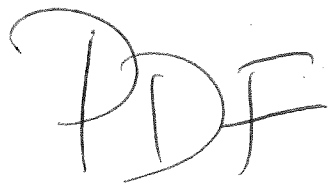
CONSIGLIO NAZIONALE DELLE RICERCHE
ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

RILEVAZIONE E CORREZIONE DI ERRORI NELLE UNITA' DI CALCOLO

M. Vannneschi*

Nota Interna B70/15
(Pisa 1970)



(*) Attualmente presso la SNAM PROGETTI, PISA

I N D I C E

I

INTRODUZIONE

Lo scopo di questa nota è essenzialmente quello di fornire una panoramica dei concetti fondamentali e delle tecniche, che sono alla base della *rilevazione* e della *correzione* di errori introdotti nella *elaborazione aritmetica di dati*, e, quindi, di errori compiuti dall'unità aritmetica di un calcolatore, in seguito ad un guasto avvenuto.

Facciamo rilevare esplicitamente, dunque, che non ci occupiamo degli errori introdotti nella trasmissione dei dati.

Le tecniche sopra dette sono svariate e, a prima vista, assai dissimili tra di loro: noi ci limiteremo ad esaminare le principali e, particolarmente, quelle che ci sono sembrate le più utili, sia dal punto di vista *informativo* (in previsione di uno studio più approfondito), sia da quello della loro effettiva *realizzabilità*.

Di tali tecniche cercheremo di mettere in evidenza le *caratteristiche* che esse hanno in comune e le differenze, nonchè i *limiti* che esse hanno riguardo alla rilevazione ed alla correzione di errori: cioè, in ultima analisi, quali sono i tipi di errori che esse sono in grado di rilevare o di correggere.

Prima di iniziare l'esposizione delle tecniche da noi scelte, è opportuno introdurre alcuni *concetti fondamentali*, di natura essenzialmente algebrica, che vedremo essere *alla* base di tutta la discussione.

1 - IL CONCETTO DI CONGRUENZA(1)

Consideriamo la seguente relazione:

$$A = r + mt$$

dove A, r, m, t sono numeri interi. Essa esprime la definizione di congruenza, cioè che "*A è congruente ad r modulo m*", e si scrive:

$$A \equiv r \bmod m$$

r è detto anche il "*RESIDUO di A modulo m*".
Ad esempio, sono valide le seguenti congruenze:

$$10 \equiv 7 \bmod 3$$
$$10 \equiv 4 \bmod 3$$
$$10 \equiv 1 \bmod 3$$

Particolare importanza assume il *minimo residuo positivo* (nell'esempio precedente: 1) detto senz'altro, d'ora in poi, senza altre specificazioni, *il "RESIDUO di A modulo m"*. Dunque, si ha sempre:

(1) $$\boxed{0 \leq r < m}$$

La scrittura che si userà quasi sempre per indicare il calcolo dal residuo r di un numero A, con un certo modulo m, è:

$$r = |A|_m$$

2 - *PROPRIETA' FONDAMENTALI DELLA CONGRUENZA*

Dalla (1) deriva subito la seguente proprietà:

$$(2) \qquad \boxed{\left|\left|A\right|_m\right|_m = \left|A\right|_m}$$

In base alla (2) ed alla definizione di congruenza, si ha allora:

$$(3) \qquad \boxed{\left|A \pm km\right|_m = \left|A\right|_m} \qquad (k \text{ intero})$$

Inoltre, si ha anche:

$$(4) \qquad \boxed{\left|A \pm B\right|_m = \left|\left|A\right|_m \pm \left|B\right|_m\right|_m}$$

Infatti:

$$\left|A \pm B\right|_m = \left|\left|A\right|_m + mt_1 + \left|B\right|_m + mt_2\right|_m =$$

$$= \left|\left|A\right|_m + \left|B\right|_m + (t_1 + t_2)m\right|_m = \left|\left|A\right|_m + \left|B\right|_m\right|_m$$

Ad esempio:

$$\left|19+8\right|_5 = \begin{cases} = \left|27\right|_5 = 2 \\ \\ = \left|\left|19\right|_5 + \left|8\right|_5\right|_5 = \left|4+3\right|_5 = \left|7\right|_5 = 2 \end{cases}$$

Analogamente si verifica la seguente proprietà:

$$(5) \qquad \boxed{\left|A \cdot B\right|_m = \left|\left|A\right|_m \cdot \left|B\right|_m\right|_m}$$

Ad esempio:

$$|19.8|_5 = \begin{cases} = |152|_5 = 2 \\ = ||19|_5 \cdot |8|_5|_5 = |4.3|_5 = |12|_5 = 2 \end{cases}$$

E' ovvio, quindi, in base alla (5), che:

(6) $$\boxed{|m.A|_m = 0}$$

Per quanto riguarda *interi negativi*, si ha la seguente relazione (A > 0).

(7) $$\boxed{|-A|_m = |m - |A|_m|_m}$$

Facciamo notare esplicitamente che *non vale una proprietà del tipo (4) o del tipo (5) per la divisione*, salvo alcune restrizioni che non hanno grande interesse ai nostri fini, e che, quindi, non stiamo ad esporre.

Due teoremi (che non dimostriamo) assai importanti sono espressi dalle seguenti relazioni:

(8) $$\boxed{|ab^n|_{b-1} = |a|_{b-1}}$$

(9) $$\boxed{|ab^n|_{b+1} = \begin{cases} |+a|_{b+1}: \text{ n pari} \\ |-a|_{b+1}: \text{ n dispari} \end{cases}}$$

3 - CONTROLLO DI PARITA' E CODICI RESIDUI

E' noto come, per un *sistema binario*, il *controllo di parità* sia definito in termini di *parità o disparità del numero dei bits della parola eguali ad 1*: se essi sono in numero

pari, cioè, si introduce un *parity check digit* che vale 0 ; se essi sono in numero dispari, tale digit vale 1.

Sotto tale forma, il controllo di parità è sufficiente per comprendere come si possa ottenere *la rilevazione di un errore singolo in un codice di trasmissione* (cioè, l'alterazione di un bit della parola durante la sua trasmissione).

Ai fini dello studio delle tecniche di rilevazione di errori aritmetici, dei quali intendiamo occuparci, è utile una generalizzazione di questa definizione, basata sulle congruenze, anche in vista di una presentazione unitaria delle tecniche di rilevazione, basate sulla parità, e di quelle basate sull'uso di codici residui, che introdurremo in seguito, e che permettono anche la correzione di errori.

II

CONTROLLO DI PARITA'

In linea del tutto generale[2], *il controllo di parità di una certa operazione aritmetica* si esegua nel seguente modo: si calcola, in base al *concetto di congruenza* e con *un opportuno modulo*, il "*check digit*" a partire dal risultato e lo si confronta con una determinata funzione dei check digits a partire *dagli operandi*. Dal confronto si può desumere se l'unità aritmetica ha introdotto o no un errore, appartenente alla classe degli errori rilevabili.

Vediamo, dunque, anzitutto, *come si calcola il "check digit" di un determinato numero N*.

Se questo è composto di n digits, e se b è la base della numerazione, esso può essere espresso nella nota forma polinomiale:

$$N = \sum_{i=1}^{n} a_i \, b^{i-1}$$

Il check digit, associato ad esso, è definito al seguente modo:

(7)
$$\boxed{p = \left| \sum_{i=1}^{n} a_i \right|_b}$$

In base alla proprietà (4), ed indicando con $\oplus$ l'*addi-*

zione modulo b, si può anche scrivere:

$$(8) \qquad p = \left| \sum_{i=1}^{n} |a_i|_b \right|_b = a_n \oplus a_{n-1} \oplus \ldots \oplus a_2 \oplus a_1$$

Ad esempio: $N = 12894$, $m = b = 10$

$$\begin{aligned} p &= 1 \oplus 2 \oplus 8 \oplus 9 \oplus 4 = \\ &= 3 \oplus 8 \oplus 9 \oplus 4 = \\ &= 1 \oplus 9 \oplus 4 = \\ &= 0 \oplus 4 = 4 \end{aligned}$$

Nel caso di *rappresentazione binaria:*

$$m = b = 2$$

si può ora agevolmente *ritrovare il concetto dal bit del controllo di parità*. Infatti, in tal caso, il simbolo $\oplus$ indica l'*addizione modulo* 2, e cioè l'*OR esclusivo*, per cui, se il numero di 1 della parola è dispari, si ottiene p=1; se è pari, p=0.
Ad esempio:

$$\begin{aligned} N_1 &= 100110 \qquad & p_{N_1} &= 1 \\ &\updownarrow \\ N_2 &= 101110 \qquad & p_{N_2} &= 0 \end{aligned}$$

Questo esempio ci mostra quanto detto in precedenza, circa la possibilità di *rilevare la presenza di un errore nel risultato di una operazione aritmetica*, effettuando un *confronto* del check digit del risultato attuale con quello del risultato corretto, ottenuto in base agli operandi dell'operazione stessa.
Un altro esempio può essere il seguente (b = 10):

risultato corretto: N = 12894 , p = 4

risultato attuale : N' = 12899 , p'= 9

Il confronto di p con p' rileva la presenza dell'errore nel risultato attuale.

Ma già da questo esempio è abbastanza chiaro quali siano i *limiti* del controllo di parità: cioè, *non vengono rilevati quegli errori tali che la somma delle cifre dell'errore stesso è congruente a 0 modulo b.*

Ad esempio:

N = 12894 , p = 4

N' = 12399 , p' = 4

In tale caso, non si ha la rilevazione dell'errore, confrontando p con p'.

Infatti, c'è stata una alterazione della 3° cifra da 8 a 3 e della 5° cifra da 4 a 9. Quindi:

$$p' = 1 \oplus 2 \oplus (8-5) \oplus 9 \oplus (4+5) =$$

$$= 1 \oplus 2 \oplus 8 \oplus 9 \oplus 4 \oplus (5-5) = p = 4$$

Cioè, la somma delle alterazioni delle cifre è $-5+5 = 0$, e dunque congruente a zero modulo 10.

