Architettura degli Elaboratori, 2008-09

Esercitazione 2

Ogni esercitazione ha lo scopo di servire da guida per la preparazione su una specifica parte del corso. È fortemente consigliato che lo studente lavori indipendentemente all'esercitazione durante lo svolgimento di tale parte a lezione e prima che sia disponibile la soluzione, approfondendo criticamente i vari aspetti e accompagnando la soluzione con adeguate spiegazioni rivolte alla comprensione ed alla esposizione dei concetti del corso.

Soluzione: venerdì 31 ottobre

Domanda 1

Si riprenda la domanda *c)* dell'Esercitazione 1. Si vuole che stessa funzionalità sia realizzata mediante una unità di elaborazione U che riceve la coppia *(J, A)* dall'unità U_M e invia i valori Z_0 e Z_1 rispettivamente all'unità U_0 e all'unità U_1. Il valore *A[J]* ricevuto identifica due operazioni esterne di U.

Realizzare U e valutarne il tempo medio di elaborazione per ognuna delle due operazioni esterne, nei seguenti due casi:

1. l'operazione esterna con *A[J] = 0* è realizzata con un algoritmo iterativo il cui numero di cicli di clock sia proporzionale al numero di bit di A,
2. l'operazione esterna con *A[J] = 0* è eseguita esattamente in un ciclo di clock,

e confrontare le due realizzazioni dai punti di vista che si ritiene significativi.

É noto il ritardo t_p di una porta logica con al massimo 8 ingressi. Una ALU ha ritardo uguale a $5t_p$.

Domanda 2

Spiegare il seguente concetto:

- la formula del Cap. IV sez. 2.6, per determinare il ciclo di clock, fornisce, in generale, una valutazione del caso peggiore, ma con un procedimento la cui complessità è indipendente dal numero di frasi del microprogramma.

Verificare questo concetto su un esempio opportuno.

Domanda 3

Una unità di elaborazione U ha al suo interno due memorie, *A* e *B*, di capacità *N* = 128K parole (32 bit). U riceve dall'unità U_0 coppie *(op, P)*, con *op* di 1 bit e *P* di 7 bit, e dall'unità U_1 valori *X* di 32 bit, e invia in uscita valori *Z* di 18 bit. Le operazioni esterne sono così definite:

op = 0: considerando lo spazio di *A* come organizzato logicamente in 128 blocchi, ognuno di 1K parole a indirizzi consecutivi, modifica il blocco di *A* identificato da *P* con i valori delle prime 1K parole ricevute da U_1;

op = 1: modifica *B* secondo la seguente funzione:

$$\forall i = 1 \ldots N-2: \qquad B[i] = \min(B[i], A[i-1], A[i+1])$$

e *Z* assume il valore uguale al numero di volte che *B[i]* rimane inalterato.

Valutare il tempo medio di elaborazione di U, supponendo che le due operazioni esterne sia equiprobabili.

É noto il ritardo t_p di una porta logica con al massimo 8 ingressi. Una ALU ha ritardo uguale a $5t_p$. Le memorie *A* e *B* sono realizzate partendo da componenti logici memoria aventi capacità 4K parole e tempo di accesso $4t_p$.

Soluzione

Domanda 1

Come nell'Esercitazione 1, sia S il registro, inizializzato a zero, che ricorda il valore precedente di A.

Nella versione 1 la funzione *potenza* è implementata come un ciclo *while* che, ad ogni iterazione, testa un bit della parola (numero assunto positivo); la terminazione si ha se si sono incontrati due 1, oppure se è stata esaminata tutta la parola. Usiamo un registro di un bit, POTENZA, inizializzato a 0. Per il controllo del ciclo si usa un registro I di 6 bit. Nella versione 2 la funzione *potenza* viene realizzata utilizzando direttamente la rete combinatoria dell'Esercitazione 1: questo permette di rispettare la specifica di avere un tempo di elaborazione di un solo ciclo di clock.

