Architettura degli Elaboratori, a.a. 2006-07

Note sull'utilizzazione di componenti logici di tipo memoria

Queste note precisano e completano il contenuto nel Cap. III, sez. 7 delle Dispense, in particolare per quanto riguarda la valutazione del tempo di accesso in memoria (sez. 7.2), la realizzazione di memorie modulari, e la valutazione del ritardo di operazioni elementari in presenza scritture in memoria.

1. Tempo di accesso di un componente logico memoria

Si assume il contenuto della sez. 7.1 del Cap. III per quanto riguarda la definizione e l'implementazione di un componente logico di memoria di tipo RAM (ROM). Nella sez. 7.2 è indicato come la realizzazione del commutatore di uscita (selezionatore d'ingresso) di una memoria sia effettuata mediante particolari realizzazioni di porte logiche con un elevato numero di ingressi (bus sincroni, realizzazione wired-or). Tutto questo è valido da un punto di vista tecnologico, ma, agli effetti dello svolgimento di problemi ed esercizi, è conveniente riferirsi ad una *realizzazione equivalente in termini di normali porte AND/OR*: questa realizzazione, ed il modo di valutare il tempo di accesso delle memorie, sono spiegati nel seguito.

1.1. Numero massimo di ingressi per porta nelle reti combinatorie

In generale, per una porta logica AND/OR è fissato un *numero massimo di ingressi*, N_0, ad esempio $N_0 = 8$. Il valore del massimo ritardo di stabilizzazione della porta, t_p, vale per un numero di ingressi $N \leq N_0$, in quanto per $N > N_0$ il ritardo di stabilizzazione aumenterebbe linearmente con forte pendenza. In pratica, non esistono porte con $N > N_0$. Di questa caratteristica tecnologica occorre tenere conto nella realizzazione di una *qualunque rete combinatoria* [1]; in particolare, nella realizzazione di commutatori (selezionatori) con un elevato numero di ingressi, come si ha anche (ma non solo) nell'implementazione dei componenti logici memoria.

Se di una funzione è stata ricavata l'implementazione come rete combinatoria *a due livelli di logica*, questa realizzazione vale sempre *da un punto di vista concettuale*, ma, dal punto di vista tecnologico, occorre tener conto del vincolo che, per ogni porta, deve essere $N \leq N_0$. Se questa condizione non si verifica, occorre *sostituire la singola porta con una struttura ad albero di arietà N_0,* che, di tutte le possibili realizzazioni, è quella che garantisce il minimo ritardo di stabilizzazione.

[1] Ad esempio, nelle reti combinatorie ω_{PC} e σ_{PC} nei casi in cui il numero complessivo delle variabili dello stato interno e delle variabili di condizionamento sia relativamente elevato.

In generale, quindi, *il ritardo di stabilizzazione comportato dalla realizzazione di un termine AND/OR varia in modo logaritmico con il numero delle sue variabili d'ingresso*.

1.2. Ritardo di stabilizzazione di un commutatore e tempo di accesso di una memoria

Si consideri un commutatore a *n* ingressi principali, $x_0, \ldots, x_{n-1}$, *m* ingressi secondari (variabili di controllo) $\alpha_0, \ldots, \alpha_{m-1}$, con $m = \lceil lg_2\, n \rceil$, e uscita *z*. Esso è definito dalla funzione:

z = **case** $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ **of**
0 0 … 0 : x_0
0 0 … 1 : x_1
…
stringa binaria equivalente al valore naturale *j* : x_j
…
1 1 ... 1 : x_{n-1}

Questo corrisponde (vedi anche Cap. II, sez. 1) alla realizzazione del commutatore come una rete combinatoria che, *concettualmente*, è a due livelli di logica, dove:

- il livello AND è costituito da *n* porte AND, la *j-esima* delle quali ha *m + 1* ingressi: uno è *xj* e gli altri *m* sono le variabili di controllo opportunamente affermate o complementate in modo da dare luogo alla stringa binaria equivalente al valore naturale *j;*
- il livello OR è costituito da un porta OR con *n* ingressi.

A questa realizzazione concettuale corrisponde una *realizzazione effettiva* in cui ogni porta AND/OR può essere sostituita da una *struttura ad albero* se il suo numero di ingressi è > N_0.