Con riferimento alla *numerazione binaria*, si può allora dire che *non vengono rilevati quegli errori tali da lasciare inalterata la parità del numero.*

Ad esempio:

N = 101101 , p = 0

N' = 110101 , p' = 0

In altri termini *vengono rilevati* tutti quegli errori che hanno dato luogo al *cambiamento* di un *numero dispari di*

bits.

In particolare, può sempre essere rilevato un "ERRORE SINGOLO", che, per ora, definiamo un errore causato dall'alterazione di un solo bit.

Facciamo notare che, nella parte III dedicata ai codici residui, definiremo come errore singolo un errore E del tipo:

$$E = \pm 2^k \qquad 0 \leq k \leq n-1$$

In questa classe rientrano gli errori causati dal cambiamento di un solo bit, ma ne sono inclusi anche altri che hanno causato il cambiamento di più di un bit.

Vedremo, comunque, (ad esempio a proposito dell'addizione) che un errore molto comune è un *"ERRORE DOPPIO"*, il quale può essere causato dal *guasto di un solo gate* dell'unità aritmetica: cioè, fin da ora, vogliamo far notare come non ci sia diretta relazione tra la molteplicità di errore e la causa fisica dell'errore stesso.

1 - CALCOLO DEL CHECK DIGIT PER OPERAZIONI ARITMETICHE

A questo punto, intendiamo fornire *le relazioni che legano il check digit dal risultato delle varie operazioni aritmetiche e di traslazione ai check digits dei singoli operandi*.

Queste relazioni sono alla base della definizione di *una rete logica per il controllo delle operazioni eseguite dall'unità aritmetica di un calcolatore*.

A tale scopo, facciamo due *premesse* abbastanza ovvie, ma necessarie:

- per la definizione stessa di controllo di parità, è evidente la sua *invarianza nei confronti di operazioni di ro*

tazione;

- inoltre, tale *invarianza* si ha anche *nei confronti della posizione della virgola*, per cui supporremo di riferirci a *numeri interi*, senza con questo perdere in generalità.

a) *Addizione*(3)

Siano A e B gli addendi:

$$A = a_n\ a_{n-1} \cdots a_1$$

$$B = b_n\ b_{n-1} \cdots b_1$$

ed S la loro somma:

$$S = s_n\ s_{n-1} \cdots s_1$$

Come è noto, il generico digit della somma stessa è dato da:

$$s_i = a_i \oplus b_i \oplus c_{i-1}$$

ed il generico riporto:

$$c_i = a_i\ b_i + (a_i + b_i)\ c_{i-1}$$

Il check digit della somma è dato da:

$$\begin{aligned} p_s &= s_n \oplus s_{n-1} \oplus \cdots \oplus s_1 = \\ &= a_n \oplus b_n \oplus c_{n-1} \oplus a_{n-1} \oplus b_{n-1} \oplus c_{n-2} \oplus \cdots \oplus \\ &\oplus a_1 \oplus b_1 \oplus c_{in} \end{aligned}$$

Cioè:

(9) $$p_s = p_A \oplus p_B \oplus p_c$$

dove p_A, p_B, p_c sono i check digits degli operandi e del riporto complessivo:

$$p_A = a_n \oplus a_{n-1} \oplus \ldots \oplus a_1$$

$$p_B = b_n \oplus b_{n-1} \oplus \ldots \oplus b_1$$

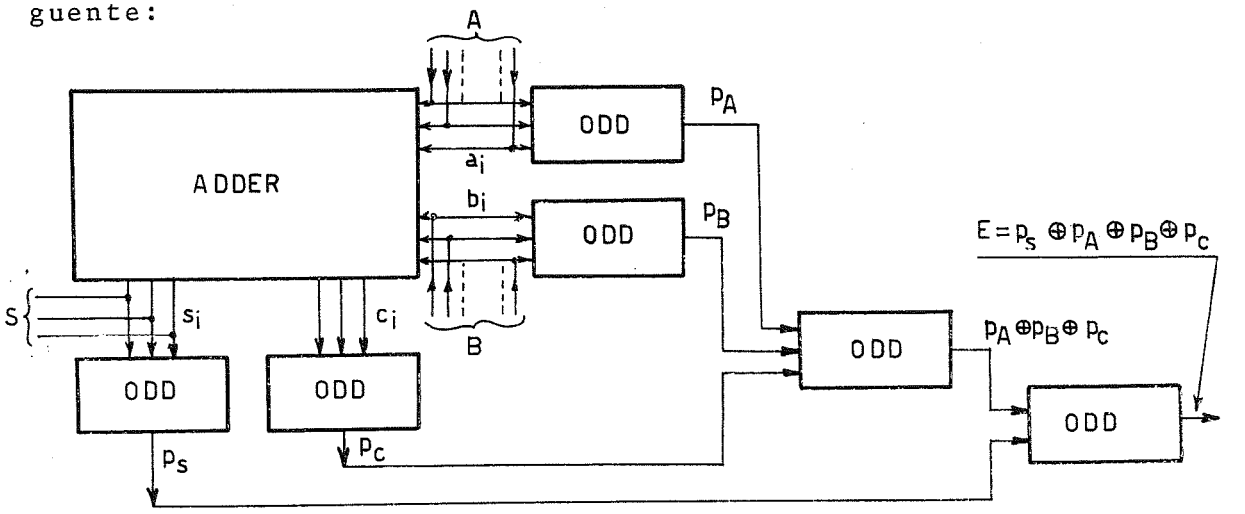
$$p_c = c_{n-1} \oplus c_{n-2} \oplus \ldots \oplus c_{in}$$

Ad esempio, nel caso di rappresentazione binaria:

C	0 0 0 1	$p_c = 1$
A	0 0 0 1	$p_A = 1$
B	0 1 0 1	$p_B = 0$
S	0 1 1 0	$p_s = 0$

$$p_A \oplus p_B \oplus p_c = 1 \oplus 0 \oplus 1 = 0 = p_s$$

Uno schema di principio di un *addizionatore dotato di rete per il controllo di parità può essere*, allora, il seguente:



I blocchi indicati con ODD non sono niente altro che *addizionatori modulo 2*, cioè *circuiti che eseguono l'operazione di OR esclusivo.*

In uscita si ottiene l'eventuale *errore* commesso:

$$E = p_s \oplus p_A \oplus p_B \oplus p_c$$

Cioè, si ha *indicazione di errore* nel caso che risulti E = 1. Ad esempio, nel caso precedente, se, per una ragione qualsiasi, la somma fosse risultata:

$$S = 0\ 1\ 1\ \underset{\downarrow}{\textcircled{1}}\ , \qquad p_s = 1$$

si sarebbe avuta una rilevazione dell'errore:

$$E = p_s \oplus p_A \oplus p_B \oplus p_c = 1 \oplus 1 \oplus 0 \oplus 1 = 1$$

Non si avrebbe rilevazione se risultassero errati due bits della somma, ma noi supponiamo di avere a che fare con un *singolo guasto* nell'unità di calcolo, per cui tale caso non ha interesse.

Notiamo subito, però, l'importanza delle considerazioni precedenti, circa i *limiti* del controllo di parità nella rilevazione degli errori: nel caso dell'addizione, cioè, *non sono rilevabili errori dovuti a riporti incorretti.*

Infatti, se il riporto di ordine i risulta *errato*, risulta senz'altro errato anche il digit di ordine i+1 della somma, per cui risulta *alterata la parità sia del riporto che della somma stessi, con che non si ha alterazione della espressione dell'errore*:

$$E = p_s \oplus p_A \oplus p_B \oplus p_c$$

Ad esempio, supponiamo che il riporto corretto e la somma corretta siano:

$$\begin{array}{lccccc} & & i+1 & i & i-1 & \\ C = & \ldots & 1 & 0 & 0 & \ldots \\ S = & \ldots & 1 & 0 & 1 & \ldots \end{array}$$

Abbiamo già visto che se risulta errato un bit della somma, siamo in presenza di un errore singolo, il quale è rilevabile con il controllo di parità.

Esaminiamo ora due casi in cui risulti errato un bit dal riporto:

$$\begin{array}{llcccccc} & & & & i+1 & i & i-1 & \\ 1°) & \text{Riporto errato} & C' & \ldots & 1 & 1 & 0 & \ldots \\ & & & & & \uparrow & \uparrow & \\ & \text{Riporto corretto} & C & \ldots & 1 & 0 & 0 & \\ & \text{Somma corretta} & S & \ldots & 1 & 0 & 1 & \ldots \\ & & & & & \downarrow & \downarrow & \\ & \text{Somma errata} & S' & \ldots & 1 & 1 & 1 & \ldots \end{array}$$

Risulta alterata la parità sia della somma che del riporto:

$$p_s' = \overline{p_s}$$

$$p_c' = \overline{p_c}$$

Siamo, dunque, in presenza di un errore non rilevabile:

$$E = p_A \oplus p_B \oplus p_c' \oplus p_s' = p_A \oplus p_B \oplus p_c \oplus p_s = 0$$

$$\begin{array}{llcccccc} & & & & i+1 & i & i-1 & \\ 2°) & & C' & \ldots & 1 & 1 & 1 & \ldots \\ & & & & & \uparrow & \uparrow & \uparrow \\ & & C & \ldots & 1 & 0 & 0 & \ldots \\ & & & & & & & \\ & & S & \ldots & 1 & 0 & 1 & \ldots \\ & & & & & \downarrow & \downarrow & \downarrow \\ & & S' & \ldots & 1 & 1 & 0 & \ldots \end{array}$$

Non risulta alterata la parità né della somma né del riporto:

$$p_s = p_s' \quad ; \quad p_c = p_c'$$

per cui siamo in presenza di un errore di ordine pari:

$$E = p_A \oplus p_B \oplus p_c' \oplus p_s' = p_A \oplus p_B \oplus p_c \oplus p_s = 0$$

e, dunque, non può aversi la rilevazione dell'errore.

