ΑΣΚΗΣΕΙΣ 11 / 1 / 2008

1. Το αλφάβητο επικοινωνίας δύο σταθμών Α και Β αποτελείται από τις 8 διαφορετικές δυαδικές λέξεις $\beta_2\beta_1\beta_0$. Για τη προστασία από λάθη, κάθε μήνυμα μεταξύ Α και Β κωδικοποιείται σαν $\beta_2\beta_1\beta_0\beta'_2\beta'_1\beta'_0P$ όπου β'_i το συμπληρωματικό του β_i και P το ψηφίο άρτιας ισοτιμίας της λέξης $\beta_2\beta_1\beta_0$.
   - Τι ικανότητες ανίχνευσης / διόρθωσης λαθών μας παρέχει αυτός ο κώδικας ? Απαιτείται απόδειξη.
   - Εξηγείστε πως ο παραλήπτης του κωδικοποιημένου μηνύματος θα πραγματοποιήσει ανίχνευση / διόρθωση λαθών (ανάλογα με την απάντησή σας στο προηγούμενο ερώτημα). *Μπορείτε να δώσετε flowchart / λογικό διάγραμμα σε υλικό για να περιγράψετε τη διαδικασία εφόσον αυτό σας βολεύει.*
   - Είναι η κωδικοποίηση που επέλεξαν ο Α και ο Β η καλύτερη ? Αν όχι, υποδείξτε καλύτερη. Κωδικοποιείστε τη λέξη $\beta_2\beta_1\beta_0$=110, με τη δικιά σας πρόταση.

2. Το ακόλουθο μήνυμα που έφτασε σε σας είναι κωδικοποιημένο σε κώδικα modified Hamming. Ελέγξτε την ορθότητά του και εφόσον σας το επιτρέπουν οι δυνατότητες του κώδικα, διορθώστε τυχόν λάθη του. Υποθέστε άρτια ισοτιμία κωδικοποίησης.

   (Πιο σημαντικό ψηφίο) → 0 0 1 1 1 1 0 1 1 0 ← (Ψηφίο Ισοτιμίας)

3. Λάβατε το ακόλουθο δυαδικό μήνυμα :

   πιο σημαντικό ψηφίο -> 1 0 0 0 0 1 0 1 0 0 1 1 1 <- λιγότερο σημαντικό ψηφίο

   Ο αποστολέας του μηνύματος το έχει κωδικοποιήσει κατά Hamming έχοντας χρησιμοποιήσει τα ΕΛΑΧΙΣΤΑ απαιτούμενα δυαδικά ψηφία ελέγχου. Ελέγξτε την ορθότητα του μηνύματος, διορθώστε τυχόν απλό λάθος που έχει συμβεί και εξάγετε την αρχική πληροφορία. Ποια είναι τα group ισοτιμίας τα οποία χρησιμοποιήσατε ?

4. Σας δίνεται ένα κύκλωμα με 4 εισόδους και 2 εξόδους. Προτείνετε δύο διαφορετικές εκδοχές εφαρμογής της υβριδικής τεχνικής sift-out modular redundancy για το κύκλωμα αυτό. Εξηγείστε τα πλεονεκτήματα και τα μειονεκτήματα κάθε εκδοχής, δίνοντας ποσοτικά αποτελέσματα όπου αυτό είναι δυνατό.

5. Εστω ένας αθροιστής κ δυαδικών ψηφίων, που προσθέτει τους $A=\alpha_{\kappa-1}\alpha_{\kappa-2}\ldots\alpha_0$, $B=\beta_{\kappa-1}\beta_{\kappa-2}\ldots\beta_0$ και το κρατούμενο εισόδου c_{-1} και παράγει το άθροισμα $=\sigma_{\kappa-1}\sigma_{\kappa-2}\ldots\sigma_0$ και το κρατούμενο εξόδου $c_{\kappa-1}$.
   - Δείξτε ότι $\sigma_{\kappa-1}\oplus\sigma_{\kappa-2}\oplus\ldots\oplus\sigma_0\oplus c_{\kappa-1} = \alpha_{\kappa-1}\oplus\alpha_{\kappa-2}\oplus\ldots\oplus\alpha_0\oplus\beta_{\kappa-1}\oplus\beta_{\kappa-2}\oplus\ldots\oplus\beta_0\oplus c_{\kappa-1}\oplus\ldots\oplus c_{\kappa-2}\oplus\ldots\oplus c_0\oplus c_{-1}$, όπου $\oplus$, η συνάρτηση αποκλειστικής διάζευξης (XOR) και $c_{\kappa-2}$ έως c_0 τα κρατούμενα μεταξύ των βαθμίδων του αθροιστή.
   - Στηριζόμενοι στο προηγούμενο ερώτημα, προτείνετε μια μέθοδο προστασίας του αθροιστή από **μονό αριθμό λαθών** και δείξτε το κύκλωμα για κ=4. Υπάρχει περίπτωση ένα **<u>απλό stuck-at fault</u>** να μην ανιχνευθεί με τη μέθοδο αυτή ?

6. Για την κωδικοποίηση σε κυκλικό κώδικα και με δεδομένο το γεννήτορα πολυώνυμο $G(X)=1+x+x^4$ σχεδιάστε : 1) το κύκλωμα κωδικοποίησης και 2) το κύκλωμα αποκωδικοποίησης. Δείξτε την κωδικοποίηση για την πληροφορία MSB -> 0111 <- LSB σε διαχωρίσιμη και μη μορφή.

7. Ποιος είναι ο ελάχιστος αριθμός δυαδικών ψηφίων ελέγχου (check bits) που πρέπει να προστεθούν σε μήνυμα μήκους κ δυαδικών ψηφίων ώστε να υπάρχει η δυνατότητα διόρθωσης ενός ή δύο απλών λαθών ? Αποδείξτε τον ισχυρισμό σας. Πόση είναι η επιβάρυνση που προκαλεί ο κώδικας όταν κ=7 ?

