Architettura degli Elaboratori

Appello straordinario per studenti fuori corso - 8 novembre 2013

Riportare nome, cognome, numero di matricola e corso A/B

Domanda 1

Definire formalmente, realizzare e valutare il ciclo di clock di una rete sequenziale così specificata:

i) riceve nondeterministicamente valori interi da due unità U_0 e U_1, con priorità al valore da U_0 in caso di presenza simultanea;

ii) ad ogni passo, modifica ogni elemento di un array di 64K interi con il massimo tra il valore dell'elemento stesso e il valore d'ingresso scelto come detto al punto precedente; il risultato è anche inviato a un'unità U_2;

iii) dopo aver operato su tutti gli elementi dell'array, l'automa rimane indefinitamente in uno stato stabile per qualunque valore dello stato d'ingresso.

Spiegare chiaramente il metodo adottato.

È noto il ritardo di stabilizzazione t_p di una porta logica con al più otto ingressi. Il ritardo di stabilizzazione di una rete combinatoria per operazioni aritmetiche è uguale a $5t_p$.

Domanda 2

Si consideri la procedura per la commutazione di contesto. Le strutture dati condivise riferite indirettamente sono gestite con il metodo degli indirizzi logici distinti. La gerarchia memoria virtuale – memoria principale ha pagine di ampiezza 512 parole. La memoria principale ha capacità massima 4G. La MMU riferisce la tabella di rilocazione mediante indirizzi fisici.

a) Elencare le strutture dati condivise riferite indirettamente utilizzate in questa procedura.

b) Scrivere e spiegare un codice D-RISC per la parte di procedura che procura i valori degli operandi dell'istruzione START_PROCESS a partire dalle informazioni contenute nei PCB del processo running e del processo destinato a entrare in esecuzione.

Domanda 3

Si consideri una CPU pipeline D-RISC. Si suppone che il livello assembler non preveda istruzioni aritmetiche in virgola mobile. Si vuole delegare tali funzionalità a un coprocessore di I/O avente una struttura parallela con unità funzionali in pipeline.

a) Spiegare in quale modo vengono compilati i programmi per utilizzare il coprocessore.

b) Spiegare come si differenzia il modello dei costi rispetto al caso in cui la stessa struttura del coprocessore faccia parte dell'Unità Esecutiva della CPU.

Traccia di soluzione

Domanda 1

Data la complessità del problema, l'elevato numero di stati e la necessità di sincronizzarsi con U_0, U_1, U_2, adottiamo il procedimento di definizione e progettazione di un'unità purché essa sia descritta da un *microprogramma di una sola microistruzione*.

Il microprogramma è infatti il seguente:

0. (I_0, RDY_0, RDY_1, ACKOUT = 0 0 0 – , 0 1 – 0, 0 0 1 0) nop, 0;

   (= 0 1 – 1) reset RDY_0, set ACK_0, max (IN_0, A[I_m]) → (A[I_m], OUT), set RDYOUT, reset ACKOUT, I + 1 → I, 0;

   (= 0 0 1 1) reset RDY_1, set ACK_1, max (IN_1, A[I_m]) → (A[I_m], OUT), set RDYOUT, reset ACKOUT, I + 1 → I, 0;

   (= 1 – – –) nop, 0 // vedi specifica *iii)* //

Il registro I (di 17 bit) è inizializzato a zero (automa con stato iniziale). La funzione *max* è realizzata facilmente come:

max (a, b) = if (segno(a - b) = 0) then a else b

L'unità è una singola rete sequenziale, secondo il modello di Moore (funzione delle uscite: *out* = *OUT*) contenente: una memoria A della quale si ricava il tempo di accesso $T_A = 8t_p$; una ALU seguita da un commutatore per realizzare la funzione *max*; su uno dei due ingressi di tale ALU è posto un commutatore K1 per la scelta tra IN0 e IN1; una ALU per incrementare il registro I; tutte le interfacce standard.

Da inserire la definizione della funzione di transizione dello stato interno.

Le espressioni logiche delle variabili di controllo (K1 e abilitazione della scrittura in tutti i registri, eccetto IN_0, IN_1) in funzione di I_0, RDY_0, RDY_1, ACKOUT sono:

$$\beta_{RDY_0} = \beta_{ACK_0} = \overline{I_0}\ RDY_0\ ACKOUT$$

$$\beta_{RDY_1} = \beta_{ACK_1} = \alpha_{K1} = \overline{I_0}\ \overline{RDY_0}\ RDY_1\ ACKOUT$$

$$\beta_A = \beta_I = \beta_{OUT} = \beta_{RDYOUT} = \beta_{ACKOUT} = \beta_{RDY_0} + \beta_{RDY_1}$$

Il ciclo di clock è dato da:

$$\tau = T_\sigma + \delta = T_A + T_{ALU} + T_K + \delta = T_A + T_{ALU} + 3\,t_p = 16\,t_p$$

in quanto tutte le variabili di controllo e K1 si stabilizzano in un tempo complessivamente minore del tempo di accesso T_A, e il ritardo della memoria per la scrittura è sovrapposto al ritardo della lettura ed a parte del ritardo della ALU per *max*.

