Otkrivanje i korekcija grešaka

U opštem slučaju postoje dva pristupa:

1. *Kontrola greške unapred*, u kome svaki karakter ili blok podataka koji se šalje sadrži dodatne (redundantne) informacije. Na osnovu ovih informacija primalac može ne samo da otkrije postojanje greške, već i da odredi koji su bitovi u primljenom podatku pogrešni i da grešku ukloni negacijom pogrešnih bitova.

2. *Kontrola greške unatrag*, odnosno *kontrola sa povratnom spregom*. U ovom slučaju svaki karakter ili blok podataka koji se šalje sadrži samo neophodne dodatne informacije koje omogućuju primaocu da otkrije prisustvo greške, ali ne i mesto na kome se ona nalazi. U tom slučaju jedini način za dobijanje korektnih podataka je ponavljanje prenosa svih karaktera ili blokova podataka koji imaju grešku.

U praksi, broj dodatnih bitova brzo raste sa porastom dužine podataka koji se prenose. Posledica toga je:

1. Metod sa kontrolom greške unatrag je najčešé koristi u telekomunikaconom prenosu podataka.

2. Metod sa kontrolom greške unapred se češće koristi pri čuvanju podataka na memorijskim medijumima u računarskom sistemu.

Metode za otkrivanje grešaka

Faktori od kojih zavisi izbor metode:

1. Broja bitova sa greškom (eng. *bit error rate, BER*) na komunikacionoj liniji. BER predstavlja verovatnoću u definisanom vremenskom intervalu da jedan bit ima grešku. Tako BER$=10^{-6}$ znači da, u proseku, 1 od 10^6 bita ima grešku u definisanom vremenskom intervalu.

   Za najveći broj mreža BER je oko 10^{-12}, dok je unutar računarskog sistema obično vrednost za BER 10^{-18} ili manje.

2. Tipa greške, tj. da li se greška javila na pojedinačnom bitu ili na grupi uzastopnih bitova. U poslednjem slučaju reč je o proširenoj (eksplozivnoj) grešci (eng. *burst error*).

Kontrola parnosti

Kontrola parnosti ili horizontalna provera redundanci, LRC (eng. *Longitudinal Redundancy Checking*)

Uz datu n-bitnu reč dodaje se još jedan bit tako da ukupan broj binanih jedinica u rezultujućoj reči dužine $n+1$ bude paran (neparan).

Kontrola parnosti u dve dimenzije

Vertikalna provera redundanci, VRC (eng. *Vertical Redundancy Checking*)

Npr. tabela kontrole ispravnosti prenosa ovom metodom pri prenosu bloka karaktera 'BEOGRAD' zapisanog u ASCII kodu je

	Bitovi parnosti	ASCII kod	Karakter
	0	1000010	B
	1	1000101	E
	1	1001111	O
	0	1000111	G
	1	1010010	R
	0	1000001	A
	0	1000100	D
reč parnosti	1	1011000	

Ovom metodom se otkriva svaki neparan broj grešaka, sve 2-bitne greške i najveći deo 4-bitnih grešaka.

Kontrola zbira bloka

Ako je poruka duža od jednog bajta, tada se za otkrivanje eventualnih grešaka može primeniti metod sa kontrolom zbira bloka (eng. *checksum*).

Algoritam:

- formira se zbir svih jedinica u bloku i prenese zajedno sa porukom
- obično se vrši skraćivanje dobijenog zbira, npr. na dužinu od 32 bita
- primalac ponovo izračunava sumu zbira i poredi sa primljenom vrednošću.

Nedostaci: ne može da otkriva invertovane podatke, zamenu blokova bajtova, niti dodavanje i uklanjanje bajtova sa svim nulama.

Ciklička provera redundanci

Metoda cikličke provere redundanci (eng. *Cyclic Redundancy Checking, CRC*) se koristi se za otkrivanje *proširenih grešaka*. CRC može da otkrije bilo koji neparan broj grešaka, sve 2-bitne greške, kao i proširene greške čija je dužina manja od broja redundantnih bitova.