In particolare:

- il livello AND è costituto da *n* alberi, ognuno di arietà N_0 e di profondità:

$$\lceil \lg_{N_0} (m + 1) \rceil = \lceil \lg_{N_0} (\lceil \lg_2 n \rceil + 1) \rceil$$

- il livello OR è costituito da un albero di arietà N_0 e di profondità:

$$\lceil \lg_{N_0} (n) \rceil$$

Indicando con t_p il ritardo di una porta AND/OR con un numero di ingressi $N \leq N_0$, il ritardo di stabilizzazione di un commutatore ad n ingressi è quindi dato da:

$$T_K (n, N_0, t_p) = t_p (\lceil \lg_{N_0} (\lceil \lg_2 n \rceil + 1) \rceil + \lceil \lg_{N_0} (n) \rceil)$$

Questa formula generale esprime, in particolare, il *tempo di accesso di un componente logico memoria di capacità C* (= n), valutato come il massimo ritardo di stabilizzazione per l'operazione di lettura:

$$t_a (C, t_p) = T_K (C, N_0, t_p) = t_p (\lceil \lg_{N_0} (\lceil \lg_2 C \rceil + 1) \rceil + \lceil \lg_{N_0} (C) \rceil)$$

Come si vede, dei due addendi domina nettamente il secondo (ritardo livello OR, con ordine di grandezza logaritmico): è questo che giustifica l'adozione di una tecnologia ad hoc per realizzare la funzione OR a molti ingressi. Comunque, tale tecnologia non modifica la valutazione effettuata con la formula di cui sopra.

Ovviamente, quando si abbia a che fare con numeri semplici il valore del ritardo può anche essere calcolato semplicemente contando le profondità degli alberi dei livelli AND e OR.

Per un selezionatore (e quindi per l'operazione di scrittura di un componente logico memoria), avendo solo il livello AND, si ha:

$$T_S (n, N_0, t_p) = t_p \lceil \lg_{N_0} (\lceil \lg_2 n \rceil + 1) \rceil$$

Può convenire trasformare le formule di cui sopra utilizzando logaritmi in base 2:

$$T_K (n, N_0, t_p) = t_p (\lceil \lg_2 (\lceil \lg_2 n \rceil + 1) / \lg_2 (N_0) \rceil + \lceil \lg_2 (n) / \lg_2 (N_0) \rceil)$$

$$T_S (n, N_0, t_p) = t_p \lceil \lg_2 (\lceil \lg_2 n \rceil + 1) / \lg_2 (N_0) \rceil$$

Ad esempio, vediamo alcuni tempi di accesso di componenti logici memoria con diverse capacità C, con C potenza di due e $N_0 = 8$:

$$t_a (C, t_p) = t_p (\lceil \lg_2 (\lg_2 C + 1) / 3 \rceil + \lceil \lg_2 (C) / 3 \rceil)$$

C = 64 $t_a = 3\ t_p$

C = 256 $t_a = 5\ t_p$

C = 512 $t_a = 5\ t_p$

C = 1K $t_a = 6\ t_p$

C = 64K $t_a = 8\ t_p$

C = 1M $t_a = 9\ t_p$

C = 64M $t_a = 11\ t_p$

C = 256M $t_a = 12\ t_p$

C = 1G $t_a = 12\ t_p$

C = 4G $t_a = 13\ t_p$

Questi valori mostrano quanto sia efficiente la realizzazione ad albero.

L'effettivo valore di questi ritardi dipende, ovviamente, dal valore di t_p, che è fortemente variabile con il tipo di tecnologia adottata. Ad esempio, allo stato attuale, come ordine di grandezza si può avere per *memorie statiche* $t_p \cong 0{,}1 \div 1$ nsec, mentre per *memorie dinamiche* $t_p \cong 10 \div 100$ nsec.

2. Casi di utilizzazione e relative valutazioni

2.1. Memorie realizzate a partire da memorie preesistenti

Di regola, le memorie di grande capacità sono realizzate a partire da componenti logici base dati, che vanno combinati opportunamente in quella che prende il nome di **memoria modulare**.

