(19) Országkód:

HU

MAGYAR KÖZTÁRSASÁG ORSZÁGOS TALÁLMÁNYI HIVATAL

SZABADALMI LEÍRÁS

(11) Lajstromszám:

199046 B

(51) Int. Cl.[4]

H 03 M 13/00

(22) Bejelentés napja: 1984.12.19. (21) 169/85 (2, 5, 6 ig.p.)
(86) Nemzetközi bejelentés száma: PCT(JP 84)00603
(87) Nemzetközi közzététel száma: WO 85/02958
(30) Bejelentés elsőbbsége:
240525/1983 1983.12.20. JP (1, 3, 4 ig.p.)
198079/1983 1983.12.23. JP (7. ig.p.)
(40) Közzététel napja: 1985.12.30.

(45) Megadás meghirdetésének dátuma a Szabadalmi Közlönyben: 1989.12.28.

(72) Feltalálók:
OZAKI Shinya,
ODAKA Kentaro,
FUKAMI Tadashi, Tokyo, (JP)

(73) Szabadalmas:
Sony Corporation,
Tokyo, (JP)

(54) ELJÁRÁS ÉS BERENDEZÉS HIBAJAVÍTÓ KÓDOLÁSÚ KÓDJELEK DEKÓDOLÁSÁRA

(57) KIVONAT

A találmány egyrészt eljárás hibajavító kódolású kódjelek dekódolására, amelynél egy $k_1 \times k_2$ kétdimenziós elrendezés oszlopaiban lévő minden egyes k_1 információs szimbólum jel számára n_1 kód hosszúságú első hibajavító kódolású kódjelek, soraiban lévő minden egyes k_2 információs szimbólum jel számára pedig n_2 kód hosszúságú második hibajavító kódolású kódjelek vannak, amely eljárás során vesszük legalább az első hibajavító kódolású kódjeleket és ezeket dekódoljuk; az első hibajavító kódolású kódjelek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képezünk és ezeket egy legalább n_2-bites memóriában tároljuk; dekódoljuk a második hibajavító kódolású kódjeleket; a második hibajavító kódolású kódjelek hibaészlelési vagy hibajavítási státuszát megadó második pointer jeleket képezünk és ezeket egy legalább k_1-bites memóriában tároljuk; majd kiadjuk az információs szimbólum jeleket. Az eljárást az jellemzi, hogy az első és a második pointer jelek állapotának alapján az információs szimbólum jelek megbizhatóságát mutató jelölőjelet állítunk elő.

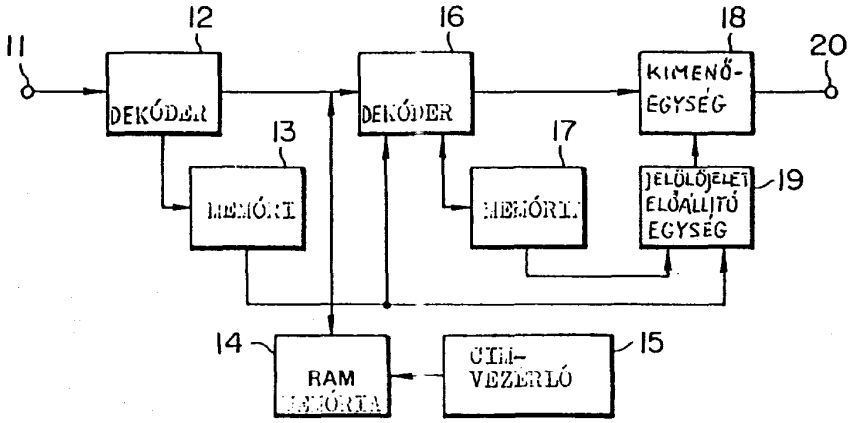
A találmány másrészt berendezés hibajavító kódolású kódjelek dekódolására, amelynek legalább az első hibajavító kódolású kódjeleket vevő szerve, ehhez csatlakoztatott, az első hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képező, valamint a második hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó második pointer jeleket képező dekódolóegysége, a dekódolóegységhez csatlakoztatott, az első pointer jeleket tároló legalább n_2-bites első memóriája, a dekódolóegységhez csatlakoztatott, a második pointer jeleket tároló legalább k_1-bites második memóriája és a dekódolóegységhez csatlakoztatott, az információs szimbólum jeleket előállító kimenőegysége van.

A berendezést az jellemzi, hogy az első és a második memóriához (13, 17) az információs szimbólum jelek megbizhatóságát mutató jelölőjelet előállító egység (19) van csatlakoztatva.

HU 199046 B

A leírás terjedelme: 12 oldal, 3 rajz, 7 ábra

FIG.3


11
12
DEKÓDER
13
MEMÓRI
16
DEKÓDER
17
MEMÓRI
18
KIMENŐ-
EGYSÉG
20
JELÖLŐJELET
ELŐÁLLÍTÓ
EGYSÉG
19
14
RAM
MEMÓRIA
CÍM-
VEZÉRLŐ
15

MŰSZAKI TERÜLET

A találmány hibajavító kódolású kódjelek dekódolására szolgáló eljárásra és berendezésre vonatkozik.

A TECHNIKA ÁLLÁSA

Ismeretesek olyan szorzat kódok, amelyeknél az információs szimbólum jelek kétdimenziós formában vannak elrendezve és ezen kétdimenziós elrendezés minden egyes sorára és oszlopára vonatkozóan hibajavító kódot képeznek oly módón, hogy minden egyes információs szimbólum jel két hibajavító kódjel sorozatban van benne. A szorzat kód dekódolása a dekódolási információ, azaz pointer jel alapján úgy történik, hogy a hibajavító kódolású kódjeleket dekódolják az egyes oszlopokra és sorokra vonatkozóan.

A hagyományos eljárásokban, mivel minden egyes információs szimbólum jel kapcsolatban áll egy pointer jellel, az szükséges, hogy a pointer jelek teljes száma legalább annyi legyen, mint az információs szimbólum jelek száma.

Továbbá abban az esetben, amikor a pointer jelek alkalmazásával hibacsomó javítást végeznek, fellép az a probléma, hogy a feldolgozási lépések száma, így pl. a memóriához fordulások, számítási műveletek stb. szükségképpen megnőnek, mivel a pointer jeleket egy pointer memóriából olvassák ki és a hibaértékeket mindegyik sorra kiszámítják.

Másrészről az esetben, amikor hibajavító kódként olyan bonyolult kódokat alkalmaznak, mint a BCH kódok, a hibaérték megállapítására szolgáló műveletek igen bonyolulttá válnak és így fellép az a probléma, hogy nagyszámú program lépésre van szükség, amennyiben a számításokat hardver hajtja végre.

A TALÁLMÁNY KINYILVÁNÍTÁSA

Jelen találmánynak az a célja, hogy olyan eljárást és berendezést szolgáltasson hibajavító kódolású kódjelek dekódolására, amely csökkenti a dekódoláshoz szükséges pointer jelek számát és ezzel együtt a pointer jelek tárolásához szükséges memóriaterület méretét, valamint a pointer jelek kiolvasási és beírási darabszámát.

A találmány egy másik célja, hogy olyan eljárást és berendezést szolgáltasson hibajavító kódolású kódjelek dekódolására, amely lehetővé teszi a feldolgozási lépések számának jelentős csökkentését azon tényből kifolyólag, hogy a pointer jelek minden egyes sorra vonatkozóan ugyanazok.

Egy további célja a találmánynak az, hogy olyan hibajavító kódolású kódjelek dekódolására szolgáló berendezést szolgáltasson, amely lehetővé teszi a számítási lépések számának csökkentését a hibacsomó javításban.

Az is célja a találmánynak, hogy hibajavító kódolású kódjelek dekódolására szolgáló olyan eljárást nyújtson, amely egyszerű felépítésű és kisszámú feldolgozási lépéssel határozza meg a hibaértékeket a dekódolás során.

A találmány tehát egyrészt eljárás hibajavító kódolású kódjelek dekódolására, amelynél egy $k_1 \times k_2$ kétdimenziós elrendezés oszlopaiban lévő minden egyes k_1 információs szimbólum jel számára n_1 kód hosszúságú első hibajavító kódolású kódjelek, soraiban lévő minden egyes k_2 információs szimbólum jel számára n_2 kód hosszúságú második hibajavító kódolású kódjelek vannak, amely eljárás során vesszük legalább az első hibajavító kódolású kódjeleket és ezeket dekódoljuk; az első hibajavító kódolású kódjelek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képezünk és ezeket egy legalább n_2-bites memóriában tároljuk; dekódoljuk a második hibajavító kódolású kódjeleket; a második hibajavító kódolású kódjelek hibaészlelési vagy hibajavítási státuszát megadó második pointer jeleket képezünk és ezeket egy legalább k_1-bites memóriában tároljuk; majd kiadjuk az információs szimbólum jeleket. Az eljárást a találmány szerint az jellemzi, hogy az első és a második pointer jelek állapotának alapján az információs szimbólum jelek megbízhatóságát mutató jelölőjelet állítunk elő.

