ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΜΣ Κυβερνοασφάλεια
και Επιστημη Δεδομενων
MSc Cybersecurity
and Data Science
DEPT OF INFORMATICS
UNIVERSITY OF PIRAEUS

EMBEDDED SYSTEMS RELIABILITY

Error Correction Codes

Μιχάλης Ψαράκης, Δημήτριος Αγιακάτσικας

Πλεονασμός πληροφορίας

- Παρέχεται με τη χρήση πρόσθετων ψηφίων στα αρχικά δεδομένα ώστε ένα σφάλμα στα ψηφία δεδομένων να ανιχνευτεί και ίσως να διορθωθεί
- Χρησιμοποιούνται κώδικες (codes)
  - Ανίχνευσης σφαλμάτων (error detecting codes)
  - Διόρθωσης σφαλμάτων (error correcting codes)
- Επιβάρυνση
  - Πρόσθετα ψηφία
  - Επιπλέον υλικό/χρόνος για την κωδικοποίηση (encoding) και την αποκωδικοποίηση (decoding)

Κώδικες: εφαρμογές

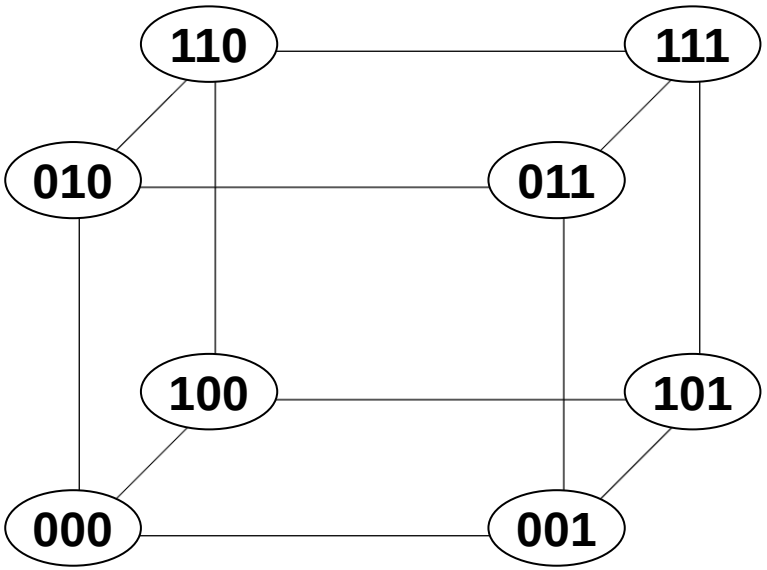
- Μετάδοση πληροφορίας μεταξύ
  - Υπολογιστών
  - Επεξεργαστή & μνήμης
  - Επεξεργαστή & συσκευών εισόδου/εξόδου
- Μετάδοση πληροφορίας μέσω
  - Δικτύων (local, wide-area networks)
  - Εσωτερικών αγωγών (high-speed internal buses)

Κώδικες: βασικά

- Λέξη πληροφορίας (data word): m bits
- Κωδική λέξη (code word): c bits (c > m)
- Μία λέξη πληροφορίας (m bits) κωδικοποιείται σε μία κωδική λέξη (c bits)
  - Δεν είναι έγκυρες όλες οι 2^c κωδικές λέξεις
- Για να εξάγουμε τα αρχικά δεδομένα (m bits) πρέπει να αποκωδικοποιήσουμε την κωδική λέξη (c bits)
  - Εάν η κωδική λέξη δεν είναι έγκυρη τότε έχει ανιχνευτεί σφάλμα
- Κάποιοι κώδικες επιτρέπουν τη διόρθωση σφαλμάτων
- Παράμετροι κωδίκων:
  - Αριθμός εσφαλμένων ψηφίων που μπορούν να ανιχνεύσουν
  - Αριθμός εσφαλμένων ψηφίων που μπορούν να διορθώσουν

Απόσταση Hamming

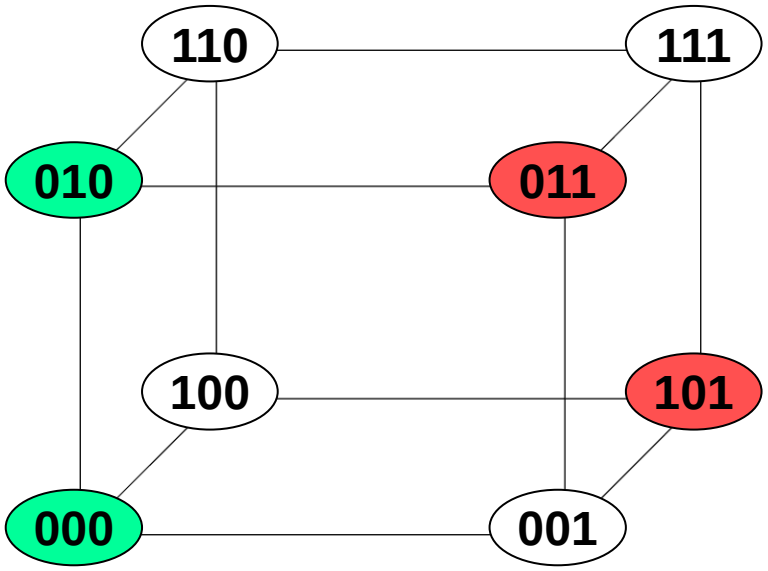
- Απόσταση Hamming (Hamming distance)
  μεταξύ δυο κωδικών λέξεων =
  αριθμός των ψηφίων που διαφέρουν οι δυο λέξεις



- Δυο κωδικές λέξεις (κόμβοι) συνδέονται με μία ακμή εάν η απόσταση Hamming είναι 1

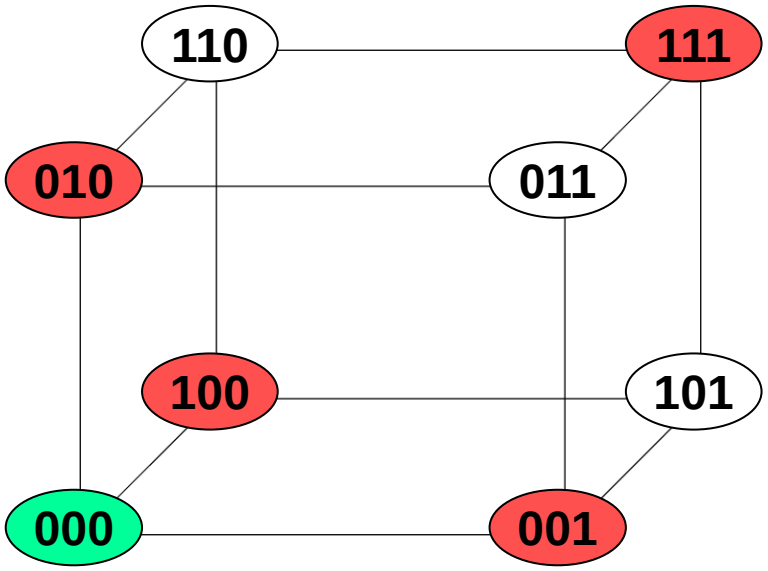
Απόσταση Hamming: παραδείγματα

- 000 και 010 διαφέρουν σε 1 θέση = απόσταση Hamming 1
  - Ένα σφάλμα στο μεσαίο ψηφίο θα μετατρέψει τη μία κωδική λέξη στην άλλη
    - Δεν είναι δυνατόν να ανιχνευτεί
- 011 και 101 διαφέρουν σε 2 θέσεις = απόσταση Hamming 2
  - Πρέπει να διατρέξουμε 2 ακμές για να πάμε από τον κόμβο 011 στον 101
  - Ένα μονό σφάλμα δεν είναι δυνατόν να μετατρέψει τη μία κωδική λέξη στην άλλη
    - Είναι δυνατόν να ανιχνευτεί



Απόσταση ενός κώδικα

- Ελάχιστη απόσταση Hamming μεταξύ δυο οποιονδήποτε έγκυρων κωδικών λέξεων
- Παράδειγμα 1: κώδικας με 4 κωδικές λέξεις {001,010,100,111} : απόσταση 2
  - Ανιχνεύει οποιοδήποτε μονό λάθος
- Παράδειγμα 2: κώδικας με 2 κωδικές λέξεις {000,111} : απόσταση 3
  - Ανιχνεύει οποιοδήποτε μονό ή διπλό λάθος
  - Εάν υποθέσουμε ότι διπλό λάθος δεν πρόκειται να συμβεί τότε μπορεί να διορθώσει οποιοδήποτε μονό λάθος



Ανίχνευση και διόρθωση

- Ένας κώδικας με απόσταση d μπορεί να ανιχνεύσει έως D σφάλματα, εάν
  - $d \geq D + 1$
- Ένας κώδικας με απόσταση d μπορεί να διορθώσει έως C σφάλματα, εάν
  - $d \geq 2C + 1$
- Ένας κώδικας με απόσταση d μπορεί να ανιχνεύσει D σφάλματα και να διορθώσει C σφάλματα, εάν
  - $d \geq D + C + 1$, $(D \geq C)$
- $d = 1 \rightarrow D = C = 0$
  - κανένας κώδικας
- $d = 2 \rightarrow D = 1, C = 0$
  - κώδικας ανίχνευσης μονού λάθους (single error detecting – SED code)
- $d = 3 \rightarrow D = 1, C = 1$
  - κώδικας διόρθωσης μονού λάθους - ανίχνευσης μονού λάθους (single error correcting single error detecting – SECSED code)
- $d = 4 \rightarrow D = 2, C = 1$
  - κώδικας διόρθωσης μονού λάθους - ανίχνευσης διπλού λάθους (single error correcting double error detecting – SECDED code)

