια των information bits
- *par:* σήμα το οποίο βρίσκεται σε υψηλό επίπεδο τάσης, δηλαδή «1» σε όλοι την διάρκεια των parity bits
- *cwd_start:* σήμα το οποίο βρίσκεται σε υψηλό επίπεδο τάσης, δηλαδή «1», όταν παρουσιάζεται το πρώτο bit της κωδικής λέξης, δηλαδή μας δείχνει την έναρξη της κωδικής λέξης
- *cwd_end:* σήμα το οποίο βρίσκεται σε υψηλό επίπεδο τάσης, δηλαδή «1», όταν παρουσιάζεται το τελευταίο bit της κωδικής λέξης, δηλαδή μας δείχνει το τέλος της κωδικής λέξης

- *cwd:* σήμα στο οποίο παρουσιάζεται η κωδική λέξη

Έχοντας αναλύσει τα σήματα εισόδου/εξόδου του κωδικοποιητή, είναι η κατάλληλη στιγμή να περιγράψουμε την συνολική λειτουργία του.

Ας υποθέσουμε ότι εκτελούμε την τρίτη περίπτωση που μας δίνει το standard, ή οποία είναι και η πιο δύσκολη στην υλοποίηση της, λόγω του ότι ο πίνακας ελέγχου ισοτιμίας είναι των διαστάσεων 4320x8640 bits και η εισερχόμενη πληροφορία αποτελείται από το διάνυσμα των 4320 ψηφίων. Στην πρώτη φάση, το σήμα *enc_mode* γίνετε «011», όπου καταλαβαίνουμε αμέσως ότι πρόκειται να κωδικοποιηθεί ένα block των 4320 information bits. Το σήμα *input* φέρνει την πρώτη 32-αδα των bits**.** Τα σήματα info_new και blk_start βρίσκονται στη σε υψηλό επίπεδο τάσης, δηλαδή «1». Το σήμα εισόδου της μνήμης *wea* γίνεται «1», που σημαίνει ότι επιτρέπει την εγγραφή του block στη μνήμη στο bank-0. Τα 4320 bits διαμοιράζονται στις 135 θέσεις μνήμης ανά 32-άδες. Η διαδικασία αυτή πραγματοποιείται σε 135 κύκλους ρολογιού. Έπειτα τα information bits θα πρέπει να οδηγηθούν στο αρχείο καταχωρητών, όπου εκεί δεσμεύονται όλοι οι διαθέσιμοι 135 καταχωρητές. Η αποθήκευση πληροφορίας επίσης απαιτεί 135 κύκλους ρολογιού. Ακριβώς παράλληλα με την αποθήκευση της πληροφορίας στο αρχείο καταχωρητών πραγματοποιείται και η εξαγωγή της στην έξοδο του κωδικοποιητή με την βοήθεια του πολυπλέκτη 136:1, για τον οποίον αναφέραμε πιο πάνω. Στη συνέχεια τα bits πληροφορίας από το αρχείο καταχωρητών δρομολογούνται στη μονάδα DPU και εκτελείτε ο υπολογισμός των parity bits σε 1 κύκλο ρολογιού. Τα υπολογισμένα parity bits αποθηκεύονται στο αρχείο καταχωρητών και αμέσως μετά επίσης εξάγονται στην έξοδο του κωδικοποιητή σχηματίζοντας με αυτόν τον τρόπο μια κωδική λέξη. Παράλληλα το σήμα *web* της μνήμης γίνεται «1» και πραγματοποιείται η εγγραφή του καινούργιου block των 4320 bits.