Ugyancsak tárgya a találmánynak a fenti eljárás egy olyan változata, amelynél a második hibajavító kódolású kódjeleket az első pointer jelek alkalmazásával oly módon dekódoljuk, hogy a második hibajavító kódolású kódjelek minden egyes sorozatára végrehajtunk egy hibacsomó javítást, majd kiadjuk az információs szimbólum jeleket. Ezt az eljárást a találmány szerint az jellemzi, hogy a második hibajavító kódolású kódjelek dekódolása során a hibacsomó javításban a hibaérték kiszámítási műveletek egy részét csak egyszer végezzük el a második hibajavító kódolású kódjelek minden egyes sorozatára.

A találmány másrészt berendezés hibajavító kódolású kódjelek dekódolására, amelyben egy $k_1 \times k_2$ kétdimenziós elrendezés oszlopaiban lévő minden egyes k_1 információs szimbólum jel számára n_1 kód hosszúságú első hibajavító kódolású kódjelek, soraiban lévő minden egyes k_2 információs szimbólum jel számára pedig n_2 kód hosszúságú második hibajavító kódolású kódjelek vannak, amely berendezésnek legalább az első hibajavító kódolású kódjeleket vevő szerve, ehhez csatlakoztatott, az első hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képező, valamint a második hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó második pointer jeleket képező dekó-

3

dolóegysége, a dekódolóegységhez csatlakoztatott, az első pointer jeleket tároló legalább n_2-bites első memóriája, a dekódolóegységhez csatlakoztatott, a második pointer jeleket tároló legalább k_1-bites második memóriája és a dekódolóegységhez csatlakoztatott, az információs szimbólum jeleket előállító kimenőegysége van. A berendezést a találmány szerint az jellemzi, hogy az első és a második memóriához az információs szimbólum jelek megbízhatóságát mutató jelölőjelet előállító egység van csatlakoztatva.

Ugyancsak tárgya a találmánynak egy olyan berendezés, amelynek legalább az első hibajavító kódolású kódolójeleket vevő szerve, ehhez csatlakoztatott, az első hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képező első dekódere, az első dekóderhez csatlakoztatott, az első pointer jeleket tároló legalább n_2-bites első memóriája, az első dekóderhez és az első memóriához csatlakoztatott, a második hibajavító kódolású kódjeleket az első pointer jelek alkalmazásával a második hibajavító kódolású kódjelek minden egyes sorozatára egy hibacsomó javítást végrehajtva dekódoló második dekódere és az információs szimbólum jeleket előállító kimenőegysége van. Ezt a berendezést a találmány szerint az jellemzi, hogy a hibacsomó javításban a második hibajavító kódolású kódjelek minden egyes sorozatára a hibaérték kiszámítási műveletek egy részét csak egyszer elvégző második dekódere van.

A találmány továbbá olyan berendezésre is vonatkozik, amelynek legalább az első hibajavító kódolású kódjeleket vevő és az információs szimbólum jeleket előállító vevő és kimenő szerve, ehhez csatlakoztatott, az első hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képező, valamint a második hibajavító kódolású kódjeleket az első pointer jelek alkamazásával a második hibajavító kódolású kódjelek minden egyes sorozatára egy hibacsomó javítást végrehajtva dekódoló dekódolóegysége és a dekódolóegységhez csatlakoztatott, az első pointer jeleket tároló legalább n_2-bites memóriája van, és az jellemzi, hogy a hibacsomó javításban a második hibajavító kódolású kódjeleket minden egyes sorozatára a hibaérték kiszámítási műveletek egy részét csak egyszer elvégző dekódolóegysége van.

A találmány tárgya továbbá egy berendezés hibajavító kódolású kódjelek dekódolására, amely kódolás nem-dualisztikus és ahol információs szimbólum jelek modulo kettő szerinti összeadása több szindróma jel egyikét eredményezi, továbbá amellyel az információs szimbólum jelek sorozatában több információs szimbólum jel hibája javítható, amely berendezésnek dekódolóegysége, bemenőegysége, szindróma jeleket előállító szindróma generátora, ehhez csatlakozatatott, a szindróma jelekből hibahelyet és hibaértéket kiszámító egysége, a bemenőegységhez, valamint a hibahely és hibaérték kiszámító egységhez csatlakoztatott, a hibaérték összegezésével egy hibás információs szimbólum jelet javító hibajavító egysége és a hibajavító egységhez csatlakoztatott kimenőegysége van. Ezt a berendezést a találmány szerint az jellemzi, hogy a hibahely és hibaérték kiszámítható egységnek a szindróma jeleket tároló szindróma regisztere és a szindróma regiszterhez csatlakoztatott, a hibaértéknek a szindróma jelekhez való hozzáadásával a szindróma regiszter számára hibaértéket előállító kizáró-VAGY kapuja van.

A találmányt a továbbiakban a rajzokon szemléltetett előnyös kiviteli alakok alapján ismertetjük.

AZ ÁBRÁK RÖVID LEÍRÁSA

Az 1. ábra a találmány egy kiviteli alakja szerinti kódoló tömbvázlata.

A 2. ábra a találmány egy kiviteli alakja működésének magyarázatát segítő vázlatrajz.

A 3. ábra a találmány egy kiviteli alakja szerinti dekóder tömbvázlatát mutatja.

A 4. ábra a találmány egy kiviteli alakjának tömbvázlata.

Az 5. ábra a találmány egy kiviteli alakjának egy részét ábrázoló tömbvázlat.

A 6. ábra a találmány egyik további kiviteli alakjának tömbvázlata.

A 7. ábra a 6. ábra szerinti egyik feldolgozó áramkör felépítését mutató tömbvázlat.

A TALÁLMÁNY ELŐNYÖS KIVITELI ALAKJAI

Az 1. ábra szorzat-kódos kódjeleket előállító kódoló felépítését mutatja. Az 1 bemeneti kapcson érkező bejövő adatjel 2 C_2 paritás generátorra és 3 szelektor egyik bemenetére jut, a 2 C_2 paritás generátor (a C_2 jelölés egy második hibajavító kódra utal) által képzett C_2 paritás adat a 3 szelektor második bemenetére jut. A 3 szelektor k_1-szer ismételi meg a műveleteket (n_2 - k_2) számú paritás adat kiválasztására, miután a k_2 információs szimbólum jeleket kiválasztotta. Ezen művelet alatt az információs szimbólum jelek és a paritás adatok egy véletlen hozzáférésű 4 RAM memóriában vannak tárolva 5 címvezérlő által meghatározott sorrendben.

A 4 RAM memóriából kiolvasott adatok 6 C_1 paritás generátorra (a C_1 jelölés egy első hibajavító kódra utal) és 7 szelektor egyik bemeneti pontjára jutnak, a 6 C_1 paritás generátor által képzett C_1 paritás adat pedig a 7 szelektor másik bemenetére jut. A 7 szelektor (n_1 - k_1) x k_2 számú C_1 paritás adatot választ ki a C_2 paritás adatokat magában foglaló (k_1 x n_2) szimbólum jel kiválasztása után. A 7 szelektor 8 kimeneti kapcsáról le-

vett kimenő kódjel továbbítható vagy rögzíthető pl. egy (nem ábrázolt) mágnesfejjel ellátott mágnesszalagos egységre. Ez esetben lehetőség van arra, hogy a kimenő kódjelet visszaírjuk a 4 RAM memóriába, és hogy más sorrendben olvassuk ki rögzítés céljából.

A 2. ábra a fent leírt kódoló által kialakított kódjelek egy konfigurációját mutatja. Az információs szimbólum jelek egy (k_1 x k_2) méretű, kétdimenziós alakzatban vannak elrendezve. A vízszintes irányban elrendezett k_2 információs szimbólum jelek, azaz a kétdimenziós elrendezés minden sora, alá van vetve a C_2 kód szerinti kódolási eljárásának. A függőleges irányban elrendezett k_1 információs szimbólum jelek, azaz a kétdimenziós elrendezés minden oszlopa, alá van vetve a C_1 kód szerinti kódolási eljárásnak. A C_1 kód szerinti kódolásban a C_2 paritás adatok is résztvesznek. A C_1 kód pl. egy (n_1, k_1) Reed-Solomon kód, amellyel maximum (n_1 - k_1)/2 szimbólum jel hibáját lehet javítani.

A Reed-Solomon kód dekódolásának általános módszere az alábbi. Az (n, k) Reed-Solomon kód - ahol n jelöli a kód hosszát és k jelöli az információs szimbólum jelek számát - Hamming-távolságát egy GF(2^m) Galois mezőben $d = n - k + 1$ alakban lehet kifejezni, a generátor polinom $\prod_{i=0}^{d-2} (x + \alpha^i)$ alakú. Ha a vett szavakat r_0, r_1, r_2,...r_{n-1} betűkkel jelöljük, akkor a szindrómát a következő kifejezés meghatározásával kaphatjuk meg:

$$S_j = \sum_{i=0}^{n-1} r_i \alpha^{ij} \quad (j = 0, 1,...d-2) \qquad (1)$$

Egy $\sigma(z)$ hibahely meghatározó polinomot és egy $w(z)$ hibaértékelő polinomot nyerünk az S_j szindróma alkalmazásával. Ezen eljárás céljaira az Euklidesz féle kölcsönös osztási eljárást, a Varlay-Camp féle eljárást, a Peterson-módszert stb. javasolják.

