Architettura degli Elaboratori, 2007-08

Esercitazione 2 di verifica

Soluzione: mercoledì 24 ottobre

Domanda 1

Una unità di elaborazione U è così definita:

i) possiede al suo interno due componenti logici memoria, A e B, entrambi di capacità 64K parole;

ii) riceve dall'unità U_0 messaggi (OP, J1, J2, X), dove OP è il codice operativo delle operazioni esterne, J1 e J2 sono indirizzi, e X è una parola;

iii) operazioni esterne:

1. copia il valore di X nella locazione di A di indirizzo J1;
2. copia il blocco di 256 parole consecutive di A, che inizia all'indirizzo J1, nel blocco di B che inizia all'indirizzo J2. In questa operazione esterna, è garantito che i valori di J1 e J2 inviati da U_0 siano multipli interi di 256;
3. invia all'unità U_1 il valore OUT uguale al numero di locazioni di A e B, *con lo stesso indirizzo*, aventi lo stesso contenuto.

È 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 devono essere realizzate mediante memorie di capacità 2K parole, ognuna avente tempo di accesso uguale a $5t_p$. La durata dell'impulso di clock è uguale a t_p.

a) Scrivere il microprogramma. Spiegare le scelte effettuate, ed in particolare spiegare come sono state scelte le variabili di condizionamento.

b) Mostrare la Parte Operativa. Ricavarne, e spiegare come sono state ricavate, la funzione delle uscite e la funzione di transizione dello stato interno, quest'ultima funzione relativamente soltanto ai seguenti registri: *i)* registro con il quale viene indirizzata la memoria B, *ii)* generica locazione della memoria B.

c) Ricavare, e spiegare come sono state ricavate, la funzione di transizione dello stato interno e la funzione delle uscite della Parte Controllo, quest'ultima funzione relativamente alla sola variabile di controllo per l'abilitazione alla scrittura nel registro con il quale viene indirizzata la memoria B.

d) Valutare il ciclo di clock in funzione di t_p, e ricavare il tempo medio di elaborazione di ogni operazione esterna separatamente.

Allo scopo di effettuare l'esercitazione passo passo con le lezioni sul Cap. IV, si consiglia di progettare U *prima* senza considerare la sincronizzazione nelle comunicazioni di U con U_0 e U_1, e *poi* introducendo tale sincronizzazione, fermo restando che, da questo momento in poi, andrà sempre adottata la versione con sincronizzazione.

e) Supponiamo che la Domanda sia formulata nel seguente modo: *scrivere il microprogramma e valutare il tempo medio di elaborazione*. Ciò significa che non è necessario che tutti i passi siano sviluppati completamente, ma è sufficiente che sia ricavato solo quanto strettamente indispensabile per arrivare a valutare il tempo medio di elaborazione. Riflettere su questo aspetto, partendo dal microprogramma scritto al punto *a)*.

Domanda 2

a) Spiegare perché, in generale, il risultato della funzione delle uscite di una PC dipende, ad ogni ciclo di clock, sia dallo stato interno presente che dallo stato d'ingresso della PC stessa.

b) Supponiamo che, nella realizzazione di una certa PO, la modifica del contenuto di un registro di un bit, la cui uscita è una variabile di condizionamento, sia ottenuta mediante una rete combinatoria tra i cui ingressi figurano anche variabili di controllo.

Dire se la seguente affermazione è vera o falsa, spiegando la risposta: "di conseguenza, la realizzazione della PO non soddisfa la condizione necessaria per la correttezza".

c) Scrivere una versione equivalente della seguente microistruzione che *non* faccia uso di controllo residuo:

4. $(A_0 = 0) A[I] \rightarrow B, 5; (A_0 = 1) A \rightarrow C, 6$

dove A e C sono registri di 32 bit, B di 8 bit, e I di 2 bit.

Con riferimento a questo esempio specifico, spiegare i vantaggi nell'utilizzazione del controllo residuo.

d) Si considerino le reti logiche oggetto delle *Domande 1, 2, 3 dell'Esercitazione 1*.

Ora vogliamo che le stesse funzionalità siano implementate da unità di elaborazione.

Scrivere il microprogramma di ognuna di tali unità e valutarne il tempo medio di elaborazione, confrontandolo con quello dell'Esercitazione1.

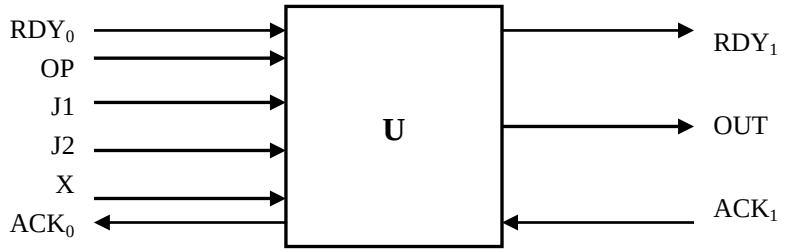
Soluzione

Domanda 1

Verrà mostrata la soluzione finale comprendente la sincronizzazione nelle comunicazioni.

a) Microprogramma

Le interfacce dell'unità U sono le seguenti:



Ogni collegamento fa capo ad un registro di interfaccia; in particolare ai collegamenti RDY e ACK corrispondono indicatori di interfaccia a transizione di livello che verranno mostrati in seguito nello schema della Parte Operativa.

Essendo le memorie A e B di capacità 64K, gli indirizzi J1 e J2 sono rappresentati su 16 bit.

