Esercitazione 1

Ogni esercitazione ha lo scopo di servire da guida per la preparazione su una specifica parte del corso. È fortemente consigliato che lo studente lavori indipendentemente all'esercitazione durante lo s volgimento di tale parte a lezione e prima che sia disponibile la soluzione, approfondendo criticamente i vari aspetti e accompagnando la soluzione con adeguate spiegazioni rivolte alla comprensione ed alla esposizione dei concetti del corso.

Soluzione: venerdì 10 ottobre

Per le seguenti reti logiche dare una realizzazione, con le caratteristiche indicate caso per caso, e valutare il tempo di stabilizzazione o il ciclo di clock in funzione del ritardo t_p di una porta logica con al massimo 4 ingressi.

Per ogni rete studiare almeno due realizzazioni che siano diverse per quanto riguarda il metodo seguito (ad esempio, per reti combinatorie partendo dalla tabella di verità o da una descrizione algoritmica, per reti sequenziali usando il modello matematico di Moore e quello di Mealy) e per quanto riguarda i componenti logici utilizzati (elementari oppure standard), ove i vari casi siano possibili.

a) Una rete logica ha quattro variabili booleane di ingresso a_0, a_1, b_0, b_1 e due variabili booleane di uscita z_0, z_1. La specifica del comportamento è la seguente:

   - se $a_0 \neq a_1$: $z_0 = max\ (b_0, b_1)$ e $z_1 = min\ (b_0, b_1)$, altrimenti z_0 e z_1 sono rispettivamente uguali alla somma ed al riporto dell'addizione di b_0, b_1.

b) Una rete logica ha quattro variabili booleane di ingresso a_0, a_1, b_0, b_1 e due variabili booleane di uscita z_0, z_1. La specifica del comportamento è la seguente:

   - se $a_0 \neq a_1$: $z_0 = max\ (b_0, b_1)$ e z_1 rimane inalterata, altrimenti z_0 rimane inalterata e $z_1 = 0$.

c) Una rete logica riceve in ingresso una sequenza di coppie (J, A), dove J è un valore naturale di 5 bit ed A è un valore intero di 32 bit in complemento a due, ed invia in uscita una sequenza di coppie (Z_0, Z_1), dove Z_0 è un valore booleano e Z_1 un valore intero di 32 bit in complemento a due.

   Sapendo che A assume il valore zero prima che inizi la sequenza di ingresso, la specifica del comportamento della rete è la seguente:

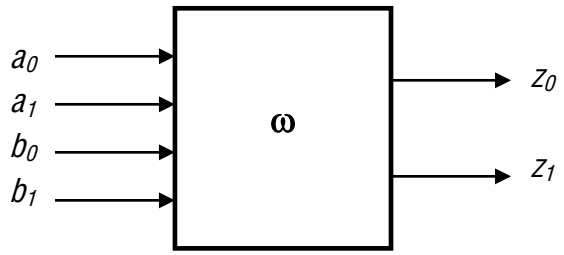
   per ogni coppia (J, A):

   - se $A[J] = 0$, Z_0 riconosce se A è una potenza di 2 e Z_1 è uguale al massimo tra il valore attuale di A ed il valore precedente di A,
   - se $A[J] = 1$, Z_0 riconosce se $1024 \leq A < 2048$ e Z_1 è uguale al minimo tra il valore attuale di A ed il valore precedente di A.

Soluzione

(a)

La rete combinatoria



implementa una trasformazione (ω) da stati d'ingresso (combinazioni di a_0, a_1, b_0, b_1) a stati di uscita (combinazioni di z_0, z_1), tale che ad ogni stato d'ingresso corrisponde uno ed un solo stato di uscita. Le specifiche di tale funzione possono essere espresse in un formalismo algoritmico:

$$z_0 = if\ (a_0 \neq\ a_1)\ then\ max\ (b_0,\ b_1\ )\ else\ sum\ (b_0,\ b_1\ )$$
$$z_1 = if\ (a_0 \neq\ a_1)\ then\ min\ (b_0,\ b_1\ )\ else\ carry\ (b_0,\ b_1\ )$$

Valgono le proprietà:

$$sum\ (b_0,\ b_1\ ) =\ b_0 \oplus b_1$$
$$carry\ (b_0,\ b_1\ ) =\ b_0\ b_1$$
$$(a_0 \neq\ a_1) =\ a_0 \oplus a_1$$
$$max(b_0,\ b_1\ ) =\ b_0 +\ b_1$$
$$min\ (b_0,\ b_1\ ) =\ b_0\ b_1$$

che verranno utilizzate nella soluzione la cui definizione sarà espressa direttamente in formalismo algoritmico. Il fatto che z_1 sia univocamente uguale a $b_0\ b_1$, indipendentemente dai valori di a_0 e a_1, non ha particolare significato, se non segnalare l'esistenza di ottimizzazioni.

