Τεχνικές διόρθωσης και ανίχνευσης σφαλμάτων

Εντοπισμός σφαλμάτων

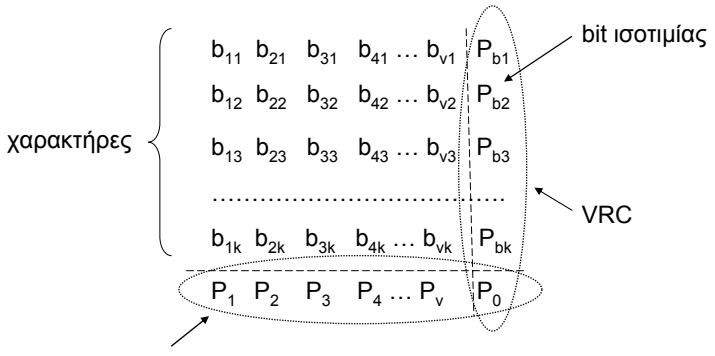
- Εντοπισμός
- Διόρθωση
- Προστίθενται bit πλεονασμού
- Αν μπορεί διορθώνει, (forward error correction)
- αλλιώς ζητά επανεκπομπή (backward error correction)
- Bit Error Rate (BER)

Έλεγχος ισοτιμίας (parity)

- Προστίθεται ένα bit ισοτιμίας
- Αν το πλήθος των "1" είναι μονός αριθμός: περιττή ισοτιμία (odd parity)
- Αν το πλήθος των "1" είναι ζυγός αριθμός: άρτια ισοτιμία (even parity)
- Έστω: 0 1001101 άρτια ισοτιμία

  parity bit, μήνυμα
- Η ισοτιμία έχει προσυμφωνηθεί
- Μπορεί να ελέγξει μόνο περιττό αριθμό σφαλμάτων
- Πλέον χρησιμοποιείται μόνο για λόγους συμβατότητας

Δισδιάστατος έλεγχος ισοτιμίας



Πάντα ανιχνεύεται περιττός αριθμός σφαλμάτων
Άρτιος αριθμός σφαλμάτων ανιχνεύεται ικανοποιητικά
και συνήθως βελτιώνεται ο εντοπισμός σφαλμάτων κατά 100 έως 1000 φορές

Απόσταση Hamming

- Έστω ότι τα μηνύματά μας αποτελούνται από *m* bits. Έστω, επίσης, ότι χρησιμοποιούμε *r* bits ελέγχου (πλεονασμού) για να ανιχνεύουμε τα σφάλματα. Οι λέξεις μεγέθους $n = m + r$ που μεταδίδονται ονομάζονται κωδικές λέξεις (codewords).
- Έστω δύο κωδικές λέξεις, οι 10001001 και 10110001. Είναι δυνατόν να πούμε σε πόσα bit διαφέρουν αν τις κάνουμε XOR και μετρήσουμε τα "1" στο αποτέλεσμα:

  10001001
  10110001

  00111000

  Αυτή η διαφορά τους ονομάζεται απόσταση Hamming των δύο λέξεων

Απόσταση Hamming

- Η σημασία της απόστασης Hamming είναι ότι αν δύο λέξεις απέχουν *d* bits, τότε αρκούν *d* σφάλματα για να μετατρέψουν τη μια στην άλλη.
- Συνήθως και τα 2^m μηνύματα είναι έγκυρα, όχι όμως όλες οι 2^n κωδικές λέξεις. Αυτές επιλέγονται με τρόπο που να μεγιστοποιεί την απόσταση Hamming
- Η απόσταση Hamming μεταξύ των κωδικών λέξεων προσδιορίζει την ικανότητα εντοπισμού σφαλμάτων και την ικανότητα διόρθωσης σφαλμάτων ενός κώδικα
- Αν έχω απόσταση $d+1$ μπορώ να ανιχνεύσω *d* απλά σφάλματα γιατί τα *d* σφάλματα δεν αρκούν να μετατρέψουν μια λέξη σε μια άλλη έγκυρη κωδική λέξη. Άρα το λάθος θα φανεί.
- Για να διορθώσω *d* σφάλματα πρέπει η απόσταση να είναι $2d+1$ γιατί τότε ακόμα κι αν συμβούν *d* σφάλματα η αρχική κωδική λέξη θα είναι κοντύτερα σε αυτή με τα σφάλματα οπότε ο δέκτης μπορεί να τις αντικαταστήσει.

