Architettura degli Elaboratori, 2007-08

Esercitazione 1 di verifica

Soluzione: mercoledì 10 ottobre

Domanda 1

Realizzare una rete combinatoria avente quattro variabili booleane di ingresso *a, b, x, y* e due variabili booleane di uscita *z, w*. La funzione delle uscite è così definita:

- se *a b = 0 0*, allora *z* è uguale alla somma di *x* e *y* e *w* è uguale al riporto dell'addizione di *x* e *y*;
- se *a b = 0 1*, allora $z = x \oplus y$ e *w = 0*;
- se *a = 1*, allora *z = y* e *w = x*.

Dare almeno due realizzazioni, delle quali una ricavata dalla tabella di verità, ed almeno una ricavata dalla definizione in forma algoritmica.

In tutte le soluzioni valutare il tempo di stabilizzazione della rete in funzione del ritardo t_p di una porta logica con al massimo 4 ingressi.

Domanda 2

Realizzare una rete sequenziale (*contatore modulo 4*) così definita:

- ha una variabile booleana di ingresso *x* e due variabili booleane di uscita *z, w*;
- ogni volta che *x* assume il valore 1, il valore della configurazione delle uscite *zw*, considerata come numero naturale di due bit, viene incrementato di 1 modulo 4.

Nel caso di modello matematico di Moore, definire la funzione delle uscite e la funzione di transizione dello stato interno nei seguenti due casi:

1. partendo dal grafo di stato,
2. utilizzando componenti logici standard,

e, nei due casi, valutare il ciclo di clock della rete in funzione del ritardo t_p di una porta logica con al massimo 4 ingressi, sapendo che il ritardo di stabilizzazione di una ALU è uguale a $4t_p$ e che l'impulso di clock ha durata t_p.

Ripetere quanto sopra per una realizzazione della rete in accordo al modello matematico di Mealy.

Domanda 3

Realizzare una rete sequenziale così definita:

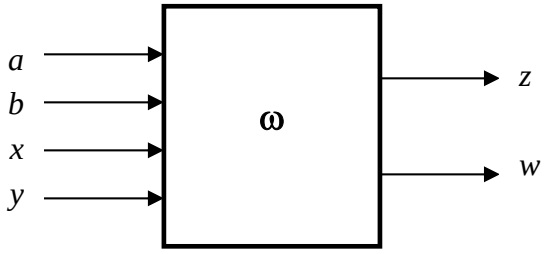
- ingressi *A* (32 bit), *B* (5 bit), *C* (32 bit), *D* (32 bit), *E* (6 bit);
- uscite $Z_0, Z_1, \ldots, Z_{63}$ tutte a 32 bit;
- se il bit *B*-esimo di *A* vale 0, allora l'uscita identificata dal valore di *E* assume il valore di $C + D$ e tutte le altre uscite rimangono inalterate;
- se il bit *B*-esimo di A vale 1, allora l'uscita identificata dal valore di *E* assume il valore di $C - D$ e tutte le altre uscite rimangono inalterate;

e valutarne il ciclo di clock nelle stesse ipotesi della Domanda 2.

Soluzione

Domanda 1

La rete combinatoria



implementa una trasformazione (ω) da stati d'ingresso (combinazioni di *a, b, x, y)* a stati di uscita (combinazioni di *z, w*), tale che ad ogni stato d'ingresso corrisponde uno ed un solo stato di uscita. La funzione è completamente definita dalle specifiche, che possono essere espresse in un formalismo algoritmico:

z = **case** *a b* **of**	*w* = **case** *a b* **of**
0 0 : *somma (x, y)*	0 0 : *riporto_addizione (x, y)*
0 1 : $x \oplus y$	0 1 : *0*
1 – . *y*	1 – . *x*

Ricordando l'implementazione dell'addizione tra due bit, si ha:

z = **case** *a b* **of**	*w* = **case** *a b* **of**
0 0 : $x \oplus y$	0 0 : *x y*
0 1 : $x \oplus y$	0 1 : *0*
1 – . *y*	1 – . *x*

da cui già si vede, in particolare, che il valore di *z* è indipendente da quello di *b*.