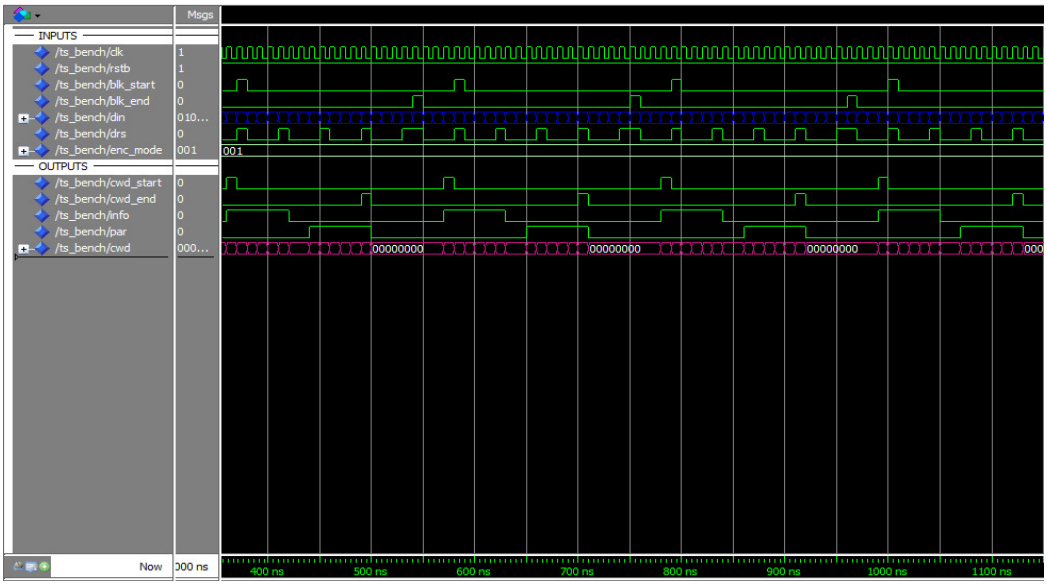


Σχήμα 5.10 : Η συνολική λειτουργία του κωδικοποιητή όταν η εισερχόμενη πληροφορία είναι 4320 bits

ΚΕΦΑΛΑΙΟ 6

Εξομοιώσεις , Αποτελέσματα και Συμπεράσματα

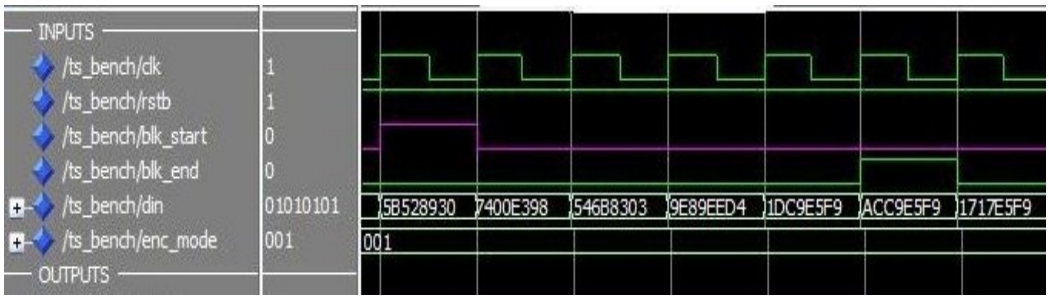
Για τις δοκιμές για την ορθότητα των σχεδίων γρήγορα και εύκολα χωρίς να πρέπει κάθε φορά να μεταφέρουμε το κύκλωμα μας πάνω στην αναπτυξιακή κάρτα FPGA, χρησιμοποιήθηκε αρχικά το λογισμικό ModelSim για την εξομοίωση της αρχιτεκτονικής και στη συνέχεια εκτελέστηκαν τα στάδια της σύνθεσης (Synthesize) και της υλοποίησης (Implement Design) του εργαλείου σε FPGAs της Xilinx (οικογένειας, Virtex-6) με χρήση της σουίτας Xilinx ISE Design Suit. Το Matlab επίσης βοήθησε και στο στάδιο της εξομοίωσης, εκτυπώνοντας τα αρχεία με τα χρήσιμα δεδομένα που ήταν απαραίτητα στη διάρκεια της εξομοίωσης. Οι μετρήσεις επικεντρώθηκαν στην διερεύνηση της μέγιστης δυνατής ταχύτητας κωδικοποίησης που μπορεί να επιτύχει η προτεινόμενη αρχιτεκτονική καθώς και στην ορθή λειτουργία του κωδικοποιητή υπό διάφορες συνθήκες. Η σωστή λειτουργία του κωδικοποιητή έχει αποδειχθεί, πραγματοποιώντας μετρήσεις για όλες τις περιπτώσεις που μας πρότεινε το standard, όμως οι μετρήσεις που παρουσιάζονται στη συγκεκριμένη εργασία είναι για τη περίπτωση 1. Δηλαδή, όταν το μήκος των εισερχόμενων ψηφίων πληροφορίας είναι 168. Ο πίνακας ελέγχου ισοτιμίας *H* είναι διαστάσεων 168x336. Το code rate είναι ½.



