Macchine Aritmetiche

Substractors

Corso di Architettura dei Calcolatori

Prof. Salvatore Venticinque

(MEI cap. 5 e 7)

Università degli Studi della Campania *Luigi Vanvitelli*

Sottrattori modulo M

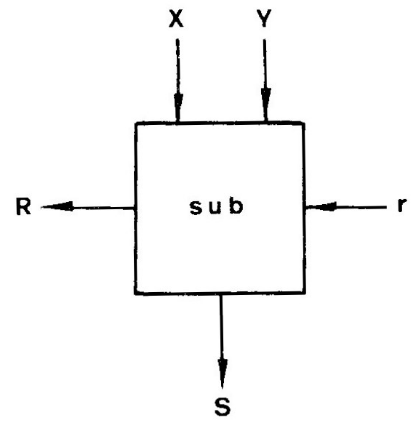
- Un sottrattore modulo M (full subtractor) è una macchina che esegue le operazioni:
  - $S=|X-Y-r|_M$
  - $R=[(X-Y-r)/M]$
- Dove:
  - X e Y sono gli operandi
  - S è la differenza
  - R è il prestito uscente, r è il prestito entrante
  - M è il modulo
- X, Y, S sono segnali che rappresentano numeri in aritmetica modulo M, R e r sono bit
- Definiamo anche la differenza ***D=X-Y-r=S-RM***

Half subtractor modulo M

- Un semisottrattore modulo M (half subtractor) è un sottrattore modulo M in cui r è identicamente nullo (ovvero privo del segnale di prestito entrante r)
  - $S=|X-Y|_M$
  - $R=[(X-Y)/M]$

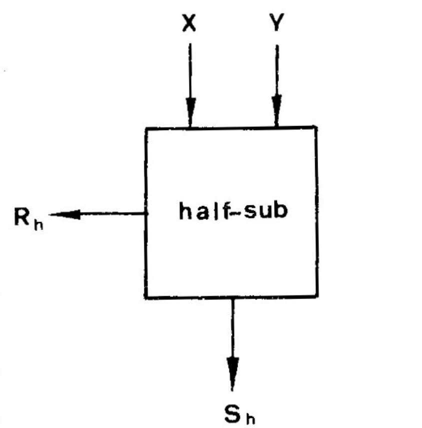
  con la stessa simbologia già introdotta

Rappresentazione



X	Y	r	S	R
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

a)



X	Y	S_h	R_h
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

b)

Figura 12.1 – Sottrattori binari elementari: a) sottrattore completo: b) semisottrattore.

Interpretazione

- Vale che:
  - Se $D \geq 0$ $S=D$, altrimenti $S=M+D$
  - Se $D \geq 0$ $R=0$, altrimenti $R=1$
- Se X, Y e S sono in aritmetica di interi modulo M, R è un segnale di overflow
- Se X, Y e S sono in aritmetica estesa la differenza è sempre definita ed è data da

  $D=S-RM$

Realizzazione di un half subtractor

- Si potrebbe realizzare un subtractor con l'algoritmo classico, ma si sfruttano gli adder
- Posto Y' = M-Y vale:
  - $|X+Y'|_M = |X-Y+M|_M = |X-Y|_M$
  - $(X+Y' \geq M) = (X-Y \geq 0)$
- Pertanto la sottrazione $|X-Y|_M$ si può ottenere:
  - Con un half subtractor $|Y-X|_M$ considerando S'=M-S e R'=R
  - Con un half adder X+(M-Y) considerando R'=R
  - Con un half adder Y+(M-X) considerando S'=M-S
- Relazioni notevoli:
  - L'half subtractor $S = |X-Y|_M$ è in relazione con $S' = |Y-X|_M$:
  - $S' = |Y-X|_M = M - |X-Y|_M = M-S$
  - $R' = (Y-X<0) = (X-Y>0) = R$

Schemi equivalenti

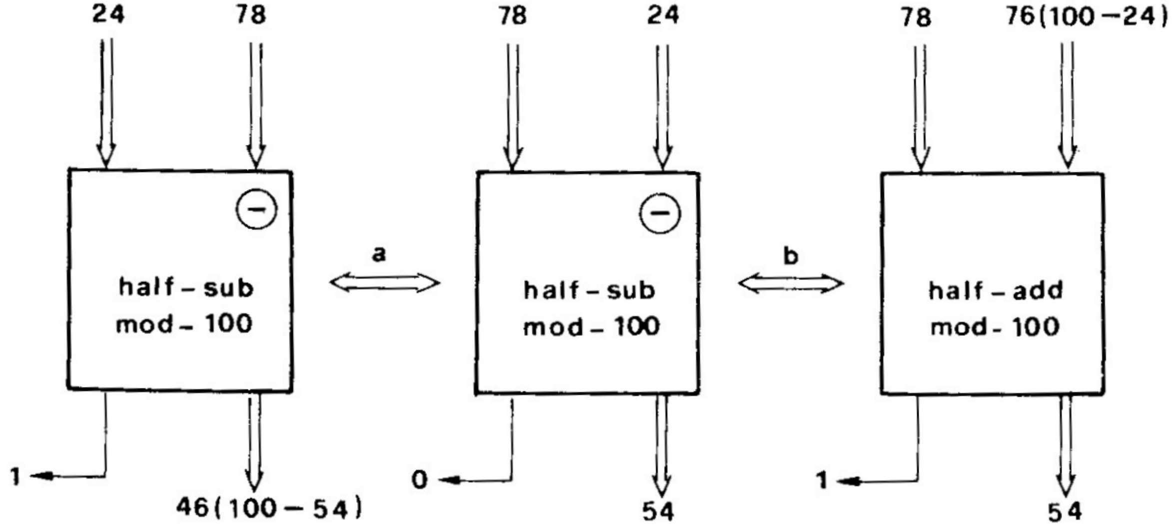


Figura 4.1 – Equivalenza fra semiaddizionatori e semisottrattori: a) relazione (4.6); b) relazione (4.7).