A $\sigma(z) = 0$ egyenlet megoldásával az X_i hibahely meghatározható. Erre a célra a Chien féle keresési módszert alkalmazzák.

Ezt követően az X_i hibahely és az $w(z)$ hibaértékelő polinom alapján az Y_i hibaérték megkapható.

A dekódolási lépések során felmerülő fenti műveleteket az X_i hibahely ($i = 1, 2,...e$; ahol e jelöli a hibák számát) és az Y_i hibaérték felvételével világítjuk meg. Mivel a Reed-Solomon kód egy lineáris kód, a szindrómát kifejezhetjük az alábbi módon:

$$S_j = \sum_{i=1}^{e} Y_i X_i^j \quad (j = 1, 2,...d-2). \qquad (2)$$

Ha a szindrómát egy

$$S(z) = \sum_{j=0}^{d-2} S_j z^j \qquad (3)$$

alakú polinommal fejezzük ki, a következő kifejezéshez jutunk:

$$S(z) = \sum_{i=1}^{e} Y_i \sum_{j=0}^{d-2} (X_i z)^j =$$

$$\sum_{i=1}^{e} = \frac{Y_i}{1 - X_i z} \quad (\text{mod. } z^{d-1}). \qquad (4)$$

Ha a hibahely meghatározó polinomot és a hibaértékelő polinomot

$$\sigma(z) = \prod_{j=1}^{e} (1 - X_j z) \qquad (5)$$

$$w(z) = \sigma(z) \quad S(z) \qquad (6)$$

alakban definiáljuk, akkor $w(z)$ az alábbi alakra hozható:

$$w(z) = \sum_{i=1}^{e} \frac{Y_i}{1 - X_i z} \prod_{j=i}^{e} (1 - X_j z) =$$

$$= \sum_{i=1}^{e} Y_i \prod_{j=i}^{e} (1 - X_j z)$$

$$(\text{mod. } z^{d-1}). \qquad (7)$$

Az Y_i hibaértéket úgy kaphatjuk meg, hogy z helyébe X_i^{-1} értéket helyettesítünk és a kifejezést a következőképpen transzformáljuk:

$$Y_i = w(X_i^{-1}) / \prod_{i=1}^{e} (1 - X_j X_i^{-1}). \qquad (8)$$

Példaként egy α^0,...α^7 gyökökkel rendelkező (32, 24) Reed-Solomon kódot ismertetünk. Mivel ezen kód Hamming-távolsága d = = 9, lehetőséget biztosít maximum négy szimbólum javítására. Ha a négy szimbólum hibahelye X_1,...X_4 és a hibaértékek Y_1,...Y_4, a szindrómát a következő kifejezésből nyerjük:

$$S_j = \sum_{i=1}^{4} Y_i X_i^j \quad (j = 0, 1, 2, 3).$$

A négy szindróma közül az S_0 szindróma:

$$S_0 = \sum_{i=1}^{4} Y_i$$

Ha az Y_1, Y_2 és Y_3 hibaértékeket meghatároztuk, a fennmaradó Y_4 hibaérték bonyolultabb számítások nélkül meghatározható:

$$Y_4 = S_0 - Y_1 - Y_2 - Y_3.$$

A GF(2^m) kódoknál a kivonás megfelel egy modulo 2 összeadásnak.

A 3. ábra egy ilyen kiviteli alak szerinti dekóder felépítését mutatja be. A 3. ábrán a beolvasott kódjelek 11 vevő szervről, amely az ábrázolt kivitelben egy bemeneti kapocs, C_1 kódot dekódoló 12 dekóder bemenetére jutnak. A 12 dekóderben történik meg a C_1 kódolású kódjelek dekódolása, ennek során az összes, de legfeljebb (n_1 - k_1)/2 hiba javításra kerül. Abban az esetben azonban, amikor a kódjelek egyetlen sorozatában a hibák száma nagyobb vagy egyenlő az (n_1 - k_1)/2 mennyiségnél, ezen sorozat C_1 pointer jelét „1" értékre, a többi pointer jelet „0" értékre állítjuk be. A 2. és 3. ábrán látható pointer 13 memória n_2 bitet tartalmaz és a C_1 kódolású kódjelek pointer jeleit tárolja. A 12 dekóder kimenetét átmenetileg a 14 RAM

memóriában tároljuk a 15 címvezérlő által meghatározott sorrendben.

A 14 RAM memóriából kiolvasott kódjelek egy C_2 kódot dekódoló 16 dekóderre jutnak, ahol a C_2 kód szerinti dekódolásuk történik meg. A 16 dekóder megkapja a C_1 pointer jelet a 13 memóriától. Mivel a C_1 pointer jel közös a C_2 kódolású kódjelek összes k_1 sorozatára, lehetőség van arra, hogy mindegyik sorozatban ugynazon eljárással dekódoljuk a C_2 kódot. A 16 dekóder maximum $(n_2 - k_2)/2$ szimbólum jel hibáját javítja és a C_2 kódra vonatkozóan háromféle pointer jelet generál, amelyeket a pointer 17 memóriában tárol.

Amikor a 16 dekóderben hibajavítás történik, az illető sorozattal kapcsolatos C_2 pointer jeleket „0" értékre állítjuk be. Amikor a 16 dekóder nem tudja kijavitani a hibát és a C_1 pointer jeleket másoljuk át azok nagy megbizhatósága miatt, a C_2 pointer jeleket „1" értékűre állitjuk be. Amikor a 16 dekóder nem tudja kijavitani a hibát és a C_1 pointer jelek kis megbizhatósága folytán az összes szimbólum jelet hibásnak tekintjük, a C_2 pointer jeleket „2" értékre állitjuk. Ennélfogva a C_2 pointer jelek 2-bitesek és a 17 memória $(2k_1)$-bites.

A 13 és 17 memóriákat kialakithatjuk attól a 14 RAM memóriától különválasztva, amelyik az információs szimbólum jeleket és a paritás adatokat tárolja a dekódolás során, avagy azok benne lehetnek a 14 RAM memóriában, felhasználva annak egy-egy memória területét.

A C_1 pointer jelet nem szükséges 1-bitesre korlátozni, igy az lehet két- vagy több-bites. Továbbá ki lehet terjeszteni a C_2 kód szerinti hibajavító feldolgozást a C_1 paritás adatokra is, ez esetben a C_2 pointer jel 17 memóriája $(2n_1)$-bites.

A 16 dekóder kimenő jelei 18 kimenőegységre, az ábrázolt kivitelben egy interpolációs áramkörre kerülnek azon szimbólum jel hibák megszüntetése céljából, amelyek javítása nem történt meg. Az interpolációs áramkör pl. a középérték alapján végzi el az interpolációt. A 18 kimenőegységet egy jelölőjelet előállító 19 egység vezérli, amelyik megkapja a C_1 és C_2 pointer jeleket a 13 és 17 memóriákból. A 18 kimenőegység kimenő jelei 20 kimeneti kapcsolóról vehetők le. A 19 egység minden egyes információs szimbólum jel esetében eldönti a C_1 és C_2 pointer jelek alapján, hogy szükség van-e interpolációra. A 2. ábrán 13' memóriában a C_1 pointer jelek, 17' memóriában a C_2 pointer jelek kombinációit szerepeltettük.

Amikor a C_2 pointer jel „0" értékű, akkor függetlenül attól, hogy a C_1 pointer jel „0" vagy „1" értékű, a 18 kimenőegység nem működik. Amikor a C_2 pointer „1" értékű és a C_1 pointer „0" értékű, mivel megállapitást nyert, hogy az információs szimbólum jel hibátlan, nem történik interpoláció. Amikor a C_2 pointer jel „1" értékű és a C_1 pointer jel is „1" értékű, mivel az nyert megállapitást, hogy a szimbólum jel hibás, interpolációra kerül sor. Továbbá amikor a C_2 pointer jel értéke „2", akkor függetlenül attól, hogy a C_1 pointer jel értéke „0" vagy „1", mivel az nyert megállapitást, hogy hibás szimbólum jelről van szó, az interpolációs művelet végrehajtásra kerül.

A C_1 pointer jelek megbizhatóságát a 16 dekóder értékeli ki. Például feltételezve, hogy a C_2 kód legfeljebb két szimbólum jel hibát tud jevitani, ha a C_2 kód révén nem lehet végrehajtani a javitást annak ellenére, hogy csak egy C_1 pointer jel „1" értékű, az nyer megállapitást, hogy a C_1 pointer jel megbizhatósága kicsi, mivel a fenti eset abnormális. Abban az esetben is, ha a C_2 kód nem javítja a hibákat, elkerülhető az interpoláció a háromféle (0, 1, 2) C_2 pointer jel alkalmazásával és a C_1 pointer jelek másolatainak az összes hibától történő elkülönitésével.

A fent emlitett 16 dekóderben a C_1 pointer jelek átmásolásakor ott történik hibacsomó javitás, ahol a C_1 pointer jelek száma kisebb vagy egyenlő mint $(n_2 - k_2)$, és amikor hibacsomó javitásra kerül sor, a C_2 pointer jel értékét „0"-ra állitjuk be.