Κώδικας Hamming για διόρθωση 1 σφάλματος

- Αν ισχύει η σχέση, τότε χρειάζονται τουλάχιστον *r* bits πλεονασμού για τη διόρθωση ενός σφάλματος σε μηνύματα *m* bits

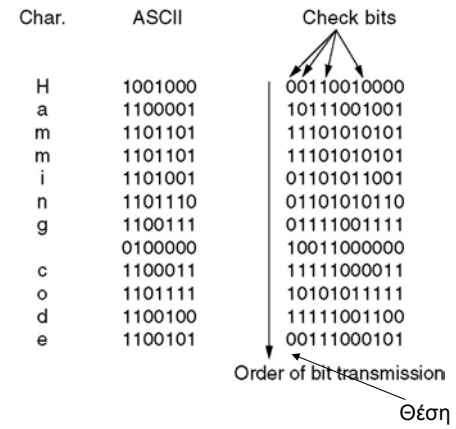
$$2^r \geq m + r + 1$$

- Έστω ένα block από 9 bit (100110100). Χρειάζονται r=4 bits ελέγχου.
- Τα bit ελέγχου τοποθετούνται στις θέσεις H, HH1H001H10100

Υπολογισμός κώδικα Hamming

- Σημείωσε όλες τις θέσεις που είναι δυνάμεις του 2, ξεκινώντας από αριστερά. Αυτές είναι τα hamming bits.
- Σε όλες τις υπόλοιπες θα τοποθετηθούν τα bit προς κωδικοποίηση.
- Κάθε parity bit (hamming bit) ελέγχει κάποια από τα bit της λέξης. π.χ. το 1 ελέγχει τα 1, 3, 5, 7, 9, …., το 2 ελέγχει τα 2, 3,.. 6, 7,.. 10, 11,.., το 4 ελέγχει τα 4, 5, 6, 7, ..,12, 13, 14, 15, .. κοκ.
- Το κάθε parity bit ορίζεται έτσι ώστε να υπάρχει

Χρήση κώδικα Hamming για διόρθωση σφαλμάτων σε ριπές



Για κάθε ακολουθία που φτάνει στο δέκτη, εκείνος αρχικοποιεί έναν μετρητή *k*.
Ελέγχει τα check bits (*k*=1, 2, 4, 8, …) αν έχουν σωστό parity.
Σε περίπτωση προσθέτει την τιμή του *k* στον μετρητή.
Στο τέλος αν *k*=0 δεν υπάρχουν σφάλματα, αλλιώς το *k* περιέχει τη θέση του σφάλματος.
Αν κατασκευαστεί η δισδιάστατη δομή του σχήματος και τα δεδομένα μεταδίδονται σε στήλες (από αριστερά προς τα δεξιά), τότε ο κώδικας μπορεί να ανιχνεύσει και ριπές μέχρι *k* σφαλμάτων.

Τα bit που βρίσκονται σε θέσεις που μπορούν να εκφραστούν σαν δυνάμεις του 2, είναι bit ελέγχου. Κάθε θέση ελέγχεται από εκείνα τα bit που την προσδιορίζουν, π.χ. το bit 11 ελέγχεται από τα 11=1+2+8.

Έλεγχος κυκλικού πλεονασμού
(Cyclic Redundancy Check – CRC)

- Αρχικά θα πρέπει να σκεφτόμαστε οποιαδήποτε ακολουθία *n*-bits ως ένα πολυώνυμο *n-1* βαθμού. Ο συντελεστής του κάθε όρου του πολυωνύμου λαμβάνει την τιμή του αντίστοιχου bit της ακολουθίας. Για παράδειγμα, η ακολουθία 1101011011 αντιστοιχεί στο πολυώνυμο.

$$M(x) = x^9 + x^8 + x^6 + x^4 + x^3 + x^1 + 1$$

- Τόσο στον αποστολέα κόμβο όσο και στον παραλήπτη είναι γνωστή εκ των προτέρων μια ειδική ακολουθία, η οποία ονομάζεται *πολυώνυμο γεννήτορας* και συμβολίζεται με *G(x)*.