Soluzione 1 – Espressioni logiche ricavate dalla tabella di verità

Dalla definizione si ricava la tabella di verità:

a	*b*	*x*	*y*	*z*	*w*
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	0
1	–	0	0	0	0
1	–	0	1	1	0
1	–	1	0	0	1
1	–	1	1	1	1

dalla quale si ottengono le espressioni logiche in *forma normale SP* delle variabili di uscita (ricavate indipendentemente l'una dall'altra), introducendo subito alcune semplificazioni in seguito all'applicazione della proprietà distributiva e dei teoremi della complementazione e dell'unione-intersezione:

$$z = \bar{a}\ \bar{b}\,\bar{x}\ y + \ \bar{a}\ \bar{b}\ x\,\bar{y} + \ \bar{a}\,b\,\bar{x}\,y + \bar{a}\,b\,x\,\bar{y} + \ a\,\bar{x}\,y + a\,x\,\bar{y} =$$
$$= \ \bar{a}\ \bar{x}\ y\left(\bar{b} + b\right) + \ \bar{a}\,x\,\bar{y}\left(\bar{b} + b\right) + \ a\,y\,(\bar{x} + x) =$$
$$= \ \bar{a}\ \bar{x}\ y + \ \bar{a}\,x\,\bar{y} + \ a\,y$$

$$w = \bar{a}\,\bar{b}\,x\,y + a\,x\,\bar{y} + a\,x\,y = \ \bar{a}\,\bar{b}\,x\,y + a\,x\,(\bar{y} + y) = \ \bar{a}\,\bar{b}\,x\,y + a\,x$$

L'implementazione in termini di porte AND, OR, NOT è immediata. Poiché tutte le porte hanno un numero di ingressi minore o uguale a 4, si tratta di una rete *a due livelli di logica*, con un primo livello di 5 porte AND in parallelo (3 per *z*, 2 per *w*) ed un secondo livello di 2 porte OR in parallelo (una per *z*, una per *w*). Il ritardo di stabilizzazione è quindi:

$$T_\omega = 2\ t_p$$

Osservazioni:

1) *la riduzione delle espressioni logiche* non è tanto importante agli effetti del numero di porte impiegate, quanto per il fatto di *non far aumentare il numero di livelli di logica*; in particolare, l'espressione non ridotta di *z* comporterebbe l'uso di una funzione OR a 6 ingressi che, con i dati del problema, dovrebbe essere realizzata con un albero di profondità due;

2) come c'era da aspettarsi, nell'espressione logica di *z* è effettivamente contenuta quella di $x \oplus y$, per cui potremmo anche trasformare ulteriormente l'espressione in:

$$z = \bar{a}\ (x \oplus y) + a\ y$$

ma questo non porterebbe vantaggio rispetto all'espressione in forma normale SP, che è caratterizzata dal minimo ritardo di stabilizzazione;

3) non è necessario costruire un'unica tabella di verità per tutte le variabili di uscita; trattandosi di funzioni indipendenti, siamo liberi di costruire *una tabella separata per ogni variabile di uscita*. Nel nostro caso, la tabella per la sola variabile di uscita *z* non avrebbe previsto la variabile d'ingresso *b*.

Soluzione 2 – Schema della rete ricavato direttamente dalla descrizione algoritmica usando componenti logici standard

Essendo molto semplice la descrizione nel formalismo algoritmico, possiamo ricavarne direttamente l'implementazione della rete in termini di *componenti logici standard:* nel nostro caso, commutatori, confrontatori, porte AND.

Consideriamo la definizione algoritmica della variabile di uscita *z*, che può essere scritta come:

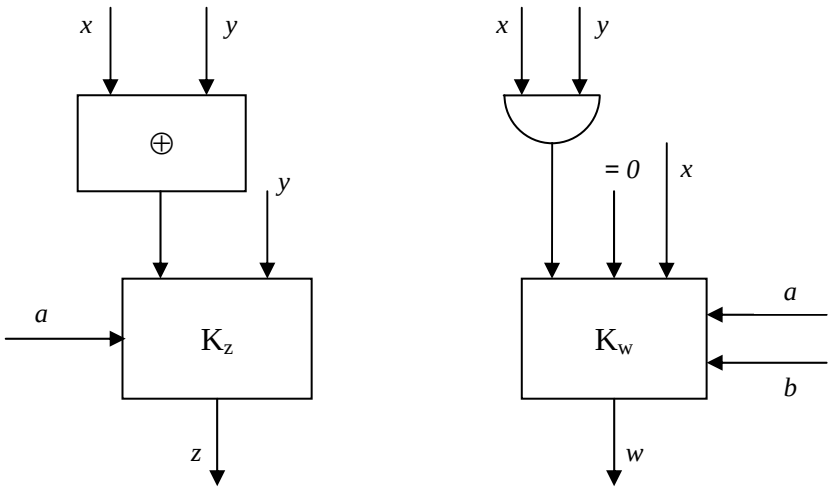
$$z = \textbf{if not } a \textbf{ then } x \oplus y \textbf{ else } y$$

La presenza di un costrutto *if then else* sta a significare che esiste un *commutatore* a due ingressi; questi assumono il valore delle funzioni presenti nei rami *then* ed *else* (rispettivamente, $x \oplus y$ e y); la variabile di controllo assume il valore di *a*.

Analogamente si ragiona per la variabile *w*, per la quale si ha un commutatore a tre ingressi (*case*) con variabili di controllo *a, b*. L'ingresso di valore identicamente uguale a 0 è implementato da una costante "cablata in hardware".

La rete combinatoria è mostrata nella figura seguente [1]:

[1] Tutti gli archi con lo stesso nome sono in realtà biforcazioni di uno stesso arco. Disegnarli in modo distinto permette di rendere più semplice e leggibile il disegno. Nel corso ci atterremo sempre a questa convenzione.



Questa soluzione mostra come, spesso, sia possibile semplificare notevolmente la sintesi di una rete combinatoria, sintesi che, partendo dalla tabella di verità, è un procedimento di *complessità esponenziale* nel numero delle variabili d'ingresso. *Quando fattibile*, il procedimento che consiste nel derivare la struttura della rete direttamente dalla specifica (a parole o, meglio, in un formalismo algoritmico) è, invece, di complessità *costante* nel numero delle variabili e *lineare* nel numero delle funzioni (funzioni nei rami *then*, *else*, o nei rami di un *case*, e predicati): questa ridotta complessità si può avere a condizione che si abbia *una conoscenza a priori delle reti combinatorie corrispondenti a tali funzioni*, tipicamente corrispondenti a componenti *standard* (ALU, K, S, ⊕, AND, OR, NOT) o comunque a reti già note.

Per contro, il *ritardo di stabilizzazione* di una soluzione del genere è spesso maggiore di quello ottenuto ricavando le espressioni logiche SP (nel caso limite più favorevole, è uguale). In questo esempio, siamo appunto in un caso con ritardo maggiore.

La parte di rete che calcola *z* (ritardo determinato dal confrontatore e da K_z in serie tra loro) e quella che calcola *w* (ritardo determinato dalla porta AND e da K_w in serie tra loro) si stabilizzano *in parallelo*, per cui occorre determinare quella avente ritardo di stabilizzazione maggiore. La prima ha ritardo di stabilizzazione $4t_p$, in quanto sia il confrontatore che il commutatore sono reti a due livelli di logica, mentre la seconda parte ha ritardo $3t_p$. Il ritardo di stabilizzazione della rete deve essere il *massimo* possibile, quindi è dato da:

$$T_\omega = 4t_p$$

che rappresenta il minimo intervallo tra due istanti discreti della sequenza temporale, che permette di recepire correttamente tutte le variazioni degli ingressi.

Quello della soluzione 2 è il tipico approccio che, nella progettazione di unità di elaborazione a livello firmware, viene adottato nella sintesi della Parte Operativa.

Soluzione 3 – Espressioni logiche SP ricavate dalla descrizione algoritmica

Spesso è possibile, partendo dalla definizione delle funzioni in forma algoritmica, arrivare a ricavare le espressioni logiche in forma SP senza andare incontro ad un procedimento di complessità esponenziale ed, al contempo, ottenendo il minimo ritardo di stabilizzazione.