Διαχωριστικότητα ενός κώδικα

- Διαχωρίσιμος κώδικας (separable code)
  - ξεχωριστά πεδία για τα ψηφία δεδομένων (data bits) και τα ψηφία ελέγχου (check bits)
  - η αποκωδικοποίηση εκτελείται αγνοώντας τα ψηφία ελέγχου
- Μη-διαχωρίσιμος κώδικας (non-separable code)
  - τα ψηφία δεδομένων και ελέγχου είναι αναμεμιγμένα
  - η αποκωδικοποίηση απαιτεί περισσότερη επεξεργασία

Κώδικες ισοτιμίας

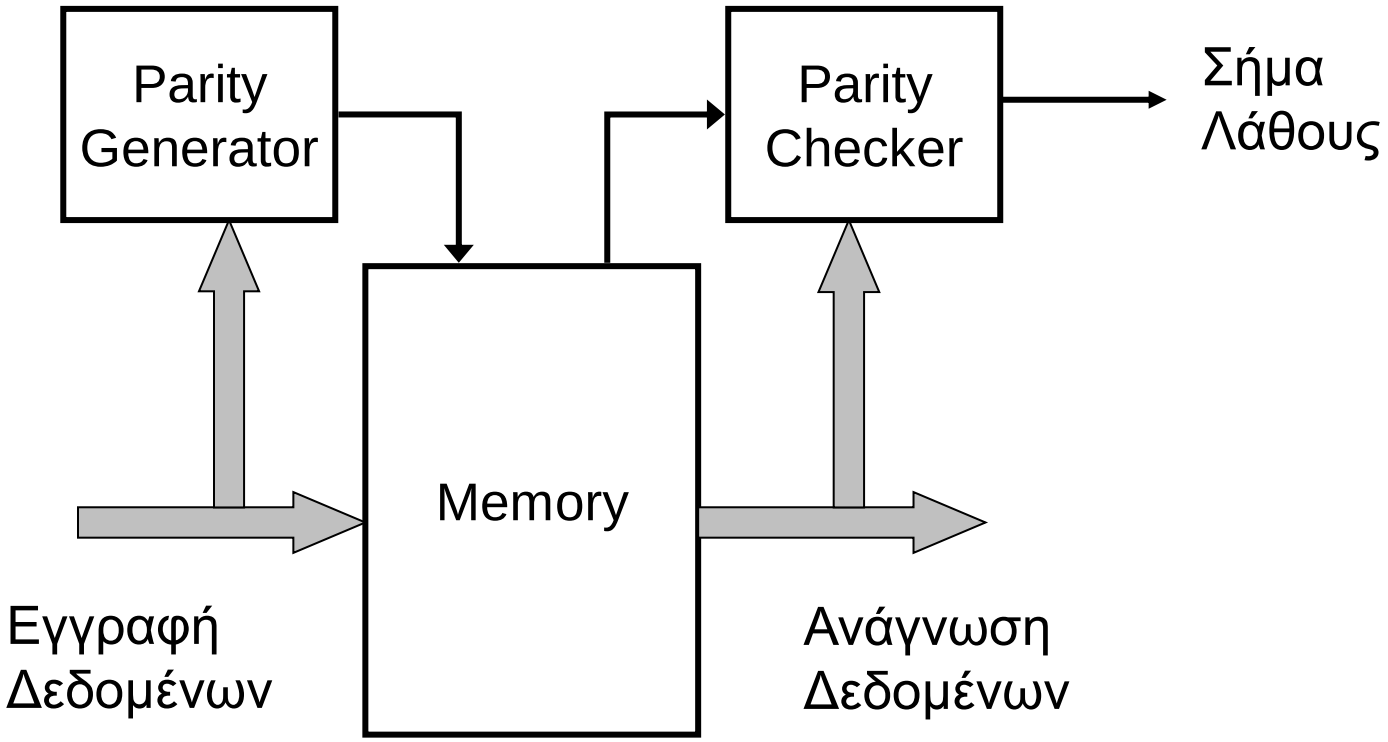
- Οι απλούστεροι διαχωρίσιμοι κώδικες
- Μία κωδική λέξη περιλαμβάνει m bit δεδομένων και 1 επιπλέον bit ισοτιμίας (parity bit)
- Περιττή (odd) και άρτια (even) ισοτιμία

Dec. digit	BCD	BCD odd parity	BCD even parity
0	0000	0000 1	0000 0
1	0001	0001 0	0001 1
2	0010	0010 0	0010 1
3	0011	0011 1	0011 0
4	0100	0100 0	0100 1
5	0101	0101 1	0101 0
6	0110	0110 1	0110 0
7	0111	0111 0	0111 1
8	1000	1000 0	1000 1
9	1001	1001 1	1001 0

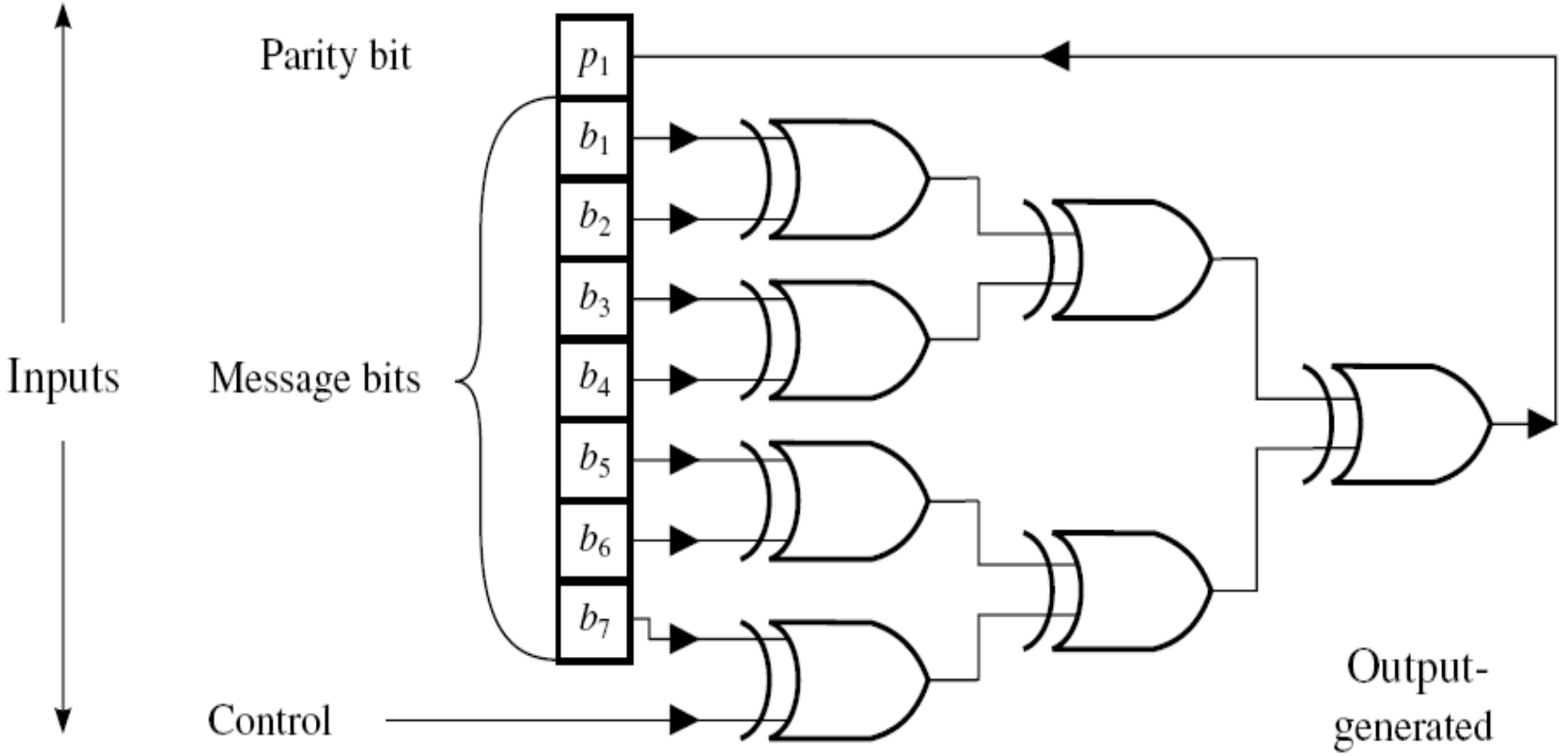
Κώδικες ισοτιμίας: ιδιότητες

- Ιδιότητες απλού κώδικα ισοτιμίας
  - Επιβάρυνση: 1/m
  - Απόσταση Hamming: 2
- Ανιχνεύει όλα τα μονά σφάλματα
  - Ανιχνεύει επίσης όλους τους περιττούς αριθμούς σφαλμάτων, π.χ. 1, 3, 5, κοκ. σφάλματα
- Δεν διορθώνει κανένα σφάλμα