Soluzione 1 – Definizione mediante tabella di verità

Dalle specifiche si ricava la tabella di verità:

a_0	a_1	b_0	b_1	z_0	z_1
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	1	1
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	1

dalla quale si ottengono le espressioni logiche in *forma normale SP* delle variabili di uscita (ricavate indipendentemente l'una dall'altra). In base alle proprietà dell'algebra di Boole possono essere introdotte varie ottimizzazioni, tra le quali si segnala la seguente:

$$z_0 = (\overline{b_0}\, b_1 \;+\; b_0\, \overline{b_1})(\overline{a_0}\;\overline{a_1} \;+\; \overline{a_0}\, a_1 + \; a_0\, \overline{a_1} \;+\; a_0\, a_1) \;+\; \overline{a_0}\, a_1 b_0\, b_1 \;+\; a_0\, \overline{a_1} b_0\, b_1$$
$$= \;\overline{b_0}\, b_1 \;+\; b_0\, \overline{b_1} + \; \overline{a_0}\, a_1 b_0\, b_1 \;+\; a_0\, \overline{a_1} b_0\, b_1$$

$$z_1 = b_0\, b_1(\overline{a_0}\;\overline{a_1} \;+\; \overline{a_0}\, a_1 + \; a_0\, \overline{a_1} \;+\; a_0\, a_1) = b_0\, b_1$$

Oltre ad ottenere il risultato atteso per z_1, l'espressione logica di z_1 contiene tutti termini AND e OR con al più quattro variabili. Quindi, la rete combinatoria corrispondente (non mostrata in questa sede) è a due livelli di logica. Il massimo ritardo di stabilizzazione è quindi:

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

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

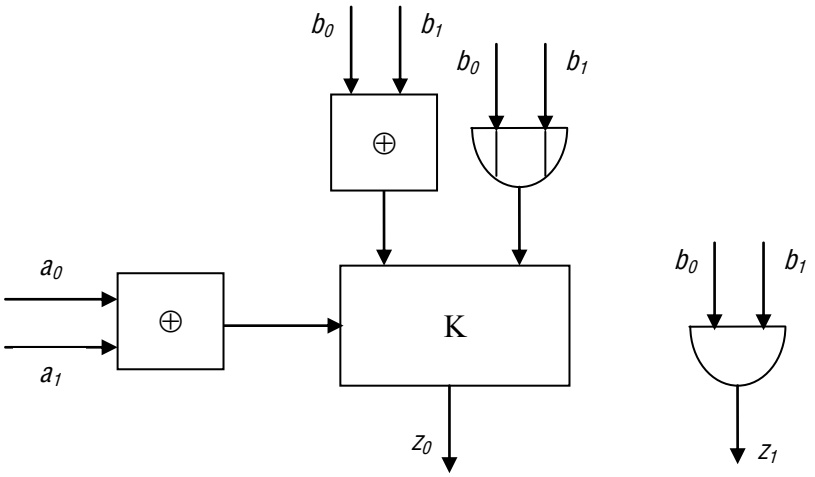
Si osservi come le minimizzazioni delle espressioni logiche siano importanti soprattutto agli effetti della possibile riduzione del ritardo di stabilizzazione.

Soluzione 2 – Definizione mediante formalismo algoritmico e realizzazione usando componenti logici standard

Essendo molto semplice la descrizione nel formalismo algoritmico:

$$z_0 = if\ (a_0 \oplus a_1)\ \ then\ (b_0 + \ b_1)\ else\ (b_0 \oplus b_1)$$
$$z_1 = b_0\, b_1$$

possiamo ricavare direttamente una prima implementazione della rete in termini di *componenti logici standard,* nel nostro caso: commutatori, confrontatori, porte AND e OR:



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 dal 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, ⊕, AND, OR, NOT) o comunque a reti già realizzate indipendentemente. 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.

Poiché i due confrontatori si stabilizzano in parallelo tra loro e con la porta OR, il ritardo di stabilizzazione della rete, che deve essere il *massimo* possibile, è dato da:

$$T_\omega = T_\oplus + T_K = 4t_p$$

Soluzione 3 – Definizione mediante formalismo algoritmico e realizzazione ricavando le espressioni logiche SP

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.

