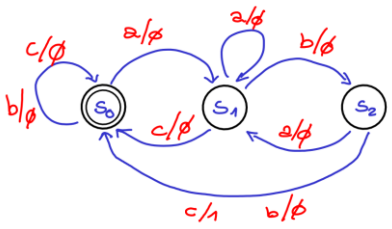
Architettura degli elaboratori – A.A. 2017–18

Appello 11 settembre 2018

Riportare in alto a destra di tutti i fogli consegnati nome, cognome, numero di matricola, corso (A o B).
I risultati saranno resi disponibili via WEB appena disponibili insieme al calendario degli orali.

Domanda 1

E' dato l'automa in figura che riconosce la stringa "**abc**" fra le stringhe di 3 caratteri definite sull'alfabeto **{a,b,c}**. Si progetti una rete sequenziale che implementa l'automa e se ne fornisca il tempo di servizio.



Domanda 2

Si consideri il seguente microcodice:

1. (RDY_{in},ACK_{out},segno(M[IN]-M[IN/16])=0--) nop, 1
   (=110) IN+M[IN]→OUT, reset ACK_{out}, set RDY_{out}, reset RDY_{in}, set ACK_{in}, 1
   (=111) IN-M[IN]→OUT, reset ACK_{out}, set RDY_{out}, reset RDY_{in}, set ACK_{in}, 1
   (=10-) nop, 1

IN (da 10 bit), RDY_{in} e ACK_{in} sono componenti dell'interfaccia verso un'altra unità che invia IN. OUT (da 32 bit), RDY_{out} e ACK_{out} sono componenti dell'interfaccia verso un'altra unità cui si invia OUT. Si fornisca un'ottimizzazione del microcodice che comporti una riduzione della lunghezza del ciclo di clock della corrispondente unità firmware. Si discuta la differenza della banda offerta dalle due unità (si assuma che $t_a = t_{alu} = 10t_p$).

Domanda 3

Dopo aver fornito la definizione di dipendenza logica relativamente al codice assembler di un processore D-RISC pipeline, si consideri un codice assembler, privo di cicli, composto da 40 istruzioni. Su un processore D-RISC pipeline classico (4 stadi, con la sola EU master che esegue tutte le operative aritmetico logiche del programma in 1t) l'esecuzione del codice impiega 50t. Si discuta cosa succede del tempo di esecuzione di due istanze dello stesso codice quando venga invece utilizzato un processore D-RISC pipeline con supporto multithreading a 2 vie, nei due casi in cui il supporto multithreading sia organizzato secondo il modello interleaving o secondo il modello blocking.

Traccia di soluzione

Domanda 1

L'automa è un automa di Mealy, visto che l'uscita dipende sia dallo stato interno che dal valore degli ingressi. Pertanto, l'automa sarà implementato da una rete sequenziale di Mealy dove

- le reti ω e σ accettano come ingressi lo stato interno corrente e la configurazione dell'input e producono, rispettivamente l'uscita booleana e lo stato interno successivo, e
- il registro di stato R, da 2 bit, contiene la rappresentazione dello stato corrente.

Assumiamo di rappresentare S_0, S_1 ed S_2 con 00, 01 e 10, rispettivamente. Assumiamo anche di rappresentare "a", "b" e "c" con 00, 01 e 10, rispettivamente. Si possono definire le reti combinatorie ω e σ mediante tabelle di verità che avranno 4 ingressi (1 livello AND) e 3 * 3 righe (3 stati, ciascuno dei quali può accettare 3 diversi valori di input). Con le solite assunzioni possiamo concludere che anche per il livello OR basta una sola porta. Quindi entrambe le reti avranno un ritardo di $2t_p$. Il ciclo di clock della rete sequenziale sarà quindi dato dal massimo fra i ritardi delle due reti (2tp) e dal tempo in cui il segnale di clock è alto (1tp) per un totale di 3tp. Il tempo di servizio è normalmente definito come inverso del ciclo di clock. In questo caso potremmo considerare che servono 3 cicli per riconoscere (o meno) una stringa di tre caratteri e dunque concludere che il tempo di servizio è 1 / 3τ.

Domanda 2

Il microprogramma di controllo definisce una unità firmware costituita da una sola rete sequenziale, visto che c'è una micro istruzione sola. Il ciclo di clock sarà dato da

$T\omega PO + T\omega PC + T\sigma PO + \delta$

La ωPC è in realtà una rete combinatoria parte della PO che calcola i valori delle variabili di controllo a partire dalle sole variabili di condizionamento. Possiamo concludere che le variabili di controllo (<8 ingressi e < 4 colonne/frasi) vengono calcolare in 2tp. La ωPO calcola le variabili di condizionamento spendendo un massimo di ta + talu. L'operazione /16 si implementa utilizzando solo una parte dei "fili" di IN, essendo 16 una potenza di 2. La σPO richiede ancora un ta + talu, anche se va osservato che il valore di M[IN] è già stato generato in fase di calcolo delle variabili di condizionamento. Dunque possiamo dire che il clock può essere calcolato come

ta + talu + 2tp (+ta) + talu + tp = 33 tp

Va notato che in questo caso paghiamo due alu in sequenza oltre al tempo di accesso in memoria. Possiamo pensare di ridurre la durata del ciclo di clock facendo in modo che i due tempi (calcolo delle var di condizionamento e calcolo di OUT) non si sommino. Questo si può fare in due modi:

1. Utilizzando il controllo residuo: le variabili di condizionamento sono calcolare insieme ad entrambe le due IN – M[IN] e IN + M[IN], che diventano gli ingressi di un commutatore da comandare con il risultato della variabile di condizionamento. In questo modo la singola instruzione risultante avrà bisogno di un clock pari a ta + talu + tk + tp, più corto di quello originale.
2. Utilizzando due microistruzioni: la prima calcola la variabile di condizionamento e la scrive in un registro, e la seconda usa il registro per decidere cosa fare. Il ciclo di clock è ridotto come nel caso precedente, ma in questo caso il tempo per trattare un ingresso è di due τ.

In entrambi i casi il ciclo di clock si accorcia, ma nel secondo caso la banda non cala perché servono due microistruzioni invece che una sola.

Domanda 3

Per la definizione di dipendenza logica, si veda il libro di testo.

Nel nostro caso, il tempo di esecuzione di 50t per 40 istruzioni significa che sulla IU abbiamo 10t persi a causa di “bolle” generate da dipendenze. Le bolle possono essere singole o doppie, per il tipo di architettura considerata.

Nei due casi, le bolle singole rimangono e quelle doppie vengono parzialmente riassorbite dall’esecuzione delle istruzioni dell’altro thread.