Χρήση κωδικών ισοτιμίας στις μνήμες

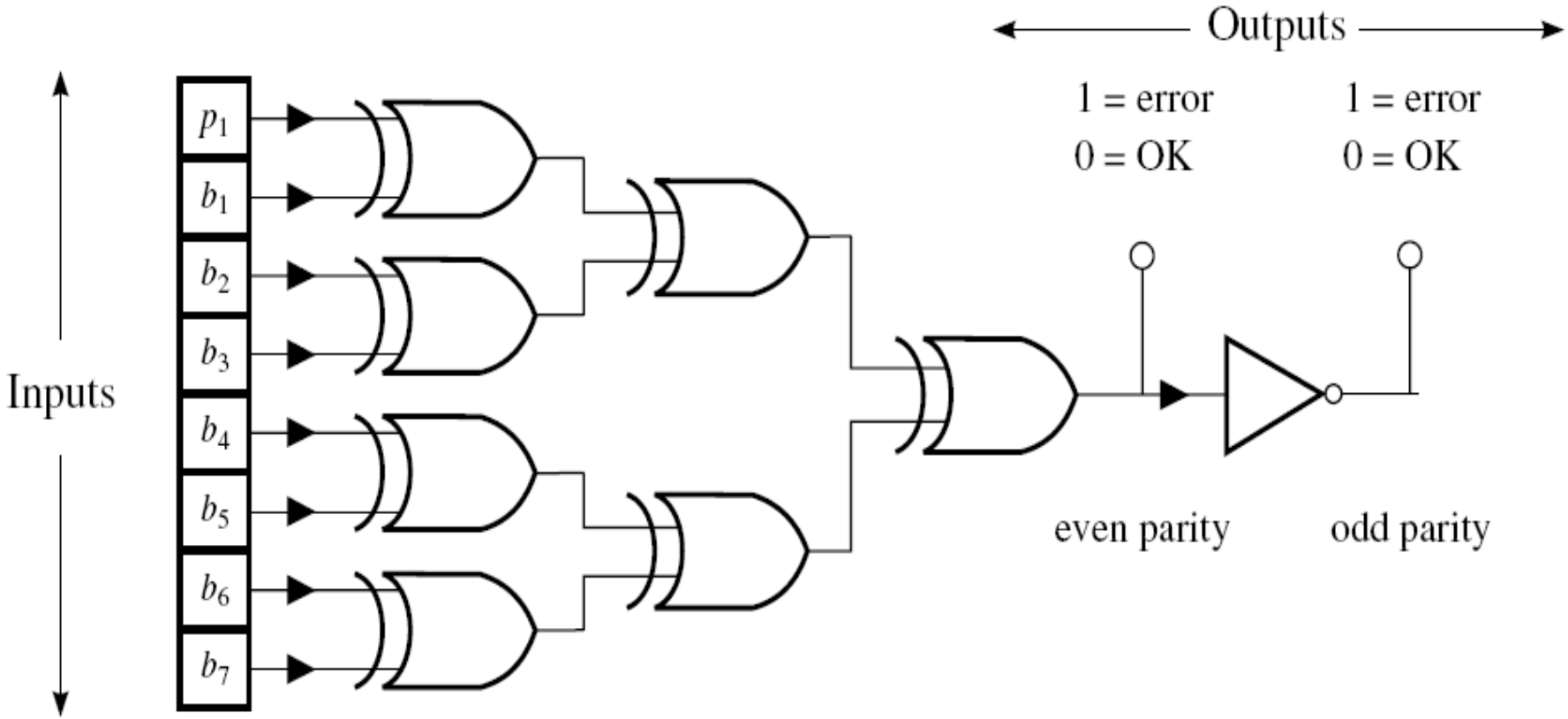


Γεννήτρια bit ισοτιμίας



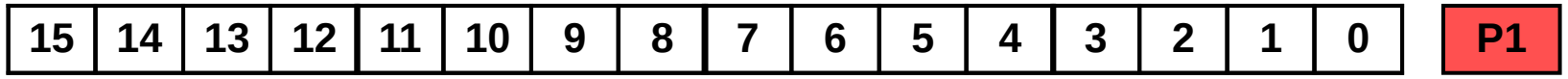
$$p_1 = b_1 \oplus b_2 \oplus b_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7$$

Ελεγκτής bit ισοτιμίας

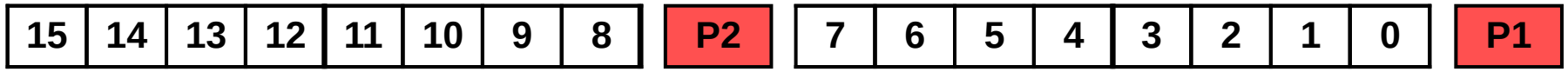


Bit ισοτιμίας ανά byte

Bit ισοτιμίας ανά λέξη



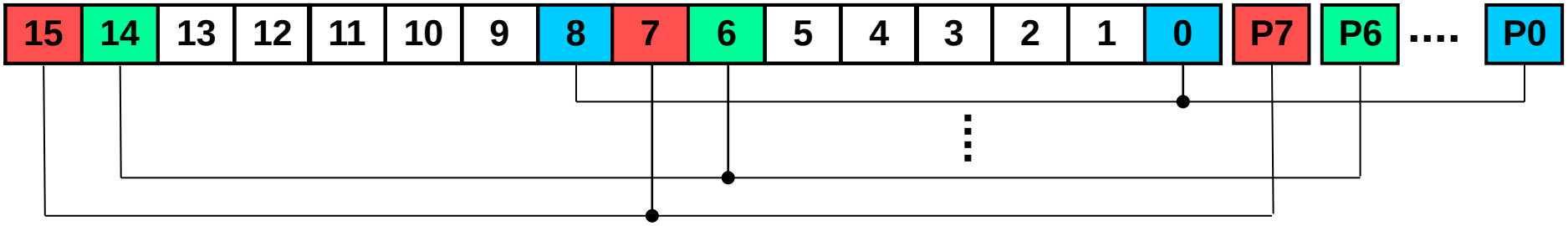
Bit ισοτιμίας ανά byte



- Επιβάρυνση: m/n, m = αριθμός των byte
- Ανιχνεύει μέχρι m σφάλματα εάν συμβούν σε διαφορετικά bytes

Κώδικας ισοτιμίας με πλέξη στα byte

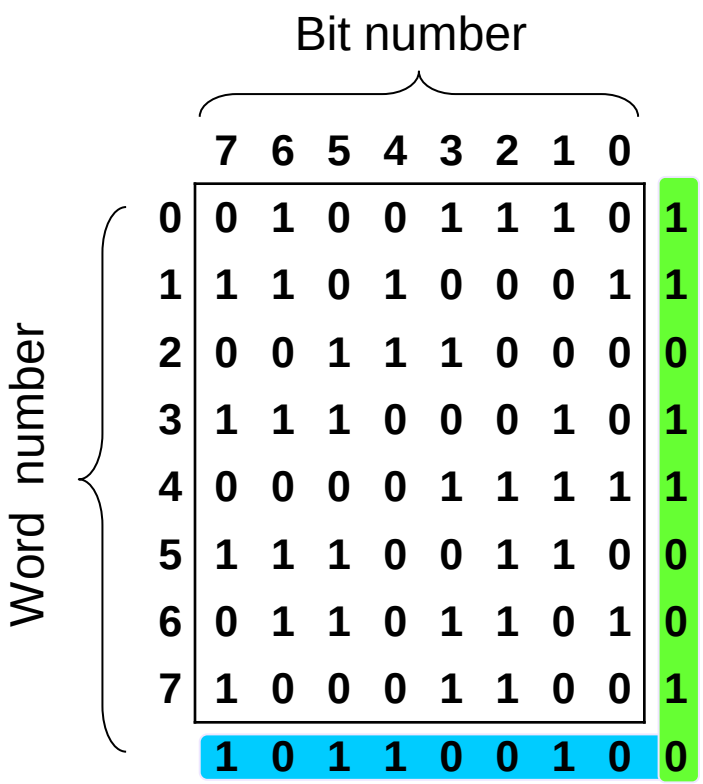
Byte-interlaced parity code



- Επιβάρυνση: 8 bit ισοτιμίας
- Αποδοτικό σχήμα όταν το βραχυκύκλωμα μεταξύ γειτονικών ψηφίων είναι συχνή περίπτωση βλάβης (π.χ. αγωγοί δεδομένων)

Κώδικες ισοτιμίας διόρθωσης σφαλμάτων

- Πιο απλή μορφή:
  οργάνωση των δεδομένων σε έναν πίνακα 2-διαστάσεων
- Εφαρμόζεται σε μπλοκ της μνήμης
- Bit ισοτιμίας γραμμής
- Bit ισοτιμίας στήλης
- Μπορεί να διορθώσει ένα μονό σφάλμα
  - ή περιττό αριθμό σφαλμάτων σε μία λέξη
- Παράδειγμα επικαλυπτόμενης ισοτιμίας (overlapping parity):
  κάθε ψηφίο καλύπτεται από περισσότερα από ένα bit ισοτιμίας



Κώδικες επικαλυπτόμενης ισοτιμίας

- Σκοπός: ανίχνευση-διόρθωση οποιουδήποτε μονού σφάλματος
- Κώδικας με m bit δεδομένα και c bit ισοτιμίας
  - Κωδική λέξη $m+c$ bit
- Υποθέτοντας μονά σφάλματα:
  - $m+c$ εσφαλμένες καταστάσεις + 1 σωστή κατάσταση → $m+c+1$ καταστάσεις
- Απαιτούνται $m+c+1$ διακριτές υπογραφές ισοτιμίας (parity signatures) για να διακρίνουμε τις καταστάσεις
- c bit ισοτιμίας → 2^c υπογραφές
- Επομένως c είναι ο μικρότερος αριθμός, ώστε:
  - $2^c \geq m + c + 1$
- Ερώτημα: πώς γίνεται η ανάθεση των υπογραφών στις καταστάσεις;