Σχήμα 6.1 : Εξομοίωση λειτουργίας κωδικοποιητή σύμφωνα με την προτεινόμενη αρχιτεκτονική

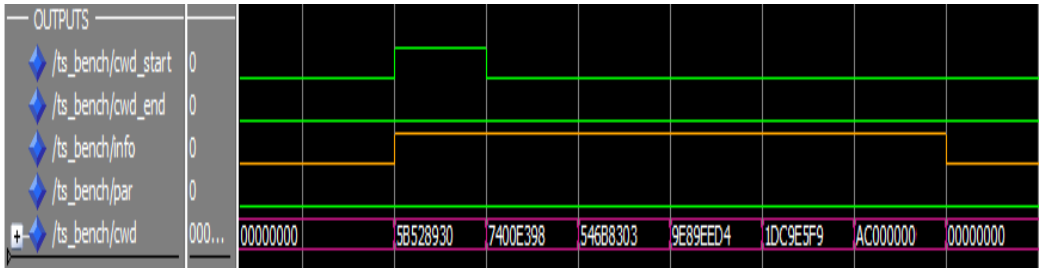
Στο *σχήμα 6.1* παρουσιάζονται οι κυματομορφές που προέρχονται από την εξομοίωση σε ModelSim της προτεινόμενης αρχιτεκτονικής. Φαίνονται τα σήματα εισόδου/εξόδου του κωδικοποιητή. Με μπλε χρώμα τονίζονται τα εισερχόμενα δεδομένα και με φούξια είναι η κωδική λέξη. Επίσης στο σχήμα μπορούμε ενοποιήσουμε την αρχή και την λήξη της εισερχόμενης πληροφορίας, της κωδικής λέξης, των information bits και των parity bits από τα οποία αποτελείται η κωδική λέξη. Το σήμα *enc_mode* μας δείχνει ότι υλοποιείται η πρώτη περίπτωση.

Αν μεγεθύνουμε την κυματομορφή που μας δίνει ο εξομοιωτής στο σήμα που φέρνει την εισερχόμενη πληροφορία, φαίνονται τα bits σε Hexadecimal μορφή, που πρόκειται να κωδικοποιήθούν, καθώς το σήματα *blk_start* (μοβ χρώμα) δείχνει την αρχή των ψηφίων και το σήμα *blk_end* (πράσινο χρώμα) την λήξη.



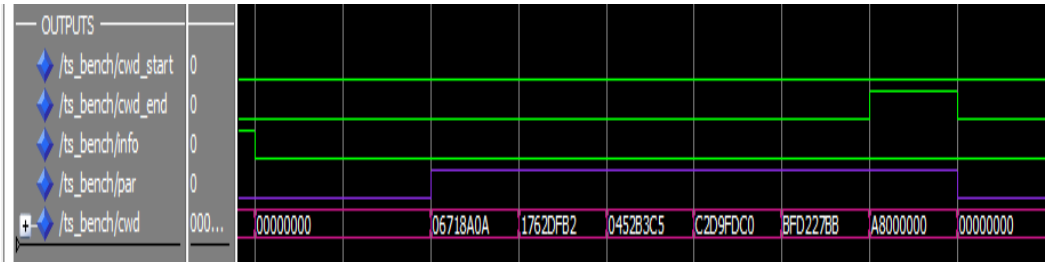
Σχήμα: 6.2 Εισερχόμενα δεδομένα που πρόκειται να κωδικοποιηθούν

Αμέσως μετά από δυο κύκλους ρολογιού το σήμα *cwd_start* γίνετε «1», που σημαίνει έναρξη της κωδικής λέξης. Το σήμα *info* (πορτοκαλί χρώμα) επίσης είναι «1» σε όλη τη διάρκεια των information bits από τα οποία αποτελείται η κωδική λέξη. Αν συγκρίνουμε τα δεδομένα του σχήματος 6.2 και τα δεδομένα του σχήματος 6.3 , θα διαπιστώσουμε ότι είναι ακριβός τα ίδια και είναι πολύ σωστά, γιατί όπως αναφέραμε και πριν, μια κωδική λέξη αποτελείται από την εισερχόμενη πληροφορία και τα parity bits.