Λειτουργία αλγορίθμου CRC

Αποστολέας κόμβος:

- Πολλαπλασίασε το *M(x)* με το x^k, όπου *k* είναι ο βαθμός του προκαθορισμένου πολυωνύμου *G(x)*. Αυτό ουσιαστικά αντιστοιχεί σε αύξηση του μήκους των δεδομένων κατά *k* bits, με αριστερή ολίσθηση (left shift) κατά *k* των αρχικών bits και ταυτόχρονη πλήρωση των κενών θέσεων με μηδενικά.
- Διαίρεσε το *M(x)*. x^k με το *G(x)*. Από αυτή τη διαίρεση προκύπτουν το πηλίκο *Q(x)* και το υπόλοιπο *R(x)*.
- Σύνθεσε το μήνυμα T(x), το οποίο θα αποστείλεις στο δίκτυο προς μεταφορά, όπου

$$T(x) = M(x) \cdot x^k + R(x)$$

Παραλήπτης κόμβος:

- Διαίρεσε το ληφθέν μήνυμα με το *G(x)*. Εάν το υπόλοιπο είναι μηδέν, δεν υπάρχει σφάλμα μεταφοράς.

Frame : 1 1 0 1 0 1 1 0 1 1
Generator: 1 0 0 1 1
Message after 4 zero bits are appended: 1 1 0 1 0 1 1 0 1 1 0 0 0 0

```
                1 1 0 0 0 0 1 0 1 0
          ┌─────────────────────────
10011     │ 1 1 0 1 0 1 1 0 1 1 0 0 0 0
            1 0 0 1 1
            ─────────
            1 0 0 1 1
            1 0 0 1 1
            ─────────
            0 0 0 0 1
            0 0 0 0 0
            ─────────
              0 0 0 1 0
              0 0 0 0 0
              ─────────
                0 0 1 0 1
                0 0 0 0 0
                ─────────
                  0 1 0 1 1
                  0 0 0 0 0
                  ─────────
                    1 0 1 1 0
                    1 0 0 1 1
                    ─────────
                      0 1 0 1 0
                      0 0 0 0 0
                      ─────────
                        1 0 1 0 0
                        1 0 0 1 1
                        ─────────
                          0 1 1 1 0
                          0 0 0 0 0
                          ─────────
                            1 1 1 0   ← Remainder
```

Προσοχή!
Σε περίπτωση που το πρώτο ψηφίο είναι το "0" κάνουμε XOR με το 00000 και όχι με το G(x)

Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 0

Οι κώδικες CRC μήκους k bits έχουν τις ακόλουθες δυνατότητες εντοπισμού σφαλμάτων μεταφοράς:

- Όλα τα σφάλματα μονού bit, αρκεί οι όροι x^k και x^0 να έχουν μη μηδενικούς συντελεστές.
- Όλα τα σφάλματα διπλού bit, αρκεί το πολυώνυμο να περιέχει τρεις τουλάχιστον όρους.
- Όλα τα σφάλματα περιττού πλήθους, αρκεί το πολυώνυμο να περιέχει ως παράγοντα τον όρο $(x+1)$.
- Όλα τα σφάλματα σε δέσμη bits με μήκος μικρότερο από k bits, όπου ως δέσμη ονομάζουμε ένα πλήθος από διαδοχικά bits. Επίσης, εντοπίζονται τα περισσότερα από τα σφάλματα μεταφοράς σε δέσμες με μήκος μεγαλύτερο από k bits.

- Απλή και οικονομική υλοποίηση. Μόνο shift και XOR.

Συχνά χρησιμοποιούμενοι κώδικες CRC

- CRC-8 (ATM): $x^8 + x^2 + x + 1$
- CRC-10 (ATM): $x^{10} + x^9 + x^5 + x^4 + x + 1$
- CRC-ITU-T (HDLC): $x^{16} + x^{12} + x^5 + 1$
- CRC-32 (ethernet):

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + \ldots$$
$$+x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Άσκηση

Έστω ότι θέλετε να μεταφέρετε τα δεδομένα:

11010011010011101001

προστατεύοντάς τα με αλγόριθμο CRC με πολυώνυμο γεννήτορα το CRC-8.

1. Ποιο μήνυμα θα στέλνατε στο δίκτυο;

2. Αν αλλοιωθούν τα bits 2, και 9 ποιο θα είναι το υπόλοιπο της διαίρεσης; Πως θα εξακριβώσει ο παραλήπτης τα σφάλματα μεταφοράς;