Nel nostro caso, dalle definizioni algoritmiche di *z* e *w* si ricavano immediatamente, *con un semplice cambiamento di sintassi*, le loro espressioni logiche:

$$z = \bar{a}\ (x \oplus y) + a\ (y)$$

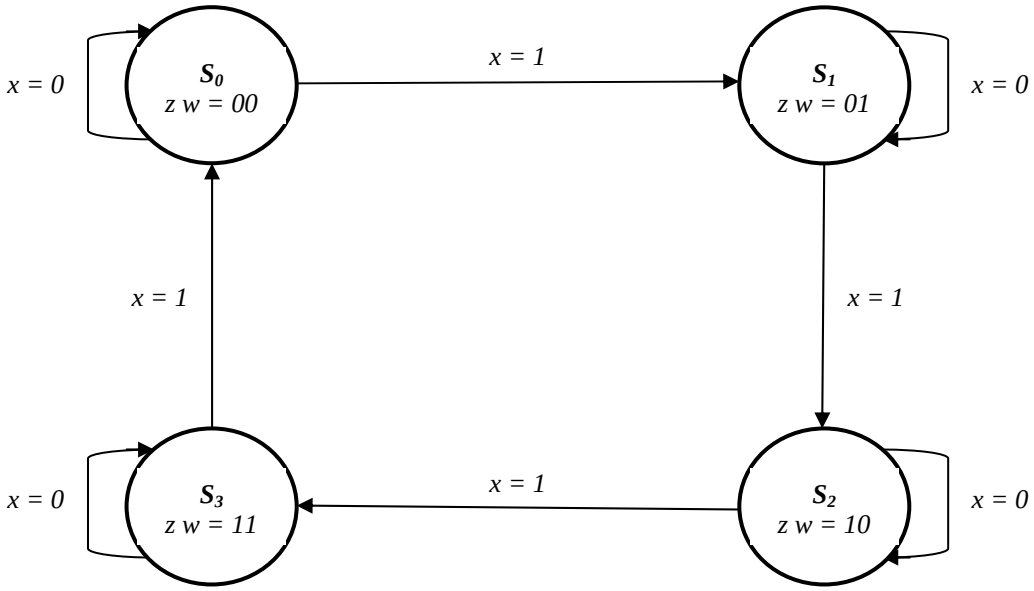
$$w = \bar{a}\bar{b}\ (x\,y) + \bar{a}b\ (0) + a\ (x)$$

sviluppando le quali (in particolare, sostituendo le espressioni logiche delle funzioni utilizzate, come $x \oplus y$) si ottengono esattamente le espressioni logiche della soluzione 1, *senza passare attraverso la tabella di verità.*

Domanda 2

1) Realizzazione a partire dal grafo di stato

Nel caso di un automa secondo il modello matematico di *Moore*, il grafo di stato è il seguente:



Si ha un associazione uno ad uno tra stati interni e stati di uscita. Ogni stato interno è stabile per lo stato d'ingresso $x = 0$, mentre, per $x = 1$, lo stato interno S_i transisce in $S_{(i+1) mod\ 4}$.

L'ovvia codifica degli stati interni mediante due variabili dello stato interno, y_0 e y_1, è quella per cui, nello stato S_i, la configurazione binaria $y_0\, y_1$ è numericamente uguale al valore naturale i.

L'espressione logica della funzione delle uscite è quindi:

$(z, w) = \omega\ (y_0, y_1)$:

$$\begin{cases} z = y_0, \\ w = y_1 \end{cases}$$

Dette Y_0, Y_1 le variabili dello stato successivo, la funzione di transizione dello stato interno $\sigma\ (x, y_0, y_1)$ è espressa dalla seguente tabella di verità:

y_0	y_1	x	Y_0	Y_1
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

dalla quale si ricavano le espressioni logiche in forma normale SP:

$(Y_0, Y_1) = \sigma\ (x, y_0, y_1)$:

$$\begin{cases} Y_0 = \overline{y_0}\ y_1\, x + y_0\, \overline{y_1} + y_0\, y_1\, \overline{x} \\ Y_1 = \overline{y_1}\, x + y_1\, \overline{x} \end{cases}$$