Ανάθεση υπογραφών: παράδειγμα (7,4) Hamming SECSED code

- Κώδικας με m = 4 bit δεδομένα → ελάχιστος αριθμός ψηφίων ισοτιμίας c = 3 (m+c+1 = 8)
- Κωδική λέξη: a3,a2,a1,a0,p2,p1,p0
  - a3,a2,a1,a0: ψηφία δεδομένων
  - p2,p1,p0: ψηφία ισοτιμίας

Κατάσταση	Σφάλματα στα bit ισοτιμίας
No error	None
Bit 1 (p0) error	p0
Bit 2 (p1) error	p1
Bit 3 (p2) error	p2
Bit 4 (a0) error	p0, p1
Bit 5 (a1) error	p0, p2
Bit 6 (a2) error	p1, p2
Bit 7 (a3) error	p0, p1, p2

Ανάθεση υπογραφών: παράδειγμα (7,4) Hamming SECSED code

- Εάν αποτύχει μόνο το p0
  - Υπάρχει σφάλμα στο bit1 (p0)
- Το p0 θα αποτύχει εάν υπάρχει σφάλμα και σε άλλα bit, π.χ. bit4 (a0)
- p0 καλύπτει τα bit 1,4,5,7
  - p0 = a0 ⊕ a1 ⊕ a3
- p1 καλύπτει τα bit 2,4,6,7
  - p1 = a0 ⊕ a2 ⊕ a3
- p2 καλύπτει τα bit 3,5,6,7
  - p2 = a1 ⊕ a2 ⊕ a3

Κατάσταση	Σφάλματα στα bit ισοτιμίας
No error	None
Bit 1 (p0) error	p0
Bit 2 (p1) error	p1
Bit 3 (p2) error	p2
Bit 4 (a0) error	p0, p1
Bit 5 (a1) error	p0, p2
Bit 6 (a2) error	p1, p2
Bit 7 (a3) error	p0, p1, p2

Σύνδρομα: υπολογισμός

- Παράδειγμα:
  - a3a2a1a0 = 1100 → p2p1p0 = 001
- Σφάλμα: 1100001 → 1000001
- Υπολογισμός ισοτιμίας: p2p1p0 = 111
- Σύνδρομο = Διαφορά των bit ισοτιμίας (xor ανά bit): 110
- Επομένως, το σφάλμα είναι στο a2 και τα σωστά δεδομένα είναι a3a2a1a0 = 1100

Κατάσταση	Σφάλματα στα bit ισοτιμίας	Σύνδρομο
No error	None	0 0 0
Bit 1 (p0) error	p0	0 0 1
Bit 2 (p1) error	p1	0 1 0
Bit 3 (p2) error	p2	1 0 0
Bit 4 (a0) error	p0, p1	0 1 1
Bit 5 (a1) error	p0, p2	1 0 1
Bit 6 (a2) error	p1, p2	1 1 0
Bit 7 (a3) error	p0, p1, p2	1 1 1

$$p0 = a0 \oplus a1 \oplus a3$$

$$p1 = a0 \oplus a2 \oplus a3$$

$$p2 = a1 \oplus a2 \oplus a3$$

Σύνδρομα: υπολογισμός

- Τα bit ισοτιμίας τοποθετούνται στην κωδική λέξη στις θέσεις που είναι δύναμη του 2, π.χ. 1, 2, 4, κοκ.
- Ο υπολογισμός του συνδρόμου δίνει τον δείκτη του ψηφίου που έχει το σφάλμα
- Οι εξισώσεις των bit ισοτιμίας προκύπτουν από τον πίνακα

$$p0 = a0 \oplus a1 \oplus a3$$

$$p1 = a0 \oplus a2 \oplus a3$$

$$p2 = a1 \oplus a2 \oplus a3$$

Bit	7	6	5	4	3	2	1
Code	a3	a2	a1	p2	a0	p1	p0
p0	1	0	1	0	1	0	1
p1	1	1	0	0	1	1	0
p2	1	1	1	1	0	0	0

Σύνδρομα: χρήση πινάκων

- Τα σύνδρομα μπορούν να υπολογιστούν απευθείας από την κωδική λέξη a3 a2 a1 p2 a0 p1 p0 με πράξεις πινάκων
- Οι προσθέσεις είναι mod 2 (xor)

$$\begin{array}{c} \begin{matrix} a3 & a2 & a1 & p2 & a0 & p1 & p0 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{bmatrix} \alpha 3 \\ \alpha 2 \\ \alpha 1 \\ p2 \\ a0 \\ p1 \\ p0 \end{bmatrix} = \begin{bmatrix} s0 & s1 & s2 \end{bmatrix}$$

Μήτρα ελέγχου
(check matrix)

p0 = a0 ⊕ a1 ⊕ a3

p1 = a0 ⊕ a2 ⊕ a3

p2 = a1 ⊕ a2 ⊕ a3

Βελτίωση της δυνατότητας ανίχνευσης

- Ο προηγούμενος κώδικας μπορεί να ανιχνεύσει/διορθώσει ένα μονό σφάλμα αλλά δεν μπορεί να ανιχνεύσει ένα διπλό σφάλμα
- Παράδειγμα:
  - a3 a2 a1 p2 a0 p1 p0 = 11000001
  - Σφάλμα: 10100001
- Σύνδρομο s2s1s0 = 011
  - Σφάλμα είναι στη θέση 3 (a0)
- Προσθήκη έξτρα bit ελέγχου p3:
  - ισοτιμία όλων των bit δεδομένων a3-a0 και ελέγχου p2-p0
- (8,4) Hamming SECDED code

$p0 = a0 \oplus a1 \oplus a3$

$p1 = a0 \oplus a2 \oplus a3$

$p2 = a1 \oplus a2 \oplus a3$

(8,4) Hamming SECDED code

- Ένα μονό λάθος:
  - Θα επηρεάσει την συνολική ισοτιμία (s3 = 1)
  - Τα bits s2 s1 s0 δείχνουν τη θέση του σφάλματος
- Ένα διπλό λάθος:
  - s3 = 0 και τα s2 s1 s0 ≠ 0

p0 = a0 ⊕ a1 ⊕ a3

p1 = a0 ⊕ a2 ⊕ a3

p2 = a1 ⊕ a2 ⊕ a3

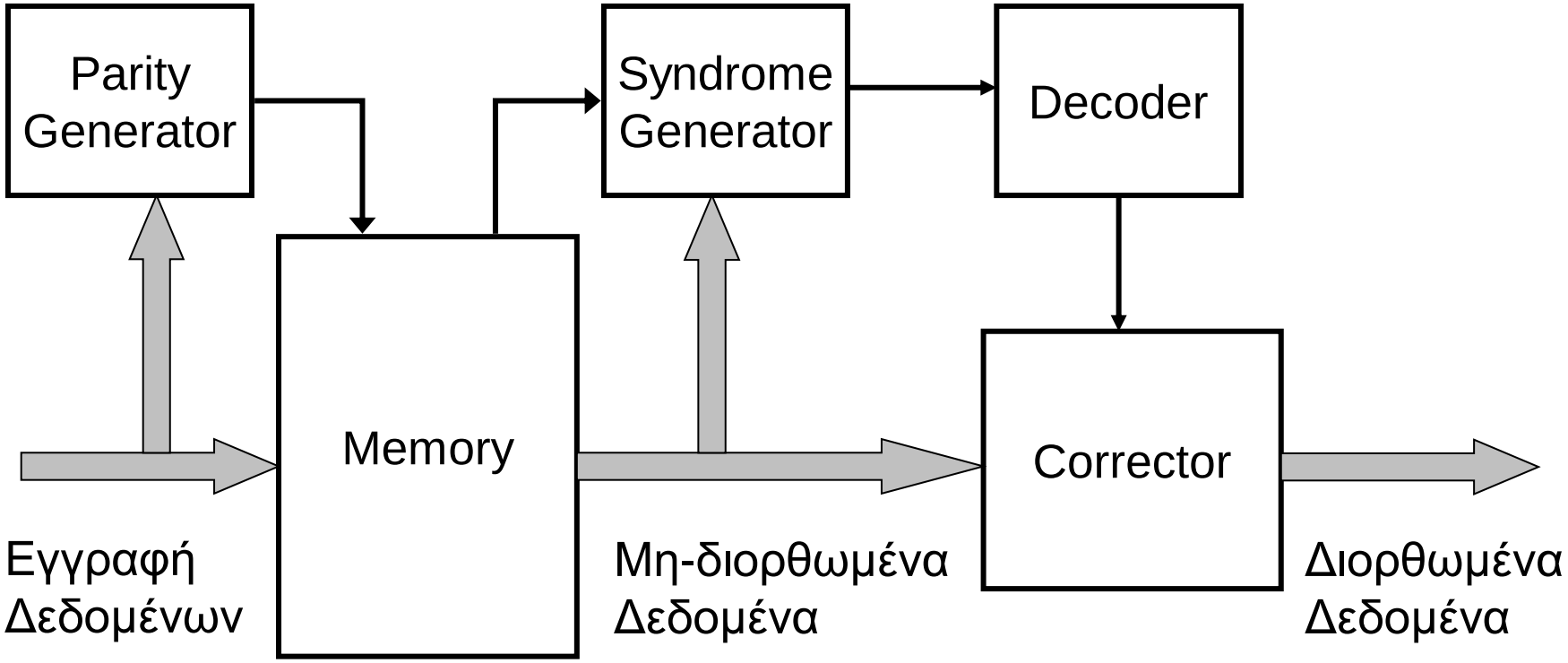
p3 = p0 ⊕ p1 ⊕ p2 ⊕ a0 ⊕ a1 ⊕ a2 ⊕ a3

p3	a3	a2	a1	p2	a0	p1	p0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
0	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1

