II.

Reti Logiche

Dispensa estratta dal testo

M. Vanneschi. Architettura degli elaboratori. Didattica e Ricerca. Manuali. Pisa University Press, 2013.

http://www.pisauniversitypress.it/scheda-libro/marco-vanneschi/architettura-degli-elaboratori-9788867411573-132417.html

La dispensa è materiale didattico integrativo e non sostituisce il libro di testo.

Reti combinatorie

Le Reti Combinatorie rappresentano l'implementazione di funzioni ("pure") a Livello Hardware. Ove si considerino funzioni "con stato", o automi a stati finiti, la loro implementazione a Livello Hardware darà luogo invece alle così dette Reti Sequenziali. Complessivamente, Reti Combinatorie e Reti Sequenziali definiscono la famiglia delle Reti Logiche, che, agli effetti del corso, permettono di realizzare il Livello Hardware di un sistema di elaborazione.

1.1 Definizione di reti combinatorie, loro formalizzazione e comportamento

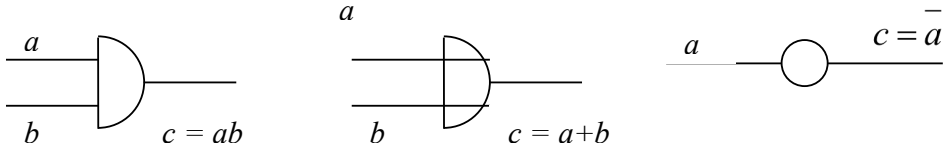
Una rete combinatoria è una rete logica che ha n ingressi binari $x_1, \ldots, x_n$ ed m uscite binarie $z_1, \ldots, z_m$. A ciascuna combinazione dei valori degli ingressi corrisponde una ed una sola combinazione dei valori delle uscite.

$x_1, \ldots, x_n$ e $z_1, \ldots, z_m$ sono dette *variabili logiche,* di ingresso e di uscita rispettivamente. Tutte le combinazioni possibili delle variabili logiche sono dette *stati,* di ingresso (in numero di 2^n) e di uscita (in numero di 2^m) rispettivamente.

Per descrivere le proprietà e la struttura interna delle reti combinatorie si usa un'algebra isomorfa all'algebra della logica, chiamata **algebra della commutazione**: si rimanda alla **dispensa di S. Antonelli "Rappresentazione dell'Informazione e Algebre Booleane"**. In particolare, in tale dispensa sono definiti

- Le proprietà dell'*algebra di Boole*, della quale l'algebra della commutazione rappresenta un caso particolare;
- gli *operatori logici* fondamentali: AND, OR, NOT;
- *la tabella di verità* per descrivere il comportamento ingresso-uscita di una rete combinatoria;
- le *espressioni logiche* in *forma normale SP* (somma di prodotti) ed in *forma canonica.*

Una volta ricavata l'espressione logica dalla tabella di verità, è immediato realizzare lo *schema logico* utilizzando i *componenti hardware elementari*, detti anche *porte logiche*, AND, OR, NOT. I simboli convenzionali di tali porte sono (è mostrato il caso AND, OR a due ingressi):

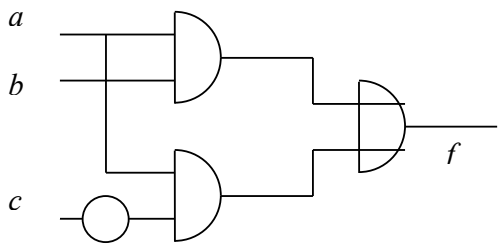


Porta AND **Porta OR** **Porta NOT**

Partendo da una espressione logica in forma SP, la rete combinatoria ottenuta è "*a due livelli di logica*", con questo intendendo che esiste un primo livello di porte AND operanti in parallelo, seguite da una porta OR finale. Ad esempio, la rete logica corrispondente all'espressione logica

$$f = ab + a\bar{c}$$

è data da:



1.2 Specifica di reti combinatorie

La specifica di una funzione logica, da implementare mediante una rete combinatoria, può essere data semplicemente "a parole". Molto utile sarà abituarsi a definire una funzione, e quindi a dare *la specifica del comportamento di una rete combinatoria*, mediante *un formalismo di tipo linguaggio di programmazione*. In effetti, è agevole rendersi conto come la specifica in tale formalismo e l'espressione logica, derivante dal procedimento di sintesi, rappresentino niente più che due diverse modalità sintattiche per dare la stessa informazione. *In più, la specifica data con un formalismo di programmazione avrà il vantaggio, essendo più sintetica, di permettere di guidare la progettazione di molte reti come composizione di reti già note.*

Le seguenti sono le specifiche di alcune reti combinatorie che, nel corso, supporremo *standard, o primitive*, cioè componenti che è possibili usare (al pari delle porte AND, OR, NOT) come blocchi basici nella progettazione di strutture più complesse:[1]:

- *confrontatore* ($\oplus$) a due ingressi *x, y* ed una uscita *z* (*z* è vero se *x, y* sono diversi):

  $z = x \oplus y =$ **not** *(x = y)*

[1] Per il momento supporremo che ogni ingresso o uscita sia corrispondente ad una sola variabile logica booleana, cioè che corrisponda ad una informazione di 1 bit.

- *commutatore* (K) a due ingressi primari *x, y*, un ingresso secondario o di controllo α, ed una uscita *z* (*z* assume il valore di *x* o di *y* a seconda che α sia falso o vero rispettivamente):

  z = **if not** α **then** x **else** y

- *selezionatore*, o selettore (S), a un ingresso primario *x*, un ingresso secondario o di controllo α, e due uscite z_1, z_2 (se α è falso z_1 assume il valore di *x* e z_2 vale falso (0), se α è vero z_1 vale falso (0) e z_2 assume il valore di *x*):

  z_1 = **if not** α **then** x **else** 0
  z_2 = **if not** α **then** 0 **else** y

oppure

if not α **then** (z_1 = x, z_2 = 0) **else** (z_1 = 0, z_2 = x)

- *operatori aritmetico-logici* [2]: addizione, sottrazione, traslazione (shift), rotazione, incremento decremento, ecc, la cui specifica è diretta. In questo corso interessa particolarmente considerare standard anche reti aritmetico-logiche *multifunzione*, o *ALU* (Arithmetic Logic Unit), capaci di eseguire sulle variabili di ingresso una delle operazioni suddette a seconda del valore assunto da un numero opportuno ($\lceil lg_2\ k \rceil$, se *k* sono le operazioni previste) di variabili di controllo.

Ritorniamo sulla specifica del *commutatore*, e generalizziamola al caso di *m* ingressi, $x_0, x_1, \ldots, x_{m-1}$., con $v = \lceil lg_2\ m \rceil$ variabili di controllo $\alpha_0, \alpha_1, \ldots, \alpha_{v-1}$.

La specifica è la seguente:

z = **case** $\alpha_0, \alpha_1, \ldots, \alpha_{v-1}$ **of**
0 0 … 0 : x_0
0 0 … 1 : x_1
…
1 1 … 1 : x_{m-1}
endcase

Questa scrittura è piuttosto significativa, in quanto esprime il concetto di *indirizzamento* o *indiciamento*: dati *m* oggetti ordinati, $x_0, x_1, \ldots, x_{m-1}$, essi sono caratterizzati da altrettanti identificatori unici, detti *indirizzi*, aventi valori decimali uguali agli *indici* loro assegnati (*0, 1,*

[2] Si veda ancora la dispensa di S. Antonelli per la parte sull'aritmetica binaria..

... , $m-1$); ogni indirizzo è espresso, in binario, mediante $\lceil lg_2\ m \rceil$ bit. Per conoscere il valore di un oggetto basta specificare il suo indirizzo o indice. Dunque, ogni volta che avremo bisogno di implementare la funzione di indirizzamento o indiciamento per conoscere una informazione, sarà sufficiente utilizzare un commutatore, senza bisogno di realizzare ex-novo la funzione stessa.

In modo analogo si specifica e si interpreta un *selezionatore a m uscite*.

1.3 Procedimento di sintesi

Una volta data la specifica di una rete (di una funzione logica), *nel caso più generale* il procedimento di sintesi della rete stessa è il seguente:

1. traduzione della specifica nella tabella di verità, per enumerazione di tutti i casi (2^n, per n variabili di ingresso);
2. per ogni variabile di uscita, scrittura della espressione logica in forma canonica SP (i termini AND corrispondono agli stati d'ingresso per i quali la variabile di uscita assume valore vero);
3. eventuale riduzione delle espressioni logiche;
4. traduzione di ogni espressione logica in uno schema di rete a due livelli di logica.

Questo procedimento ha complessità $\mathbf{O}(2^n)$, ed è quindi praticamente applicabile solo nei casi in cui il numero degli ingressi n assuma un valore "contenuto". Ad esempio, è impraticabile applicarlo quando gli ingressi della rete siano costituiti da una o più parole.

A maggior ragione, il passo 3 (riduzione o minimizzazione) è, in generale, di complessità esponenziale: si conoscono solo delle buone euristiche in casi limitati. Chi sia interessato, troverà una trattazione completa in [GER]. Per i nostri scopi, è sufficiente l'applicazione delle proprietà dell'algebra di Boole per la riduzione delle espressioni logiche nella sintesi di *reti standard*, in quanto ogni altra rete verrà ricavata come composizione di tali reti, oppure (come vedremo nella parte del corso dedicata al firmware) utilizzando anche *memorie* viste esse stesse come componenti standard.

Esempio: sintesi di un commutatore a 2 ingressi

Partendo dalla specifica data in precedenza, si ricava la tabella di verità:

α	x	y	z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

L'espressione della variabile di uscita z in forma canonica SP è:

$$z = \bar{\alpha} x \bar{y} + \bar{\alpha} x y + \alpha \bar{x} y + \alpha x y$$

Applicando le proprietà distributiva, della complementazione e dell'intersezione [Antonelli] si ottiene:

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

Si noti come, al di là delle diversa sintassi, si sia ottenuto esattamente quanto espresso con la specifica data con un formalismo di programmazione.

A questo punto è immediato disegnare lo schema della rete a due livelli di logica.

Esempio: sintesi di un selezionatore a due uscite

Si ottiene subito:

$$z_1 = \bar{\alpha} x$$

$$z_2 = \alpha y$$

e quindi una rete ad un solo livello di logica.

