



UNIVERSITA' DEGLI STUDI DI PARMA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Fondamenti di Informatica B

Lezione n.9

Alberto Broggi – Gianni Conte

A.A. 2005-2006

Fondamenti di Informatica B

Lezione n.9

- DESCRIZIONE E PROGETTO A LIVELLO RTL
- ESEMPIO DI SISTEMA A LIVELLO RTL:
IL MOLTIPLICATORE BINARIO
 - DESCRIZIONE DELL'ALGORITMO

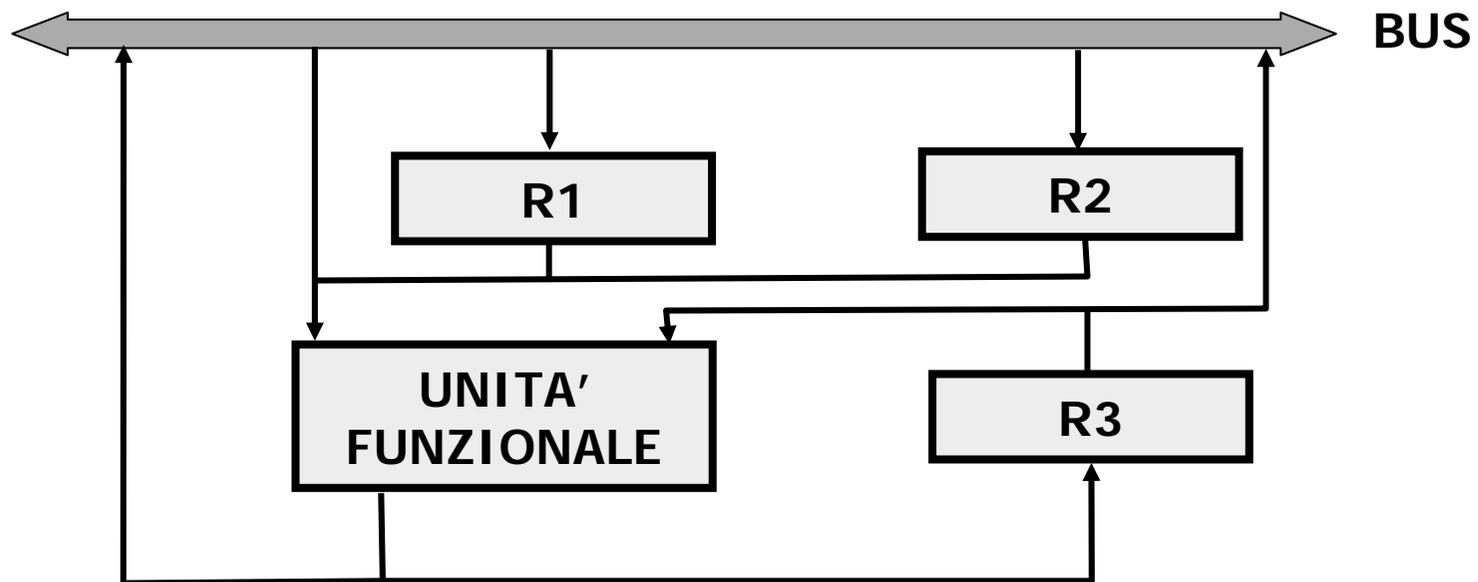
In questa lezione si continua la descrizione degli elementi di base utilizzati a livello RTL.

Si discute la suddivisione fra percorso dati e unità di controllo. In questa lezione verrà descritto a livello RTL un sistema: un moltiplicatore binario per numeri con rappresentazione in modulo e segno.

Verrà prima descritto l'algoritmo che consente di realizzare il prodotto e quindi sarà mostrata come esempio la metodologia di progettazione.

Tecniche di descrizione

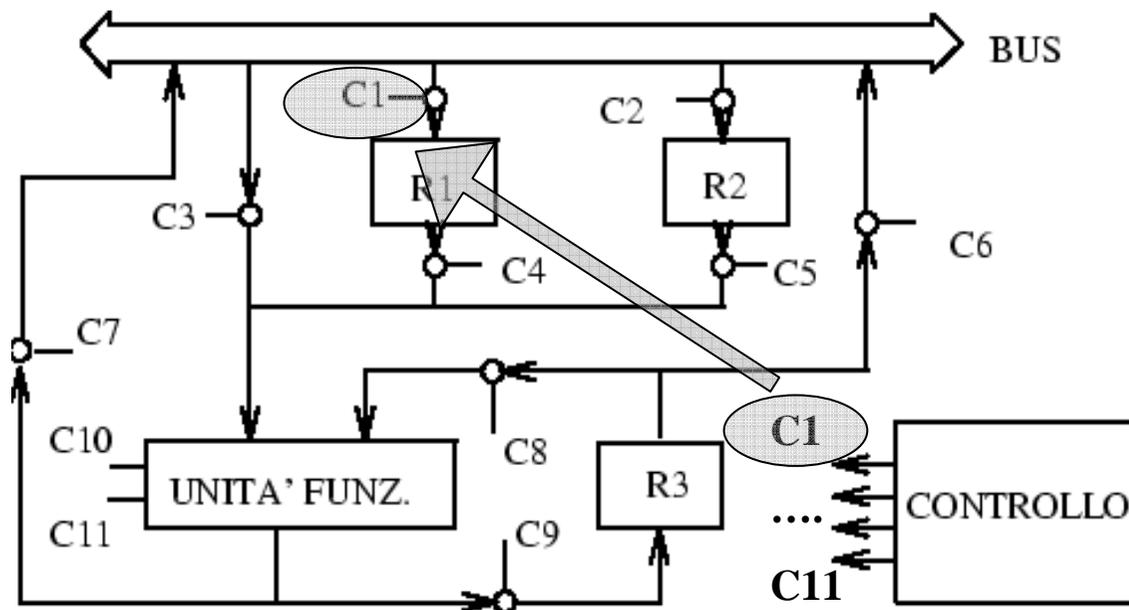
- La descrizione grafica definisce la *struttura* di un sistema a livello RTL
- La descrizione *comportamentale* è data mediante un linguaggio specifico