Realizzazione di un full subtractor

- Si può realizzare un full subtractor come visto per l'half subtractor complementando anche il riporto entrante ed utilizzando i complementi a M-1 anziché a M
- Infatti
  - $S' = |X' + Y + r|_M = |M - Y - X + r - 1|_M = M - |X - Y - r|_M - 1 = M - S - 1$
  - $R' = (X' + Y + r \geq M) = (M - X - 1 + Y + r \geq M) = (X - Y - r + 1 \leq 0) = (X - Y - r < 0) = R$
- e
  - $S' = |X + Y' + r'|_M = |X + M - Y + 1 - r - 1|_M = |X - Y - r|_M = S$
  - $R' = (X + Y' + r' \geq M) = (X + M - Y - r \geq M) = (X - Y - r \geq 0) = R$
- ed infine
  - $S' = |Y - X - r'|_M = |Y - X + r - 1|_M = |Y - X + r|_M - 1 = M - |X - Y - r|_M - 1 =$

    $= M - S - 1$
  - $R' = (Y - X - r' < 0) = (X - Y - r + 1 > 0) = (X - Y - r > -1) = (X - Y - r \geq 0) = R$

Schemi equivalenti

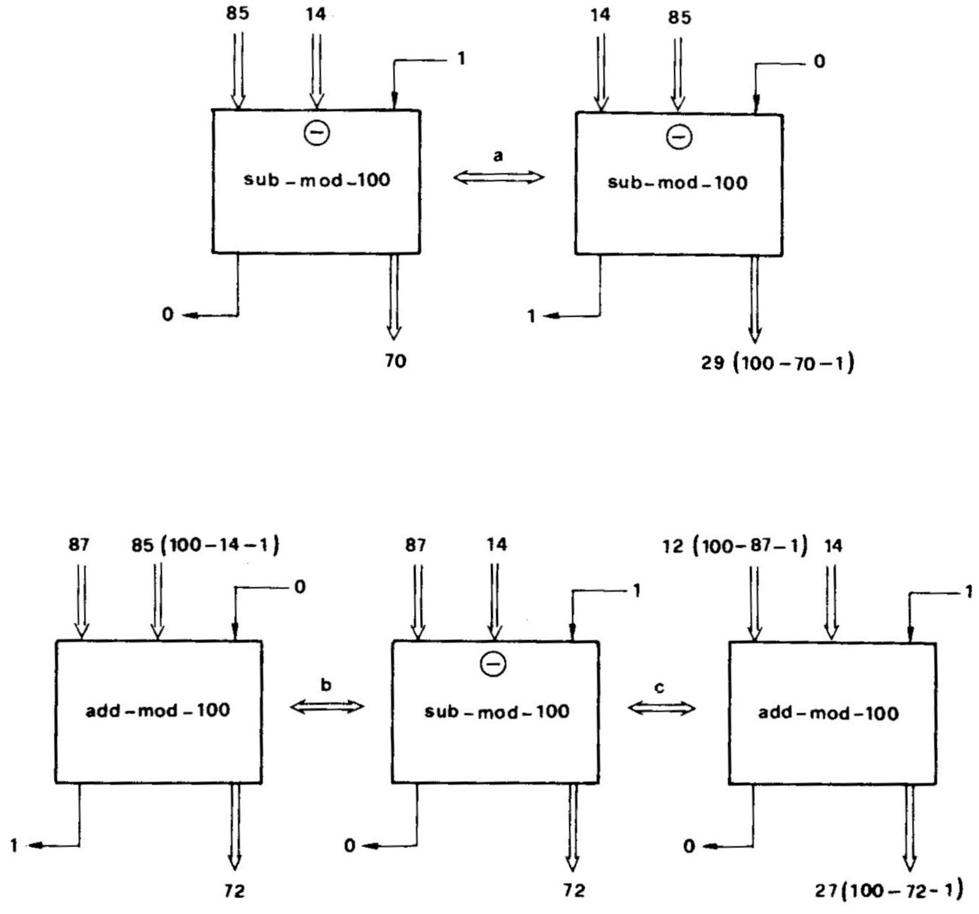


Figura 4.2 – Equivalenza tra addizionatori e sottrattori in modulo: a) relazione (4.6″); b) relazione (4.7″); c) relazione (4.8″).