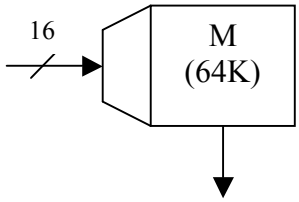
Ad esempio, supponiamo di voler realizzare una memoria M, di tipo ROM, avente capacità 64K parole, a partire dalla disponibilità di memorie ROM di capacità 1K parole.

La realizzazione della memoria da 64K si ottiene mediante 64 *moduli di memoria* da 1K tra loro indipendenti, ed ottenendo di M l'uscita mediante un commutatore a 64 ingressi, come mostrato nella figura a pagina seguente.

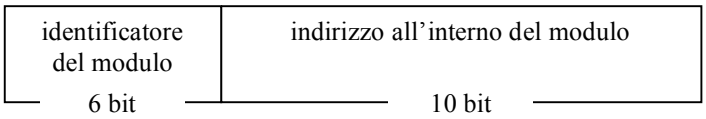
Per una memoria RAM, analoga struttura va realizzata per il selezionatore d'ingresso.

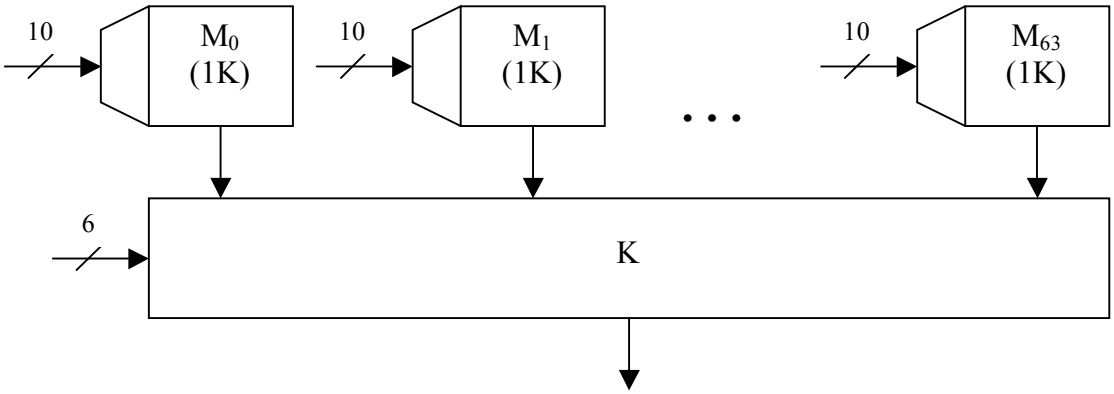
È importante notare come è utilizzato l'indirizzo di 16 bit:

- i 10 bit meno significativi (**indirizzo all'interno del modulo di memoria**) indirizzando le memorie da 1K operanti in parallelo,
- i 6 bit più significativi (**identificatore del modulo di memoria**) comandano il commutatore di uscita.



indirizzo:





I moduli di memoria da 1K hanno un loro tempo di accesso noto, t_{ao}. Il tempo di acceso complessivo si ottiene come:

$$t_a = t_{ao} + T_K$$

Nell'esempio: $T_K = 3\ t_p$. Se $t_{ao} = 7\ t_p$ (si noti che tale valore *non* va ricavato utilizzando la formula della sez. precedente, in quanto rappresenta un dato del problema associato alle memorie che vengono fornite; esso viene espresso come multiplo di t_p solo per comodità), si ottiene $t_a = 10\ t_p$.

2.2. Valutazione di operazioni che comportano una scrittura in memoria

Supponiamo che il microprogramma di una certa unità contenga l'operazione elementare, che comporti una *lettura* da un a memoria A interna alla PO:

$$A[J] + B \rightarrow C$$

Il suo ritardo di stabilizzazione, agli effetti della valutazione del ritardo della funzione σ_{PO}, è dato dalla somma del tempo di accesso alla memoria A (stabilizzazione del commutatore di uscita K_A), del ritardo di una ALU e dei ritardi di eventuali commutatori presenti sull'ingresso di C, e/o di ALU e/o sugli ingressi dello stesso K_A.

