Capitolo 3: La macchina a stati finiti

Al più alto livello di astrazione il progetto logico impiega un modello, la cosiddetta **macchina a stati finiti**, per descrivere e classificare comportamento e struttura di ogni blocco di hardware. Il modello è importante in due diversi momenti dell'attività progettuale:

- all'inizio del procedimento di sintesi, quando occorre puntualizzare che cosa dovrà fare la macchina da costruire e come dovrà essere organizzata;
- alla fine del procedimento di analisi, quando si vuole capire il comportamento e le modalità d'impiego di una macchina che altri hanno in precedenza costruito.

3.1- Macchine combinatorie e sequenziali

Elaborazione di stringhe

Tramite i segnali binari, ogni macchina che elabora informazioni riceve dall'esterno **sequenze temporali** di simboli **i** appartenenti ad un certo **alfabeto di ingresso I** e restituisce all'esterno **sequenze temporali** di simboli **u** appartenenti ad un certo **alfabeto di uscita U**.

Digitale è sinonimo di **discreto**. Le stringhe di bit di codifica hanno una prefissata lunghezza ed i simboli di informazione appartengono di conseguenza ad **insiemi discreti**.

Anche il tempo di funzionamento è **discretizzato**. L'acquisizione di un nuovo simbolo d'ingresso può verificarsi soltanto in corrispondenza di istanti di tempo ben distanziati uno dall'altro: la macchina ha infatti bisogno di tempo per interpretare correttamente l'informazione che le viene fornita in ingresso, per sottoporla a tutte le elaborazioni richieste e per attribuire al termine all'informazione d'uscita una durata che ne consenta la corretta interpretazione da parte di chi la deve ricevere. Due sono dunque gli aspetti che occorre mettere bene in evidenza nella descrizione del processo **P** che trasforma le stringhe d'ingresso in stringhe d'uscita:

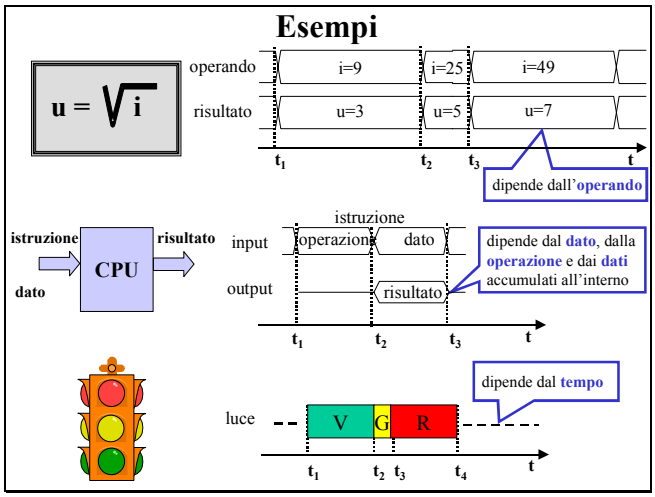
- la **discretizzazione del tempo** di funzionamento della macchina.
- la **dipendenza**, in ogni intervallo di funzionamento, del simbolo d'uscita dalla sequenza di simboli d'ingresso pervenuta fino a quel momento.

E' abbastanza intuitivo prevedere che ci saranno tre differenti tipi di relazione ingresso/uscita: il simbolo d'uscita può infatti dipendere o dal solo simbolo presente in ingresso, o da questo e da quelli che l'hanno preceduto, o dal puro e semplice "trascorrere del tempo".

ESEMPI - L'alfabeto di uscita del semaforo contiene tre colori; quello d'ingresso i comandi per il "fuori servizio" (luci spente), per il "servizio notturno" (giallo intermittente) e per il "servizio diurno" (sequenza verde-giallo-rosso). Nei due ultimi casi il processo P *dipende dal tempo*, dovendo modificare lo stato delle lampade ad istanti prefissati.

Il risultato fornito da una macchina che calcola la radice quadrata di un numero *dipende solo dal valore dell'operando* che le viene fornito.

In una CPU entra dapprima un ordine di operazione (ad esempio *somma*, *moltiplica*, ecc.) e poi il valore di uno dei due operandi (l'altro è il risultato di operazioni precedenti ed è memorizzato all'interno della macchina). Il valore di uscita *dipende* dal *valore in ingresso*, dal *precedente ordine di operazione* e dalle *istruzioni* e dai *dati* che in istanti ancora precedenti hanno guidato il funzionamento della macchina.



Vediamo di dare una descrizione formale a queste intuizioni. A tal fine definiamo **evento**

- il **verificarsi** di una **modifica del valore d'ingresso**,
- lo **scadere** di un prefissato **intervallo di tempo**.

Per esprimere la dipendenza dell'uscita dagli eventi indichiamo con

- $t_0, t_1, .., t_{n-1}, t_n$ una sequenza di istanti in cui si sono verificati eventi,
- $i(t_0), i(t_1), \dots, i(t_{n-1}), i(t_n)$ la corrispondente sequenza di valori d'ingresso,
- $s(t_0)$ la condizione, o **stato**, in cui la macchina si trovava all'istante iniziale.

L'espressione dell'ennesimo simbolo d'uscita è dunque:

$$u(t_n) = P(s(t_0), i(t_0), i(t_1), \dots, i(t_{n-1}), i(t_n)).$$

Ogni macchina che ha questo comportamento è detta **sequenziale.**

ESEMPI - L'*orologio* è una macchina sequenziale. L'ora indicata sul quadrante dipende dalla posizione delle lancette ad un certo istante iniziale e dal numero di comandi di avanzamento che esse hanno successivamente ricevuto.

Lo *sportello Bancomat* è una macchina sequenziale. Per accedere al servizio occorre aspettare che la macchina sia "pronta" e digitare poi una sequenza di cinque cifre; l'accesso viene dato se e solo se l'intero numero corrisponde al "segreto" che Banca e Cliente hanno concordato.

I blocchi di *Controllo* e di *Percorso dei dati* che compongono ogni macchina digitale complessa sono sequenziali, fornendo risposte che dipendono dal tempo e dalla sequenza di stimoli ricevuti.

Il *relè ad autoritenuta* è una macchina sequenziale. Per avviare e poi mantenere il flusso di corrente nella bobina occorre premere e poi rilasciare il pulsante M; per azzerare la corrente occorre premere e poi rilasciare A.

Un caso particolare di macchina sequenziale è la macchina **combinatoria**, in cui il simbolo d'uscita dipende solo dal simbolo d'ingresso contemporaneo:

$$u(t_n) = P(i(t_n))$$

La relazione di causa/effetto di una macchina combinatoria è dunque una **funzione**

$$P: I \rightarrow U$$

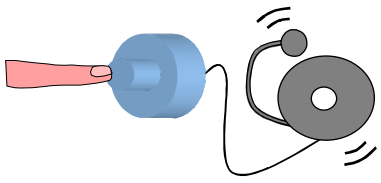
e la sua descrizione **non richiede alcuna indicazione di tempo**: $u = P(i)$.

ESEMPI – Le *disposizione in serie* e/o *in parallelo* di interruttori (v. cap.1) realizzano macchine combinatorie.

Il normale *campanello* di casa è una macchina combinatoria. La situazione di premuto/rilasciato che riusciamo ad imporre al pulsante con il dito (l'alfabeto d'ingresso ha quindi due simboli) genera o meno lo spostamento del "batocchio" contro la superficie della "suoneria" e determina la presenza/assenza dell'effetto sonoro (anche l'alfabeto d'uscita è quindi formato da due soli simboli). La funzione svolta dalla macchina è tabulata in figura.

Esistono anche campanelli che generano particolari effetti sonori (ad es. *din-din-din-don*). A tal fine è necessario che ogni pressione del pulsante determini una sequenza di colpi di durata diversa sulla suoneria In questi casi la macchina che muove il batocchio non è combinatoria, ma sequenziale: deve infatti compiere azioni diverse in istanti diversi.

Un esempio di macchina combinatoria: il campanello



u = F(i)

i: Pulsante	*u: Suoneria*
Premuto	Suono
Rilasciato	Nessun Suono

Le *tavole matematiche*, il *dizionario* ed il *regolo* sono in linea di principio strumenti di tipo combinatorio.

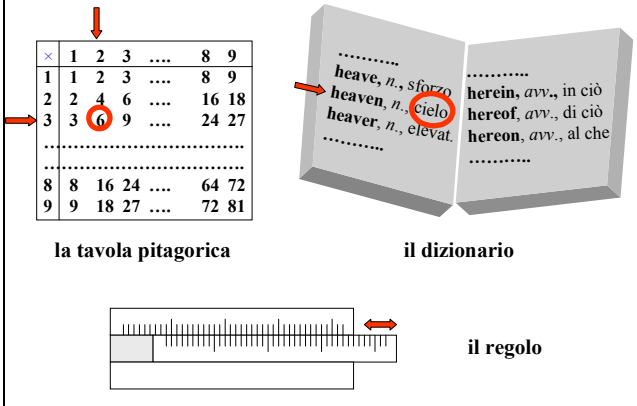
In realtà la risposta si ha solo dopo un certo tempo, dato che il loro uso mette in gioco azioni che devono essere svolte direttamente dal loro utilizzatore.

Si noti che per dare una dimensione contenuta ad una tavola matematica si deve necessariamente limitare il numero di operandi presi in considerazione: in figura ad esempio è tabulato il prodotto di due cifre decimali.

Questo non impedisce l'uso della tavola quando si deve calcolare il prodotto di numeri formati da due o più cifre: basta infatti eseguire una sequenza ordinata di moltiplicazioni tra due cifre, di moltiplicazioni per 10 e di addizioni.

Un problema combinatorio può dunque essere risolto con un **algoritmo**: per eseguirne i passi in modo automatico è necessaria una macchina sequenziale.