Il codice operativo OP è di 2 bit, OP_0 e OP_1, che faranno parte delle variabili di condizionamento assieme a RDY_0 e ACK_1.

La codifica delle operazioni esterne è: *00* per la 1, *01* per la 2, *1 –* per la 3.

Nel microprogramma dell'operazione esterna 2, che comporta un loop ripetuto 256 volte, occorre salvare gli indirizzi J1 e J2 in due registri, INDA e INDB, per indirizzare le memorie A e B rispettivamente.

Questi registri verranno usati con lo stesso scopo anche nel microprogramma dell'operazione esterna 3. Pur se in questa sarebbe sufficiente usare uno stesso registro per indirizzare sia A che B (ad esempio, INDA), conviene comunque usare gli stessi registri nell'operazione esterna 2 e nella 3, soprattutto per ridurre il numero di possibili sorgenti per l'indirizzamento delle memorie (numero che, in generale, contribuisce ad aumentare il tempo di accesso). Nessun vantaggio si otterrebbe ad introdurre un ulteriore registro di indirizzamento da usare solo nel microprogramma dell'operazione esterna 3.

Invece, nel microprogramma dell'operazione esterna 1, che è eseguibile in un singolo ciclo di clock, per i valori di J1 e X verranno usati direttamente i registri di ingresso corrispondenti.

Per il controllo del loop nei microprogrammi delle operazioni esterne 2 e 3 verrà fatto uso di un registro I di 17 bit, inizializzato rispettivamente ai valori 255 e 64K − 1, e decrementato di uno ad ogni passo. La condizione di uscita dal loop corrisponde a $I_0 = 1$, con I_0 bit del segno, che costituisce una variabile di condizionamento. Alternativamente, si sarebbero potuti adottare due registri contatori, uno di 9 bit (operazione esterna 2) ed uno di 17 bit (operazione esterna 3), inizializzati e fatti variare secondo più soluzioni.

Sia C (17 bit) il temporaneo per il risultato dell'operazione esterna 3.

Nel microprogramma dell'**operazione esterna 3**, importante è la scelta della variabile di condizionamento per testare, ad ogni passo del loop, la condizione *if (A[INDA] = B[INDB])*. Sono possibili almeno due soluzioni. La *prima soluzione* è del tipo:

i. … zero(A[INDA] – B[INDB]) → Z, … , i+1; …

i+1. (… Z … = … 0 …) Frase 0 ; (… Z … = … 1 …) Frase 1 ; …

con Z registro di 1 bit posto sull'uscita secondaria "zero" di una ALU.

La *seconda soluzione* è del tipo:

i. (… zero(A[INDA] – B[INDB]) … = … 0 …) Frase 0 ;

(… zero(A[INDA] – B[INDB]) … = … 1 …) Frase 1 ;

Si noti che, al posto della funzione *zero(A[INDA] – B[INDB])*, si sarebbe potuto usare qualunque altra funzione equivalente per il confronto tra due numeri, ad esempio *zero(A[INDA]* $\oplus$ *B[INDB])*: in ogni soluzione, avendo a che fare con numeri rappresentati su parola, occorre sempre usare una ALU, e quindi tutte queste soluzioni hanno lo stesso impatto sul ritardo della funzione ω_{PO} (nel secondo caso) o della σ_{PO} (nel primo caso).

La seconda soluzione comporta un numero di cicli di clock complessivo uguale alla metà del numero di cicli di clock della prima, ed un ciclo di clock più lungo a causa del ritardo maggiore della funzione ω_{PO} (ritardo di stabilizzazione della ALU per valutare il predicato "zero"). Poiché presumibilmente il ciclo di clock della seconda soluzione sarà minore del doppio rispetto alla prima, il tempo medio di elaborazione *dell'operazione esterna 3* con la seconda soluzione sarà inferiore.

Non ci sono, però, informazioni per poter dire se la soluzione 2 sia migliore della 1 in assoluto, cioè *rispetto al complesso di tutte le operazioni esterne*, in quanto non sono note le probabilità di occorrenza delle singole operazioni per valutare il tempo medio di elaborazione globale.

Affinché la soluzione 2 sia *corretta*, cioè affinché la rete sequenziale PO risponda al modello matematico di Moore, occorre verificare che il valore della variabile di condizionamento *zero(A[INDA] – B[INDB])* dipenda, <u>*ad ogni ciclo di clock*</u>, soltanto dallo stato interno della PO stessa e non da valori di variabili di controllo; in particolare, devono essere rispettate due condizioni:

cond-1 venga usata una ALU capace di eseguire solo l'operazione di sottrazione ed i cui ingressi sia dati esclusivamente dalle uscite delle memorie A e B (oppure: usando una ALU multifunzione o con più ingressi, ma con variabili di controllo ottenute come uscite di registri, ove possibile);

cond-2 le memorie A e B siano indirizzate esclusivamente l'una da INDA e l'altra da INDB, oppure per l'operazione di lettura siano usati esclusivamente tali indirizzi (a meno che eventuali variabili di controllo di commutatori posti sull'indirizzo delle memorie siano ottenute come uscite di registri).