Addizionatori in modulo diminuito

- Sono addizionatori in modulo $M \neq 2^k$
- E’ possibile costruire un addizionatore in modulo diminuito M adoperandone uno in modulo $M' = 2^k$ (con $M < M'$)
  - Posto $M = M' - k$ (con $0 < k < M$) e $\Sigma = X + Y + r$,
  - $R = (\Sigma \geq M)$, $S = |\Sigma|_M$, $R' = (\Sigma \geq M')$, $S' = |\Sigma|_{M'}$:
  - $R = (\Sigma \geq M) = (\Sigma \geq M' - k) = (\Sigma \geq M') \text{or} (M \leq \Sigma < M') =$

    $= (\Sigma \geq M') \text{ or } (S \geq M) = R' \text{ or } (S' \geq M)$
  - $S = S'$ se $R = 0$, $S = |S' + k|_{M'}$ se $R = 1$

Schema

- Algoritmo:
  - R=R’ or S’≥M’-k
  - If R then q=k else q=0
  - Somma q e S’
    (R’’ è ignorato)

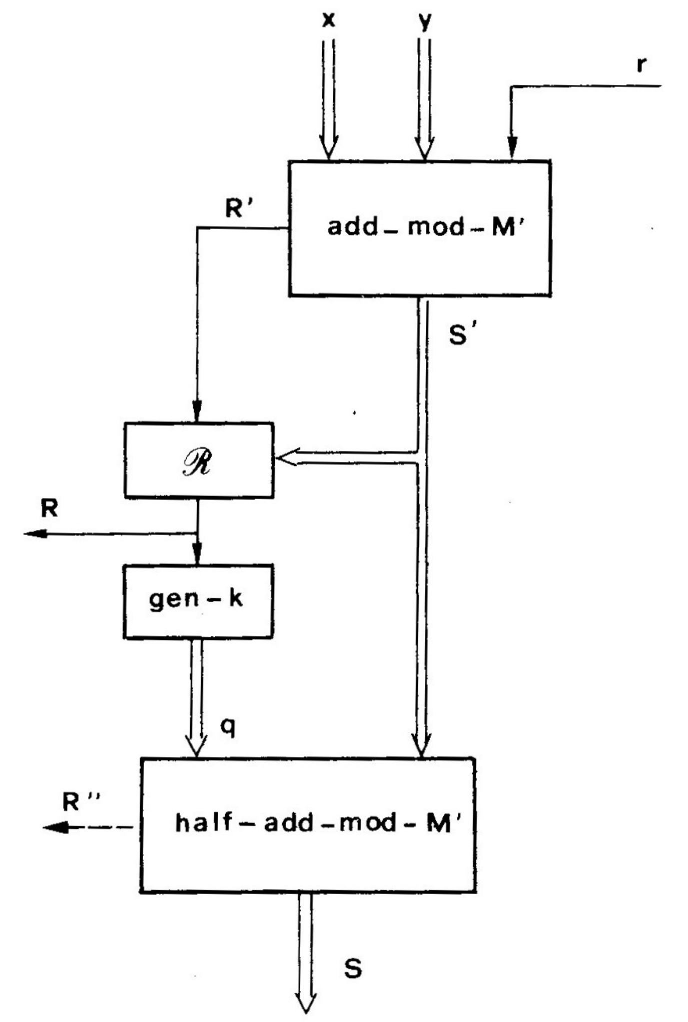


Figura 5.1 – Architettura di un addizionatore in modulo diminuito.

Addizionatori di numeri relativi

- Caso semplice: rappresentazione in segno e modulo, usando i complementi (si genera un overflow che segnali il cambio illecito di segno):

```
if (segno(X) = segno(Y)) {
    |Z| = |X| + |Y| con un half adder
    if (segno(Z) = segno(X))
    overflow=0
}
else {
    |Z| = |X| - |Y| con un half subtractor
    if R {
    |Z| = M - |Z|
    segno(Z) = segno(Y)
    }
    else segno(Z) = segno(X)
```

- L'half subtractor può essere sostituito come visto con un half adder

Addizionatori di numeri relativi

- Rappresentazione in complementi $[M_1,M_2)$: vale che
  - M pari: $M_1 = -M/2$, $M_2 = M/2-1$
  - M dispari: $M_1 = -M/2$, $M_2 = M/2$
- Detti X e Y le rappresentazioni di x e y,
  - $|x+y|_M = ||x|_M+|y|_M| = |X+Y|_M$
- Quindi l'half adder funziona anche in caso di numeri relativi rappresentati per complementi purchè si generi un segnale di overflow dato da

(X<M/2) and (Y<M/2) and (Z≥M/2) or

(X≥M/2) and (Y≥M/2) and (Z<M/2)

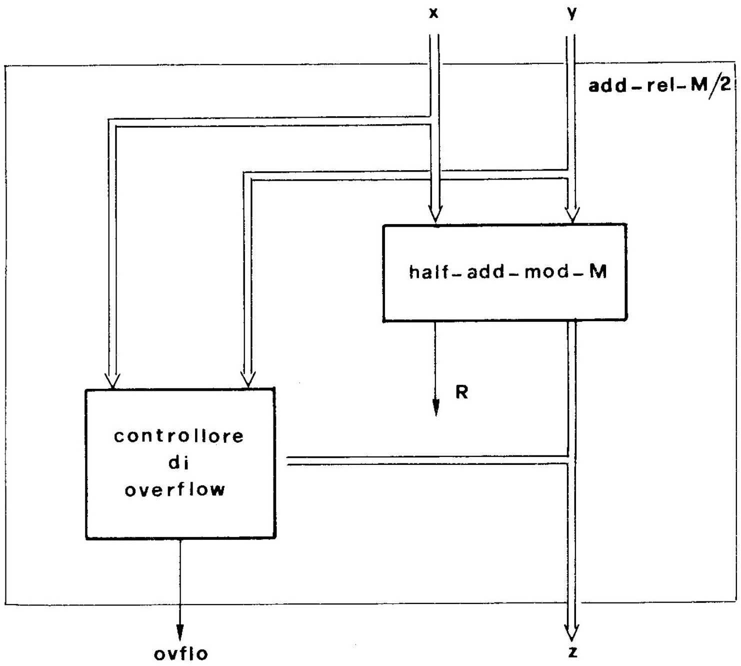


Figura 7.1 – Architettura di un addizionatore di numeri relativi rappresentati in complementi.