Niente cambia, funzionalmente e prestazionalmente, se, nella descrizione a microprogramma, il predicato *max* è inserito nelle condizioni logiche.

Domanda 2

a) Le strutture dati condivise riferite indirettamente da questa procedura sono:

- il primo PCB in Lista Pronti del processo (Next) che entra in esecuzione, puntato dalla struttura condivisa (riferita direttamente) Testa della Lista;
- la Tabella di Rilocazione di Next, puntata da un campo del PCB di Next,
- il secondo PCB in Lista Pronti, puntato da un campo del PCB di Next.

I loro indirizzi logici nello spazio del processo in esecuzione sono ottenuti con il metodo degli identificatori unici degli oggetti condivisi riferiti indirettamente e mediante una tabella privata del processo stesso per la corrispondenza identificatori – indirizzi logici.

b) I parametri della START_PROCESS sono l'indirizzo di ripristino (immagine di IC) del processo Next e l'indirizzo fisico della Tabella di Rilocazione di Next, che deve essere passato a MMU. Il primo parametro è ottenuto facilmente leggendolo dal PCB di Next, il secondo traducendo esplicitamente l'indirizzo logico (nello spazio del processo in esecuzione) della Tabella di Rilocazione di Next mediante la Tabella stessa.

Segue il codice D-RISC a partire dalla conoscenza degli indirizzi logici del PCB_Next (in Rpcb_next) e PCB_running (in Rpcb_running) e utilizzando costanti già inizializzate in opportuni registri generali (ad esempio, Rpage_size = 512, Rnumero_pagine_fisiche = 8 Mega):

```
LOAD   Rpcb_next, Roffset_immagine_IC, Rimmagine_IC

LOAD   Rpbc_next, Roffset_id_tabril, Rid_tabril
LOAD   Rpcb_running, Roffset_puntatore_tab_corrispondenza, Rindirizzo_tab_corrispondenza
LOAD   Rindirizzo_tab_corrispondenza, Rid_tabril, Rindirizzo_logico_tabril
DIV    Rindirizzo_logico_tabril, Rpage_size, Ripl
LOAD   Rindirizzo_logico_tabril, Ripl, Rentry_tabril
MOD    Rentry_tabril, Rnumero_pagine_fisiche, Ripf
MUL    Ripf, Rpage_size, Rindirizzo_fisico_ tabril

START_PROCESS   Rimmagine_IC, Rindirizzo_fisico_ tabril
```

Domanda 3

a) La compilazione di un'operazione in virgola mobile $c = op(a, b)$ consiste in una sequenza di istruzioni per comunicare con l'unità di I/O coprocessore: usando la tecnica del Memory Mapped I/O, la delega al coprocessore consiste in tre istruzioni STORE, rispettivamente con valore sorgente *op*, *a*, *b*; il risultato *c* viene letto con una LOAD. Il microprogramma dell'unità di I/O fa sì che il servizio della richiesta di LOAD contenga la necessaria sincronizzazione (attesa che il valore *c* sia pronto). Se l'unità non fosse progettata in questo modo, occorre un metodo di sincronizzazione più classico, ad esempio via semafori e/o interruzione.

Tali istruzioni sono opportunamente inserite nel codice del programma allo scopo di ottimizzarne il tempo di completamento, tipicamente distanziando il più possibile la LOAD I/O dall'ultima STORE I/O per parallelizzare l'elaborazione del coprocessore con l'elaborazione delle istruzioni nella CPU, e distanziando il più possibile la LOAD dall'istruzione che utilizza il risultato *c*..

b) Per quanto riguarda il modello dei costi, e quindi l'impatto del coprocessore sulle prestazioni, occorre rivedere il ritardo Δ dovuto alle dipendenze logiche indotte su IU, direttamente o indirettamente, dalle operazioni in virgola mobile: la latenza del servente aumenta, rispetto ai valori $1+L_{pipe\text{-}k}$ e $1+L_{pipe\text{-}h}$ (vedi materiale didattico), di una quantità data dalla *latenza delle comunicazioni via Bus di I/O* nei due sensi (nel caso di sincronizzazione via interruzione occorre anche considerare l'ulteriore latenza del suo trattamento). Infatti, concettualmente Δ è il tempo di risposta del servente, dato da

$$R_Q = W_Q(\rho) + L_S$$

Il termine W_Q non cambia rispetto alla soluzione in cui la stessa struttura è interna alla EU, in quanto il tempo di servizio del servente rimane lo stesso per definizione. Invece, il termine L_S viene appunto aumentato delle suddette latenze.

Come ordine di grandezza, l'aumento di latenza è molto significativo, dell'ordine di diverse decine di cicli di clock – formalmente, circa $8\ (\tau + T_{tr})$ –, quindi preponderante rispetto alle stesse latenze $1+L_{pipe\text{-}k}$ e $1+L_{pipe\text{-}h}$.

Le ottimizzazioni di cui al punto *a)* tendono, come sempre, a mascherare almeno in parte le latenze del servente: per quanto detto sopra, ciò assume un'importanza addirittura molto maggiore rispetto al caso di operazioni realizzate all'interno della EU.