Unità dati e unità di controllo

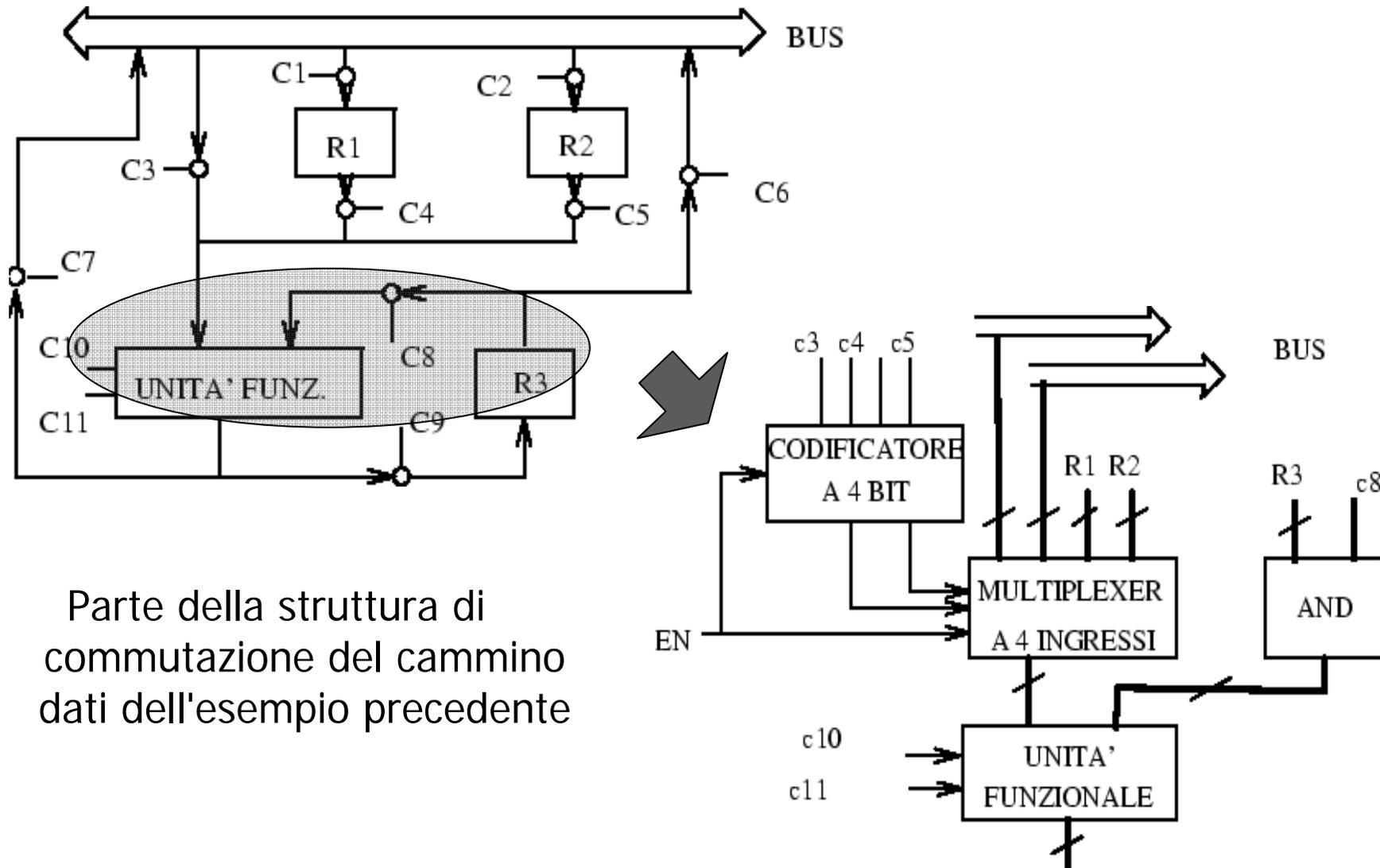
COMMUTAZIONE CAMMINO DATI

- Le connessioni non sono attive tutte contemporaneamente
- L'unità di controllo seleziona i cammini che risultano attivi in un determinato istante



Esempio:
Se il segnale C1 è attivo, l'interruttore corrispondente è chiuso e l'informazione può transitare dal BUS ad R1

Commutazione cammino dati



Parte della struttura di commutazione del cammino dati dell'esempio precedente

Linguaggio RTL

L'istruzione base RTL è la seguente:

$$Z \leftarrow f(X_1, X_2, \dots X_n)$$

dove:

- Z, X_1, X_2, \dots, X_n sono registri (o celle di memoria).

con il significato di:

"Calcola la funzione f con i dati contenuti nei registri X_1, X_2, \dots, X_n e trasferisci il risultato nel registro Z ."

Linguaggio RTL

Oltre alle istruzioni base possono essere presenti:

- Dichiarazioni strutturali del tipo:

declare register A(0:7), B(0:7), COUNT(0:2)

A(0:7), B(0:7) sono registri a 8 bit
COUNT(0:2) è un registro a 3 bit.