CRC može da se implementira hardverski. Zasnovana je na:

1. Aritmetici u polju celih brojeva po modulu 2. U toj aritmetici nema prenosa pri operaciji sabiranja niti pozajmice sa prethodnog mesta pri operaciji oduzimanja. Tako za sabiranje važi $0+0=1+1=0$, $0+1=1+0=1$, $1-1=0-0=0$, $0-1=1-0=1$; slična pravila važe i za množenje i deljenje.

2. Deljenju polinoma, pri čemu se i ovo deljenje izvodi u aritmetici po modulu 2.

U CRC metodi niz bitova se posmatra kao niz koeficijenata polinoma. Tako niz bitova $a_n a_{n-1} \ldots a_1 a_0$ odgovara polinomu $M(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$. Na primer,

$$\begin{array}{rcl} 11100110 & \longrightarrow & M(x) = x^7 + x^6 + x^5 + x^2 + x^1 \\ 11001 & \longrightarrow & M(x) = x^4 + x^3 + 1 \end{array}$$

CRC algoritam se može predstaviti na sledeći način:

- Kodiranje

  1. Odabrati polinom generator $G(x)$.
  2. Izračunati $x^k M(x)/G(x)$ gde je k stepen polinoma $G(x)$. Dobijeni ostatak se označava sa $R(x)$.
  3. Dodati koeficijente ostatka na kraj poruke i poruku proslediti primaocu.

- Dekodiranje

  1. Podeliti primljenu polinomijalnu kodnu reč sa $G(x)$.
  2. Ako je ostatak deljenja nula, nema grešaka pri prenosu.
  3. Ako ostatak deljenja nije nula, greške pri prenosu postoje.

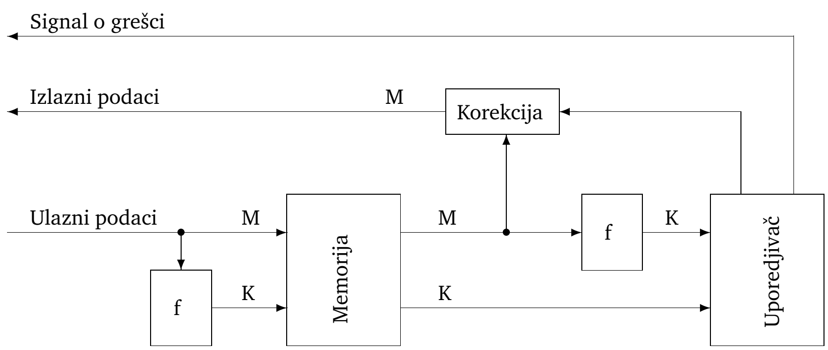
$$\begin{array}{lcl} \text{CRC-16} & = & x^{16} + x^{15} + x^2 + 1 \\ \text{CRC-CCITT} & = & x^{16} + x^{12} + x^5 + 1 \\ \text{CRC-CCITT} & = & x^{32} + x^{26} + x^{23} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + 1 \end{array}$$

Metode za otkrivanje i korekciju grešaka

Greške koje se javljaju pri zapisu podataka u poluprovodničkoj memoriji se mogu podeliti u dve kategorije:

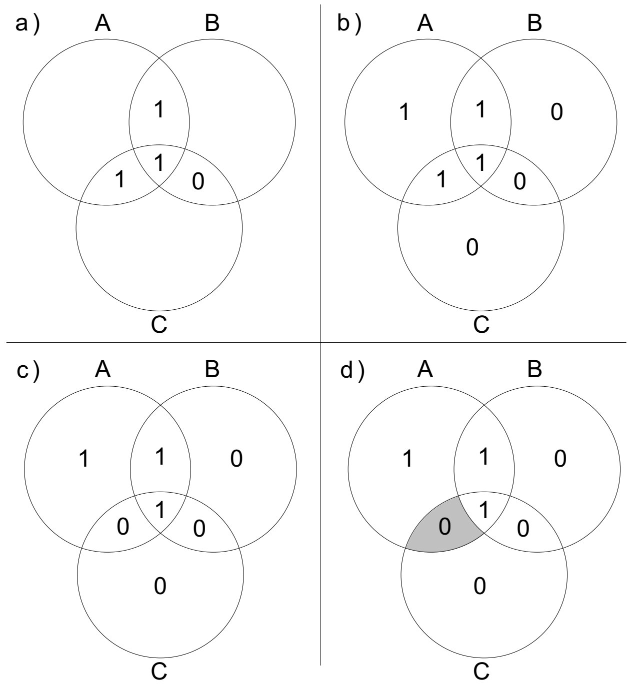
1. Takozvani tvrdi, odnosno stalno prisutni defekti. Manifestuju se tako što memorijska ćelija ne može pouzdano da čuva podatke, već bez ikakvog spoljašnjeg uzroka menja sadržaj sa 0 na 1 i obratno.

