Una unità firmware U è interfacciata con una unità Ua mediante un'interfaccia che comprende un registro in ingresso IN da 32 bit, un registro in ingresso OP da 1 bit, oltre agli indicatori per realizzare una comunicazione a domanda-risposta.
U riceve da Ua una coppia N (nel registro IN, N>1) e OP (OP=0 rappresenta la somma e OP=1 rappresenta la moltiplicazione) e successivamente, mediante il registro IN, riceve N valori interi x1, … , xN. L'unità calcola quindi x1 OP x2 OP … OP xN controllando che non vi sia overflow. Se l'operazione si conclude senza overflow, il risultato viene inviato ad una seconda unità Ub, quando lo richiede, assieme ad ESITO=0 (sempre usando una comunicazione a domanda-risposta). Se viceversa si ha un overflow, all'unità Ub viene semplicemente mandato ESITO = 1. L'unità U deve in ogni caso ricevere l'intera serie di N valori da Ua, prima di comunicare il risultato e l'esito ad Ub.

Si realizzi l'unità firmware U:

- avendo cura di minimizzare la lunghezza del ciclo di clock. Le porte logiche si stabilizzano in 1tp ed hanno al massimo 8 ingressi. Ci sono a disposizione solamente due ALU che eseguono operazioni aritmetiche. U **non** è dotata di memoria interna.
- Inoltre, si indichi il tempo (in termini di τ) necessario ad U per eseguire tutta l'elaborazione richiesta.
- Si progetti dettagliatamente la rete combinatoria di σPC.
- Indicare se si può usare il controllo residuo, e se si, dove.

Microcodice
0. // attendo che Ua mi mandi una richiesta
(RDYin = 0) nop, 0.
// se mi arriva, copio i parametri in registri locali e mi predispongo a
// ricevere gli Xi
(=1) IN -1 -> I, reset RDYin, set ACKin, 1.
1. // ricevo il primo valore, serve per inizializzare l'accumulatore
(RDYin=0) nop, 1.
// quando lo ricevo inizializzo l'accumulatore, flag di owerflow e conto
// un elemento ricevuto
(=1) IN -> ACC, 0 -> OW, I – 1 -> I, 2.
2. // se ho già ricevuto l'ultimo elemento, disponibilità a
// ricevere per Ub.
(I31, RDYin, ACKout,OP = 1 - 1 -) ACC -> OUT, OW -> ESITO, set RDYout, reset ACKout, 0.
// fine lavori ma non c'è disponibilità a ricevere da Ub, attendo
(= 1 – 0 -) nop, 2.
// non ho finito, aspetto se non c'è diponibilità dati in ingresso
(= 0 0 - -) nop, 2.
// ulteriore dato in ingresso disponibile, operazione somma
(=0 1 – 0) ACC + IN -> ACC, Overflow(ACC + IN) OR OW -> OW, I - 1 -> I, set ACKin, reset RDYin, 2.
// idem, moltiplicazione
(=0 1 – 1) ACC * IN -> ACC, Overflow(ACC * IN) OR OW -> OW, I - 1 -> I, set ACKin, reset RDYin, 2.

StrutturaPO:
Oltre ai registri delle interfacce (RDYin, RDYout, ACKin, ACKout, IN, OP, ESITO e OUT), si useranno i registri I e ACC da 32 bit, e OW da 1 bit. I riceve in ingresso l'uscita di una ALU, e quindi non ha bisogno di un commutatore, così come OUT ed ESITO. Un commutatore a due ingressi da 32 bit è necessario per ACC, e ad 1 bit per OW. Una ALU la dedichiamo a fare I-1→I oppure IN-1→I (e quindi serve un commutatore da 32 bit sull'ingresso di sinistra della ALU), e l'altra per fare ACC+IN→ ACC oppure ACC*IN→ ACC, senza bisogno di commutatore, ma dovendo scegliere se far fare + o *. I31 rappresenta il bit più significativo di I.

Minimo τ:

TωPO nullo (solo registri o indicatori a transizione di livello)

TσPO al massimo Talu+Tk1+1tp oppure Talu + Tk32, a seconda che si usi un commutatore a 32 bit realizzato da 32 commutatori da 1 bit in parallelo o ad albero. Nel seguito si considera la seconda ipotesi.

PC con 3 stati e 4 var cond (6 ingressi al livello AND) e 9 frasi, di cui 4 frasi nop, che per definizione hanno tutti i β a 0 e gli α non specificati (quindi che possono essere messi anche loro a 0), e pertanto non ifluiscono nel progetto delle reti combinatorie di PC. Inoltre, i nop fanno rimanere nello stato in cui sono utilizzati. Ciò implica che il numero massimo di "1" per ciascuna delle α o β sono al più 6 e quindi basta un unico livello OR (in ogni caso, avendo gates ad 8 ingressi, basta che gli "1" non siano più di 8; se lo fossero, si implementa il complemento della funzione desiderata e poi si nega il risultato). Il ritardo introdotto da TσPC e TωPC sarà quindi minimo (2tp). Pertanto:

$$\tau = 0 + talu + tk + 2tp + \delta = talu + tk + 3tp$$

Controllo residuo:

Le ultime due frasi della 2. possono essere implementate con una frase sola utilizzando il controllo residuo. La parte che calcola il nuovo valore dell'accumulatore e l'OR del vecchio e nuovo flag OW si può realizzare con una ALU che fa somme e moltiplicazioni il cui α sia preso da OP (lettura da registro, non introduce ritardo nell'operazione che rimane un'operazione con ritardo Talu + Tk). Diminuisco il numero delle variabili di condizionamento, ma questo non impatta sui ritardi della PC, che sono già minimi.

Progetto dettagliato della rete combinatoria di σPC:
nel seguito si presenta solamente la tabella di verità, e si lascia il disegno delle reti combinatorie per i due bit di stato allo studente (per semplicità, le righe della tabella sono nello stesso ordine delle frasi del microprogramma).

S1	S0	I31	RDYin	ACKout	OP	S1	S0
0	0	-	0	-	-	0	0
0	0	-	1	-	-	0	1
0	1	-	0	-	-	0	1
0	1	-	1	-	-	1	0
1	0	1	-	1	-	0	0
1	0	1	-	0	-	1	0
1	0	0	0	-	-	1	0
1	0	0	1	-	0	1	0
1	0	0	1	-	1	1	0