Moltiplicatore modulo M

- Un moltiplicatore modulo M è una macchina che realizza

  $P=|XY|_M$

  $Q=[XY/M]$

  in cui P rappresenta il prodotto degli operandi e Q segnala se XY>M
- In aritmetica estesa, Z=XY ϵ [0,M2) e vale

  $Z=|Z|_M+[Z/M]M=P+QM$ in cui P,Q ≤ M
- In aritmetica dei frazionari vale:

  x’=X/M, y’=Y/M, z’=x’y’ = XY/M^2 con x,y,z ϵ [0,1) e non si ha mai overflow, quindi si può assumere Z come valido purchè scalato ($z'=Z/M^2=Q/M+P/M^2$) e Q è il valore approssimato del prodotto, P indica l’errore di approssimazione
- In aritmetica di interi in complementi [-M/2.M/2) vale

  $|xy|_M=||x|_M|y|_M|=|XY|_M=P$

  ma Q perde di significato e non si ha alcuna condizione di overflow, quindi si preferisce usare i resti modulo M^2

Moltiplicatore di interi positivi

- Si consideri $M=b^n$:
- X, Y, Q, P hanno n “cifre”, Z ha 2n “cifre”
- Vale $Z = Qb^n + P$ quindi Q rappresenta le prime n cifre di Z e P le ultime n
- In aritmetica di interi in [0,M):
  - in assenza di overflow Z=P,
  - in aritmetica estesa Z vale quanto visto,
  - in aritmetica di frazionari se X rappresenta x’=X/M e Y rappresenta y’=Y/M la rappresentazione di z’=x’y’ è data dalle cifre di Q lette con il punto frazionario in prima posizione e P rappresenta l’approssimazione con un fattore di scala b^{-2n}

Esempi

Se n=4, b=10, X=0025, Y=0152

Q=0000, P=3800

In aritmetica di interi in [1, 10000) è 25x152=3800

In aritmetica frazionaria è 0.0025x0.0152=0.00003800 quindi Z≈Q=0 (underflow) con errore 0.000038

Se n=4, b=10, X=2500, Y=5555

Q=1389, P=3055

In aritmetica di interi in [1, 10000) è 2500x5555=13893055

che è rappresentabile in [1, 10^8)

In aritmetica frazionaria è 0.2500x0.5555=0.13893055 quindi Z≈Q=0.1389 con errore 0.00003

Realizzazione

- Per quanto visto basta realizzare un moltiplicatore di interi
- Algoritmo manuale:

$Z=0$

for $i=0$ to $n-1$

$\qquad Z=Z+XY_ib^i$

- E' quindi necessario un moltiplicatore di un numero per una cifra $C=XY_i$

	$3\ 2\ 3\ 5\ \times$	
	$3\ 5$	
$Q^{(1)} = 1617$	$1\ 6\ 1\ 7\ 5$	$P^{(1)} = 5$
	$9\ 7\ 0\ 5$	$C^{(2)} = 9705$
$Q^{(2)} = 1132$	$1\ 1\ 3\ 2\ 2\ 5$	$P^{(2)} = 25$
	$R^{(1)} = Q^{(1)} + C^{(1)}$	

Figura 10.1 – Esempio di moltiplicazione manuale.

Realizzazione

- Nel caso binario Y_i vale 0 o 1 pertanto $C=XY_i$ si realizza come

  if Y_i $C=X$ else $C=0$ ($C=X$ and Y_i)

- L'algoritmo richiede uno (seriale) o n (parallelo) addizionatori su 2n cifre (dimensionati per rappresentare Z), ma in ogni passo si usano solo n cifre
- Si noti che dopo ogni ciclo $Z=XY'$ con Y' formato dalle ultime i+1 cifre di Y e vale $Z=XY'=P+QM'$ con Y in $[0,b^i)$, $M'=b^i$ e Q,P valori parziali di quoziente e resto per il valore corrente di Z
- Quindi si può trasformare il for in modo da calcolare separatamente Q e P

Realizzazione

- In analogia col procedimento manuale in cui Q è la parte di sinistra e P è la parte di destra del prodotto parziale, in ogni ciclo si calcolano:

  $C=XY_i$

  $R=Q+C$

- e quindi il nuovo Q si ottiene staccando da R la cifra meno significativa che va in testa al P precedente

  $Q=[R/b]$

  $P=P+|R|_b * b^i$

- Quindi:
  - Il calcolo di R avviene tra un addendo C di n+1 cifre e un augendo di n cifre -> uso un adder a n+1 cifre
  - Il calcolo di P avviene ponendo $P_i=R_0$
- Il calcolo di Q consiste nello shift a destra di R con perdita dell'ultima cifra

Algoritmo e schema parallelo

Q=0

P=0

for (i=0 to n-1) {

  moltiplica C=XY$_i$

  somma R=Q+C

  P$_i$=R$_0$

  Q=[R/b]