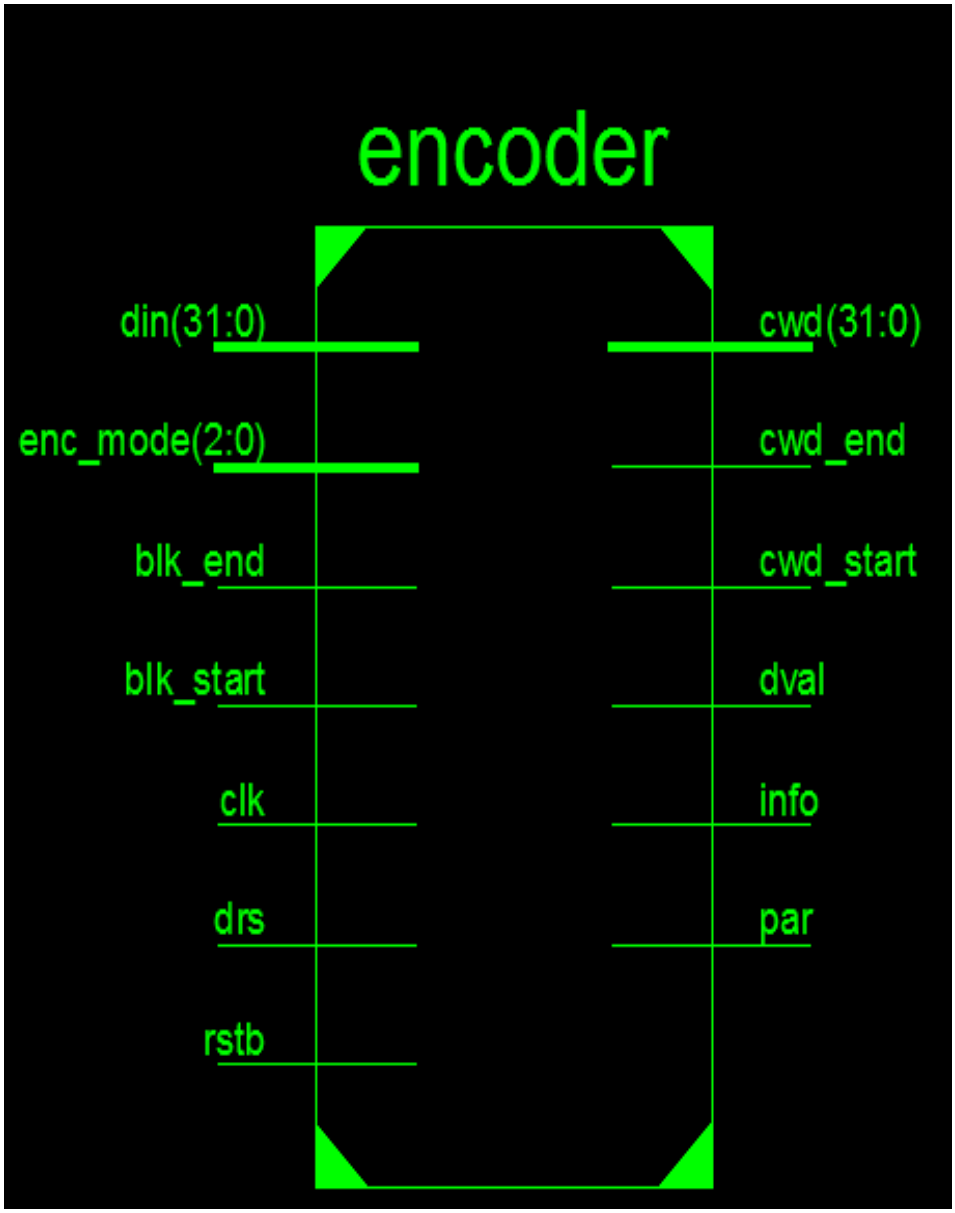
Σχήμα: 6.3 Information bits της κωδικής λέξης

Τέλος, μετά από δυο κύκλους ρολογιού το σήμα *par* (μοβ χρώμα) γίνεται «1» σε όλη τη διάρκεια των parity bits, και στο σήμα cwd εμφανίζονται τα σωστά δεδομένα των parity bits, σύμφωνα με το αρχείο δεδομένων που εκτυπώθηκε μέσω του Matlab.