- Istruzioni standard di controllo:

if COUNT ≠ 7 then goto ADD

ADD è l'etichetta di una particolare istruzione del programma

Esempio di un programma RTL

*A(0:7), M(0:7), Q(0:7), COUNT(0:2),
INBUS(0:7), OUTBUS (0:7)*

BEGIN: A ← 0, COUNT ← 0

*INPUT: M ← INBUS;
Q ← INBUS;*

ADD: A(0:7) ← A(1:7) + M(1:7) × Q(7);

SHIFT: A(0) ← 0; A(1:7).Q ← A.Q(0:6);

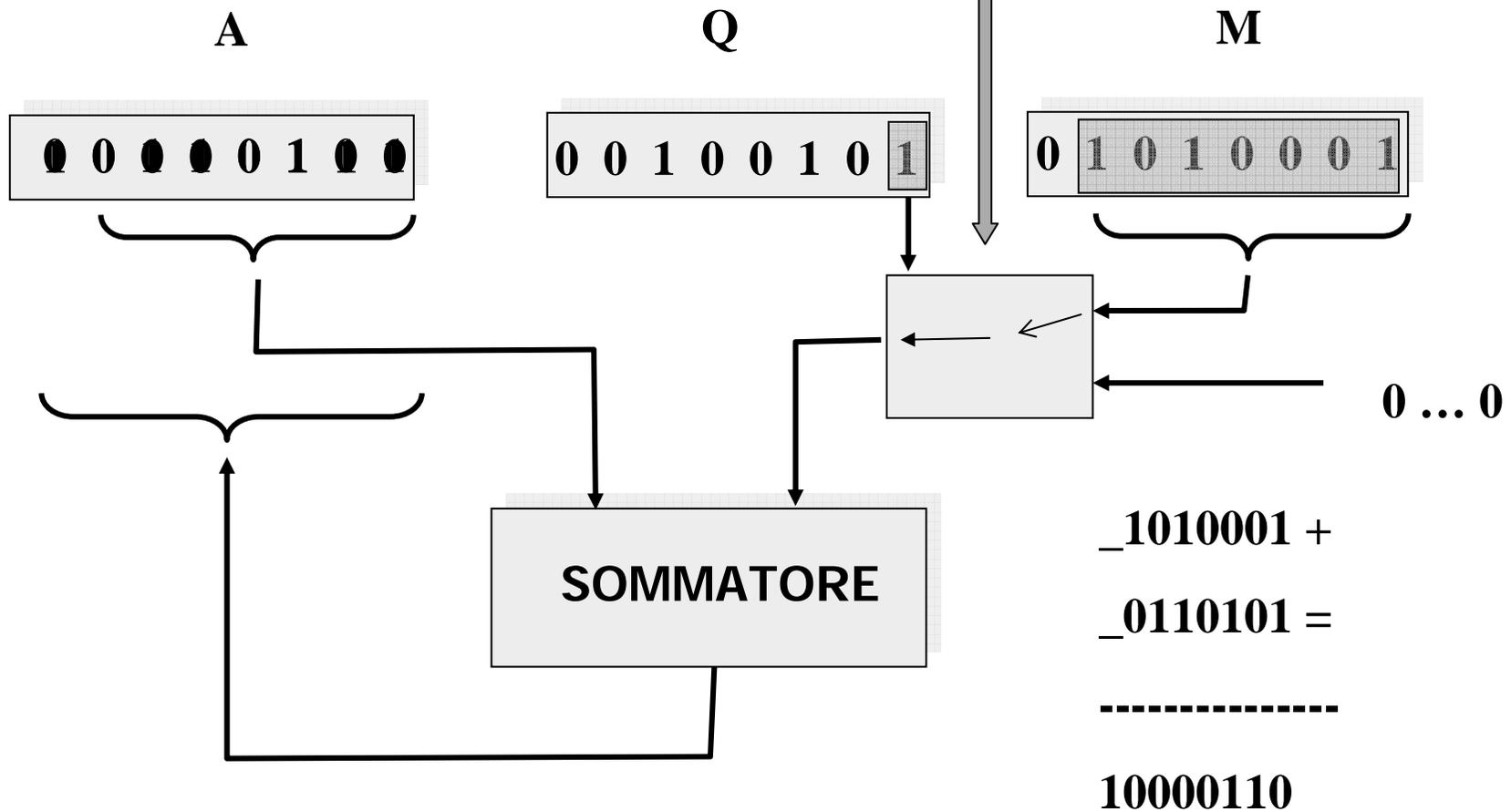
*TEST: COUNT ← COUNT + 1;
if COUNT ≠ 7 then go to ADD;*

FINISH: A(0) ← M(0) ⊕ Q(7), Q(7) ← 0;