Μήτρα ελέγχου
(check matrix)

$$\begin{bmatrix} p3 \\ a3 \\ a2 \\ a1 \\ p2 \\ a0 \\ p1 \\ p0 \end{bmatrix} = \begin{bmatrix} s0 & s1 & s2 & s3 \end{bmatrix}$$

Διόρθωση σφαλμάτων στις μνήμες



Κώδικες σφαλμάτων ριπής (burst error codes)

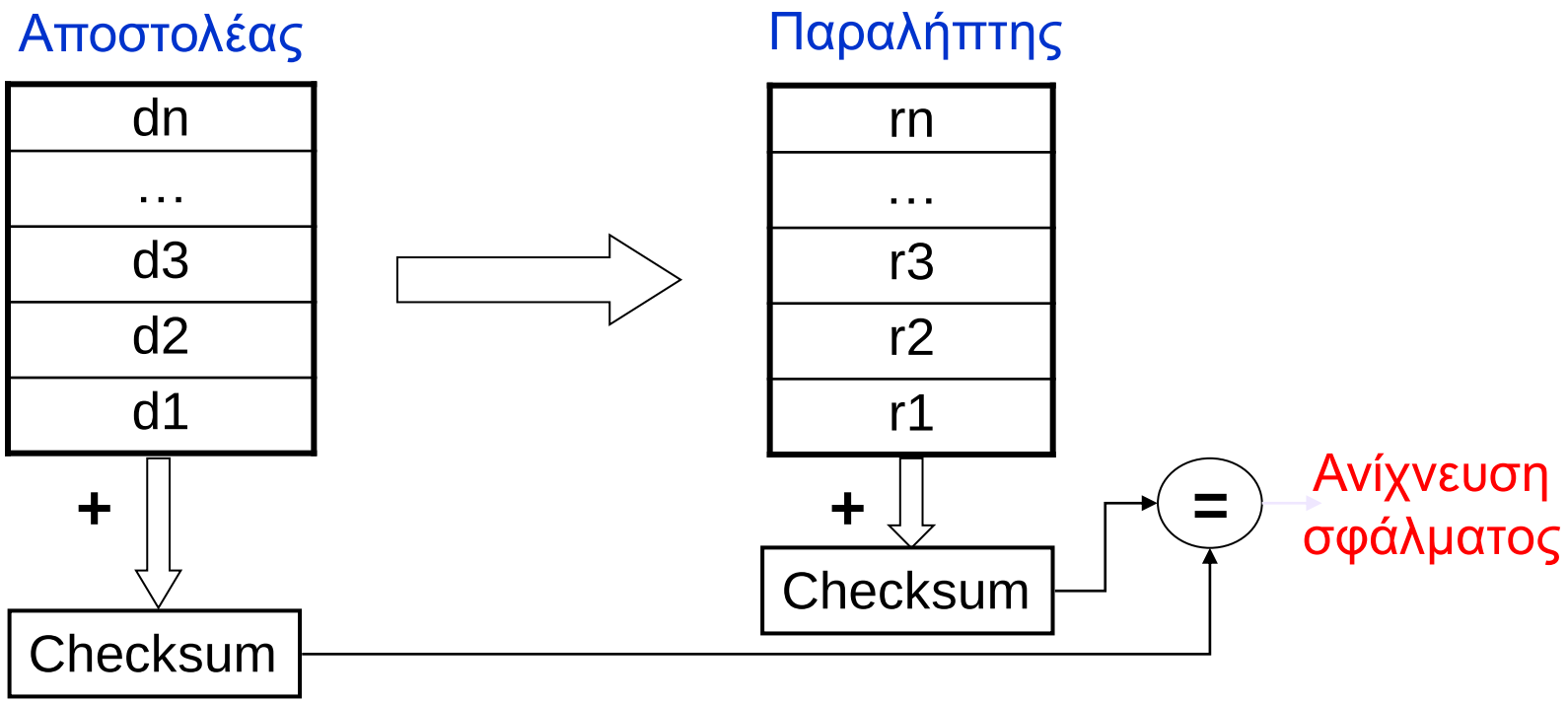
- Σε μερικές περιπτώσεις έχουμε εμφάνιση σφαλμάτων ριπής
  - Όχι μονά σφάλματα ή ανεξάρτητα μεταξύ τους πολλαπλά σφάλματα
  - Απαιτούνται κώδικες σφαλμάτων ριπής για την ανίχνευσή τους
- Εφαρμογές
  - Μέσα αποθήκευσης: μαγνητικοί δίσκοι, οπτικοί δίσκοι
  - Δίκτυα
- Αιτίες ελαττωμάτων
  - Ατέλειες ή σωματίδια σκόνης στην επιφάνεια του δίσκου
  - Θόρυβος στις κεφαλές ανάγνωσης/εγγραφής
  - Ηλεκτρομαγνητικές παρεμβολές στην μετάδοση των δεδομένων
- Όσο αυξάνεται η πυκνότητα αποθήκευσης ή η ταχύτητα μετάδοσης των δεδομένων, αυξάνεται ο αριθμός των bit που επηρεάζονται από σφάλματα ριπής

Κώδικες σφαλμάτων ριπής (burst error codes)

- Κώδικες αθροίσματος ελέγχου (checksum codes)
- Κυκλικοί κώδικες (cyclic codes)

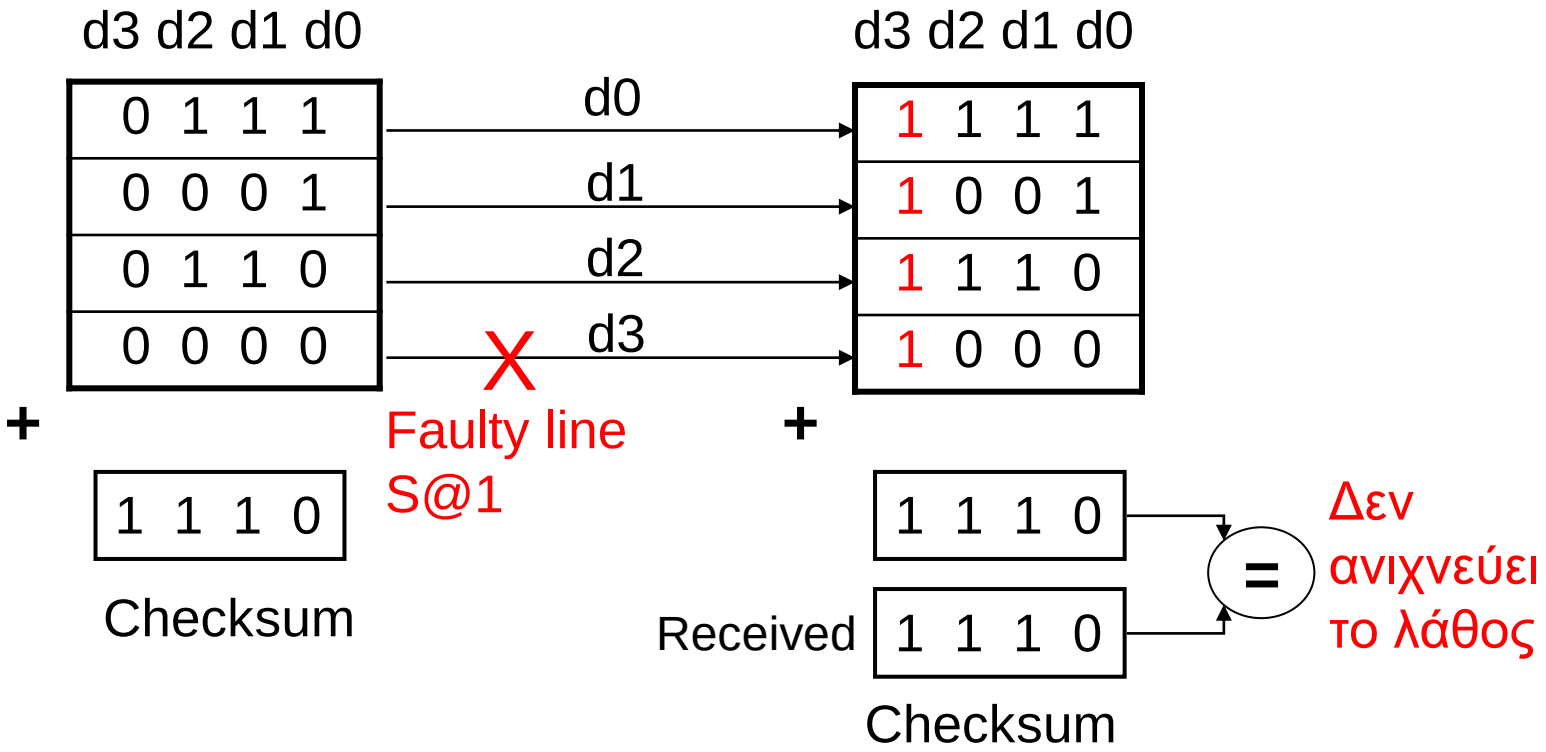
Άθροισμα ελέγχου – Βασική ιδέα

- Ο αποστολέας προσθέτει τα μπλοκ δεδομένων που πρόκειται να μεταδοθούν και μεταδίδει και το άθροισμά τους (checksum)
- Ο παραλήπτης προσθέτει τα μπλοκ δεδομένων που λαμβάνει και συγκρίνει το αποτέλεσμα με το άθροισμα που έλαβε