Altri esempi



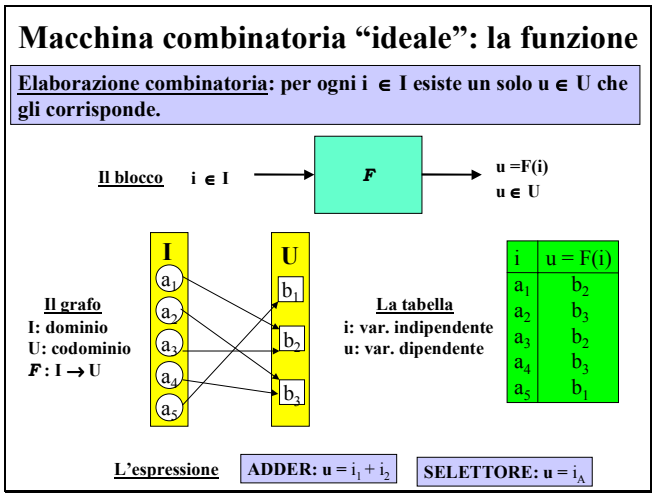
la tavola pitagorica **il dizionario**

il regolo

La macchina combinatoria

La *funzione* da I a U che caratterizza il comportamento di una macchina combinatoria è descrivibile con tre diverse tecniche:

- il *grafo* con rami orientati che partono dagli elementi del *dominio* e raggiungono elementi del *codominio*,
- la *tabella* che elenca i valori della *variabile indipendente* e li affianca con il corrispondente valore della *variabile dipendente*,
- l'*espressione*, una stringa di *costanti*, *variabili* e *simboli di operazione* che fornisce il valore d'uscita per ogni possibile valore d'ingresso.

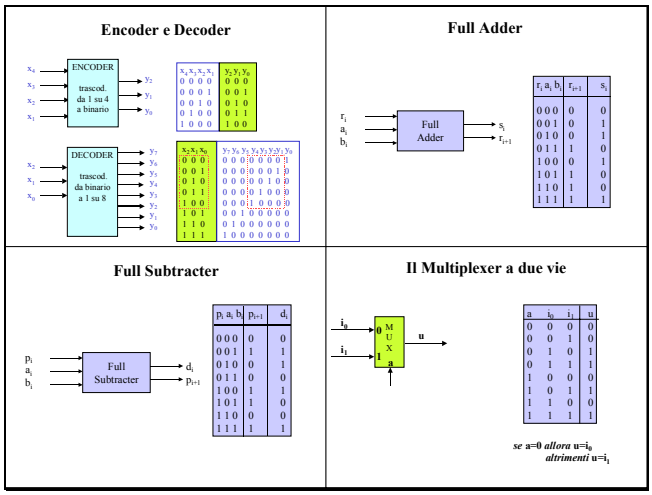


CASI DI STUDIO – L'Encoder, il Decoder, il Full Adder, il Full Subtracter ed il Multiplexer sono macchine combinatorie.

La codifica binaria dei loro ingressi e delle loro uscite determina tabelle di comportamento contenenti soltanto "zeri" e "uni".

L'addizione tra due numeri in base 2 è affidabile ad un blocco combinatorio (lo chiameremo ADDER) con due frecce d'ingresso $\mathbf{i_1}$, $\mathbf{i_2}$ ed una freccia d'uscita **u**. La relazione ingresso/uscita può essere descritta dall'espressione $\mathbf{u = i_1 + i_2}$.

Il comportamento del Mux a due vie d'ingresso $\mathbf{i_0}$, $\mathbf{i_1}$ può essere descritto dall'espressione $\mathbf{u = i_A}$, ove **A** è la cifra binaria che *seleziona* il valore d'ingresso da riprodurre sull'uscita.



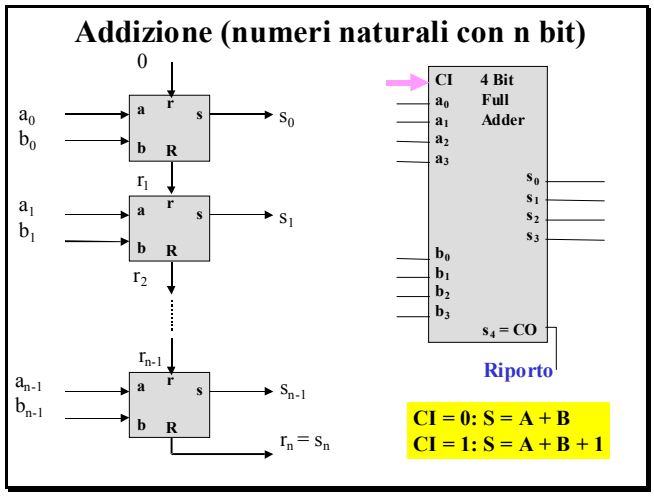
Il modello della macchina combinatoria ha un'importante proprietà: **la disposizione in serie e/o in parallelo di macchine combinatorie è ancora una macchina combinatoria**.

CASO DI STUDIO - L'addizione di numeri binari di **n** bit richiede la **disposizione in cascata** di altrettanti FULL ADDER.

Il numero di FULL ADDER integrati all'interno di un solo chip è progressivamente aumentato (2, 4, 8, 16, 32, 64); per semplicità di disegno in figura è considerato il caso del 4 BIT-FULL ADDER.

Il bit CI (*carry in*) consente di disporre in serie 2, 3, 4, ecc. 4 BIT-FULL ADDER per poter eseguire l'addizione di numeri con 8, 12, 16, ecc. bit. CI può essere visto anche come un bit di programmazione del comportamento: **0** fornisce la somma dei due operandi, **1** la incrementa di un'unità.

Il bit **riporto** CO (*carry out*) è la cifra più pesante del risultato della addizione; nel caso di disposizione in cascata ogni CO deve essere collegato al CI dell'ADDER successivo.



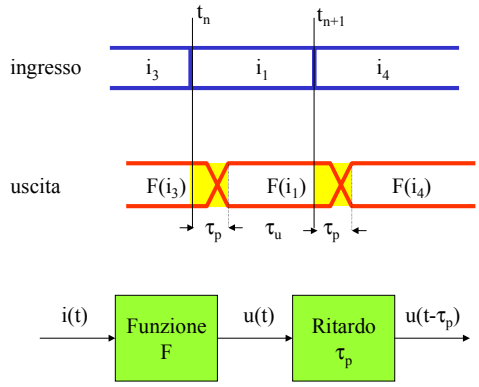
Dalla precedente proprietà discende che una macchina combinatoria può essere realizzata disponendo in serie e/o in parallelo degli interruttori. Non bisogna però dimenticare che questo modello è una rappresentazione astratta del comportamento: l'aspetto volutamente omesso e di cui occorre tener conto quando si usa la corrispondente macchina fisica è che **dopo una modifica di**

ingresso occorre aspettare un po' di tempo prima di poter riscontrare la corrispondente modifica in uscita.

L'intervallo di mantenimento di uno stesso valore d'ingresso deve essere non inferiore alla somma del *ritardo* τ_p con cui la macchina presenta il risultato e del tempo τ_u che la destinazione impiega per acquisirlo.

Una **macchina combinatoria reale** ha dunque un ritmo massimo di funzionamento, lo *throughput*, che dipende sia dalla tecnologia impiegata per realizzarla, sia dallo throughput delle macchine con cui comunica. Per evidenziarlo, nella descrizione strutturale è usuale indicare un **blocco di ritardo** *a valle* di quello che rappresenta la funzione (v. pag.17).

Macchina combinatoria "reale": throughput



La macchina sequenziale

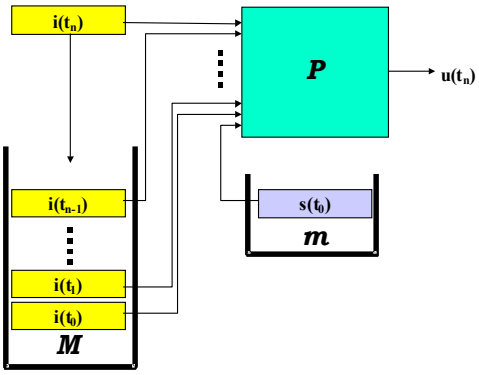
Una macchina sequenziale è formata da una parte "combinatoria" e da una parte "memoria": quest'ultima deve infatti consentire di impiegare come argomenti della prima tutti i precedenti simboli d'ingresso e la condizione iniziale da cui la macchina è partita.

In figura è indicata una organizzazione interna che si avvale di due memorie:

- **M** per i dati precedenti,
- **m** per lo stato iniziale.

Questo modello può però essere impiegato solo quando l'uscita dipende da un numero **finito e prefissato** di eventi. La limitazione è notevole; esistono comunque casi in cui può essere utilmente impiegato.

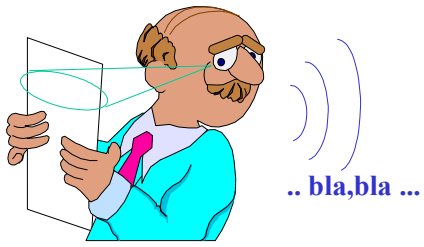
Macchina sequenziale: memorie e funzione



ESEMPI - Un esempio di elaborazione di una stringa di lunghezza finita si ha nel caso della traduzione di un testo da una lingua all'altra. Se la traduzione è fatta da una macchina, la sua memoria **M** deve essere riempita con tutti i caratteri che compongono una frase della lingua originaria (ad es. si può prevedere un migliaio di caratteri); la memoria **m** deve invece essere stata preventivamente caricata con il dizionario e con le regole sintattiche di entrambe le lingue.

Anche il Bancomat elabora una stringa di lunghezza nota. Per decidere se dare o meno accesso al servizio, la macchina deve partire da una situazione di attesa del codice (questo *stato iniziale* viene immesso nella memoria **m** nel momento in cui viene introdotta una scheda) e deve poi controllare solo cinque cifre digitate in sequenza (ed immesse nella memoria **M**) per decidere se corrispondono o meno a quelle del proprietario della scheda. In realtà, per sicurezza, si vuole che la macchina consenta al più tre tentativi. Ciò può essere ottenuto estendendo la capacità della memoria **M** da 5 a 15 cifre (quante appunto sono necessarie per far fare tre tentativi).