*OUT: OUTBUS ← Q;
OUTBUS ← A;*

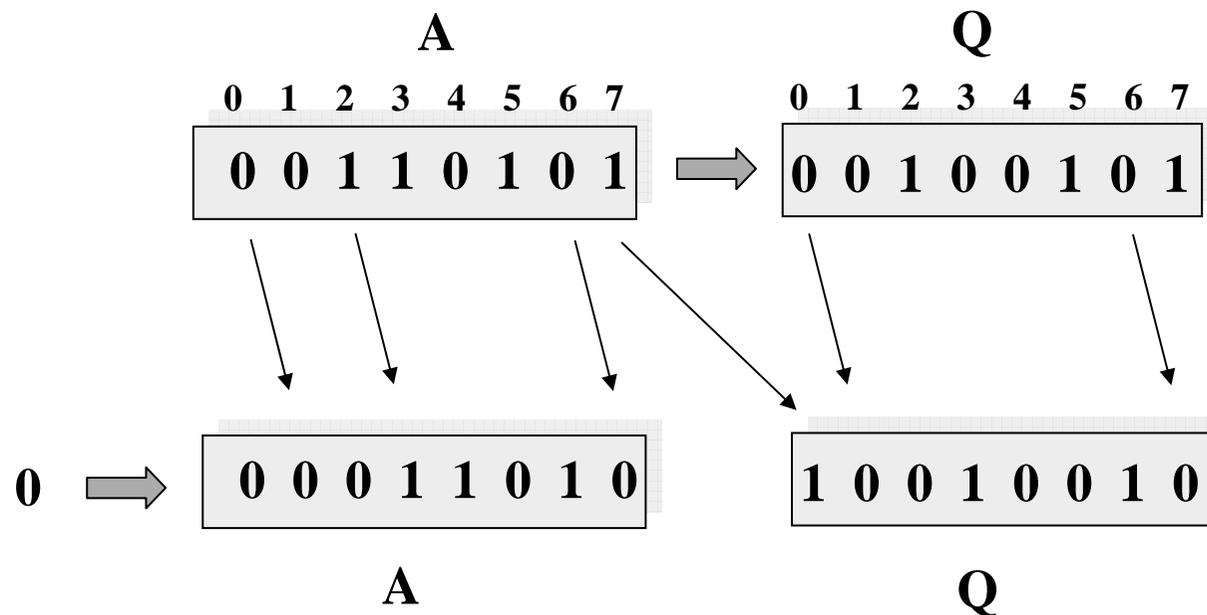
Significato di alcune istruzioni

ADD: $A(0:7) \leftarrow A(1:7) + M(1:7) \times Q(7);$



Significato di alcune istruzioni

SHIFT: $A(0) \leftarrow 0; A(1:7).Q \leftarrow A.Q(0:6);$



E' una operazione di scorrimento. I due registri da 8 bit sono collegati in modo da formare un unico registro a 16 bit

Comportamento

- Una macchina RTL definisce con i suoi moduli e con i relativi collegamenti un insieme di possibili operazioni elementari che può svolgere sui dati.
- Ogni operazione elementare a livello RTL è un trasferimento del tipo:

$$Z \Leftarrow f (X_1, X_2, \dots X_n)$$

- Un algoritmo è definito da una sequenza di operazioni elementari.
- La sequenza di operazioni elementari definisce il comportamento del sistema.

Fasi di progetto I

- Definire l'algoritmo come sequenza S di operazioni RTL.
- Analizzare S per individuare l'insieme minimo di componenti necessari.
- Costruire il diagramma a blocchi D della unità di elaborazione per realizzare tutti i necessari cammini per i dati.
- Analizzare D e S per introdurre tutti i punti di controllo necessari.
- Progettare l'unità di controllo.
- Effettuare eventuali minimizzazioni.

Fasi di progetto II

- Progettazione fisica
 - Selezione blocchi RTL
 - Collegamento blocchi RTL
 - Definizione dei punti di controllo

- Progettazione comportamentale
 - Definizione dell'algoritmo
 - Implementazione in linguaggio RTL
 - Generazione dell'unità di controllo (vettori di bit)

Frazioni binarie

Il numero N rappresentato da: $X_M = x_0 x_1 \dots x_n$

Vale: $N = (-1)^{x_0} \sum_{i=1}^n x_i 2^{-i}$

Rappresentazione in modulo e segno (x_0).

Numero maggiore: 01111111 $\Rightarrow 1 - (1/128)$

Numero minore: 11111111 $\Rightarrow -1 + (1/128)$

Doppia rappresentazione dello zero: 10000000 e
00000000

Ad esempio ($n=7$):

01000000 $\Rightarrow (+ 1/2) = 0,5$

11001000 $\Rightarrow (-[1/2 + 1/16]) = -0,5625$

I numeri utilizzati in seguito sono frazioni binarie.

Moltiplicatore binario II

Algoritmo:

segno $p_0 = x_0 \oplus y_0$

Ripetere per i da 0 a 6 $P_i \leftarrow P_i + x_{7-i} Y_M$

$$P_{i+1} \leftarrow 2^{-1} P_i$$

- Questo algoritmo coincide con quello utilizzato per il prodotto di due numeri in modo manuale.
- La differenza principale consiste nel sommare e accumulare i prodotti parziali invece di sommarli al termine.
- Si usa un solo registro di memoria per immagazzinare la somma parziale invece di tanti registri che memorizzano tutti gli addendi.

Moltiplicatore binario III

Passo	Operaz.	Result. parziale	
0	Inizializzazione	00000000	
1	$P_0 + x_4 * Y$	10110000	$Y = 1.1011$
2	$P_1 * 2^{-1}$	01011000	$X = 0.0101$
3	$P_2 + x_3 * Y$	01011000	
4	$P_3 * 2^{-1}$	00101100	
5	$P_4 + x_2 * Y$	11011100	
6	$P_5 * 2^{-1}$	01101110	
7	$P_6 + x_1 * Y$	01101110	
8	$P_7 * 2^{-1}$	00110111	

Si deduce che, per la versione a 8 bit, sono necessari due registri a 8 bit per immagazzinare il moltiplicando e il moltiplicatore e un registro a 15 bit per il risultato.