In ultima analisi, *tutta la procedura è valida solo se i riporti sono corretti*. Per tale ragione un addizionatore controllato con la parità, dovrà essere dotato di *una rete di duplicazione per il calcolo dei riporti: ogni riporto, cioè, verrà ad essere confrontato con il suo duplicato*; se chiamiamo p_{cf} il risultato di questo confronto, si potrà calcolare l'errore come:

$$E = p_A \oplus p_B \oplus p_c \oplus p_s \oplus p_{cf}$$

Se si ha un errore su un riporto, $p_{cf} = 1$, e *l'errore viene rilevato*, se si suppone, naturalmente, di avere ancora a che fare *con un singolo guasto all'addizionatore, mentre la rete di duplicazione è esente da guasti*.

Notiamo, comunque, che tutto quanto detto in precedenza è valido *solo se sono disponibili, all'esterno dell'addizionatore, tutti i riporti*, ciò che normalmente non si ha con le recenti tecniche di costruzione a circuiti integrati.

Abbiamo voluto insistere volutamente sul controllo di parità di una addizione, ed in particolare sui problemi conseguenti a *riporti incorretti*, poichè quasi tutte le altre operazioni sono basate sulla addizione stessa, e, dunque, tali problemi torneranno quasi sempre di *attualità* nel seguito.

b) *Complementazione*

b_1) Complemento alla radice diminuita di 1.

Sia A il numero da complementare:

$$A = a_n \, a_{n-1} \cdots a_1$$

ed $\overline{A}$ il suo complemento alla radice diminuita di 1:

$$\overline{A} = \overline{a}_n \; \overline{a}_{n-1} \; \cdots \; \overline{a}_1$$

dove:

$$a_i \oplus \overline{a}_i = b - 1$$

Si può dimostrare facilmente la seguente relazione tra i check digits di A e di $\overline{A}$:

(10) $$\boxed{p_A \oplus p_{\overline{A}} \oplus n \text{ Mod } b = 0}$$

(essendo n il numero di bits della parola).

Nel caso di rappresentazione binaria, la formula precedente si semplifica osservando che:

$$|n|_2 = \begin{cases} 0 & n \text{ pari} \\ 1 & n \text{ dispari} \end{cases}$$

b_2) Complemento alla radice.
Sia A il numero da complementare ed A' il suo complemento alla radice.

La procedura che si segue per il controllo di parità è diversa a seconda del metodo che si usa per il calcolo di A'.

1°) Il più noto e comune è il seguente:

$$A' = \overline{A} + 1$$

In questo caso, si ottiene facilmente:

(11) $$\boxed{p_{A'} = 1 \oplus p_{\overline{A}} \oplus \sum_{i=2}^{n} c_i \text{ Mod } b}$$

Come si vede, si va incontro al *problema dei riporti incorretti*, che è stato esposto a proposito dell'addizione, con tutte le conseguenze dette allora.

2°) Un altro procedimento per il calcolo di A' può essere il seguente: si comincia ad esaminare i bits di A partendo da quello di ordine più basso; si esegue il complemento alla radice del primo bit che non è eguale a zero; i rimanenti si complementano alla radice diminuita di 1.

Se q è il numero di zeri che precedono il primo bit eguale ad 1, si può dimostrare la seguente relazione:

(12) $$\boxed{p_A \oplus p_A' \oplus (n-q) \text{ Mod } b = 1}$$

Pur essendo tale procedimento forse *meno comune* del 1°, *non si va incontro, stavolta, al problema dei riporti incorretti*, ciò che può costituire un notevole vantaggio.

c) *Traslazione*

Il calcolo del check digit di una parola che abbia subito una traslazione, in funzione del check digit della parola stessa prima di subire la traslazione, è piuttosto ovvio, in base alla definizione di controllo di parità, e tenendo conto dei vari tipi di shift, definiti al seguente modo:

Shift logico a destra:

$$0 \to a_n \,, \quad a_{i+1} \to a_i \qquad i = 1, \ldots, n-1$$

Shift logico a sinistra:

$$0 \to a_1 \,, \quad a_i \to a_{i+1} \qquad i = 1, \ldots, n-1$$

Shift aritmetico a destra:

$$a_s = a_n \quad a_{i+1} \to a_i \,, \quad a_s \to a_s \qquad i = 1, \ldots, n-1$$

Shift aritmetico a sinistra:

$$a_s = a_n$$

$$0 \rightarrow a_1 \ , \quad a_i \rightarrow a_{i+1} \ , \quad a_s \rightarrow a_s \ , \quad i = 1, \ldots, n-2$$

Negli shifts aritmetici, il bit del segno (a_s) non viene alterato. E' chiaro che nelle traslazioni logiche, poichè quando viene introdotto un nuovo bit, questo è eguale a zero, si tratterà di *testare il bit che è stato perduto: se questo era eguale a 0, il check digit non cambia; se era eguale ad 1, il check digit cambia.*

Nel caso di *shift logico a destra*:

$$A = a_n \ a_{n-1} \cdots a_2 \ a_1$$

$$A_{slr} = 0 \ a_n \ a_{n-1} \cdots a_2$$

(13)
$$\boxed{p_A = p_{Aslr} \oplus a_1}$$

Nel caso di *shift logico a sinistra*:

$$A = a_n \ a_{n-1} \cdots a_2 \ a_1$$

$$A_{sll} = a_{n-1} \ a_{n-2} \cdots a_1 \ 0$$

(14)
$$\boxed{p_A = p_{Asll} \oplus a_n}$$

Nel caso di *shift aritmetico a destra*:

$$A = a_s \ a_{n-1} \cdots a_2 \ a_1$$

$$A_{sar} = a_s \ a_s \ a_{n-1} \cdots a_2$$

(15)
$$\boxed{p_A = p_{Asar} \oplus a_1 \oplus a_s}$$

In questo caso, dunque, occorre *tener conto del bit del segno*.

Nel caso di *shift aritmetico a sinistra*:

$$A = a_s\ a_{n-1} \cdots a_2\ a_1$$

$$A_{sal} = a_s\ a_{n-2} \cdots a_1\ 0$$

(16) $$\boxed{P_A = P_{Asal} \oplus a_{n-1}}$$

d) *Moltiplicazione*

Sia A il moltiplicando, D il moltiplicatore e P il risultato:

$$P = A \times D$$

Essendo la moltiplicazione eseguita, in un calcolatore, con *una serie di somme e traslazioni a sinistra*, si può eseguire il controllo, controllando *ogni somma parziale*, osservando che, per il caso dei numeri rappresentati in modulo e segno, le traslazioni non alterano il numero di 1 delle somme parziali, si può scrivere:

(17) $$\boxed{P_P = (P_A \circledcirc P_D) \oplus \sum_i c_i \text{ Mod } b}$$

avendo indicato con il simbolo $\circledcirc$ la *moltiplicazione modulo b*, ed essendo i c_i *i riporti generati nelle varie somme parziali*. Siano, dunque, ancora in presenza del problema dei *riporti inesatti*, con le tutte le conseguenze ormai note.

e) *Divisione*

Sia X il dividendo, Y il divisore, Q il quoziente ed R il resto:

$$\frac{X}{Y} = Q + \frac{R}{Y}$$

Dalla relazione:

$$R = X - QY$$

si ottiene, per le proprietà fondamentali della congruenza:

(18) $$\boxed{p_R = p_X \oplus (p_Q \odot p_{\overline{Y}}) \oplus \sum_i c_i \text{ Mod } b}$$

Valgono gli stessi ragionamenti fatti per la moltiplicazione; in particolare, bisogna ancora fare attenzione al problema di eventuali *riporti incorretti*.

2 - *CONCLUSIONI*

In ultima analisi, possiamo dire che il controllo di una unità aritmetica con la parità porta a *circuiti relativamente semplici*, come si può vedere tentando di implementare le formule relative al check digit delle varie operazioni, esaminate nel paragrafo precedente.

Il grosso inconveniente del metodo resta quello della presenza di *eventuali riporti incorretti*, ogni volta che è presente una operazione di addizione (ed abbiamo visto che ciò accade piuttosto spesso). Attualmente, la migliore soluzione è quella, già accennata, di *duplicare la rete che calcola i riporti*. Inoltre, ricordiamo che occorre disporre di *addizionatori, per i quali si possa accedere ai riporti*.

III

CONTROLLO BASATO SULL'USO DI CODICI RESIDUI

Abbiamo visto come il concetto di parità, o, più precisamente, di "*parità digitale*", definito in termini di congruenze, possa essere applicato per ottenere la rilevazione di determinati errori su operazioni aritmetiche.

A questo punto, vogliamo introdurre un concetto analogo, il quale ci consentirà la rilevazione, ed anche la correzione, di una *classe più estesa* di errori di tipo aritmetico (2).

Si introduce, cioè, il concetto di quella che è da alcuni chiamata "*PARITA' NUMERICA*", così definita:
dato un numero N ed un modulo m, entrambi *interi*, si dice "*CIFRA DI PARITA' NUMERICA*", o, ancora, "*check digit*", la seguente quantità:

$$\boxed{p = |N|_m}$$

Ad esempio, se m = 3 ed N = 12895 (b=10), si ha:

$$p = |12895|_3 = 1$$

Rileviamo subito esplicitamente come il check digit, p, che nel caso di controllo di parità era una funzione delle singole cifre del numero N (precisamente, la loro somma modulo b), nel caso di parità numerica è funzione della grandezza del numero stesso.

1 - PROPRIETA' DELLA PARITA' NUMERICA

Consideriamo una certa operazione aritmetica, la quale debba dare come risultato *corretto* un certo numero C.

Supponiamo che tale risultato, per una ragione qualsiasi, sia affetto da un *errore*, E, così da avere come risultato *attuale*:

$$C \pm E$$

Consideriamo la cifra di parità numerica del risultato attuale:

$$|C \pm E|_m$$

e quella del risultato corretto, ottenibile in base agli operandi dell'operazione, nel modo che vedremo:

$$|C|_m$$

Confrontando queste cifre, si otterrà la *rilevazione* dell'errore se esse sono diverse:

$$|C \pm E|_m \neq |C|_m$$

Premesso questo, si vede subito scrivendo la cifra di parità numerica del risultato attuale nella forma seguente:

$$|C \pm E|_m = ||C|_m \pm |E|_m|_m$$

che: *condizione necessaria affinchè l'errore possa effettivamente essere rilevato è che non sia*:

$$|E|_m = 0$$

Quindi, il passo successivo della nostra discussione sarà rivolto ad esporre i criteri di *scelta del modulo m*, in rela-

zione al *tipo di errore* che intendiamo rilevare.