Un esempio: l'interprete



Per tradurre un testo da una lingua all'altra è necessario leggere e memorizzare non solo tutti i caratteri di una parola, ma anche quelli di parole precedenti e successive.
Stato iniziale necessario per il processo di traduzione : il foglio non deve essere bianco, l'interprete deve conoscere le due lingue e non deve essere cieco e afono.

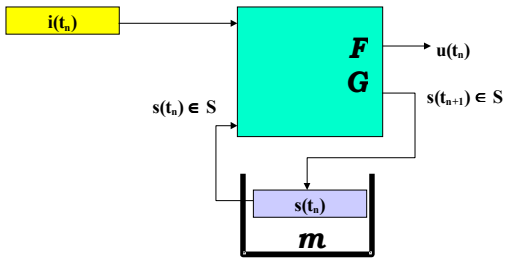
Un diverso modello di macchina sequenziale elimina la necessità della memoria **M** ed i problemi connessi al suo dimensionamento.

L'idea di base è quella di consentire alla macchina, tramite una **retroazione**, di modificare il contenuto della memoria **m** per avere a disposizione in ingresso una **informazione riassuntiva** e sempre **aggiornata** di tutto quello che è successo in precedenza.

Il nuovo modello prevede l'impiego di un terzo insieme di simboli, indicato con **S** e detto **insieme degli stati interni**, nell'ambito del quale la macchina deve di volta in volta scegliere quello che, una volta inserito in **m**, le consentirà di sapere quali eventi *salienti* si sono già verificati.

Anche l'insieme S degli stati interni, come quello di ingresso e quello di uscita, contiene un numero finito di elementi e ciò giustifica la denominazione di **macchina a stati finiti** (o di **automa a stati finiti**) attribuita al modello. L'acronimo è **FSM** da ***Finite State Machine***.

Macchine sequenziali: stato interno presente e futuro



Il comportamento della macchina a stati finiti è descritto da due funzioni e da una relazione temporale tra quello che c'è in memoria (detto **stato interno presente**) e quello che ci dovrà essere (detto **stato interno futuro**).

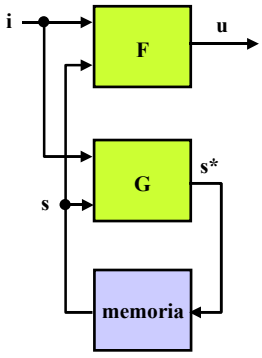
La **funzione F:S×I→U** calcola il simbolo d'uscita a partire dai simboli presenti in ingresso ed in memoria: **$u = F(s, i)$.**

La **funzione G:S×I→S** calcola, a partire dagli stessi dati, quello che dovrà essere il prossimo contenuto della memoria: **$s^* = G(s, i)$.**

La **relazione $s(t_{i+1}) = s^*(t_i)$** modella il comportamento della memoria **in retroazione** a G: nella macchina reale, per un corretto calcolo delle funzioni F,G, occorre infatti far passare una certa quantità di tempo tra l'istante in cui è in teoria disponibile lo stato futuro **(t_i)** e quello in cui questo dato può venir *scritto* in memoria e diventare così il nuovo stato presente **(t_{i+1})**.

La macchina a stati finiti (FSM: Finite State Machine)

Sistema matematico
M = {I, U, S, F, G}
formato da 3 INSIEMI
I: $\{i_1, i_2, \ldots, i_n\}$ *alfabeto di ingresso*
U: $\{u_1, u_2, \ldots, u_m\}$ *alfabeto di uscita*
S: $\{s_1, s_2, \ldots, s_k\}$ *insieme degli stati*
da 2 FUNZIONI
F : S × I → U *funzione di uscita*
G : S × I → S *funzione di stato*
e da una MEMORIA
*che mantiene il "vecchio stato" s fino a quando non è necessario sostituirlo con il "nuovo stato" s**



In quest'ultima notazione sono stati volutamente impiegati i simboli che in precedenza hanno denotato istanti di evoluzione del processo di elaborazione. La modifica dello stato interno è infatti un **evento** che la macchina si procura al suo **interno** per elaborare in modo appropriato gli **eventi esterni** causati dalle modifiche di ingresso.

Il modello è molto potente, ma lo vedremo tra poco. Prima è importante convincersi che il comportamento complessivo è ancora quello di pag. 36; a tal fine è sufficiente sostituire, più e più volte, al simbolo di stato **$s(t_i)$** la notazione **$G(s(t_{i-1}), i(t_{i-1}))$,** che indica come è stato calcolato:

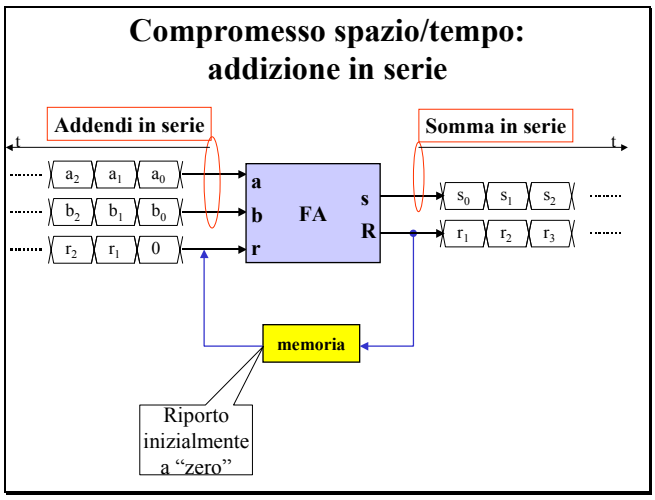
$$u(t_n) = F(s(t_n), i(t_n))$$
$$= F(G(s(t_{n-1}), i(t_{n-1})), i(t_n)) = \ldots = F(G(\ldots(G(G(s(t_0), i(t_0)), i(t_1)), i(t_2)), \ldots), i(t_n))$$
$$= P(s(t_0), i(t_0), i(t_1), \ldots, i(t_{n-1}), i(t_n))$$

Gli esempi che seguono hanno invece l'obiettivo di convincere chi ancora non ci crede che l'elaborazione di una stringa di ingresso può essere fatta sia memorizzando tutti i suoi simboli, sia, molto più convenientemente, aggiornando continuamente un riassunto di quelli già pervenuti.

ESEMPI – Consideriamo il problema dell'addizione di due numeri binari nell'ambito del compromesso spazio/tempo. Se i coefficienti degli addendi sono introdotti *in sequenza* nella macchina a partire da quelli di minor peso, non è necessario prima memorizzarli tutti e poi elaborarli: l'algoritmo di somma "colonna per colonna" ha bisogno infatti solo di un riassunto dei bit già pervenuti, il *riporto*, per generare una dopo l'altra le cifre del risultato.

Il riporto vale o 0 o 1 e costituisce lo stato interno della macchina sequenziale che esegue questo algoritmo; occorre "inizializzarlo" a zero ed aggiornarlo al termine di ogni passo.

Il calcolo dei bit di somma richiede così un solo FULL ADDER; il numero di cifre degli addendi può essere inoltre grande quanto si vuole.



Introducendo la retroazione, si può eliminare del tutto la memoria M dalla struttura della macchina per il servizio Bancomat. Una volta partita dallo stato di attesa, ad ogni digitazione la macchina controlla se è giusta od errata la cifra attualmente in ingresso e modifica poi lo stato per ricordare che cosa è avvenuto e per fare la stessa cosa sulla cifra successiva. In caso di codice segreto errato la macchina modifica di nuovo lo stato per riavviare lo stesso controllo sulla seconda e poi sulla terza immissione. Al terzo errore la macchina entra in uno stato finale di allarme, che le consente di interrompere il colloquio con l'Utente e di trattenere la scheda.

L'aspetto delicato del modello con retroazione è il meccanismo di aggiornamento dello stato interno.

Abbiamo già detto, con frasi e con formule, che il **calcolo** dello stato futuro è fatto all'istante $\mathbf{t_i}$ (cioè in corrispondenza del verificarsi dell'evento i-esimo) e che l'**assegnamento** allo *stato presente* del risultato ottenuto avviene invece all'istante $\mathbf{t_{i+1}}$ (cioè quando la macchina deve procedere all'elaborazione dell'evento i+1-esimo).

Anche le immagini possono servire. In figura si è quindi cercato di dare una ulteriore descrizione del meccanismo.

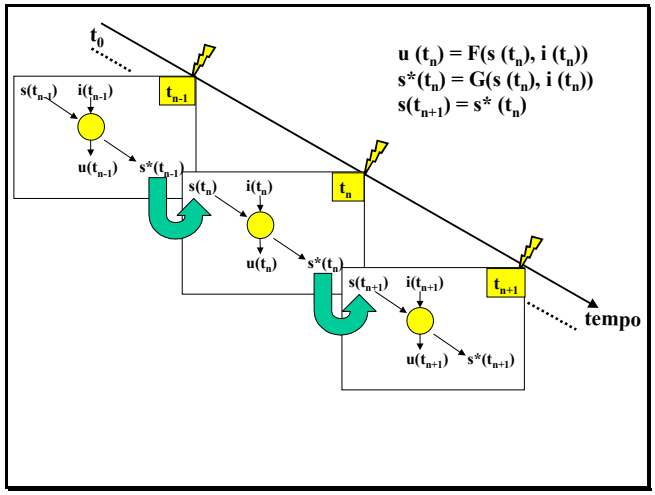


Tabella di flusso e grafo degli stati

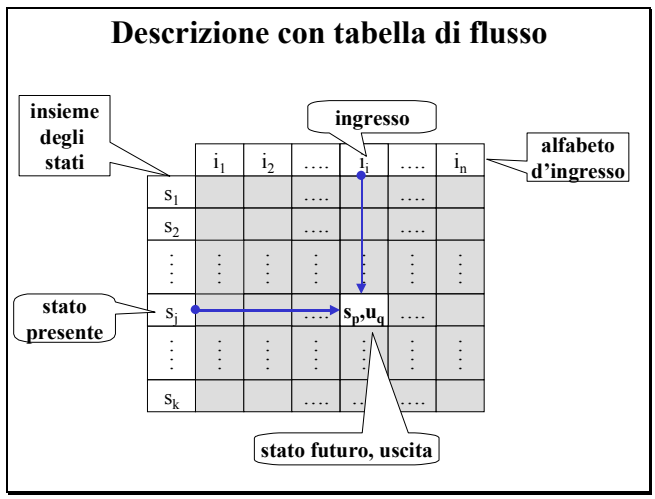
Premesso tutto ciò, è venuto il momento di prendere confidenza con le due tecniche che consentono di descrivere puntualmente il comportamento di una macchina sequenziale.