Esempio: sintesi di un confrontatore

Anche in questo caso, senza bisogno di minimizzazioni, si ottiene:

$$z = x \bar{y} + \bar{x} y$$

cui corrisponde una rete a due livelli di logica.

Esempio: con riferimento ad un confrontatore, verificare che $\overline{\overline{z}} = z$

L'espressione di $\bar{z}$ si ricava osservando, nella tabella di verità, tutti gli stati di ingresso cui corrisponde uscita falsa:

$$\bar{z} = \bar{x}\bar{y} + xy$$

da cui:

$$\overline{\overline{z}} = \overline{(\bar{x}\bar{y}) + (xy)}$$

Applicando due volte il Teorema di De Morgan si ottiene:

$$\overline{\overline{z}} = \overline{(\bar{x}\bar{y})}\,\overline{(xy)} = (\overline{\bar{x}} + \overline{\bar{y}})(\bar{x} + \bar{y}) = (x + y)(\bar{x} + \bar{y})$$

Applicando le proprietà distributiva, della complementazione e dell'unione, si ottiene:

$$\overline{\overline{z}} = x\bar{x} + \bar{x}y + x\bar{y} + y\bar{y} = \bar{x}y + x\bar{y} = z$$

1.4 Comportamento temporale delle reti combinatorie

I *segnali* corrispondenti ai valori assunti dalle variabili logiche sono tutti del tipo *a livelli*, cioè subiscono transizioni in istanti tra loro scorrelati [3].

Per valutare le prestazioni di un sistema, occorre cominciare a saper valutare le prestazioni di reti combinatorie. Ogni rete reale è caratterizzata da un *ritardo* t_r, necessario affinchè, in seguito ad una variazione dello stato d'ingresso, si produca la corrispondente variazione dello stato di uscita. Solo dopo questo tempo si dice che la rete si è *stabilizzata*: durante l'intervallo di durata t_r, detto appunto anche *tempo di stabilizzazione*, il comportamento della rete, in ogni suo punto, è impredicibile ed, in particolare, il valore delle uscite non è significativo.

Se lo stato di ingresso varia *durante* l'intervallo t_r, a partire dall'istante in cui si è verificata tale variazione occorre attendere un tempo t_r per avere la stabilizzazione.

Nel seguito del corso supporremo che

a) le variazioni dello stato d'ingresso siano *sincronizzate*, cioè che avvengano ad istanti discreti, ed equidistanti, di una sequenza temporale. Ciò significa che le eventuali variazioni delle variabili d'ingresso avvengono *contemporaneamente* in tali istanti;

b) *gli istanti di tale sequenza temporale siano distanziati di un periodo* $\geq t_r$.

[3] Al contrario, in un segnale *impulsivo* ogni transizione da 0 a 1 è sempre seguita da una transizione da 1 a 0 a distanza di un intervallo di tempo costante. Segnali impulsivi verranno usati solo per sincronizzare registri usati nelle reti sequenziali.

Per una porta logica, indichiamo con t_p il tempo di stabilizzazione. Allo stato attuale della tecnologia esso assume valori dell'ordine di *1 nsec* fino ascendere a circa *0,1 nsec*. Nel seguito supporremo che le porte NOT abbiano ritardo nullo, o meglio inglobato nel ritardo delle porte AND/OR connesse alle usci delle porte NOT.

E' importante ricordare che t_p è il **massimo** valore che può assumere il tempo di stabilizzazione di una porta, e che non è mai possibile ipotizzare di conoscere il valore preciso del ritardo entro tale massimo.

Dalle assunzioni a), b) deriva che il massimo tempo di stabilizzazione di una rete a *L livelli di logica* è dato da

$$t_r = L\ t_p$$

Il valore di t_p dipende dal numero di ingressi n della porta. La funzione $t_p(n)$ ha un andamento monotono crescente; la crescita è relativamente lenta per $n \leq n_c$, tale da poter considerare t_p costante, mentre è molto più rapida per $n > n_c$. Valori tipici di n_c allo stato attuale variano da 4 a 8. Il valore di t_p che si associa ad una porta logica è quello per $n \leq n_c$.

La conseguenza di questa caratteristica è che, ove l'espressione logica di una variabile di uscita contenga un termine AND con più di n_c variabili d'ingresso, la porta AND della rete deve essere *decomposta in più porte in cascata*, ognuna con al più n_c ingressi. Lo stesso dicasi per una porta OR finale.

Ad esempio, il tempo di stabilizzazione di un commutatore è $2t_p$ solo se il numero di variabili di controllo è $< n_c$.

Questa ora esposta è una delle ragioni per le quali può essere necessario realizzare reti mediante **schemi a più di due livelli di logica**. L'altra ragione principale la vedremo nella sez. successiva.

1.5 Reti combinatorie operanti su parole

Nell'applicazione pratica che faremo delle reti combinatorie sarà molto frequente il caso in cui le variabili di ingresso e di uscita sono da considerare raggruppate in parole, cioè la funzione che definisce la rete è applicata a parole di N bit (ad esempio, 32) invece che a semplici variabili booleane. Ne sono esempi:

- *commutatore* a m ingressi ognuno dei quali a N bit. L'uscita è quindi su N bit, mentre le variabili di controllo rimangono in numero di $\lceil lg_2\ m \rceil$;

- analogamente per un *selezionatore* su parola di *N* bit;
- *confrontatore* a *N* bit: il confronto è eseguito su ogni coppia di bit corrispondenti dei due ingressi a *N* bit, generando un bit dell'uscita a *N* bit;
- *addizionatore* a *N* bit: vengono sommati due numeri interi su *N* bit, ottenendo il valore della somma su *N* bit ed un bit di riporto finale.

In tutti questi casi, la complessità del procedimento di sintesi delle reti combinatorie impedisce di ricorre al metodo generale basato sulla tabella di verità. Si cerca invece di *comporre reti ad 1 bit, del tipo corrispondente, per ottenere la rete ad N bit*; ad esempio, realizzare un commutatore ad *N* bit utilizzando commutatori ad 1 bit.

La complessità del procedimento di sintesi è ora $\mathbf{O}(1)$, ovviamente ottimale *a condizione che il risultato sia accettabile in termini di prestazioni*, cioè che il tempo di stabilizzazione della rete ad *N* bit sia confrontabile con quello della rete corrispondente a 1 bit.

- Per alcune reti la composizione è *parallela*: la rete ad *N* bit è composta da *N* reti ad 1 bit tutte *indipendenti*, cioè nessuna utilizza in ingresso le uscite di altre. Il ritardo della rete ad *N* bit è dunque uguale a quello della rete ad 1 bit. E' il caso del *commutatore, selezionatore, confrontatore.* Nel seguito del corso queste reti saranno assunte come *standard.*
- Per altre reti, la composizione è *in cascata*: la rete ad *N* bit è composta da *N* reti ad 1 bit tali che le uscite dell'i-esima costituiscono ingressi della (i+1)-esima. Il ritardo della rete ad *N* bit è ora *N* volte quello della rete ad 1 bit. E' il caso dell'*addizionatore*, a causa della propagazione del riporto.

  È possibile ottimizzare il ritardo dell'addozionatore usando la così detta tecnica del *salto del riporto*, mediante la quale è possibile ridurre da *N* ad un valore relativamente basso e costante il fattore per cui va moltiplicato il tempo di stabilizzazione dell'addizionatore ad 1 bit per ottenere il ritardo dell'addizionatore ad *N* bit. In pratica, il tempo di stabilizzazione di un tale addizionatore è dell'ordine di $h\ t_p$, con *h* variabile tra 4 e 8. Nel seguito del corso assumeremo di disporre di addizionatori con salto del riporto come reti *standard*.

La sintesi di *ALU* a *N* bit, esse stesse considerate *standard* nel seguito del corso, si effettua applicando tutte le conoscenze finora acquisite e la teoria sull'aritmetica binaria.

1.6 Esercizi

1. Dire se le seguenti affermazioni sono vere o false, oppure vere sotto certe condizioni da specificare, spiegando chiaramente la risposta:

a) il confrontatore è una rete combinatoria avente due stati d'ingresso ed uno stato di uscita;

b) è necessario che un commutatore ad m ingressi abbia esattamente $\lceil lg_2\ m \rceil$ variabili di controllo;

c) un commutatore ad m ingressi può avere m variabili di controllo;

d) una rete combinatoria con 16 ingressi e 20 uscite non può essere realizzata a due livelli di logica.

2. Progettare un commutatore a 4 ingressi: *a)* utilizzando solo porte logiche AND, OR, NOT, *b)* utilizzando anche decodificatori. Confrontare gli schemi *a)* e *b)* dal punto di vista delle prestazioni e del costo.

3. Progettare un addizionatore ad 1 bit (a tre ingressi e due uscite). Ricavarne lo schema logico in due versioni: *a)* utilizzando solo porte AND, OR, NOT, *b)* utilizzando anche confrontatori e/o altre reti standard. Confrontare gli schemi *a)* e *b)* dal punto di vista delle prestazioni e del costo.

4. Dimostrare che l'operazione di traslazione (shift) destra di una posizione di un numero naturale di N bit equivale alla divisione intera per 2 del numero stesso. Analogamente, dimostrare che l'operazione di traslazione sinistra di una posizione di un numero naturale di N bit equivale alla moltiplicazione per 2 del numero stesso.

5. Progettare una ALU che esegua somma, sottrazione, traslazione destra e traslazione sinistra (le traslazioni applicate all'ingresso sinistro della ALU). La ALU dispone dei Flag (risultati di predicati): **S**egno del risultato dell'operazione eseguita, risultato uguale a **Z**ero, **C**arry (i riporto della cifra più significativa), **O**verflow, e **S**hifted (bit espulso nelle traslazioni).

6. Sintetizzare la rete definita dalla seguente tabella di verità, dove il simbolo "–" sta per "non specificato" (può valere sia 0 che 1):

x_1	x_2	x_3	x_4	z_1	z_z
0	0	–	–	1	0

0	1	–	–	0	0
1	–	0	–	0	1
1	–	1	0	1	1
1	–	1	1	1	0

2. Reti sequenziali

In questo corso le reti sequenziali vengono utilizzate principalmente per studiare la realizzazione della Parte Controllo e della Parte Operativa di unità di elaborazione; pertanto, la trattazione sarà orientata a questo scopo e tralascerà molti altri aspetti della teoria delle reti sequenziali. Per chi sia interessato ad approfondire qualsiasi tema sull'argomento, il testo [GER] è quanto di più completo si possa trovare.