2 - SCELTA DEL MODULO

Notiamo subito, da come è stata definita la parità numerica, che deve essere:

$$\boxed{m \neq b}$$

Infatti, in tal caso, si avrebbe:

$$p = |N|_b = \left| \sum_{i=1}^{n} a_b \, b^{i-1} \right|_b = |a_1|_b = a_1$$

Cioè, la cifra di parità numerica sarebbe sempre eguale al digit meno significativo del numero, e non funzione dell'intero numero, come vogliamo che sia.

O meglio, poichè un errore che interessi una cifra diversa da quella meno significativa si può sempre scrivere:

$$E = \pm \; x.b^{i} \quad , \quad \text{con} \quad i \neq 0 \quad , \quad 0 < x < b$$

si avrebbe sempre

$$|E|_b = 0$$

e, quindi, per quanto detto in precedenza, non si avrebbe mai la rilevazione dell'errore stesso.

E' ovvio, inoltre, che non potrà mai nemmeno essere:

$$m = 1$$

dal momento che ogni numero è congruente a 0 Mod 1.

A questo punto, esaminati i valori da scartare per la scelta di m, è opportuno dare una definizione della moltepli

cità di errore, più in generale di quella data nella parte II, allo scopo di ben definire le classi di errori che più ci interessano, e, conseguentemente, procedere alla scelta del modulo opportuno per ognuna di esse.

3 - MOLTEPLICITA' DI ERRORE(4)

Sia N il risultato esatto di una certa operazione ed N' quello affetto da errore.

Tale errore si può sempre scrivere in forma polinomiale:

$$E = N' - N = \sum_{i=1}^{n} a_i \cdot b^{i-1} \qquad (-b < a_i < +b)$$

Si definisce molteplicità di errore *il numero minimo di termini non nulli necessari per scrivere E nella precedente forma polinomiale.*

Premesso che, d'ora in poi, ci riferiremo al solo caso di rappresentazione binaria, diamo i seguenti esempi:

1°) N = 1 0 0 0 0 → 16
N' = 0 1 1 1 0 → 14

$E = N' - N = -2 = -2^1$

Si tratta di un *errore singolo.*

2°) N = 1 0 0 0 0 → 16
N' = 1 1 1 1 0 → 30

$$E = N' - N = 14 = +1 \cdot 2^4 - 1.2^2$$

Si tratta di un *errore doppio*.

Da questi esempi, è emerso il fatto importante che *non c'è relazione diretta tra la molteplicità di errore ed il numero di cifre affette da errore* (come invece accadeva per il controllo di parità digitale).

In particolare, *un errore su una sola cifra del risultato è sempre un errore singolo; ma un errore singolo può risultare da valori errati di più cifre consecutive*.

Per riferirci ad un caso ampiamente esaminato nella parte II, vediamo quali tipi di errore si possono avere nell'operazione di addizione, in seguito ad un singolo guasto avvenuto nell'addizione.

Esaminiamo tutte le possibili conseguenze che può avere un tale guasto, riferendoci al seguente esempio:

		i+2	i+1	i	+-1
C	(1)	0	0	0	0
A		1	0	1	0
B		1	1	0	0
S		0	1	1	0

1°) Errore su un bit di somma:

S 0 1 1 0

S' 0 1 1 1 ↓

$$E = S' - S = 2^{i-1}$$

Siamo in presenza di un *errore singolo*.

2°) Errore su un bit del riporto:

a) caso in cui non ci sia propagazione dell'errore

$$
\begin{array}{lcccccc}
C' & (1) & 0 & 0 & 0 & 1 & \\
 & & & & & \uparrow & \\
C & (1) & 0 & 0 & 0 & 0 & \\
\hline
 & & 1 & 0 & 1 & 0 & \\
 & & 1 & 1 & 0 & 0 & \\
\hline
S & & 0 & 1 & 1 & 0 & \\
 & & & & & \downarrow & \\
S' & & 0 & 1 & 1 & 1 &
\end{array}
$$

$$E = S' - S = 2^{i-1}$$

Siamo in presenza di un *errore singolo*.

b) caso in cui non ci sia propagazione dell'errore:

$$
\begin{array}{lcccccc}
C' & (1) & 1 & 1 & 1 & 0 & \\
 & & \uparrow & \uparrow & \uparrow & & \\
C & (1) & 0 & 0 & 0 & 0 & \\
\hline
 & & 1 & 0 & 1 & 0 & \\
 & & 1 & 1 & 0 & 0 & \\
\hline
S & & 0 & 1 & 1 & 0 & \\
 & & \downarrow & \downarrow & \downarrow & & \\
S' & & 1 & 0 & 0 & 0 &
\end{array}
$$

$$E = S' - S = 2^{i+2} - 2^{i+1} - 2^{i} = 2^{i}$$

Siamo ancora in presenza di un *errore singolo*.

3°) Errore sia su un bit del riporto che su un bit di somma:

a) senza propagazione dell'errore:

$$
\begin{array}{lcccccc}
C' & (1) & 0 & 0 & 0 & 1 & \\
 & & & & & \uparrow & \\
C & (1) & 0 & 0 & 0 & 0 & \\
\hline
 & & 1 & 0 & 1 & 0 & \\
 & & 1 & 1 & 0 & 0 & \\
\hline
S & & 0 & 1 & 1 & 0 & \left\{\begin{array}{l} 0 \rightarrow 1 \\ 1 \rightarrow 0 \end{array}\right. \\
 & & & & & \downarrow & \\
S' & & 0 & 1 & 1 & 1 & \\
 & & i+2 & i+1 & i & i-1 & i-2
\end{array}
$$

Abbiamo esaminato, cioè, il caso in cui un guasto nell'elemento (i-2)esimo dell'addizionatore abbia portato ad un errore sia sulla somma che sul riporto generati da tale elemento.

Nel caso più favorevole avremo:

$$E = S' - S = 2^{i-1} - 2^{i-2} = 2^{i-2}$$

per cui siamo in presenza di un *errore singolo*.

Nel caso più sfavorevole, avremo:

$$E = S' - S = 2^{i-1} + 2^{i-2}$$

Siamo in presenza di un particolare *errore doppio*, comunemente detto "*BURST DOPPIO*".

b) Con propagazione dell'errore:

C'	(1)	1	1	1	0
		↑	↑	↑	
C	(1)	0	0	0	0
		1	0	1	0
		1	1	0	0
S		0	1	1	0
		↓	↓	↓	↓
S'		1	0	0	1

Cioè, abbiamo supposto che un guasto nell'elemento (i-1)esimo dell'addizionatore abbia dato luogo ad un errore sia sulla somma che sul riporto generati da tale stadio.

Si ha:

$$E = S' - S = 2^{i+2} - 2^{i-1} - 2^{i} + 2^{i-1} = 2^{i} + 2^{i-1}$$

Siamo ancora in presenza di un BURST DOPPIO.

In conclusione, abbiamo visto che un errore su un bit di somma o su un bit del riporto dà sempre luogo ad un erro

re singolo (a differenza di quanto avevamo concluso adottando la definizione di molteplicità di errore, data per il controllo di parità digitale, con la quale un errore di un solo bit del riporto può generare un errore multiplo sulla somma).

Nel caso più sfavorevole possibile, in cui risultino errati sia un bit di somma che un bit del riporto, si può avere un burst doppio.

In ogni caso, con la scelta opportuna del modulo, come vedremo tra poco, il controllo di parità numerica è in grado di rilevare un qualsiasi errore avvenuto nell'operazione di addizione, e, quindi, essendo l'addizione chiamata in causa in quasi tutte le operazioni, un qualsiasi errore avvenuto nell'unità di calcolo, in seguito ad un *singolo guasto* in essa avvenuto.

4 - RILEVAZIONE DI ERRORI SINGOLI

Vediamo quale modulo è opportuno scegliere, per avere la rilevazione di un qualsiasi errore singolo, cioè del tipo:

$$E = \pm 1 \cdot 2^{i}$$

Ovviamente, si cerca sempre *il valore più basso possibile del modulo stesso*, in modo da codificare p con il minimo numero di bits.

Essendo già stati esclusi $m = 1$ ed $m = 2$, possiamo verificare che $m = 3$ è sufficiente allo scopo proposto.
Deve essere: $|E|_3 = |\pm 1 \cdot 2^{i}|_3 \neq 0$ per ogni i
Per la relazione 9)

$$|\pm 1 \cdot 2^{i}|_3 = \left\{ \begin{matrix} |\pm 1|_3 , & i \text{ pari} \\ |\overline{+}1|_3 , & i \text{ dispari} \end{matrix} \right\} \quad \begin{matrix} |+1|_3 = 1 \\ |-1|_3 = 2 \end{matrix}$$

Dunque, il controllo di parità numerica, con $m = 3$, permette di *rilevare tutti gli errori singoli*, compresi quelli che sfuggivano al controllo di parità digitale.

5 - *RILEVAZIONE DI UN BURST DOPPIO*

Abbiamo già detto che un burst doppio è del tipo:

$$E = \pm 1 \cdot 2^{i+1} \pm 1 \cdot 2^{i} = \begin{cases} \pm 3 \cdot 2^{i} \\ \\ \pm 1 \cdot 2^{i} \end{cases}$$

Scriviamo tali espressioni *in forma quaternaria*, anzichè binaria; cioè:

$$E = \begin{cases} \pm 3 \cdot 4^{i/2} \\ \\ \pm 1 \cdot 4^{i/2} \end{cases} \qquad (i \text{ pari})$$

ciò che è sempre possibile.