La **tabella di flusso** ha due dimensioni.

In orizzontale si elencano tutti i simboli dell'alfabeto di ingresso; in verticale si elencano gli stati che la macchina impiega per riassumere la sua storia passata.

A questo punto è disponibile una rappresentazione degli elementi del *prodotto cartesiano* S×I: in ogni incrocio, di solito separati da una virgola, si inseriscono i valori delle funzioni G e F per la coppia di coordinate che lo individuano.

Manca ogni indicazione temporale, cioè



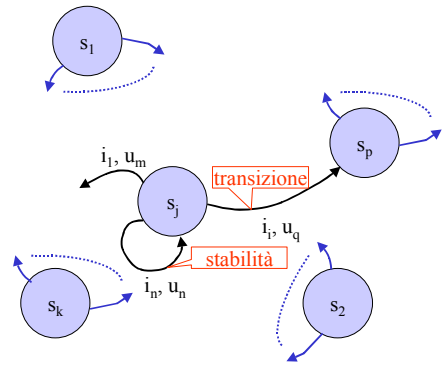
non è precisato l'istante in cui lo stato futuro diventa stato presente, ma questo è più un pregio, che un difetto. Non mettendo esplicitamente in conto l'intervallo di tempo durante il quale un certo stato resta in memoria si dispone di una immagine *più astratta*, e quindi *più generale*, del comportamento della FSM; la *temporizzazione* è infatti un argomento che può essere rinviato ad un livello di progettazione sottostante, in cui sarà invece indispensabile concentrare l'attenzione sul *come* e sul *quando* deve essere fatta l'azione di aggiornamento.

Il **diagramma degli stati** è un grafo a rami orientati. Ogni *nodo* rappresenta uno stato ed ogni *ramo* una *transizione* dallo stato presente da cui parte allo stato futuro in cui va a finire: il simbolo di ingresso che determina la transizione è indicato sul ramo, unitamente al simbolo di uscita che deve essere contemporaneamente generato.

Da ogni nodo devono uscire tanti rami quanti sono i simboli di ingresso.

Un ramo può richiudersi sul nodo da cui è partito: si parla in questo caso di *stabilità dello stato interno*. Anche il grafo degli stati non entra nel dettaglio della temporizzazione.

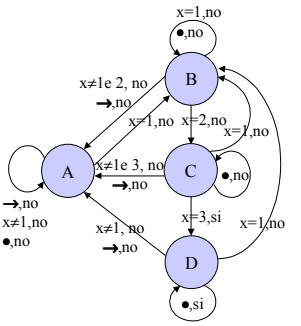
Descrizione con grafo degli stati



CASO DI STUDIO - L'alfabeto di ingresso di una certa macchina contiene le dieci cifre decimali, un simbolo di separazione (•) sempre presente nella stringa d'ingresso dopo ogni cifra ed un simbolo (→) posto all'inizio di ogni nuova stringa. L'alfabeto di uscita contiene i due simboli "si" e "no". Compito della macchina è tenere sotto continua osservazione la sequenza di simboli in ingresso per poter rispondere "si" solo se tre cifre consecutive sono rispettivamente 1,2 e 3.

Esempio

Stringa: →x• x• x• x• x• x• x• … con x: {0,1,..,9}
Risposta: "si" se x• x• x• = 1• 2• 3• , "no" in ogni altro caso



	→	•	1	2	3	4÷9
A	A,no	A,no	B,no	A,no	A,no	A,no
B	A,no	B,no	B,no	C,no	A,no	A,no
C	A,no	C,no	B,no	A,no	D,si	A,no
D	A,no	D,si	B,no	A,no	A,no	A,no

Giustifichiamo prima il grafo degli stati indicato in figura. Lo stato A è la condizione iniziale in cui la macchina si porta quando vede → e da cui non si muove finché non vede 1. Lo stato B le ricorda che questo evento si è verificato e la obbliga a controllare se la cifra successiva è 2. A questo punto si hanno tre casi. Se arriva 2 il nuovo stato è C; se arriva 1 si resta in B per tornare a controllare la cifra successiva; se non è né 1, né 2 si riparte da A. Lo stato C riassume il fatto che le due cifre precedenti sono state nell'ordine 1 e 2. L'ingresso 3 porta allo stato finale D con risposta "si". Il valore 1 in ingresso impone di ricominciare l'analisi della stringa a partire dallo stato B. Tutti gli altri simboli determinano una transizione verso lo stato iniziale A (si noti la possibilità di indicarli tutti su un solo ramo per avere una figura meno complessa).

La tabella di flusso non ha bisogno di giustificazione, essendo una diretta trasposizione di quanto è stato indicato nel grafo degli stati.

Tabella di flusso e grafo degli stati sono solo modi diversi per descrivere una stessa macchina. Il grafo è di solito utile all'inizio del procedimento di sintesi, quando non si sa ancora quanti stati occorrono alla macchina, ed alla fine del procedimento di analisi, quando si vuole disporre di un ausilio grafico per dedurre meglio il comportamento.

La tabella è una forma di rappresentazione del comportamento molto più ordinata ed è quindi impiegata nel secondo passo della sintesi e nel penultimo passo dell'analisi. Nel caso di progetto assistito dal calcolatore è proprio la descrizione della tabella, con un linguaggio appropriato, ad avviare ed a concludere i procedimenti automatizzati di sintesi e di analisi.

Il progetto logico impiega due differenti modelli di automa, quello che abbiamo appena esaminato, detto automa di **Mealy**, ed un suo caso particolare, l'automa di **Moore**, in cui l'uscita dipende **solo** dallo stato interno presente: **u = F(s)**.

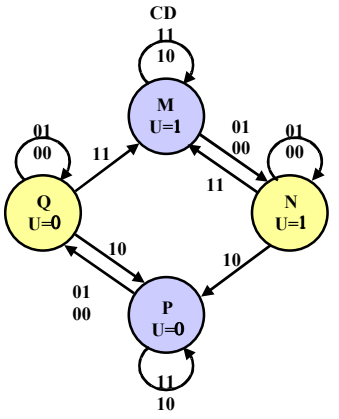
La restrizione del dominio della funzione d'uscita ha due conseguenze: nel grafo i simboli d'uscita sono direttamente indicati all'interno dei nodi; nella tabella di flusso si impiega una ulteriore colonna per indicare il valore d'uscita corrispondente ad ogni stato.

ESEMPIO – Una macchina deve esaminare il valore di un suo segnale d'ingresso **D** solo quando un altro suo segnale d'ingresso **C** presenta un fronte di salita. Il valore di **D** visto in questo momento deve essere fornito sull'uscita **U** della macchina fino a quando non si verifica un nuovo fronte di **C**. Per ipotesi **D** è costante in corrispondenza dei fronti di salita di **C**.

La macchina si porta nello stato **M** se il fronte di salita di **C** si verifica con **D=1**, nello stato **P** se invece si verifica con **D=0**. Al fronte di discesa di **C** la macchina si prepara per un nuovo "campionato" di **D**, pur mantenendo in uscita l'ultimo valore esaminato: se è in **M** si porta in **N**, se è in **P** si porta in **Q**.

La tabella di flusso indica lo stato futuro per ogni stato presente e per ogni configurazione d'ingresso; la quinta colonna indica il valore d'uscita generato da ogni stato presente.

Campionamento e ricostruzione di un segnale



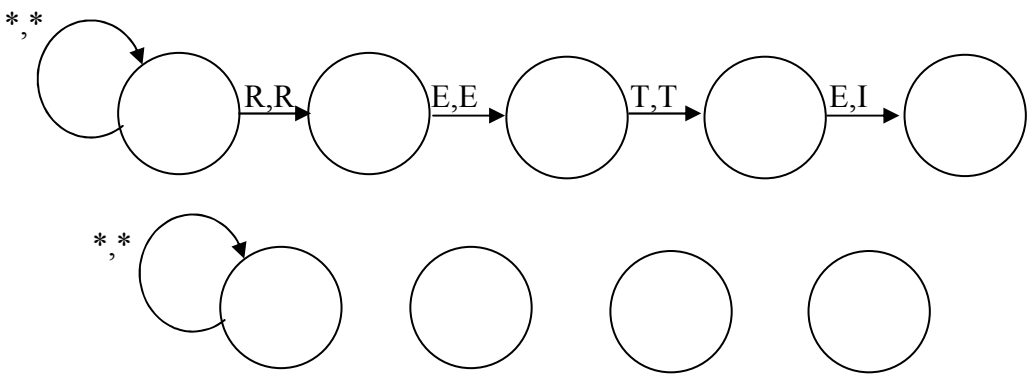
CD	11	10	01	00	U
M	M	M	N	N	1
N	M	P	N	N	1
P	P	P	Q	Q	0
Q	M	P	Q	Q	0

Quando si usa il modello dell'automa occorre tenere ben presente che, a meno di casi molto particolari, si vuole progettare **macchine che non si fermano mai**. Il comportamento a regime deve essere quindi sempre descritto da un grafo **strettamente connesso**, cioè deve esistere almeno una sequenza d'ingresso che consenta di passare da uno stato arbitrariamente scelto ad un altro stato arbitrariamente scelto: in fase di sintesi non si deve dunque mai prevedere né uno **stato iniziale**, cioè un nodo del grafo dotato di solo frecce uscenti, né uno **stato finale**, cioè un nodo dotato di sole frecce entranti.