2. Takozvane mekane ili prolazne greške koje predstavljaju slučajne, nedestruktivne dogadjaje koji menjaju sadržaj jedne ili više memorijskih ćelija bez oštećenja same memorije.

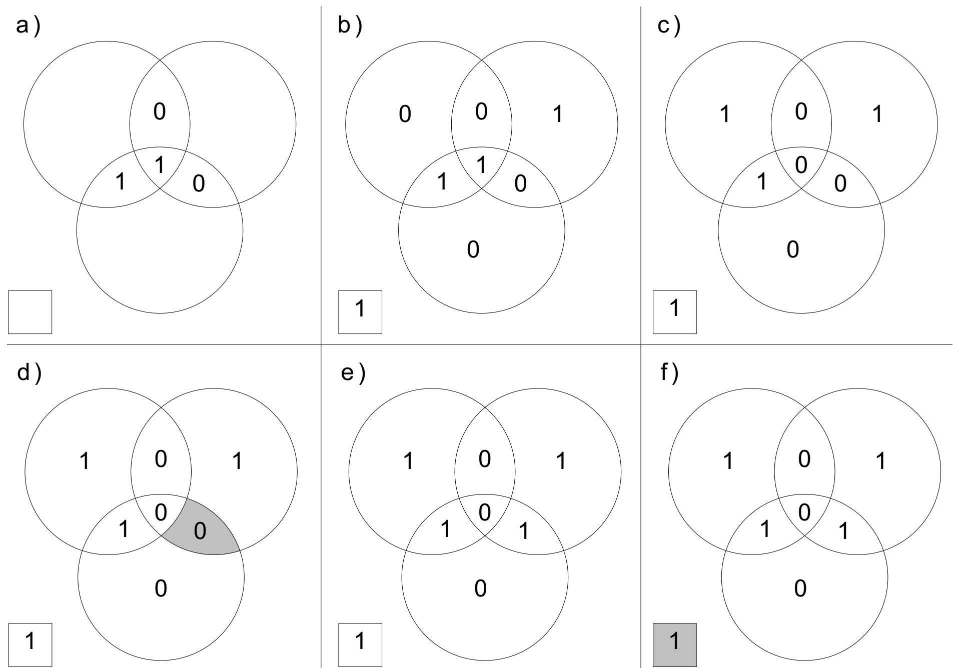


Slika 1: Proces čuvanja i provere korektnosti zapisa podataka

Hamingov kod



Slika 2: Hamingov kod za korekciju greške



Slika 3: Hamingov SEC-DED kod

Sindrom reč

Sindrom reč se dobija poredjenjem dve K-bitne vrednosti bit po bit uzimanjem eksluzivne disjunkcije. Sindorm reč je neoznačen ceo broj dužine K koji ima vrednost izmedju 0 i 2^K-1.

Vrednost 0 označava da nema grešaka u zapisu, dok ostalih 2^K-1 vrednosti odredjuju mesto greške, u slučaju da greška postoji. Kako greška može da se javi na bilo kom od M bitova podatka i K bitova koji se koriste u proveri, mora da važi $2^K - 1 \geq M + K$. Na osnovu ove nejednakosti možemo da odredimo potreban broj bitova kao funkciju dužine reči za koju se vrši provera.