Da questa scrittura, si vede che, per avere la rilevazione di ogni errore di questo tipo, il modulo non può essere 3,4 (e nemmeno, come sempre, 1,2).

La scrittura stessa ci indica anche che il minimo modulo possibile è:

$$\boxed{m = 5}$$

Infatti, ancora in base alla relazione 9) si ha:

$$\left| \pm 3 \cdot 4^{i/2} \right|_5 = \left\{ \begin{array}{l} \left| \pm 3 \right|_5 \quad i/2 \text{ pari} \\ \\ \left| \mp 3 \right|_5 \quad i/2 \text{ dispari} \end{array} \right\} \quad \begin{array}{l} \left| +3 \right|_5 = 3 \\ \\ \left| -3 \right|_5 = 2 \end{array}$$

$$|\pm 1 \cdot 4^{i/2}|_5 = \left\{ \begin{array}{l} |\pm 1|_5 \quad i/2 \text{ pari} \\ |\mp 1|_5 \quad i/2 \text{ dispari} \end{array} \right\} \quad \begin{array}{l} |+1|_5 = 1 \\ |-1|_5 = 4 \end{array}$$

Notiamo anche il modulo

$$\boxed{m = 7}$$

permette ancora la rilevazione di un burst doppio di errore. Anzi, vedremo in seguito che è opportuno calcolare il check digit per la rilevazione di tale tipo di errore, utilizzando m = 7, poichè in tal modo si hanno delle semplificazioni realizzative.

6 - *POSSIBILITA' DI RILEVARE ERRORI IN OPERAZIONI ARITMETICHE MEDIANTE IL CONTROLLO DI PARITA' NUMERICA.*

A questo punto, vogliamo vedere, più approfonditamente, per quali ragioni il controllo di parità numerica è particolarmente adatto per la rilevazione di errori in operazioni aritmetiche.

La ragione fondamentale ci è mostrata dalle stesse proprietà fondamentali (esposte nella parte I), che ci forniscono il modo di calcolare il residuo del risultato delle operazioni di addizione, sottrazione e moltiplicazione; precisamente:

a) *Addizione*:

$$\left| A + B \right|_m = \left| |A|_m + |B|_m \right|_m$$

b) *Sottrazione*:

$$\left| A - B \right|_m = \left| |A|_m - |B|_m \right|_m$$

c) *Moltiplicazione*:

$$\left|A \cdot B\right|_m = \left|\left|A\right|_m \cdot \left|B\right|_m\right|_m$$

Rileviamo esplicitamente le possibilità offerteci da queste relazioni, allo scopo di controllare le corrispondenti operazioni mediante la parità numerica.

Intanto, come si vede, *il check digit del risultato corretto si può calcolare in funzione dei soli check digits degli operandi.*

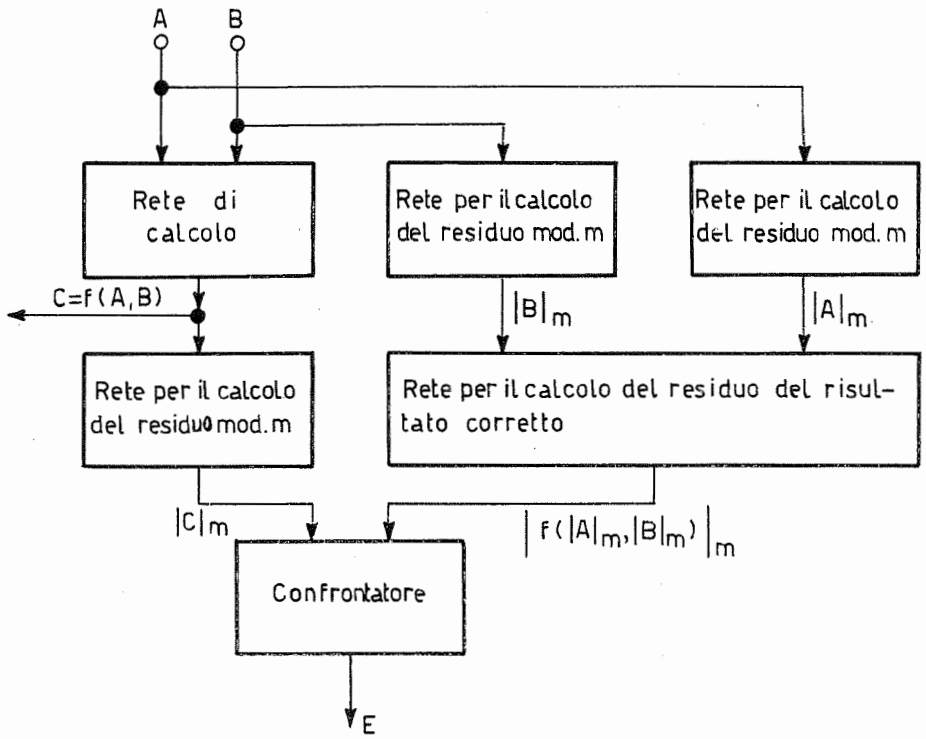
In altri termini, tali relazioni sono *valide senza particolare riguardo al metodo con cui le operazioni stesse vengono eseguite*, cosa che, invece, accadeva con il controllo di parità digitale.

In particolare, una delle caratteristiche più importanti è quella secondo cui *non si deve considerare i riporti avvenuti nell'operazione di addizione.* Ci rialacciamo, quindi, al discorso già fatto in precedenza: se, a causa di un guasto dell'unità di calcolo, risulta errato un riporto, questo può dar luogo ad errore singolo oppure ad un burst doppio nella somma, ma, se abbiamo cura di scegliere un modulo opportuno, essi possono sempre essere rilevati, come si è visto nei par. 4 e 5.

Inoltre, *non è affatto necessario disporre dei riporti stessi all'esterno dell'addizionatore*, ciò che costituiva un altro problema, non indifferente, del controllo di parità digitale.

Detto questo, il metodo generale per la rilevazione di un errore commesso dall'unità aritmetica è quello di *confrontare la cifra di parità numerica del risultato attuale (come esce dall'unità stessa) con quella del risultato corretto, ottenuta in funzione delle singole cifre di parità degli operandi.*

Questo metodo può essere esplicitato nel seguente *schema a blocchi di un'unità di calcolo controllata mediante la parità numerica:*

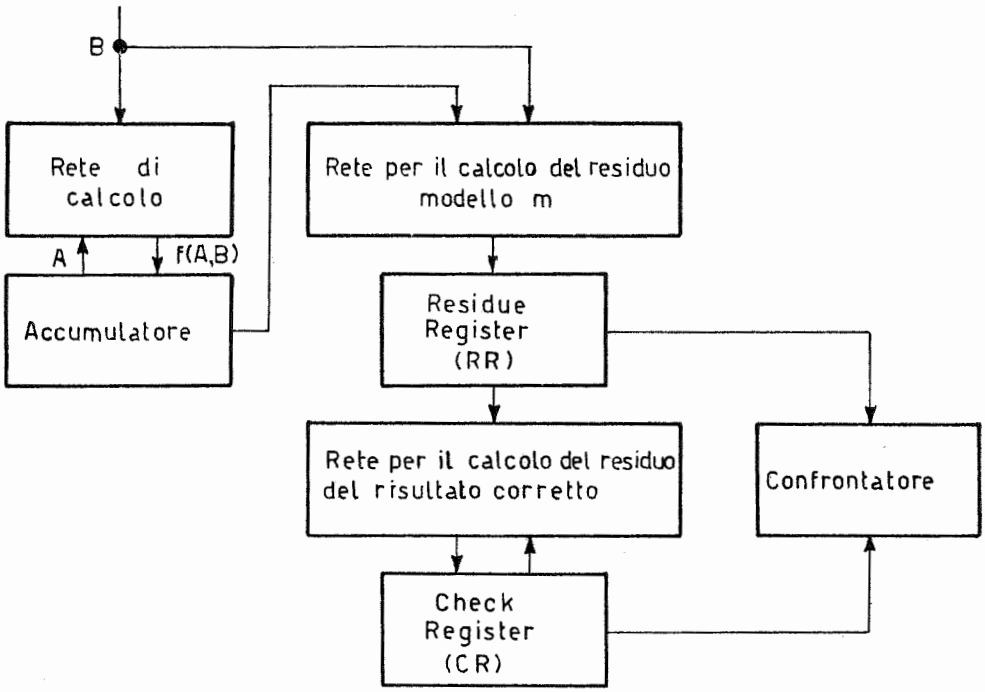


Vedremo tra poco come si realizza una Rete per il calcolo, del residuo Mod m, nei casi di $m = 3$ ed $m = 7$.

Nei casi pratici, inoltre, *si può fare a meno di usare tre distinti circuiti per il calcolo del residuo Mod m, presentando a tempi opportuni operandi e risultato ad uno stesso circuito.*

Uno schema che sfrutta tale principio può essere il seguente (5), con riferimento ad un calcolatore ad un accumulatore, per il quale le operazioni avvengono nel seguente modo: un operando, A, è contenuto nell'accumulatore e l'altro, B, in memoria; il risultato dell'operazione viene immagazzinato nell'accumulatore stesso:

$$f\ (A,\ B) \rightarrow A$$



Per riferirci ad un caso pratico, riferiamoci all'operazione di addizione:

$$A + B \rightarrow A$$

In tal caso, nella Rete per il Calcolo del residuo del risultato corretto ha interesse solo un addizionatore modulo m.

Prima che inizi l'operazione, nell'accumulatore si trova A, per cui in CR si trova $|A|_m$.

Quando inizia l'operazione, l'operando B, contenuto in memoria, viene portato sia all'unità di calcolo che alla Rete per il calcolo del Residuo mod m: dunque, in RR si trova $|B|_m$, per cui avviene l'operazione:

$$\left| |A|_m + |B|_m \right|_m$$

nell'addizionatore modulo m, tra il contenuto di CR e quello di RR. Il risultato è memorizzato in CR.

Intanto è stata eseguita l'addizione di A e B, per cui nell'accumulatore si trova (A+B); di tale quantità, ne viene calcolato il residuo modulo m: $|A+B|_m$, il quale viene memorizzato in RR.