A fent leirtak szerint a Reed-Solomon kód dekódolása úgy történik, hogy mindegyik sorra vonatkozóan kiszámitjuk a $\sigma(z)$ hibahely meghatározó polinomot és az $\omega(z)$ hibakiértékelő polinomot, és felhasználjuk az egyes sorokban lévő n_2 szimbólum jel révén nyert szindróma jelet. Hibacsomó javitás esetében, mivel azokat a helyeket, ahol a C_1 pointer jelek értéke „1", hibahelyekként értelmezzük, lehetőség van arra, hogy az X_i hibahelyből és az $\omega(z)$ hibakiértékelő polinomból határozzuk meg az Y_i hibaértéket. Vagyis z helyére X_i^{-1}-et helyettesitve Y_i meghatározható a (8) összefüggés segitségével:

$$Y_i = \frac{\omega(X_i^{-1})}{\prod_{j=i}^{s}(1 - X_j X_i^{-1})} ,$$

ahol i = 1, 2, 3,...s; s jelöli a szimbólum jelek számát. A fenti kifejezésben a nevezőt egyedül a hibahelyek határozzák meg. Például feltéve, hogy a C_1 pointer jel által mutatott hibahelyek X_1, X_2 és X_3, a kifejezés nevezőjének alakja Y_1, Y_2 és Y_3-ra vonatkozóan a következő:

Y_1 nevezője: $(1 - X_2X_1^{-1})$ $(1 - X_3X_1^{-1})$
Y_2 nevezője: $(1 - X_1X_3^{-1})$ $(1 - X_2X_3^{-1})$
Y_3 nevezője: $(1 - X_1X_2^{-1})$ $(1 - X_3X_2^{-1})$

Itt a 13 memóriában tárolt pointer jelek azonosak a C_2 kódolású kódjelek összes k_1 sorozatára. Ezért elegendő csak egyszer elvégezni a számításokat a hibaértéket meghatározó fenti kifejezés nevezőjére nézve a k_1 sorozatokra vonatkozóan.

A 4. ábra a fent emlitett 12 és 16 dekóderrel felhasználható hibajavító dekóder felépitését mutatja. A vett kódjelek 21 bemeneti

kapocsra érkeznek, ahonnan 22 késleltető áramkörre és 23 szindróma generátorra jutnak. A 23 szindróma generátor által képzett szindróma jelek hibahely és hibaérték kiszámító 24 egységre jutnak, amelyből a kiadott hiba adatok 25 kizáró-VAGY kapura jutnak és modulo 2 hozzáadódnak a 22 késleltető áramkörön keresztül érkező vett kódjelekhez. A 22 késleltető áramkörön keresztül érkező kódjelek és a 25 kizáró-VAGY kapun keresztül érkező hibajavításnak alávetett kódjelek 26 szelektorra jutnak. a 26 szelektort a 24 egység hibahely adata vezérli. A hibahelyeknél a 26 szelektor a 25 kizáró-VAGY kapu kimenetét választja ki, hogy az kerüljön a 27 kimeneti kapocsra a vett kódjelek helyett.

Hangfrekvenciás PCM jelek rögzítését és visszajátszását végző készülék esetén a leolvasott kódjelek egy RAM memóriába kerülnek. A RAM memóriából kiolvasott kódjelek felhasználásával állapítjuk meg a szindróma jelet és ennek alapján számítjuk ki a hibahelyeket és a hibaértékeket. Az 5. ábra a hibahely és hibaérték kiszámító 24 egység egy részét mutatja. A kódjelek és szindróma jelek stb. továbbítása 28 adatbuszon történik.

Az 5. ábrán 29 szindróma regiszterben tároljuk a 28 adatbuszon, a 30 busz pufferen és a 31 kizáró-VAGY kapun keresztül érkező S_0 szindróma jelet. A $GF(2^m)$ mezőn definiált Reed-Solomon kód esetében az S_0 szindróma jel m-bites. A 29 szindróma regiszterből az S_0 szindróma jel a 31 kizáró-VAGY kapura és a 32 busz pufferre jut.

Amikor az S_0 szindróma jelet a 29 szindróma regiszterben tároljuk, a kapott Y_1, Y_2, Y_3 hibaértékeket a 28 adatbuszról érkező sorrendben a 30 busz pufferen keresztül a 31 kizáró-VAGY kapura juttatjuk. Ily módon a 31 kizáró-VAGY kapu kimenete $(S_0 \oplus Y_1)$, $(S_0 \oplus Y_1 \oplus Y_2)$ és $(S_0 \oplus Y_1 \oplus Y_2 \oplus Y_3 = Y_4)$ lesz. Az Y_4 hibaérték a 29 szindróma regiszterben marad. Az Y_4 hibaértéket a 32 busz pufferen keresztül tesszük ki a 28 adatbuszra és hibajavításra használjuk.

A 6. ábra egy másik hardver megoldást mutat a hibacsomó javításnál alkalmazott 44 dekódolóegységre. A fő 35 RAM memória 33 beírási regiszteren és 34 olvasási regiszteren keresztül van összekötve a 28 adatbusszal. A 28 adatbuszra kapcsolódik ezen kívül a 29 szindróma regiszter, a munkaterületet képviselő 36 RAM memória és a 37 logikai műveleti áramkör.

A Reed-Solomon kód szerinti hibacsomó javítás a következő n-ed rendű lineáris szimultán egyenletrendszer megoldásával nyerhető, hasonló módon a (2) kifejezéshez:

$$S_\nu = \sum_{k=1}^{n} X_k^\nu \, Y_k, \qquad (9)$$

ahol

ν = 0-tól (d-2)-ig

n : a hibacsomó hossza

X_k: a k-ik hibahely

S_ν: a szindróma jel

Y_k: a k-ik hibahelyen lévő hiba nagysága

d: a kód minimális távolsága.

Itt n, X_k és S_ν ismeretek, Y_k ismeretlen.

A fenti egyenlet megoldására hagyományosan a következő módszert alkalmazzák. Ha

$$\Lambda(z) = \prod_{i=1}^{n} (1 - X_i Z)$$

$$\omega(z) = S(z)\Lambda(z) \bmod Z^{d-1}$$

alakú, akkor Y_i a (8) kifejezéshez hasonlóan az alábbi módon nyerhető:

$$Y_i = \omega(X_i^{-1}) \; / \prod_{j \neq i}^{n} (1 - X_j X_i^{-1}). \qquad (10)$$

Ezzel az eljárással azonban, ha például d=9 és n=8 esetére megnézzük a szükséges műveleti lépések számát, akkor azt kapjuk, hogy:

(i) $\Lambda(z)$ sorbafejtésére a szorzások száma: $1 + 2 + \ldots + 7 = 28$, az összeadások száma: $1 + 2 + \ldots + 7 = 28$.

(ii) Az Y_i, $\prod_{j=1}^{8}{}_{j \neq i} (1 - X_j X_i^{-1})$ nevezőjének előzetes kiszámításánál
a reciprok képzések száma: $1 \times 8 = 8$,
a szorzások száma: $(7 + 6) \times 8 = 104$,
az összeadások száma: $7 \times 8 = 56$.

(iii) Az $\omega(z) = S(z)\Lambda(z) \bmod Z^8$ kifejezés kiszámításánál
a szorzatok száma: $1 + 2 + \ldots + 7 = 28$,
az összeadások száma: $1 + 2 + \ldots + 7 = 28$.

(iv) Az $\omega(X_i^{-1})$ kiszámításához
a reciprok képzések száma: $1 \times 8 = 8$,
a szorzások száma: $7 \times 8 = 56$,
az összeadások száma: $7 \times 8 = 56$.

(v) Az Y_i kiszámításához az osztások száma: $1 \times 8 = 8$.

Ha ezen számítások mindegyikét egy műveleti lépésnek tekintjük, akkor összesen 408 műveleti lépésre van szükség.

A 6. ábrán mutatott áramkörben a (9) kifejezés gyökeit az alábbiak szerint számítjuk ki:

$$Y_i = \Big(\sum_{j=0}^{n-1} \Lambda_{nj} \; S_{j+l}\Big) \; / \; X_i^l \prod_{k=1, k \neq i}^{n} (X_k + X_i), \qquad (11)$$

ahol

Λ_{nj}: Z^j együtthatója $\left[\prod_{k=1, k \neq i}^{n} (z + X_k)\right]$

alakban és

l: bármely nullánál nagyobb vagy azzal egyenlő egész szám, amely kielégíti az $l \leq d - n - 1$ egyenlőtlenséget.

Vagyis ahhoz, hogy hibacsomó javítást hajtsunk végre, a (11) kifejezésbe $l = 0$ és $i = n$ értékeket helyettesítünk, így

$$Y_n = \left(\sum_{j=0}^{n-1} \Lambda_{nnj} \; S_j\right) / \; / \prod_{k=1}^{n-1} (X_k + X_n). \quad (11')$$

Ezen kifejezés kiszámításának eredményeképpen megkapjuk Y_n értékét és minden egyes S szindróma jelhez hozzáadjuk $Y_n X_n^{\nu}$ értékét az alábbiak szerint:

$$S_{\nu} \leftarrow S_{\nu} + Y_n X_n^{\nu},$$

ahol $\nu = 0,\ldots, n-2$.