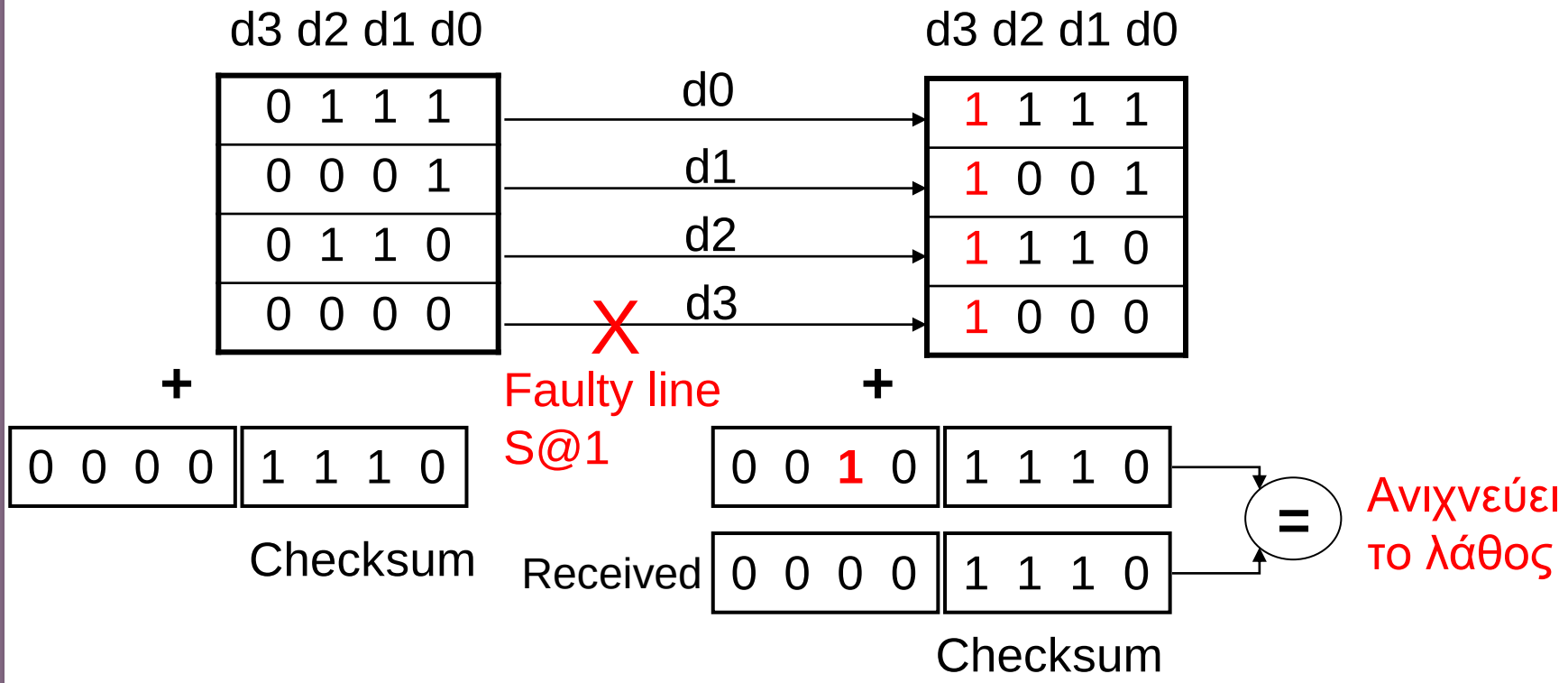
Άθροισμα ελέγχου μονής ακρίβειας (single precision checksum)

- Πρόσθεση modulo 2^d (d = μήκος της λέξης)
- Το κρατούμενο εξόδου (υπερχείλιση) αγνοείται



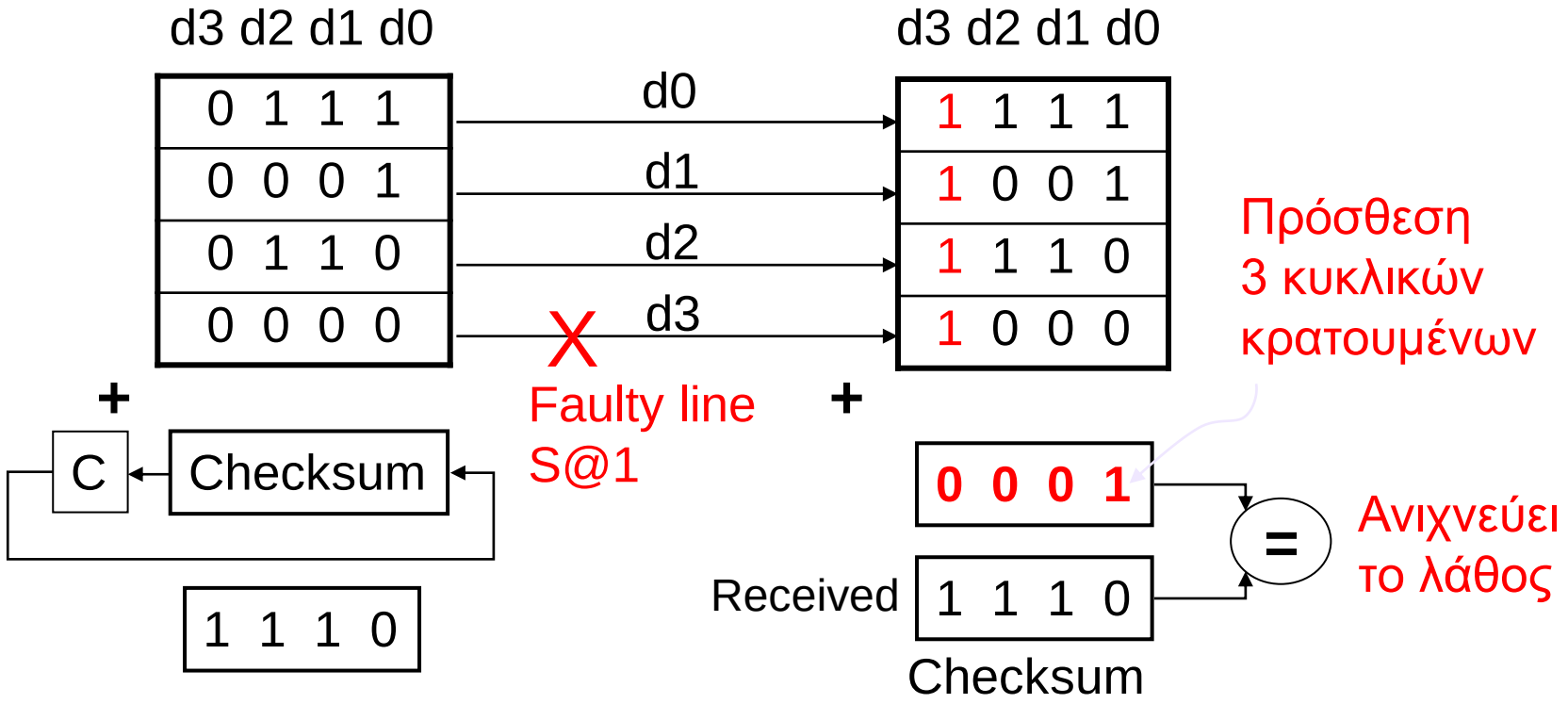
Άθροισμα ελέγχου διπλής ακρίβειας (double precision checksum)

- Πρόσθεση modulo 2^{2d} (d = μήκος της λέξης)
- Το πρόβλημα της υπερχείλισης παραμένει, αλλά πρέπει να συμβεί υπερχείλιση σε μία 2d-bit λέξη