Una seconda osservazione importante: la descrizione nei termini di un automa di un blocco che elabora sequenze di stringhe **non è unica**. In particolare, ma ce ne occuperemo più avanti, un certo comportamento può essere assegnato da grafi dotati di un differente numero di stati.

Esercitazione (da un compito d'esame)

Un editor riceve in ingresso, uno dopo l'altro, i caratteri di una stringa e li deve di norma riprodurre immediatamente in uscita; l'unica eccezione è l'occorrenza in ingresso della sequenza RETE, che deve essere sostituita in uscita dalla sequenza RETI. Ai fini dell'elaborazione richiesta si adotti, in ingresso ed in uscita, l'alfabeto {E,R,T,*}, in cui * rappresenta ogni carattere diverso da E, R, T. Individuare un grafo di Mealy a 4 stati ed uno a 5 stati.



Macchine sequenziali asincrone e sincrone

Notiamo innanzitutto che la macchina a stati finiti ha come caso particolare la *macchina combinatoria*: se S è vuoto, o se contiene un unico stato, la risposta ad ogni istante dipende infatti unicamente dallo stimolo di ingresso contemporaneo.

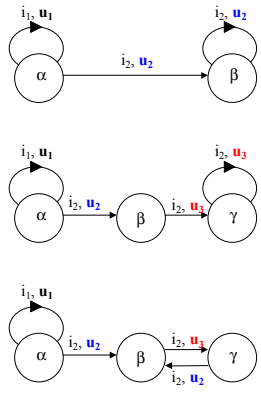
In generale S è tutt'altro che vuoto ed **una macchina combinatoria con una memoria in retroazione** è la struttura necessaria per elaborarlo. Per decidere come deve essere realizzata la retroazione è utile introdurre una classificazione dei comportamenti basata sul *numero di variazioni di uscita* che si debbono verificare *dopo una variazione di ingresso*.

Classificazione dei comportamenti

①: un simbolo in ingresso genera un solo simbolo in uscita
Evento: modifica dell'ingresso

②: un simbolo in ingresso genera una stringa finita di simboli in uscita
Eventi: modifica dell'ingresso e trascorrere del tempo

③: un simbolo in ingresso genera una stringa finita e ripetuta di simboli in uscita
Eventi: modifica dell'ingresso e trascorrere del tempo



Il comportamento di tipo ① considera come eventi le sole modifiche d'ingresso: ad ogni nuovo simbolo d'ingresso deve essere generato un solo nuovo simbolo d'uscita.

La descrizione di questo comportamento in termini di grafo degli stati evidenzia che la macchina si trova dapprima in una *situazione di stabilità* per lo stato e per l'uscita, poi esegue una transizione ed infine si riporta in una situazione di stabilità.

Macchine che devono passare da una ad un'altra condizione di stabilità sono dette **asincrone**, non avendo alcuna necessità di tenere conto della variabile *tempo*: la durata di ogni simbolo d'uscita è infatti stabilita dalla durata del simbolo d'ingresso che lo ha generato.

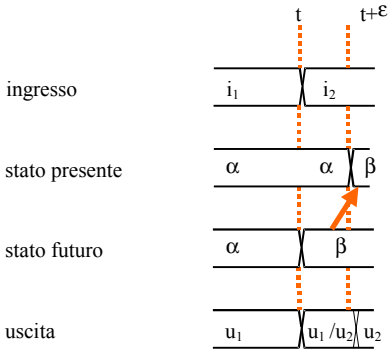
I comportamenti di tipo ② e ③ richiedono che la macchina modifichi più volte il suo stato per poter ottenere dalla F la desiderata sequenza di simboli di uscita, pur in presenza di uno stesso simbolo d'ingresso. E' inoltre necessario che la macchina sappia fare *misure di tempo* per attribuire ad ogni simbolo la *durata* appropriata. Ciò richiede che ogni transizione di stato si svolga in un **prefissato intervallo di tempo** (l'*unità di misura* del tempo) e che ogni simbolo d'uscita sia generato da una sequenza di stati *instabili* atta ad attribuirgli la durata prevista.

Macchine che hanno questi comportamenti sono dette **sincrone**.

3.2 – La macchina asincrona

Sia **t** il generico istante di tempo in cui si verifica il cambiamento da $\mathbf{i_1}$ a $\mathbf{i_2}$ del simbolo d'ingresso di una macchina asincrona; sia **α** lo stato stabile in presenza del quale si verifica questo evento e **β** il suo stato futuro per $\mathbf{i_2}$; siano infine $\mathbf{u_1}$ il simbolo d'uscita generato prima di **t** ed $\mathbf{u_2}$ quello che dovrà sostituirlo.

La macchina asincrona (comportamento)



La macchina tiene in memoria lo stato presente α per poter calcolare lo stato futuro β.
Una volta che il calcolo è stato fatto, β sostituisce α in memoria.
La memoria può dunque essere un "piccolo" ritardo.

Supponiamo ideali le macchine combinatorie preposte al calcolo delle funzioni F e G.

Prima di **t** stato e uscita sono costanti:

$$\mathbf{G(\alpha, i_1) = \alpha}$$
$$\mathbf{F(\alpha, i_1) = u_1}$$

All'istante **t** si ha:

$$\mathbf{G(\alpha, i_2) = \beta}$$
$$\mathbf{F(\alpha, i_2) = u_1} \textit{ oppure } \mathbf{u_2}$$

All'istante $\mathbf{t+\varepsilon}$ la macchina deve aggiornare il suo stato e la sua uscita e porsi in attesa di un nuovo evento:

$$\mathbf{G(\beta, i_2) = \beta}$$
$$\mathbf{F(\beta, i_2) = u_2}$$

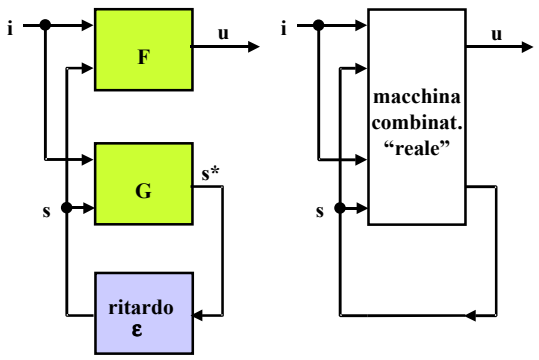
Per un tempo **ε** lo stato futuro è diverso dallo stato presente: nella macchina asincrona il ruolo di memoria è dunque affidabile ad un "ritardo" posto in retroazione alla funzione G.

Abbiamo però già osservato che una macchina combinatoria "reale" ha un ritardo nell'esecuzione dei suoi calcoli.

Per realizzare la macchina asincrona "reale" è dunque sufficiente una **retroazione diretta** tra l'uscita che calcola lo stato futuro e l'ingresso che esamina lo stato presente.

Una ovvia regola di corretto impiego è che il simbolo d'ingresso venga modificato solo quando la macchina è in uno stato stabile.

La macchina asincrona (struttura)



CASO DI STUDIO – L'analisi del **relè con autoritenuta** fatta a pag. 22 ci ha consentito di individuare uno schema formato sia da simboli logici disposti in serie/parallelo, sia da un ritardo disposto in retroazione.

Questo uso particolare di un relè realizza dunque una **macchina sequenziale asincrona**; la macchina ha **due stati**: il segnale **I** trasporta l'informazione di stato futuro ed il segnale **i** quella di stato presente

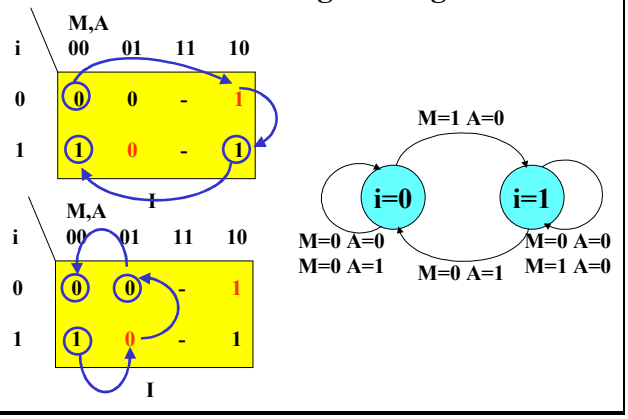
Ci siamo anche procurati una tabella di esperimenti e ci possiamo ora porre l'obiettivo di dedurre due descrizioni più astratte: la **tabella di flusso** ed il **grafo degli stati**.

La tabella di flusso avrà i valori di **i** come indicatori di riga ed i valori di **M**,**A** (i segnali che trasportano l'informazione d'ingresso) come indicatori di colonna; negli incroci dovranno essere apposti i valori di I. Per renderla più astratta in figura è stato usato l'alfabeto **{0,1}**: il simbolo **1** indica il valore **SI** per la corrente **I** e la situazione di contatto chiuso per **M**, **A** e **i**; il simbolo **0** indica il valore **NO** per **I** e la situazione di contatto aperto per **M**, **A** e **i**.

E' stata impiegata anche una **lineétta** (la cosa ci servirà più avanti) per indicare che non interessa il valore di **I** in corrispondenza della contemporanea pressione dei pulsanti **M**, **A**.

La tabella così ottenuta conferma che il comportamento è di tipo ①: l'uscita della funzione **G** (il segnale **I**) cambia di valore solo se cambiano gli ingressi; il cambiamento di I, a causa della retroazione, si propaga anche sul segnale **i** e poi si determina sempre una situazione di stabilità.

