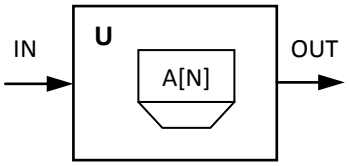
Architettura degli Elaboratori

appello del 13 luglio 2010

informazioni e indicazioni sul retro

Domanda 1 *(per tutti)*

Una unità di elaborazione U è descritta dal seguente microprogramma, con *IN* parola di 32 bit e *A* memoria di *N* parole:



0. (RDYIN = 0) nop, 0;

   (= 1) reset RDYIN, set ACKIN, IN $\rightarrow$ X, 0 $\rightarrow$ I, 0 $\rightarrow$ C, 1

1. (I_0, segno (A[I_m] – X), ACKOUT = 0 0 –) I + 1 $\rightarrow$ I, 1;

   (= 0 1 –) I + 1 $\rightarrow$ I, C + 1 $\rightarrow$ C, X $\rightarrow$ A[I_m], 1;

   (= 1 – 0) nop, 1;

   (= 1 – 1) reset ACKOUT, set RDYOUT, C $\rightarrow$ OUT, 0

a) *Definire formalmente* la rete sequenziale Parte Operativa di U, spiegando chiaramente come tale definizione è stata ricavata ed usando la corretta terminologia.

b) Spiegare dove la definizione data al punto *a)* è influenzata dal fatto che la rete sequenziale Parte Operativa è *sincrona*.

c) Si desidera una seconda versione di U, che impieghi lo stesso numero di cicli di clock, con la caratteristica che il microprogramma faccia uso di controllo residuo e di sole variabili di condizionamento semplici. Confrontare il ciclo di clock delle due versioni, assumendo $t_{ALU} = t_A = 5t_p$.

Domanda 2 *(per tutti)*

Si consideri una cache associativa su insiemi.

a) Spiegare come è definito il metodo di indirizzamento di una tale cache.

b) Spiegare l'implementazione di una unità cache che risponda a tale definizione.

c) Ricavare il ciclo di clock di tale unità senza considerare le azioni conseguenti alla rilevazione del fault di blocco e senza tenere conto di vincoli imposti dalla consistenza con il ciclo di clock delle altre unità della CPU.

Domanda 3

NEW, OLD-0

Con riferimento ad una CPU pipeline D-RISC, spiegare come può essere implementata, nell'interprete firmware, la tecnica del delayed branch.

OLD-1

Un programma parallelo LC contiene un processo interno Q ed un processo esterno I/O. Q invia un segnale a I/O, attende da I/O un array A di 1024 interi, ed esegue una computazione F consistente nell'incrementare di uno ogni elemento di A.

Spiegare quali implicazioni si hanno sul tempo di completamento della sola parte *F* di Q nei seguenti due casi: *a)* l'unità corrispondente ad I/O opera solo in Memory Mapped I/O, *b)* l'unità corrispondente ad I/O opera anche in DMA.

Per questo insegnamento sono previsti, durante l'intero a.a., 6 appelli ed un numero massimo di 5 prove.

Gli studenti del vecchio ordinamento devono scegliere se presentarsi sul programma di esame 2009-10 con l'aggiunta del Cap. VII (Processi), sez. 5, 6 e 7, oppure sul programma di esame 2008-09. La scelta va fatta la prima volta che lo studente si iscrive all'esame, e vale per tutti gli eventuali successivi appelli dell'a.a. ai quali lo studente si presenti.

Riportare su tutti i fogli consegnati nome, cognome, numero di matricola, corso di appartenenza, e la sigla NEW (per nuovo ordinamento), oppure OLD-0 (per vecchio ordinamento, programma 2009-10), oppure OLD-1 (per vecchio ordinamento, programma 2008-09).

I risultati verranno pubblicati sulle pagine web del corso/dei docenti appena disponibili.

Soluzione

Domanda 1 *(per tutti)*

a) La definizione formale della rete sequenziale PO richiede l'*analisi* della rete stessa una volta che la sua struttura sia stata ottenuta, in base al procedimento standard, a partire dal microprogramma. La struttura della PO consta di:

- tre ALU per: $A[I_m] - X$, $I + 1$, $C + 1$; per quanto il massimo parallelismo nell'uso delle ALU, espresso dal microprogramma, sia uguale a due, la prima ALU deve essere aggiunta per rispettare la condizione necessaria della correttezza (PO di Moore);
- registro X, variabile di controllo β_X;
- memoria A avente come indirizzo I_m e come ingresso X; variabile di controllo β_A;
- registri I, C, ognuno di $1+\log_2 N$ bit con un commutatore in ingresso; variabili di controllo α_I, β_I, α_C, β_C;
- registro OUT con ingresso C, variabile di controllo β_{OUT};
- indicatori a transizione di livello RDYIN, ACKIN, RDYOUT, ACKOUT: variabili di controllo β_{RDYIN}, β_{ACKIN}, β_{RDYOUT}, β_{ACKOUT}.