}

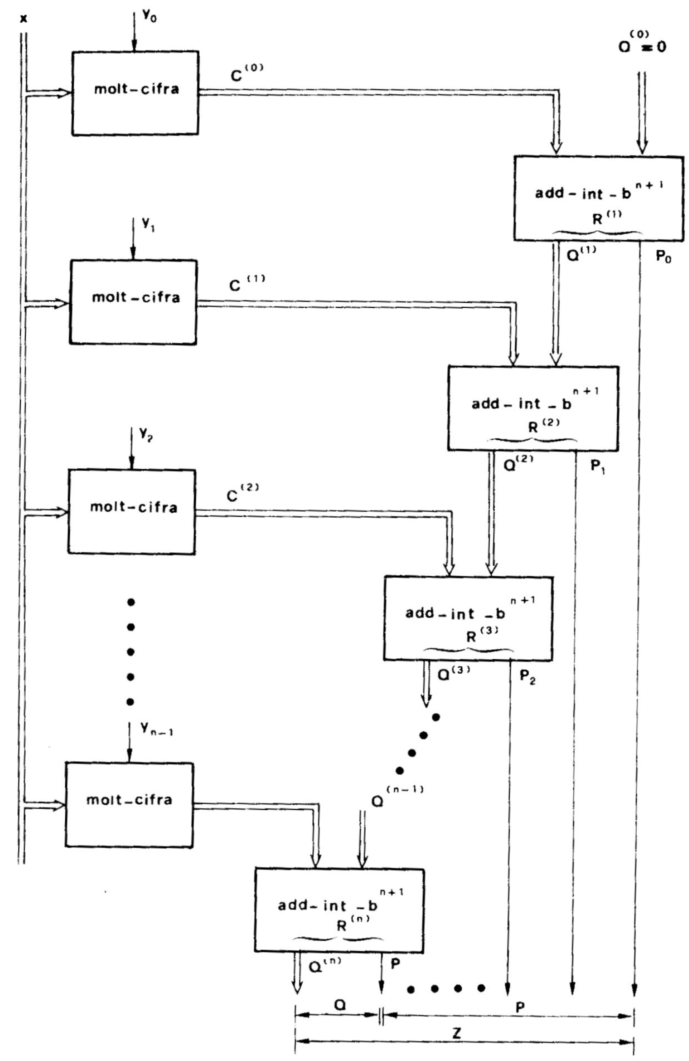


Figura 10.2 – Architettura di un moltiplicatore *mod-M* parallelo.

Algoritmo e schema seriale

Si usa uno shift register di 2n+1 bit che contiene Z tramite Q e P

```
Z.Q=0
Z.P=Y
for (i=0 to n-1) {
    moltiplica C=XZ0
    somma R=Z.Q+C
    Z.Q=R
    Shr (Z,0)
}
```

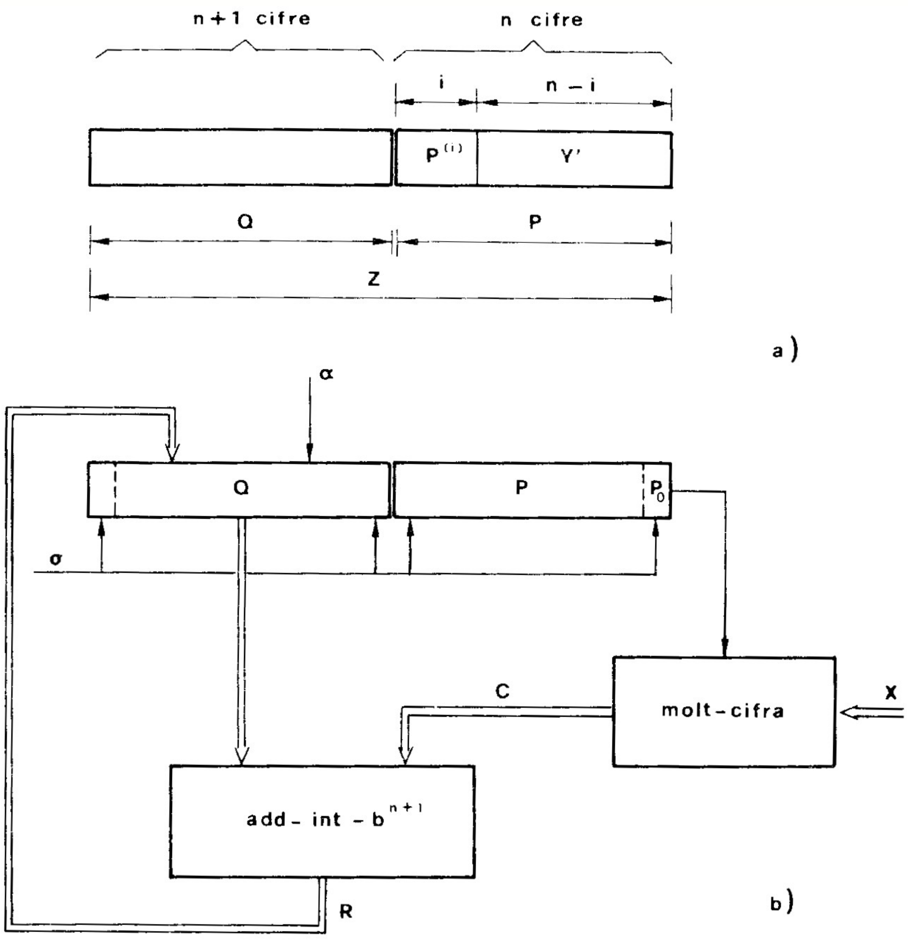


Figura 10.3 – Architettura di un moltiplicatore *mod-M* seriale.