2.1 Definizione di reti sequenziali, loro formalizzazione e comportamento

Un *automa a stati finiti* è una macchina caratterizzata da

- *n variabili logiche di ingresso, e corrispondentemente h = 2n stati d'ingresso X1, …, Xh;*
- *m variabili logiche di uscita, e corrispondentemente* $k = 2^m$ *stati d'uscita* $Z_1, \ldots, Zk;$
- *r variabili logiche dello stato interno, e corrispondentemente* $p = 2^r$ *stati interni* $S_1, \ldots, Sp;$
- una *funzione di transizione dello stato interno:*

$$\sigma: \quad X \times S \to S$$

tale funzione definisce la trasformazione dello stato interno dal valore *presente* al valore *successivo* in corrispondenza del valore dello stato d'ingresso;

- una *funzione delle uscite:*

$$\omega: \quad X \times S \to Z$$

tale funzione definisce la trasformazione dello stato di uscita in corrispondenza del valore dello stato d'ingresso e dello stato interno *presente*.

Una rete logica sequenziale implementa un automa a stati finiti.

In un *modello strutturale ideale* di rete sequenziale di tipo *sincrono*, le variazioni degli stati avvengono in corrispondenza degli istanti di una sequenza temporale discreta $t_o, t_1, \ldots, t_n, \ldots$di intervallo (periodo) costante $\Delta = t_{i+1} - t_i$.

Si possono definire due distinti *modelli matematici di automa,* e quindi di rete sequenziale: il modello di *Mealy* e il modello di *Moore.*

In entrambi i modelli, considerando il comportamento dell'automa al tempo *t*, lo stato interno successivo *S(t+1)* [4] dipende tanto dallo stato d'ingresso al tempo *t*, *X(t),* quanto dallo stato interno presente, *S(t):*

[4] Questo simbolo indica convenzionalmente lo stato interno *successivo,* che diverrà stato presente al tempo $t+\Delta$.

$$S(t+1) = \sigma\ (X(t), S(t))$$

Nel modello di Mealy, lo stato di uscita al tempo *t*, *Z(t),* dipende tanto dallo stato d'ingresso al tempo *t*, *X(t),* quanto dallo stato interno presente, *S(t):*

$$Z(t) = \omega\ (X(t), S(t))$$

Nel modello di Moore, *Z(t)* dipende solo da *S(t):*

$$Z(t) = \omega\ (S(t))$$

Nel modello di Moore la dipendenza tra stati di uscita e stati di ingresso è quindi espressa da:

$$Z(t) = \omega\ (S(t)) = \omega\ (\sigma\ (X(t-1), S(t-1))) = \omega_1\ (X(t-1), S(t-1))$$

Ne discende che, *dati due automi di Mealy e di Moore equivalenti, con stati interni iniziali equivalenti e per la stessa sequenza d'ingresso si ha che la sequenza di uscita dell'automa di Moore è ritardata, rispetto a quella dell'automa di Mealy, di un intervallo* Δ *della sequenza temporale.*

2.2 Un esempio

Vediamo un esempio, tipico della problematica degli automi: un *riconoscitore di sequenza*.

La *specifica* di un automa viene sempre data un termini di trasformazione da stati di ingresso a stati di uscita e di caratteristiche dell'automa nei confronti delle *sequenze* di ingresso e di uscita; nel dare questa specifica è implicito il concetto di *memoria* dell'automa, o stato interno, anche se, da essa, non è sempre immediatamente evidente il numero ed il significato degli stati interni: questi devono essere ricavati durante la sintesi dell'automa.

Consideriamo un automa avente una variabile di ingresso *x* ed una variabile di uscita *z*. L'uscita assume valore 1 se e solo se nella sequenza d'ingresso si presentano due "1" consecutivi senza sovrapposizioni (ad esempio: se si verifica la sottosequenza di ingresso 01110 la corrispondente sottosequenza di uscita è 00100; se in ingresso si presenta 011110 in uscita si ha 001010).

Scegliamo inizialmente il modello matematico di *Mealy*.

Per realizzare l'automa, cominciamo a determinare gli *stati interni* [5]. Abbiamo bisogno di *ricordare* le seguenti situazioni: *a)* nella sequenza di ingresso si è presentato uno 0, oppure è lo stato iniziale; *b)* nella sequenza di ingresso si è presentato un 1.

Indichiamo con S0, S1 gli stati interni corrispondenti alle situazioni *a), b)* rispettivamente. L'evoluzione dell'automa può essere rappresentata dal *grafo di stato* di Fig. 1, secondo le convenzioni descritte in [GER, 2.1.3]: i nodi corrispondono a stati interni; gli archi corrispondono a transizioni tra stati interni, ed a questo scopo sono marcati con lo stato d'ingresso che provoca la transizione e con lo stato di uscita corrispondente (X / Z).

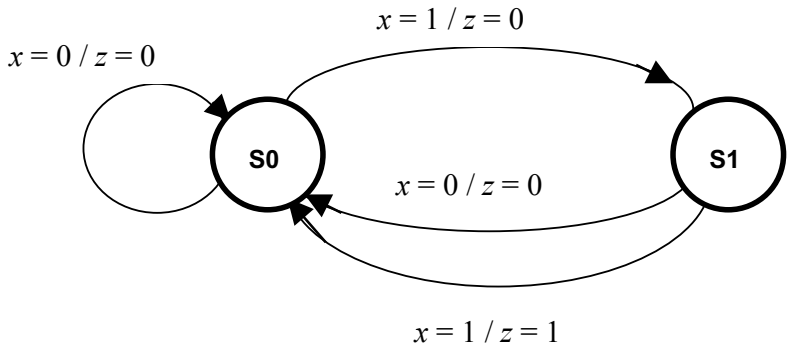


Figura 1

Per realizzare l'automa mediante una rete sequenziale, diamo una *codifica degli stati interni* mediante il minimo numero di variabili ($\lceil \lg_2 2 \rceil = 1$) Indichiamo con *y* la *variabile dello stato interno presente* e con *Y la variabile dello stato interno successivo*; una possibile codifica è:

stato int	***y***
S0	0
S1	1

Dobbiamo ora esprimere, mediante espressioni logiche, le due funzioni:

- *funzione delle uscite:*

[5] E' possibile che, con il ragionamento iniziale, si riconoscano più stati interni rispetto allo stretto necessario. Come determinare formalmente eventuali stati equivalenti, e di conseguenza eliminare quelli ridondanti, è descritto in [GER].

$$z = \omega\,(x, y)$$

- *funzione di transizione dello stato interno:*

$$Y = \sigma\,(x, y)$$

La realizzazione di tali funzioni mediante reti combinatorie darà luogo alla *parte combinatoria della rete sequenziale* cercata, la quale verrà completata con la "richiusura" dello stato interno per collegare Y a y (su quest'ultimo aspetto torneremo in seguito), come mostrato in Fig. 2.

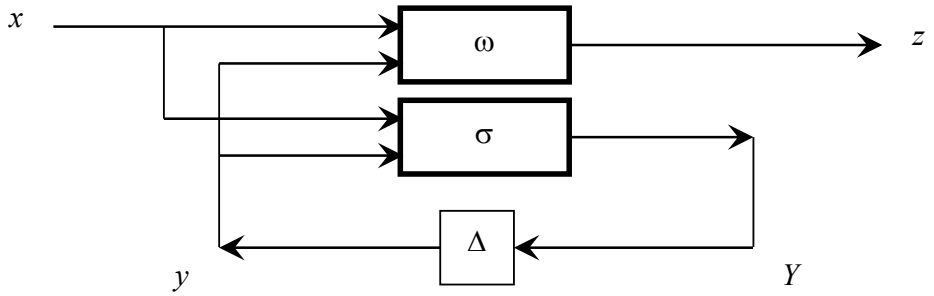


Figura 2

Dal grafo di stato si ricava la seguente *tabella di verità* per ω e σ:

		ω	**σ**
y	*x*	*z*	*Y*
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

Si ricavano quindi le espressioni logiche che definiscono le funzioni delle uscite e di transizione dello stato interno:

$$z = x\,y$$

$$Y = x\,\overline{y}$$

A questo punto è immediato realizzare lo schema delle reti ω e σ (che, nel caso specifico, risultano ad un solo livello di logica).

Realizziamo ora lo stesso riconoscitore di sequenza mediante un automa secondo modello matematico di *Moore*. Il grafo di stato è mostrato in Fig. 3.

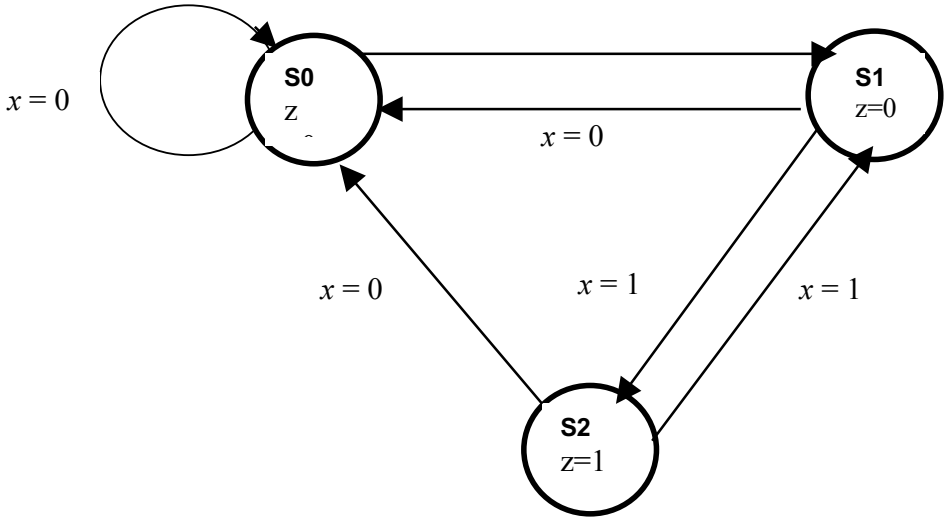


Figura 3

Poiché nel modello di Moore lo stato di uscita è univocamente associato allo stato interno presente, sono ora necessari tre stati interni :

- S0, con stato di uscita $z = 0$, del tutto equivalente a quello dell'automa di Mealy e considerato stato iniziale;
- S1, con stato di uscita $z = 0$, corrispondente allo stato S1 dell'automa di Mealy;
- S2; con stato di uscita $z = 1$, corrispondente alla situazione in cui è stata riconosciuta la sottosequenza d'ingresso "11". Si noti che lo stato S1 aveva il significato "si è presentato un 1 nella sequenza di ingresso", ma, poiché a S1 è univocamente associato lo stato di uscita $z = 0$, nel caso si presenti in ingresso ancora un 1, occorre passare ad uno stato interno cui corrisponda stato di uscita $z = 1$: questo è appunto S2.

In generale, *il numero di stati interni dell'automa di Moore è maggiore o uguale del numero di stati interni dell'automa di Mealy equivalente*. Nel nostro caso, si è verificata la relazione, piuttosto frequente, di strettamente maggiore.

Per procedere nella sintesi della rete sequenziale, diamo una *codifica degli stati interni* mediante il minimo numero di variabili ($\lceil \lg_2 3 \rceil = 2$). Indichiamo con y_1, y_2 le *variabili dello stato interno presente* e con Y_1, Y_2 *le variabili dello stato interno successivo*; una possibile codifica è:

stato int	y_1	y_2
S0	0	0
S1	0	1
S2	1	–

Ricaviamo le due funzioni che definiscono l'automa:

- *funzione delle uscite:*

$$z = \omega\ (y_1, y_2)$$

- *funzione di transizione dello stato interno:*

$$(Y_1, Y_2) = \sigma\ (x, y_1, y_2)$$

La realizzazione di tali funzioni mediante reti combinatorie darà luogo alla *parte combinatoria della rete sequenziale* cercata. Quest'ultima verrà completata con le "richiusure" dello stato interno per collegare Y_1, Y_2 a y_1, y_2 come mostrato in fig. 4.

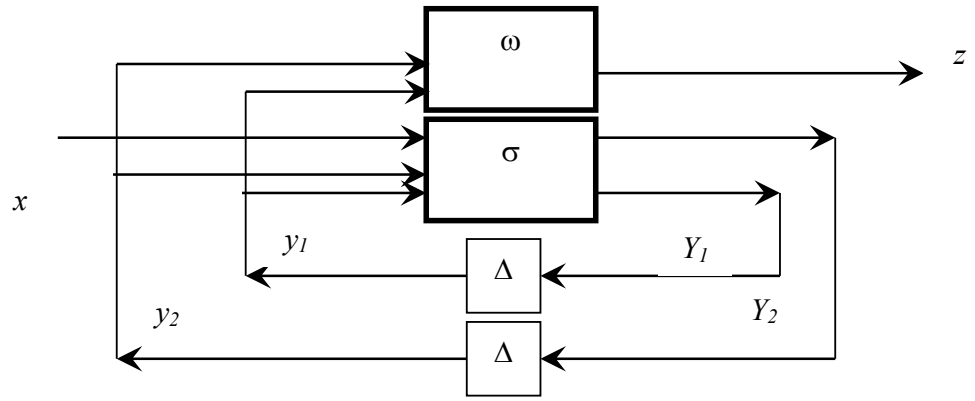


Figura 4

Dal grafo di stato si ricava la seguente tabella di verità per ω e σ:

			ω	σ	
y_1	y_2	x	z	Y_1	Y_2
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	0
0	1	1	0	1	–
1	–	0	1	0	0
1	–	1	1	0	1

Si ricavano quindi le espressioni logiche che definiscono le funzioni delle uscite e di transizione dello stato interno:

$$z = y_1$$

$$Y_1 = x\ \overline{y_1}\ y_2$$

$$Y_2 = x\ \overline{y_1}\ y_2 + x\, y_1$$

L'espressione per Y_2 è stata ricavata specificando al valore 0 l'entrata non specificata. Scegliendo, invece, di specificarla al valore 1, si ottiene:

$$Y_2 = x\ \overline{y_1}\ \overline{y_2} + x\ \overline{y_1}\ y_2 + x\, y_1 = x\ \overline{y_1}\ (\overline{y_2} + y_2) + x\, y_1 = x\,(\overline{y_1} + y_1) = x$$

A questo punto è immediato realizzare lo schema delle reti combinatorie ω e σ (nel caso specifico, la prima è a zero livelli di logica; la seconda è a due livelli di logica usando la prima espressione di Y_2, ad un solo livello di logica usando la seconda espressione minimizzata).

Verifichiamo nel nostro esempio che, partendo da stati interni iniziali equivalenti e per la stessa sequenza d'ingresso, la sequenza di uscita dell'automa di Moore è ritardata, rispetto a quella dell'automa di Mealy, di un intervallo Δ della sequenza temporale.

Lo stato iniziale è S0 per entrambe le reti.

Consideriamo la sequenza di ingresso:

$$t_0 \qquad t_1 \qquad t_2 \qquad t_3 \qquad t_4$$

1 0 1 1

Per la rete di Mealy si ha:

$\underline{t = t_0}$:

stato d'ingresso: $x(t_0) = 1$,
stato interno presente: $y(t_0) = 0$
stato interno successivo: $Y(t_0) = x(t_0)\ \overline{y(t_0)} = 1$
stato di uscita: $z(t_0) = x(t_0)\ y(t_0) = 0$

$\underline{t = t_1 = t_0 + \Delta}$:

stato d'ingresso: $x(t_1) = 0$,
stato interno presente: $y(t_1) = Y(t_0) = 1$
stato interno successivo: $Y(t_1) = x(t_1)\ \overline{y(t_1)} = 0$
stato di uscita: $z(t_1) = x(t_1)\ y(t_1) = 0$

$\underline{t = t_2 = t_1 + \Delta}$:

stato d'ingresso: $x(t_2) = 1$,
stato interno presente: $y(t_2) = Y(t_1) = 0$
stato interno successivo: $Y_1(t_2) = x(t_2)\ \overline{y(t_2)} = 1$
stato di uscita: $z(t_2) = x(t_2)\ y(t_2) = 0$

$\underline{t = t_3 = t_2 + \Delta}$:

stato d'ingresso: $x(t_3) = 1$,
stato interno presente: $y(t_3) = Y(t_2) = 1$
stato interno successivo: $Y(t_3) = x(t_3)\ \overline{y(t_3)} = 0$
stato di uscita: $z(t_3) = x(t_3)\ y_1(t_3) = 1$

Per la rete di Moore si ha:

$\underline{t = t_0}$:

stato d'ingresso: $x(t_0) = 1$,
stato interno presente: $y_1(t_0) = 0, y_2(t_0) = 0$
stato interno successivo: $Y_1(t_0) = x(t_0)\ \overline{y_1(t_0)}\ y_2(t_0) = 0,\ Y_2(t_0) = x(t_0) = 1$
stato di uscita: $z(t_0) = y_1(t_0)) = 0$ [6]

$\underline{t = t_1 = t_0 + \Delta}$:

stato d'ingresso: $x(t_1) = 0$,
stato interno presente: $y_1(t_1) = Y_1(t_0) = 0, y_2(t_1) = Y_2(t_0) = 1$
stato interno successivo: $Y_1(t_1) = x(t_1)\ \overline{y_1(t_1)}\ y_2(t_1) = 0,\ Y_2(t_1) = x(t_1) = 0$
stato di uscita: $z(t_1) = y_1(t_1) = 0$

$\underline{t = t_2 = t_1 + \Delta}$:

stato d'ingresso: $x(t_2) = 1$,
stato interno presente: $y_1(t_2) = Y_1(t_1) = 0, y_2(t_2) = Y_2(t_1) = 0$
stato interno successivo: $Y_1(t_2) = x(t_2)\ \overline{y_1(t_2)}\ y_2(t_2) = 0,\ Y_2(t_2) = x(t_2) = 1$
stato di uscita: $z(t_2) = y_1(t_2) = 0$

$\underline{t = t_3 = t_2 + \Delta}$:

stato d'ingresso: $x(t_3) = 1$,

[6] Questo stato di uscita non è significativo, in quanto non dipendente dalla sequenza d'ingresso. La sequenze di uscita da confrontare sono per la rete di Mealy quella negli istanti temporali $t_0, \ldots, t_3$, per quella di Moore quella negli istanti $t_1, \ldots, t_4$.

stato interno presente: $y_1(t_3) = Y_1(t_2) = 0, y_2(t_3) = Y_2(t_2) = 1$
stato interno successivo: $Y_1(t_3) = x(t_3)\ \overline{y_1(t_3)}\ y_2(t_3) = 1,\ Y_2(t_3) = x(t_3) = 1$
stato di uscita: $z(t_3) = y_1(t_3) = 0$

$t = t_4 = t_3 + \Delta$:
stato interno presente: $y_1(t_4) = Y_1(t_3) = 1, y_2(t_4) = Y_2(t_3) = 1$
stato di uscita: $z(t_4) = y_1(t_4) = 1$

In conclusione, si è verificato che, per la stessa sequenza d'ingresso:

	t_0	t_1	t_2	t_3	t_4
	1	0	1	1	

le sequenze di uscita sono:

	t_0	t_1	t_2	t_3	t_4
per la rete secondo il modello di Mealy:	0	0	0	1	
per la rete secondo il modello di Moore:	(0)	0	0	0	1

2.3 Reti sequenziali reali di tipo sincrono

La trattazione generale e completa di questo tipo di rete è contenuta in [GER]. In questo capitolo daremo una caratterizzazione della realizzazione di reti sequenziali sincrone più limitata, ma sufficiente per gli scopi di questo corso (studio di sistemi costituiti dall'interconnessione di unità di elaborazione).

Il modello strutturale ideale

Nel modello ideale le uscite Y_i (variabili dello stato interno successivo) della rete combinatoria σ sono collegate agli ingressi y_i (variabili dello stato interno presente) della rete combinatoria ω (solo se la rete è di Mealy) e della rete σ mediante *elementi di memoria* temporanea che introducono un *ritardo* di valore Δ (si vedano gli esempi di Fig. 2 e 4): il valore di ogni y_i diviene uguale al valore di Y_i dopo un intervallo di tempo Δ.

Nel modello ideale si assume che

a) le funzioni ω e σ abbiano ritardo nullo,

b) le variabili di ingresso della rete sequenziale varino, tra loro tutte contemporaneamente, a istanti temporali discreti distanziati di Δ .

Di conseguenza, nel modello ideale *tutti gli ingressi delle reti combinatorie* ω *e* σ*,* siano essi di tipo x_j (variabili d'ingresso) che di tipo y_i (variabili dello stato presente), variano *contemporaneamente in corrispondenza degli istanti della sequenza temporale, e solo in corrispondenza di questi*.

Problemi del modello strutturale ideale

Il suddetto funzionamento deve fare i conti con il fatto che, nella realtà, la precedente assunzione *a)* non è valida. Come discusso in F3, le reti combinatorie, come ω e σ nel nostro caso, sono caratterizzate da un certo *tempo di stabilizzazione*, cioè da un ritardo non nullo necessario per avere le uscite in forma stabile (significativa) a partire dall'istante in cui gli ingressi sono stabili. Abbiamo visto in F3 come questo ritardo può essere valutato per ogni rete combinatoria.

Il *problema della stabilizzazione* è, nel caso delle reti sequenziali, molto più serio che per le reti combinatorie.

Si parta dalla situazione in cui, ad un certo istante t, tutti gli ingressi esterni x_j e tutti gli ingressi dello stato presente y_i sono stabili: le uscite delle reti combinatorie ω e σ saranno stabili all'istante $t + t_r$, dove t_r è il tempo di stabilizzazione delle reti combinatorie stesse [7].

Se, invece, gli ingressi y_i variassero in istanti diversi rispetto agli x_j , le reti combinatorie ω e σ potrebbero non stabilizzarsi mai, in quanto le uscite della rete σ sono esse stesse ingressi delle due reti combinatorie, e della σ in particolare. Continuando a mantenere la precedente assunzione *b)*, potremmo allora pensare di scegliere il valore di Δ tale che:

$$\Delta = t_r$$

Purtroppo, nemmeno in questo modo viene risolto il problema della stabilizzazione: come sappiamo dalla F3, t_r è il **massimo** tempo di stabilizzazione di una rete combinatoria e non è mai possibile ipotizzare di conoscere il valore preciso del ritardo entro tale massimo. Di conseguenza, gli ingressi y_i delle reti ω e σ *variano in istanti temporali non coincidenti con gli istanti della sequenza temporale*, in corrispondenza dei quali (e solo di quelli) si assume che varino gli ingressi esterni x_j.

[7] Senza perdere in generalità, assumeremo che i ritardi delle due reti siano uguali. In realtà, è sufficiente considerare il ritardo della rete σ.

Verso il modello strutturale reale

La soluzione, che permetta comunque di mantenersi aderenti al modello di funzionamento sincrono, deve essere basata su *ritardi* nelle richiusure che, invece che costanti ad un certo valore come nel modello ideale, siano *variabili*.

Continuiamo a mantenere l'assunzione *b),* caratterizzandola così: le variabili di ingresso della rete sequenziale variano, tra loro tutte contemporaneamente, a istanti temporali discreti distanziati di un intervallo

$$\tau \geq t_r$$

Nelle richiusure dello stato interno inseriamo degli elementi di memoria funzionanti come "cancelli temporizzati": in *qualunque* istante varino le uscite Y_i della rete combinatoria σ, i valori di y_i assumono i valori degli Y_i *solo in istanti temporali ben determinati* ("il cancello si apre solo in istanti ben determinati").

Il problema della stabilizzazione è ora risolto imponendo che gli istanti della sequenza temporale siano distanziati dello stesso intervallo τ di cui in precedenza.

Registri impulsati

Un elementi di memoria funzionante come un "cancello temporizzato" è realizzato da un ulteriore componente hardware standard, detto *registro*, o meglio *registro impulsato*.

Un registro R di un bit è una rete sequenziale, assunta primitiva, avente un ingresso primario *a*, una uscita *b*, ed un ingresso secondario *p*. I segnali di *a* e *b* sono a *livelli*, il segnale di *p* è *periodico* e *impulsivo* (si veda la definizione in [GER 1.13.2] e in F3) ed è detto comunemente *impulso di clock*. Il comportamento di R è il seguente:

when *p* **do** *b*:= *a*

cioè: quando sull'ingresso *p* è presente impulso (valore 1 del segnale impulsivo), allora l'uscita assume il valore presente sull'ingresso; si dice anche che, quando è presente impulso, il valore di *a* viene *scritto in R* ("apertura del cancello").

Il simbolo ed il funzionamento di un registro sono illustrati in Fig. 5: la funzione di "cancello temporizzato" è evidente dal fatto che non c'è nessuna relazione tra gli istanti in cui varia l'ingresso *a* e quelli in cui varia l'uscita *b*; l'uscita, se varia, varia solo a istanti di tempo ben determinati, e cioè in corrispondenza dell'impulso di clock. Precisamente, poiché la durata δ dell'impulso non è nulla, la scrittura in R avviene sul *fronte di discesa* dell'impulso.

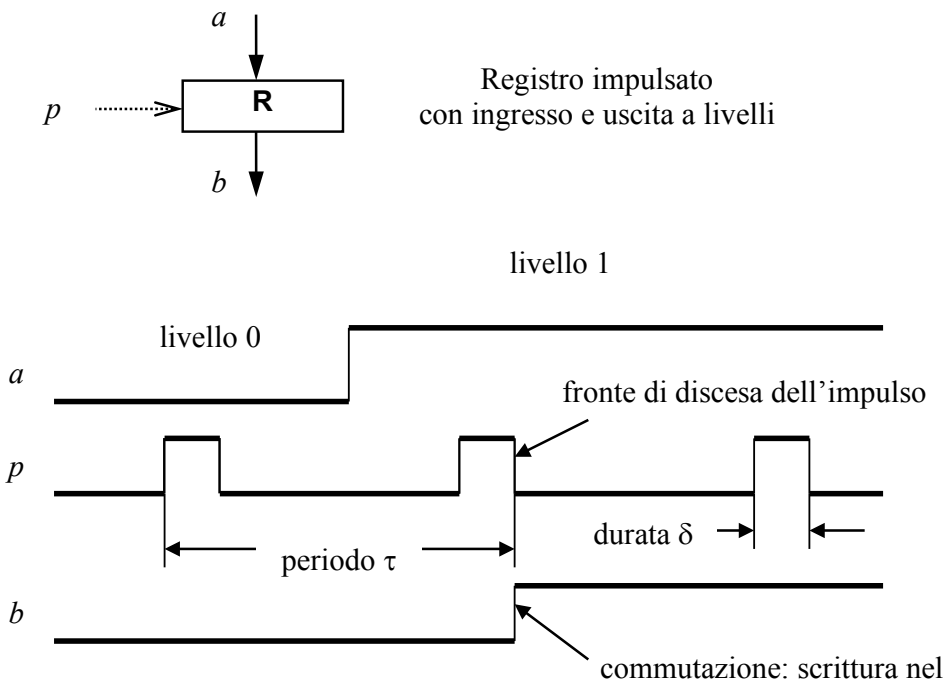


Figura 5

Easistono vari tipi di registri, o *flip-flop*. Quello definito in precedenza è di gran lunga il più usato ed è del tutto sufficiente per i nostri scopi; è chiamato *flip-flop F* o, nella documentazione tecnica, *latch*.

Un *registro a N bit* è realizzato semplicemente mettendo in parallelo N registri da 1 bit, tutti impulsati dallo stesso segnale di clock.

Corretta stabilizzazione dei registri

Come detto il segnale impulsivo *p* è *periodico*. Nel periodo va contata anche la durata δ dell'impulso (che comunque è normalmente piuttosto piccola rispetto al periodo stesso). *Occorre fare in modo che l'ingresso del registro non vari durante l'intervallo* δ; se ciò avvenisse, l'uscita sarebbe indeterminata per un tempo finito ma illimitato. Infatti, il registro è esso stesso una rete sequenziale (realizzata secondo il modello asincrono) ed il suo periodo di stabilizzazione è proprio δ.

Il fenomeno è detto dello "stato metastabile" dei registri impulsati.

Reti sequenziali sincrone LLC

A questo punto disponiamo del modello strutturale di rete sequenziale sincrona reale, mostrato in Fig. 6 per il modello di Mealy (rete con *n* variabili di ingresso, *m* variabili di uscita, $\lceil lg_2\, k \rceil$ stati interni). Esso comprende le due reti combinatorie reali ω (funzione delle uscite) e σ (funzione di transizione dello stato interno), e le richiusure dello stato interno sono realizzate con *k* registri in parallelo impulsati dallo stesso segnale di clock (registro di *k* bit).

Il modello è detto LLC: **L**evel input, **L**evel output, **C**locked, a significare che i segnali su cui si applicano le funzioni ω e σ sono ancora a livelli, mentre l'unico segnale impulsivo è quello per la sincronizzazione dei (per provocare la scrittura nei) registri.

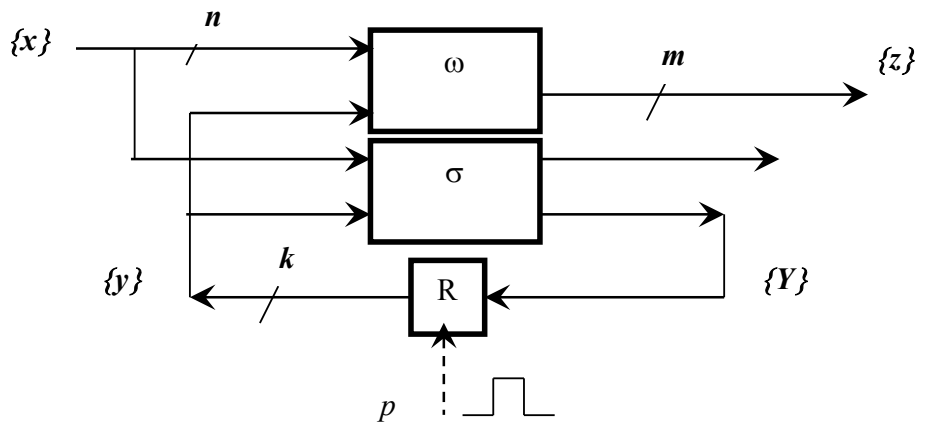


Figura 6

Il periodo τ dell'impulso è detto **ciclo di clock** della rete sequenziale.