Lo stato interno è memorizzato in un registro impulsato *S* di 2 bit, i cui ingressi sono rappresentati da Y_0, Y_1 e le uscite da y_0, y_1. *Le uscite z, w della rete sono prelevate all'uscita del registro di stato S.*

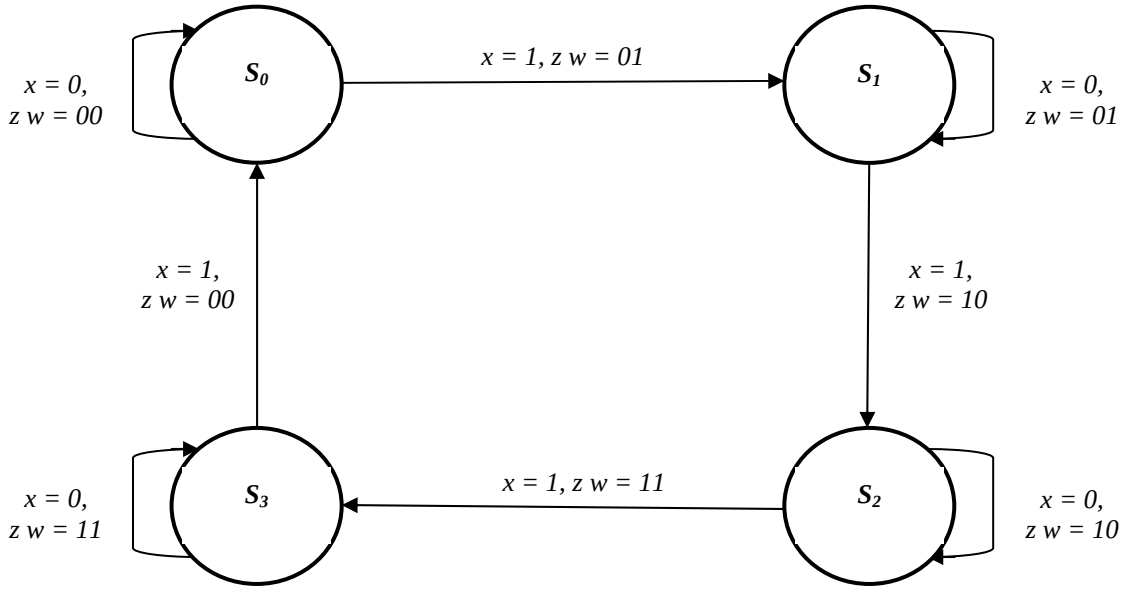
I massimi ritardi di stabilizzazione delle funzioni ω e σ sono dati da:

$$T_\omega = 0 \qquad , \qquad T_\sigma = 2t_p$$

Quindi, il ciclo di clock è dato da:

$$\tau = max\,(\,T_\omega, T_\sigma\,) + \delta = 3t_p$$

Per ottenere l'automa secondo il modello matematico di *Mealy*, osserviamo che questo deve *anticipare* di un ciclo di clock la sequenza di uscita rispetto all'automa equivalente di Moore posto nello stato iniziale equivalente. Il più immediato grafo di strato per ottenere questo comportamento è il seguente:



Ad esempio, poniamoci nello stato interno S_0 con stato d'ingresso $x = 1$. L'automa di Moore produce comunque (cioè, indipendentemente dal particolare stato d'ingresso) uno stato di uscita $z\,w = 00$ e (con quello specifico stato d'ingresso) transisce nello stato S_1, per cui lo stato di uscita $z\,w = 01$ (che rappresenta la "reale" conseguenza al fatto che, nello stato S_0, si è presentato lo stato d'ingresso $x = 1$) sarà prodotto *nel prossimo* ciclo di clock rispetto a quello con S_0 e $x = 1$. L'automa di Mealy , invece, produce lo stato di uscita $z\,w = 01$ *nello stesso* ciclo di clock con S_0 e $x = 1$, oltre a transire in S_1.

Questo metodo di trasformare un automa di Moore nell'equivalente automa di Mealy è sempre applicabile ma, in generale, comporta un numero di stati interni ridondante per l'automa di Mealy. Questo non è però il nostro caso, in quanto, per un contatore modulo 4, il numero minimo di stati è proprio 4.

