Progettazione di un sistema per il prodotto di numeri naturali

P. Rullo

Si vuole progettare un processore che esegua l'algoritmo di moltiplicazione manuale tra numeri naturali. A tal fine, ricordiamo che la moltiplicazione binaria segue le medesime regole posizionali di una moltiplicazione decimale.

Esempio. Eseguiamo la moltiplicazione 111x101

```
   111 x
   101 =
   -------
    111
   0000
  11100
 -----------
 1000 11
```

Si noti che ogni risultato parziale è slittato verso sinistra di una posizione rispetto al risultato parziale precedente, ed è o uguale al moltiplicando (111), o a 0, a seconda che la cifra corrente del moltiplicatore sia, rispettivamente, 1 o 0. Si noti inoltre che il risultato è composto da un numero massimo di 2n bit, quando sia il numero di bit del moltiplicando (m1) che del moltiplicatore (m2) è pari a n. Nell'esempio sopra riportato, il risultato è di 6 bit, essendo sia m1 che m2 di 3 bit ciascuno.

L'algoritmo di moltiplicazione m1 x m2 può essere codificato come segue :

Algoritmo Prodotto Binario

```
1) leggi m1, m2
2) Prod = 0;
3) for (int i= 0; i < n; i++)          /* n è il numero di cifre del moltiplicatore m2
4)    {if (m2(i) == 1 ) {Prod = Prod + m1}
5)     m1 = m1*2  }                    /* equivale ad aggiungere uno zero a m1
```

Nell'esempio precedente, m2[0]=1 e, dopo il primo passo, Prod=111 e m1=1110. Al passo successivo, Prod non cambia (essendo m2[1]=0), e m1=11100. Al terzo passo, m2[2]=2 e Prod = 111+11100 = 100011, che è il risultato finale. Nella seguente tabella sono riportati i valori delle variabili ai vari passi dell'algoritmo.

i=0	Prod=0	m2(i)=1	Prod= 0 +m1= 111	m1 = 1110
i=1	Prod=111	m2(i)=0	Prod=111	m1 = 11100
i=2	Prod=111	m2(i)=1	Prod=111+m1=100011	m1= 111000

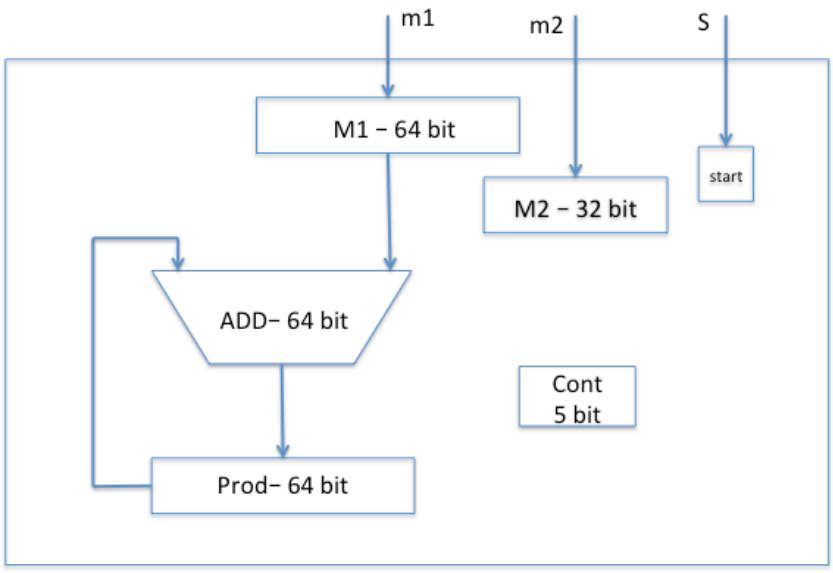
Modalità di funzionamento del Sistema di Elebaorazione

Il Sistema di Elaborazione è attivato da un segnale esterno Start. Quando Start=1 allora esso carica dall'esterno i valori m1 e m2 degli operandi e procede al calcolo del prodotto attraverso il suddetto algoritmo.
Si assume che gli operandi siano a n=32 bit, per cui il risultato sarà in generale di 2n = 64 bit.