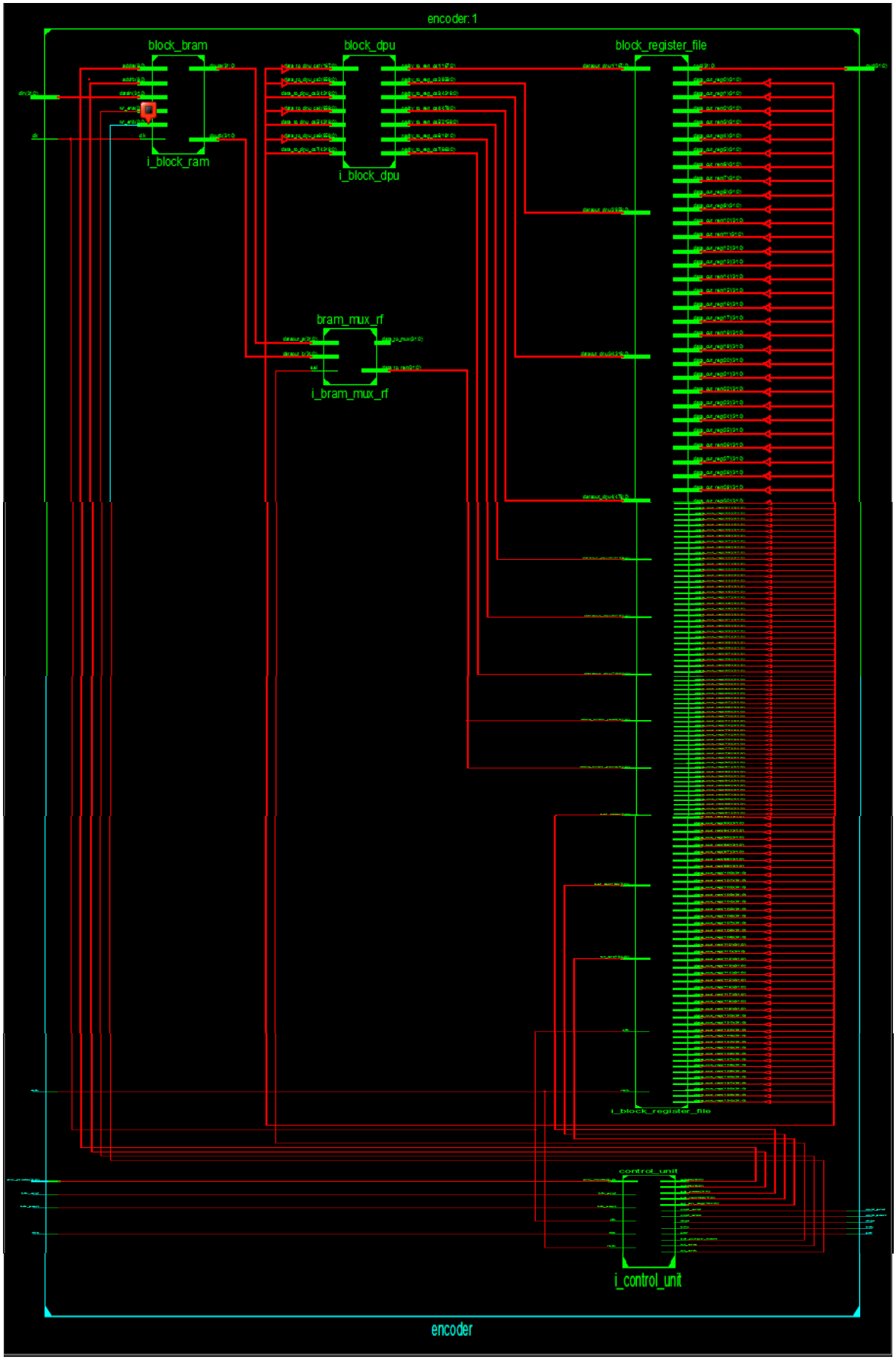


Σχήμα: 6.4 Parity bits της κωδικής λέξης

Ακολουθεί η εικόνα του top level του κωδικοποιητή LDPC, την οποία παράγει η φάση της σύνθεσης του εργαλείου Xilinx ISE Design Suit. Αν κάνουμε double click στο συγκεκριμένο επίπεδο, θα εμφανιστεί η RTL σχηματική αναπαράσταση της σχεδίασης. Φαίνονται ξεκάθαρα όλα τα blocks από τα οποία αποτελείτε η προτεινόμενη αρχιτεκτονική και οι συνδέσεις μεταξύ τους.