Con le scelte discusse finora, il microprogramma di U con la *seconda soluzione* è il seguente:

0. (RDY_0, OP_0, OP_1 = 0 – –) nop, 0;

(= 1 0 0) reset RDY_0, set ACK_0, X $\rightarrow$ A[J1], 0;

(= 1 0 1) reset RDY_0, set ACK_0, J1 $\rightarrow$ INDA, J2 $\rightarrow$ INDB, 255 $\rightarrow$ I, 1;

(= 1 1 –) reset RDY_0, set ACK_0, 0 $\rightarrow$ INDA, 0 $\rightarrow$ INDB, (64K – 1) $\rightarrow$ I, 0 $\rightarrow$ C, 2

1. (I_0 = 0) A[INDA] $\rightarrow$ B[INDB], INDA + 1 $\rightarrow$ INDA, INDB + 1 $\rightarrow$ INDB, I – 1 $\rightarrow$ I ,1;

(= 1) "nop", 0

// questa "nop" è una solo abbreviazione: può essere eliminata fondendo la microistruzione 0 nella 1; in ogni caso, l'impatto sulle prestazioni è trascurabile //

2. (I_0, zero(A[INDA] – B[INDB]), ACK_1 = 0 0 –) INDA + 1 $\rightarrow$ INDA, INDB + 1 $\rightarrow$ INDB, I – 1 $\rightarrow$ I, 2;

(= 0 1 –) C + 1 $\rightarrow$ C, INDA + 1 $\rightarrow$ INDA, INDB + 1 $\rightarrow$ INDB, I – 1 $\rightarrow$ I, 2;

(= 1 – 0) nop, 2;

(= 1 – 1) reset ACK_1, set RDY_1, C $\rightarrow$ OUT, 0

Oltre alla due soluzioni descritte, è possibile una *terza soluzione*, basata su

- inizializzazioni e terminazioni opportune delle variabili,
- adeguato parallelismo tra microoperazioni e condizioni logiche,

tendente ad ottimizzare sia il numero di cicli di clock che la lunghezza del ciclo di clock.

In particolare, è possibile scrivere un microprogramma caratterizzato (con ottima approssimazione) dallo stesso ciclo di clock della prima soluzione e dallo stesso numero di cicli di clock della seconda, quindi tale da ottimizzare le prestazioni *e rendere le scelte di progetto praticamente indipendenti dalle probabilità di occorrenza delle operazioni esterne*:

0. (RDY_0, OP_0, OP_1 = 0 – –) nop, 0;

   (= 1 0 0) reset RDY_0, set ACK_0, X → A[J1], 0;

   (= 1 0 1) reset RDY_0, set ACK_0, J1 → INDA, J2 → INDB, 255 → I, 1;

   (= 1 1 –) reset RDY_0, set ACK_0, **1 → INDA, 1 → INDB, (64K – 2) → I,** 0 → C, **zero(A[0] – B[0]) → Z**, 2

1. (I_0 = 0) A[INDA] → B[INDB], INDA + 1 → INDA, INDB + 1 → INDB, I – 1 → I ,1;

   (= 1) "nop", 0

2. (I_0, Z, ACK_1 = 0 0 –) INDA + 1 → INDA, INDB + 1 → INDB, I – 1 → I, **zero(A[INDA] – B[INDB]) → Z**, 2;

   (= 0 1 –) C + 1 → C, INDA + 1 → INDA, INDB + 1 → INDB, I – 1 → I, **zero(A[INDA] – B[INDB]) → Z**, 2;

   (= 1 – 0) nop, 2;

   (= 1 0 1) reset ACK_1, set RDY_1, C → OUT, 0;

   (= 1 1 1) reset ACK_1, set RDY_1, **C + 1 → OUT**, 0

Per ragioni didattiche, d'ora in poi ci riferiremo alla seconda soluzione che contiene un maggior numero di spunti di riflessione.

