Automazione I

16 Gennaio 2014

Esercizio 1

Un committente richiede di progettare lo scheduling dei task di un sistema di automazione industriale a livello di campo, di coordinamento e di supervisione. Il mono-processore che deve elaborare l'algoritmo di scheduling è caratterizzato da una frequenza di elaborazione di 20 Mhz. Ipotizzando che ad ogni colpo di clock del mono-processore corrisponda una time unit (t.u.), è necessario portare a termine i seguenti task periodici a livello di campo:

1. ogni 0.15 μs viene effettuata la misura dei sensori, impiegando 50 ns;
2. ogni 0.25 μs viene elaborata la legge di controllo in un tempo 0.1 μs;
3. ogni 0.45 μs vengono attuate le azioni di controllo in un tempo 50 ns.

Ai task suddetti si aggiunge un task di coordinamento che deve intervenire con un intervallo minimo di 0.75 μs e durare al massimo 0.1 μs. I task periodici sono totalmente indipendenti l'uno dall'altro e non c'è alcuna dipendenza logica o temporale tra essi. Tutti i task periodici devono essere gestiti con una modalità di scheduling hard real time. Il sistema di automazione è supervisionato da un sistema informativo che, in maniera deterministica ma non periodica, interroga il mono-processore per fare il log del suo stato. In particolare, l'activation time della prima occorrenza di tale task non periodico è all'istante $a_5(1) = 1$ μs, il computation time è pari a $C_5(1) = 50$ ns e la deadline assoluta è pari a $D_5(1) = 2.25$ μs. Rispetto ai task descritti precedentemente, questo task deve essere servito con una modalità di scheduling soft real time.

- Verificare che il problema di scheduling hard real time non sia inammissibile.
- Nel caso in cui il problema non sia inammissibile, verificare se sussiste la condizione sufficiente per l'ammissibilità del problema con *RMPO* (tenendo conto che $2^{1/4} \simeq 1.1892$):
  - se la condizione sufficiente *è verificata*, mostrare il risultato dello scheduling usando *RMPO* come algoritmo di task scheduling hard real time e un servizio in background con scheduling di tipo *FIFO* per i task soft real time;
  - se la condizione sufficiente *NON è verificata*, mostrare il risultato dello scheduling usando *EDF* come algoritmo di task scheduling hard real time e un servizio in background con scheduling di tipo *FIFO* per i task soft real time.
- Determinare se il task soft real time viene eseguito entro la deadline assoluta.

Esercizio 2

Rispondere a ciascuna domanda con un testo al massimo di 10 righe (e, se del caso, 2 formule):

1. Illustrare il metodo di accesso al mezzo in una rete Fieldbus, giustificandolo rispetto alle esigenze e alle specifiche di trasmissione in una rete real time.
2. Introdurre il fenomeno del windup dell'azione integrale e descrivere un possibile accorgimento realizzativo per minimizzarne gli effetti.
3. Descrivere vantaggi e svantaggi dell'uso di una dinamo tachimetrica rispetto a quello di un encoder incrementale per la misura della velocità angolare di un motore elettrico.

Esercizio 3

In Figura 1 è rappresentata un'operazione di *pick-and-place* di un oggetto tra due stazioni S_{in} e S_{out} di una cella di lavoro automatizzata. Essendo le stazioni tra loro distanti, l'operazione è eseguita in modo coordinato da due robot R_1 e R_2 che si scambiano l'oggetto in un punto di incontro P_{meet}. A tal fine, ciascun robot è dotato di un gripper che può essere nello stato aperto o chiuso. L'operazione può essere scomposta in una sequenza di cinque fasi elementari definite sull'oggetto:

- prelievo (*pick*) dalla stazione di ingresso S_{in} da parte del robot R_1
- trasporto dalla stazione S_{in} al punto di incontro P_{meet}
- scambio dal robot R_1 al robot R_2
- trasporto dal punto di incontro P_{meet} alla stazione di uscita S_{out}
- deposito (*place*) nella stazione S_{out} da parte del robot R_2.

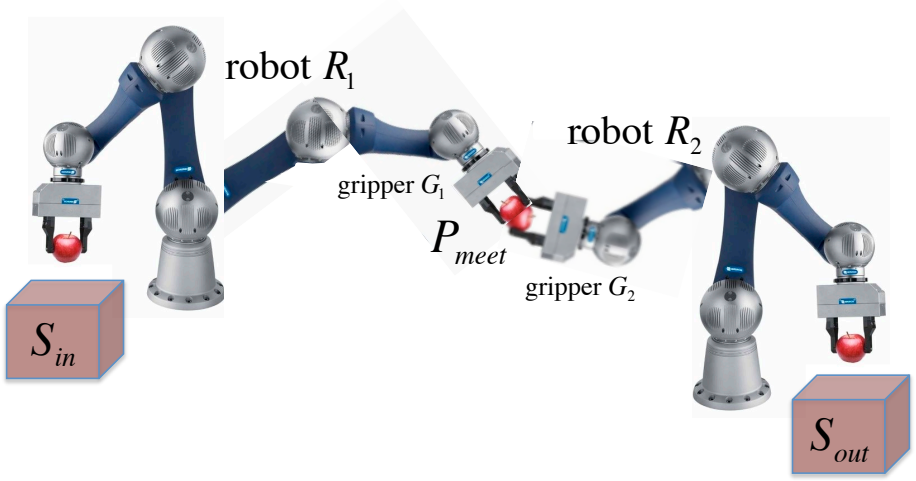

Figura 1: Una cella con due robot che si coordinano in un'operazione di *pick-and-place*