Si tratta di sostituire ad ogni funzione la sua espressione logica e sviluppare l'espressione ottenuta fino a ridurla ad una forma SP accettabile.

Nel nostro caso, notando che

$$b_0 + b_1 = b_0 \oplus b_1 + b_0\, b_1$$

e ricordando che

$$a_0 \oplus a_1 = \overline{a_0}\, a_1 + a_0\, \overline{a_1}$$
$$\overline{a_0 \oplus a_1} = \overline{a_0}\, \overline{a_1} + a_0\, a_1$$

si ottiene:

$$z_0 = (\overline{a_0}\, \overline{a_1} + a_0\, a_1)(\overline{b_0}\, b_1 + b_0\, \overline{b_1}) + (\overline{a_0}\, a_1 + a_0\, \overline{a_1})(\overline{b_0}\, b_1 + b_0\, \overline{b_1} + b_0\, b_1)$$

da cui si arriva immediatamente all'espressione logica della soluzione 1.

(b)

Il fatto che, in alcune situazioni, le variabili di uscita debbano rimanere *inalterate* comporta che siamo in presenza di un *automa*, quindi da realizzare come rete sequenziale.

Per dare la definizione dell'automa, una volta noti gli stati d'ingresso e di uscita, si tratta anzitutto di definire gli ***stati interni***. Nel nostro caso, il concetto di stato interno è legato al fatto di *ricordare il valore dello stato di uscita precedente*. È quindi naturale fare coincidere gli stati interni con gli stati di uscita. Da qui discende anche come convenga partire dalla definizione dell'automa secondo il modello matematico di *Moore*.

Dette y_0, y_1 le *variabili dello stato interno presente*, abbiamo subito la definizione della funzione delle uscite in un formalismo algoritmico:

$$\omega: \begin{cases} z_0 = y_0 \\ z_1 = y_1 \end{cases}$$

La funzione di transizione dello stato interno è definita in modo tale da dare luogo ai valori Y_0, Y_1 (*variabili dello stato interno successivo*) che le variabili y_0, y_1, e quindi z_0, z_1, assumeranno al prossimo ciclo di clock [1].

Usando un formalismo algoritmico, ciò può essere espresso secondo due metodi equivalenti.

[1] Il fatto che z_1 assuma sempre il valore 0 non è significativo. Eventualmente, lo studente può modificare l'esercizio scegliendo una funzione diversa.

Il primo è:

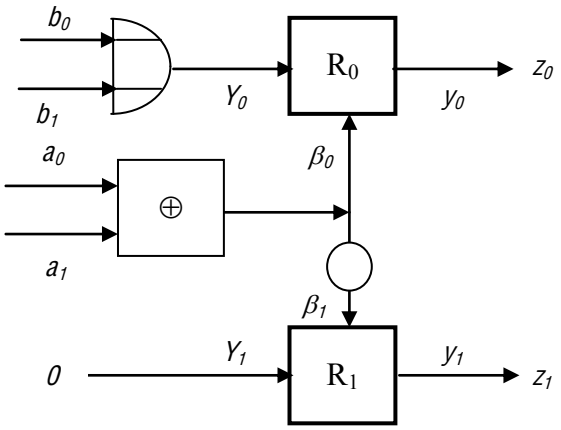
$$\sigma: \begin{cases} Y_0 = if\ (a_0 \oplus a_1)\ then\ (b_0 + b_1)\ else\ y_0 \\ Y_1 = if\ not\ (a_0 \oplus a_1)\ then\ 0\ else\ y_1 \end{cases}$$

nel quale il fatto di non modificare il valore di una variabile dello stato interno viene realizzato *riassegnando esplicitamente alla variabile dello stato successivo il valore della variabile dello stato presente*. Questo comporta che non venga fatto uso di segnali di abilitazione alla scrittura nei registri, in quanto in ogni registro è prevista la scrittura ad ogni ciclo di clock.

Il secondo metodo consiste nell'utilizzare i *segnali di abilitazione allo scrittura nei registri*:

$$\sigma: \begin{cases} Y_0 = when\ (a_0 \oplus a_1)\ do\ (b_0 + b_1) \\ Y_1 = when\ not\ (a_0 \oplus a_1)\ do\ 0 \end{cases}$$

Passando alla realizzazione con componenti logici standard, lo schema della rete sequenziale con il secondo metodo è:



Assumendo la durata dell'impulso di clock

$$\delta = t_p$$

il ciclo di clock è dato da :

$$\tau = max\ (T_\omega,\ T_\sigma) + \delta = max\ (0,\ 2t_p) + t_p = 3t_p$$

Si noti che, con il primo metodo, le variabili dello stato successivo hanno *in ingresso un commutatore* che, in generale, potrebbe far aumentare il ciclo di clock rispetto al secondo metodo. È il caso di questo esercizio, in quanto il commutatore si stabilizzerebbe in serie al confrontatore.

Definiamo ora l'automa secondo il modello matematico di *Mealy*. Avendo già a disposizione l'automa di Moore, conviene passare all'automa equivalente di Mealy mediante semplici trasformazioni di quello di Moore. In questo caso, poiché l'automa di Moore ha le variabili di uscita coincidenti con quelle dello stato interno presente (uscite dei registri), *uno dei possibili* automi equivalenti di Mealy ha le variabili di uscita *sostanzialmente* coincidenti con quelle dello stato interno successivo (ingressi dei registri), cioè la funzione delle uscite viene *sostanzialmente* a coincidere con quella di transizione dello stato interno ("Moore anticipato"). Il termine"sostanzialmente" sta a significare che:

- se si adotta il *primo metodo* per esprimere la funzione σ (senza segnali di abilitazione alla scrittura nei registri), allora le due funzioni coincidono esattamente:

$$\sigma: \begin{cases} Y_0 = if\ (a_0 \oplus a_1)\ then\ (b_0 + b_1)\ else\ y_0 \\ Y_1 = if\ not\ (a_0 \oplus a_1)\ then\ 0\ else\ y_1 \end{cases}$$

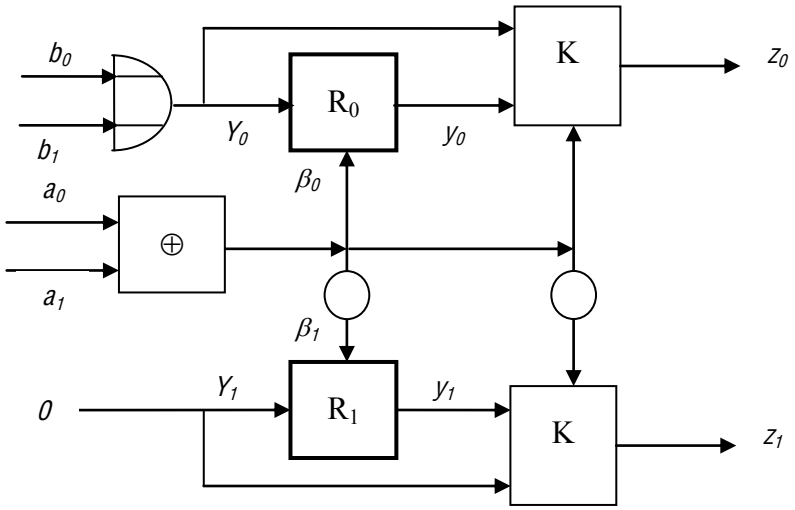
$$\omega: \begin{cases} z_0 = Y_0 \\ z_1 = Y_1 \end{cases}$$

- se si adotta il *secondo metodo* per esprimere la funzione σ (con segnali di abilitazione alla scrittura nei registri), allora la funzione ω deve essere espressa in modo da riprodurre la semantica della σ del primo metodo, pur non coincidendo con essa sintatticamente (non ha senso la clausola *when* per le varaibili di uscita):

$$\sigma: \begin{cases} Y_0 = when\ (a_0 \oplus a_1)\ \ do\ (b_0 + b_1) \\ Y_1 = when\ not\ (a_0 \oplus a_1)\ \ do\ \ 0 \end{cases}$$

$$\omega: \begin{cases} z_0 = if\ (a_0 \oplus a_1)\ \ then\ Y_0\ else\ y_0 \\ z_1 = if\ not\ (a_0 \oplus a_1)\ \ then\ Y_1\ else\ y_1 \end{cases}$$

Nel caso del secondo metodo, lo schema della rete sequenziale è:



Formalmente, il confrontatore dovrebbe comparire due volte, una per la funzione delle uscite ed una per quella di transizione dello stato interno. Ovviamente, rilevando che due parti di una rete sono identiche, è sempre consentito metterne una sola a comune.

Il ciclo di clock è dato da:

$$\tau = max\ (T_\omega,\ T_\sigma) + \delta = max\ (4t_p,\ 2t_p) + t_p = 5t_p$$

Si verifichi con un esempio che i due automi sono effettivamente equivalenti, cioè, per una stessa sequenza d'ingresso ed essendo inizializzati nello stesso stato interno, l'automa di Moore dà luogo ad una sequenza di uscita identica ma ritardata di un ciclo di clock rispetto alla sequenza di uscita di Mealy.

(c)

Per motivi analoghi a quelli della domanda *b)*, siamo in presenza di un automa. Dato l'elevato numero di stati, a maggior ragione occorre passare da una definizione mediante formalismo algoritmico.

L'individuazione di "cosa è stato" è semplice: occorre ricordare il valore dello stato d'ingresso ad ogni ciclo di clock affinché esso venga utilizzato (calcolo del massimo e minimo) nel prossimo ciclo di clock.

Utilizziamo una variabile S, intera a 32 bit, come insieme delle variabili dello stato interno. S, che verrà implementata mediante un registro a 32 bit, è inizializzata a zero.

Ne consegue subito la definizione della funzione di transizione dello stato interno:

$$\sigma:\ \ in_S = A$$

ad ogni ciclo di clock.

In questo esempio, dalla specifica risulta più diretto partire dal modello matematico di *Mealy*. La funzione delle uscite è data da

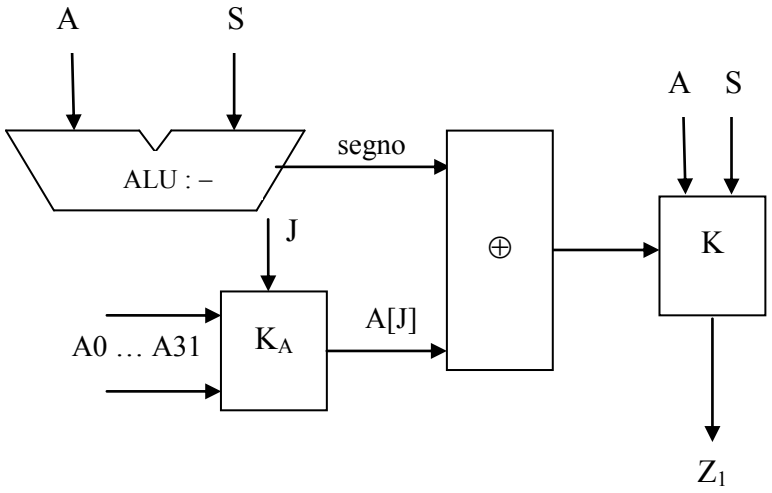
$$\omega: \begin{cases} Z_0 = \ if\ not\ A[J]\ \ then\ \ potenza\ (A)\ else\ \ compreso\ (A) \\ Z_1 = \ if\ not\ A[J]\ \ then\ \ max\ (A,S)\ else\ \ min\ (A,S) \end{cases}$$

dove le funzioni *potenza* e *compreso* devono a loro volta essere realizzate con un procedimento tipico delle reti combinatorie, *max* e *min* sono realizzabili mediante componenti logici standard, e *A[J]* è implementato considerando A di tipo array di bit indiciato mediante il valore di J.

Vediamo Z_1 in termini delle funzioni massimo e minimo. La definizione di tali funzioni è basata sul valore della variabile segno della sottrazione di A e S. La definizione di Z_1 può allora essere:

$$Z_1 = \ case\ A[J], segno\ (A-S) of$$
$$00 : A$$
$$01 : S$$
$$10 : S$$
$$11 : A$$

La sua realizzazione può essere data mediante un commutatore a due ingressi (A, S) con variabili di controllo in numero ridondante, oppure utilizzando la funzione del confronto per comandare un commutatore con il minimo numero di variabili di controllo. La seconda soluzione è la seguente:



Assumiamo il ritardo di stabilizzazione di una ALU uguale a $5t_p$. Usando porte logiche con al massimo 4 ingressi, il commutatore K_A ha ritardo di stabilizzazione anch'esso uguale a $5t_p$ (parte AND a 2 livelli di logica e parte OR a 3 livelli di logica). Il ritardo di stabilizzazione di questa rete vale quindi:

$$T_{Z_1} = 9\ t_p$$

Per quanto riguarda Z_0, la funzione *potenza* è definita dalla condizione che un solo bit della parola sia uguale ad uno e tutti gli altri a zero. La definizione può quindi essere data direttamente mediante l'espressione logica in forma canonica SP di 31 termini:

$$potenza\ (A) = \ A_0\,\overline{A_1}\ \ldots\ \overline{A_{31}} + \ \overline{A_0}\,A_1\ \ldots\ \overline{A_{31}} + \ \ldots + \ \overline{A_0\,A_1}\ \ldots\ A_{31}$$

La sua realizzazione è una rete combinatoria a 6 livelli di logica, quindi con ritardo di stabilizzazione $6t_p$.