In entrambe le versioni, la funzione *compreso* viene realizzata utilizzando direttamente la semplice rete combinatoria dell'Esercitazione 1. Le funzioni *massimo* e *minimo* sono realizzate anch'esse in un solo ciclo di clock usando la variabile di condizionamento *segno (A – S)*; in alternativa, per il calcolo del massimo e minimo si sarebbe potuto utilizzare controllo residuo, utilizzando *segno (A – S)* come variabile di controllo e risparmiando una variabile di condizionamento.

Un'altra variabile di condizionamento complessa, che *deve* essere usata per rispettare le specifiche, è *A[J]*. Useremo quindi anche $S[I_m]$ come variabile di condizionamento per la prima operazione esterna.

Indipendentemente dalle specifiche sull'implementazione di *potenza*, in questo problema l'uso di variabili di condizionamento complesse permetterebbe comunque di minimizzare il tempo medio di elaborazione.

Nei microprogrammi useremo simboli auto esplicativi per gli indicatori di interfaccia.

Si noti che, in presenza di operazioni che, come quella per *A[J] = 0*, inviano i risultati su interfacce diverse in tempi diversi, non ha importanza l'ordine con cui ciò avviene e se vengano inviati contemporaneamente o meno; sceglieremo di inviare ogni risultato il prima possibile.

Versione 1

Microprogramma:

0. (RDYM, A[J], segno (A – S), ACK0, ACK1 = 0 – – – – , 1 0 – – 0) nop. 0;

// prima operazione esterna: inizializzazione del ciclo while per il calcolo di "potenza", e calcolo immediato del massimo o del minimo //

(= 1 0 0 – 1) reset RDYM, set ACKM, A → Z1, set RDY1, reset ACK1, 0 → I, 0 → POTENZA, A → S, 1;

(= 1 0 1 – 1) reset RDYM, set ACKM, S → Z1, set RDY1, reset ACK1, 0 → I, 0 → POTENZA, A → S, 1;

// seconda operazione esterna: calcolo immediato di "compreso" e del massimo o del minimo //

(= 1 1 – 0 0, 1 1 – 01, 1 1 – 10) nop, 0;

(= 1 1 0 1 1) reset RDYM, set ACKM, S → Z1, set RDY1, reset ACK1, compreso (A) → Z0, set RDY0, reset ACK0, A → S, 0;

(= 1 1 1 1 1) reset RDYM, set ACKM, A → Z1, set RDY1, reset ACK1, compreso (A) → Z0, set RDY0, reset ACK0, A → S, 0

// prima operazione esterna: ciclo while per il calcolo di "potenza //

1. (I_0, $S[I_m]$, POTENZA, ACK0 = 0 0 – –) I + 1 → I, 1;

(= 0 1 0 –) I + 1 → I, 1 → POTENZA, 1;

(= 0 1 1 0, 1 – – 0) nop, 1;

(= 0 1 1 1) 0 → Z0, set RDY0, reset ACK0, 0;

(= 1 – – 1) POTENZA → Z0, set RDY0, reset ACK0, 0

Detto τ_1 il ciclo di clock della versione 1, il tempo medio di elaborazione delle due operazioni è dato da:

$$T_{0\text{-}1} = (m + 2)\ \tau_1$$

$$T_{1\text{-}1} = \tau_1$$

dove *m* è il numero medio di iterazioni eseguite dall'algoritmo *potenza* ($1 < m \leq 32$), il cui valore non è ulteriormente valutabile in quanto dipendente dal valore di A; è comunque noto il valore massimo.

Per semplicità di editing, in questa sede la Parte Operativa verrà descritta, invece che con un disegno, con una tabella nella quale sono riportate le caratteristiche di tutte le risorse usate (lo studente mostrerà comunque il disegno, o le sue parti più significative):

Rete combinatoria	**Operazioni (variabili di controllo)**	**Primo ingresso (variabili di controllo)**	**Secondo ingresso (variabili di controllo)**	**Note**
ALU 1	Sottrazione (nessuna)	A (nessuna)	S (nessuna)	ALU a parte per la variabile di condizionamento *segno (A – S)*
ALU 2	+ 1 (nessuna)	I (nessuna)	--	Incrementatore di I
compreso	Funzione *compreso* (nessuna)	A (nessuna)	--	Vedi Esercitazione 1
Commutatore_codice_operazione esterna	A[J]	A (J)	--	Commutatore per valutare A[J] mediante controllo residuo
Commutatore_generico_bit_di_S	$S[I_m]$	S (I_m)	--	Commutatore per valutare $S[I_m]$ mediante controllo residuo