A questo punto avviene il confronto tra il contenuto di RR (residuo Mod m del risultato attuale) con quello di CR (residuo Mod m del risultato corretto): nel caso che essi siano diversi, si ha un segnale di rilevazione di errore.

Vogliamo far notare come la complessità logica della parte controllo sia essenzialmente solo quella apportata dalla "Rete per il Calcolo del residuo del risultato corretto", la quale si può considerare una "duplicazione modulo m" dell'unità di calcolo, e, dunque, assai meno complessa di quest'ultima.

Gli altri circuiti presenti sono, infatti, di complessità assai poco rilevante, come vedremo tra poco quando daremo gli schemi delle Reti per il calcolo del residuo mod 3 e mod 7.

Inoltre, i registri RR e CR sono di 2 o 3 bits, ed il confrontatore dei contenuti di tali registri è anch'esso assai semplice.

7 - CALCOLO DEL RESIDUO MODULO 3

Abbiamo visto come, per la *rilevazione di errori singoli*, sia sufficiente usare il modulo:

$$m = 3$$

Vediamo ora come si calcola il residuo mod 3 di un numero N, rappresentato in binario, allo scopo di *ricavare una funzione che possa essere esplicitata in una rete logica, da inserire nello schema del par. 6.*
Sia, dunque

$$N = a_n\, 2^{n-1} + a_{n-1}\, 2^{n-2} + \ldots + a_2\, 2^1 + a_1\, 2^0$$

Allo scopo di usare una *rappresentazione quaternaria*, che ci sarà particolarmente utile, serviamo:

$$N = (a_1+2a_2)\cdot 2^0 + (a_3+2a_4)\cdot 2^2 + \ldots + (a_{n-1}+2a_n)\cdot 2^{n-2} =$$

$$= (a_1+2a_2)\cdot 4^0 + (a_3+2a_4)\cdot 4^1 + \ldots + (a_{n-1}+2a_n)\cdot 4^{\frac{n-2}{2}}$$

A questo punto, si ha:

$$|N|_3 = \left|\left|(a_1+2a_2)\cdot 4^0\right|_3 + \left|(a_3+2a_4)\cdot 4^1\right|_3 + \ldots + \left|(a_{n-1}+2a_n)\cdot 4^{\frac{n-2}{2}}\right|_3\right|_3$$

Ricordando la relazione 8) si ha:

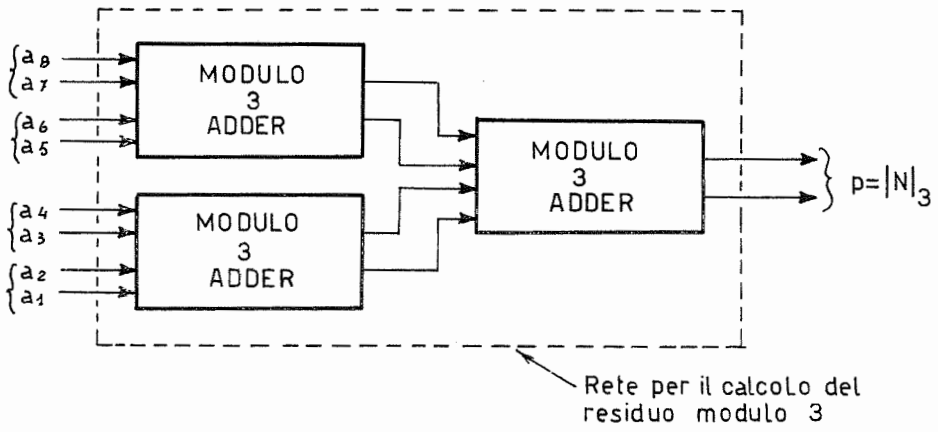
$$\left|(a_i + 2a_{i+1})\ 4^{\frac{i-1}{2}}\right|_3 = \left|a_i + 2a_{i+1}\right|_3 = \left|a_{i+1}\ a_i\right|_3$$

dove la *coppia* $a_{i+1}\ a_i$ rappresenta *una cifra quaternaria codificata in binario.*
Dunque:

$$(19)\quad \boxed{|N|_3 = \Big|\,|a_n a_{n-1}|_3 + \ldots + |a_6 a_5|_3 + |a_4 a_3|_3 + |a_2 a_1|_3\,\Big|_3}$$

Questa relazione ci fornisce un semplice metodo per realizzare il calcolo di $|N|_3$: si tratta di *suddividere la parola in coppie di bits* (partendo dal meno significativo) e *sommarle in un addizionatore modulo 3, successivamente*, sino ad ottenere *una sola coppia di bits*, la quale costituisce il desiderato $|N|_3$.

Ad esempio, nel caso di un numero di 8 bits, si ha il seguente schema:



8 - CALCOLO DEL RESIDUO MODULO 7

Abbiamo visto come, per la *rilevazione di un burst doppio di errore*, sia sufficiente usare il modulo:

$$m = 5$$

In tal modo, essendo:

$$0 \leq |N|_5 \leq 4$$

occorrono *3 bits per codificare la cifra di parità numerica*. Notiamo che *gli stessi 3 bits bastano, allo scopo sopra detto*, per codificare il residuo del numero *modulo 7*, essendo:

$$0 \leq |N|_7 \leq 6$$

Come abbiamo già accennato, il modulo:

$$m = 7$$

permette ancora *la rilevazione di un burst doppio di errore*, ed, inoltre, permette *una più facile realizzazione della rete per il calcolo del residuo di un dato numero*, poichè si può fare ancora uso della relazione 8), (analogamente al caso di $m = 3$), ciò che non è possibile per $m = 5$.

A tale scopo, scriviamo il numero N *in ottale*, al seguente modo:

$$N = a_1 2^0 + a_2 2^1 + a_3 2^2 + a_4 2^3 + a_5 2^4 + a_6 2^5 + \ldots +$$

$$+ a_{n-2} 2^{n-3} + a_{n-1} 2^{n-2} + a_n 2^{n-1} = (a_1+2a_2+4a_3) 2^0 +$$

$$+ (a_4+2a_5+4a_6) 2^3 + \ldots + (a_{n-2}+2a_{n-1}+4a_n) 2^{n-3} =$$

$$= (a_1+2a_2+4a_3) 8^0 + (a_4+2a_5+4a_6) 8^1 + \ldots +$$

$$+ (a_{n-2}+2a_{n-1}+4a_n) 8^{\frac{n-3}{3}}$$

A questo punto:

$$|N|_7 = \left| |(a_1+2a_2+4a_3) 8^0|_7 + \ldots + |(a_{n-2}+2a_{n-1}+4a_n) 8^{\frac{n-3}{3}}|_7 \right|_7$$

Dalla relazione 8) si ha:

$$\left| (a_i + 2a_{i+1} + 4a_{i+2})\ 8^{\frac{i-1}{3}} \right|_7 = \left| a_i + 2a_{i+1} + 4a_{i+2} \right|_7 =$$

$$= \left| a_{i+2}\ a_{i+1}\ a_i \right|_7 .$$

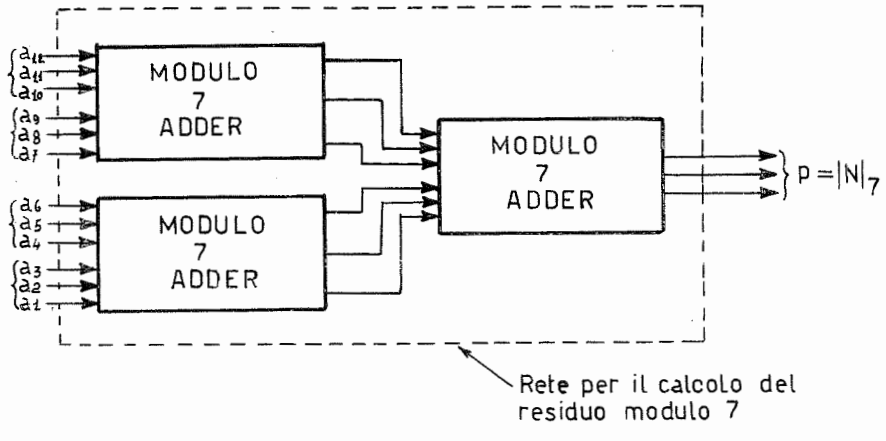
dove la terna $a_{i+2}\ a_{i+1}\ a_i$ rappresenta una cifra ottale codificata in binario.

Dunque:

$$(20) \qquad \boxed{|N|_7 = \left| \left| a_n a_{n-1} a_{n-2} \right|_7 + \ldots + \left| a_6 a_5 a_4 \right|_7 + \left| a_3 a_2 a_1 \right|_7 \right|_7}$$

Questa relazione ci permette di realizzare il calcolo di $|N|_7$ mediante una *serie di addizioni modulo 7 su gruppi di 3 bits della parola* (ottenuti partendo dal meno significativo), *sino ad ottenere un solo gruppo di 3 bits, che costituisce il desiderato* $|N|_7$.

Ad esempio, nel caso di un numero di 12 bits, si ha il seguente schema:



NOTA:

Il metodo, che stiamo descrivendo, di operare il controllo di operazioni aritmetiche è del tutto analogo a quello, a tutti noto, della "*PROVA DEL 9*" sulle operazioni di addizione, sottrazione e moltiplicazione.

In questo caso, cioè, la base della numerazione è b=10, ed il modulo usato per il controllo di parità numerica è m = 9.

Il calcolo del residuo si esegue applicando la relazione 8):

$$\left|ab^n\right|_{b-1} = a$$

Da notare, però, che, nei riguardi della molteplicità di errore, tale metodo non garantisce la rilevazione di qualunque errore singolo. Ad esempio, non viene rilevato lo scambio di una cifra decimale da 0 a 9 o viceversa.