Quindi, in presenza di lettura in memoria, tutte le stabilizzazioni delle risorse in gioco in una stessa operazione elementare avvengono in serie e, di conseguenza, i ritardi interessati si sommano.

Si consideri invece la seguente operazione elementare, che comporta una *scrittura* nella memoria interna alla PO:

$$C + B \rightarrow A[J]$$

Per la valutazione del suo ritardo, occorre ora considerare che la stabilizzazione del selezionatore d'ingresso S_A, più l'eventuale stabilizzazione di un commutatore su S_A, avvengono *in parallelo* alla stabilizzazione della ALU e di altri eventuali commutatori sull'ingresso della ALU stessa e sull'ingresso del dato di A. Si ricordi infatti (Cap. III, sez. 7.1) che S_A opera solo sulla variabile di controllo β_A per abilitare la scrittura in memoria A, e non sul dato d'ingresso di A.

Di conseguenza, per il ritardo di stabilizzazione di questa operazione elementare occorre considerare il *massimo* dei due ritardi tra loro in parallelo.

2.3. Esempi

Come esempi dei casi illustrati nelle sez. 2.1.e 2.2, si vedano gli esercizi del Cap. III, sez. 7, quelli raccomandati in "Esercizi sulla strutturazione firmware", e l'esercizio svolto contenuto nella note "Correzione della prima prova di verifica intermedia del 2004-05".

Si rifletta anche sulla possibilità di realizzazione di "grosse" reti combinatorie, con un procedimento di complessità indipendente dal numero di variabili, utilizzando direttamente memorie (ROM) per memorizzare la tabella di verità.

3. Organizzazioni di memoria modulare

3.1. Organizzazione sequenziale

La memoria esemplificata nella sez. 2.1 rappresenta una delle due modalità base con cui si può organizzare una memoria modulare, detta **organizzazione sequenziale**.

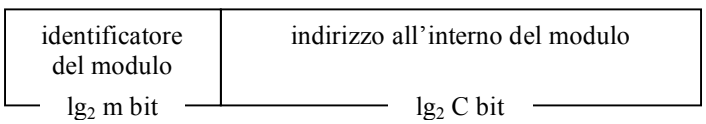
La memoria modulare sequenziale è caratterizzata dalla distribuzione degli indirizzi sequenzialmente all'interno di un modulo: quando si esaurisce la capacità di un modulo, si passa a distribuire gli indirizzi sequenzialmente nel modulo successivo.

Formalmente, per una memoria con *m* moduli, sia *C* è la capacità del singolo modulo. Dato un indirizzo *ind*, allora l'identificatore del modulo *id* e l'indirizzo all'interno del modulo *displ* si ottengono rispettivamente come quoziente e resto della divisione intera di *ind* per *C*:

$$id = ind \,/\, C \quad , \quad displ = ind \,\%\, C$$

Per *C* potenza di due, come di regola, *id* e *displ* sono ottenuti direttamente da campi dell'indirizzo con la nota regola:

indirizzo:



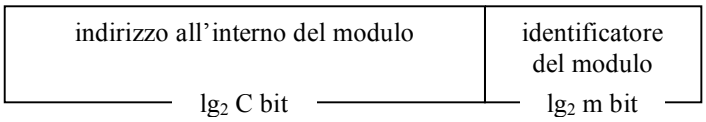
3.2. Organizzazione interallacciata

L'organizzazione alternativa di memoria modulare, che verrà usata in altre parti del corso (in particolare, a proposito delle memorie cache), è detta **organizzazione interallacciata**, per la quale si ha:

$$id = ind \,\%\, m \quad , \qquad disp = ind \,/\, m$$

cui corrisponde:

indirizzo:



Ciò significa che, nella memoria modulare interallacciata, *m* parole eventi indirizzi consecutivi sono allocate in altrettanti moduli di memoria distinti e consecutivi.

Poiché tale organizzazione viene usata anche (e soprattutto) per accedere in parallelo a tutte le parole ad un blocco di *m* parole consecutive, tutte le uscite dai moduli di memoria sono anche rese disponibili indipendentemente.

L'esempio di una memoria interallacciata di capacità 1M parole realizzata con 8 moduli di 128K parole ciascuno è:

indirizzo: