Note introduttive sulle macchine a stati finiti sincrone

Cos'è una macchina a stati finiti sincrona

Una macchina a stati finiti (MSF) sincrona è una struttura conveniente per realizzare una funzione logica sequenziale in cui le uscite (e i segnali interni) vengono aggiornate in sincronia con le transizioni di un opportuno segnale periodico (clock). In generale, una MSF sincrona può essere rappresentata con uno schema a blocchi simile a quello riportato in fig. 1.

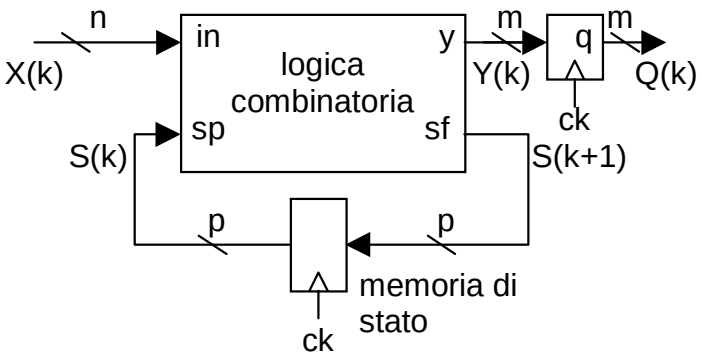

Fig. 1 Schema a blocchi di una generica macchina a stati finiti sincrona

La memoria di stato è un registro realizzato con p flip-flop *edge triggered* che memorizzano le variabili interne (lo stato presente sp) del circuito sequenziale. Ad ogni transizione attiva del clock ck (nella figura, la transizione LH), il contenuto della memoria di stato (cioè, lo stato presente) viene aggiornato e passa da S(k) a S(k+1) e nel registro di uscita q viene memorizzato Q(k)=Y(k), dove l'indice k numera progressivamente i periodi del segnale di clock. Sia il valore S(k+1) dello stato futuro che il valore Y(k) dell'uscita asincrona sono funzioni combinatorie del valore X(k) dell'ingresso e del valore S(k) dello stato presente:

$$\begin{cases} S(k+1) = F[S(k), X(k)] \\ Y(k) \quad\; = G[S(k), X(k)] \end{cases}$$

Il registro di uscita q sincronizza il cambiamento di valore dei segnali di uscita con la transizione del clock.

Le MSFvengono normalmente distinte in macchine di Mealy, in cui il valore delle uscite Y(k) dipende sia dal valore dello stato presente S(k) che dal valore degli ingressi X(k), e macchine di Moore, in cui Y(k) è solo funzione del solo stato presente S(k). Per esplicitare la dipendenza o meno di Y(k) da X(k), il diagramma della fig. 1 viene spesso rappresentato in un modo alternativo, proposto in fig. 2. Nel caso di una macchina di Moore, la dipendenza del blocco **G** da X(k) scompare.

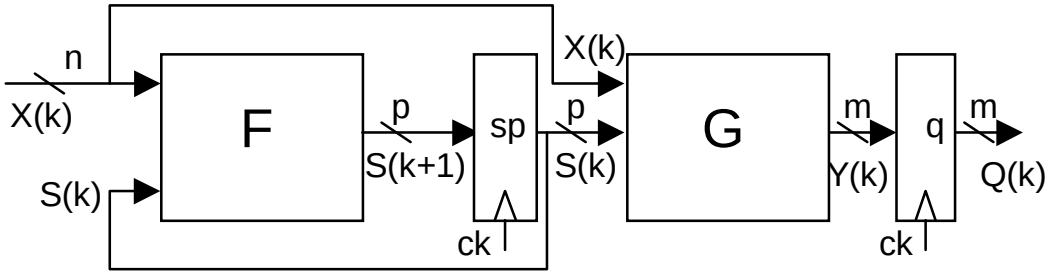

Fig. 2 Schema a blocchi di una generica MSF di Mealy. Per passare alla MSF di Moore basta eliminare l'ingresso X(k) dal blocco G.

Modi di rappresentare una MSF sincrona

Ci sono vari modi per rappresentare una specifica MSF sincrona, tra cui risultano essere particolarmente interessanti il *diagramma a stati* e la *tabella di transizione degli stati e delle uscite*. Pur essendo le due rappresentazioni in questione assolutamente equivalenti, il diagramma a stati è più conveniente nella fase di traduzione delle specifiche sulla funzione che la MSF deve svolgere, mentre la tabella di transizione è utile nella fase di sintesi (manuale) della rete logica che implementa la MSF.