	SEC		SEC-DED	
Dužina podatka	Bitova za proveru	% povećanja	Bitova za proveru	% povećanja
8	4	50.00	5	62.50
16	5	31.25	6	37.50
32	6	18.75	7	21.88
64	7	10.94	8	12.50
128	8	6.25	9	7.03
256	9	3.52	10	3.91

Tabela 1: Povećanje dužine reči u slučaju implementacije korekcije grešaka

Želja je da generisana sindrom reč ima sledeće karakteristike:

- Ako su svi bitovi u njoj 0 tada nije otkrivena nikava greška.
- Ako u reči postoji tačno jedna 1, tada greška postoji u jednom od 4 bita za proveru, dok je zapis podatka u redu i ne zahteva nikavu korekciju.
- Ako sindrom sadrži više od jednog bita koji ima vrednost 1, tada vrednost sindrom reči posmatrane kao ceo neoznačen broj odredjuje poziciju na kojoj se nalazi greška. Korekcija se sastoji u komplementiranju vrednosti odgovarajućeg bita podatka.

Da bi ove karakteristike i ostvarili uredimo bitove podatka i bitove za proveru u reč dužine 12 bita na način prikazan u tabeli 2:

Pozicija bita	Broj pozicije	Bit za proveru	Bit podatka
12	1 1 0 0		M8
11	1 0 1 1		M7
10	1 0 1 0		M6
9	1 0 0 1		M5
8	1 0 0 0	C4	
7	0 1 1 1		M4
6	0 1 1 0		M3
5	0 1 0 1		M2
4	0 1 0 0	C3	
3	0 0 1 1		M1
2	0 0 1 0	C2	
1	0 0 0 1	C1	

Tabela 2: Pozicije bitova podataka i bitova za proveru

$$
\begin{aligned}
C_1 &= M_1 \oplus M_2 \oplus M_4 \oplus M_5 \oplus M_7 \\
C_2 &= M_1 \oplus M_3 \oplus M_4 \oplus M_6 \oplus M_7 \\
C_3 &= M_2 \oplus M_3 \oplus M_4 \oplus M_8 \\
C_4 &= M_5 \oplus M_6 \oplus M_7 \oplus M_8
\end{aligned}
$$

gde $\oplus$ označava operaciju ekskluzivne disjunkcije.

Kao primer otkrivanja grešaka pomoću ovog koda pretpostavićemo da je M = 10110101 reč koju treba sačuvati. Za ovu reč izračunate vrednosti bitova provere su

$$
\begin{array}{lll}
C_1 = & 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 & = 0 \\
C_2 = & 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 & = 1 \\
C_3 = & 0 \oplus 1 \oplus 0 \oplus 1 & = 0 \\
C_4 = & 1 \oplus 1 \oplus 0 \oplus 1 & = 1
\end{array}
$$

Pretpostavim dalje da je posle čuvanja došlo do promene bita na mestu broj 5 koji je sa 1 promenjen na 0. Vrednosti bitova za proveru koji se odredjuju pri čitanju tako dobijenog podatka su

$$
\begin{array}{lll}
C'_1 = & 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 & = 1 \\
C'_2 = & 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 & = 1 \\
C'_3 = & 0 \oplus 1 \oplus 0 \oplus 1 & = 0 \\
C'_4 = & 0 \oplus 1 \oplus 0 \oplus 1 & = 0
\end{array}
$$

Uporedjujući ove dve grupe bitova za proveru dobija se

$$
\begin{array}{cccccc}
 & C_4 & C_3 & C_2 & C_1 & = \quad 1\,0\,1\,0 \\
 & C'_4 & C'_3 & C'_2 & C'_1 & = \quad 0\,0\,1\,1 \\
\hline
\oplus & & & & & = \quad 1\,0\,0\,1
\end{array}
$$

Na osnovu rezultata reči vidi se da u zapisu reči M postoji greška. Kako je dobijena vrednost 1001, to je greška u bitu M_5 koji se nalazi na poziciji kojoj odgovara ta binarna vrednost u tabeli 2.