Relè con autoritenuta: tabella di flusso e grafo degli stati



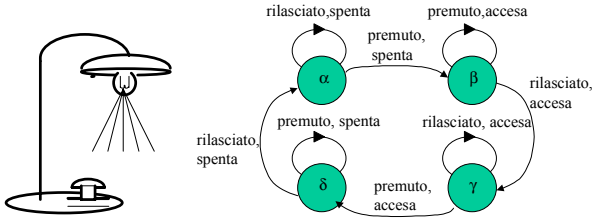
Il grafo evidenzia ancor meglio le condizioni in cui la macchina è stabile e quelle in cui compie una transizione di stato interno. Con i suoi due stati la macchina è dunque in grado di ricordare quale tra due eventi si è verificato per ultimo, cioè si comporta come una **memoria binaria** (più avanti le attribuiremo la denominazione di ***latch set/reset***).

CASO DI STUDIO – Il comportamento di una lampada da tavolo azionata da un pulsante è asincrono e richiede quattro stati.

Lo stato **α** riassume un passato in cui si è verificato un *numero pari* di pressioni sul pulsante a partire dalla situazione di lampada spenta. Finché il pulsante non verrà di nuovo azionato si deve avere stabilità dello stato e mantenimento dell'uscita. Quando il pulsante viene di nuovo premuto, il n° di azionamenti diventa *dispari* e la macchina riassume questo fatto entrando dapprima nello stato **β** e rimanendovi poi per tutto il tempo di pressione.

Il tempo di percorrenza di un ramo del grafo di una macchina asincrona è dunque pari al valore del ritardo con cui la sua parte combinatoria renderà disponibili i risultati dei suoi calcoli.

Un esempio di macchina asincrona: la lampada da tavolo



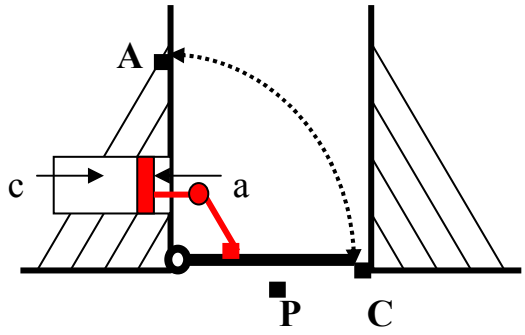
pulsante i ∈ I:{rilasciato,premuto}
lampadina u ∈ U:{spenta, accesa}

	rilasciato	premuto
α	α , spenta	β , spenta
β	γ , accesa	β , accesa
γ	γ , accesa	δ , accesa
δ	α , spenta	δ , spenta

Dato che questo tempo è comunque molto piccolo, non cambia nulla per l'utente della lampada se la variazione di uscita si ha quando α è ancora stato presente o solo quando lo diventa β. Il rilascio del pulsante fa compiere alla macchina un nuovo cambiamento di stato: γ ricorda che la lampada deve essere tenuta accesa anche se il pulsante non è più premuto. Con una nuova pressione si modifica sia lo stato (δ consente alla macchina di ricordare che il numero complessivo di comandi è ritornato *pari*), sia l'uscita. Non occorrono altri stati: quando il pulsante verrà di nuovo rilasciato, sarà sufficiente riportare la macchina nello stato da cui siamo partiti.

Esercitazione (da un compito d'esame)

Dovete automatizzare l'apertura/ chiusura di una porta sulla base delle seguenti specifiche:
" la porta deve essere di norma tenuta chiusa; il valore 1 di un sensore P, adibito a rilevare la presenza di una persona sulla soglia, è l'evento che determina o un movimento di apertura o il mantenimento di porta completamente aperta".



Un attuatore idraulico consente di muovere la porta: il comando a=1 la apre ed il comando c=1 la chiude. I due comandi devono essere attivi **solo** durante il movimento della porta ed **uno solo alla volta**. Il sensore A individua se la porta è (A=1) o non è (A=0) completamente aperta. Il sensore C individua se la porta è (C=1) o non è (C=0) completamente chiusa.

Decidete di realizzare il controllo della porta con una macchina combinatoria.

1. Quali sono i segnali d'ingresso? ……
2. Quali configurazioni d'ingresso sono "impossibili"? ……
3. Tabulare la relazione ingresso/uscita

0 0 0		
0 0 1		
0 1 1		
0 1 0		
1 0 0		
1 0 1		
1 1 1		
1 1 0		

4. Nel precedente progetto riscontrate un comportamento indesiderato ogniqualvolta la porta comincia ad aprirsi e la persona avanza subito per entrare. Cosa succede?

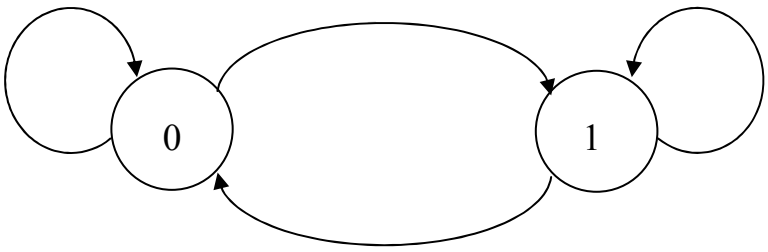
……..

5. Per eliminare il comportamento indesiderato è necessario aggiungere una frase alla specifica:

……

Per rispettarla è necessaria una macchina sequenziale asincrona a due stati.

6. Come è fatto il grafo degli stati?



7. Come è fatta la tabella di flusso?

	ingressi:							
stato:	000	001	011	010	100	101	111	110
0								
1								

3.3 - La macchina sincrona

Per ottenere una esatta misura del tempo da una macchina sincrona occorre accettare un vincolo: la *modifica* dei simboli d'ingresso, di stato e di uscita deve verificarsi solo in corrispondenza di istanti t_n , detti di *sincronismo*, che distano uno dall'altro di una quantità T_0 prefissata.

Il funzionamento della macchina viene così suddiviso in intervalli di *eguale durata*, durante ciascuno dei quali **ingresso, stato e uscita sono costanti**. Per esprimere questa situazione è utile apporre un *apice* ai simboli di ingresso, di stato e d'uscita ed utilizzarlo per indicare l'intervallo a cui si fa riferimento:

$$u^n = F(s^n, i^n)$$

$$s^{n+1} = s^{*n} = G(s^n, i^n)$$

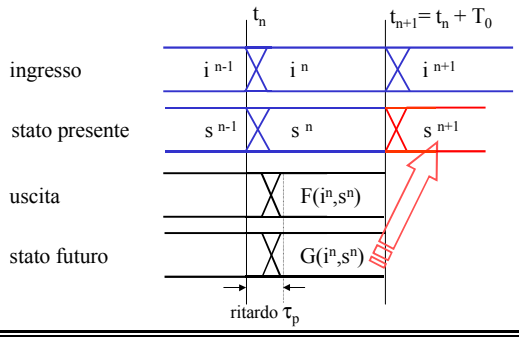
Alla fine di ogni intervallo nella memoria viene scritto il simbolo di stato futuro che la funzione G ha generato all'inizio dell'intervallo (o poco dopo, se è una macchina reale).Anche in questo caso il ruolo di memoria può essere affidato ad un **ritardo**.

A differenza della macchina asincrona, in cui il valore del ritardo può essere qualsiasi, la macchina sincrona ha però bisogno di un ritardo esattamente di valore T_0.

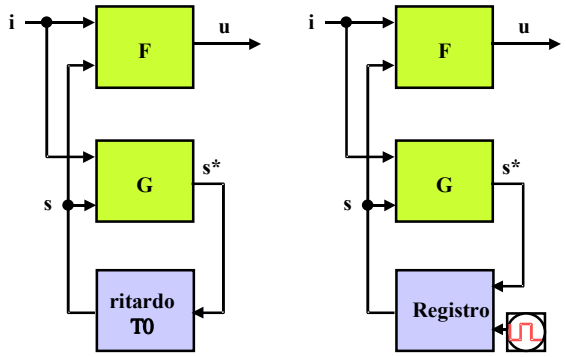
Tecnica usuale per realizzarlo è quella di predisporre all'esterno un *oscillatore a onda quadra* (il **clock)** con periodo T_0 ed impiegare poi o l'uno o l'altro dei due *fronti* del suo segnale d'uscita per indicare ad una particolare macchina asincrona, detta **registro**, gli istanti in cui deve sostituire lo stato presente con lo stato futuro.

La macchina sincrona (comportamento)

τ_p : intervallo di tempo impiegato dal calcolo di F e di G
T_0 : intervallo di tempo in cui la macchina non modifica il suo stato



La macchina sincrona (struttura)



Flip-flop e Registro

L'uscita della macchina asincrona da porre in retroazione alla funzione G di una macchina sincrona deve avere la proprietà di modificarsi solo in corrispondenza degli istanti di sincronismo.

Il caso più semplice è la retroazione di un bit.

CASO DI STUDIO – A pag. 42 abbiamo studiato una macchina asincrona in grado di presentare in uscita i valori assunti da un segnale **D** in corrispondenza dei fronti di salita di un segnale **C**.

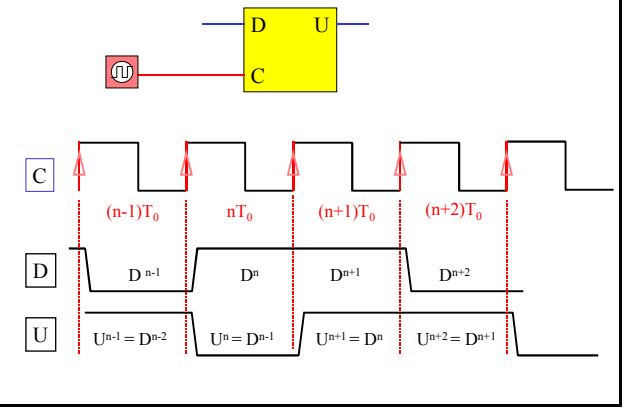
Facciamo ora le seguenti ipotesi:

- il segnale **C** è periodico con periodo T_0 ,
- le modifiche di valore del segnale **D** possono verificarsi solo poco dopo il verificarsi su **C** di un fronte di salita.