Si potrebbe ottenere la rilevazione sicura di ogni errore singolo, usando un modulo:

$$m = 11$$

(cioè fare la "prova dell'11"!); in tal caso, però, in base alla relazione 9):

$$\left|a\ b^n\right|_{b+1} = \begin{cases} a & n \text{ pari} \\ \left|-a\right|_{b+1} = b+1-a & n \text{ dispari} \end{cases}$$

occorrebbe testare, di volta in volta, se stiamo esaminando una cifra pari o dispari.

9 - *CALCOLO DEL CHECK DIGIT PER OPERAZIONI ARITMETICHE*

A questo punto, riportiamo le formule per il calcolo del check digit del risultato delle operazioni aritmetiche e di altre operazioni, comunemente usate nelle unità di calcolo, come le traslazioni e le rotazioni, in funzione dei check digits degli operandi.

Queste formule definiscono completamente quella che, nello schema del par. 6, era stata chiamata "*Rete per il calcolo del residuo del risultato corretto*".

a) *Addizione*

Se A e B sono gli addendi, il check digit della somma è subito calcolabile come:

$$(21) \qquad \left| |A + B|_m = \left| |A|_m + |B|_m \right|_m \right|$$

b) *Complementazione*

b_1) Complemento alla radice diminuita di 1.

Dato un numero A, il suo complemento alla radice dimunuita di 1, $\overline{A}$, è legato ad esso dalla relazione:

$$\overline{A} = -A + (2^n - 1)$$

essendo n il numero di bits del numero stesso.

Si ottiene, dunque, la relazione:

$$(22) \qquad \left| |\overline{A}|_m = \left| -|A|_m + |2^n - 1|_m \right|_m \right|$$

Nel caso di m = 3, per la relazione 9) si ha:

$$|2^n-1|_3 = \begin{cases} 0 \text{ se n pari} \\ \\ 1 \text{ se n dispari} \end{cases} \quad ; \quad |\overline{A}|_3 = \begin{cases} |-A|_3 & \text{n pari} \\ \\ |-A+1|_3 & \text{n dispari} \end{cases}$$

b_2) Complemento alla radice

Dato un numero A di n bits, il suo complemento alla radice, A', è legato ad esso dalla relazione:

$$A' = 2^n - A$$

Dunque:

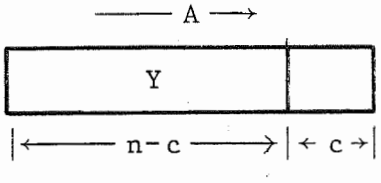
(23) $$\boxed{|A'|_m = \left|-|A|_m + |2^n|_m\right|_m}$$

Nel caso di $m = 3$, per la relazione 9) si ha:

$$|A'|_3 = \begin{cases} \text{n pari} & |1-A|_3 \\ \\ \text{n dispari} & |2-A|_3 \end{cases}$$

c) *Traslazione*

c_1) Traslazione logica a destra di c posti



Con le notazioni di figura, il risultato dell'operazione è:

$$Y \cdot 2^{-c}$$

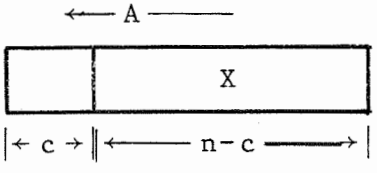
Dunque:

(24) $$\boxed{|Y\cdot 2^{-c}|_m = \left||Y|_m \cdot |2^{-c}|_m\right|_m}$$

Nel caso di $m = 3$, per la relazione 9) si ha:

$$|Y\cdot 2^{-c}|_3 = \begin{cases} |Y|_3 & c \text{ pari} \\ |\overline{Y}|_3 & c \text{ dispari} \end{cases}$$

c_2) Traslazione logica a sinistra di c posti



Con le notazioni di figura, il risultato dell'operazione è:

$$X \cdot 2^c$$

Dunque:

(25) $$\boxed{|X\cdot 2^c|_m = \left||X|_m \cdot |2^c|_m\right|_m}$$

Nel caso di $m = 3$, per la 9) si ha:

$$|X\cdot 2^c|_3 = \begin{cases} |X|_3 & c \text{ pari} \\ |\overline{X}|_3 & c \text{ dispari} \end{cases}$$

Nei casi di traslazione aritmetica, a destra ed a sinistra, si ottengono formule del tutto analoghe.

d) *Rotazione*

Ci limitiamo, per semplicità, al caso di:

m = 3

Nel caso di n pari (dove n è il numero di bits), si possono facilmente dimostrare le seguenti relazioni:

d_1) Rotazione destra o sinistra di c bits.

Il risultato dell'operazione è rispettivamente: $A \cdot 2^c$ oppure $A \cdot 2^{n-c}$.

Per i check digits valgono le relazioni:

(26)
$$\left.\begin{array}{l} |A\cdot 2^c|_3 \\ |A\cdot 2^{n-c}|_3 \end{array}\right\} = \begin{cases} |A|_3 & c \text{ pari} \\ |\overline{A}|_3 & c \text{ dispari} \end{cases}$$

Nel caso di *n dispari*, si ha:

d_2) Rotazione a sinistra di c bits.

Il risultato dell'operazione è: $\left|A\cdot 2^c\right|_{2^n-1}$.

Per il check digit si ha:

(27)
$$\left|\left|A\cdot 2^c\right|_{2^n-1}\right|_3 = \begin{cases} \left||X|_3 + |A|_3\right|_3 & c \text{ dispari} \\ \overline{\left||X|_3 + |A|_3\right|_3} & c \text{ pari} \end{cases}$$

dove X è lo stesso indicato nella figura per la traslazione logica a sinistra di c posti.

d_3) Rotazione a destra di c bits

Il risultato dell'operazione è: $\left|A\cdot 2^{-c}\right|_{2^n-1}$.

Per il check digit si ha:

$$(28)\qquad \left|\left|A\cdot 2^{-c}\right|_{2^n-1}\right|_3 = \begin{cases} \left|\left|Y\right|_3 + \left|A\right|_3\right|_3 & c \text{ dispari} \\ \overline{\left|\left|Y\right|_3 + \left|A\right|_3\right|_3} & c \text{ pari} \end{cases}$$

dove Y è lo stesso indicato nella figura per la traslazione logica a destra di c posti.

e) *Moltiplicazione*

Se A e B sono gli operandi, il check digit del risultato è subito dato da:

$$(29)\qquad \left|A\cdot B\right|_m = \left|\left|A\right|_m \cdot \left|B\right|_m\right|_m$$

f) *Divisione*

Se X è il dividendo, Y il divisore, Q il quoziente ed R il resto, essi sono legati dalla relazione:

$$X = QY + R$$

da cui si ottiene subito:

$$(30)\qquad \left|X\right|_m = \left|\left|Q\right|_m \cdot \left|Y\right|_m + \left|R\right|_m\right|_m$$

10 - CONCLUSIONI SULLA RILEVAZIONE DI ERRORI MEDIANTE IL CONTROLLO DI PARITA' NUMERICA.

Per concludere, ricapitoliamo i vantaggi del controllo basato sull'uso di codici residui, per la rilevazione di errori aritmetici, nei confronti del controllo di parità (di-

gitale).

a) Purchè si usi *un modulo opportuno*, può essere rilevato un errore, di molteplicità qualunque.

b) Il check digit del risultato di una qualsiasi operazione aritmetica si calcola *indipendentemente dal particolare metodo adottato per eseguire l'operazione stessa* : esso dipende solo dai *check digits degli operandi*.

c) Usando come modulo m = 3 , si rilevano *tutti gli errori singoli*, compresi quelli causati da un riporto incorretto in operazioni di addizione.

d) Se un errore su un bit del riporto ed un simultaneo errore nello stesso bit di somma danno luogo ad un *burst doppio di errore*, esso può essere rilevato usando un modulo m = 5 oppure m = 7.

e) Le formule che danno i check digits dei risultati delle varie operazioni sono generalmente *più semplici*, per cui *la realizzazione pratica del metodo non dà luogo ad una grande complessità logica*.

11 - CENNO SULLA CORREZIONE DI ERRORI SINGOLI (4)

Il problema della correzione di errori aritmetici è, in generale, molto *più complesso* di quello della loro rilevazione, sia dal punto di vista della sua *risoluzione*, che da quello del *costo* delle reti che manipolano i check digit, a causa del *gran numero di bits necessari per codificare la cifra di parità numerica*, la qual cosa è *diretta conseguenza dei valori piuttosto elevati che deve assumere il modulo*.

Queste considerazioni, già valide per la correzione di *errori singoli*, lo sono, a maggior ragione, se si richiede

la correzione di *errori multipli*.

Esaminiamo brevemente il caso di *ERRORI SINGOLI*, per numeri binari.

Dato un numero di *n bits*, possono aver luogo *2n possibili errori singoli, diversi tra loro*, dal momento che l'errore su ogni posizione può essere positivo o negativo.

Il modulo m da scegliere per la correzione di un errore singolo deve essere tale che:

$$|E|_m = |\pm 1 \cdot 2^i|_m$$

sia *diverso per ogni diverso errore singolo* (cioè per ogni diverso i = 0, 1, ..., n), *incluso il caso di E = 0*.

Ne deriva che condizione necessaria affinchè sia possibile correggere un errore singolo in un numero di n cifre binarie, mediante il modulo m, è che sia:

$$\boxed{m \geq 2n + 1}$$

In *mancanza di una condizione sufficiente*, il problema è quello di determinare euristicamente il *minimo valore di m, che soddisfi la relazione precedente*, che sia reprocicamente primo con la base della numerazione e tale che, *ad ogni possibile errore singolo E* (compreso il caso di assenza di errore: E = 0), *sia associato, univocamente, un diverso valore di*:

$$|E|_m = \left| |N'|_m - |N|_m \right|_m$$

detto ancora "*check digit*", essendo N il risultato corretto ed N' quello affetto da errore.

In tal modo, *dato il valore di* $|E|_m$, *si può procedere alla correzione del risultato*.

Come esempio, riportiamo *un codice correttore, per numeri binari di 5 bits, usando un modulo* $m = 11$ (cioè, il minimo possibile, il quale si vede essere anche *sufficiente* allo scopo).