Sulla base di questa struttura, l'analisi della rete sequenziale è definita dalla seguente quintupla:

- *insieme delle variabili di ingresso*: variabili esterne *rdyin*, *ackout, in* (32), variabili di controllo β_A, α_I, βI, α_C, β_C, β_{OUT}, β_{RDYIN}, β_{ACKIN}, β_{RDYOUT}, β_{ACKOUT};
- *insieme delle variabili di uscita*: variabili esterne *ackin, rdyout, out* ($1+\log_2 N$), variabili di condizionamento x_0, x_1, x_2, x_3;
- *insieme delle variabili dello stato interno*: IN (32), X (32), A (N x 32), I ($1+\log_2 N$), C ($1+\log_2 N$), OUT ($1+\log_2 N$), S-RDYIN, Y-RDYIN, ACKIN, RDYOUT, S-ACKOUT, Y-ACKOUT;
- *funzione delle uscite*, ω_{PO}: ad ogni ciclo di clock
  - x_0 = RDYIN
  - $x_1 = I_0$
  - x_2 = segno $(A[I_m] - X)$: *questa funzione è nota solo attraverso l'analisi della struttura della PO, NON dal semplice microprogramma*
  - x_3 = ACKOUT
  - ackin = ACKIN
  - rdyout = RDYOUT
  - out = OUT

  si verifica che, *ad ogni ciclo di clock*, il risultato della funzione delle uscite dipende dal solo stato interno della PO;
- *funzione di transizione dello stato interno*, σ_{PO}: ad ogni ciclo di clock
  - in_{IN} = *when* clock *do* in
  - in_X = *when* $\beta_X = 1$ *do* IN
  - $in_{A[j]}$ = *when* $\beta_A = 1$ *and* $j = I_m$ *do* X
  - in_I = *when* $\beta_I = 1$ *do if not* α_I then 0 *else* $I + 1$
  - in_C = *when* $\beta_C = 1$ *do if not* α_C then 0 *else* $C + 1$

- in_{OUT} = *when* $\beta_{OUT} = 1$ *do* C
- $in_{S\text{-}RDYIN}$ = *when* clock *do* rdyin
- $in_{Y\text{-}RDYIN}$ = *when* $\beta_{RDYIN} = 1$ *do not* Y-RDYIN
- in_{ACKIN} = *when* $\beta_{ACKIN} = 1$ *do not* ACKIN
- in_{RDYOUT} = *when* $\beta_{RDYOUT} = 1$ *do not* RDYOUT
- $in_{S\text{-}ACKOUT}$ = *when* clock *do* ackout
- $in_{Y\text{-}ACKOUT}$ = *when* $\beta_{ACKOUT} = 1$ *do not* Y-ACKOU

intendendo che una scrittura come *when* $\beta = 1$ sta per *when* clock *and* $\beta = 1$.

b) In una rete sequenziale sincrona lo stato interno presente assume il valore dello stato interno successivo in ben determinati istanti di una sequenza temporale discreta. Questa condizione è *implementata* inserendo, in ogni richiusura binaria dello stato interno, un registro impulsato così che gli istanti di cui sopra coincidono con gli istanti di fine degli impulsi di clock.

Per tale motivo, le variabili dello stato presente coincidono con le uscite di tali registri, e le variabili dello stato successivo con i rispettivi ingressi. Questo rende ragione della notazione $in_{registro}$, per esprimere il valore assunto dal codominio della funzione di transizione dello stato interno, e della notazione *when clock,* per indicare quando tale valore diventa valore della corrispondente variabile dello stato interno presente. Inoltre, la transizione di stato è condizionata dall'abilitazione alla scrittura in alcuni registri, e ciò rende ragione della notazione *when* $\beta = 1$.

Infine, così come per la funzione di transizione dello stato interno, anche il risultato della funzione delle uscite è significativo solo per ogni ciclo di clock.

c) La variabile di condizionamento *segno* $(A[I_m] - X)$ è eliminata. Il risultato di tale funzione è invece usato per controllare la scrittura nel registro C e nella memoria A:

0. (RDYIN = 0) nop, 0;

   (= 1) reset RDYIN, set ACKIN, IN $\rightarrow$ X, 0 $\rightarrow$ I, 0 $\rightarrow$ C, 1

1. (I_0, ACKOUT = 0 –) I + 1 $\rightarrow$ I, (C + 1 $\rightarrow$ C, X $\rightarrow$ $A[I_m]$) | segno $(A[I_m] - X) = 1$, 1;

   (= 1 0) nop, 1;

   (= 1 1) reset ACKOUT, set RDYOUT, C $\rightarrow$ OUT, 0

La struttura della PO rimane sostanzialmente invariata, salvo che le variabili di controllo β_C, β_A coincidono con il flag del segno all'uscita della prima ALU.