Registro	**Ingressi**	**Variabili di controllo**
A (32 bit)	esterno	--
J (5)	esterno	--
S (32)	A	β_S
I (6)	0, alu2	α_I, β_I
POTENZA (1)	$\alpha_{POTENZA}$	$\alpha_{POTENZA}, \beta_{POTENZA}$
Z1 (32)	A, S	α_{Z1}, β_{Z1}
Z0 (1)	compreso, 0, POTENZA	$\alpha_{Z01}, \alpha_{Z02}, \beta_{Z0}$
RDYM	esterno	β_{RDYM}
ACKM	β_{ACKM}	β_{ACKM}
RDY0	β_{RDY0}	β_{RDY0}
ACK0	esterno	β_{ACK0}
RDY1	β_{RDY1}	β_{RDY1}
ACK1	esterno	β_{ACK1}

La funzione della uscite della PO è espressa da (riportando le sole variabili di condizionamento):

ω_{PO} ::

$$\begin{cases} x_0 = RDYM \\ x_1 = A[J] \\ x_2 = segno\ (A - S) \\ x_3 = ACK0 \\ x_4 = ACK1 \\ x_5 = I_0 \\ x_6 = S[I_m] \\ x_7 = POTENZA \end{cases}$$

Avendo realizzato a parte la ALU per il calcolo di x_2, è verificata la condizione necessaria per la correttezza (PO secondo il modello di Moore).

La funzione di transizione dello stato interno della PO va espressa per ogni ingresso di ogni registro. Ad esempio:

$\sigma_{PO/Z0}$ ::

in_{Z0} = ***when*** $\beta_{Z0} = 1$ *do*

case α_{Z01}, α_{Z02} ***of***

0 0 : compreso (A)

0 1 : 0

1 – : POTENZA

La Parte Controllo può essere realizzata utilizzando, per la definizione delle funzioni ω_{PC} e σ_{PC}, direttamente la struttura del microprogramma, ricavando i termini AND delle espressioni logiche delle variabili di controllo e delle variabili dello stato interno successivo dalle frasi condizionali, e mettendo in OR i termini AND di una stessa espressione logica. Indicando con *y* la variabile dello stato interno presente, si ha, ad esempio:

ω_{PC} ::

$$\begin{cases} \beta_S = \bar{y}\ RDYM\ \overline{A[J]}\ ACK1\ +\ \bar{y}\ RDYM\ A[J]\ ACK0\ ACK1 \\ \beta_{RDYM} = \beta_{ACKM} = \beta_S \\ \alpha_I = y\ \overline{I_0}\ \overline{S[I_m]}\ +\ y\ \overline{I_0}\ S[I_m]\ \overline{POTENZA} \\ \beta_I = \bar{y}\ RDYM\ \overline{A[J]}\ ACK1 + y\ \overline{I_0}\ \overline{S[I_m]}\ +\ y\ \overline{I_0}\ S[I_m]\ \overline{POTENZA} \\ \dots \end{cases}$$

σ_{PC} ::

$$Y = \bar{y}\ RDYM\ \overline{A[J]}\ ACK1 + y\ \overline{I_0}\ \overline{S[I_m]}\ +\ y\ \overline{I_0}\ S[I_m]\ \overline{POTENZA}\ +\ y\ \overline{I_0}\ S[I_m]\ POTENZA\ ACK0$$

Per la valutazione del ciclo di clock, si ha:

$$T_{\omega PC} = T_{\sigma PC} = 2\ t_p$$

$T_{\omega P0}$: i commutatori per calcolare *A[J]* e *S[I_m]* si stabilizzano in $3t_p$ in quanto la funzione OR va realizzata a due livelli di logica. Quindi, predomina il ritardo di stabilizzazione di *segno(A – S)* = T_{ALU}:

$$T_{\omega P0} = 5\ t_p$$

$T_{\sigma P0}$: per la scrittura nel registro I il ritardo di stabilizzazione è dato da:

$$T_{ALU} + T_K = 7\ t_p$$

Per la scrittura nel registro Z0 il ritardo di stabilizzazione è dato da (con numero massimo di ingressi per porta uguale a 8, la funzione *compreso* si stabilizza in $2t_p$):

$$T_{compreso} + T_K = 4\ t_p$$

Tutte le altre operazioni di trasferimento tra registri hanno ritardo nettamente minore delle precedenti. Quindi:

$$T_{\sigma P0} = 7\ t_p$$

da cui:

$$\tau_1 = T_{\omega PO} + max(T_{\omega PC} + T_{sPO}\ , T_{sPC}) + \delta = 15\ t_p$$

Il confronto con la soluzione dell'Esercitazione 1, consistente in un'unica rete sequenziale ($\tau = 10\ t_p$), ci dice che la versione 1 della realizzazione come unità di elaborazione è caratterizzata da un ciclo di clock maggiore e, soprattutto, da un tempo di elaborazione sostanzialmente superiore per la prima operazione esterna (per la seconda operazione esterna il tempo di elaborazione è confrontabile), a causa dell'implementazione iterativa della funzione *potenza*, anche se (come sempre accade) il procedimento di progettazione delle unità di elaborazione fornisce sempre una soluzione di complessità accettabile.

Versione 2

Utilizzando le rete combinatoria *potenza* (che, con al massimo 8 ingressi per porta, ha ritardo di stabilizzazione $4t_p$), si ottiene facilmente il microprogramma consistente in una sola microistruzione:

0. (RDYM, A[J], segno (A – S), ACK0, ACK1 = 0 – – – – , 1 0 – – 0) nop. 0;

// prima operazione esterna //

(= 1 0 0 – 1) reset RDYM, set ACKM, A → Z1, set RDY1, reset ACK1, potenza (A) → Z0, set RDY0, reset ACK0, A → S, 0;

(= 1 0 1 – 1) reset RDYM, set ACKM, S → Z1, set RDY1, reset ACK1, potenza (A) → Z0, set RDY0, reset ACK0, A → S, 0;

// seconda operazione esterna //

(= 1 1 – 0 0, 1 1 – 01, 1 1 – 10) nop, 0;

(= 1 1 0 1 1) reset RDYM, set ACKM, S → Z1, set RDY1, reset ACK1, compreso (A) → Z0, set RDY0, reset ACK0, A → S, 0;

(= 1 1 1 1 1) reset RDYM, set ACKM, A → Z1, set RDY1, reset ACK1, compreso (A) → Z0, set RDY0, reset ACK0, A → S, 0

La PO è una semplice rete sequenziale consistente in una rete combinatoria che collega le interfacce di ingresso alle interfacce di uscita. Questa rete combinatoria include anche la realizzazione della PC. Infatti, la PC è una *rete combinatoria* che, avendo in ingresso le variabili di condizionamento, genera i valori delle variabili di controllo necessarie per gli indicatori di interfaccia (valori dei β non sempre uguali ad 1) e per commutare i valori da scrivere in Z0 e Z1.

Complessivamente, l'unità U è realizzata come *singola rete sequenziale*.

Il tempo medio di elaborazione di entrambe le operazioni esterne vale τ_2: i valori di $T_{\omega PC}$ *e* $T_{\omega PO}$ sono gli stessi della versione 1; inoltre:

$$T_{\sigma P0} = T_{potenza} + T_K = 6\ t_p$$

Quindi:

$$\tau_2 = 14\ t_p$$

In conclusione, con la versione 2 abbiamo ottenuto, come nell'Esercitazione 1, una singola rete sequenziale con un ciclo di clock confrontabile (leggermente maggiore). Una volta note le reti combinatorie *potenza* e *compreso*, il vantaggio dell'approccio basato su unità di elaborazione è quello di offrire un procedimento standard e, in particolare, di permettere di descrivere correttamente la sincronizzazione con le altre unità.

Domanda 2

Calcolando il ciclo di clock come il massimo dei valori ottenuti separatamente per ogni frase condizionale, si otterrebbe, con un procedimento di complessità proporzionale al numero delle frasi condizionali distinte, un valore minore o uguale a quello della formula del Cap. V sez. 2.6. Infatti, calcolati frase per frase, i ritardi di stabilizzazione di una o più delle quattro funzioni di PC, PO risulteranno minori o uguali a quelli della formula che sono i massimi in assoluto su tutto il microprogramma. Inoltre, potrà accadere che la funzione σ_{PO} inizi a stabilizzarsi in parallelo alla ω_{PO}.

Ad esempio, si consideri il microprogramma:

0. (RDYIN = 0) nop, 0;
   (= 1) IN1 + 1 $\rightarrow$ A, IN2 – 1 $\rightarrow$ B, reset RDYIN, set ACKIN, 1
1. (segno (A – B), ACKOUT = – 0) nop, 1;
   (= 0 1) A $\rightarrow$ OUT, set RDYOUT, reset ACKOUT, 0;
   (= 1 1) B $\rightarrow$ OUT, set RDYOUT, reset ACKOUT, 0

Con i valori dei ritardi della Domanda 1, l'applicazione della formula dà come risultato $\tau = 13\ t_p$.

Esaminando separatamente ogni frase, sarebbero sufficienti $8\ t_p$ per la stabilizzazione della microistruzione 0, e $10\ t_p$ per la 1, quindi sarebbe sufficiente un ciclo di clock di $10\ t_p$.

Domanda 3

Per la prima operazione esterna, utilizziamo, come contatore di passi del ciclo *for*, un registro I di 11 bit inizializzato a zero e incrementato ad ogni iterazione. Per l'indirizzamento di A utilizziamo un registro IND1 inizializzato come la concatenazione del valore ricevuto P (identificatore del blocco di dati) e di una costante di dieci zeri.

Nella seconda operazione esterna, per indirizzare B (in lettura e scrittura) utilizziamo un registro J inizializzato a N–2. Realizziamo A con doppio indirizzamento, utilizzando lo stesso IND1, inizializzato a N–3, per leggere *A[j–1]* (oltre che per scrivere *X* nella prima operazione esterna), ed un registro IND2, inizializzato a N–1, per leggere *A[j+1]*. I tre registri sono decrementati in parallelo ad ogni iterazione del ciclo *for*. Così facendo riusciamo a limitare a due il numero di indirizzi distinti della memoria A permettendo, al contempo, il massimo parallelismo.

La valutazione della funzione minimo può essere basata sulla seguente proprietà:

```
min(a, b, c) = if a < b
                  then if a < c
                          then a
                          else c
               else if b < c
                          then b
                          else c
```

da realizzare come:

case segno(a – b), segno(a – c), segno(b – c) of

1 1 – : a

1 0 – : c

0 – 1 : b

0 – 0 : c

In effetti, utilizzeremo tre variabili di condizionamento complesse come uscite ausiliarie di altrettante ALU dedicate. Questa scelta è motivata dalla minimizzazione del numero di cicli di clock della seconda operazione esterna avente tempo di elaborazione molto maggiore della prima (ciclo *for* ripetuto 128K volte contro 1K volte) e stessa probabilità di occorrenza.

Il microprogramma può essere il seguente:

0. (RDY0, OP = 0 –) nop, 0;

(= 1 0) reset RDY0, set ACK0, P ∘ dieci_zeri → IND1, 0 → I, 1;

(= 1 1) reset RDY0, set ACK0, 0 → C, N–2 → J, N–3 → IND1, N–1 → IND2, 2

// prima operazione esterna //

1. (I_0, RDY1 = 0 0) nop, 1;

(= 0 1) reset RDY1, set ACK1, X → A[IND1], IND1 + 1 → IND1, I + 1 → I, 1;

(= 1 –) "nop", 0 *// nop eliminabile fondendo la microistruzione 1 con la 0 //*

// seconda operazione esterna //

2. (OR(J), segno(B[J] – A[IND1]), segno(B[J] – A[IND2]), segno(A[IND1] – A[IND2]), ACKOUT =

= 1 0 – 0 –, 0 1 0 – –) A[IND2] → B[J], J – 1 → J, IND1 – 1 → IND1, IND2 – 1 → IND2, 2;