Architettura Parte Operativa

L'architettura della PO è schematizzata nella seguente figura. Essa include un circuito sommatore a 64 bit, utilizzato per effettuare le somme tra i risultati parziali (istruzione 3 di *Algoritmo Prodotto Binario)*, oltre ai seguenti registri:

- M1: contiene il moltiplicando m1; è a scorrimento sinistro per supportare l'operazione 4 di *Algoritmo Prodotto Binario*. Siccome tale operazione viene ripetuta tante volte per quanti sono I bit del moltiplicatore m2 (al più 32), il registro M1 deve essere a 64 bit
- M2: contiene il moltiplicatore m2, quindi è a 32 bit; per supportare il test *m2(i)==1* di *Algoritmo Prodotto Binari,* M2 è a scorrimento destro; operando uno shift destro ad ogni passo del ciclo **for**, per leggere il valore m2[i] basta leggere l'ultimo bit più a destra M2[0]
- Prod: contiene il risultato, quindi è a 64 bit
- Cont: contatore a decremento a 5 bit (conta da 31 a 0) che viene utilizzato per implementare il ciclo for del programma
- Start: flip flop RS per la memorizzazione del bit Start per l'avvio del calcolo



Unità di Controllo

L'Unità di Controllo (UC) ha il compito di coordinare la Parte Operativa nell'esecuzione dell'algoritmo Prodotto Binario. Essa riceve il segnale Start dall'esterno, nonché i segnali di condizione dalla PO, ed invia a questa i segnali di controllo.

Microporgramma

Il microprogramma che implementa l'algoritmo *Prodotto Binario*, data la suddetta architettura della PO, è il seguente:

1) if Start == 0 **Φ** goto 1
2) m1→M1, m2→ M2, Azz(Prod); cont = 11111 /* cont= $(31)_{10}$
3) if M2(0) == 0 **Φ** goto 5;
4) M1+Prod → Prod;
5) shiftS(M1), shiftD(M2);
6) if cont > 0 decr(Cont); goto3
7) Start = 0; goto 1.

Si noti che, ad ogni passo, viene effettuato

- uno scorrimento sinistro di M1 (linea 5): equivale ad eseguire il prodotto m1*2 aggiungendo uno 0 a destra – istruzione 5 dell'algoritmo Prodotto Binario
- uno scorrimento destro di M2 (linea 5) - serve a posizionare la cifra corrente m2(i) del moltiplicatore in M2[0].

Quando M2(0) = 1 allora M1 viene sommato al risultato corrente in Prod (linea 4 del microcodice che corrisponde all'istruzione 3 dell'algoritmo Prodotto Binario). Il ciclo **for** dell'algoritmo *Prodotto Binario* è implementato con l'**if** del microprogramma alla linea 6 ed il relativo **goto**3. Il programma termina quando ogni singola cifra del moltiplicatore in M2 è stata moltiplicata per il moltiplicando in M1 (fine del ciclo **for** dell'algoritmo Prodotto Binario). Ciò avviene quando il registro Cont, inizializzato a 31 (linea 2) e decrementato ad ogni passo (linea 6), raggiunge il valore 0.

Segnali di controllo

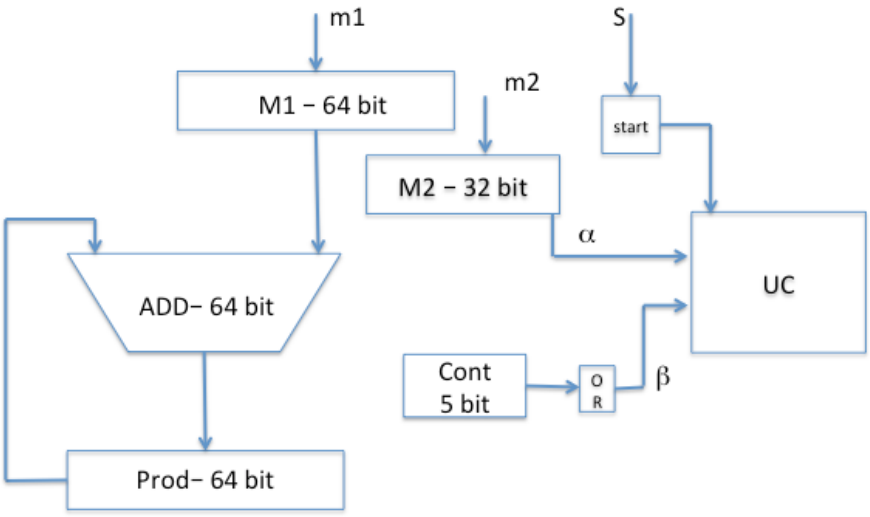
Assumiamo che i registri siano realizzati con flip-flop FAc. Quindi, i segnali di abilitazione A vengono usati per abilitare la modifica dello stato interno – attraverso caricamento dall'esterno, o altre operazioni come scorrimento o incremento/decremento di un registro contatore, ecc.

