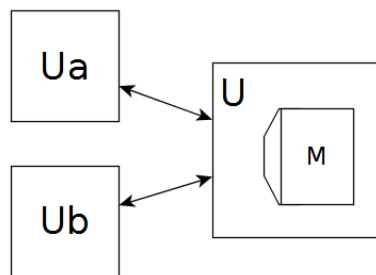


Architettura degli elaboratori – Prima prova intermedia A.A. 2015/16 – 2/11/2015

*Riportare su tutti i fogli consegnati Nome, Cognome, Matricola, Corso A o B.
I risultati saranno resi disponibili via WEB appena disponibili (home page docenti e/o didawiki.di.unipi.it)*

Si richiede di progettare una unità firmware U che interagisce con altre due unità firmware U_a e U_b . U mantiene al suo interno una memoria M di 2K posizioni (parole da 32 bit) e implementa tre distinte operazioni esterne:

- scrive un valore X nella memoria all'indirizzo IND
- cerca un valore X nella memoria e ne sostituisce tutte le eventuali occorrenze con $X/8192$ (si assume che tutte le posizioni di memoria siano significative ai fini della ricerca)
- calcola il minimo fra tutti gli elementi contenuti nella memoria e lo restituisce come risultato dell'operazione esterna



Le prime due operazioni vengono richieste solo dall'unità U_a , la terza solo dall'unità U_b . U_a ed U_b interagiscono con U secondo un protocollo a domanda risposta.

Si richiede di progettare l'unità U e di fornirne il tempo medio di elaborazione, sotto le seguenti assunzioni:

- siano disponibili porte logiche AND e OR da al più 4 ingressi
- il ritardo introdotto da ALU che implementano operazioni standard sia $t_{alu}=6t_p$
- siano disponibili unicamente ALU che eseguono sottrazione e addizioni fra parole da 32 bit, quindi con un α di controllo da 1 bit
- il tempo di accesso alla memoria M sia $t_a=10t_p$
- il numero di microistruzioni da eseguire per ciascuna delle operazioni esterne sia minimo.

Traccia di soluzione

Interfacce

Con Ua:

- in ingresso X (32 bit), IND (11 bit), RDY1 (indicatore a transizione di livello), OP (1 bit);
- in uscita ACK1 (indicatore a transizione di livello)

Con Ub:

- in ingresso RDY2 (indicatore a transizione di livello)
- in uscita ACK2 (indicatore a transizione di livello), OUTMIN (32 bit)

Microprogramma

Prima versione: diamo precedenza alle richieste di Ua (arbitrariamente). Utilizzando variabili di condizionamento complesse possiamo scrivere il codice come segue:

0. (RDY1, RDY2, OP = 00-) nop, 0;
(=1-0) X → MEM[IND], reset RDY1, set ACK1, 0;
(=1-1) 0 → I, 1;
(=01-) MEM[0] → MIN, 1 → I, 2
1. (I0, zero(MEM[I]-X)=00) I+1 → I, 1;
(=01) SHIFTR(MEM[I],13) → MEM[I], I+1 → I, 1;
(=1-) reset RDY1, set ACK1, 0;
2. (I0, segno(MIN-MEM[I]=00) I+1 → I, 2;
(=01) MEM[I] → MIN, I+1 → I, 2;
(=1-) MIN → OUTMIN, reset RDY2, set ACK2, 0

Il numero di microistruzioni eseguite è minimo. Per ciclare su N posizioni servono almeno N istruzioni più una istruzione per l'inizializzazione (o la finalizzazione) del calcolo. Questo accade per le operazioni esterne 2 e 3, mentre per l'operazione esterna 1 si esegue un'unica microistruzione.

Classi di registri

- OUTMIN: vi si scrive solo MIN, direttamente, serve BETAoutmin
- I: si inizializza e si incrementa, serve un commutatore in ingresso, con relativo ALPHA_kI, e un BETA_i
- MEM: indirizzata con IND, 0 e I. Serve una memoria a doppia porta visto che MEM[I] è utilizzato per le variabili di condizionamento. Sul primo indirizzo serve un commutatore per scegliere fra IND e 0, dunque serve un ALPHA_{ind}. Vi si scrive X o il risultato dello shift, quindi serve un commutatore sugli ingressi con relativo ALPHA_kM. Serve un BETA_m per regolare le scritture.
- MIN: si inizializza e si aggiorna o con le due uscite della memoria, dunque serve un commutatore con relativo ALPHA_kmin. Serve un BETA_{min}.
- Per RDY1, ACK1, RDY2, ACK2 servono i relativi BETA per set e reset.