8. Σχεδιάστε με όση λεπτομέρεια κρίνετε απαραίτητη έναν προσθετή 8 δυαδικών ψηφίων χρησιμοποιώντας τη τεχνική RESWO. Υπολογίστε την επιβάρυνση σε υλικό που απαιτεί η υιοθέτηση αυτής της τεχνικής θεωρώντας ότι κάθε πολυπλέκτης από δύο δυαδικά ψηφία σε ένα απαιτεί υλικό ίσο με το 1% του προσθετή, η αποθήκευση ενός δυαδικού ψηφίου απαιτεί 2% του υλικού του προσθετή και η σύγκριση δύο δυαδικών ψηφίων απαιτεί 1% του υλικού του προσθετή. Τέλος, εντοπίστε τυχόν απλά λάθη στην RESWO δομή τα οποία ΔΕΝ ανιχνεύονται και προτείνετε βελτιώσεις / αλλαγές.

9. Σε ένα RNS με βάση τα υπόλοιπα {13, 11, 7, 5, 2} ποιο είναι το εύρος της αναπαράστασης ? Αναπαραστήστε τους 137_{10} και 26_{10}. Δείξτε τη διαδικασία πρόσθεσής τους. Πως μπορείτε να αυξήσετε τις ικανότητες ανίχνευσης λαθών σε αυτό το σύστημα με το **ελάχιστο** δυνατό κόστος ? Δώστε αναλυτικά τις νέες αναπαραστάσεις και δείξτε την νέα διαδικασία πρόσθεσης.

10. Η κρυφή (cache) μνήμη είναι ένα στοιχείο που προστίθεται σε ένα υπολογιστικό σύστημα **αποκλειστικά και μόνο για την αύξηση της απόδοσής του και όχι για την ορθή λειτουργία του. (Αρα ακόμα και το να την απενεργοποιήσω, είναι μια μορφή προστασίας από τα κατασκευαστικά της ελαττώματα, που όμως επιφέρει και την περισσότερη επιβάρυνση στην απόδοση του συστήματός μου).** Ως εκ τούτου κάθε τεχνική προστασίας της από κατασκευαστικά ελαττώματα **πρέπει να επιφέρει την ελάχιστη υποβάθμιση της απόδοσης του συστήματος.** Με βάση τα παραπάνω, θεωρώντας μια κρυφή μνήμη άμεσης οργάνωσης (direct mapped cache), η οποία αποτελείται από τη μνήμη δεδομένων, τη μνήμη ετικετών και το συγκριτή ετικετών, υποδείξτε έναν κατά τη γνώμη σας αποδοτικό τρόπο προστασίας από κατασκευαστικά (μόνιμα) ελαττώματα.

(Υποδείξεις :

α) Τα ελαττώματα μπορεί να είναι απλά ή πολλαπλά όπως και τα σφάλματα που επιφέρουν.

β) Κάθε προσπάθεια εφαρμογής κώδικα ή πλεονασμού μπορεί να μην είναι δόκιμη, αφού επιβαρύνουν το χρόνο προσπέλασης της κρυφής μνήμης ακόμα και σε πλαίσια που δεν έχουν υποστεί ελαττώματα).

11. Ποια τεχνική πλεονασμού θα προτείνατε σε συνδυασμό με τεχνική αυτοκάθαρσης ώστε να διαχωρίζετε τις περιπτώσεις μόνιμων και παροδικών σφαλμάτων ? Εξηγείστε τα πλεονεκτήματα ενός τέτοιου συνδυασμού.

12. Πόσα λανθασμένα modules μπορούν να ανεχθούν οι τεχνικές :

- 5MR ?
- Standby sparing με 5 συνολικά modules ?
- Αυτοκάθαρση με 5 συνολικά modules ?

13. Υποθέστε ότι η πληροφορία μας αποτελείται από λέξεις των 4 δυαδικών ψηφίων, εξαιρουμένης της πληροφορίας 0000. Πόσα δυαδικά ψηφία απαιτεί η κωδικοποίησή της σε κώδικα Berger ?

14. Δίνεται ο ακόλουθος κώδικας :

0 0 0 0 0 0
0 0 0 0 1 1
0 0 0 1 1 0
0 0 1 0 0 1
0 0 1 1 0 0
0 0 1 1 1 1
0 1 0 0 1 0
0 1 0 1 0 1
0 1 1 0 0 0
0 1 1 0 1 1
0 1 1 1 1 0
1 0 0 0 0 1
1 0 0 1 0 0
1 0 0 1 1 1
1 0 1 0 1 0
1 0 1 1 0 1