- Reg. M1: segnali A_{M1}, Sx – Se A_{M1} = 0 allora M1 mantiene il contenuto invariato; altrimenti, se Sx=0 allora M1 carica dall'esterno, altrimenti M1 opera uno shift sinistro
- Reg. M2: segnali A_{M2}, Dx – Come M1
- Reg. Prod: segnali A_p, Z_p - A_p = 0 allora Prod mantiene il contenuto invariato; altrimenti, se Z_p=0 Prod carica dall'ALU, altrimenti Prod viene azzerato

- Reg. Cont: segnali A_c, D_c – se A_c = 0, allora Cont rimane invariato; se A_c=1 allora, se D_c = 0, Cont viene inizializzato a 11111 (31 in base 10), altrimenti viene decrementato
- Z_S: segnale di azzeramento flip flop Start - coincide con il segnale R del flip flop RS

Segnali di Condizione

- α = M2(0): bit meno significativo di M2 – serve per implementare la condizione della linea 3 del microporgramma
- β = OR(Cont): vale 0 sse Cont=0 - serve per implementare la condizione della linea 6 del microporgramma.
- Start: serve per implementare la condizione al passo 1



Utilizzando i suddetti segnali di condizione, il microprogramma assume la seguente forma:

1) if Start == 0 Φ goto 1
2) leggi A, B; Azz(Prod); cont = 11111 /* cont= $(31)_{10}$
3) if α== 0 Φ goto 5;
4) M1+Prod → Prod;
5) shiftS(M1), shiftD(M2);
6) if β == 1 decr(Cont); goto 3
7) Start = 0; goto 1.

