Architettura degli elaboratori A.A. 2018-2019 Prima prova di verifica intermedia – 5/11/2019
Compito Tipo A

Indicare in alto a destra su tutti i fogli ricevuti il proprio nome, cognome, numero di matricola e tipo di compito (A o B).

Progettare la seguente unità funzionale U in modo da minimizzare **la lunghezza del ciclo di clock .**
U riceve da un'unità esterna U1, prima un valore N (N<=64) e un bit OP che rappresenta un'operazione da svolgere (OP=0 moltiplicazione, OP=1 divisione intera) e successivamente N coppie di dati (tutti diversi da zero) di 32 bit, che costituiscono gli elementi a[i] e b[i] di due vettori a e b, e calcola gli elementi di un vettore c di 2N-1 elementi, secondo la regola :

```
for i=0, i<N, i++
        {for j=i, j<N, j++
                {c[i+j]= c[i+j]+ a[i ] OP b[j ]}
        }
```

I vettori a, b, e c sono memorizzati in tre memorie interne ad U, Ma ed Mb di 64 parole, Mc di 128 parole. Al termine del calcolo di c, U invia ad una terza unità, U2, il vettore c, un elemento alla volta, quando U2 è pronta a riceverlo.
Le porte logiche hanno un massimo di 8 ingressi, le ALU a disposizione richiedono un tempo pari a 6tp per ogni operazione aritmetica che svolgono, è a disposizione una sola ALU che effettua sia moltiplicazioni che divisioni intere, mentre non c'è limite al numero di ALU che fanno addizioni o sottrazioni. Le memorie sono a porta singola, e all'inizio di ogni operazione tutte le loro parole contengono 0.

Si chiede di fornire:

1) il microprogramma utilizzato
2) le risorse di stato e di calcolo usate nella PO, e le loro connessioni
3) il calcolo della lunghezza del ciclo di clock, in termini di tp
4) giustificazione della minimalità della lunghezza del ciclo di clock
5) le espressioni booleane per le α di ωPC
6) la possibilità di usare il controllo residuo, e, in caso affermativo, dove può essere usato

Partendo da uno tra i molti microprogrammi senza minimalizzazione della lunghezza del ciclo di clock, come ad esempio il seguente:

// ricezione di N ed OP da U1
0. (RDY1=0) NOP, 0
(=1) INA→ N, 0→ i, reset RDY1, set ACK1, 1
//ricezione dei vettori a e b
1. (RDY1, EQ(N,i)=0 -) NOP, 1
(=1 0) INA→ Ma[i], INB→ Mb[i], i+1→ i, reset RDY1, set ACK1, 1
(=1 1) 0 → i, 0 → j, 2
//controllo se fine calcolo o inizializzazione di c[i+j]
2. (EQ(N,i), OP=0 0) Mc[i+j]+ Ma[i]*Mb[j]→ Mc[i+j], j+1→ j, 3
(=0 1) Mc[i+j]+ Ma[i] div Mb[j]→ Mc[i+j], j+1→ j, 3
(=1 -) 0→ i, SHL(N,1)-1→ N, 4
//calcolo di c o incremento di i e inizializzazione di j
3. (EQ(N,j), OP=0 0) Mc[i+j]+ Ma[i]*Mb[j]→ Mc[i+j], j+1→ j, 3
(=0 1) Mc[i+j]+ Ma[i] div Mb[j]→ Mc[i+j], j+1→ j, 3
(=1 -) i+1→ i, i+1→ j, 2
// invio di c ad U2
4. (EQ(N,i), ACK2=- 0) NOP, 4
(=0 1) Mc[i]→ OUT, set RDY2, reset ACK2, 4
(=1 1) NOP, 0

si eliminano le variabili di condizionamento complesse (ad esempio, ponendo EQ(N,i)→ Fi in una microistruzione, e testando Fi nella successiva), in modo da avere $t\omega PO=0$, e si spezzano le operazioni che richiedono risorse di calcolo in cascata (come per esempio Mc[i+j], che richiede l'uso di una ALU e successivamente l'accesso alla memoria, per un totale di (talu+tac)tp) in due microistruzioni consecutive, in modo da ridurre $t\sigma PO$, si può ottenere il seguente

microprogramma
// ricezione di N ed OP da U1
0. (RDY1=0) NOP, 0
(=1) INA→ N, 0→ i, 0→ Fi, reset RDY1, set ACK1, 1
//ricezione dei vettori a e b
1. (RDY1, Fi=0 -) NOP, 1
(=1 0) INA→ Ma[i], INB→ Mb[i], i+1→ i, EQ(N,i)→ Fi, reset RDY1, set ACK1, 1
(=1 1) 0 → i, 0→ j, 0→ Fi, 2
//incremento di i e inizializzazione di j
2. (Fi=0) EQ(N,i)→ Fi , 0→ Fj, 3
(=1) 0→ i, SHL(N,1)-1→ N, 0→ Fi, 7
//calcolo di c
3 (OP=0) i+j→ ij, Ma[i]*Mb[j]→ temp2, 4
(=1) i+j→ ij, Ma[i] div Mb[j]→ temp2, 4

4. Mc[ij]→ temp1, 5
5. (Fj=0) temp1+ temp2 → temp3, j+1→ j, EQ(N,j)→ Fj, 6
(=1 -) i+1→ i, i+1→ j, 2
6. temp3→ Mc[ij], 3
// invio di c ad U2
7. (ACK2,Fi=0 -) NOP, 7
(=1 0) Mc[i]→ OUT, EQ(N,i)→ Fi , set RDY2, reset ACK2, 7
(=1 1) Mc[i]→ OUT, EQ(N,i)→ Fi , set RDY2, reset ACK2, 0