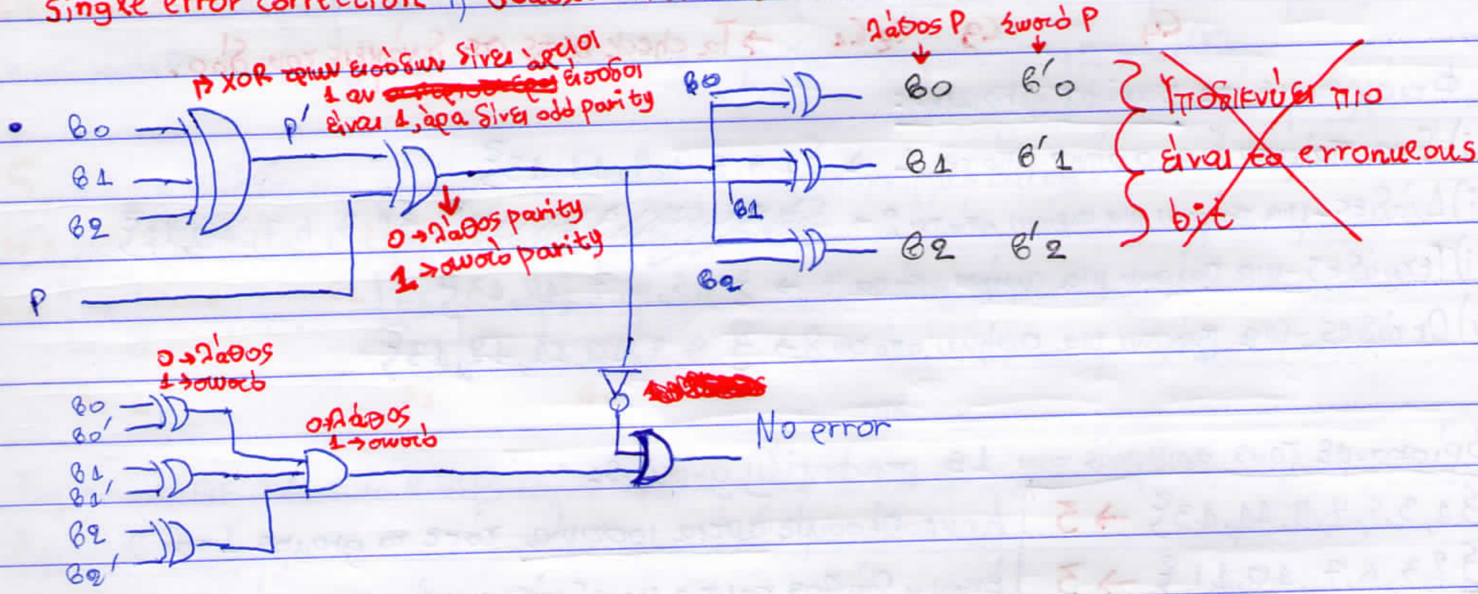
- Σε ποια κατηγορία κωδίκων ανήκει ?
- Είναι διαχωρίσιμος ?
- Ποια είναι η απόστασή του ?
- Τι είδους λάθη μπορεί να ανιχνεύσει ή να διορθώσει ?
- Πως θα κάνατε τη κωδικοποίηση και την αποκωδικοποίηση για τον κώδικα αυτό και πως την ανίχνευση λαθών ?

ΑΣΚΗΣΕΙΣ ΦΥΛΛΑΔΙΟΥ

1. $b_2 b_1 b_0 b'_2 b'_1 b'_0$ P

ελάχιστη απόσταση

• Αν αλλάξει ένα bit b_i, θα αλλάξει και το b'_i και το parity bit οπότε τότε η απόσταση κώδικα θα είναι 3. (Md=3). Αυτό σημαίνει ότι υπάρχει δυνατότητα single error correction ή double error detection.



• Εδώ: 3 bits data, 4 control bits (3 compl + Parity bit)

Τύπος: $2^c \geq n+c+1$

$2^c \geq 3+c+1 \Leftrightarrow 2^c \geq c+4$ (για $c=3$) $\to 2^3 \geq 7 \to 8 > 7$ ✓

Άρα $c=3$, έστω C_1, C_2, C_3

Η σειρά των bits θα είναι

• 6 5 4 3 2 1 → 2^k

$d_2\ d_1\ C_3\ d_0\ C_2\ C_1$

Για την πληροφορία $b_2 b_1 b_0 = 110$ η κωδικοποίηση είναι

$1 1 C_3 0 C_2 C_1 \to 110011$

2.

9	8	7	6	5	4	3	2	1	Parity
0	0	1	1	1	1	0	1	1	$\boxed{0}$

Άρτια ισοτιμία → Parity bit σωστό

Ελέγχουμε τα groups ισοτιμίας: (1, 3, 5, 7, 9) → περιττή ισοτιμία → λάθος

Άρα έχουμε σωστό parity και μη-μηδενικό syndrome

↳ δε γίνεται διόρθωση και η λέξη απορρίπτεται (ανίχνευση διπλού λάθους)

13 12 11 10 9 8 7 6 5 4 3 2 1

3. 1 0 0 0 0 1 0 1 0 0 1 1 1

Τα ελάχιστα δυαδικά ψηφία ελέγχου δίνονται από τον τύπο $2^c \geq m+c+1$

Εδώ είναι $m+c=13 \Rightarrow 2^c \geq 14 \Rightarrow \boxed{c=4}$

Άρα η λέξη είναι:

2^3 2^2 2^1 2^0

1 0 0 0 0 1 0 1 0 0 1 1 1

C_4 C_3 C_2 C_1 → Τα check bits σε δυνάμεις του δύο.

Φτιάχνουμε τα parity groups:

i) Ένα παίρνω ένα αφήνω από το 1 → {1,3,5,7,9,11,13}

ii) Δυάδες - μια παίρνω μια αφήνω από το 2 → {2,3,6,7,10,11}

iii) Τετράδες - μια παίρνω μια αφήνω από το 4 → {4,5,6,7,12,13}

iv) Οκτάδες - μια παίρνω μια αφήνω από το 8 → {8,9,10,11,12,13}

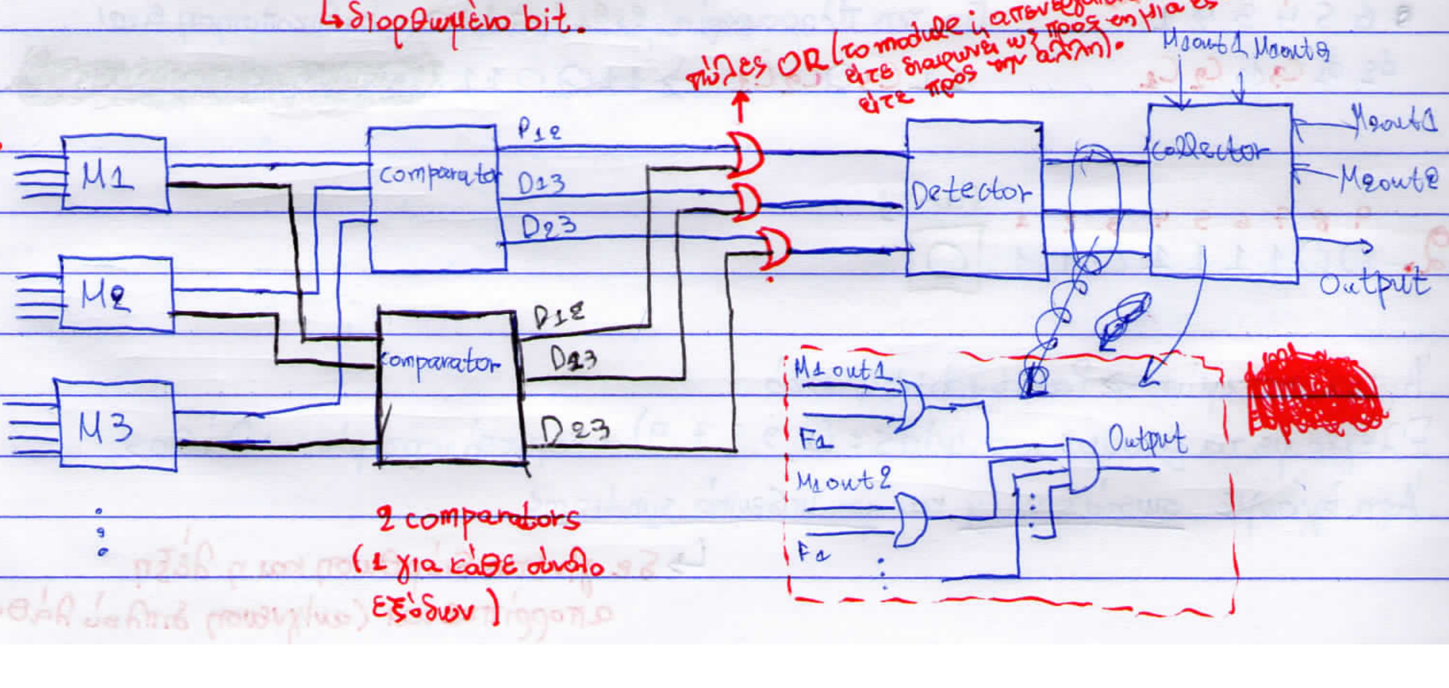
Βρίσκουμε τους αριθμούς των 1s στα parity groups:

{1,3,5,7,9,11,13} → 3
{2,3,6,7,10,11} → 3
{4,5,6,7,12,13} → 2
{8,9,10,11,12,13} → 2

Αν υποθέσουμε άρτια ισοτιμία τότε τα groups 1 και 2 έχουν λάθος και το μοναδικό στοιχείο που υπάρχει στα 1 και 2, αλλά όχι στα τρία και τέσσερα είναι το **3**. Συνεπώς η αρχική πληροφορία (αφού αφαιρέσουμε και τα check bits) είναι η 1 0 0 0 0 0 0 1 0 0 → διορθωμένο bit

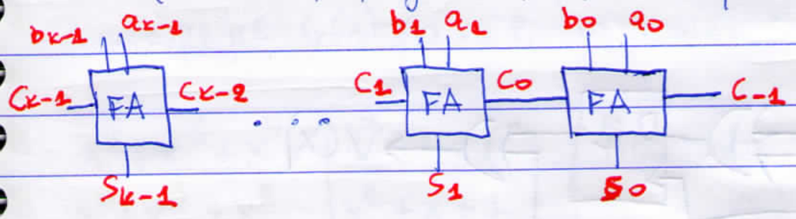
Για περιττή ισοτιμία με το ίδιο σκεπτικό, το faulty bit είναι το 12 και η αρχική πληροφορία είναι η 1 1 0 0 0 0 1 0 1.
↳ διορθωμένο bit.

4.



Η δεύτερη εκδοχή λέει να απενεργοποιώ χωριστά τις εισόδους αν διαφωνούν και αν κάποια από τις Mout;1 και Mout;2 αφωνεί με τις άλλες να μην απενεργοποιείται από τον detector. Το σχήμα απαιτεί 2 detectors και 2 collectors μετά τους comparators.
Τα πλεονεκτήματα του πρώτου τρόπου είναι μειονεκτήματα του δεύτερου. Πλεονεκτήματα:
1ος τρόπος: Με 1 detector και ένα collector έχουμε μικρότερη h/w redundancy (αν και ο collector χρειάζεται επιπλέον πύλες OR) [(100% h/w redundancy)] 2ος τρόπος
2ος τρόπος: Μπορούμε να κρατήσουμε τις σωστές εξόδους για τους υπολογισμούς. Αν μια έξοδος ενός module είναι π.χ. stuck-at-1, μπορούμε να χρησιμοποιήσουμε την άλλη.

5. Χωρίς βλάβη της γενικότητας, θεωρούμε αθροιστή ripple carry:



Για τον κάθε FA των 2 bits ισχύει: $s_i = a_i \oplus b_i \oplus c_{i-1}$ (XOR)

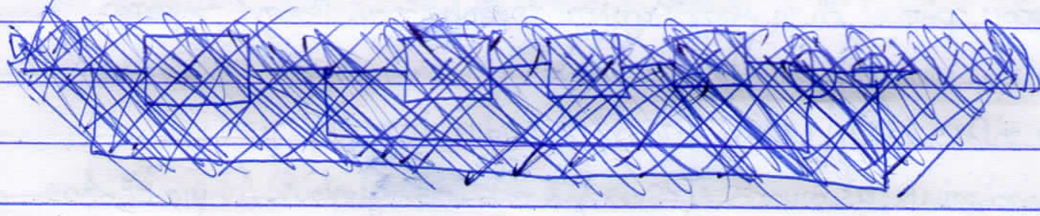
Άρα έχουμε:

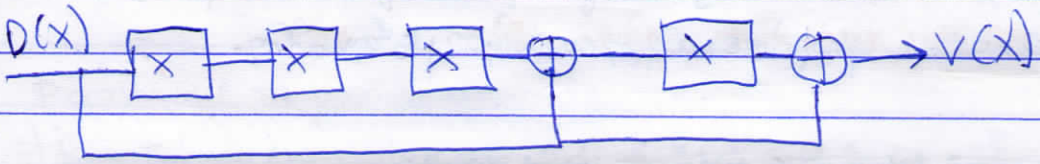
$$\left.\begin{aligned} s_0 &= a_0 \oplus b_0 \oplus c_{-1} \\ s_1 &= a_1 \oplus b_1 \oplus c_0 \\ &\vdots \\ s_{k-1} &= a_{k-1} \oplus b_{k-1} \oplus c_{k-2} \end{aligned}\right\} \xrightarrow[\oplus\, c_{k-1}]{\text{XOR κατά μέλη}} \left\{\begin{aligned} & s_0 \oplus s_1 \oplus \ldots \oplus s_{k-1} \oplus c_{k-1} = \\ & = a_0 \oplus \ldots \oplus a_{k-1} \oplus b_0 \oplus \ldots \oplus b_{k-1} \oplus \\ & \oplus c_{-1} \oplus \ldots \oplus c_{k-1} \end{aligned}\right\} \text{απόδειξη σχέσης ζητούμενου}$$

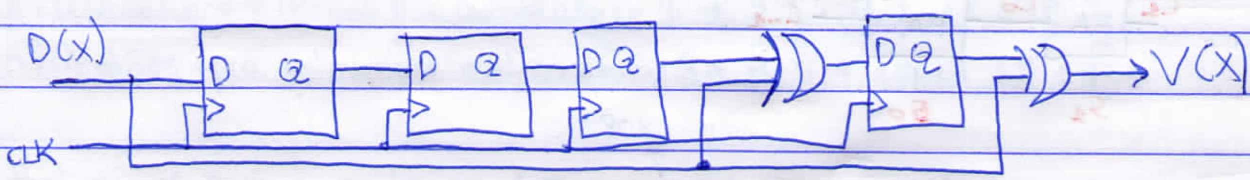
• Για να μην ισχύει η παραπάνω σχέση, πρέπει να αλλάξει το αποτέλεσμα των XOR κάποιου μέλους και αυτό συμβαίνει όταν γίνει μονός αριθμός λαθών. Επομένως αν ενώσουμε όλες τις εισόδους του αθροιστή (a_i, b_i, c_{i-1}) με κάποιο τρόπο (κύκλωμα XOR ή XOR πολλών εισόδων) ώστε να γίνεται XOR μεταξύ τους και το ίδιο κάνουμε και στις εξόδους (s_i και c_{out}) τότε προστατεύουμε τον αθροιστή από μονό αριθμό λαθών.
Ωστόσο, αν γίνει κάποιο λάθος στο κρατούμενο εισόδου με τέτοιο τρόπο που να επηρεάσει το άθροισμα στην έξοδο, τότε τα c_{i-1} και s_i είναι σε διαφορετικά μέλη οπότε μπορεί το απλό stuck-at-fault να μην ανιχνευτεί.

6. $G(X) = 1 + X + X^4$

1) Κωδικοποίηση: $V(X) = D(X)\,G(X) = D(X)\,(1+X+X^4) = D(X)\,[X(1+X^3)] + D(X)$



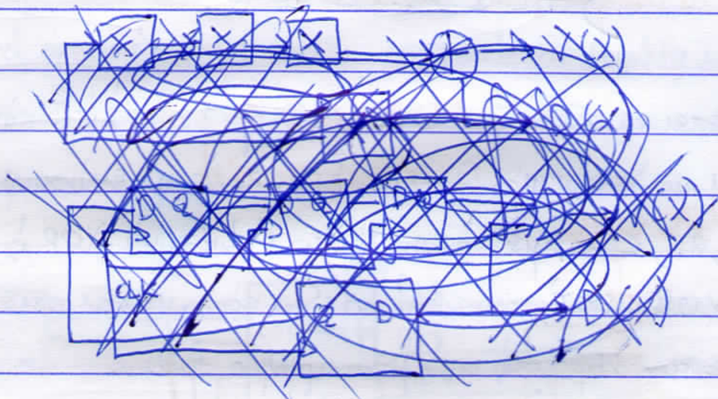


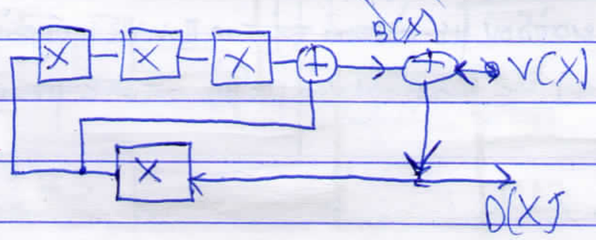


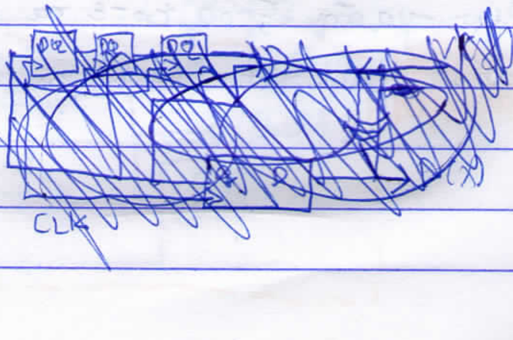
2) Αποκωδικοποίηση: $V(X) = D(X)\,G(X) = (1+X+X^4)\,D(X) = D(X) + (X+X^4)\,D(X)$