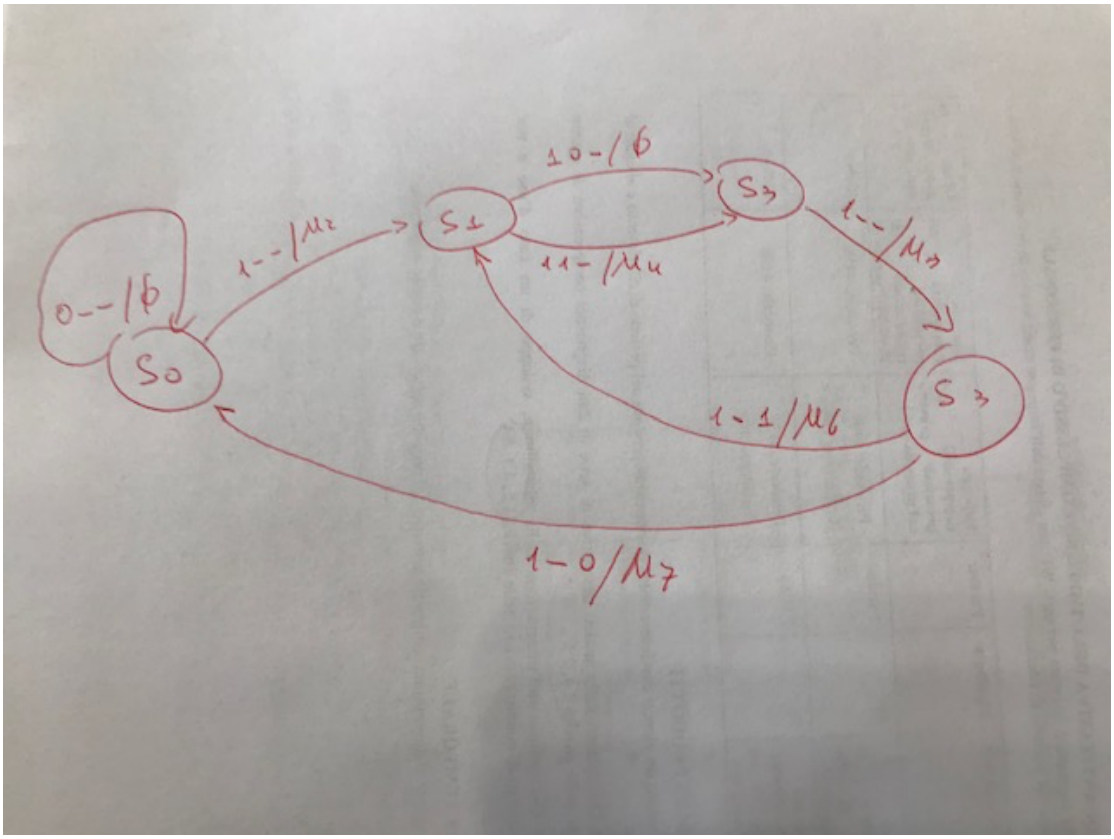
Codifica delle microistruzione

A titolo d'esempio, di seguito è riportata la codifica della miscroistruzione 2) – leggi A, B; Prod=0; cont = 11111.

A_{M1}	Sx	A_{M2}	Dx	A_p	Z_p	A_c	D_c	Zs
1	0	1	0	1	1	1	0	-

Unità di Controllo come Automa a Stati Finiti

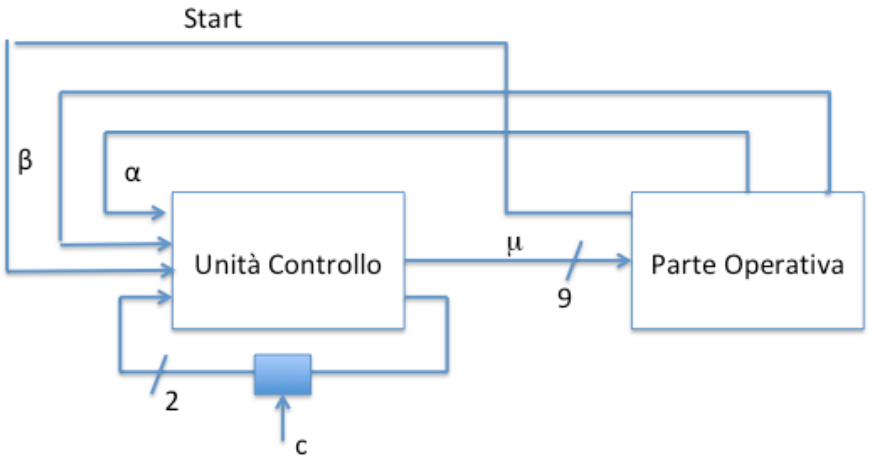
Nel seguente grafo, le etichette degli archi hanno la forma (Start α β/μ_i), dove Start, α e β sono i valori dei segnali di ingresso alla UC, e μ_i rappresenta la codifica binaria della i-esima microistruzioni del microprogramma; ad esempio, nella etichetta (11-/μ_4), 1 e 1 sono i valori, rispettivamente, di Start e α, mentre il valore di β è indifferente, e μ_4 è la codifica binaria della microistruzione 4 del microprogramma



Architettura Logica del Sistema di Elaborazione

L'unità di controllo è una rete sequenziale che implementa l'automa sopra riportato. Essa ha 3 variabili d'ingresso Start, α e β, e 2 variabili di anello (in quanto l'automa ha 4 stati).

La parte operativa riceve in ingresso 9 segnali di controllo che codificano le varie microistruzioni, e fornisce in uscita i valori delle variabili di condizione.



Una rete combinatoria per il prodotto di due numeri naturali a 4 cifre