Nel microprogramma ci sono 10 frasi e vengono testate al più 3 variabili di condizionamento. Servono 2 bit di stato perché abbiamo 3 microistruzioni. Dunque gli ingressi per il livello AND della sigma e omega PC sono 5. Gli ingressi per il livello OR sono sempre 5 (10 frasi, al più 5 "1" (altrimenti si codificano gli zeri negando l'uscita). Con porte da 4 ingressi a disposizione servono due livelli di porte AND e due livelli di porte OR. Il ritardo delle reti sigma e omega PC sarà dunque di 4tp.

Costo delle singole frasi (omega PO e sigma PO)

0.0	0, 0
0.1	0, tk + ta (i commutatori in ingresso e sugli indirizzi stabilizzano in parallelo)
0.2	0, tk
0.3	0, tk + ta + tk
1.0	ta + talu, tk
1.1	ta+talu, tk + ta
1.2	ta+ talu, 0
2.0	ta + talu, tk
2.1	ta + talu, tk + ta + tk
2.2	ta + talu, tk

Calcolo del ciclo di clock

Il ciclo di clock è calcolato come quello necessario per completare la frase più lunga, cioè la 2.1

$$\tau = ta + talu + \max\{4tp, 4tp+tk+talu+tk\} + tp = 10tp + 6tp + 4tp+2tp+6tp+2tp + tp = 31tp$$

Calcolo del tempo medio di elaborazione

Per le operazioni esterne:

- k = 1 per la prima operazione esterna
- k = 1 + 2K per la seconda e per la terza

Non essendo specificato nulla riguardo la frequenza con cui vengono richieste le operazioni esterne, assumiamo che siano equiprobabili ed otterremo

$$T = \tau/3 + (1 + 2K)\tau/3 + (1 + 2K)\tau/3 = \tau/3 + (1 + 2K)2\tau/3 \approx 4K\tau/3$$

Seconda versione: arbitraggio fair.

Si mantiene una variabile turno per dare precedenza, in caso di richiesta di operazione da parte di entrambe le unità Ua e Ub, all'unità che non ha richiesto l'ultima operazione eseguita.

Assunto che tutti i registri siano inizializzati a zero, utilizziamo TURNO per decidere a chi tocca in caso di richieste multiple: TURNO = 0 → precedenza a Ua, TURNO = 1 → precedenza a Ub.

Il microcodice è quasi identico a quello precedente, a parte il fatto che si testa anche TURNO e si assegna TURNO nelle varie fasi:

0. (RDY1, RDY2, Op, TURNO = 00-0) nop, 0;
(=100-) X → MEM[IND], reset RDY1, set ACK1, 1 → TURNO, 0;
(=101-) 0 → I, 1 → TURNO, 1;
(=01--) MEM[0] → MIN, 1 → I, 0 → TURNO, 2
(=1100) X → MEM[IND], reset RDY1, set ACK1, 1 → TURNO, 0;
(=1110) 0 → I, 1 → TURNO, 1;
(=11-1) MEM[0] → MIN, 1 → I, 0 → TURNO, 2
1. (I0,zero(MEM[I]-X)=00) I+1 → I, 1;
(=01) SHIFTR(MEM[I],13) → MEM[I], I+1 → I, 1;
(=1-) reset RDY1, set ACK1, 0;
2. (I0,segno(MIN-MEM[I]=00) I+1 → I, 2;

(=01) MEM[I] → MIN, I+1 → I, 2;

(=1-) MIN → OUTMIN, reset RDY2, set ACK2, 0

Aumentano il numero delle frasi (ma non tanto da richiedere tre livelli OR, essendo <16 in totale) e aumenta il numero delle variabili testate nella singola frase (4) e quindi il numero degli ingressi del livello AND (6), che comunque rimane minore di 16 e quindi permette di avere due livelli di porte AND come prima. Dunque il ritardo di sigmaPC e omegaPC rimane lo stesso. OmegaPO non cambia (aggiunta di una variabili di condizionamento registro) e non cambia sigmaPO, visto che aggiungiamo unicamente assegnamenti a registro (con commutatore) in frasi che già hanno un ritardo maggiore o uguale a tk. Complessivamente quindi le valutazioni che portano al tempo medio di elaborazione sono le stesse del caso precedente.