Moltiplicatore binario IV

NECESSITA' HW

- 1 registro a 8 bit \Rightarrow Q Moltiplicatore X
- 1 registro a 8 bit \Rightarrow M Moltiplicando Y
- 1 registro a 16 bit \Rightarrow A Prodotto P
- 1 sommatore a 7 bit
- Porta EX-OR

E' possibile risparmiare utilizzando lo stesso registro per immagazzinare in tempi diversi dati diversi.

Dopo che i bit di X sono stati utilizzati per il prodotto parziale non sono più necessari.

E' possibile condividere la parte del registro per accumulare il moltiplicatore e la parte meno significativa del risultato.

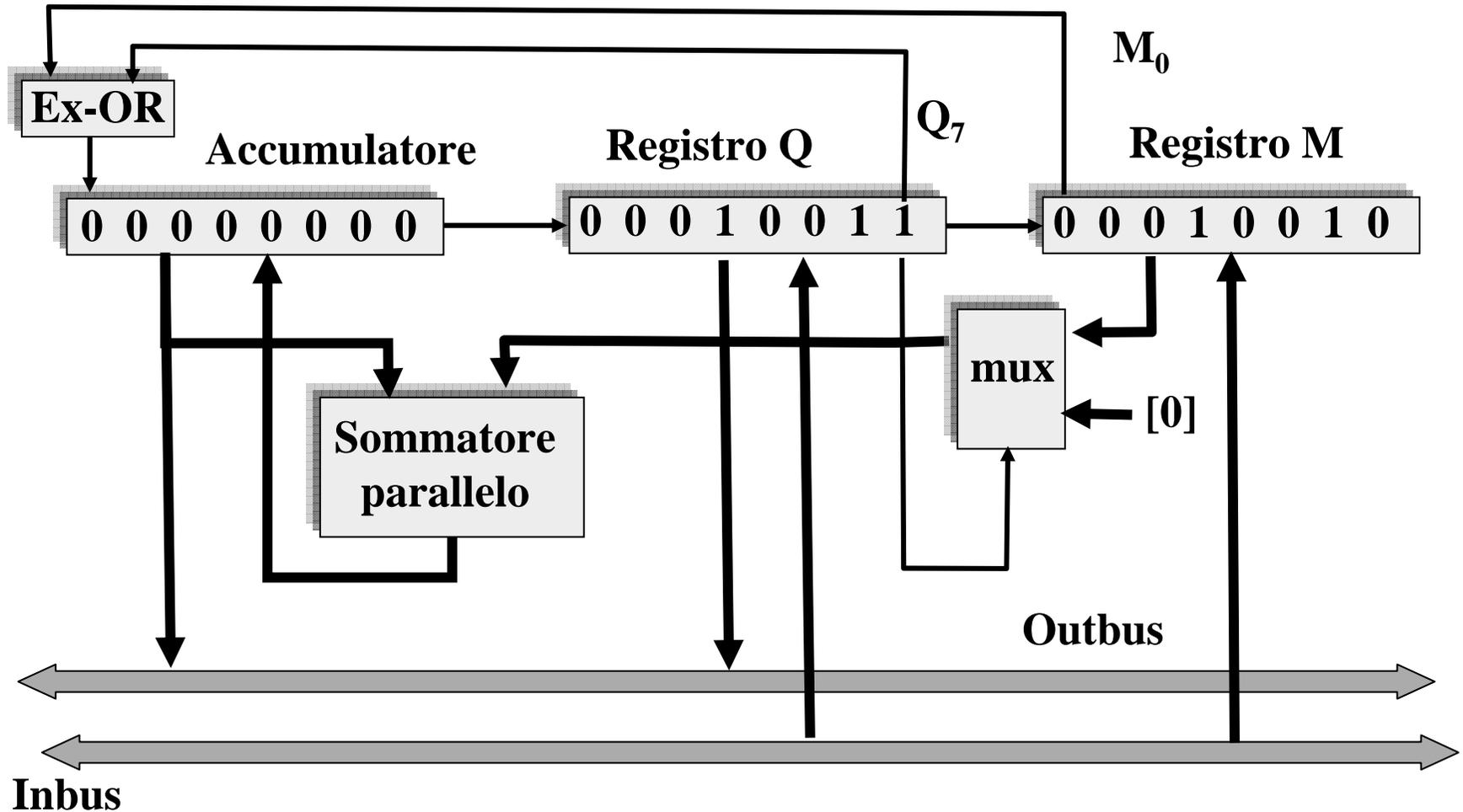
Moltiplicatore binario V

Passo	Operaz.	Accum. + Reg. Q
0	Iniz.	0000 0000
1	$P_0 + x_4 * Y$	1011 0000
2	$P_1 * 2^{-1}$	0101 1000
3	$P_2 + x_3 * Y$	0101 1000
4	$P_3 * 2^{-1}$	0010 1100
5	$P_4 + x_2 * Y$	1101 1100
6	$P_5 * 2^{-1}$	0110 1110
7	$P_6 + x_1 * Y$	0110 1110
8	$P_7 * 2^{-1}$	0011 0111

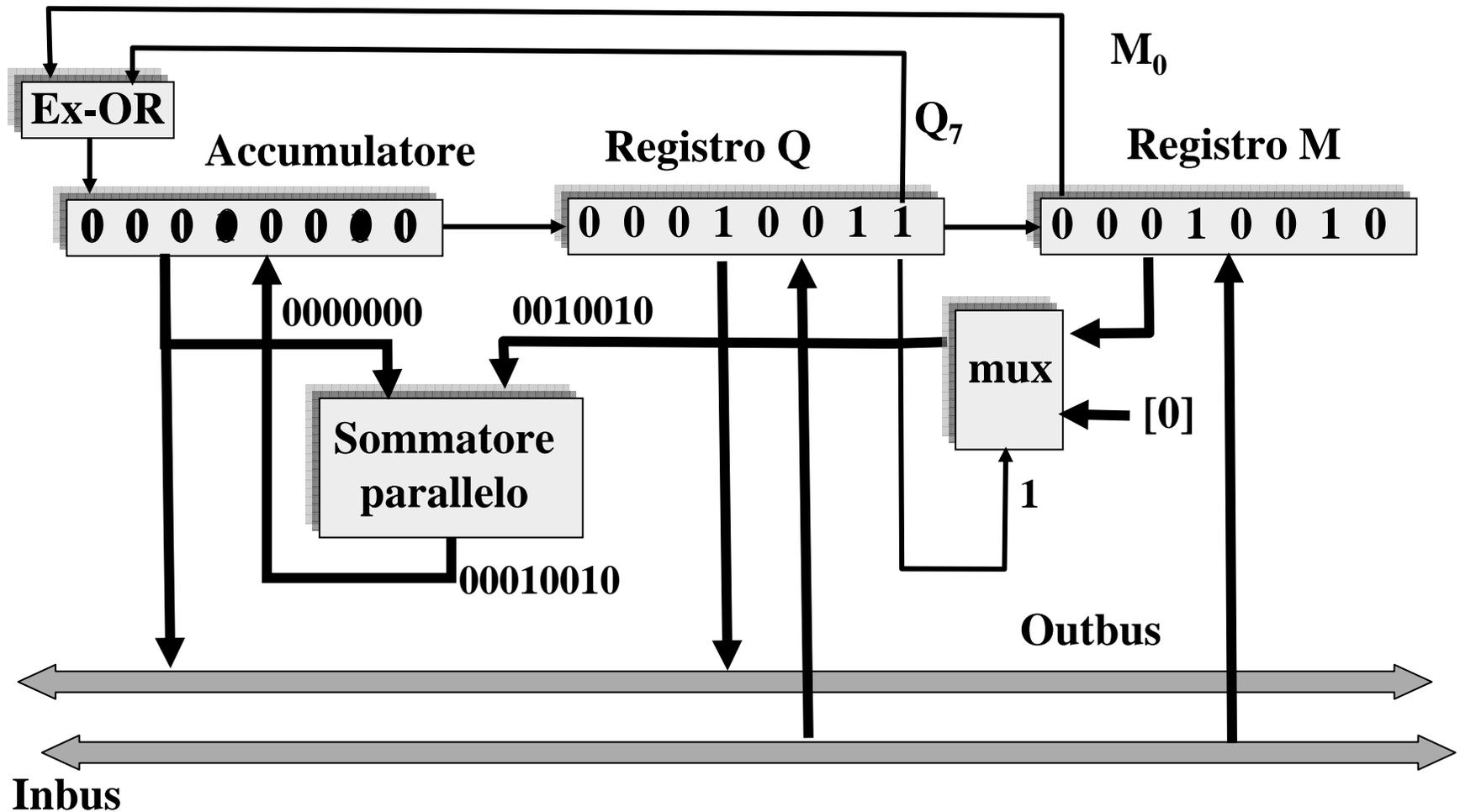
Moltiplicatore binario VI

Passo	Operaz.	Accum. + Reg. Q
0	Iniz.	0000 0000
1	$P_0 + x_4 * Y$	1011 0101
2	$P_1 * 2^{-1}$	0101 1010
3	$P_2 + x_3 * Y$	0101 1010
4	$P_3 * 2^{-1}$	0010 1101
5	$P_4 + x_2 * Y$	1101 1101
6	$P_5 * 2^{-1}$	0110 1110
7	$P_6 + x_1 * Y$	0110 1110
8	$P_7 * 2^{-1}$	0011 0111

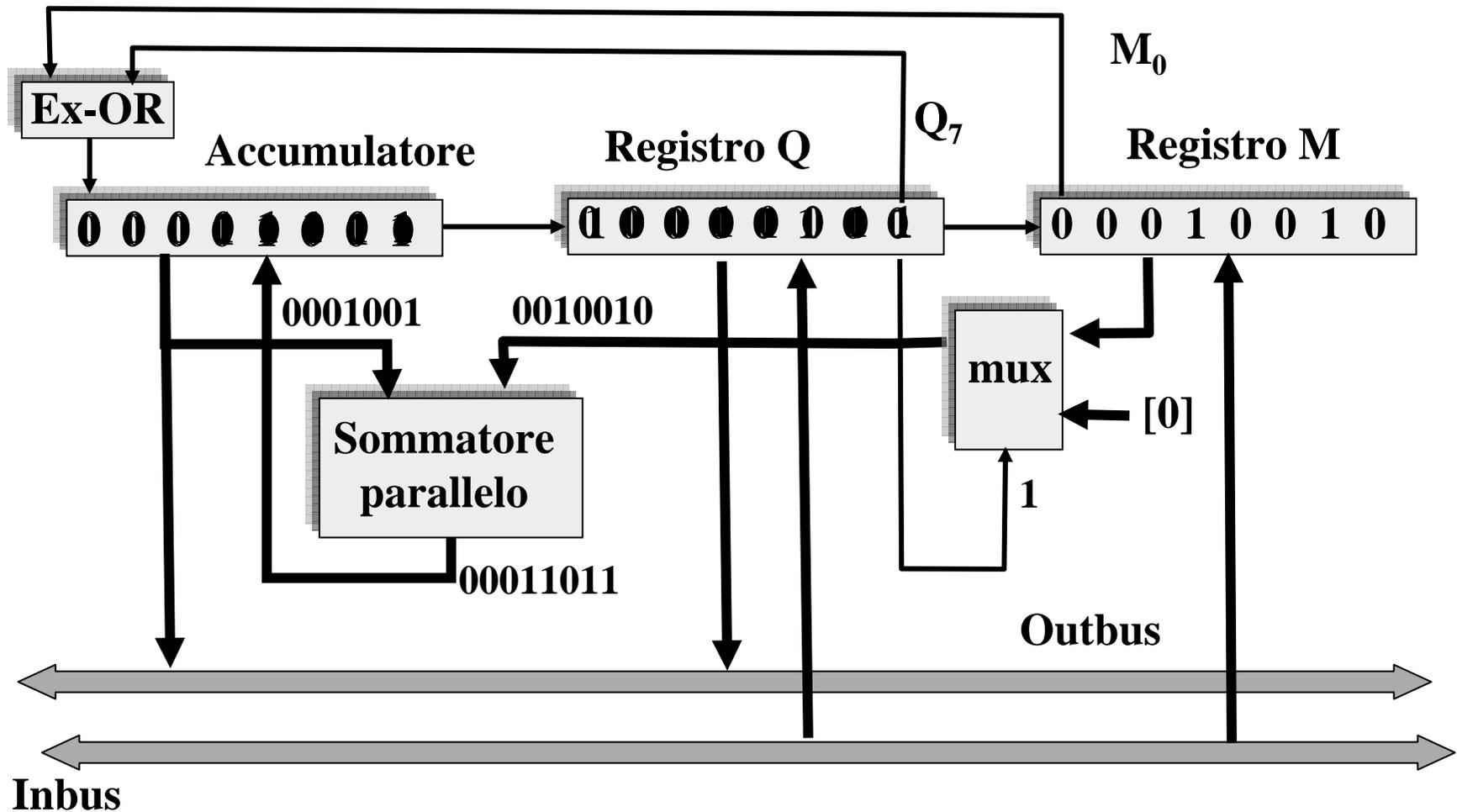
Esempio di funzionamento



Esempio di funzionamento



Esempio di funzionamento



Algoritmo del moltiplicatore

*A(0:7), M(0:7), Q(0:7), COUNT(0:2),
INBUS(0:7), OUTBUS (0:7)*

BEGIN: A ← 0, COUNT ← 0;

INPUT: M ← INBUS;
Q ← INBUS;

ADD: A(0:7) ← A(1:7) + M(1:7) × Q(7);

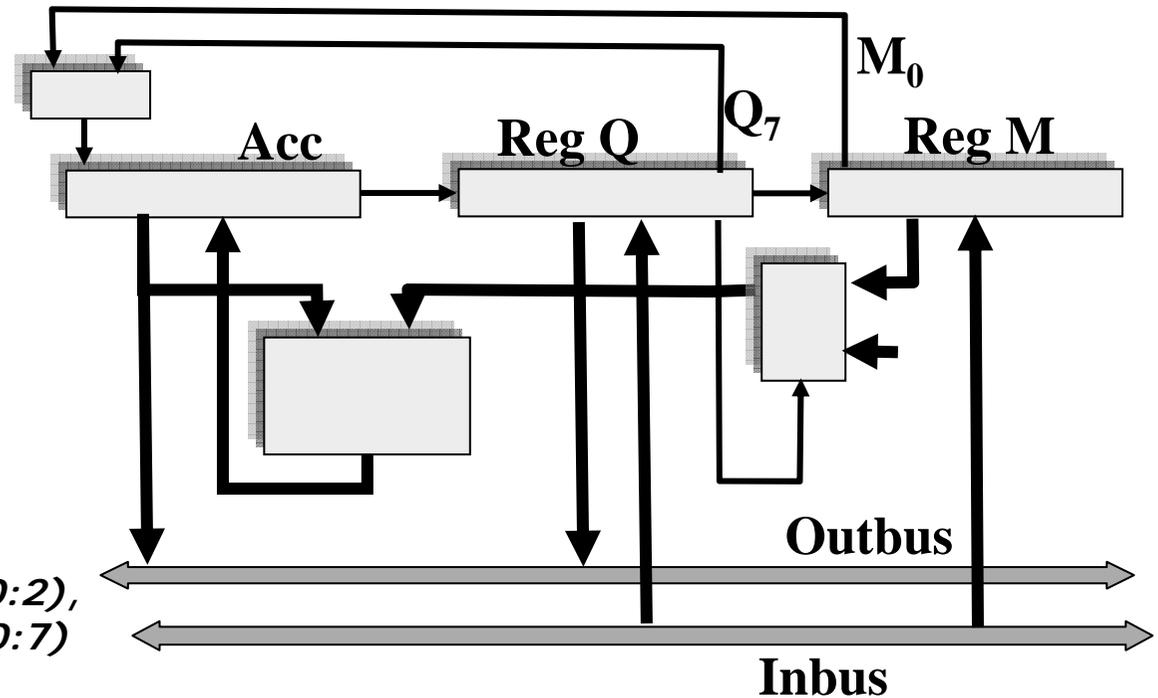
SHIFT: A(0) ← 0, A(1:7).Q ← A.Q(0:6);

TEST: COUNT ← COUNT + 1;
if COUNT ≠ 7 then go to ADD;

FINISH: A(0) ← M(0) ⊕ Q(7), Q(7) ← 0;

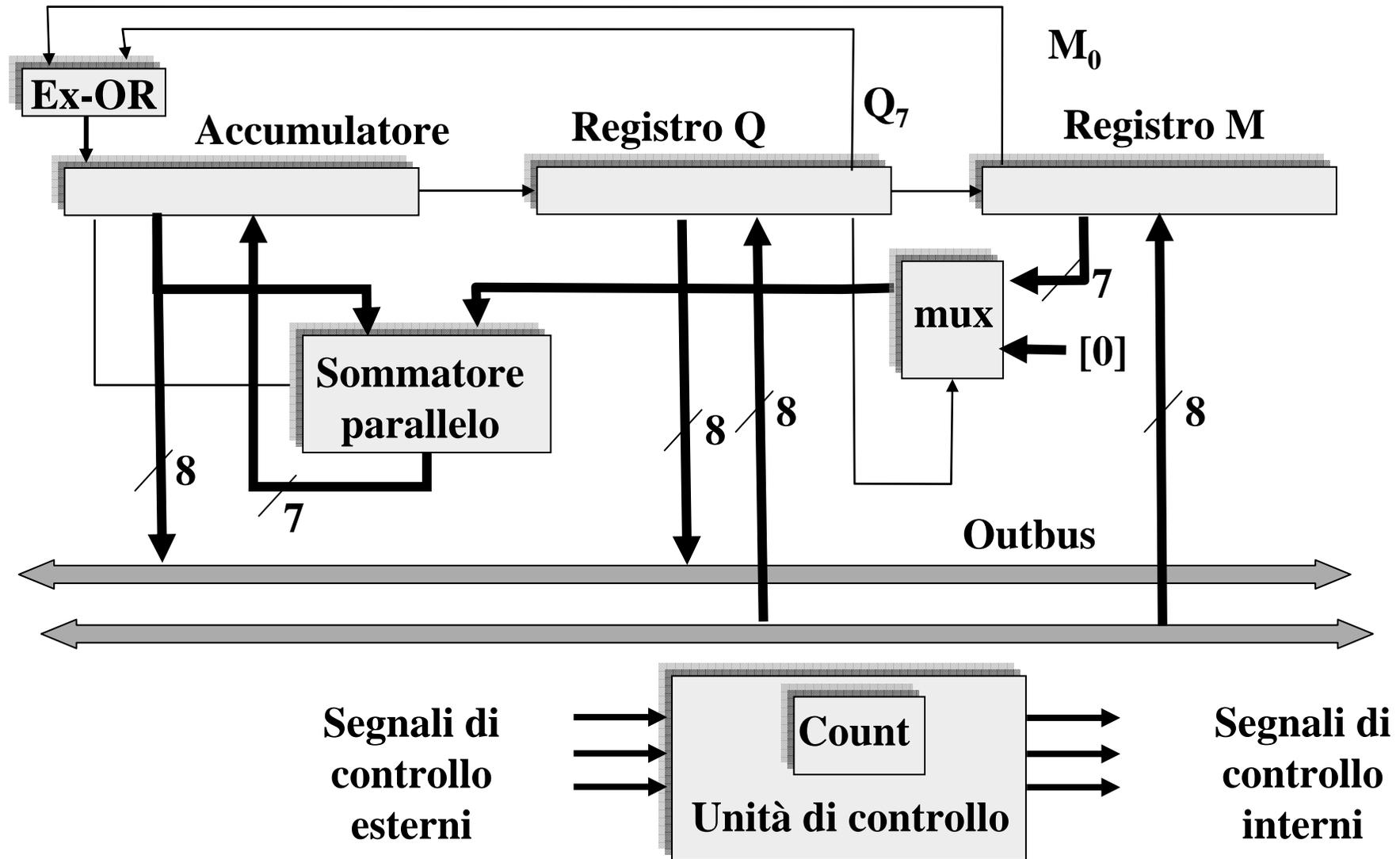
OUT: OUTBUS ← Q;
OUTBUS ← A;

Algoritmo del moltiplicatore



$A(0:7)$, $M(0:7)$, $Q(0:7)$, $COUNT(0:2)$,
 $INBUS(0:7)$, $OUTBUS(0:7)$
BEGIN: $A \leftarrow 0$, $COUNT \leftarrow 0$;
INPUT: $M \leftarrow INBUS$;
 $Q \leftarrow INBUS$;
ADD: $A(0:7) \leftarrow A(1:7) + M(1:7) \times Q(7)$;
SHIFT: $A(0) \leftarrow 0$, $A(1:7).Q \leftarrow A.Q(0:6)$;
TEST: $COUNT \leftarrow COUNT + 1$;
 if $COUNT \neq 7$ then go to **ADD**;
FINISH: $A(0) \leftarrow M(0) \oplus Q(7)$, $Q(7) \leftarrow 0$;
OUT: $OUTBUS \leftarrow Q$;
 $OUTBUS \leftarrow A$;

Segnali di controllo I



Segnali di controllo II

Segnali	Operazioni
C_0	$A \leftarrow 0$
C_1	$COUNT \leftarrow 0$
C_2	Carica $A(0)$
C_3	$M \leftarrow INBUS$
C_4	$Q \leftarrow INBUS$
C_5	$A(1:7) \leftarrow ADDER$
C_6	$INGRESSO\ ADDER \leftarrow M \circ 0$
C_7	scorrimento a destra di $A.Q$
C_8	Incrementa $COUNT$
C_9	$A(0) \leftarrow c_{out} \circ M(0) \oplus Q(7)$
C_{10}	$Q(7) \leftarrow 0$
C_{11}	$OUTBUS \leftarrow A$
C_{12}	$OUTBUS \leftarrow Q$

Segnali di controllo III