Σχήμα :6.5 Top level του κωδικοποιητή LDPC



Σχήμα :6.6 RTL σχηματική αναπαράσταση της σχεδίασης

Ολοκληρώνοτας την φάση Place&Route με επιτυχεία το λογισμικό Xilinx ISE Design Suit παράγει αρχείο .par, το οποίο τυπώνεται παρακάτω. Ουσιαστικά είναι ένα αρχείο, όπου περιγράφονται τα στατιστικά στοιχεία του design μας, δηλαδή εμφανίζεται ο αριθμός των διάφορων κυκλωμάτων που χρησιμοποιούνται καθώς και ο χρόνος εκτέλεσης του κωδικοποιητή. Σύμφωνα με το πινακάκι που εμφανίζει το αρχείο, στο οποίο δίνεται ο χρόνος και συγκεκριμένα στο πεδίο Best Case Achievable έχουμε τον χρόνο 3,8 ns. Αυτό σημαίνει ότι χρησιμοποιώντας ένα ρολόι συχνότητας 100 MHz, το Throughput rate το οποίο πετυχαίνουμε είναι 3.8 Gbps. Ο αριθμός αυτός είναι αρκετά πιο υψηλός από τον αρχικό μας στόχο, όμως να υπενθυμίσουμε ότι στη προκειμένη περίπτωση εξετάζουμε μόνο το case-1.

```
Device Utilization Summary:

Slice Logic Utilization:
  Number of Slice Registers:                4,511 out of 408,000    1%
    Number used as Flip Flops:              4,511
    Number used as Latches:                     0
    Number used as Latch-thrus:                 0
    Number used as AND/OR logics:               0
  Number of Slice LUTs:                     2,142 out of 204,000    1%
    Number used as logic:                   2,135 out of 204,000    1%
      Number using O6 output only:          1,996
      Number using O5 output only:              0
      Number using O5 and O6:                 139
      Number used as ROM:                       0
    Number used as Memory:                      0 out of  70,200    0%
    Number used exclusively as route-thrus:     7
      Number with same-slice register load:     7
      Number with same-slice carry load:        0
      Number with other load:                   0

Slice Logic Distribution:
  Number of occupied Slices:                1,801 out of  51,000    3%
  Number of LUT Flip Flop pairs used:       6,144
    Number with an unused Flip Flop:        1,644 out of   6,144   26%
    Number with an unused LUT:              4,002 out of   6,144   65%
    Number of fully used LUT-FF pairs:        498 out of   6,144    8%
    Number of slice register sites lost
      to control set restrictions:              0 out of 408,000    0%

 IO Utilization:
  Number of bonded IOBs:                       76 out of     600   12%


Generating "PAR" statistics.

------------------------------------------------------------------------------------------------------
  Constraint                                |    Check    | Worst Case |  Best Case | Timing |   Timing
                                            |             |    Slack   | Achievable | Errors |    Score
------------------------------------------------------------------------------------------------------
  Autotimespec constraint for clock net clk | SETUP       |         N/A|     3.800ns|     N/A|           0
  _BUFGP                                    | HOLD        |     0.122ns|            |       0|           0
------------------------------------------------------------------------------------------------------


Generating Pad Report.

All signals are completely routed.

Peak Memory Usage:  1190 MB

Placer: Placement generated during map.
Routing: Completed - No errors found.

Number of error messages: 0
Number of warning messages: 0
Number of info messages: 2

PAR done!
```