La conseguenza è evidenziata in figura: il segnale **U** è una copia di **D** ritardata di T_0.

L'ipotesi che le variazioni di D si verifichino con un piccolo ritardo rispetto al fronte di C non dve preoccupare: il ritardo con cui la macchina reale **G** mette a disposizione i bit di stato futuro garantisce la loro costanza in corrispondenza degli istanti di sincronismo.

Il flip-flop (o registro da un bit)



Componenti primitivi di questo tipo (che d'ora in poi chiameremo **flip-flop**) risolvono dunque il problema della retroazione in qualsiasi macchina sincrona.

ESEMPIO – L'addizionatore di numeri binari trasmessi in serie (v. pag.40) è una macchina sincrona. La memorizzazione del bit di riporto deve essere affidata ad un flip-flop.

Se si ignora la presenza del clock e tutto il conseguente meccanismo interno dei cambiamenti di stato, la descrizione del comportamento di un flip-flop richiede di mettere in gioco un ingresso binario D, un'uscita binaria U ed una sola variabile di stato Q:

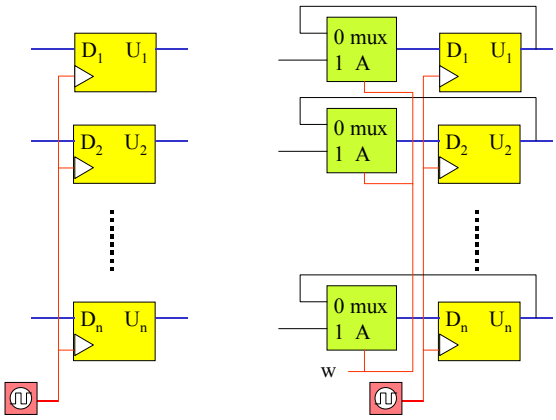
$$U^n = Q^n$$
$$Q^{n+1} = D^n$$

Il flip-flop può dunque essere considerato la macchina sincrona **elementare**.

La memorizzazione di uno stato interno di **n** bit richiede la disposizione in parallelo di **n** flip-flop. Il simbolo logico del flip-flop indica con un "triangolino" l'ingresso del clock e l'azionamento da parte di un fronte di salita; quando è usato il fronte di discesa, il triangolino è preceduto da un "pallino".

In molte applicazioni è necessario che il registro accetti un nuovo dato solo quando un comando sincrono **w** (comando di *write*) assume il valore 1. A tal fine ogni flip-flop deve essere preceduto da un selettore a due vie: quando w = 0, si ha $D_i = U_i$ ed all'arrivo del fronte del clock viene scritto nel registro il dato che conteneva già; quando w = 1 viene invece memorizzato il dato proveniente dall'esterno.

Il registro da n bit



Contatore

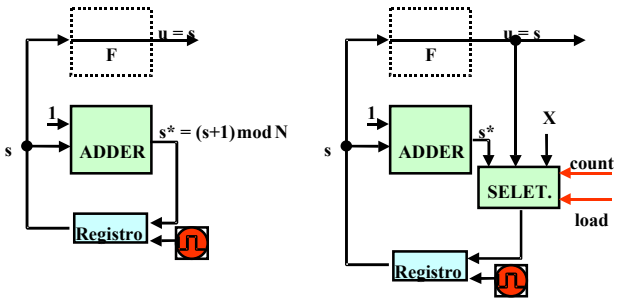
Il **contatore** è una macchina sequenziale sincrona tipicamente impiegata per "numerare" eventi. Gli stati interni sono gli interi 0, 1, 2, N-1 ed il grafo degli stati è un anello che li contiene tutti, uno dopo l'altro.

Supponiamo per ora che il contatore debba numerare solo intervalli di tempo consecutivi, quali risultano scanditi dal clock che lo aziona. In ogni intervallo elementare deve avvenire una transizione: detto **s** un generico stato presente, il suo stato futuro è espresso da (**s**+1)mod N.

In ogni intervallo l'uscita coincide con lo stato presente.

La realizzazione di questa macchina richiede soltanto un ADDER ed un REGISTRO che lo retroaziona.

Schema a blocchi di un contatore



Per attribuire al contatore maggiore flessibilità d'impiego (numerazione di eventi che possono anche non verificarsi in ogni intervallo) è utile aggiungere nella struttura un SELETTORE *a tre vie* e due comandi (*count* e *load*) per scegliere di volta in volta quale via abilitare. La tabella a lato evidenzia le tre differenti modalità di comportamento ottenibili con i due comandi.

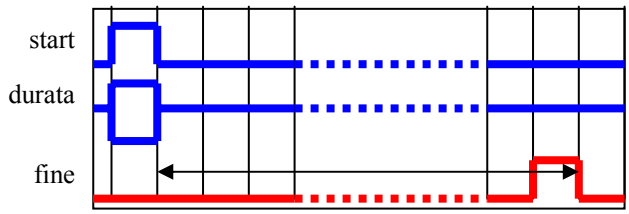
count	load	stato futuro
0	0	$s^{n+1} = s^n$,
1	0	$s^{n+1} = (s+1)^n \bmod N$
-	1	$s^{n+1} = X^n$

Studio di casi

La capacità della macchina sequenziale sincrona di interpretare e/o generare stringhe di simboli in cui l'informazione è anche associata alla **durata** di ciascun simbolo consente di affrontare e risolvere qualsivoglia problema di tipo 2 e di tipo 3.

1. Il ***timer*** o ***watch-dog*** è un "server" hardware di un calcolatore elettronico.

Questa macchina sincrona riceve ogni tanto dal processore un comando di *start* e la *durata* di un intervallo di tempo da misurare; al termine della misura il servitore genera, per un tempo T_0, una segnalazione *di fine intervallo* e si mette in attesa di un nuovo comando. Il comportamento è quindi di tipo ②.



Per semplificare il progetto supponiamo che il comando di attivazione del timer sia *start*=1, che duri un solo intervallo elementare e che si possa ripresentare solo dopo l'intervallo in cui *fine* ha assunto valore 1.

Supponiamo inoltre che la *durata* D dell'intervallo da misurare sia comunicata alla macchina quando *start*=1 e valga al più 4 T_0; quando *start*=0 il valore sull'ingresso D non è significativo

Il grafo degli stati è indicato in figura.

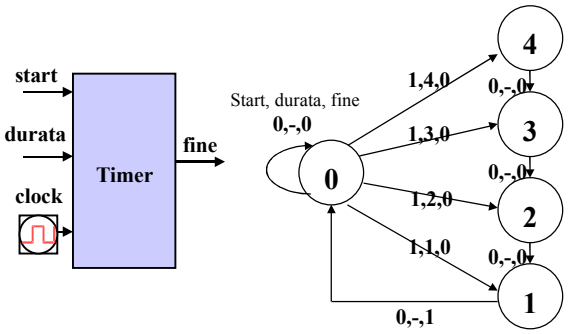
Lo stato 0 è stabile per *start* = 0, per consentire alla macchina di rimanere in attesa di un nuovo comando.

Quando start assume il valore 1, lo stato 0 è sostituito o da 4, o da 3, o da 2, o da 1 a seconda del valore di durata presente in ingresso in quell'intervallo.

Da quel momento in poi la macchina conta "all'indietro", generando fine=1 solo durante la transizione dallo stato 1 allo stato 0.

Si noti l'uso della lineétta sulle frecce uscenti dai nodi 4, 3, 2 e 1, per ricordare che nei corrispondenti intervalli la durata può avere un valore qualsiasi.

Una macchina sincrona di tipo 2: il timer



La macchina usa il periodo del clock come "unità di misura" del tempo.

2. Il **semaforo stradale** in funzionamento diurno ha un comportamento di tipo ③.

Il sistema è autonomo, cioè l'alfabeto d'ingresso è vuoto.

Consideriamo per semplicità solo le uscite necessarie per una sola direttrice di marcia.

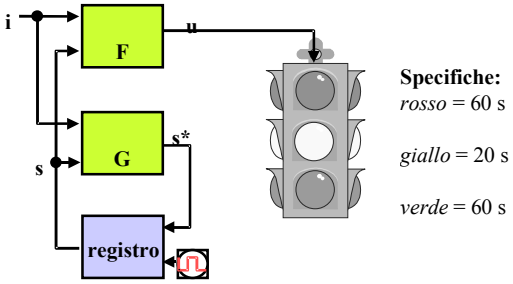
Con le specifiche indicate in figura il periodo del comportamento è di 140 secondi. Una volta assunto un clock con periodo di 20 s, il comportamento ha periodo 7 T_0 ed è formato da 7 intervalli elementari.

Il grafo, necessariamente di Moore, richiede dunque una successione di sette stati, in ciascuno dei quali la macchina permane per un tempo T_0.

Nella prima colonna della tabella di flusso sono elencati i possibili stati presenti. I corrispondenti valori della funzione G sono indicati nella seconda colonna.

La funzione F ha bisogno di tre colonne per indicare quale deve essere la situazione delle tre lampade in corrispondenza di ogni stato presente.