E	$\|E\|_{11}$
0	0
$+ 2^0$	1
$+ 2^1$	2
$- 2^3$	3
$+ 2^2$	4
$+ 2^4$	5
$- 2^4$	6
$- 2^2$	7
$+ 2^3$	8
$- 2^1$	9
$- 2^0$	10

Ad esempio, supponiamo che il risultato corretto di una certa operazione sia:

$$N = 10110 \rightarrow 22$$

A causa di un guasto nell'unità di calcolo, si ottenga un risultato errato dato da:

$$N' = 00110 \rightarrow 6$$

Come si vede, si tratta di un *errore singolo*, dato che:

$$E = N' - N = -2^4$$

Il check digit è dato da: $|E|_{11} = ||N'|_{11} - |N|_{11}|_{11} = 6$

La conoscenza di tale check digit ci è sufficiente ad affermare che è stato connesso un *errore di* -2^4, per cui si può procedere alla *correzione*.

IV

CENNO SUI CODICI NON SISTEMATICI (6)

Abbiamo visto come il controllo di parità numerica si ottiene *associando ad ogni numero* in generale *più cifre di parità numerica (check digits), nel loro insieme*, esse costituiscono il così detto "*CODICE RESIDUO*" *del numero*.

Abbiamo visto anche che il numero ed il codice residuo sono *trattati separatamente da due unità di calcolo distinte*.

Un codice residuo è detto anche "*CODICE SISTEMATICO*" , per distinguerlo da un altro tipo di codice, usato sempre per la rilevazione e la correzione di errori aritmetici, detto "*CODICE NON SISTEMATICO*": esso consiste nel trasmettere, invece che numero e codice separatamente, una *opportuna funzione del numero*, la quale *contiene da sè sia l'informazione che i relativi check digits, in modo del tutto inscindibile*.

Tale codice è una *funzione del numero N da codificare*. Ad esempio, i codici non sistematici più usati sono del tipo:

$$C\ (N) = A.N$$

$$C\ (N) = A.N + B$$

dove A e B sono due costanti da scegliersi opportunamente.

Abbiamo anche già detto che, per la loro stessa natura, questi codici sono tali che *tutte le cifre della parola vengono trattate da una medesima unità di calcolo, venendo co-*

sì a fare a meno di una rete di controllo separata.

Per contropartita, però, *l'unità di calcolo stessa risulterà più pesante,* sia perchè *la parola codificata ha un certo numero di bits ridondanti*, sia perché *vengono alterate le modalità di esenzione delle operazioni aritmetiche.*

1 - CODICE TIPO AN

In questo caso, il numero codificato si ottiene moltiplicando il numero N per una costante intera A, scelta opportunamente. Ovviamente, un tale codice *si preserva in seguito ad operazioni di addizione e sottrazione:*

$$C(X) = AX \; ; \quad C(Y) = AY$$

$$C(X) \pm C(Y) = C(X \pm Y)$$

Il concetto base che permette la *RILEVAZIONE* di errori è il seguente: calcolando il *residuo mod A del risultato di una certa operazione,* se questo è corretto si deve ottenere zero; se, invece, è stato introdotto un errore, si otterrà un numero diverso da zero, che ci fornirà la rilevazione dell'errore stesso.

E' chiaro come, a questo scopo, A debba essere, intanto, *primo con la base della numerazione.*

Inoltre, *non devono essere divisibili per A gli errori appartenenti alla classe che vogliamo rilevare*, come si vede scrivendo il risultato errato N' nella forma:

$$N' = N \pm E$$

(con N risultato corretto), da cui

$$|N'|_A = \left| |N|_A \pm |E|_A \right|_A$$

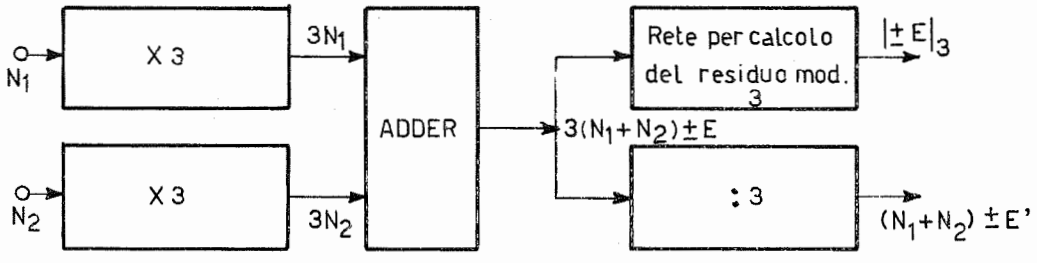
Nel caso di *RILEVAZIONE DI ERRORI SINGOLI*, cioè del tipo:

$$E = \pm 1 \cdot 2^i$$

si potrà scegliere, come *valore minimo* (per la relazione 9):

$$\boxed{A = 3}$$

Un possibile schema a blocchi di addizionatore, controllato con codice tipo AN per la rilevazione di errori singoli, può essere il seguente, *ammesso che i numeri siano rappresentati, esternamente all'unità di calcolo, nella loro forma naturale:*



La ridondanza di un codice di tipo AN (cioè, il numero dei bits ridondanti della parola codificata rispetto a quella non codificata), si può calcolare al seguente modo:

$$\lg_2 A - 1 < R < \lg_2 A + 1$$

Nel caso di A = 3 si possono avere, dunque, *1 oppure 2 bits ridondanti.*

A questo punto, comunque, è importante notare come un codice di tipo AN risulti alterato in seguito ad operazioni di *moltiplicazione, divisione* e *complementazione*. Cioè, in questi casi, *il risultato non viene ottenuto direttamente nella forma AN, applicando le regole aritmetiche dei numeri naturali.*

E' facile verificare che *il risultato può essere ricon*

dotto alla forma AN, mediante una opportuna *correzione*.

2 - CODICE TIPO AN + B

Un tale codice si ottiene aggiungendo una costante B opportuna a quello tipo AN: in tal modo, *sono preservare le stesse capacità di controllo proprie del codice AN stesso*, ma, in più, il codice AN + B possiede la proprietà assai importante di essere *autocomplementante*; cioè, *si ottiene il complemento del numero codificato, semplicemente complementandone i singoli digits*.

Infatti, premesso che la somma viene alterata per una costante additiva B:

$$(AN_1 + B) + (AN_2 + B) = A\ (N_1 + N_2) + 2B$$

la proprietà precedente si vede essere verificata in base alla seguente relazione:

$$AN + B + A\ [N_{MAX} - N] + B = 2^n - 1$$

Cioè:

$$A\ N_{MAX} + B = 2^n - 1$$

essendo proprio, $N_{MAX} = 2^n - 1$, per un numero di n bits.

La relazione precedente permette di *ricavare il valore di B*, una volta scelto quello di A, in base alle considerazioni del paragrafo precedente.

BIBLIOGRAFIA

(1) I.FLORES : The logic of Computer Arithmetic. Prentice Hall, 1963 - Cap.18

In questo capitolo, oltre ai concetti fondamentali della congruenza, sono riportate le proprietà della notazione a più moduli, che può trovare applicazione in sistemi di calcolo a cui sia richiesta una elevatissima affidabilità .

(2) H.L.GARNER : Generalized Parity Checking - IRE Trans. EC , 1958, pag.207

In questo articolo è riportata una generalizzazione della parità, basata sull'uso delle congruenze, la quale conduce sia al controllo di parità digitale che a quello di parità numerico, facente uso dei codici residui.

(3) F.F.SELLERS, H.Y.HSIAO, L.W.BEARNSON : Error detecting Logic for digital computers. McGRAW HILL, 1968

In questo libro é data una panoramica dei metodi più usati per la diagnosi in hardware, che vanno dall'uso della differenza booleana al controllo di parità digitale e numerica; in più, sono date altre tecniche di un certo interesse, che tendono ad ottenere il controllo di parti di un sistema di calcolo diverse dall'unità aritmetica. Per ogni applicazione è dato un gran numero di realizzazioni circuitali pratiche, molto utili.

(4) SZABO, TANAKA : Residue Arithmetic and Its Applications to Computer Technology-Cap.14-McGraw-Hill, 1967

Questo capitolo riporta alcuni concetti fondamentali e del tutto generali circa i codici residui, sistematici e non sistematici; la loro applicazione è estesa sia alla rilevazione che alla correzione di errori nelle unità di calcolo.

(5) T.R.N. RAO : Error - Checking Logic for Arithmetic - IEEE Trans. EC, Settembre 1968, pg. 845.

Questo articolo riporta uno schema pratico di una unità aritmetica controllata mediante l'uso di codici residui, per quan

to riguarda la rilevazione di errori singoli.

(6) D.T. BROWN : Error Detecting and Correcting Binary Codes for Arithmetic Operations. IRE Trans. EC, 1960, pag. 333.

Questo articolo passa in rivista i principali codici non sistematici, le loro proprietà e le loro applicazioni, sia per quanto riguarda la rilevazione, che per quanto riguarda la correzione di errori nelle unità di calcolo.

Altre pubblicazioni interessanti su-l'argomento trattato in questa nota sono:

(7) GARNER : Error Codes for Arithmetic Operations. IEEE Trans. EC, 1966, pag. 763.

(8) GADDES : An Error - Detecting Binary Adder: A Hardware-Shared Implementation. IEEE Trans. EC, Gennaio 1970, pag. 34.

(9) Arithmetic Divisible Codes with Correction for independent Errors. DADAYEV. ENGINEERING CY BERNETICS, 1965, n. 6, pag. 79.

(10) W.W. PETERSON : Error Correcting Codes. The M.I.T. Press, Cambridges, Mass., and John Wiley and Sons, Inc., New York, 1961.

(11) J.L.MASSEY : Survey of residue coding for arithmetic errors - ICC Bullettin, vol.3, 1964.

In questo articolo sono dimostrate rigorosamente le possibilità dei vari codici residui di rilevare e correggere errori di determinata molteplicità su base di concetti di " peso " e " distanza ".

(12) T.R.N. RAO : Biresidue Error-Correcting Codes for Computer Arithmetic - IEEE Trans. c-19,maggio 1970.