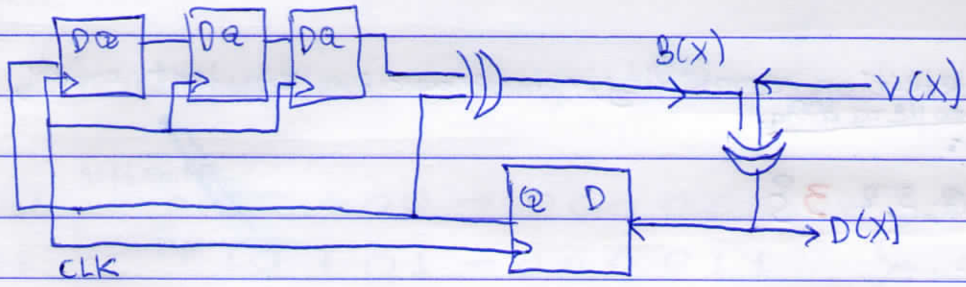
$\Leftrightarrow V(X) \underset{\text{mod-2}}{-} (X+X^4)\,D(X) = D(X) \Leftrightarrow V(X) \underset{\text{mod-2}}{+} (X+X^4)\,D(X) = D(X) \Leftrightarrow$

$\Leftrightarrow V(X) + (X^3+1)\,X\,D(X) = D(X)$









Πληροφορία: $\overset{x^3}{0}\overset{x^2}{1}\overset{x}{1}\overset{1}{1}$

Άρα $D(X) = X^2 + X + 1$

Για να είναι διαχωρίσιμη μορφή πρέπει να το πολ/σουμε με X^{n-k}, όπου n η δύναμη του V(X) και k η δύναμη του D(X). Άρα το πολ/ζουμε με τη δύναμη του G(X) → X^4

$X^4 D(X) = X^6 + X^5 + X^4$

Διαιρούμε με G(X) για να βρούμε το υπόλοιπο R(X):

mod-2

$$
\begin{array}{l|l}
\cancel{X^6} + X^5 + X^4 & X^4 + X + 1 \\
\cancel{X^6} + X^3 + X^2 & X^2 + X + 1 \\
\cancel{X^5} + X^3 + X^4 + \cancel{X^2} & \\
\cancel{X^5} + \cancel{X^2} + X & \\
\cancel{X^4} + X^3 + \cancel{X} & \\
\cancel{X^4} + \cancel{X} + 1 & \\
X^3 + 1 \rightarrow R(X) &
\end{array}
$$

Άρα είναι $V(X) = X^{n-k} D(X) + R(X)$

$\Rightarrow V(X) = X^4 D(X) + R(X)$

$\Rightarrow V(X) = X^6 + X^5 + X^4 + X^3 + 1$

Και η κωδικοποίηση της πληροφορίας 0111 σε διαχωρίσιμη μορφή είναι:

$\underbrace{0111}_{\text{αρχική πληροφορία}}|\overbrace{1001}^{\text{control bits}}$

9. Το εύρος της αναπαράστασης είναι από 0 ως product{moduli} − 1. Εδώ:
$13 \cdot 11 \cdot 7 \cdot 5 \cdot 2 = 10.010$

Άρα εύρος αναπαράστασης [0 10.009]

Αναπαράσταση και πρόσθεση: {13, 11, 7, 5, 2}

$$
\begin{array}{lcl}
(137)_{10} & = & \{7, 5, 4, 2, 1\}_{RNS} \\
+\ (26)_{10} & = & \{0, 4, 5, 1, 0\}_{RNS} \\
\hline
(163)_{10} & = & \{7, 9, 2, 3, 1\}_{RNS}
\end{array}
$$

→ Τσεκάρουμε τα υπόλοιπα του 163 με τα moduli για επαλήθευση.

mod 13 mod 11 mod 7 mod 5 mod 2

π.χ. 7+0 mod 13 = 7

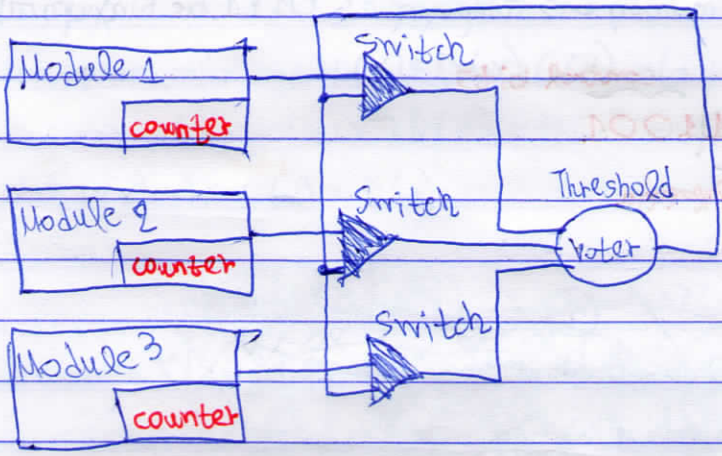
π.χ. 4+5 mod 7 = 2

Για να βελτιώσουμε error detection capability, προσθέτουμε ένα redundant moduli, το μικρότερο δυνατό και πρώτο με τα άλλα. Εδώ:

moduli set $= \{13, 11, 7, 5, 2, 3\}$

$$
\begin{aligned}
137_{10} &= \{7, 5, 4, 2, 1, 2\}_{RNS} \\
+\ 26_{10} &= \{0, 4, 5, 1, 0, 2\}_{RNS} \\
\hline
163_{10} &= \{7, 9, 2, 3, 1, 1\}_{RNS}
\end{aligned}
$$

11. Μπορούμε να συνδυάσουμε την τεχνική self purging με τεχνικές πλεονασμού στο χρόνο. Ενσωματώνουμε σε κάθε module έναν counter διαφωνιών. Αν διαφωνεί κάποιο module στον πρώτο κύκλο υπολογισμών του, τότε αυξάνουμε τον counter. Αν διαφωνεί και στον δεύτερο κύκλο, αυξάνουμε πάλι τον counter. Αν ο counter είναι μεγαλύτερος από μια τιμή που θεωρούμε πως αντιστοιχεί σε χρονικό διάστημα μόνιμου σφάλματος, τότε το module αυτο-απομακρύνεται από το σύστημα.



12. 5MR → 2 faulty modules, 2 ταυτόχρονα

S.S. 5 modules → 2 faulty modules, όχι 2 ταυτόχρονα

S.p. 5 modules → 3 faulty modules, 2 ταυτόχρονα μόνο στην αρχή.

13. Ο αριθμός των συμβόλων είναι $2^n - 1$, αφού το 0000 δε χρησιμοποιείται.

Άρα πρέπει $k = \lceil \log_2 (I + 1 - 1) \rceil \Leftrightarrow k = \lceil \log_2 I \rceil \Leftrightarrow$

$\Leftrightarrow k = \lceil \log 4 \rceil \Leftrightarrow k = 2$

Χρειάζονται 2 checkbits για την κωδικοποίηση σε Berger code.

14. Είναι 3N κώδικας γιατί:

$000 \xrightarrow{\text{leftshift}} 00 + 00 \rightarrow 000000$

$001 \xrightarrow{\text{leftshift}} 10 + 01 \rightarrow 000011$

$010 \xrightarrow{\text{leftshift}} 100 + 10 \rightarrow 000110$

$011 \xrightarrow{\text{leftshift}} 110 + 11 \rightarrow 001001$

⋮ κλπ

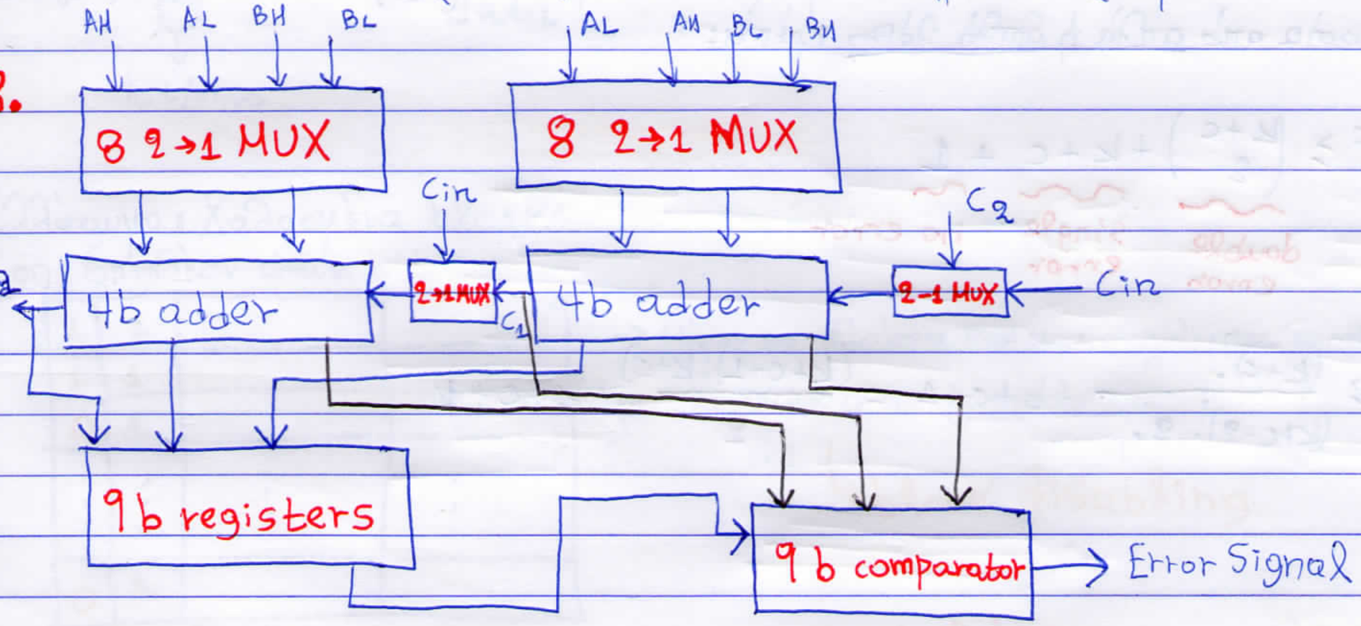
$$\left.\begin{aligned} 3N &= 2N + N \\ &= N \ll 1 + N \\ &(\text{leftshift} + \text{add}) \end{aligned}\right.$$

- Είναι 3N κώδικας άρα ανήκει στην κατηγορία των **αριθμητικών κωδίκων**
- **Δεν** είναι διαχωρίσιμος (από το 9 δε μπορώ π.χ. να πάρω το 3)
- Hd → δεν είναι 1
  → Η πρώτη με τη δεύτερη codeword έχουν απόσταση 2, άρα η απόσταση κώδικα ως η ελάχιστη απόσταση μεταξύ οποιωνδήποτε δύο codewords είναι **Hd = 2.**
- Εφόσον Hd = 2, ισχύει: **Διόρθωση κανενός λάθους.**
  Ανίχνευση απλού λάθους.
- **Κωδικοποίηση:** Πρόσθεση N με 2N. Η ποσότητα 2N προκύπτει με leftshift κατά μια θέση του binary N.

Αποκωδικοποίηση: Με τη χρήση look-up table. Για 6 ψηφία που έχουμε εδώ, χρειαζόμαστε μνήμη 2^6 θέσεων των 4 bits. (n+2 bits codeword για n bits πληροφορίας).

Ανίχνευση λαθών: Χωρίζουμε τις codewords σε δυάδες και προσθέτουμε modulo-3. Αν κάποια πρόσθεση δε δώσει 0, τότε ανιχνεύεται σφάλμα. (mod-3 adders)

8.