I due robot sono a disposizione esclusiva di questa operazione. Descrivere il sistema mediante una rete di Petri (non temporizzata), modellando in particolare le azioni dei gripper nelle varie fasi. Indicare una marcatura iniziale appropriata della rete.

[180 minuti; libri aperti]

Soluzioni

16 Gennaio 2014

Esercizio 1

Dato che la frequenza di elaborazione del mono-processore è di 20 Mhz, il singolo colpo di clock ha una durata di 50 ns. Per ipotesi, ad ogni colpo di clock corrisponde una time unit (t.u.), pertanto vale la corrispondenza 1 t.u. = 50 ns. I quattro task periodici saranno così caratterizzati:

- A_1 (misura): $T_1 = 3$ t.u., $C_1 = 1$ t.u.;
- A_2 (elaborazione della legge di controllo): $T_2 = 5$ t.u., $C_2 = 2$ t.u.;
- A_3 (attuazione delle azioni di controllo): $T_3 = 9$ t.u., $C_3 = 1$ t.u.;
- A_4 (coordinamento): $T_4 = 15$ t.u., $C_4 = 2$ t.u.

Il quinto task aperiodico sarà così caratterizzato:

- A_5 (supervisione): $a_5(1) = 20$ t.u., $C_5(1) = 1$ t.u., $D_5(1) = 45$ t.u.

La verifica di non inammissibilità si effettua calcolando il fattore di utilizzazione dei task periodici hard real time:

$$U = \frac{1}{3} + \frac{2}{5} + \frac{1}{9} + \frac{2}{15} = \frac{15 + 18 + 5 + 6}{45} = \frac{44}{45} \simeq 0.978 < 1.$$

La condizione necessaria è dunque soddisfatta. Verifichiamo quindi la condizione sufficiente per l'applicabilità dell'algoritmo *RMPO* per un numero di task periodici $n = 4$:

$$U_{lsm}(RMPO) = n\left(2^{1/n} - 1\right) = 4\left(2^{1/4} - 1\right) \simeq 0.7568.$$

Dato che $U > U_{lsm}$ la condizione sufficiente **non** è verificata.

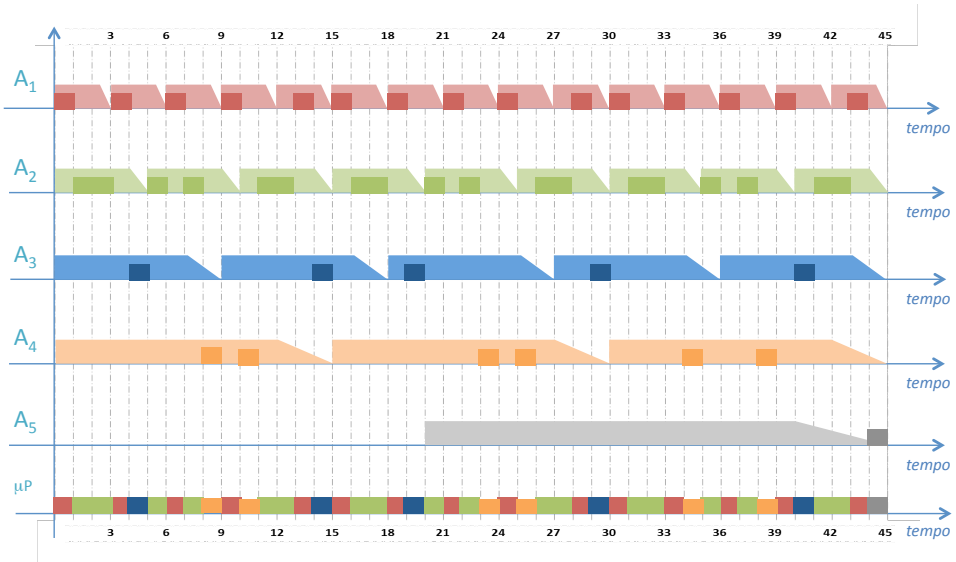

Figura 2: Soluzione del task scheduling con algoritmi *EDF* (hard real time) e *FIFO* (soft real time)

Pertanto per lo scheduling dei quattro task periodici si utilizzerà l'algoritmo *EDF* (la cui soluzione sappiamo già esistere). La Fig. 2 mostra la soluzione del problema posto ottenuta utilizzando *EDF* come algoritmo di task scheduling hard real time e un servizio in background con scheduling di tipo *FIFO* per i task soft real time. Dalla figura si evince che il task soft real time può essere portato a termine entro la deadline assoluta.

Esercizio 3

Una rete di Petri che modellizza l'operazione richiesta è mostrata in Fig. 3. Ciascun gripper G_i ($i = 1, 2$) può trovarsi in una tra due possibili condizioni: {open, close}. La marcatura iniziale mostrata corrisponde alla situazione in cui i due robot sono disponibili nello stato opportuno (R_1 in S_{in} con gripper $G_1 = \{\text{open}\}$ e R_2 in P_{meet} con gripper $G_2 = \{\text{closed}\}$) e un oggetto è pronto in S_{in}.

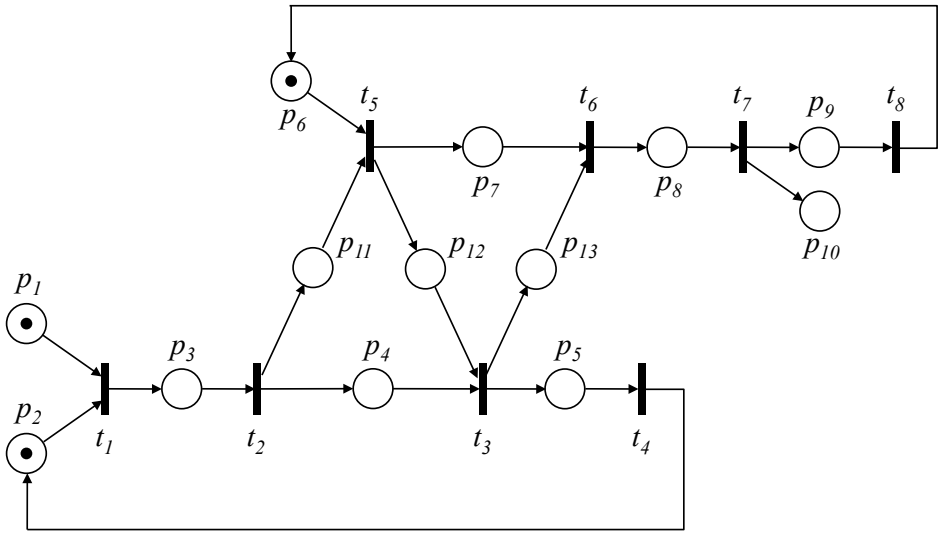

Figura 3: Rete di Petri relativa all'operazione di *pick-and-place* coordinata con due robot

Il significato dei 13 posti è il seguente:

p_1 = oggetti in S_{in}
p_2 = robot R_1 in S_{in} con gripper $G_1 = \{\text{open}\}$
p_3 = robot R_1 in S_{in} con gripper $G_1 = \{\text{closed}\}$
p_4 = robot R_1 in P_{meet} con gripper $G_1 = \{\text{closed}\}$
p_5 = robot R_1 in P_{meet} con gripper $G_1 = \{\text{open}\}$
p_6 = robot R_2 in P_{meet} con gripper $G_2 = \{\text{open}\}$
p_7 = robot R_2 in P_{meet} con gripper $G_2 = \{\text{closed}\}$
p_8 = robot R_2 in S_{out} con gripper $G_2 = \{\text{closed}\}$
p_9 = robot R_2 in S_{out} con gripper $G_2 = \{\text{open}\}$
p_{10} = oggetti in S_{out}
p_{11} = richiesta a R_2 di chiusura gripper G_2 (*posto di sincronizzazione*)
p_{12} = richiesta a R_1 di apertura gripper G_1 (*posto di sincronizzazione*)
p_{13} = abilitazione al movimento di R_2 (*posto di sincronizzazione*)

Il significato delle 8 transizioni è il seguente:

t_1 = close G_1
t_2 = move R_1: $S_{in} \rightarrow P_{meet}$
t_3 = open G_1
t_4 = move R_1: $P_{meet} \rightarrow S_{in}$
t_5 = close G_2
t_6 = move R_2: $P_{meet} \rightarrow S_{out}$
t_7 = open G_2
t_8 = move R_2: $S_{out} \rightarrow P_{meet}$

La rete è marcata inizialmente con un solo token (= un solo oggetto) nella stazione di ingresso (posto p_1), ma funzionerebbe in modo corretto anche se si trovasse a gestire (sequenzialmente) un numero arbitrario n di oggetti in ingresso. A tale scopo, è sufficiente mettere n token in p_1. Per svincolarsi dal considerare un numero prefissato di oggetti, si può aggiungere una transizione t_{in} (senza posti in ingresso, quindi sempre abilitata) in ingresso a p_1 e togliere da questo posto il token iniziale. Analogamente, si può aggiungere una transizione t_{out} in uscita al posto p_{10}, così da togliere gli oggetti ogni volta che raggiungano la stazione di uscita. La rete (ossia il sistema) funzionerà quindi in modalità permanente.

* * * * *