b) Parte Operativa

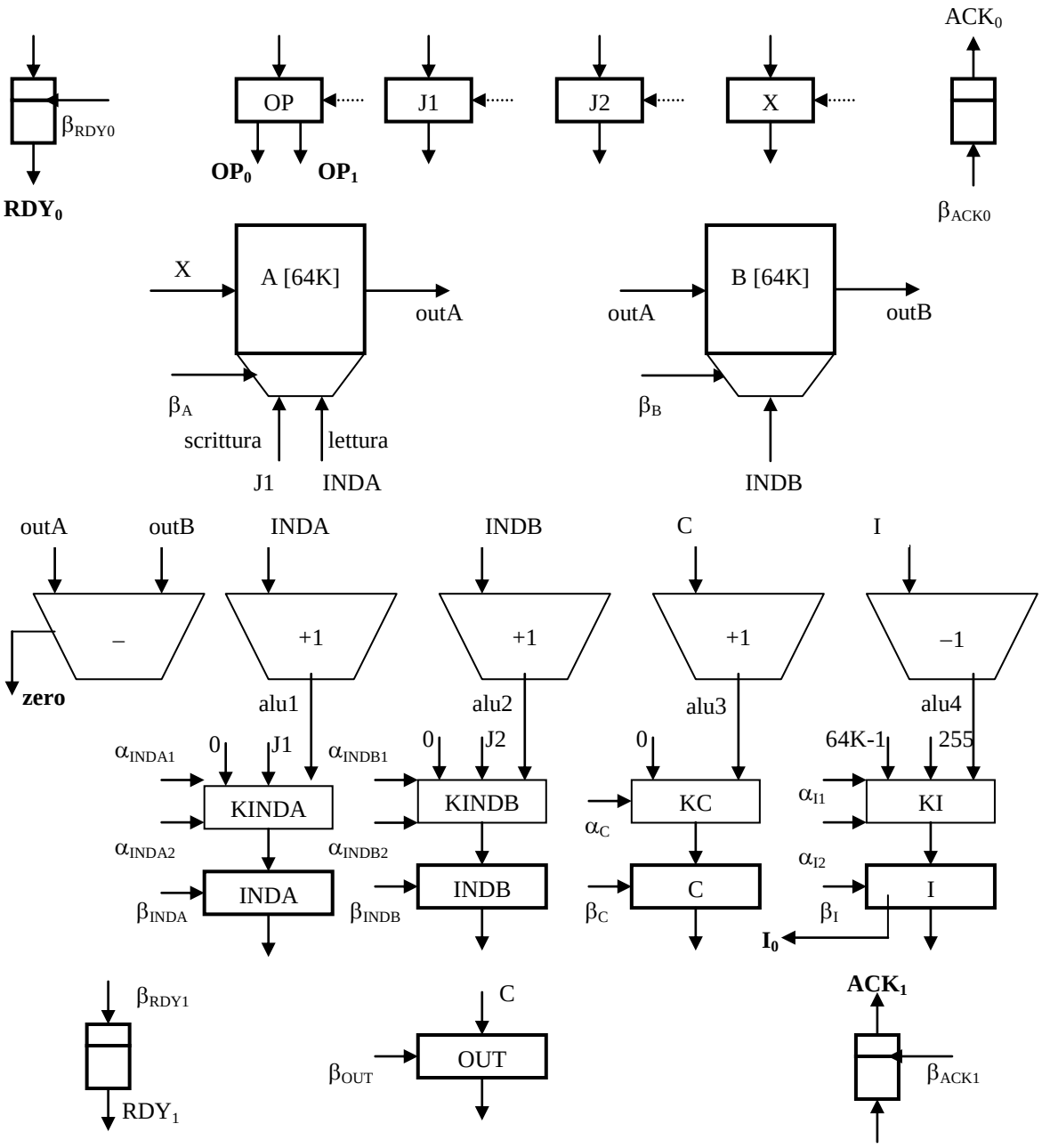
Lo schema della PO è mostrato nella figura a pagina seguente. Per ricavare questo schema, notiamo che:

1) le reti ALU sono in numero di cinque: quattro servono *per permettere il parallelismo* nella microoperazione della seconda frase della microistruzione 2 (tre incrementatori ed un decrementatore), la quinta (sottrattore) per disporre di una ALU indipendente allo scopo di soddisfare la condizione *cond-1* relativamente ad una PO di Moore;
2) è importante ribadire che le ALU utilizzate sono nel numero *minimo* per permettere il parallelismo presente nel microprogramma.

   Sarebbe invece un errore introdurre tante ALU per quante sono le operazioni aritmetico-logiche presenti nel microprogramma indipendentemente dal parallelismo esplicitato.

   Inoltre, il fatto che nella PO di figura non ci siano commutatori sugli ingressi delle ALU è, in questo caso, del tutto fortuito, e non va assolutamente considerata una regola;
3) la memoria A è indirizzata in scrittura mediante il registro J1, in lettura mediante il registro INDA. Infatti, per soddisfare la condizione *cond-2* relativamente ad una PO di Moore, occorre che tale memoria disponga di una logica separata per l'indirizzamento in scrittura (selezionatore controllato da J1) e per l'indirizzamento in lettura (commutatore controllato da INDA).

   Si noti che *non ha senso usare questa tecnica per un numero qualunque di logiche di indirizzamento distinte* (ad esempio non è proponibile avere qualcosa del tipo: 3 logiche per l'indirizzamento in lettura e 2 in scrittura);
4) i registri C, I, INDA, INDB hanno una struttura tra loro analoga;
5) lo schema delle interfacce è quello standard con la soluzione *a transizione di livello*: gli indicatori di uscita (ACK_0, RDY_1) hanno un unico ingresso dato da una variabile di controllo (di tipo β) per consentire l'operazione *set*, quelli di ingresso (RDY_0, ACK_1) hanno, oltre all'ingresso esterno, un ulteriore ingesso dato da una variabile di controllo (di tipo β) per consentire l'operazione *reset*.



Funzione delle uscite di PO

Indicando con lettere *x* le variabili di condizionamento e con lettere *z* le uscite esterne, *dall'analisi della struttura della rete sequenziale PO* si ricavano le espressioni logiche di tutte le variabili di uscita in funzione dello stato interno della PO stessa, verificando che in tali espressioni non compiano variabili di controllo:

$$\begin{cases} x_0 = RDY_0 \\ x_1 = OP_0 \\ x_2 = OP_1 \\ x_3 = I_0 \\ x_4 = \text{zero}(A[INDA] - B[INDB]) \\ x_5 = ACK_1 \end{cases} \qquad \begin{cases} z_0 = ACK_0 \\ z_1 = RDY_1 \\ z_2 = \text{OUT (parola)} \end{cases}$$

In particolare, dall'espressione di x_4 si verifica che, *con la realizzazione data*, PO è una rete sequenziale di Moore (condizione necessaria per la correttezza).