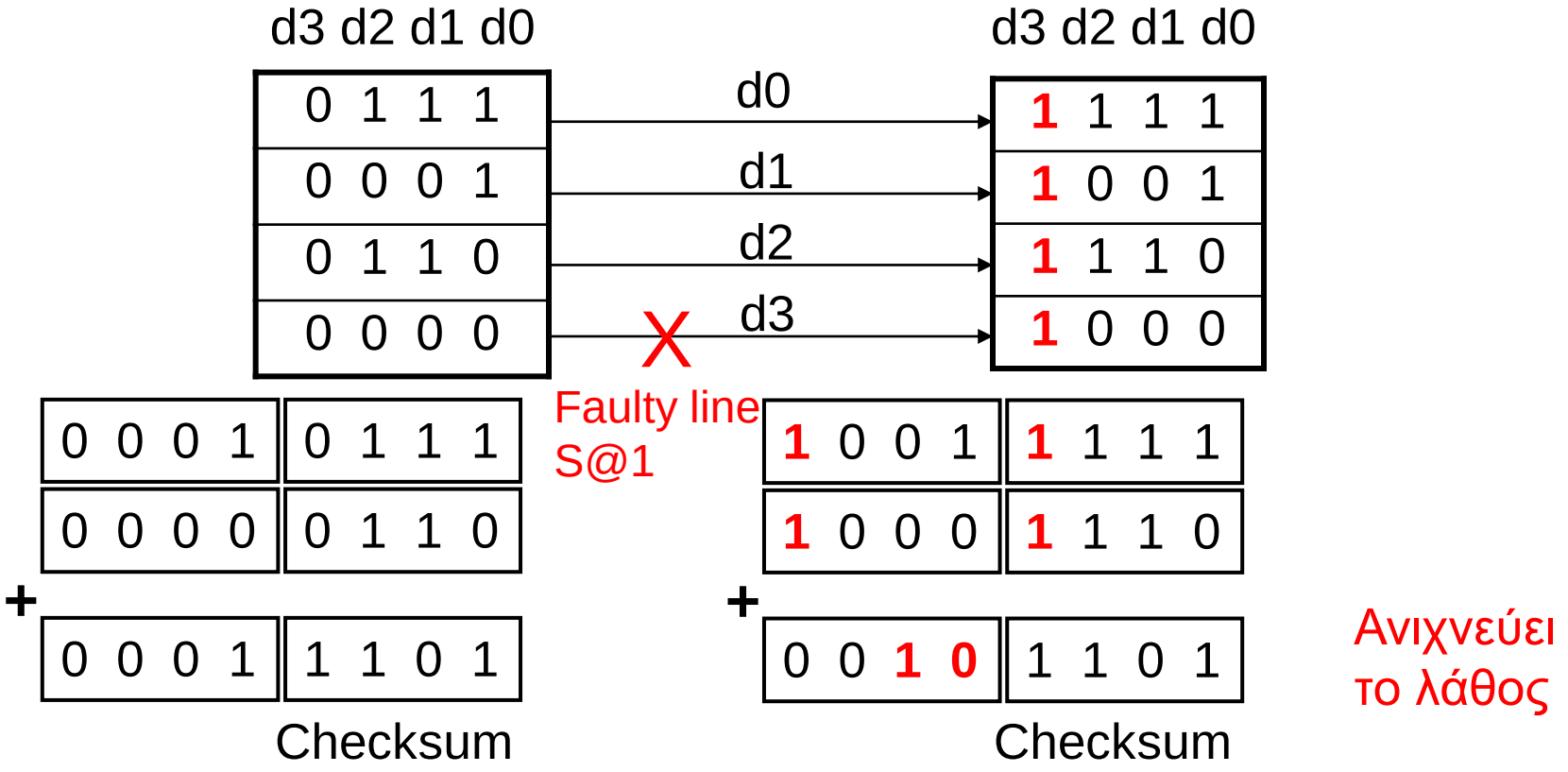
Άθροισμα ελέγχου υπολοίπου (residue checksum)

- Εκτελεί πρόσθεση modulo 2^d, (όπως στη μονή ακρίβεια)
- Το κρατούμενο εξόδου δεν αγνοείται, αλλά προστίθεται στο άθροισμα σαν κυκλικό κρατούμενο (end-around carry)



Άθροισμα ελέγχου Honeywell (Honeywell checksum)

- Συνενώνει τις λέξεις σε ζευγάρια και εκτελεί πρόσθεση modulo 2^{2d}
- Προφυλάσσει από περιπτώσεις όπου τα λάθη συμβαίνουν στην ίδια θέση της λέξης



Κυκλικοί κώδικες (cyclic codes)

- Κωδικοποίηση
  - Πολλαπλασιάζουμε (modulo 2) τη λέξη δεδομένων με ένα σταθερό αριθμό
  - Γινόμενο: κωδική λέξη
- Αποκωδικοποίηση
  - Διαιρούμε (module 2) με τον ίδιο σταθερό αριθμό
  - Εάν το υπόλοιπο είναι $\neq 0 \Rightarrow$ σφάλμα
- Αποκαλούνται κυκλικοί κώδικες γιατί:
  - Εάν $(a_{n-1},a_{n-2},\ldots, a_1,a_0)$ είναι κωδική λέξη τότε και η κυκλική της ολίσθηση $(a_0,a_{n-1},\ldots, a_2,a_1)$ είναι κωδική λέξη
- Παράδειγμα: 5-bit κυκλικός κώδικας
  - {00000,10000,01000,00100,00010,00001,11000,01100, 00110,00011,10001,11110,01111,10111,11011,11101}

Κυκλικοί κώδικες - θεωρία

- k : # bit της λέξης δεδομένων
- n : # bit της κωδικής λέξης
- Η κωδική λέξη προκύπτει πολλαπλασιάζοντας τη λέξη δεδομένων (k-bit) με έναν πολλαπλασιαστή (n-k+1-bit)
  - Η λέξη δεδομένων και ο πολλαπλασιαστής αναπαριστώνται σαν πολυώνυμα
- Τα 1 και 0 του πολλαπλασιαστή (n-k+1-bit) είναι συντελεστές ενός πολυωνύμου με βαθμό n-k
  - παράγων πολυώνυμο (generating polynomial)
- Παράδειγμα: πολλαπλασιαστής 11001
  - Παράγων πολυώνυμο :

    $G(x) = \mathbf{1}\, X^4 + \mathbf{1}\, X^3 + \mathbf{0}\, X^2 + \mathbf{0}\, X^1 + \mathbf{1}\, X^0 = X^4 + X^3 + 1$

Πράξεις πολυωνύμων

- Χρήση αριθμητικής modulo 2
- Συντελεστές του πολυωνύμου 0 ή 1
- Πρόσθεση και αφαίρεση (modulo 0) είναι ίδιες

Πολλαπλασιασμός

$$
\begin{array}{rl}
 & X^4 + X^3 + X^2 + 1 \\
\times & \qquad\qquad X^3 + X \\
\hline
 & X^5 + X^4 + X^3 + X \\
+ & X^7 + X^6 + X^5 + \qquad X^3 \\
\hline
 & X^7 + X^6 + \qquad X^4 + \qquad X
\end{array}
$$

Διαίρεση

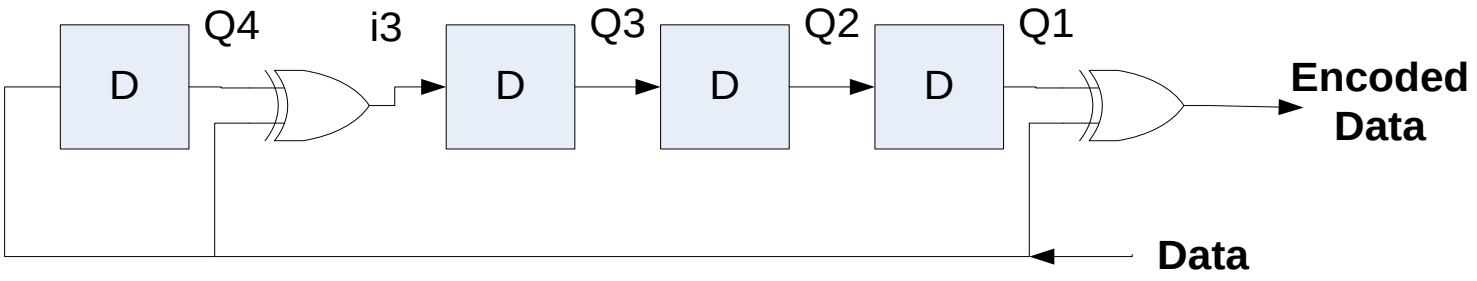
$$
\begin{array}{rl|l}
 & X^5 + X^4 & X^4 + X^3 + X^2 + 1 \\
\hline
- & X^5 + X^4 + X^3 + X & X \\
 & \qquad\qquad X^3 + X & \text{Πηλίκο} \\
 & \text{Υπόλοιπο} &
\end{array}
$$

(n,k) κυκλικοί κώδικες

- Ένας (n,k) κυκλικός κώδικας
  - μέγεθος κωδικής λέξης n-bit
  - παράγων πολυώνυμο βαθμού n-k
- Ανιχνεύει όλα τα μονά σφάλματα και ριπές σφαλμάτων (σε γειτονικά ψηφία) μικρότερες από n-k
- Το παράγων πολυώνυμο ενός (n,k) κυκλικού κώδικα πρέπει να είναι παράγοντας του $X^n + 1$
- Παράδειγμα: $X^4 + X^3 + 1$ παράγοντας του $X^{15} + 1$
  - Παράγων πολυώνυμο του (15,11) κυκλικού κώδικα
- Για έναν 5-bit κώδικα, (X+1) παράγων πολυώνυμο
  - $X^5 + 1 = (X + 1)(X^4 + X^3 + X^2 + X + 1)$
  - Πολλαπλασιάζοντας τα {0000, …,1111} με (X+1) παράγουμε όλες τις κωδικές λέξεις ενός (5,4) κώδικα

Κωδικοποίηση: υλοποίηση στο υλικό

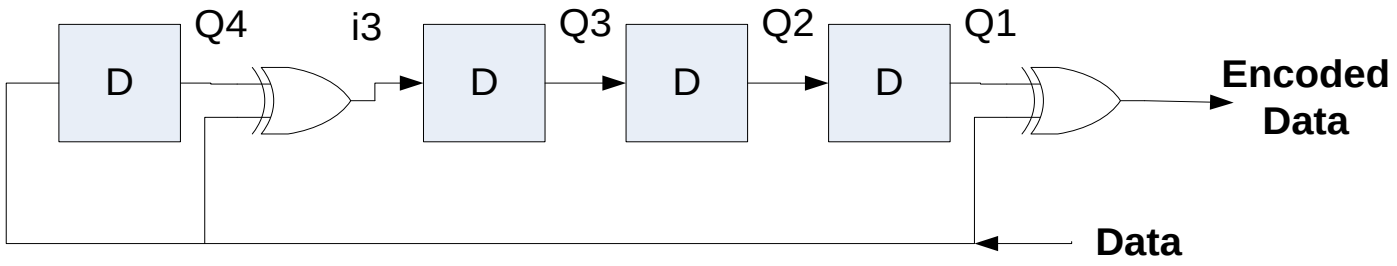
- Ο πολλαπλασιασμός υλοποιείται με καταχωρητές ολίσθησης (shift registers) και πύλες XOR
- Παράδειγμα:
  - (15,11) κυκλικός κώδικας
  - $X^4 + X^3 + 1$ παράγων πολυώνυμο