Con la stessa codifica delle variabili dello stato interno dell'automa di Moore, si ottiene subito che:

- la funzione di transizione dello stato interno $\sigma\,(x, y_0, y_1)$ è identica a quella dell'automa di Moore,
- la funzione delle uscite $\omega\,(x, y_0, y_1)$ coincide con quella di transizione dello stato interno.

In altri termini, si tratta di un particolare automa di Mealy detto "Moore anticipato". *Le uscite z, w della rete sono prelevate all'ingresso del registro di stato S.*

I massimi ritardi di stabilizzazione delle funzioni ω e σ sono dati da:

$$T_\omega = 2t_p \qquad , \qquad T_\sigma = 2t_p$$

Quindi, il ciclo di clock è dato da:

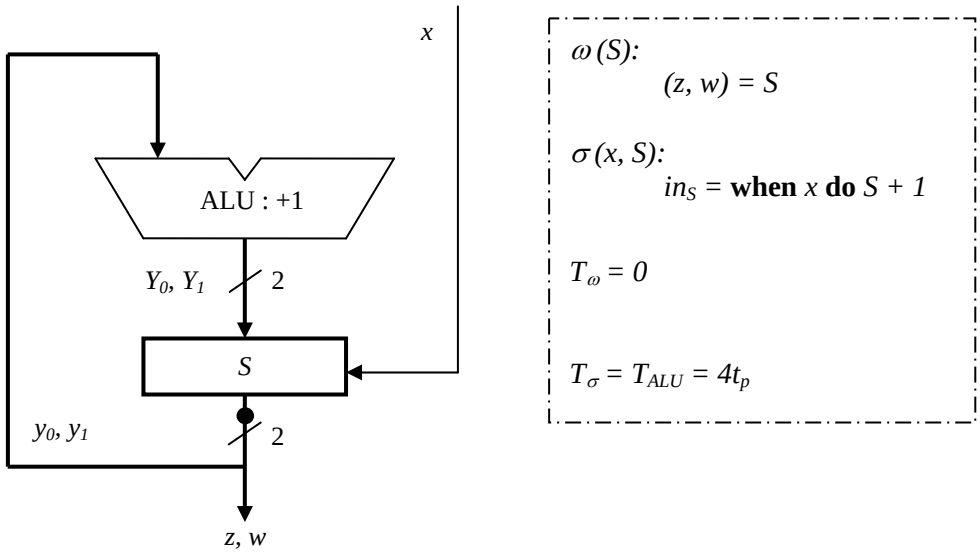
$$\tau = max\,(\,T_\omega, T_\sigma\,) + \delta = 3t_p$$

2) Realizzazione con componenti logici standard

Analogamente al caso delle reti combinatorie, la realizzazione con componenti standard (reti combinatorie standard e registri), se fattibile, comporta una complessità di progettazione ridottissima (costante rispetto al numero di stati d'ingresso, interni e di uscita) al costo di un ciclo di clock più lungo (nel caso più favorevole, uguale).

Nel nostro caso, si tratta di usare, oltre al registro di stato *S* di 2 bit, una ALU capace dell'operazione di incremento su numeri di 2 bit (quando si verifica traboccamento, questo viene ignorato e il risultato vale di nuovo zero); la variabile d'ingresso *x* funge da variabile di controllo per l'abilitazione alla scrittura in *S* (*x* è messa in AND con l'impulso di clock).

Nel caso della rete di *Moore* si ha:



Ricaviamo la rete di *Mealy* a partire da quella di Moore in versione "Moore anticipato", come è stato fatto con il metodo del grafo di stato. Dobbiamo ottenere che:

- la funzione di transizione dello stato interno $\sigma(x, y_0, y_1)$ è identica a quella dell'automa di Moore,
- la funzione delle uscite $\omega(x, y_0, y_1)$ coincide con quella di transizione dello stato interno.

