Un’unità firmware U riceve *contemporaneamente* da una seconda unità firmware Uc gli indirizzi base (inda e indb) di due vettori di numeri interi presenti in una unità Um **esterna all’unità U**, e a *porta singola*, la lunghezza dei vettori (n) e due costanti intere x e y, (quindi 5 dati) e calcola la combinazione lineare dei vettori in cui l’i-esimo elemento del vettore a (a[i]) è *sostituito* (in Um) da

a[i]-(x* b[i])+ y

Ad Um si può chiedere di effettuare letture e scritture, e Um restituisce sempre (anche) l’esito dell’operazione: EM=0 se l’operazione è stata eseguita correttamente, EM=1 viceversa. In questo caso, si deve saltare alla gestione dell’errore (da non descrivere). L’unità Uc interagisce con U utilizzando un protocollo a domanda-risposta. *U può utilizzare al massimo 4 alu.*

Si richiede la progettazione di U in modo che il **numero di cicli di clock** richiesti per il **completamento** dell’operazione esterna (cioè del calcolo - *su tutto il vettore* - della combinazione lineare richiesta) sia **minimo**.

In particolare, si richiede di calcolare la lunghezza del ciclo di clock e il tempo (in funzione di n, ta, talu, tk e tp) di completamento relativo al calcolo di una *singola* combinazione lineare.

Successivamente, si chiede di calcolare il tempo di completamento in funzione di n, ta, talu, tk e tp.

Infine, si chiede di realizzare la PO con componenti standard, e le reti combinatorie di ωPC, limitatamente agli α dei commutatori utilizzati.

Un possibile **microprogramma** è il seguente:
0.
(RDYop=0) nop, 0
// si ricevono i 5 dati da Uc, e si ordina la prima lettura, dal fondo di "a"
(=1) N-1 + INDa → IND, "read" → OP, set RDYm, N-1→ ITER, 1.
1.
// fine lavori
(ACKm,EM,ITER31=--1) reset RDYop, set ACKop, 0.
(=0--) nop, 1.
// ricevo a[i] e ordino lettura b[i]
(=10-) ITER + INDb → IND, DATAIN + Y→ TEMP, reset ACKm, set RDYm, 2.
(=11-) NOP, trattamento_eccezione.
2. *(ACKm,EM=0-)*, nop, 2.
// ricevo b[i], completo il risultato, e ordino la scrittura
(=10) TEMP-DATAIN*X → DATAOUT, ITER+INDa→IND, "write"→OP, reset ACKm, set RDYm, 3.
(=11) NOP, trattamento_eccezione.
3. *(ACKm,EM=0-)* nop, 3
// attendo esito scrittura e torno in ciclo
(=10) ITER-1 + INDa → IND, "read"→ OP, reset ACKm, set RDYm, ITER-1→ ITER, 1
(=11) NOP, trattamento_eccezione

PO
L'interfaccia d'ingresso avrà 5 registri da 32 bit (inda, indb, n, x, y) e una coppia di indicatori (RDYop, ACKop). L'interfaccia verso la memoria è (quasi) quella standard, con i registri ACKm, RDYm, IND, OP, DATAIN, DATAOUT ed EM. Inoltre, sono necessari due registri a 32 bit, ITER e TEMP, che contengono il valore dell'indice per scorrere i vettori, e il valore intermedio calcolato, rispettivamente. Infine, si useranno 4 commutatori (in ingresso ad OP, alla alu che fa sottrazioni e due commutatori in ingresso alla alu che fa addizioni per il calcolo degli indirizzi) e 4 ALU. In alternativa, si possono risparmiare tre dei commutatori usando più alu. In questo caso, l'istruzione più lentra richiederebbe 2talu (vedi sotto).

Tempi

Il numero di microistruzioni è minimo dal momento che servono, per ognuna delle iterazioni, almeno tre microistruzioni per attendere l'esito dell'operazione richiesta al sottosistema di memoria. Un'ulteriore istruzione è necessaria per ordinare la prima operazione in memoria.
Per calcolare il tempo di completamento dobbiamo calcolare la lunghezza del ciclo di clock e moltiplicarlo per il numero di microistruzioni eseguite, sommando i tempi necessari per gli accessi in memoria.

Occorrono *1 + 3*N + 1* istruzioni (la 0. poi N volte la 1. 2. e 3. ed infine una 1. che prende la prima frase). Le 3*N istruzioni centrali costano (τ+ta), per via dell'accesso in memoria.
Il ciclo di clock lo possiamo valutare calcolando i tempi per la stabilizzazione della ωPO, σPO, ωPC, σPC nella frase piu lunga.
ωPC, σPC sono ovviamente uguali per tutte le frasi. Abbiamo 4 microistruzioni (2 bit di stato) e un massimo di 3 variabili di condizionamento testate contemporaneamente. Dunque 5 ingressi per il livello AND. Abbiamo 12 frasi, quindi al più 8 ingressi per il livello OR (se sono di più codifichiamo gli zeri e neghiamo l'uscita). Questo comporta che la PC introduca al più un ritardo di 2 tp sia per la ωPC che per la σPC.
La frase più lunga dal punto di vista della PO è senz'altro la seconda frase della microistruzione 0:

N-1 + INDa → IND, "read" → OP, set RDYm, N-1→ ITER, 1.

che richiede 2talu+2tk. Il calcolo delle variabili di condizionamento costa zero, essendo tutte uscite di registri.
Questo porta ad avere un ciclo di clock

$$\tau = 2tp + 2talu + 2tk + tp = 3tp+2talu+2tk$$

Dunque il tempo di completamento dell'operazione esterna sarà

$$(3N+2)\,(3tp+2talu+2tk) + (3Nta)$$

Reti combinatorie di ωPC

Abbiamo 4 commutatori, che chiamiamo K- (in ingresso alla alu che fa sottrazioni), K+1 (per il primo ingresso alla alu che calcola l'indirizzo), K+2 (per l'altro ingresso alla stessa alu), e Ko (in ingresso a OP). Possiamo trascurare le frasi (righe della tabella) che consistono in *nop* perché nei nop poniamo tutti gli α a zero, e codifichiamo gli "1". Inoltre,

possiamo trascurare anche le frasi in cui non si usano i commutatori, per lo stesso motivo. La tabella allora avrà solo 3 righe e sarà:

s1	s0	R	A	E	I	$\alpha-$	$\alpha+1$	$\alpha+2$	αo
0	1	–	1	0	–	0	1	1	0
1	0	–	1	0	–	0	1	0	1
1	1	–	1	0	–	1	0	0	0

da cui si ricavano immediatamente i circuiti per gli α.