Mivel most már az X_n helyen lévő adat hibátlan, a szindróma jel (n-1) hosszúságú hibacsomót foglal magában. Ennélfogva n értékét 1-gyel csökkentve megkaphatjuk Y_{n-1} értékét:

$$Y_{n-1} = \left(\sum_{j=0}^{n-2} \Lambda_{n-1,\; n-1,\; j,\; s_j}\right) / \; / \prod_{k=1}^{n-2} (X_k + X_{n-1}).$$

Ezen kifejezés kiszámításának eredményeképpen megkapjuk Y_{n-1} értékét, és az $Y_{n-1} X_{n-1}^{\nu}$ értéket minden egyes szindróma jelhez hozzáadjuk a következőképpen:

$$S_{\nu} \leftarrow S_{\nu} + Y_{n-1} \; X_{n-1}^{\nu},$$

ahol $\nu = 0,\ldots, n-3$.

A fenti eljárást ismételve az utolsó Y_1 hibaértéket az alábbi módon kapjuk meg:

$Y_1 = S_0$.

A fent ismertetett eljárással hibacsomó javítást lehet megvalósítani. Ez esetben a szükséges műveletek lépésszámát ugyanúgy számítjuk ki, mint a hagyományos módszer esetében, feltételezve, hogy d=9 és n=8:

(i) Λ_{nnj} sorbafejlesztéséhez a szorzások száma $1 + 2 + \ldots + 6 = 21$, az összeadások száma: $1 + 2 + \ldots + 6 = 21$.

(ii) Előzetes számítások az Y_n, $\Pi_n = \prod_{k=1}^{n-1} (X_k + X_n)$ alakú nevezőjének kiszámítására, ahol elegendő csak a $\Pi_3, \ldots \Pi_8$ értékeket kiszámítani, mivel $\Pi_2 = X_1 + Y_2 = \Lambda_{331}$,
a szorzások száma: $1 + 2 + \ldots + 6 = 21$,
az összeadások száma: $1 + 2 + \ldots + 6 = 21$.

(iii) Az Y_n számlálójának kiszámítására
a szorzások száma: $7 + 6 + \ldots + 1 = 28$,
az összeadások száma: $7 + 6 + \ldots + 1 = 28$.

(iv) Mivel $Y_1 = S_0$, az Y_n kiszámításához szükséges osztások száma: 7.

(v) $S_{\nu} \leftarrow S_{\nu} + Y_n X_n^{\nu}$ művelethez
a szorzások száma: $6 + 5 + \ldots + 1 = 21$,
az összeadások száma: $7 + 6 + \ldots + 1 = 28$.

A fenti műveletek összes lépésszáma tehát 196.

Ezért a (11) összefüggés használatával 50 százalékkal csökkenthető a számítási műveletek lépésszáma a hagyományos (10) összefüggés alkalmazásához képest.

Továbbá abban az esetben, ha a fenti hibajavító kód egy szorzat kód és feltételezzük, hogy 30 szimbólum jelet helyezünk el függőleges irányban és 128 szimbólum jelet a vízszintes irányban, valamint, hogy a C_2 kódot a vízszintes irányban képezzük, a hibacsomó a C_1 pointer jelek szerinti helynek felel meg, és az X_k ($k = 1,\ldots, n$) hibahely az összes C_2 kód sorozatban ugyanaz. Ez azt jelenti, hogy előre ki lehet számítani a (11') kifejezésben szereplő

$$\Lambda_{nnj} \text{ és } \prod_{k=1}^{n-1} (X_k + X_n)$$

értékeket és nem szükséges ezeket a műveleteket minden C_2 dekódolásnál újra végrehajtani. Ez azt jelenti, hogy jelen példában a fenti műveleteket csak egyszer kell elvégezni, mialatt a C_2 dekódolást 30-szor hajtjuk végre.

Ezért a fenti műveleti lépések számában azt is figyelembe kell venni, hogy az (i) és (ii) kiszámítására a 30-szor végrehajtott C_2 dekódolás során csak egyszer kerül sor, így feltételezve azt, hogy az (i) és (ii) kifejezések kiszámításának átlagos lépésszáma 84/30 = 2,8, az összes számítási lépésszámra 115 adódik, amit ha összehasonlítunk a hagyományos eljárásra nyert értékekkel, ahol az (i) és (ii) kifejezések kiszámításának átlagos lépésszáma 224/30 = 7,5 és az összes számítási lépésszám 191,5, akkor látható, hogy a szükséges lépésszámot mintegy 40 százalékkal tudjuk csökkenteni.

Következésképpen a fent leírt eljárásnak megvan az az előnye, hogy jelentősen csökkenti a számítási lépések számát, a feldolgozási időt, a feldolgozás hardver terhelését stb. a hagyományos eljáráshoz képest.

A fenti számítások elvégzésére a 6. ábrán mutatott 37 logikai műveleti áramkört alkalmazzuk és az $S_{\nu} \leftarrow S_{\nu} + X_i^{\nu} Y_i$ kiszámítására a 7. ábrán mutatott kiviteli alakot használhatjuk. A 7. ábrán a 28 adatbuszhoz 29 szindróma regiszter, 36 RAM memória, 38 és 40 regiszter és 43 szelektor van csatlakoztatva. A 43 szelektor kimenetére 39 regiszter csatlakozik. A 38 és 39 regiszterek kimenetei 41 összeadó bemeneteire, a 39 és 40 regiszterek kimenetei 42 szorzó bemeneteire vannak csatlakoztatva. A 41 összeadó kimenete a 28 adatbuszhoz, a 42 szorzó kime-

8

nete a 43 szelektor egy további bemenetére csatlakozik.

Az áramkörben

(1) az S_0 jel a 38 regiszterbe, az Y_1 jel a 39 regiszterbe és az X_1 jel a 40 regiszterbe kerül a 28 adatbuszon keresztül a 29 szindróma regiszterből, illetve a munkaterületet képviselő 36 RAM memóriából. Az $S_0 + Y_1$ jel a 41 összeadóból kerül ki a 28 adatbuszra.

(2) A 42 szorzó X_1Y_1 tartalma a 43 szelektoron keresztül vissza van csatolva a 39 regiszterbe és az S_1 jel a 28 adatbuszról bekerül a 38 regiszterbe. Ily módon a 41 összeadó kimenetéről az $S_1 + X_1Y_1$ jel kerül ki a 28 adatbuszra.

(3) Továbbá a 42 szorzó $X_1^2Y_1$ tartalma a 43 szelektoron keresztül vissza van csatolva a 39 regiszterbe és az S_2 jel a 28 adatbuszról bekerül a 38 regiszterbe. Ily módon a 41 összeadó kimenetéről az $S_2 + X_1^2Y_1$ jel kerül ki a 28 adatbuszra.

(4) A fenti lépések ismétlése révén $S_3 + X_1^3Y_1$, $S_4 + X^4{}_1Y_1$,... jelek értékeit kapjuk meg egymást követően, amelyeket a 28 adatbuszon keresztül juttatunk el a 29 szindróma regiszterbe az összes érték átírása céljából. A számítások végrehajtása a fenti lépésekben történik.

A fenti ismertetésben, bár az összes $X_1,...X_n$ hibacsomóként szerepel, abban az esetben, ha csak $X_1,...X_{n-1}$ a hibacsomó és X_n egy hiba, lehetséges a szimbólum jelek javítása. Ez esetben az (n + 1) darab ismeretlen $Y_1,...Y_n$ és X_n mennyiség megkapható Λ_{nnj} felhasználásával a következőképpen:

$$X_n = \frac{\sum_{j=0}^{n-1} \Lambda_{nnj} S_{j+1}}{\sum_{j=0}^{n-1} \Lambda_{nnj} S_{j+1}} \qquad (12)$$

Ezáltal ismeretlen mennyiségnek $Y_1,... Y_n$ maradnak és ezeket ugyanúgy kaphatjuk meg, mint a hibacsomó javításnál.

Például abban az esetben, amikor a szorzat kódban d=9 (6-os hibacsomó + 1 hiba):

(1) Ellenőrizzük az X_1, X_2,... X_6 hibacsomó helyeket.

(2) Megállapítjuk és tároljuk az Λ_{nnj} értékeket, itt összesen 21 értéket: $\Lambda_{221} = X_1$, $\Lambda_{331} = \Lambda_{221} + X_2$, $\Lambda_{332} = \Lambda_{221} X_2$,..., Λ_{771},... Λ_{776}.

(3) Kiszámítjuk és tárojuk a $\prod_{k=1}^{n-1}$ $(X_k + X_n)$ kifejezés meghatározása során nyert öt szimbólumot (n = 2, ...6).

A fenti feldolgozási lépéseket 30 esetre vonatkozóan csak egyszer kell elvégezni.

(4) Kiszámítjuk az $S_0,...S_7$ szindróma jeleket.