Επιβάρυνση σε υλικό:

18 MUX 2→1: 18%
9 registers → 18%
9 comparisons → 9%
} 45% του adder.

Λάθη που δεν ανιχνεύονται: (απλά)
- Στα buses των adders με τους registers και τον comparator
- Στους registers
- Στον comparator

Πρόταση: Εφαρμογή κάποιου seperable (= εύκολη αποκωδικοποίηση) code στα buses και στους registers (?) Duplication του comparator και σύγκριση των error signals (?) Comparator → two rail code (TSC κύκλωμα - σημειώσεις πρώτη σελίδα) (?) Software (?)

7. Όπως γνωρίζουμε για να προστατεύσουμε k bits πληροφορίας με c ψηφία ελέγχου, πρέπει οι δυνατές τιμές των check bits να είναι περισσότερες από τα απλά λάθη που μπορούν να συμβούν, τα οποία είναι c+k. Άρα πρέπει για απλά λάθη:

$$2^c \geq \underbrace{k+c}_{\text{single error}} + \underbrace{1}_{\text{no error}}$$

Με το ίδιο σκεπτικό, σε k+c bits μπορούν να συμβούν $\binom{k+c}{2}$ διπλά λάθη. Άρα για προστασία από απλά ή διπλά λάθη πρέπει:

$$2^c \geq \underbrace{\binom{k+c}{2}}_{\text{double error}} + \underbrace{k+c}_{\text{single error}} + \underbrace{1}_{\text{no error}}$$

Άρα:

$$2^c \geq \frac{(k+c)!}{(k+c-2)!\cdot 2!} + k+c+1 = \frac{(k+c-1)(k+c)}{2} + k+c+1$$

Αν έχουμε k=7 bits πληροφορίας τότε από τον παραπάνω τύπο:

$$2^c \geq \frac{(6+c)(7+c)}{2} + 8 + c \Leftrightarrow 2^{c+1} \geq (6+c)(7+c) + 16 + 2c$$

$$\Leftrightarrow 2^{c+1} \geq 42 + 13c + c^2 + 16 + 2c \Leftrightarrow 2^{c+1} \geq c^2 + 15c + 58$$

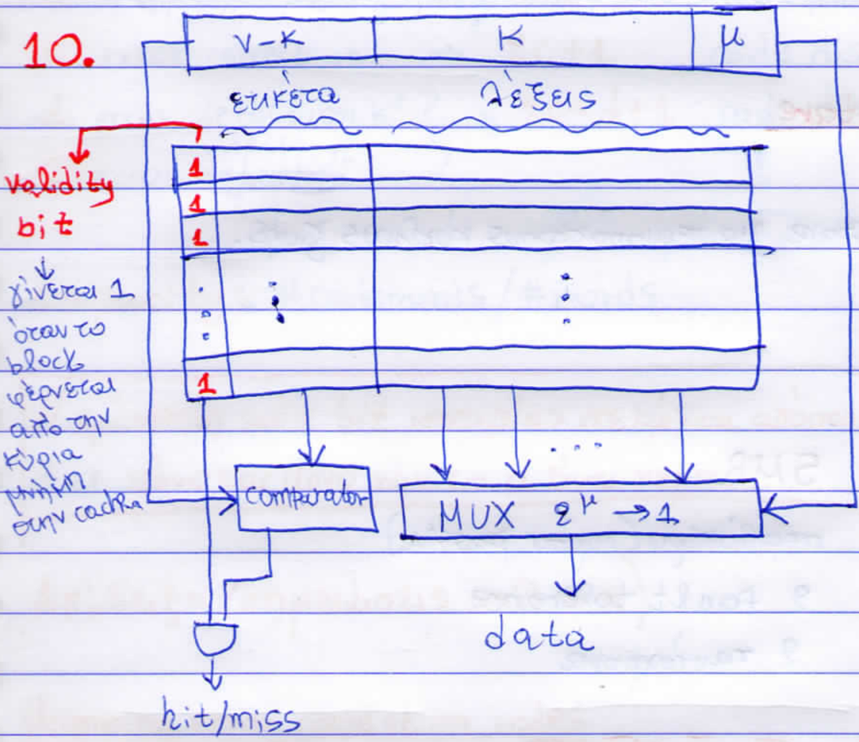
⇔ (Με δοκιμές)

$c = 6 \rightarrow 2^7 \geq 36 + 15 \cdot 6 + 58 \rightarrow$ όχι

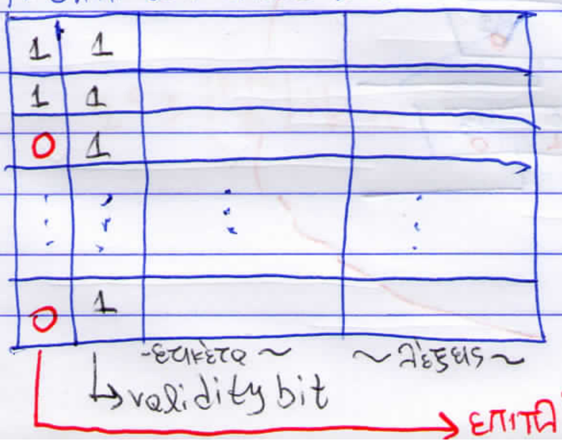
$c = 7 \rightarrow$ ναι

Άρα για k=7 bits πληροφορίας χρειαζόμαστε c=7 bits ελέγχου, άρα η επιβάρυνση σε αυτή την περίπτωση είναι **100%**.

10.



Ελάττωμα: Χαλασμένα blocks
Λύση: Επιπλέον check bit



→ Μόνο τα πλαίσια που αντιστοιχούν σε προβληματικά bits απενεργοποιούνται.

→ block disabling

1 : ok πλαίσιο

επιπλέον bit → 0 : faulty πλαίσιο