Come visto, deve essere $\tau \geq t_r$. In realtà, per tenere conto della durata δ dell'impulso, e quindi per evitare il fenomeno dello stato metastabile dei registri, si ha che *il ciclo di clock della rete va determinato come*

$$\tau \geq t_r + \delta$$

La durata δ dell'impulso di clock può essere assunta dello stesso ordine di grandezza del tempo di stabilizzazione t_p di una porta logica (vedi F3).

Ad esempio, se t_p = 1 nsec, il ciclo di clock delle reti del cap. 2 risulta di 2 nsec. Il *generatore di impulsi di clock* ha, in questo caso, una frequenza $f = 1 / \tau$ = 500 MegaHz (0,5 GigaHz).

Sincronizzazione degli ingressi

Rimane da chiarire come assicurare la validità dell'assunzione sulla variabilità degli ingressi esterni solo in corrispondenza di istanti ben determinati e distanziati di un intervallo τ.

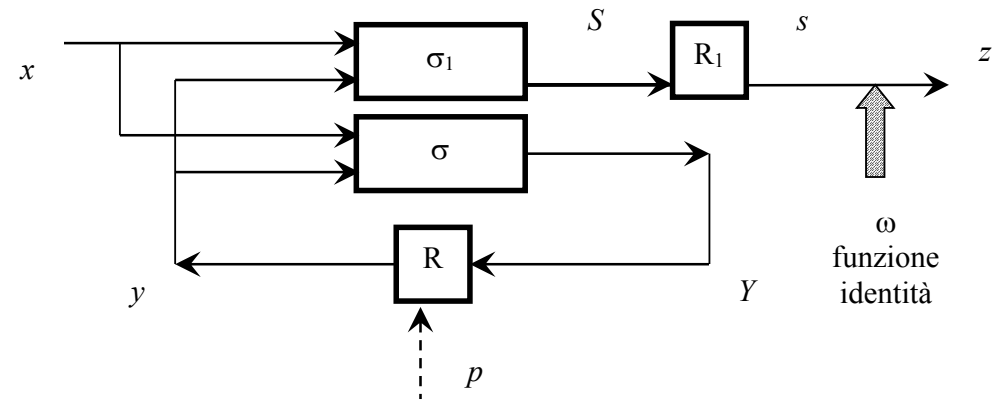
1. In certi casi questa caratteristica è assicurata dal comportamento delle reti che producono i valori d'ingresso alla rete in questione: esiste cioè una *sincronizzazione implicita* tra le reti interconnesse, che assicura la reciproca stabilizzazione entro un intervallo di durata τ. Un caso notevole sarà costituito dall'interazione tra Parte Controllo e Parte Operativa all'interno del ciclo di clock *complessivo* dell'unità di elaborazione.
2. Quando non sia possibile adottare la soluzione precedente, si ricorre ancora all'uso di registri come sincronizzatori: *sugli ingressi della rete sequenziale sono disposti altrettanti registri impulsati con lo stesso clock della rete stessa.* E' questo il caso degli ingressi esterni di una unità di elaborazione, ingressi che fanno capo alla Parte Operativa in aggiunta quelli (citati in precedenza) provenienti dalla Parte Controllo. In questo caso esiste la possibilità che si verifichi lo stato metastabile, a meno che non sia possibile imporre precisi vincoli alle relazioni tra i cicli di clock delle reti sequenziali interconnesse; altrimenti, esistono metodi per rilevare l'eventuale occorrenza di stati metastabili in modo da innescare azioni di correzione degli errori.

2.4 Casi particolari notevoli di reti di Mealy e di Moore

La trasformazione formale tra automi equivalenti di Mealy e di Moore è trattata in [GER]. Per i nostri scopi, è sufficiente considerare alcuni semplici casi particolari.

Sia data una rete sequenziale di Mealy. Un semplice modo di trasformarla in rete di Moore equivalente consiste nell'aggiungere un registro su ogni uscita: in tal modo, si aggiungono tante variabili dello stato interno per quante sono le uscite, e le uscite stesse coincidono con tali variabili prese all'uscita dei registri. La rete ottenuta, il cui modello è detto anche di *Mealy ritardato*, in generale non è minimizzata, ma la semplicità e l'economicità del procedimento sono fuori di dubbio.

In Fig. 7 è mostrata la rete di Moore equivalente alla rete di Mealy di Fig. 2 (la Fig. 2 va prima modificata per rappresentare la rete reale, sostituendo l'elemento di ritardo Δ con un registro impulsato R).



ω : $z = s$

σ : $S = x\,y,\ Y = x\ \bar{y}$

Figura 7

Si noti come la rete di Moore così ottenuta sia diversa, ma equivalente, rispetto a quella progettata direttamente nel cap. 2; in particolare, ha lo stesso numero di stati interni.

Viceversa, data una rete sequenziale di Moore, *una* rete equivalente di Mealy è ottenibile applicando alle uscite della funzione σ la stessa funzione ω della rete di Moore. Più in particolare, se le variabili di uscita della rete di Moore coincidono con le variabili dello stato interno presente, allora le uscite della rete equivalente di Mealy coincidono con le variabili dello stato successivo. Il modello della rete ottenuta prende anche il nome di *Moore anticipato*.

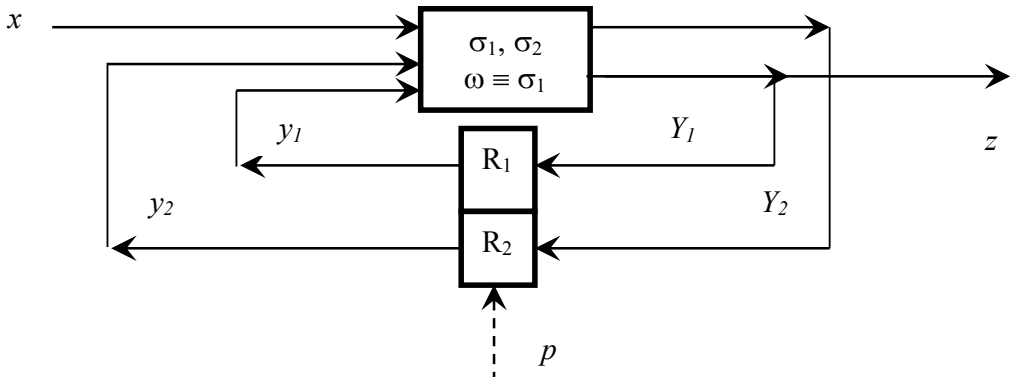
In Fig. 8 è mostrata la rete di Mealy (Moore anticipato) equivalente alla rete di Moore di Fig. 4 (questa va prima modificata sostituendo i due elementi di ritardo con un registro impulsato a due bit: R_1, R_2). Con σ_1, σ_2 si indicano le sottofunzioni di σ relative a Y_1, Y_2 rispettivamente; la funzione delle uscite di Moore anticipato è data, nell'esempio, da σ_1.

Si noti come la rete di Mealy così ottenuta sia diversa, ma equivalente, rispetto a quella progettata direttamente nel cap. 2; in particolare, ha un maggior numero di stati interni.

2.5 Reti sequenziali realizzate con componenti standard e loro analisi

Come detto, nella parte del corso dedicata al firmware saremo interessati a sintetizzare, per ogni unità di elaborazione, due specifiche reti sequenziali: la Parte Controllo e la Parte Operativa, per ognuna delle quali adotteremo due diverse metodologie.

La *Parte Controllo* ha spesso un numero relativamente basso di stati (interni, d'ingresso, di uscita); inoltre, dal microprogramma è possibile ricavare formalmente una sua descrizione tipo grafo di stato. Di conseguenza, la sua sintesi viene effettuata con il metodo classico visto finora; le reti combinatorie ω e σ sono realizzate mediante porte AND, OR, NOT. Quando (come nel caso di processori general-purpose) la complessità di sintesi risulti troppo elevata a causa dell'elevato numero di stati, le due reti combinatorie vengono realizzate con componenti logici di tipo memoria.



ω : $z = x\ \overline{y_1}\ y_2$

σ : $Y_1 = x\ \overline{y_1}\ y_2$

$Y_2 = x$

Figura 8

La *Parte Operativa* presenta, anche per le unità più semplici, un numero di stati (interni, d'ingresso, di uscita) relativamente molto grande: basti pensare che i dati su cui si opera sono tipicamente parole a N bit: per N = 32 il numero di stati è dell'ordine di molti miliardi ! In questo caso l'approccio è completamente diverso e si basa sulla composizione di componenti standard a partire dalla specifica delle operazioni elementari delegate alla Parte Operativa, specifica ricavata formalmente dal microprogramma. Il procedimento è di complessità lineare nel numero dei registri (a N bit) e degli operatori aritmetico-logici.

In questo capitolo intendiamo esaminare le caratteristiche di reti sequenziali realizzate come combinazione di componenti standard e studiare come si effettua la loro analisi.

I componenti standard usati sono:

- *registri* impulsati a N bit,
- reti combinatorie (vedi sez. 1):
  - *reti di calcolo* aritmetico-logiche sotto forma di reti multifunzione o *ALU*,
  - *commutatori*,
  - (talvolta) *selezionatori*.

Come esempio, consideriamo la rete sequenziale di Fig. 9. I collegamenti con freccia piena sono su parola (32 bit), gli altri su singolo bit. A è un registro impulsato di 32 bit; M e Q sono ingressi esterni di 32 bit ciascuno (provenienti da altre unità), rappresentanti numeri interi in complemento a 2. K1, K2 sono commutatori a due ingressi di N bit. La ALU è capace di eseguire somma o sottrazione; il suo valore di uscita U è inviato all'esterno (ad un'altra unità) ed anche scritto in A *ad ogni ciclo di clock*; l'uscita secondaria S è il Flag del segno del risultato (inviato alla Parte Controllo). α_1, α_2, α_a sono variabili di controllo (provenienti dalla Parte Controllo).