(5) Kiszámítjuk X_7 értékét az alábbi kifejezés alapján

$$X_7 = \frac{\sum_{j=0}^{6} \Lambda_{77j} S_{j+1}}{\sum_{j=0}^{6} \Lambda_{77j} S_j}$$

(6) Kiszámítjuk Y_7 értékét a következő kifejezés segítségével és visszacsatoljuk az Y_7X_7 szindróma jelet:

$$Y_7 = \left(\sum_{j=0}^{6} \Lambda_{77j} S_j\right) / \prod_{k=1}^{n-1} (X_k + X_7).$$

Ezután az $Y_6,...Y_1$ (=S_0) mennyiségeket egymás után megkapjuk.

Bár ezen példában egy olyan esetet mutattunk be, ahol a (11') kifejezést használtuk, a művelet ugyanaz, ha $l \neq 0$.

A fent bemutatott eljárás szerint lehetőség van hibacsomókat és egy hibát tartalmazó hibák javítására. Ez esetben a műveleti lépések száma ugyanúgy csökkenthető, mint a fent leírt hibacsomó javítási eljárásnál.

Az alábbiakban a (11) és (12) kifejezések bizonyítását adjuk meg:

Lemma: Ha

$$\Lambda_{mi}(z) = \prod_{\substack{k=1 \\ k\neq i}}^{n} (z + X_k) \text{ és}$$

$$n_{ij} = [\Lambda_{mi}(z)]_j,$$

akkor ebből a következő kifejezést nyerjük:

$$\sum_{j=0}^{n-1} \Lambda_{nij} Z^j = \prod_{\substack{k=1 \\ k\neq i}}^{n} (z + X_k).$$

1. Tétel: Az alábbi n-ed rendű lineáris szimultán egyenletrendszernek

$$S_\nu = \sum_{k=1}^{n} X_k^{\nu} Y_k$$

(ahol $\nu = 0,..., n-1$; Y_i az ismeretlen) a következő gyökei vannak:

$$Y_i = \left(\sum_{j=0}^{n-1} \Lambda_{nij} S_j\right) / \prod_{\substack{k=1 \\ k=i}}^{n} (X_k + X_i).$$

Bizonyítás:

Jobb oldal = $\left(\sum_{j=0}^{n-1} \Lambda_{nij} \sum_{k=1}^{n} X_k{}^j Y_k\right) /$

$$/ \prod_{\substack{k=1 \\ k=i}}^{n} (X_k + X_i) = \left(\sum_{k=1}^{n} Y_k \sum_{j=0}^{n-1} \Lambda_{nij} X_k{}^j\right) /$$

$$/ \prod_{\substack{k=1 \\ k=i}}^{n} (X_k + X_i) = \left[\sum_{k=1}^{n} Y_k \prod_{\substack{j=1 \\ j=i}}^{n} (x_k + X_j)\right] /$$

$$/\prod_{\substack{k=1\\k=i}}^{n}(X_k + X_i) = \left[Y_i \prod_{j=1\ j=i}^{n}(X_k + X_j)\right] /$$

$$/\prod_{\substack{k=1\\k=i}}^{n}(X_k + X_i) = Y_i.$$

1. Következmény:

$$Y_i = \left(\sum_{j=0}^{n-1} \Lambda_{nij}\ S_{j+1}/X_i^j\right) \prod_{\substack{k=1\\k=i}}^{n}(X_k + X_i),$$

ahol ($l \geq 0$).

Bizonyítás: Az alábbi helyettesítést alkalmazva

$$S_{j+1} = \sum_{k=1}^{n} X_k^{j+1} Y_k.$$

2. Tétel: Ha $X_1, \ldots X_{n-1}$ hibacsomó és X_n hiba,

$$X_n = \frac{\sum_{j=0}^{n-1} \Lambda_{nnj}\ S_{j+1}}{\sum_{j=0}^{n-1} \Lambda_{nnj}\ S_j}.$$

Bizonyítás: Az 1. következményből

$$\text{Jobb oldal} = \frac{Y_n X_n^{l+1} \prod_{k=1}^{n-1}(X_k + X_n)}{Y_n X_n^{l} \prod_{k=1}^{n-1}(X_k + X_n)} = \text{Bal oldal}.$$

Ennélfogva az 1. tételből következik, hogy a hibacsomó javítás során lehet hibákat javítani az $S_1 - S_n$, $S_2 - S_{n+1}, \ldots$, $S_{d-1-n} - S_{d-2}$ szindróma jelek bármelyikének felhasználásával az $S_0 - S_{n-1}$ sorozatból. Vagyis n hosszúságú hibacsomó javításához n folytonos szindróma jelre van szükség, a további szindróma jelek ellenőrzési célokra szolgálhatnak, amennyiben $n \leq d-1$.

Továbbá ahhoz, hogy a 2. tételből megkapjuk X_n értékét, az $S_1 - S_{n+1}$ összesen n+1 szindróma jelre van szükség, amennyiben $n \leq d-2$. Ez esetben a hibacsomón belüli hibák száma $n-1 \leq d-3$. A fennmaradó szindróma jelek a fenti esethez hasonlóan ellenőrzési célokra használhatók.

A hagyományos eljárásnál a hibajavító kódnak megfelelő (n_1, n_2) számú pointer területre van szükség. A jelen találmány szerint azonban a pointer jelek számát ($n_2 + 2n_1$) értékre lehet csökkenteni, tovább csökkentve a dekódoláshoz szükséges memória kapacitást. Továbbá a jelen találmány lehetővé teszi a pointer jelek beírási és kiolvasási lépésszámának csökkentését. Továbbá jelen találmány lehetővé teszi, hogy ebben az esetben, amikor a C_2 dekódolás során történő hibacsomó javítás a C_1 dekódolás során képzett pointer jelek felhasználásával történik, annak következtében, hogy a pointer jelek mindegyik C_2 kódjelsorozatra vonatkozóan azonosak, és hogy a hibaértékek kiszámítási műveleteinek egy része közös, lehetőség van arra, hogy a műveleteket csak egyszer hajtsuk végre. Ennélfogva lehetőség van arra, hogy jelentősen csökkentsük a dekódolást végző feldolgozási eljárás lépéseinek számát és nagysebességű műveletvégzést valósítunk meg a dekódolás során. Továbbá jelen találmány szerint nincs szükség arra, hogy bonyolult hibaértékelő polinom segítségével határozzuk meg az összes hibaértéket, hanem lehetőség van arra, hogy az egyik hibaértéket egyszerű módon számítsuk ki, és hogy csökkentsük a feldolgozási lépések számát. Ezenfelül a jelen találmány szerint lehetőség van a hibacsomó javítás számítási lépéseinek jelentős csökkentésére.

SZABADALMI IGÉNYPONTOK

1. Eljárás hibajavító kódolású kódjelek dekódolására, amelynél egy k_1 x k_2 kétdimenziós elrendezés oszlopaiban lévő minden egyes k_1 információs szimbólum jel számára n_1 kód hosszúságú első hibajavító kódolású kódjelek, soraiban lévő minden egyes k_2 információs szimbólum jel számára pedig n_2 kód hosszúságú második hibajavító kódolású kódjelek vannak, amely eljárás során veszszük legalább az első hibajavító kódolású kódjeleket és ezeket dekódoljuk; az első hibajavító kódolású kódjelek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képezünk és ezeket egy legalább n_2-bites memóriában tároljuk; dekódoljuk a második hibajavító kódolású kódjeleket; a második hibajavító kódolású kódjelek hibaészlelési vagy hibajavítási státuszát megadó második pointer jeleket képezünk és ezeket egy legalább k_1-bites memóriában tároljuk; majd kiadjuk az információs szimbólum jeleket, *azzal jellemezve,* hogy az első és a második pointer jelek állapotának alapján az információs szimbólum jelek megbízhatóságát mutató jelölőjelet állítunk elő. (Elsőbbség: 1983.12.20.)

2. Eljárás hibajavító kódolású kódjelek dekódolására, amelynél egy k_1 x k_2 kétdimenziós elrendezés oszlopaiban lévő minden egyes k_1 információs szimbólum jel számára n_1 kód hosszúságú első hibajavító kódolású kódjelek, soraiban lévő minden egyes k_2 információs szimbólum jel számára pedig n_2 kód hosszúságú második hibajavító kódolású kódjelek vannak, amely eljárás során veszszük legalább az első hibajavító kódolású kódjeleket és ezeket dekódoljuk; az első hibajavító kódolású kódjelek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képezünk és ezeket egy legalább n_2-bites memóriában tároljuk; dekódoljuk a má-

sodik hibajavító kódolású kódjeleket az első pointer jelek alkalmazásával oly módon, hogy a második hibajavító kódolású kódjelek minden egyes sorozatára végrehajtunk egy hibacsomó javítást; majd kiadjuk az információs szimbólum jeleket, *azzal jellemezve*, hogy a második hibajavító kódolású kódjelek dekódolása során a hibacsomó javításban a hibaérték kiszámítási műveletek egy részét csak egyszer végezzük el a második hibajavító kódolású kódjelek minden egyes sorozatára. (Elsőbbség: 1984.12.19).