Funzione di transizione dello stato interno di PO

Limitatamente al registro INDB, occorre esprimere il valore che, ad ogni ciclo di clock, viene assunto dall'ingresso di tale registro, in funzione dello stato interno della PO e dello stato d'ingresso della PO (variabili di controllo), *nel caso che* tale valore rappresenti effettivamente lo stato interno al prossimo ciclo di clock. Si ha:

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

case $\alpha_{INDB1}\ \alpha_{INDB2}$ **of**

0 0 **:** 0

0 1 : J2

1 – **:** INDB + 1

dove la scrittura *when* $\beta_{INDB} = 1$ sta a significare che il *valore* dell'espressione, che segue la parola chiave *do*, rappresenta effettivamente il valore dello stato successivo solo a condizione che venga effettuata la scrittura nel registro alla fine del ciclo di clock.

Per la generica locazione di indirizzo *i* della memoria B si ha:

$in_{B[i]}$ = **when** $\beta_B = 1$ **do**

if (INDB = i) **then** A[INDA]

dove occorre evidenziare, oltre alla scrittura *when* $\beta_B = 1$, la presenza del comando *if (INDB = i) then*, stante a significare che l'unica locazione modificata è quella di indirizzo *i*, mentre tutte le altre rimangono inalterate.

c) Parte Controllo

Codifichiamo, nella maniera più naturale, gli stati interni della PC (corrispondenti alle microistruzioni del microprogramma) mediante due *variabili dello stato interno*, y_0 e y_1:

$$\begin{cases} \text{stato } 0 \equiv y_0\, y_1 = 0\ 0 \\ \text{stato } 1 \equiv y_0\, y_1 = 0\ 1 \\ \text{stato } 2 \equiv y_0\, y_1 = 1\ - \end{cases}$$

Dato il numero limitato di espressioni logiche da ricavare, conviene *lavorare direttamente sulla struttura del microprogramma* invece che riempire la tabella di verità delle due funzioni della PC (si tratta di rappresentazioni del tutto equivalenti). Indicando con Y_0 e Y_1 le variabili dello stato successivo, la parte di microprogramma che ci interessa è:

y_0y_1 = 00 (RDY0, OP_0, OP_1 = 0 – –) …, **Y_0Y_1 = 00**

(= 1 0 0) …, **Y_0Y_1 = 00**

(= 1 0 1) $\beta_{INDB} = 1$, …, **Y_0Y_1 = 01**

(= 1 1 –) $\beta_{INDB} = 1$, …, **Y_0Y_1 = 1–**

y_0y_1 = 01 (I_0 = 0) $\beta_{INDB} = 1$, …, **Y_0Y_1 = 01**

(= 1) ….., **Y_0Y_1 = 00**

y_0y_1 = 1– (I_0, zero (A[INDA] – B[INDB]), ACK1 = 0 0 –) $\beta_{INDB} = 1$, …,, **Y_0Y_1 = 1–**

(= 0 1 –) $\beta_{INDB} = 1$, …., **Y_0Y_1 = 1–**

(= 1 – 0) …., **Y_0Y_1 = 1–**

(= 1 – 1) … , **Y_0Y_1 = 00**

Funzione di transizione dello stato interno di PC

La funzione di transizione dello stato interno della PC è quindi data dalle seguenti espressioni logiche :

$$\begin{cases} Y_0 = \overline{y_0}\ \overline{y_1}\ RDY_0\ OP_0 + y_0\ \overline{I_0} + y_0\ I_0\ \overline{ACK_1} \\ Y_1 = \overline{y_0}\ \overline{y_1}\ RDY_0\ \overline{OP_0}\ OP_1 + \overline{y_0}\ y_1\ \overline{I_0} \end{cases}$$

Per ogni variabile dello stato interno successivo, sono stati espressi i termini AND per i quali tale variabile vale uno, e tutti i termini AND trovati sono combinati in OR, applicando semplici riduzioni.

Funzione delle uscite di PC

La funzione delle uscite della PC, relativamente alla sola variabile di controllo β_{INDB}, è data dalla seguente espressione logica:

$$\beta_{INDB} = \overline{y_0}\ \overline{y_1}\ RDY_0\ OP_1 + \overline{y_0}\ y_1\ \overline{I_0} + y_0\ \overline{I_0}$$

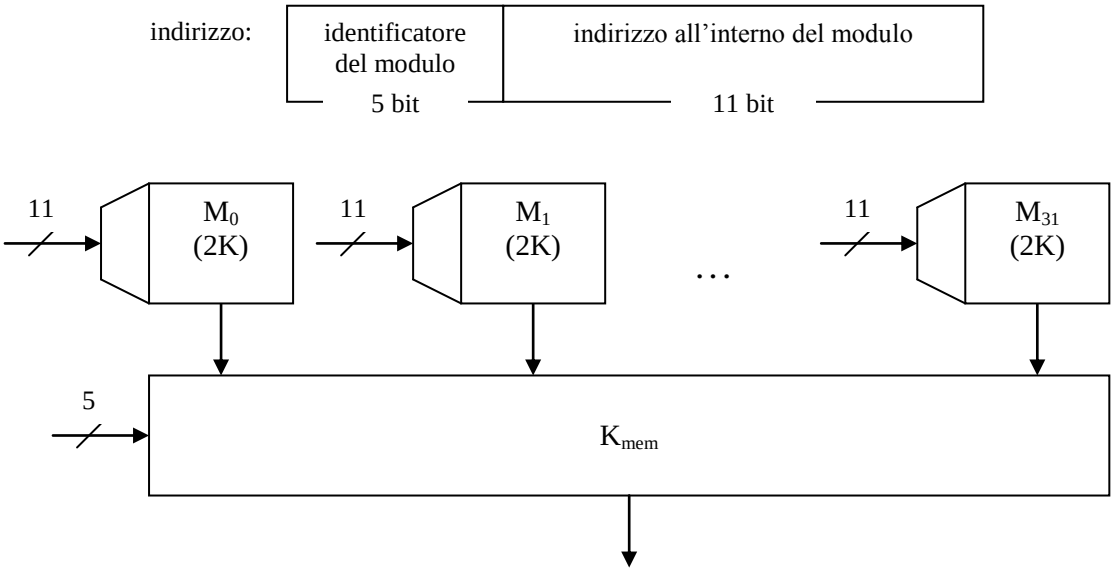
Sono stati espressi i termini AND per i quali la variabile β_{INDB} vale uno, e tutti i termini AND trovati sono combinati in OR, applicando semplici riduzioni.