La funzione *compreso* ($1024 \leq A < 2048$) è vera se il bit di posizione 10 (la posizione 0 corrisponde al bit meno significativo) è uguale ad uno e tutti quelli dalla posizione 11 alla 31 sono uguali a zero, quindi:

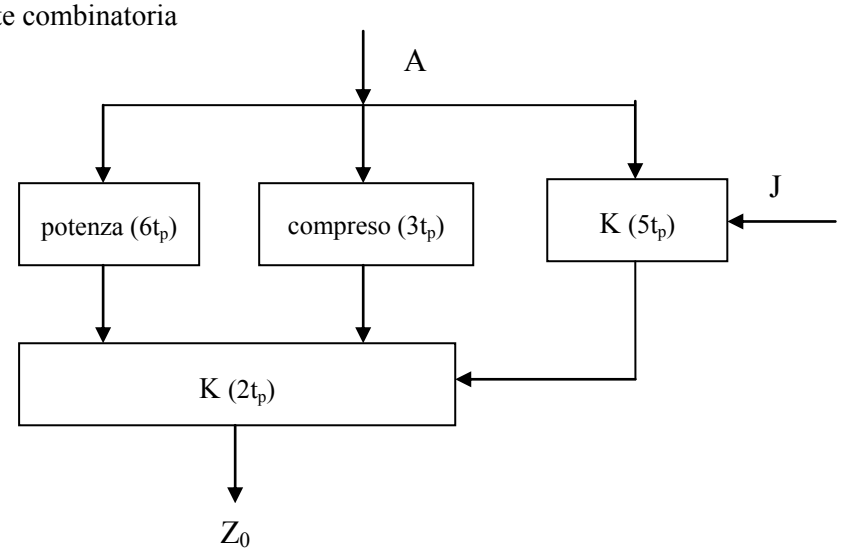
$$compreso\ (A) = \ (A[10] = 1)\ and\ (or(A[11 \dots 31]) = 0) =$$
$$= A_{10}\ (\overline{A_{11} + \ \dots + \ A_{31}}) = \ A_{10}\ \overline{A_{11}} \dots \overline{A_{31}}$$

realizzata con una rete combinatoria a 3 livelli di logica AND, quindi con ritardo di stabilizzazione $3t_p$.

Una volte note le due reti *potenza* e *compreso*, la realizzazione di

$$Z_0 \ = \ if\ not\ A[J]\ \ then\ \ potenza\ (A)\ else\ \ compreso\ (A)$$

è la seguente rete combinatoria



avente ritardo di stabilizzazione:

$$T_{Z_0} \ = \ 8\ t_p$$

In conclusione:

$$T_\omega \ = max\ (T_{Z_0}, T_{Z_1}) \ = \ 9\ t_p$$

Il ciclo di clock vale:

$$\tau = max\ (T_\omega, T_\sigma) + \delta = max\ (9t_p, 0) + t_p = 10t_p$$

Per definire l'automa di *Moore* equivalente, consideriamo che, in una delle possibili realizzazioni, la funzione di uscita dell'automa di Mealy diventa *parte* della funzione di transizione dello stato interno di Moore. Conservando ancora la variabile dello stato interno S (32 bit), introduciamo le ulteriori variabili dello stato interno S_0 (1 bit) e S_1 (32 bit), ottenendo:

$$\sigma: \begin{cases} in_S = A \\ in_{S_0} = if\ not\ A[J]\ \ then\ \ potenza\ (A)\ else\ \ compreso\ (A) \\ in_{S_1} = \ \ if\ not\ A[J]\ \ then\ \ max\ (A,S)\ else\ \ min\ (A,S) \end{cases}$$

La funzione delle uscite è quindi:

$$\omega: \begin{cases} Z_0 = \ S_0 \\ Z_1 = \ S_1 \end{cases}$$

In altre parole, abbiamo ricavato formalmente la soluzione per cui una possibile rete sequenziale di Moore equivalente è ottenuta dalla rete sequenziale di Mealy inserendo registri su tutte le variabili di uscita ("Mealy ritardato").

Il ciclo di clock della rete sequenziale di Moore è, in questo caso, uguale a quello della rete sequenziale di Mealy.