3. Berendezés hibajavító kódolású kódjelek dekódolására, amelyben egy k_1 x k_2 kétdimenziós elrendezés oszlopaiban lévő minden egyes k_1 információs szimbólum jel számára n_1 kód hosszúságú első hibajavító kódolású kódjelek, soraiban lévő minden egyes k_2 információs szimbólum jel számára pedig n_2 kód hosszúságú második hibajavító kódolású kódjelek vannak, amely berendezésnek legalább az első hibajavító kódolású kódjeleket vevő szerve, ehhez csatlakoztatott, az első hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képező, valamint a második hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó második pointer jeleket képező dekódolóegysége, a dekódolóegységhez csatlakoztatott, az első pointer jeleket tároló legalább n_2-bites első memóriája, a dekódolóegységhez csatlakoztatott, a második pointer jeleket tároló legalább k_1-bites második memóriája és a dekódolóegységhez csatlakoztatott, az információs szimbólum jeleket előállító kimenőegysége van, *azzal jellemezve*, hogy az első és a második memóriához (13, 17) az információs szimbólum jelek megbízhatóságát mutató jelölőjelet előállító egység (19) van csatlakoztatva. (Elsőbbség: 1983.12.20.)

4. A 3. igénypont szerinti berendezés, *azzal jellemezve*, hogy a dekódolóegységnek a vevő szervhez (11) csatlakoztatott, az első hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képező első dekódere (12) és az első dekóderhez (12) csatlakoztatott, a második hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó második pointer jeleket képező második dekódere (16) van, továbbá az első memória (13) az első dekóderhez (12), a második memória (17) a második dekóderhez (16), a kimenőegység (18) pedig a második dekóderhez (16) van csatlakoztatva. (Elsőbbség: 1983.12.20.)

5. Berendezés hibajavító kódolású kódjelek dekódolására, amelyben egy k_1 x k_2 kétdimenziós elrendezés oszlopaiban lévő minden egyes k_1 információs szimbólum jel számára n_1 kód hosszúságú első hibajavító kódolású kódjelek, soraiban lévő minden egyes k_2 információs szimbólum jel számára pedig n_2 kód hosszúságú második hibajavító kódolású kódjelek vannak, amely berendezésnek legalább az első hibajavító kódolású kódjeleket vevő szerve, ehhez csatlakoztatott, az első hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képező első dekódere, az első dekóderhez csatlakozatatott, az első pointer jeleket tároló legalább n_2-bites első memóriája, az első dekóderhez és az első memóriához csatlakoztatott, a második hibajavító kódolású kódjeleket az első pointer jelek alkalmazásával a második hibajavító kódolású kódjelek minden egyes sorozatára egy hibacsomó javítást végrehajtva dekódoló második dekódere és az információs szimbólum jeleket előállító kimenőegysége van, *azzal jellemezve*, hogy a hibacsomó javításban a második hibajavító kódolású kódjelek minden egyes sorozatára a hibaérték kiszámítási műveletek egy részét csak egyszer elvégző második dekódere (16) van. (Elsőbbség: 1984.12.19.)

6. Berendezés hibajavító kódolású kódjelek dekódolására, amelyben egy k_1 x k_2 kétdimenziós elrendezés oszlopaiban lévő minden egyes k_1 információs szimbólum jel számára n_1 kód hosszúságú első hibajavító kódolású kódjelek, soraiban lévő minden egyes k_2 információs szimbólum jel számára pedig n_2 kód hosszúságú második hibajavító kódolású kódjelek vannak, amely berendezésnek legalább az első hibajavító kódolású kódjeleket vevő és az információs szimbólum jeleket előállító vevő és kimenő szerve, ehhez csatlakoztatott, az első hibajavító kódolású kódjeleket dekódoló és ezek hibaészlelési vagy hibajavítási státuszát megadó első pointer jeleket képező, valamint a második hibajavító kódolású kódjeleket az első pointer jelek alkalmazásával a második hibajavító kódolású kódjelek minden egyes sorozatára egy hibacsomó javítást végrehajtva dekódoló dekódolóegysége és a dekódolóegységhez csatlakoztatott, az első pointer jeleket tároló legalább n_2-bites memóriája van, *azzal jellemezve*, hogy a hibacsomó javításban a második hibajavító kódolású kódjelek minden egyes sorozatára a hibaérték kiszámítási műveletek egy részét csak egyszer elvégző dekódolóegysége (44) van. (Elsőbbség: 1984.12.19.)

7. Berendezés hibajavító kódolású kódjelek dekódolására, amely kódolás nem-duálisztikus és ahol információs szimbólum jelek modulo kettő szerinti összeadása több szindróma jel egyikét eredményezi, továbbá amellyel az információs szimbólum jelek sorozatában több információs szimbólum jel hibája javítható, amely berendezésnek dekódolóegysége, bemenőegysége, szindróma jeleket előállító szindróma generátora, ehhez csatlakoztatott, a szindróma jelekből hibahelyet és hibaértéket kiszámító egysége, a bemenőegységhez, valamint a hibahely és hibaérték ki-

11

számító egységhez csatlakoztatott, a hibaérték összegzésével egy hibás információs szimbólum jelet javító hibajavító egysége és a hibajavító egységhez csatlakoztatott kimenőegysége van, azzal jellemezve, hogy a hibahely és hibaérték kiszámító egységnek (24) a szindróma jeleket tároló szindróma regisztere (29) és a szindróma regiszterhez (29) csatlakoztatott, a hibaértéknek a szindróma jelekhez való hozzáadásával a szindróma regiszter (29) számára hibaértéket előállító kizáró-VAGY kapuja (31) van. (Elsőbbség: 1983.12.23.).

Kiadja az Országos Találmányi Hivatal, Budapest - A kiadásért felel: Himer Zoltán osztályvezető
R 4892 - KJK

90.2642.66-13-2 Alföldi Nyomda Debrecen - Felelős vezető: Benkő István vezérigazgató

FIG.1

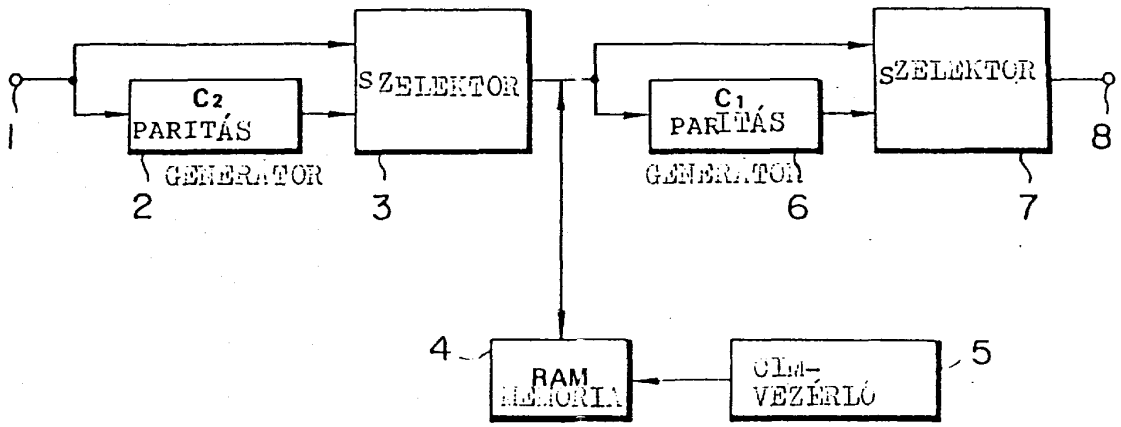

1
C2
PARITÁS
GENERÁTOR
2
SZELEKTOR
3
C1
PARITÁS
GENERÁTOR
6
SZELEKTOR
7
8
4
RAM
MEMÓRIA
CÍM-
VEZÉRLŐ
5

FIG.2

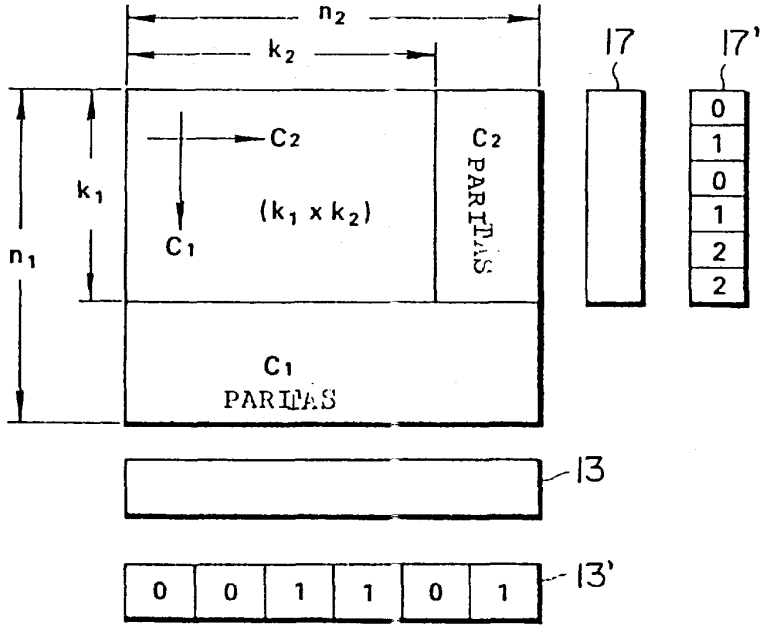

n2
k2
17
17'
C2
C2
PARITÁS
k1
(k1 x k2)
n1
C1
C1
PARITÁS
0
1
0
1
2
2
13
13'
0 0 1 1 0 1