Algoritmo seriale con b=2

- Nel caso binario il prodotto è una and e quindi si può ottenere una realizzazione seriale semplice sommando comunque e aggiornando la somma parziale solo se necessario:

```
Z.Q=0
Z.P=Y
for (i=0 to n-1) {
    somma R=Z.Q+C
    if Z0 Q=R
    Shr (Z,0)
}
```

- Questa realizzazione è più veloce perché se non devo ricaricare lo shift register posso direttamente azionare lo shift e risparmiare tempo

Moltiplicatori binari veloci

- Nel caso binario:
  - il caso parallelo è costoso (e ha ritardi a cascata) perché di solito il numero di bit è elevato
  - Il caso seriale è lento nonostante il risparmio della versione ottimizzata
- Si usa la tecnica di moltiplicazione per stringhe: per quanto visto, se si presenta una stringa di k 0 in ingresso si può scorrere di k posizioni in una volta
- E’ possibile operare analogamente per stringhe di k 1 con una notazione “alternativa” considerando l’uguaglianza:

  $\Sigma_{i=j->k-1}2^i=2^k-2^j$

- per la quale una stringa di 1 è la differenza tra due potenze di 2

Notazione “alternativa”

- Di conseguenza se Y_i=1 per $j \le i < k$ e Y_k=0 si può rappresentare lo stesso numero Y ponendo:

  $Y=\Sigma_{i=0->n-1} Y_i 2^i = \Sigma_{i=0->n-1} Y'_i 2^i$

  dove ogni Y'_i può valere:

  - $Y'_i = Y_i$ se $i<j$ e $i>k$
  - $Y'_i = -1$ se $i=j$
  - $Y'_i = 0$ se $j<i<k$
  - $Y'_i = 1$ se $i=k$

  trasformando una stringa di 1 in una stringa di 0 (*stringa primaria trasformata*) che costituisce la rappresentazione ideale di Y usando le cifre (1, 0, -1)

- Ad esempio ($\underline{1}$ rappresenta -1):

  $10100111100101 \text{ -> } 10101000\underline{1}00101$

  o anche su più stadi se si creano nuove stringhe di 1 come nell’esempio:

  $000101110001111 \text{ ->} 0001100\underline{1}001000\underline{1} \text{ -> } 0010\underline{1}00\underline{1}001000\underline{1}$

Vantaggio

- Tale notazione “alternativa” massimizza i casi in cui è possibile operare con shift di k posizioni tramite la creazione artificiale di stringhe di k 0
- La trasformazione applicata a Y consente di effettuare una moltiplicazione eseguendo in ciclo:
  - Operazioni di shift
  - Operazioni di addizione
  - Operazioni di sottrazione

  a seconda che la cifra trasformata sia 1, 0 o -1
- Nella realizzazione, poiché non si alterano fisicamente le rappresentazioni di Y, si controlla l’addizionatore seriale mediante un automa riconoscitore di sequenze di lunghezza k che tiene traccia della “cifra artificiale”
- Per motivi pratici l’automa controlla solo sequenze di lunghezza fissa k per limitare la casistica degli ingressi

Divisore modulo M

- Un divisore modulo M è una macchina che realizza
  - $Q=[D/Y]$
  - $R=|D|_Y$

  in cui R rappresenta il resto della divisione e D è il dividendo:
  - $D \in [0, M^2)$; $Y, Q, R \in [0, M)$
  - $D/Y<M$
  - $D=QY+R$
- Se R=0 questa macchina è l'inverso del moltiplicatore esteso D=QY
- Q è una approssimazione del rapporto (reale) D/Y: Q=D/Y-R/Y con R/Y approssimazione
- Per l'aritmetica dei frazionari bisogna usare una macchina lievemente diversa che scali il divisore moltiplicandolo per M allo scopo di salvaguardare il calcolo di Q

Divisore di interi positivi

- Si usa l'algoritmo manuale per la divisione (D e Y interi a n cifre)
- Detto $D^{(i)}=[D/b^{n-i}]$ il numero formato dalle prime i cifre di D l'algoritmo calcola al passo i-esimo
  - $Q^{(i)}=[D^{(i)}/Y]$ quoziente parziale
  - $R^{(i)}=|D^{(i)}|_Y$ resto parziale

  con $D^{(0)}=Q^{(0)}=R^{(0)}=0$, $D^{(n)}=D$; $Q^{(n)}=Q$; $R^{(m)}=R$
- Si costruisce un dividendo parziale P aggiungendo al resto corrente le cifre di D e da questo si calcolano le cifre q di Q
- Algoritmo (si sceglie un $P^{(k)}=D^{(k)}$ tale che $P^{(k-1)}<Y\leq P^{(k)}$):

```
for (i=k to n) {
    calcola cifra di Q q(i)=[P(i)/Y]   (0≤q(i)<b)
    calcola resto parziale R(i)=P(i)-q(i)Y (0≤R(i)<p(i))
    calcola nuovo dividendo parziale con una cifra di D P(i+1)=R(i)b+Dn-1
}
```

Esempio

```
  9827   : 0341
– 682      28
  3007
– 2728
   279
```

i	D	P	q	Q	R
0	0	0	0	0	0
1	9	9	0	0	9
2	98	98	0	0	98
3	982	982	2	2	
					300
4	9827	3007	8	28	
					279

a)

```
  341000   : 478
– 3346       713
   640
  –478
  1620
 –1434
   186
```

i	D	P	q	Q	R
0	0	0	0	0	0
1	3	3	0	0	3
2	34	34	0	0	34
3	341	341	0	0	341
4	3410	3410	7	7	
					64
5	34100	640	1	71	
					162
6	341000	1620	3	713	
					186

b)

Figura 14.1 – Esempi di divisione manuale: a) Divisione fra interi: $n = 4$; $m = n$; $D = 9827$; $Y = 341$; $Q = 28$; $R = 279$; $k = 3$; b) Divisione fra frazionari: $n = 3$; $m = 2n$; $D = 341000$; $Y = 478$; $Q = 713$; $R = 186$; $k = 4$.

Algoritmo per interi positivi

Algoritmo:

Q=0

P=0

for (i=1 to n) {

P=Pb+D_{n-1}

q=[P/y]

Q=Qb+q

R=P-qY

P=R

}

In questo caso la scelta P=0 non fa altro che eseguire cicli in cui si producono 0 come prime cifre del risultato

Algoritmo seriale (interi positivi)

- Uso il registro-quoziente Z composto da Z.P e Z.Q che contengono rispettivamente ad ogni ciclo $R^{(i)}$ ($P^{(i)}$) e le cifre di D non ancora usate (X) seguite da $Q^{(i)}$:

```
Z.P=0
Z.Q=X
for (i=1 to n) {
    Shl(Z,0)
    calcola quoto e resto q e R di Z/Y
    Z.P=R
    Z0=q
}
```

  considerando che lo shift esegue insieme Q=Qb e l'arricchimento del dividendo parziale $P=Pb+X_{n-i}$

- E' necessaria una macchina che calcola quoto e resto q=[P/Y] e R=P-C=P-qY

Schema seriale divisore

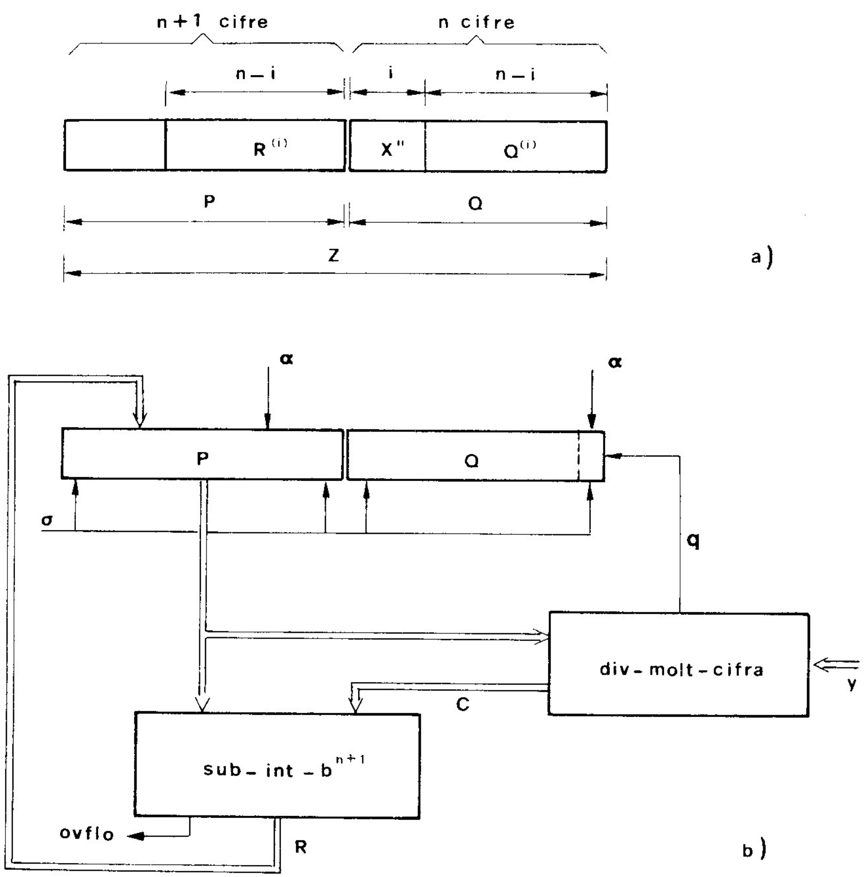


Figura 14.2 – Architettura di un divisore *mod_M* seriale.

```
  9827  : 0341
− 682     28
  3007
− 2728
   279
```

i	D	P	q	Q	R
0	0	0	0	0	0
1	9	9	0	0	9
2	98	98	0	0	98
3	982	982	2	2	
					300
4	9827	3007	8	28	
					279

a)

```
  341000   : 478
− 3346       713
   640
  −478
  1620
 −1434
   186
```

i	D	P	q	Q	R
0	0	0	0	0	0
1	3	3	0	0	3
2	34	34	0	0	34
3	341	341	0	0	341
4	3410	3410	7	7	
					64
5	34100	640	1	71	
					162
6	341000	1620	3	713	
					186

b)

Figura 14.1 – Esempi di divisione manuale: a) Divisione fra interi: $n = 4$; $m = n$; $D = 9827$; $Y = 341$; $Q = 28$; $R = 279$; $k = 3$; b) Divisione fra frazionari: $n = 3$; $m = 2n$; $D = 341000$; $Y = 478$; $Q = 713$; $R = 186$; $k = 4$.

Realizzazione del divisore base

- Esistono due metodi fondamentali:
  - Metodo di addizioni e sottrazioni (restoring)
  - Metodo di divisione per eccessi (non restoring)

  che realizzano entrambi:
  - $Q=[P/Y]$
  - $R=P-qY$

  seguite dalle memorizzazioni:
  - $P=R$
  - $Z_0=q$

Metodo del restoring

- R=P-qY può essere realizzata tramite q sottrazioni R=P-Y seguite da memorizzazioni P=R
- q non è noto a priori: la sottrazione può essere ripetuta finchè non risulti P<0 e comunque al massimo b-1 volte

```
q=0
repeat
    R=P-Y
    P=R
    Q=q+1
until P<0 or q=b-1
if P<0 {
    R=P+Y
    q=q-1
}
```

- Restoring modificato: esegue P=R, Q=q+1 solo se R>=0 introducendo un if dopo R=P-Y ed elimina l'ultimo if

Schema a restoring modificato

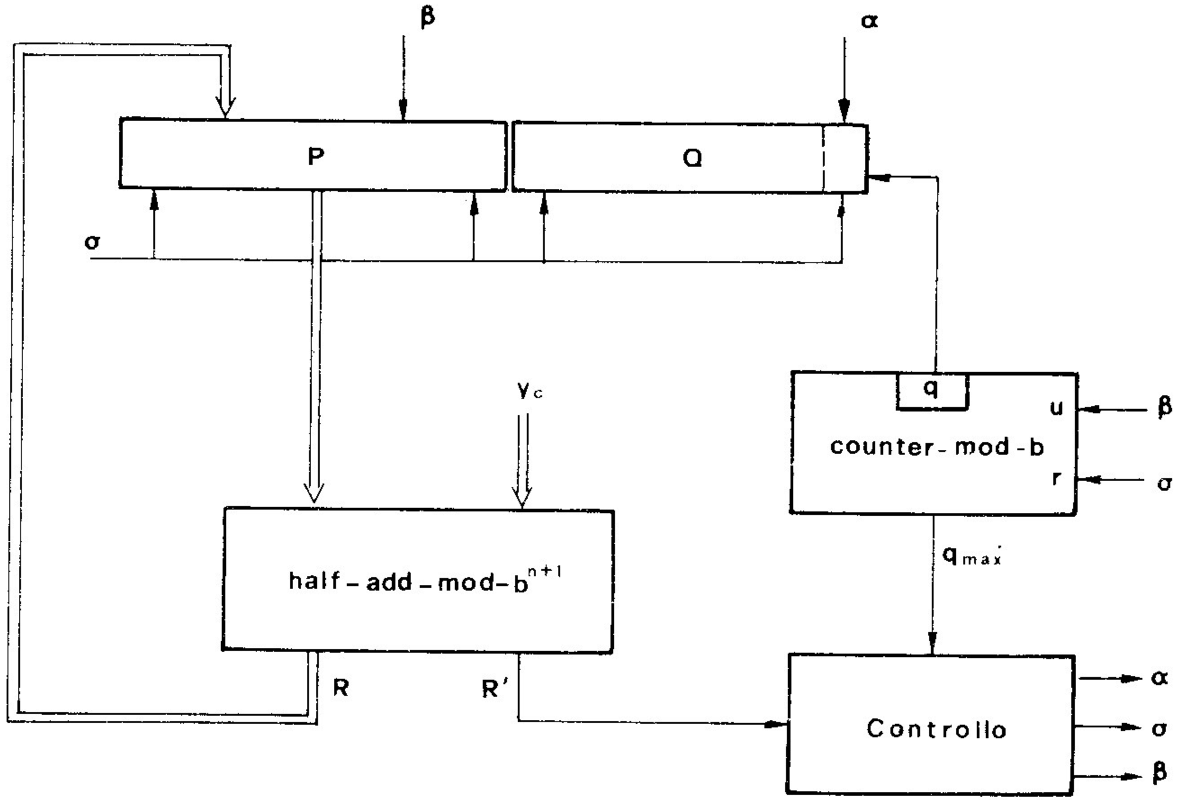


Figura 15.1 – Architettura di un divisore che implementa il metodo del restoring modificato.

Metodo del restoring per b=2

Per b=2 q≤1 e quindi il repeat si esegue una sola volta (b-1=1) e non serve il contatore che genera q

```
R=P-Y
P=R
q=1
if P<0 {
    R=P+Y
    q=0
}
```

o in forma di restoring modificato, considerando che dopo lo shift vale già che Z_0=0 e che per q=0 P resta inalterato:

```
R=P-Y
if R>=0 {
    P=R
    Z0=1
```

Metodo del non restoring

- L'idea è di effettuare le sottrazioni dovute e tenere l'eventuale resto negativo, assumendo come cifra-quoziente
  $P^{(i)}=q^{(i)}Y+R^{(i)}$ con $R^{(i)}<0$
  (una cifra "per eccesso") e come quoziente
  $D^{(i)}=YQ^{(i)}+R^{(i)}$ con $R^{(i)}<0$
- L'algoritmo può continuare regolarmente e calcola
  $P^{(i+1)}=R^{(i)}b+D_{n-1}$ con $P^{(i)}<0$,
  quindi con una nuova "cifra negativa" e un resto positivo
  $P^{(i+1)}=q^{(i+1)}Y+R^{(i+1)}$ con $q^{(i+1)}<0$, $R^{(i+1)}>0$

  $D^{(i+1)}=YQ^{(i+1)}+R^{(i+1)}$ con $R^{(i+1)}>0$

		i	*D*	*P*	*q*	*Q*	*R*
9827	:0341	3	982	982	3	3	
−1023	3(−2)						− 41
− 410							
+ 7							
− 403		4	9827	− 403	− 2	28	
+ 682							279
+ 279							

Figura 15.2 – Esempio di divisione per eccessi (coincide con esempio 14.1a).