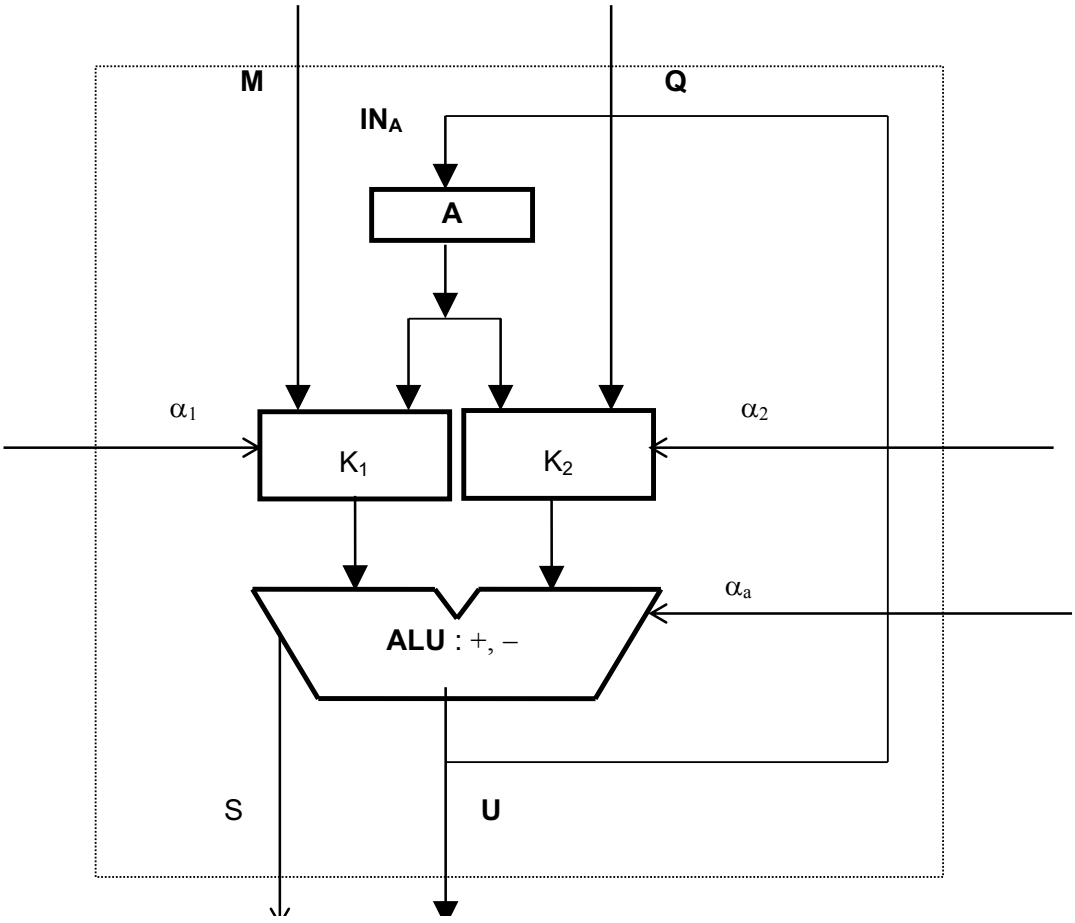


Figura 9

Assumiamo che *tutti gli ingressi rimangano stabili durante un qualsiasi ciclo di clock* (applicando le tecniche di cui al cap. 3, sincronizzazione degli ingressi).

Lo *stato interno* è dato dal contenuto del registro A. Si hanno quindi 2^{32} possibili stati interni. Secondo lo schema generale, *l'ingresso di A* (indicato con IN_A in Fig. 9) *rappresenta lo stato interno successivo, l'uscita di A rappresenta lo stato interno presente.*

Gli *stati d'in*gresso sono rappresentati dalle combinazioni dei bit di M, Q, α_1, α_2, α_a. Si hanno quindi 2^{67} possibili stati d'ingresso.

Gli *stati di uscita* sono rappresentati dalle combinazioni dei bit delle uscite U, S della ALU. Si hanno quindi 2^{33} possibili stati di uscita.

Il modello matematico di questa rete è quello di *Mealy*. Infatti, *ad ogni ciclo di clock*, l'uscita dipende dai valori di A, M, Q, α_1, α_2, α_a. Si noti che, se anche M e Q fossero memorizzati in registri (andando così a far parte dello stato interno), l'uscita della rete dipenderebbe ugualmente, *ad ogni ciclo di clock*, non solo dallo stato interno, ma anche dallo stato d'ingresso attraverso le variabili di controllo.

Dal punto di vista operativo, *ad ogni ciclo di clock* questa rete è in grado di eseguire, a seconda del valore delle variabili di controllo α_1, α_2, α_a, una delle seguenti *operazioni elementari*:

$M + Q \rightarrow A$	$M - Q \rightarrow A$
$A + Q \rightarrow A$	$A - Q \rightarrow A$
$M + A \rightarrow A$	$M - A \rightarrow A$
$A + A \rightarrow A$	$A - A \rightarrow A$

Inoltre, ad ogni ciclo di clock, il risultato U dell'operazione eseguita dalla ALU, ed il segno S di tale risultato, sono resi disponibili in uscita.

Effettuiamo l'**analisi** della rete sequenziale data. Ciò significa ricavare, dato lo schema, la definizione della funzione delle uscite ω e della funzione di transizione dello stato interno s.

Essendo improponibile esprimere queste funzioni sotto forma di tabella di verità o di espressioni logiche, adotteremo un *formalismo tipo linguaggio di programmazione* (vedi F3).

Si ha:

- *funzione delle uscite*:

  $(U, S) = \omega\ (A, M, Q, \alpha_1, \alpha_2, \alpha_a)$, dove:

  U = **case** $\alpha_1, \alpha_2, \alpha_a$ **of**

  0 0 0 : M + A

  0 0 1 : M – A

  0 1 0 : M + Q

  0 1 1 : M – Q

  1 0 0 : A + A

  1 0 1 : A – A

  1 1 0 : A + Q

  1 1 1 : A – Q

  S = **case** $\alpha_1, \alpha_2, \alpha_a$ **of**

  0 0 0 : segno(M + A)

  0 0 1 : segno(M – A)

  0 1 0 : segno(M + Q)

  0 1 1 : segno(M – Q)

  1 0 0 : segno(A + A)

  1 0 1 : segno(A – A)

  1 1 0 : segno(A + Q)

  1 1 1 : segno(A – Q)

- *funzione di transizione dello stato interno*:

  $IN_A = \sigma\ (A, M, Q, \alpha_1, \alpha_2, \alpha_a)$ = **case** $\alpha_1, \alpha_2, \alpha_a$ **of**

  0 0 0 : M + A

  0 0 1 : M – A

  0 1 1 : M – Q

  1 0 0 : A + A

  1 0 1 : A – A

  1 1 0 : A + Q

  1 1 1 : A – Q

Consideriamo ora la variante alla rete precedente, mostrata in Fig. 10.

Su tutti i collegamenti di uscita sono stati inseriti dei registri (OUT, SIGN). La nuova rete sequenziale risponde quindi al modello di *Moore*. In effetti, è agevole rendersi conto che il

passaggio da una rete sequenziale all'altra è ottenuto mediante le semplici trasformazioni di equivalenza del cap. 4: data la rete di Fig. 9, quella di Fig. 10 risponde al modello di *Mealy ritardato*; data la rete di Fig. 10, quella di Fig. 9 risponde al modello di *Moore anticipato*.

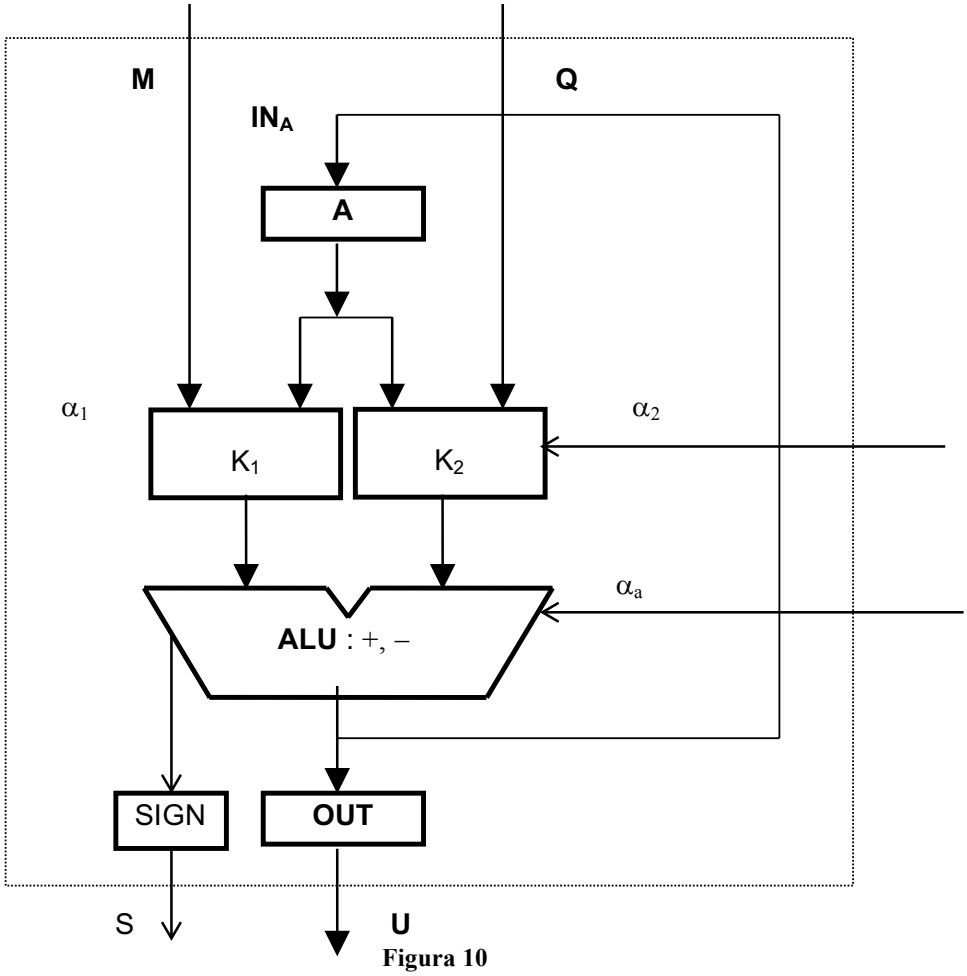


Figura 10

L'**analisi** della rete di Fig. 10 è espressa come segue:

- variabili di ingresso: M, Q, α_1, α_2, α_a; 2^{67} stati di ingresso
- variabili di uscita: U, S; 2^{33} stati di uscita
- variabili dello stato interno presente: A, OUT, SIGN; 2^{65} stati interni
- variabili dello stato interno successivo: IN_A, IN_{OUT}, IN_{SIGN}
- *funzione delle uscite*:

  (U, S) = ω (A, OUT, SIGN), dove:

U = OUT,

S = SIGN

- *funzione di transizione dello stato interno*:

  $(IN_A, IN_{OUT}, IN_{SIGN}) = \sigma$ (A, OUT, SIGN, M, Q, α_1, α_2, α_a), dove:

  $IN_A = IN_{OUT}$ = **case** α_1, α_2, α_a **of**

  0 0 0 : M + A

  0 0 1 : M – A

  0 1 0 : M + Q

  0 1 1 : M – Q

  1 0 0 : A + A

  1 0 1 : A – A

  1 1 0 : A + Q

  1 1 1 : A – Q

  IN_{SIGN} = **case** α_1, α_2, α_a **of**

  0 0 0 : segno(M + A)

  0 0 1 : segno(M – A)

  0 1 0 : segno(M + Q)

  0 1 1 : segno(M – Q)

  1 0 0 : segno(A + A)

  1 0 1 : segno(A – A)

  1 1 0 : segno(A + Q)

  1 1 1 : segno(A – Q)

Abilitazione alla scrittura nei registri

Aggiungiamo alle operazioni elementari (M + Q $\rightarrow$ A, ecc), eseguibili dalle rete sequenziale di Fig. 9, la possibilità di *non modificare* il registro A, cioè di non scrivervi il risultato disponibile all'uscita della ALU (che comunque è presente sull'uscita U ad ogni ciclo di clock).

Allo scopo, dotiamo ogni registro di un'ulteriore variabile di controllo (a livelli), indicata con β_A in Fig. 11, avente il significato di *abilitare / disabilitare la scrittura* in A. Il segnale β_A è messo in AND con l'impulso di clock, quindi la scrittura in A è abilitata solo se $\beta_A = 1$.

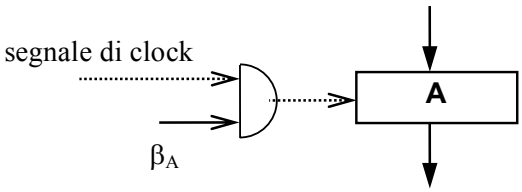


Figura 11

La nuova rete sequenziale, che modifica quella di Fig. 9, è mostrata in Fig. 12. Nella rappresentazione convenzionale non viene mostrato l'ingresso impulsivo in AND con β_A.

Il valore presente sull'ingresso di A non è significativo quando $\beta_A = 0$, in quanto tale valore non verrà scritto in A nello stesso ciclo di clock. La *funzione di transizione dello stato interno* si esprime allora come segue:

$IN_A = \sigma (A, M, Q, \alpha_1, \alpha_2, \alpha_a, \beta_A)$ = **when** $\beta_A = 1$ **do**

case $\alpha_1, \alpha_2, \alpha_a$ **of**

0 0 0 : M + A

0 0 1 : M – A

0 1 0 : M + Q

0 1 1 : M – Q

1 0 0 : A + A

1 0 1 : A – A

1 1 0 : A + Q

1 1 1 : A – Q

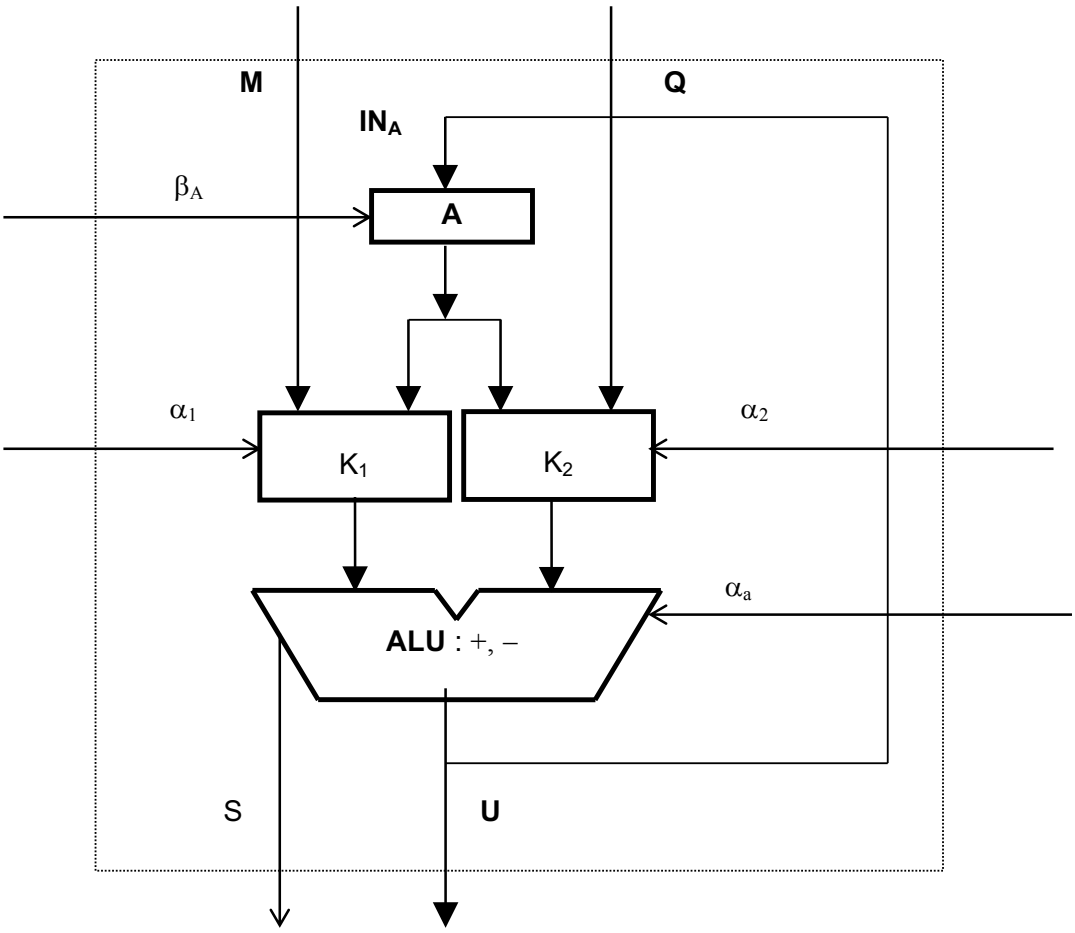


Figura 12

2.6 Esercizi

1. Un contatore modulo 2 è una rete sequenziale C a un ingresso x ed una uscita z, tale che $z(t)$ è uguale alla somma modulo 2 di tutti i valori di $x(time)$ nella sequenza di ingresso con $0 \leq time \leq t$. Sintetizzare C secondo il modello di Mealy e secondo il modello di Moore.

2. Sintetizzare, secondo il modello di Mealy e secondo il modello di Moore, una rete sequenziale a due ingressi x, y ed una uscita z, tale che: per $y = 0$ z assume il valore 1 se e quando x cambia di livello; se $y = 1$ z assume il valore 0 per qualunque valore di x.

3. Valutare il ciclo di clock delle reti sequenziali di Fig. 9 e 10 in funzione del ritardo t_p di una porta logica, assumendo che il tempo di stabilizzazione della ALU sia uguale a 5 t_p e la durata dell'impulso di clock sia uguale a t_p.

4. Fare l'analisi della rete sequenziale di Fig. 13 e valutarne il ciclo di clock nelle stesse ipotesi dell'esercizio precedente. Le operazioni di shift (R = verso destra, L = verso sinistra) sono applicabili sia al primo che al secondo ingresso della ALU.

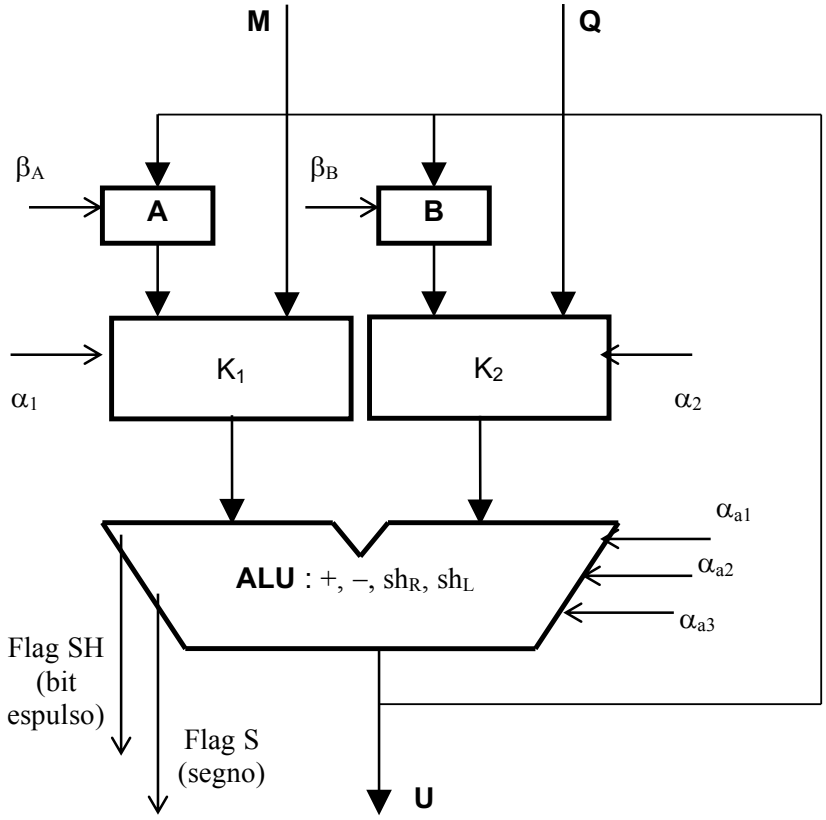


Figura 13

5. Trasformare la rete sequenziale di Fig. 13 nella rete equivalente secondo l'altro modello matematico e farne l'analisi.

6. Modificare la rete di Fig. 13 in modo da assicurare che gli ingressi M, Q rimangano stabili per tutta la durata del ciclo di clock. Fare l'analisi della rete così modificata.