Συμπεράσματα

Σύμφωνα με τις πληροφορίες για τα χαρακτηριστικά, τις επίδοσης και την πολυπλοκότητα των αλγορίθμων LDPC που δίνονται στη παρούσα εργασία, μπορούμε να βγάλουμε το συμπέρασμα ότι οι κώδικες χαμηλής πυκνότητας ελέγχου ισοτιμίας αποτελούν άξιους ανταγωνιστές των πολλών κωδίκων ανίχνευσης και διόρθωσης σφαλμάτων . Το γεγονός ότι προσεγγίζουν το όριο Shannon κάνουν τους κώδικες LDPC να βρίσκονται σε εξέχουσα θέση.
Σε αυτή τη πτυχιακή εργασία μελετήθηκαν τα βασικά χαρακτηριστικά των κωδίκων, τρόποι αναπαράστασής τους καθώς επίσης δόθηκε έμφαση στους αλγόριθμους κωδικοποίησης. Με βάση τις προδιαγραφές του standard, σχεδιάστηκε και υλοποιήθηκε ένας κωδικοποιητής VLSI για κώδικες LDPC . Στα πλαίσια της ανάπτυξης πραγματοποιήθηκαν αρκετές μελέτες όσον αφορά την επιλογή του κατάλληλου αλγορίθμου κωδικοποίησης, καθώς και ιδιαίτερη προσοχή στην ανάπτυξη του. Ο κώδικας που περιγράφει τον αλγόριθμο σε γλώσσα vhdl, είναι πλήρως παραμετρικός και αποτελεί ένα στοιχείο που δίνει την δυνατότητα στον χρήστη να τροποποιήσει του κώδικα σε οποιοδήποτε σημείο και οποιαδήποτε στιγμή. Έχοντας ως στόχο τη σχεδίαση ενός κωδικοποιητή με ρυθμό απόδοσης να υπερισχύει το 1 Gbps, υλοποιήθηκε η κατάλληλη παράλληλη αρχιτεκτονική, εφαρμόζοντας με αποδοτικό τρόπο την τεχνική pipelining, με σκοπό να μειωθεί το κρίσιμο μονοπάτι της αρχιτεκτονικής και να αυξηθεί η ταχύτητα στην διαδικασία κωδικοποίησης.
Η προτεινόμενη μέθοδος υλοποίησης του κωδικοποιητή LDPC αποδείχθηκε κατάλληλη και αποδοτική καθώς πετύχαμε τον αρχικό σκοπό, τηρώντας όλες τις προδιαγραφές και απατήσεις.

Αναφορές-Βιβλιογραφία

- "Αρχές Τηλεπικοινωνιακών Συστημάτων"- Herbert Taub , Donald L. Schilling
- *"Low-Density Parity-Check Codes"*- R. G. Gallager, Cambridge
- "Near Shannon Limit Performance of Low Density Parity Check Codes"- David J. C. MacKay, Radford M. Neal
- "Efficient Encoding of Low-Density Parity-Check Codes"- Thomas J. Richardson and Rüdiger L. Urbanke
- "A Flexible Hardware Encoder for Low-Density Parity-Check Codes" - Dong-U Lee and Wayne Luk, Connie Wang, Christopher Jones, Michael Smith, and John Villasenor
- "Encoder Architecture with Throughput Over 10 Gbit/sec for Quasi-cyclic LDPC Codes"- Zhiyong He, Sebastien Roy, and Paul Fortier, Laval University Canada
- "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes"-Thomas J. Richardson, M. Amin Shokrollahi, and Rüdiger L. Urbanke
- "Code Construction and FPGA Implementation of a Low-Error-Floor Multi-Rate Low-Density Parity-Check Code Decoder"- Lei Yang*, Student Member, IEEE*, Hui Liu*, Senior Member, IEEE*, and C.-J. Richard Shi
- *"*A Tutorial on Low Density Parity-Check Codes"- Tuan Ta -The University of Texas at Austin
- *"*Programmable Quasi Cyclic LDPC Encoder Architecture"- Ming ZHAO, Liang LI, Xiujun ZHANG, Yin SUN, Xiang CHEN
- *"*A Low-Complexity Hybrid LDPC Code Encoder for IEEE 802.3an (10GBase-T) Ethernet"- Aaron E. Cohen, Member, IEEE, and Keshab K. Parhi, Fellow, IEEE
- "Σχεδίαση ψηφιακών συστημάτων με τη γλώσσα VHDL"- Stephen Brown, Zvonko Vranesic