(= 1 0 – 1 –) A[IND1] → B[J], J – 1 → J, IND1 – 1 → IND1, IND2 – 1 → IND2, 2;

(= 1 1 1 – –) C + 1 → C, J – 1 → J, IND1 – 1 → IND1, IND2 – 1 → IND2, 2;

(= 0 – – – 0) nop, 2;

(= 0 – – – 1) reset ACKOUT, set RDYOUT, C → Z, 0

Utilizzando tre ALU dedicate alle variabili di condizionamento *segno(...)* e grazie alla struttura di indirizzamento di A, la funzione delle uscite della PO è data da:

ω_{PO} ::

$$\begin{cases} x_0 = RDY0 \\ x_1 = OP \\ x_2 = I_0 \\ x_3 = RDY1 \\ x_4 = OR(J) \\ x_5 = segno(B[J] - A[IND1]) \\ x_6 = segno(B[J] - A[IND2]) \\ x_7 = segno(A[IND1] - A[IND2]) \\ x_8 = ACKOUT \end{cases}$$

che soddisfa la condizione necessaria per la correttezza (PO di Moore).

Le due operazioni esterne hanno tempo medio di elaborazione:

$$T_0 \sim 1\ K\ \tau \qquad , \qquad T_1 \sim 128\ K\ \tau$$

Quindi, il tempo medio di elaborazione di U vale:

$$T = p_0\ T_0 + p_1\ T_1 \sim 128{,}5\ K\ \tau$$

Siamo interessati a valutare il ciclo di clock τ senza studiare in dettaglio la realizzazione di PC e PO.

Per quanto riguarda PC, si vede che, nelle espressioni logiche della funzione delle uscite,

- non possono esistere termini AND con più di 8 variabili (l'upper bound è 7, avendo due variabili dello stato interno ed al più 5 variabili di condizionamento per microistruzione; comunque questo valore non viene mai raggiunto grazie alla presenza di non specificati),
- non possono esistere più di 8 termini AND per la stessa funzione (l'upper bound è 7, uguale al numero delle frasi non contenenti *nop*; comunque questo valore non viene mai raggiunto).

Quindi:

$$T_{\omega PC} = 2\ t_p$$

Anche se $T_{\sigma PC}$ risultasse superiore (e non lo è), non sarebbe comunque tale da influenzare il valore di τ.

Nella PO, ognuna delle memorie A e B è realizzata come combinazione di 32 componenti logici memoria di capacità 4K e tempo di accesso $4t_p$, ognuno indirizzato con i 12 bit meno significativi dell'indirizzo; i 5 bit più significativi dell'indirizzo controllano un commutatore per la lettura (e il selezionatore per la scrittura), avente 3 livelli di logica (un livello AND e due livelli OR). Quindi, il tempo di accesso di A e B vale

$$t_a = 7\ t_p$$

Le variabili di condizionamento *segno(…)* si stabilizzano con un ritardo:

$$T_{\omega P0} = T_{ALU} + t_a = 12\ t_p$$

Per quanto riguarda σ_{PO}, valutiamo i ritardi più significativi (i valori N–1, N–2, N–3 sono disponibili come costanti):

- avendo quattro ALU (oltre le tre per le variabili di condizionamento *segno(…)*), qualcuna ha un commutatore in ingresso, quindi per almeno uno dei registri I, J, IND1, IND2, C il ritardo di stabilizzazione vale:

  $$T_{ALU} + 2\ T_K = 9\ t_p$$

- l'operazione elementare X $\rightarrow$ A[IND1] si stabilizza con ritardo

  $$t_a = 7\ t_p$$

  in quanto non ci sono operazioni che si sovrappongono alla scrittura.

- le operazioni elementari A[IND1] $\rightarrow$ B[J] e A[IND2] $\rightarrow$ B[J] si stabilizzano con ritardo

  $$t_a + T_K = 9\ t_p$$

  in quanto è presente un commutatore all'ingresso del dato di B, e questo ritardo copre interamente quello di scrittura.

Quindi:

$$T_{\sigma PO} = 9\ t_p$$

$$\tau = 24\ t_p$$

$$T \sim 128{,}5\ K\ \tau \sim 3{,}16 \times 10^6\ t_p$$