Si rifletta sul fatto che la scrittura

in_S = **when** x **do** $S + 1$

ha il seguente significato, ad ogni ciclo di clock:

S = **if** $(x = 1)$ **then** $S + 1$ **else** S

Quindi, per la rete di Mealy si ha:

$\omega(x, S)$:

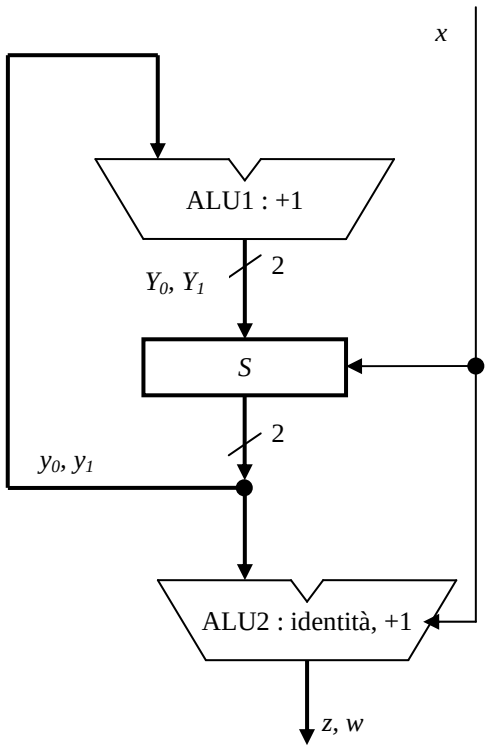
(z, w) = **if not** x **then** S **else** $S + 1$

$\sigma(x, S)$:

in_S = **when** x **do** $S + 1$

Si noti che, se, rispetto alla rete di Moore, prelevassimo l'uscita (z, w) all'ingresso di S, non avremmo dipendenza dello stato di uscita dallo stato d'ingresso ad ogni ciclo di clock (lo stato di uscita varrebbe S + 1), ed il funzionamento non rispetterebbe le specifiche.

La struttura della rete di Mealy con componenti standard è quindi la seguente:



$\omega\,(x, S)$:

(z, w) = **if not** x **then** S **else** $S + 1$

$\sigma\,(x, S)$:

in_S = **when** x **do** $S + 1$

$T_\omega = T_{ALU} = 4t_p$

$T_\sigma = T_{ALU} = 4t_p$

In entrambi i casi il ciclo di clock è dato da:

$$\tau = max\,(\,T_\omega, T_\sigma\,) + \delta = 5t_p$$

Domanda 3

Si noti anzitutto che si tratta di una rete *sequenziale*, in quanto ad ogni ciclo di clock solo una delle uscite può variare e le altre devono *rimanere inalterate*. Ciò richiede l'esistenza di stato interno, corrispondente appunto al valore della configurazione delle uscite ad ogni ciclo di clock.

Dato il numero molto elevato di variabili d'ingresso, di uscita e dello stato interno è proponibile solo una realizzazione basata sull'uso di componenti logici standard.

Scegliendo il modello di *Moore* per l'automa, si usano 64 registri a 32 bit, $Z_0, \ldots, Z_{63}$, le cui uscite sono contemporaneamente variabili dello stato interno presente e variabili di uscita. Dette $out_0, \ldots, out_{63}$ le variabili di uscita, ognuna di 32 bit, la definizione della rete può essere data in un formalismo algoritmico nel seguente modo:

$(out_0\,, \ldots, out_{63}\,) = \omega\,(Z_0, \ldots, Z_{63})$:

$\forall\, i = 0\,..\,63 : out_i = Z_i$

$(in_{Z0}\,, \ldots, in_{Z63}\,) = \sigma\,(A, B, C, D, E, Z_0, \ldots, Z_{63})$:

$\forall\, i = 0\,..63 : in_{Zi}$ = [**when** clock **do] if** $(i = E)$ **then**

if not $A[B]$ **then** $C + D$ **else** $C - D$

dove la parte tra parentesi quadra è pleonastica, sapendo che la transizione di stato interno può avvenire solo in corrispondenza dell'impulso di clock. Fondamentale è invece la scrittura **if** $(i = E)$ stante a indicare che la variazione di stato interno avviene solo per quella parte di stato memorizzata nel registro Z_i con $i = E$.