Risorse di stato e di calcolo

L'interfaccia per comunicare con U2, oltre a RDY1 e ACK1, prevede due registri a 32 bit, INA e INB, ed un registro ad 1 bit, OP, per comunicare, rispettivamente, N ed il vettore a, il vettore b, ed il tipo di operazione da effettuare. Quella per comunicare con U2 prevede, oltre a RDY2 e ACK2, un registro a 32 bit, OUT, per inviare ad U2 il vettore c. Saranno inoltre necessari i registri i ed N, di 7 bit, e j, di 6 bit, tutti dotati di un commutatore sugli ingressi: quello di i per selezionare tra 0 ed i+1, quello di N per selezionare tra INA o 2N-1, e quello di j per selezionare tra 0, j+1 e i+1 (che nel seguito deve commutare tra tre ingressi, ma si possono ridurre a due considerando che il secondo ed il terzo possono essere uscite della stessa ALU). Si useranno anche i registri ad un bit Fi ed Fj, ciascuno dotato di un commutatore per selezionare tra 0 ed EQ (per evitare l'uso di variabili di condizionamento complesse). Poi, si useranno un registro ij da 7 bit (per contenere i+j), e tre registri da 32 bit, temp1, temp2 e temp3. Questi ultimi quattro registri non hanno bisogno di commutatori. Inoltre, sono utilizzate tre memorie Ma, Mb, Mc come descritte nel testo. Si useranno 2 ALU, una per addizioni e l'altra per sottrazioni, ed una ALU per moltiplicazione o divisione intera (come richiesto dal testo). Per il calcolo di 2N, si farà uno shift sinistro di una posizione: si inserisce direttamente nel registro N il suo bit i-esimo come (i+1)-esimo, si scarta il più significativo, e si inserisce uno 0 come bit meno significativo (per effettuare l'operazione più velocemente). Infine, per testare l'uguaglianza o meno di due valori, (EQ), si userà un circuito costituito da confrontatori ad un bit in parallelo, le cui uscite saranno gli ingressi di un gate OR, con uscita negata, per un totale di 3tp. Notare che non si usano mai risorse di calcolo o memorie in cascata, e le variabili di condizionamento sono tutte semplici. Questo per minimizzare la lunghezza del ciclo di clock.

Lunghezza del ciclo di clock

Il tempo di accesso alle memorie è dominato dalla lettura. Le memorie Ma ed Mb hanno al massimo $64=2^6$ parole, quindi indirizzi a 6 bit ed un bit ad 1 in ogni riga della tabella di verità del commutatore di lettura, quindi un solo livello di AND. Per l'OR, notiamo che l'uscita Z ha 64 valori ad 1, e quindi bastano due livelli di gate OR, con taa e tab (i due tempi di accesso)=3tp. Per Mc, invece, avendo $128=2^7$ parole, saranno necessari 3 livelli di gate OR, con tac=4tp.

Per la parte controllo, notiamo che ci sono 8 microistruzioni (quindi i bit di stato sono 3), e 5 variabili di condizionamento, di cui al più 2 testate contemporaneamente. Inoltre, abbiamo 16 frasi: se i bit ad "1" di una uscita sono più di 8, codifichiamo gli "0" e complementiamo il risultato. Pertanto, sia per ωPC che per σPC bastano un livello di AND ed uno di OR, che porta ad avere tωPC

=tσPC=2tp. Non ci sono variabili di condizionamento complesse, e pertanto tωPO=0tp. Per quanto riguarda tσPO, la microoperazione più lenta è, ad esempio: Ma[i]*Mb[j]→ temp2, che richiede una ALU per calcolare il prodotto, preceduta da un accesso ad Ma e ad Mb (che avvengono in contemporanea), e analogamente per la divisione intera, per un totale di talu+tab=9tp. Pertanto, essendo τ=tωPO+max{(tωPC+tσPO),tσPC}+δ, si ha che:

$$\tau = 0tp+2tp+9tp+1tp=12tp$$

Minimalità della lunghezza del ciclo di clock

Ovvia perché tωPO è zero, tωPC assume il valore minimo, e per tσPO meno di talu+tab è impossibile (accesso alla memoria e successiva operazione aritmetica è il minimo per il problema da risolvere).

Espressioni booleane per le α di ωPC

La tabella di verità per le α sarà (non si considerano le frasi "NOP" perché in quelle si pone α=0) :

s2	s1	s0	RDY1	Fi	OP	Fj	ACK2	α_{Fi}	α_{Fj}	α_N	α_i	α_{j1}	α_{j0}
0	0	0	1	-	-	-	-	0	-	0	0	-	-
0	0	1	1	0	-	-	-	1	-	-	1	-	-
0	0	1	1	1	-	-	-	0	-	-	0	0	0
0	1	0	-	0	-	-	-	1	0	-	-	-	-
0	1	0	-	1	-	-	-	0	-	1	0	-	-
1	0	1	-	-	-	0	-	-	1	-	-	1	0
1	0	1	-	-	-	1	-	-	-	-	1	0	1
1	1	1	-	0	-	-	1	1	-	-	-	-	-
1	1	1	-	1	-	-	1	1	-	-	-	-	-

Notare che le ultime due righe possono essere fuse in una sola, mettendo "-" nella colonna di Fi. Ad esempio, si ottiene che

$$\alpha_{Fi} = \underline{s2}\ \underline{s1}\ s0\ RDY1\ \underline{Fi} + \underline{s2}\ s1\ \underline{s0}\ \underline{Fi} + s2\ s1\ s0\ ACK2$$

(dove $\underline{x}$ rappresenta la negazione di x, ed x il valore non negato). Analogamente per le altre α.

Controllo residuo

Il controllo residuo si può avere usando OP come α per la ALU che fa moltiplicazione o divisione intera.