d) Prestazioni

Il *massimo ritardo della funzione delle uscite della PO* è quello per la stabilizzazione della variabile di condizionamento $x_4 = zero(A[INDA] - B[INDB])$:

$$T_{\omega PO} = t_a + t_{ALU}$$

t_a è il tempo di accesso delle memorie A e B (ovviamente, le due letture avvengono in parallelo).



Ogni memoria è realizzata mediante 32 memorie da 2K poste tra loro in parallelo; ciascuna memoria da 2K viene quindi indirizzata dagli 11 bit meno significativi di IND (INDA o INDB). Le uscite delle memorie da 2K sono ingressi di un commutatore K_{mem}, le cui 5 variabili di controllo sono i corrispondenti bit più significativi di IND.

Ogni porta AND di K_{mem} ha dunque 6 ingressi, e quindi è realizzabile con un solo livello di logica.

Le 32 uscite delle porte AND vanno invece combinate in OR mediante un albero a 2 livelli.

Dunque, essendo il tempo di accesso t_{a0} delle memorie di 2K uguale a $5t_p$:

$$t_a = t_{a0} + t_{Kmem} = 5t_p + 3t_p = 8t_p$$

da cui:

$$T_{\omega PO} = 8t_p + 5t_p = 13t_p$$

Per ricavare *il massimo ritardo della funzione di transizione dello stato interno della PO*, occorre considerare i ritardi di stabilizzazione di tutte le operazioni elementari di trasferimento tra registri che compaiono nelle microoperazioni. Non considerando le operazioni identità e le operazioni sugli indicatori di interfaccia, tutte con ritardo nullo, si ha:

$X \rightarrow A[J1]$: è il tempo di stabilizzazione del selezionatore di ingresso della memoria A. Trattandosi di una rete combinatoria ad un solo livello di logica, in base a quanto prima discusso si ha un ritardo:

$$t_{a0} + t_p = 6t_p$$

In assenza di altre informazioni, il ritardo di scrittura per una memoria data si assume uguale al ritardo di lettura (anche se, presumibilmente, sarà inferiore).

$A[INDA] \rightarrow B[INDB]$: durante il ritardo di stabilizzazione della lettura di A, in parallelo avviene anche la stabilizzazione del selezionatore in ingresso a B (si ricordi che il selezionatore opera sul β di abilitazione alla scrittura, non sul dato d'ingresso). Dunque, il ritardo è

$$t_a = 8t_p$$

$INDA + 1 \rightarrow INDA$, e tutti i casi analoghi di incremento/decremento di un registro. Poiché un commutatore a due o tre ingressi è realizzabile a due livelli di logica, il ritardo è dato da

$$t_{ALU} + t_K = 5t_p + 2t_p = 7t_p$$

Dunque:

$$T_{\sigma PO} = 8t_p$$

Il *massimo ritardo della funzione di transizione dello stato interno della PC* si ricava dalle espressioni di Y_0 e Y_1 ottenute in precedenza:

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

Il *massimo ritardo della funzione delle uscite della PC* si deduce rapidamente osservando che:

i) il massimo numero di variabili presenti in un termine AND di una qualunque variabile di controllo è cinque (due variabili dello stato interno e al massimo tre variabili di condizionamento, come avviene nelle microistruzioni 1 e 3);

ii) il massimo numero di termini AND combinati in OR, per una qualunque variabile di controllo, è cinque, come si ricava osservando β_I (nel microprogramma compaiono cinque operazioni elementari che provocano la scrittura nel registro I).

Dunque, anche la funzione delle uscite della PC è realizzabile a due livelli di logica:

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

Il ciclo di clock è quindi dato da:

$$\tau = T_{\omega PO} + max\,(T_{\omega PC} + T_{\sigma PO}\,,\, T_{\sigma PC}) + \delta = 13t_p + 2t_p + 8t_p + t_p = 24t_p$$

L'operazione esterna 1 ha tempo medio di elaborazione:

$$T_1 = \tau = 24\, t_p$$

L'operazione esterna 2 viene eseguita come segue: una volta viene eseguita la microistruzione 0 (inizializzazione), 256 volte la microistruzione 1 (loop), ed una volta ancora la microistruzione 1 (uscita dal loop); dunque il suo tempo medio di elaborazione è dato da:

$$T_2 = 258\ \tau = 6{,}192 * 10^3\, t_p$$

L'operazione esterna 3 viene eseguita come segue: una volta viene eseguita la microistruzione 0 (inizializzazione), 64K volte la microistruzione 2 (loop), ed una volta ancora la microistruzione 2 (uscita dal loop); dunque il suo tempo medio di elaborazione è dato da:

$$T_3 \cong 64K\ \tau = 64 * 1024 * 24\ t_p = 1{,}573 * 10^6\ t_p$$

e) Procedimento abbreviato

Lo sviluppo completo della PO e della PC non è necessario agli effetti della determinazione del ciclo di clock.

Per la determinazione dei ritardi di stabilizzazione delle funzioni della PC si ragiona come visto sopra:

1) si determina facilmente il numero massimo di variabili (di condizionamento + dello stato interno) che possono comparire in un termine AND della ω_{PC} e della σ_{PC}, e si verifica che tale numero sia minore o uguale del massimo numero di ingressi per porta logica. Nel nostro caso, il numero di variabili *specificate* è 5 (al massimo si hanno 2 variabili dello stato interno specificate e 3 variabili di condizionamento specificate); notiamo che la condizione è soddisfatta addirittura dall'*upper bound* (cioè, senza considerare i non specificati), che è uguale 8 (2 variabili dello stato interno e 6 variabili di condizionamento). Nel caso che la condizione sul massimo numero di ingressi per porta non sia soddisfatta, si determinerà un upper bound del numero di livello di logica per i termini AND;

2) per quanto riguarda i termini OR della σ_{PC}, si determina il massimo numero di occorrenze di un "1" nelle colonne delle variabili dello stato successivo nella tabella di verità (o meglio, la sua rappresentazione direttamente sul microprogramma). Nel caso che la condizione sul massimo numero di ingressi per porta non sia soddisfatta, si determinerà un upper bound del numero di livello di logica per i termini OR;

3) analogamente per quanto riguarda i termini OR della ω_{PC}; in particolare, si considera il massimo numero (o l'upper bound) dei termini AND nelle espressioni delle variabili di controllo di tipo β, in quanto le espressioni logiche di queste variabili contengono un numero di "1" maggiore rispetto alle variabili α. Nel caso che la condizione sul massimo numero di ingressi per porta non sia soddisfatta, si determinerà un upper bound del numero di livello di logica per i termini OR.

Per quanto riguarda i ritardi di stabilizzazione delle funzioni della PO, è sufficiente una rappresentazione "ridotta" della PO, deducibile facilmente dal microprogramma, che si limiti alle operazioni elementari ed alle condizioni logiche aventi ritardo non nullo: per esse vengono riconosciuti i ritardi delle funzioni presenti (ALU, K, memorie). Particolare attenzione va posta nella valutazione del *numero di commutatori* necessari (ingressi di registri o memorie, ingressi di ALU, ingressi di indirizzi di memorie, applicazioni del controllo residuo).

Domanda 2

a) L'asserto "il risultato della funzione delle uscite di PC dipende, ad ogni ciclo di clock, sia dallo stato interno presente che dallo stato d'ingresso di PC" (cioè che PC sia un automa di Mealy) va dimostrato.

Esso è una conseguenza della *struttura di frase* delle microistruzioni del microlinguaggio adottato: ad ogni ciclo di clock, lo stato di uscita di PC (configurazione delle variabili di controllo) corrisponde all'esecuzione di una ben determinata microoperazione, appartenente ad una ben determinata frase, *la quale è individuata dall'etichetta della microistruzione cui appartiene (stato interno) e dal valore delle variabili di condizionamento testate (stato d'ingresso):*

i. $(X_{i.0}, X_{i.1}, \ldots, X_{i.k}$*-= ……..) microoperazione, …; …..*

b) L'affermazione è falsa, in quanto la variabile di condizionamento in questione è prelevata direttamente all'uscita di un registro (X) e quindi, <u>*ad ogni ciclo di clock*</u>, il risultato della funzione delle uscite di PO dipende solo dallo stato interno di PO (almeno per quanto riguarda la variabile in questione).

La modifica del contenuto del registro X *non* riguarda la funzione delle uscite di PO, *bensì* la funzione di transizione dello stato interno, il cui risultato deve dipendere, ad ogni ciclo di clock, anche dal valore delle variabili di controllo: ma questo non ha niente a che vedere con la condizione di correttezza della PO (automa di Moore), che riguarda solo la funzione delle uscite.

c) In questo esempio A è considerato un *array di quattro byte*; il controllo residuo consiste nel pilotare, mediante i due bit di uscita del registro I, un commutatore avente in ingresso i quattro byte di A.

Senza usare controllo residuo, la microistruzione sarebbe la seguente:

4. (A_0, I_0, I_1 = 0 0 0) A[0] $\rightarrow$ B, 5;

   (A_0, I_0, I_1 = 0 0 1) A[1] $\rightarrow$ B, 5;

   (A_0, I_0, I_1 = 0 1 0) A[2] $\rightarrow$ B, 5;

   (A_0, I_0, I_1 = 0 1 1) A[3] $\rightarrow$ B, 5;

   (A_0, I_0, I_1 = 1 – –) A $\rightarrow$ C, 6

dove l'indirizzo di A è ottenuto mediante due variabili di controllo in uscita dalla PC.

Questa realizzazione, che usa due variabili di condizionamento in più (I_0, I_1), è caratterizzata da una complessità di progettazione della Parte Controllo che, come ordine di grandezza, è 2^2 volte maggiore rispetto a quella che fa uso di controllo residuo. Questo è il vantaggio principale.