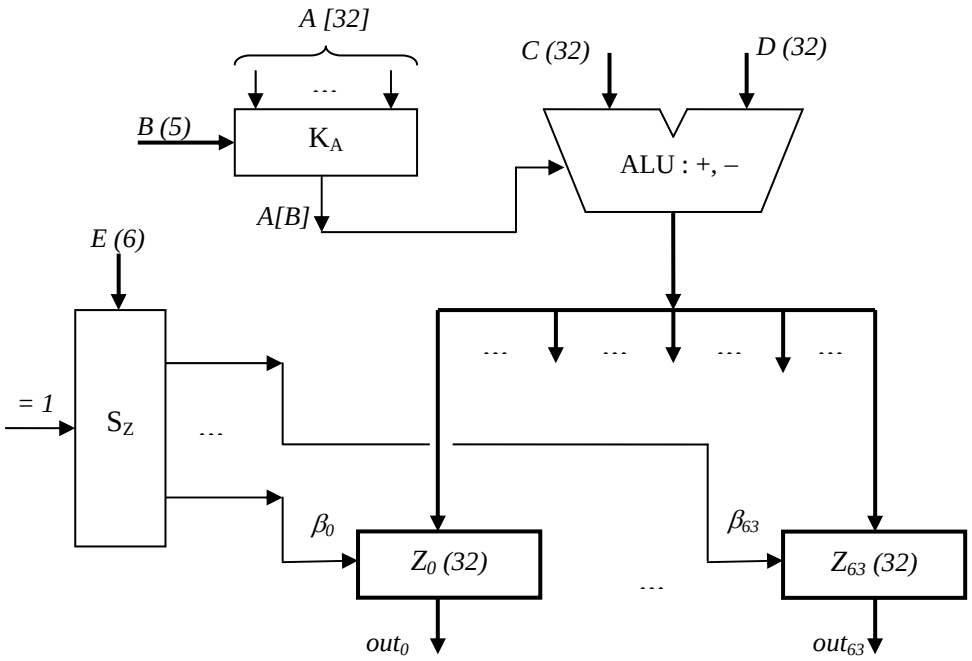
Formalmente, dalla descrizione della funzione di transizione dello stato interno si ricava quanto segue:

a) il bit B-esimo di *A* è ottenuto considerando *A* come un *array di 32 bit*, con indice dato dal valore di *B*; quindi il bit *A[B]* si ottiene come uscita di un commutatore K_A avente i 32 bit di *A* come ingressi primari ed i 5 bit di *B* come variabili di controllo;

b) disponendo di una ALU per eseguire *C* + *D* oppure *C* – *D*, la variabile di controllo di questa ALU è data proprio da *A[B];*

c) l'ingresso di tutti i registri Z_0, …, Z_{63} è la stessa uscita della ALU; *il segnale di abilitazione alla scrittura in tali registri è dato dall'uscita di un selezionatore*, avente come ingresso primario la costante cablata 1 e come variabili di controllo i 6 bit di *E*.

La struttura della rete sequenziale è quindi:



Il ritardo di stabilizzazione del commutatore K_A vale

$$T_{KA} = 5t_p$$

in quanto la sua espressione logica in forma SP consta di 32 termini AND, ognuno con 6 variabili (5 variabili di controllo ed un ingresso primario), e di un termine OR con 32 variabili; per implementare ognuno dei 32 termini AND occorrono due livelli di porte AND con al massimo 4 ingressi; per implementare il termine OR occorrono 3 livelli di porte OR con al massimo 4 ingressi.

Il selezionatore S_Z si stabilizza, in un tempo $2t_p$ (solo funzione AND, con struttura ad albero di profondità uguale a due), *in parallelo* alla struttura data da K_A e ALU operanti tra loro in serie all'interno di un ciclo di clock.

I ritardi di stabilizzazione delle funzioni caratterizzanti l'automa sono quindi:

$$T_\omega = 0$$

$$T_\sigma = max\,(\,T_{KA} + T_{ALU}\,,\,T_{SZ}\,) = T_{KA} + T_{ALU} = 9t_p$$

da cui il valore del ciclo di clock:

$$\tau = max\,(\,T_\omega\,,\,T_\sigma\,) + \delta = 10t_p$$