Il numero di cicli di clock è identico nelle due versioni ($N + 2$).

Premesso che in entrambe le versioni $T_{\omega PC} = T_{\sigma PC} = 2t_p$, consideriamo la prima versione avente

$$T_{\omega PO} = t_A + t_{ALU} = 10\ t_p$$

Poiché il selezionatore della memoria A si stabilizza in parallelo alle ALU:

$$T_{\sigma PO} = t_{ALU} + t_K = 7\ t_p$$

Utilizzando, per default, la formula che fornisce l'upper bound del ciclo di clock:

$$\tau_1 = 20\ t_p$$

Cercando il minimo valore in assoluto, si nota che tutte le ALU e la memoria si stabilizzano in parallelo a ω_{PO}, mentre solo la stabilizzazione dei commutatori necessita di variabili di controllo stabili, quindi:

$$\tau_{1\text{-}min} = t_A + t_{ALU} + T_{\omega PC} + t_K + \delta = 15\ t_p$$

Nella versione con controllo residuo, avente $T_{\omega PO} = 0$, si ha che le ALU ed i commutatori si stabilizzano in parallelo all'esecuzione di *segno (A[I_m] – X)*. Applicando la formula che fornisce l'upper bound:

$$\tau_2 = T_{\omega PC} + t_A + t_{ALU} + \delta = 13\ t_p$$

In realtà, anche la funzione ω_{PC} si stabilizza in parallelo alle ALU, ottenendo:

$$\tau_{2\text{-min}} = t_A + t_{ALU} + \delta = 11\ t_p$$

In tutte le modalità di valutazione, la versione con controllo residuo ha ciclo di clock inferiore.

Domanda 2 *(per tutti)*

a, b) La *definizione* riguarda solo la specifica del metodo di *corrispondenza* tra identificatori di blocco di memoria principale e identificatori di blocchi di cache, mentre l'*implementazione* riguarda come passare dalla definizione al microprogramma ed allo schema dell'unità. *Vedere il libro*.

c) In accordo alle specifiche, si deve valutare il ciclo di clock del microprogramma costituito da una sola microistruzione, nella quale la verifica della presenza del blocco e valutazione dell'indice del blocco all'interno dell'insieme precedono l'accesso vero e proprio alla memoria cache e la scrittura del risultato nell'interfaccia di uscita. Quindi, applicando alla scrittura in cache ed all'interfaccia di uscita la tecnica del controllo residuo pilotato da "not fault":

$$\tau = t_{verifica} + t_{indice} + t_{mem\text{-}cache} + \delta$$

dove la funzione "verifica" comporta la stabilizzazione di una RAM (t_a), di confrontatori su parola ($2t_p$) e di una porta OR su parola ($2t_p$), e la funzione "indice" la stabilizzazione di una rete combinatoria a due livelli di logica ($2t_p$), quindi:

$$\tau = 2t_a + 7t_p$$

Domanda 3

NEW, OLD-0

In D-RISC le istruzioni di salto possono contenere l'annotazione per il delayed branch.

Decodificando una simile istruzione, nel caso di salto da effettuare, la IU, oltre ad inviare l'indirizzo di salto (e identificatore unico) alla IM, setta un registro DB di un bit per ricordare la situazione di delayed branch.

La prima istruzione ricevuta dalla IM avente identificatore unico (ad esempio, IC) *diverso* da quello in possesso della IU viene accettata ed eseguita *se* DB è settato (dopo di che DB viene resettato), altrimenti viene ignorata.

OLD-1

L'esecuzione della *send* da parte di I/O per inviare il valore di A, su un certo canale (fisicamente allocato in MI/O nel caso *a)*, in MI/O o in M nel caso *b)*), rileva che Q è in attesa, quindi provvede a copiare A nella variabile targa di Q. Questa variabile è fisicamente allocata in MI/O nel caso *a)* o in M nel caso *b)*.

Nel caso *a)* l'esecuzione di *F* legge e scrive *1024* parole da/in MI/O. Se la struttura di interconnessione CPU-I/O è un classico bus, questo comporta l'uso inefficiente, o la disabilitazione, della cache dati. Il tempo di completamento di *F* è dell'ordine di *2048* $t_{MI/O}$.

Nel caso *b)*, viene regolarmente usata la cache dati, generando *1024/*σ fault (la funzione *F* è caratterizzata da sola località), quindi il tempo di completamento è dell'ordine di *1024* t_M/σ.