Supponiamo ad esempio di dover realizzare una MSF che prende in ingresso un segnale x a un bit, e quando riconosce la sequenza "1101" pone l'uscita y a "1". Questa descrizione verbale è il punto di partenza del progetto della MSF. Il primo passo è la costruzione del diagramma degli stati a partire dalla descrizione precedente. Il diagramma è un grafo orientato i cui vertici sono gli stati che può assumere la MSF e i rami rappresentano le transizioni tra gli stati. Nel caso del nostro esempio, possiamo dire che la MSF deve spostarsi in un nuovo stato ogni volta che riconosce un ingresso appartenente alla sequenza "1101", mentre deve arretrare allo stato iniziale o ad uno stato intermedio (a seconda dei casi) se l'ingresso non appartiene alla sequenza data. Questo porta al diagramma rappresentato nella fig. 3, in cui A, B, C e D sono i simboli associati ai quattro stati che la MSF può assumere, mentre le coppie x/y associate a ciascun ramo rappresentano il valore assunto dal bit di ingresso x e dal bit di uscita y in corrispondenza della transizione.

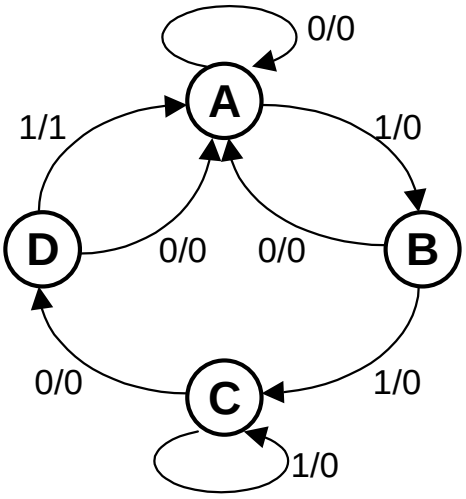

Fig. 3 Diagramma degli stati della MSF che riconosce la sequenza "1101"

Dal diagramma si evince che la MSF inquestione è una macchina di Mealy, in quanto nelle transizioni che portano la MSF dallo stato D allo stato A il valore del bit di uscita y dipende anche dal valore del bit di ingresso x.

Il passo successivo consiste nel ricavare dal diagramma la tabella di transizione degli stati e delle uscite, che consente di rappresentare in forma di tabella di verità le funzioni **F** e **G** che calcolano lo stato futuro S(k+1) e le uscite Y(k) a partire dallo stato presente S(k) e dagli ingressi X(k). Nel nostro caso, la tabella può essere rappresentata nella forma seguente:

Tab. 1 Tabella di transizione degli stati e delle uscite

S(k)	S(k+1)		Y(k)	
	X(k) = 0	X(k) = 1	X(k) = 0	X(k) = 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	A	0	1

In alternativa, la variabile X(k) può essere riportata in una colonna adiacente a S(k), il che però richiede una tabella con 8 righe.
Per poter procedere alla sintesi logica delle funzioni **F** e **G** bisogna assegnare una codifica binaria agli stati. A questo scopo non esiste una procedura univoca: la scelta della codifica può influenzare, a seconda dei casi, la complessità della logica dello stato futuro e delle uscite, il numero di bit che cambia valore ad ogni transizione, il consumo di potenza, e altri parametri del circuito. Le due codifiche più popolari sono la codifica sequenziale e la *codifica one-hot*.
La codifica sequenziale consiste semplicemente nell'assegnare a ciascuno stato un numero binario crescente, partendo da 0. Il numero di bit richiesti per codificare tutti gli N stati della MSF (quindi, il numero di FF nella memoria di stato) è pari al primo intero maggiore o uguale a $\log_2 N$. Il vantaggio pricipale di questo tipo di codifica è che minimizza le dimensioni della memoria di stato.
La codifica *one-hot* invece utilizza una memoria con un numero di FF pari al numero di stati della macchina, assegnando a ciascun FF uno specifico stato. Di conseguenza, ad ogni istante il contenuto della memoria di stato è un codice in cui N-1 bit sono a "0" mentre uno solo (quello corrispondente allo stato in cui si trova la MSF in quell'istante) è a "1". Rispetto alla codifica sequenziale, la *one-hot* utilizza un umero di FF maggiore, ma può semplificare notevolmente la logica dello stato futuro e delle uscite. La tabella sottostante mette a confronto il risultato delle due codifiche applicate alla MSF della fig. 3:

Tab. 2 Codifica sequenziale e codifica one-hot

stato	sequenziale	one-hot
A	00	0001
B	01	0010
C	10	0100
D	11	1000

Una volta sostituita nella tab. 1 la codifica prescelta, si può procedere alla sintesi logica delle funzioni **F** e **G**. Scegliendo la codifica sequenziale si ottiene:

$$\begin{cases} S_1(k+1) = \overline{x} \cdot S_1(k) \cdot \overline{S}_0(k) + x \cdot S_1(k) \cdot \overline{S}_0(k) + x \cdot \overline{S}_1(k) \cdot S_0(k) \\ S_0(k+1) = \overline{x} \cdot S_1(k) \cdot \overline{S}_0(k) + x \cdot \overline{S}_1(k) \cdot \overline{S}_0(k) \\ y \quad\quad\quad = x \cdot S_1(k) \cdot S_0(k) \end{cases}$$

dove i pedici delle variabili di stato individuano ciascuno dei due bit necessari per la codifica sequenziale degli stati della MSF in oggetto. Dalle equazioni booleane dello stato futuro e dell'uscita è possibile ricavare la rete logica che realizza le corrispondenti funzioni **F** e **G**: in questo caso sono sufficienti 5 AND a 3 ingressi (nota che un termine è comune a due equazioni), una OR a 3 ingressi, una OR a 2 ingressi e tre FF di tipo D *edge-triggered* (due per la memoria di stato, uno per il registro di uscita). Le AND e le OR in cascata possono essere trasformate in NAND (applicando il teorema di de Morgan) per una realizzazione più efficiente in tecnologia CMOS. Più comunemente, la logica combinatoria di una MSF integrata in tecnologia CMOS viene però realizzata usando una struttura di tipo PLA (vedi cap. 10 del testo di Rabaey).

Modello VHDL di macchine a stati finiti sincrone

Ci sono diversi approcci efficaci per realizzare un modello VHDL di una macchina a stati finiti. Il più semplice, e anche il più vicino alla rappresentazione della generica MSF in Fig. 2 consiste nell'utilizzo di due processi, uno che modella la parte combinatoria della macchina, il secondo che modella l'aggiornamento dei registri in sincronia con il segnale di clock. Il processo associato alla parte combinatoria deve essere sensibile agli ingressi X(k) della macchina e allo stato presente S(k), mentre il processo associato ai registri deve essere sensibile unicamente al segnale di clock.

Di seguito è riportata una possibile realizzazione del modello VHDL della macchina definita nel diagramma degli stati di Fig. 3. Da notare che, nel modello VHDL, non viene fatta una scelta esplicita della codifica degli stati, che vengono rappresentati con un dato di tipo enumerazione.

```
-- Modello di macchina a stati finiti che riconosce la sequenza di
bit "1101".

-- librerie con i tipi di dati e le funzioni di I/O
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_textio.all;

library std;
use std.textio.all;

use work.all;

entity msf1 is
  port(msf_in, ck: in std_logic;
       msf_out: out std_logic);
end msf1;

architecture rtl of msf1 is
  -- parte dichiarativa: "Stato" e` un tipo di dato che puo`
  -- assumere solo i valori A, B, C, D. I due segnali stato_pres
  -- e stato_fut sono dichiarati di questo tipo
  type Stato is (A, B, C, D);
  signal stato_pres, stato_fut: Stato := A;
  signal y: std_logic;
begin
  -- Il processo P1 modella le funzioni F e G della macchina,
  -- quindi si attiva quando c'e` un evento su stato_pres o
  -- sul segnale di ingresso msf_in
  P1: process(stato_pres, msf_in)
  begin
    -- la tabella di verita` delle funzioni F e G e` descritta con
    -- istruzioni sequenziali "case" e "if then else"; prova a
    -- riscriverla usando solo istruzioni "case" o solo istruzioni
    -- "if then else" annidate
    case stato_pres is
      when A =>
        if msf_in = '0' then
          stato_fut <= A;
          y <= '0';
        else
          stato_fut <= B;
          y <= '0';
        end if;
      when B =>
        if msf_in = '0' then
          stato_fut <= A;
          y <= '0';
```

```
        else
          stato_fut <= C;
          y <= '0';
        end if;
      when C =>
        if msf_in = '0' then
          stato_fut <= D;
          y <= '0';
        else
          stato_fut <= C;
          y <= '0';
        end if;
      when D =>
        if msf_in = '0' then
          stato_fut <= A;
          y <= '0';
        else
          stato_fut <= A;
          y <= '1';
        end if;
    end case;
  end process P1;

  -- Il processo P2 modella i registri della macchina, quindi
  -- si attiva solo quando c'e` un evento sul segnale ck
  P2: process(ck)
  begin
    -- Inoltre, controlla se l'evento corrisponde ad una
    -- transizione LH o HL (e in quest'ultimo caso non fa
    -- niente). L'espressione "ck'event" in questo contesto
    -- e` superflua perche` se il processo P2 si attiva e`
    -- proprio perche' c'e` stato un evento sul segnale ck
    if ck'event and ck = '1' then
      stato_pres <= stato_fut;
      msf_out <= y;
    end if;
  end process P2;

end architecture rtl;
```