Πολλαπλασιασμός

- Το 5ο bit του γινομένου είναι το άθροισμα (modulo 2) του αντίστοιχου bit του πολλαπλασιαστέου ολισθημένου κατά 0, 3 και 4 θέσεις
- Ο πολλαπλασιαστέος εισάγεται σειριακά και ολισθαίνει
- Το άθροισμα (modulo 2) υλοποιείται από τις πύλες XOR
- Ο κυκλικός κώδικας είναι μη-διαχωρίσιμος
  - Τα ψηφία δεδομένων και ελέγχου εντός της κωδικής λέξης 110000100011101 δεν είναι διαχωρίσιμα

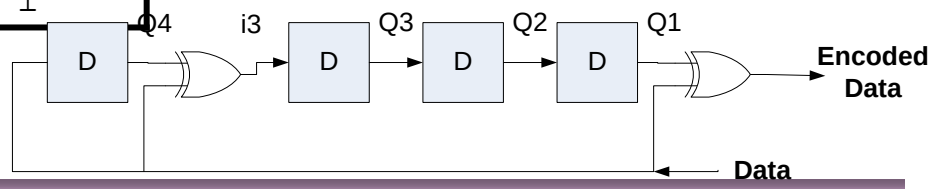
```
   10001100101
×      11001
-------------
   10001100101
  00000000000
 00000000000
10001100101
+10001100101
-------------
110000100011101
```



Πολλαπλασιασμός

Clock	Input Data	Q4	i3	Q3Q2Q1	Encoded Data
1	1	0	1	0 0 0	1
2	0	1	1	1 0 0	0
3	1	0	1	1 1 0	1
4	0	1	1	1 1 1	1
5	0	0	0	1 1 1	1
6	1	0	1	0 1 1	0
7	1	1	0	1 0 1	0
8	0	1	1	0 1 0	0
9	0	0	0	1 0 1	1
10	0	0	0	0 1 0	0
11	1	0	1	0 0 1	0
12	0	1	1	1 0 0	0
13	0	0	0	1 1 0	0
14	0	0	0	0 1 1	1
15	0	0	0	0 0 1	1

```
   10001100101
 ×        11001
 ------------------
   10001100101
  00000000000
 00000000000
10001100101
+10001100101
 ------------------
1100001000111101
```



Διαίρεση

- Διαίρεση της κωδικής λέξης με 11001
- Υπόλοιπο = 0 ⇒ δεν ανιχνεύει σφάλμα
- Εάν συμβεί σφάλμα, π.χ. 1100001001**1**11101 ⇒ υπόλοιπο ≠ 0

```
1100001000111101 | 11001
-----------------|------
11001            | 10001100101
    10100
    11001
     11010
     11001
      11111
      11001
       11001
       11001
Υπόλοιπο 00000
```

```
1100001001111101 | 11001
-----------------|------
11001            | 10001100110
    10100
    11001
     11011
     11001
      10111
      11001
       11100
       11001
Υπόλοιπο 01011
```

Τριπλά σφάλματα

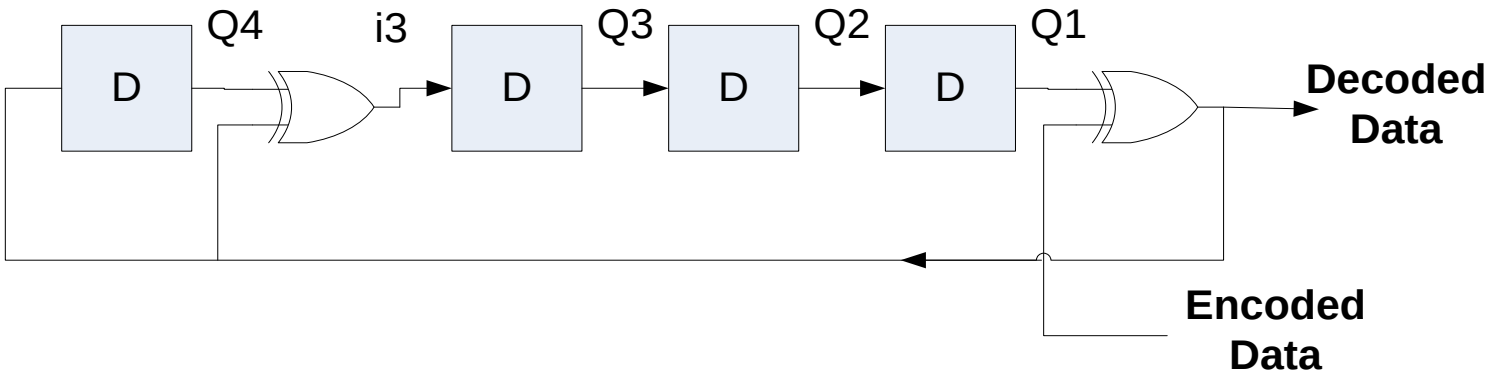
- 1100001110101**0**1 αντί για 1100001000111**0**1:
  - Υπόλοιπο = 0 ⇒ δεν ανιχνεύει σφάλμα (λανθασμένα)
- 1100000**11**011101 (σφάλμα ριπής)
  - Υπόλοιπο ≠ 0 ⇒ ανιχνεύει σφάλμα

```
1100001110101 | 11001
11001         | 10001101101
-----
 10111
 11001
 -----
  11100
  11001
  -----
   10110
   11001
   -----
    11111
    11001
    -----
     11001
     11001
     -----
Υπόλοιπο 00000
```

```
1100000011011101 | 11001
11001            | 10001110011
-----
 10011
 11001
 -----
  10100
  11001
  -----
   11011
   11001
   -----
    10110
    11001
    -----
     11111
     11001
     -----
Υπόλοιπο 00110
```

Αποκωδικοποίηση: υλοποίηση στο υλικό

- Η διαίρεση υλοποιείται με πολλαπλασιασμό (καταχωρητές ολίσθησης & XOR) και ανάδραση (feedback)
- Αρχικά παράγει τα 11 bit του πηλίκου (ψηφία δεδομένων) και στη συνέχεια τα 4 bit του υπολοίπου (ψηφία ελέγχου)

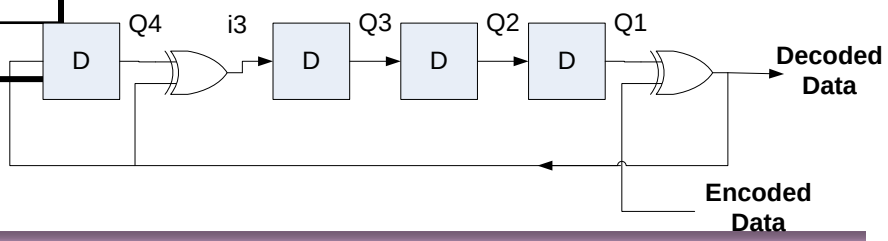


Διαίρεση

Clock	Encoded Data	Q4	I3	Q3Q2Q1	Decoded Data
1	1	0	1	0 0 0	1
2	0	1	1	1 0 0	0
3	1	0	1	1 1 0	1
4	1	1	1	1 1 1	0
5	1	0	0	1 1 1	0
6	0	0	1	0 1 1	1
7	0	1	0	1 0 1	1
8	0	1	1	0 1 0	0
9	1	0	0	1 0 1	0
10	0	0	0	0 1 0	0
11	0	0	1	0 0 1	1
12	0	1	1	1 0 0	0
13	0	0	0	1 1 0	0
14	1	0	0	0 1 1	0
15	1	0	0	0 0 1	0

Πηλίκο (data bits)

Υπόλοιπο (check bits)



Ανίχνευση σφαλμάτων ριπής

- Στους (n,k) κυκλικούς κώδικες ο αριθμός των ψηφίων ελέγχου n-k είναι «ανεξάρτητος» του αριθμού των ψηφίων δεδομένων k
  - Μπορούμε να αυξήσουμε το k χωρίς να χρειάζεται να αυξήσουμε τον βαθμό n-k του παράγων πολυωνύμου
    - χωρίς να αυξήσουμε την πολυπλοκότητα των κυκλωμάτων κωδικοποίησης και αποκωδικοποίησης
  - Όσο αυξάνεται το k η δυνατότητα του κώδικα να ανιχνεύει σφάλματα ριπής μειώνεται
    - Ανιχνεύει σφάλματα ριπής μήκους n-k
- Κυκλικοί κώδικες τύπου (16+k,k) χρησιμοποιούνται και εγγυώνται την ανίχνευση σφαλμάτων ριπής μήκους 16 ή μικρότερο
- Οι πιο διαδεδομένοι κώδικες:
- CRC-16 (16-bit Cyclic Redundancy Code)
  - Παράγων πολυώνυμο: $G(X) = X^{16} + X^{15} + X^2 + 1$
- CRC-CCITT
  - Παράγων πολυώνυμο: $G(X) = X^{16} + X^{12} + X^5 + 1$