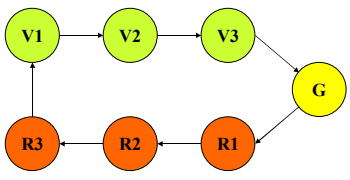
Una macchina sincrona di tipo 3: il semaforo



Specifiche:
rosso = 60 s
giallo = 20 s
verde = 60 s

IPOTESI - La macchina riceve un impulso di clock ogni 20 s, evento in corrispondenza del quale provvede ad aggiornare il suo stato interno

Il grafo degli stati



La tabella di flusso

stato presente	stato futuro	lampada verde	lampada giallo	lampada rosso
V1	V2	accesa	spenta	spenta
V2	V3	accesa	spenta	spenta
V3	G	accesa	spenta	spenta
G	R1	spenta	accesa	spenta
R1	R2	spenta	spenta	accesa
R2	R3	spenta	spenta	accesa
R3	V1	spenta	spenta	accesa

Una volta individuata la tabella di flusso, il procedimento di sintesi richiede che *stato interno* ed *uscita* vengano rappresentati con stringhe binarie.

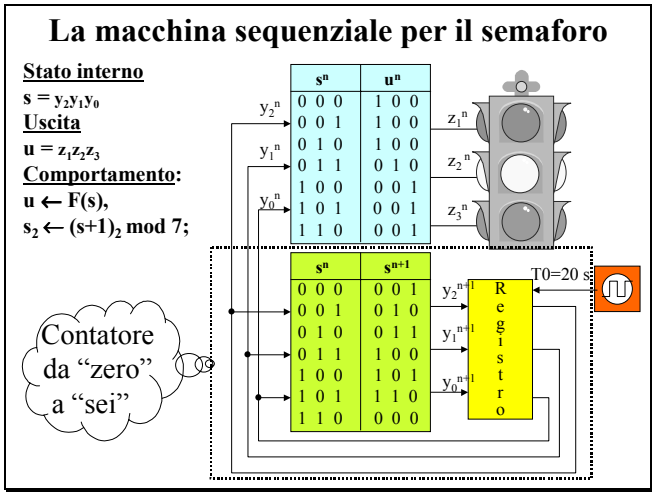
Si ottiene così la **tabella delle transizioni**.

In figura si è deciso di codificare lo stato con un codice non ridondante (sette delle otto possibili configurazioni di 3 bit ed in particolare i primi sette interi del sistema di numerazione in base 2); l'uscita è già di fatto codificata con il codice ridondante "1 su 3".

I tre bit di codifica dello stato presente sono memorizzati da un **registro**.

La funzione di aggiornamento G richiede che il numero binario corrispondente allo stato presente sia incrementato di una unità ad ogni comando del clock: il comportamento complessivo è dunque quello di un **contatore** che conta modulo 7.

La funzione di uscita F trascodifica i numeri binari dello stato interno in configurazoni del codice "1 su 3" e consente di tenere accesa dapprima la lampada verde per tre intervalli consecutivi, poi la lampada gialla per un intervallo ed infine la lampada rossa nei successivi tre intervalli.



3. Una **lavapiatti**, o meglio la macchina che ne controlla il funzionamento, ha un comportamento di tipo ②.

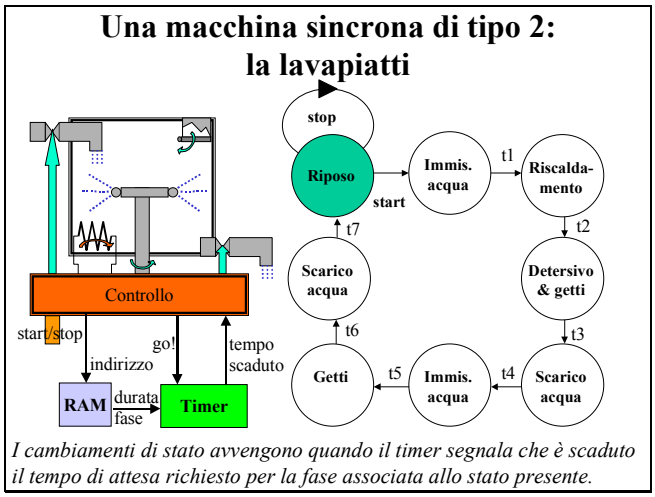
In figura è mostrato uno schema di principio per evidenziare l'esistenza di una parte Controllo e di una parte Percorso dei dati.

Il Controllo ha tanti stati quante sono le "fasi" del processo, cioè le differenti situazioni di comando da inviare agli *attuatori* (valvole per l'ingresso e l'uscita dell'acqua, resistenza per il riscaldamento dell'acqua, motore che ruota la girandola per i getti, levetta che apre lo sportello del detersivo).

La misura della durata di ciascuna fase è delegata al Percorso dati, ove si trovano un *timer* ed una *memoria* che contiene tutti i tempi da misurare.

Fino a quando il pulsante di start non viene premuto, il Controllo resta nello stato di "riposo" e mantiene in uno stato analogo il timer.

Quando il pulsante viene premuto, il Controllo entra nello stato di "immissione acqua" che determina la chiusura della valvola di uscita e l'apertura di quella di ingresso; durante la transizione il controllo estrae dalla memoria la durata di questa prima fase e la presenta in ingresso al timer con un apposito comando di start. Da quel momento in poi è la segnalazione di fine intervallo l'evento che determina il cambiamento di fase, la lettura della sua durata ed il riavvio del timer. Il solo stato finale di "scarico acqua" interpreta questa segnalazione come comando di stop e riporta la macchina in condizioni di riposo. Con un ulteriore comando manuale, preso in considerazione allo start, è possibile far fare alla macchina o un "prelavaggio" o un "lavaggio completo".



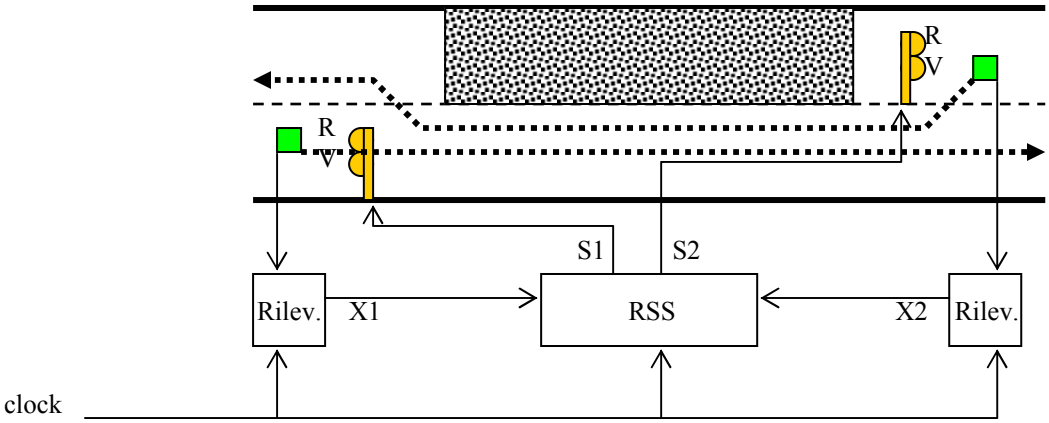
I cambiamenti di stato avvengono quando il timer segnala che è scaduto il tempo di attesa richiesto per la fase associata allo stato presente.

Esercitazione - Modificare la tabella di flusso del semaforo per poter gestire due direttrici di marcia e per attribuire a ciascuna gli stessi tempi di "verde", di "giallo" e di "rosso".

Esercitazione (da un compito d'esame)

Una carreggiata di una strada a due direzioni di marcia è interrotta per lavori in corso ed è quindi necessario utilizzare l'altra a senso alternato. A tal fine vengono installati (vedi figura):

- due semafori, ciascuno con due lampade V,R ed un segnale di comando S (1 verde, 0 rosso);
- due rilevatori di "presenza macchina" che forniscono due segnali sincroni X1 e X2 (il valore 1 indica che nel precedente intervallo T_0 il rilevatore ha "visto" almeno una macchina);
- una RSS azionata dallo stesso clock dei rilevatori ed incaricata di generare i valori di S1, S2 esaminando i valori assunti da X1, X2 in ogni intervallo elementare di funzionamento.

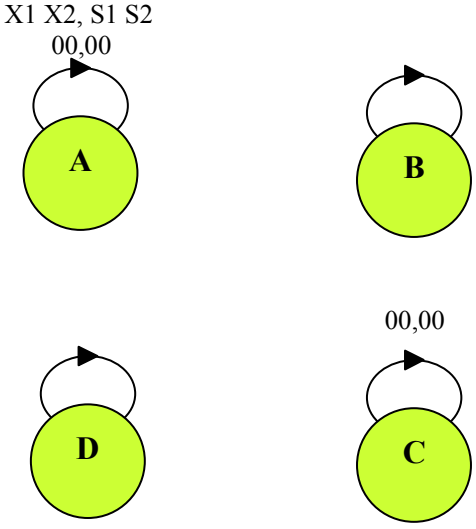


Comportamento di RSS – In assenza di macchine entrambi i semafori devono avere il "rosso" acceso. Se nell'intervallo successivo c'è un solo sensore a 1, il "verde" deve apparire solo sul semaforo davanti a cui è ferma la macchina; se le macchine in attesa sono due, il "verde" deve essere dato alla direzione di marcia che da più tempo non lo ha avuto.

Una situazione di "verde" deve essere mantenuta negli intervalli successivi solo se c'è un'altra macchina che vuole procedere nello stesso senso ed a patto che non ce ne sia una che vuole procedere in senso opposto; in quest'ultimo caso deve venire dato il "rosso" su entrambe le direzioni per un tempo sufficiente a garantire che anche l'ultima macchina entrata nel tratto a senso unico possa uscirne senza incrociare traffico nella direzione opposta.

1 – Il tratto a senso unico è lungo 500 metri e la prescritta velocità di percorrenza è di 30 Km/ora. Quanti minuti deve durare il "rosso" su entrambe le direzioni per garantire lo smaltimento del traffico in corso sul tratto a senso alternato?_____

2 –Assumendo tale durata come periodo del clock, tracciare il grafo degli stati e la tabella di flusso.



stato^n	(X1X2)^n 00	01	10	11
A				
B				
C				
D				

stato^{n+1} , uscita^n