Il tempo medio di elaborazione è lo stesso, a condizione che il ciclo di clock sia uguale (un maggior numero di variabili di condizionamento *potrebbe* comportare un maggior ritardo di stabilizzazione delle reti combinatorie della PC).

d) In tutti e tre i casi dell'Esercitazione 1, la realizzazione mediante unità di elaborazione è basata su un *microprogramma che*

- *consta di una sola microistruzione, quindi la PC si riduce ad una rete combinatoria (ω_{PC});*
- *contiene microoperazioni corrispondenti alle realizzazioni date nell'Esercitazione 1, eventualmente usando componenti logici standard.*

Facciamo corrispondere ad ogni variabile logica un registro d'ingresso (ad esempio, X per *x*) o un registro di uscita (ad esempio, Z per *z*).

<u>*Unità corrispondente alla rete combinatoria della Domanda 1, Esercitazione 1*</u>

0. (RDYIN, ACKOUT = 0 –, 1 0) nop, 0;

   (= 1 1) reset RDYIN, set ACKIN, set RDYOUT, reset ACKOUT**, F_z (A, B, X, Y) $\rightarrow$ Z,**
   F_w (A, B, X, Y) $\rightarrow$ W, 0

F_z e F_w sono le realizzazioni delle funzioni di *z* e *w* con le reti a due livelli di logica per minimizzare il ciclo di clock, oppure con componenti logici standard al costo di un ciclo di clock maggiore.

La PO consta delle interfacce d'ingresso e delle interfacce di uscita separate dalle reti F_z e F_w, quindi:

$$T_{\omega PO} = 0 \quad , \quad T_{\sigma PO} = 2\,t_p \quad \text{(usando le reti a due livelli di logica)}$$

La funzione delle uscite di PC serve esclusivamente a settare e resettare gli indicatori di interfaccia, quindi

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

che rappresenta il ritardo della porta AND con ingressi RDYIN e ACKOUT.

Il ciclo di clock vale quindi

$$\tau = 4\ t_p$$

In pratica, eccetto per la presenza di una PC combinatoria ("piccola", spesso riconducibile a soli collegamenti), è esattamente la realizzazione della rete combinatoria dell'Esercitazione 1 con l'aggiunta delle interfacce, che, d'altra parte, sono necessarie per garantire la sincronizzazione con le altre unità.

Unità corrispondente alla rete sequenziale della Domanda 2, Esercitazione 1

Il microprogramma è quello a cui corrisponderà una PO (secondo il modello di Moore) come quella dell'Esercitazione 1 realizzata con componenti standard, con S operante anche da registro di uscita:

0. (RDYIN, ACKOUT, X = 0 – –, 1 0 –) nop, 0;

   (= 1 1 0) reset RDYIN, set ACKIN, set RDYOUT, reset ACKOUT, 0;

   (= 1 1 1) reset RDYIN, set ACKIN, set RDYOUT, reset ACKOUT, **S + 1 → S**, 0

Ancora più aderente alla realizzazione dell'Esercitazione1 è quella che fa uso di *controllo residuo* tramite X per abilitare la scrittura in S. Usando la notazione del Cap. IV, sez. 4.2, punto 3:

0. (RDYIN, ACKOUT = 0 – , 1 0) nop, 0;

   (= 1 1) reset RDYIN, set ACKIN, set RDYOUT, reset ACKOUT, **S + 1 → S | (X = 1)** , 0

Ragionando come sopra, e con i dati dell'Esercitazione 1 ($T_{ALU} = 4\ t_p$), il ciclo di clock vale:

$$\tau = 6\ t_p$$

Unità corrispondente alla rete sequenziale della Domanda 3, Esercitazione 1

Il microprogramma, che fa pesantemente uso di *controllo residuo*, è il seguente:

0. (RDYIN, **ACKOUT [E]**, A [B] = 0 – –, 1 0 –) nop, 0;

   (= 1 1 0) reset RDYIN, set ACKIN, **set RDYOUT [E]**, **reset ACKOUT [E]**, C + D → Z [E], 0;

   (= 1 1 1) reset RDYIN, set ACKIN, **set RDYOUT [E]**, **reset ACKOUT [E]**, C – D → Z [E], 0

La variabile di condizionamento ACKOUT [E] è ottenuta come uscita di un commutatore avente in ingresso tutti i 64 bit ACKOUT e controllato da E.

La modifica dei β dei 64 RDYOUT, ACKOUT e Z è effettuata tramite un selezionatore avente in ingresso un β e controllato da E.

Estendendo il controllo residuo anche alla determinazione dell'operazione eseguita dalla ALU, si può scrivere (Cap. IV, sez. 4.2, punto 2):

0. (RDYIN, ACKOUT [E] = 0 –, 1 0) nop, 0;

   (= 1 1) reset RDYIN, set ACKIN, set RDYOUT [E], reset ACKOUT [E],
   ALU (A [B], C, D) → Z [E], 0;

dove *ALU (A[B], ...)* = addizione se A[B] = 0, *ALU (A [B], ...)* = sottrazione se A[B] = 1.

Ora si ha, con i dati dell'Esercitazione 1 (porte con al massimo 4 ingressi):

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

$$T_{\omega PO} = T_{K\text{-}ACKOUT} = T_{AND} + T_{OR} = 2\ t_p + 3\ t_p = 5\ t_p$$

$$T_{\sigma PO} = T_{K\text{-}A} + T_{ALU} = 9\ t_p$$

Quindi:

$$\tau = 16\ t_p$$