FIG.3

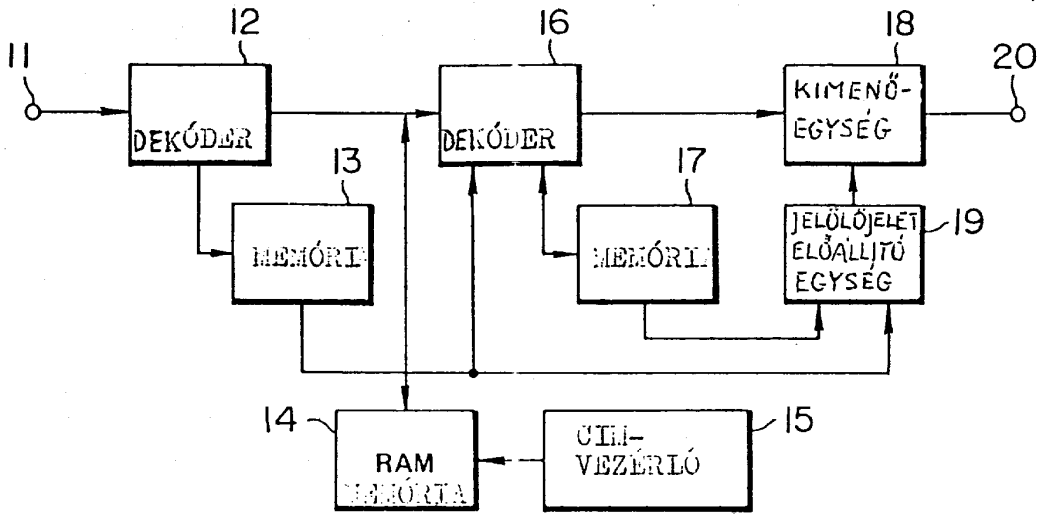


FIG.4

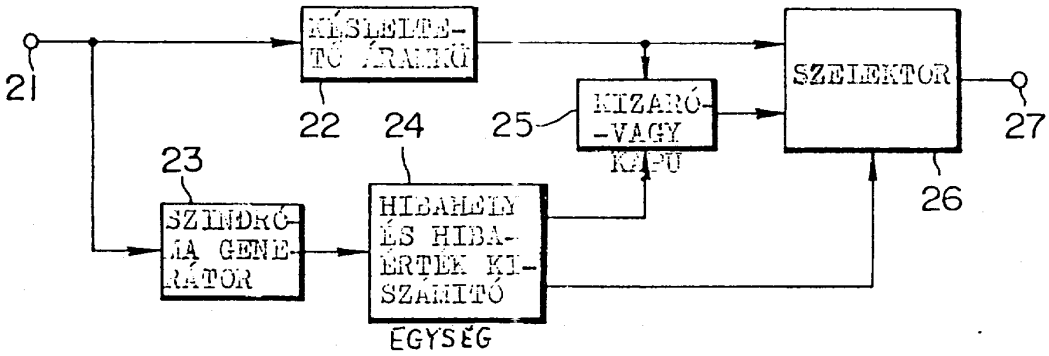


FIG.5

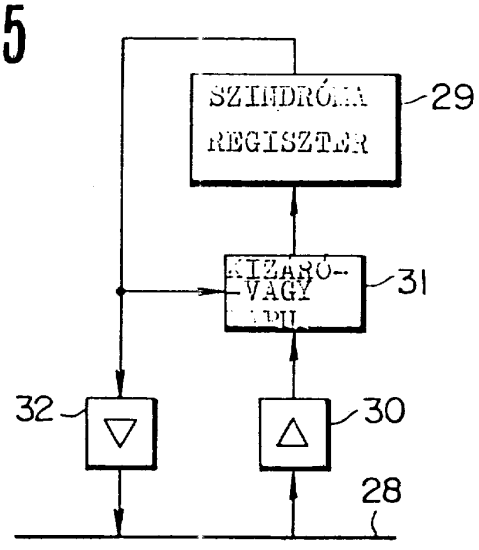


FIG.6

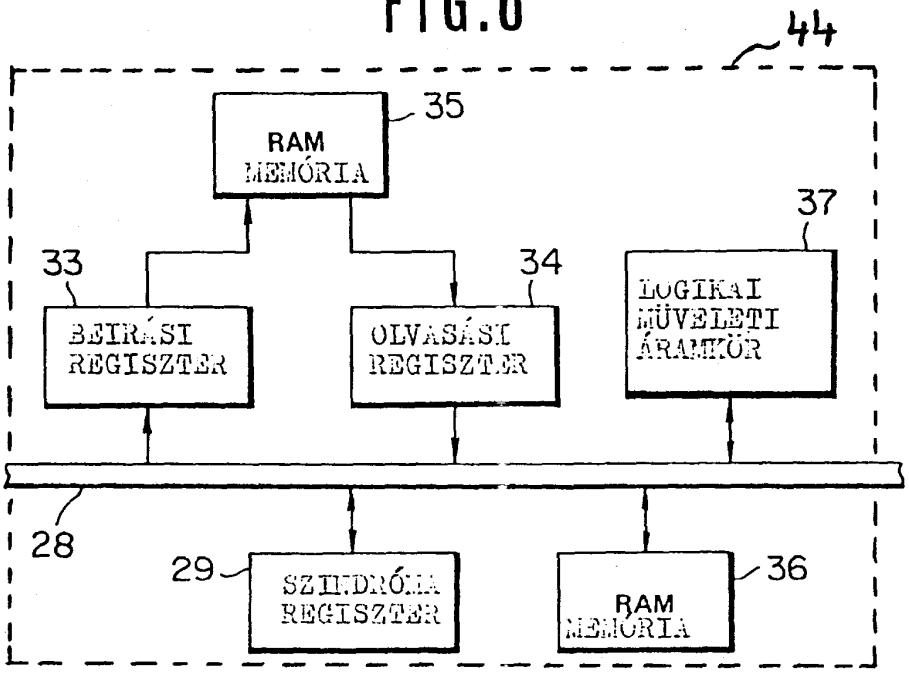
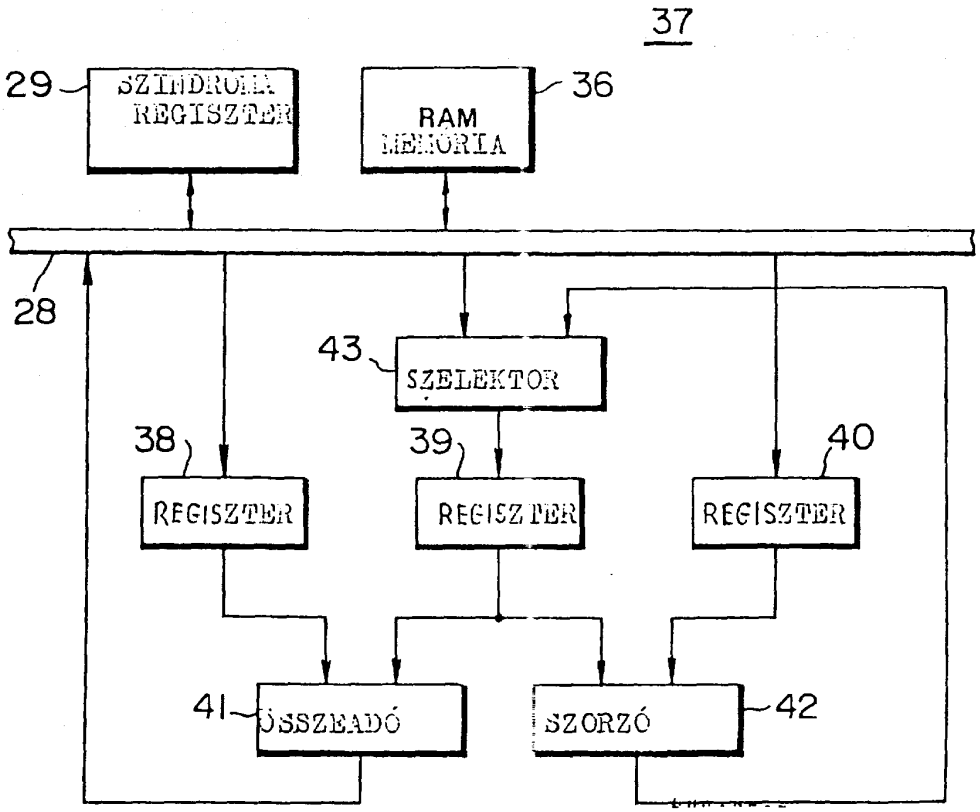

44
35
RAM
MEMÓRIA
37
33
34
BEIRÁSI
REGISZTER
OLVASÁSI
REGISZTER
LOGIKAI
MÜVELETI
ÁRAMKÖR
28
29
SZINDRÓMA
REGISZTER
RAM
MEMÓRIA
36

FIG.7


37
29
SZINDRÓMA
REGISZTER
RAM
MEMÓRIA
36
28
43
SZELEKTOR
38
39
40
REGISZTER
REGISZTER
REGISZTER
41
ÖSSZEADÓ
SZORZÓ
42