Logica e Reti Logiche
(Episodio 15: Macchine a stati finiti)

Francesco Pasquale

4 dicembre 2023

Dato un circuito *combinatorio* possiamo sempre scrivere una formula della logica proposizionale che corrisponde a quel circuito, e quindi la sua tabella di verità. Nello scorso episodio abbiamo introdotto dei blocchetti funzionali, i *Flip-Flop*, che sono in grado di *memorizzare* un bit. Se abbiamo un circuito che contiene dei *Flip-Flop* e dei cicli possiamo ancora descriverlo utilizzando formule e tabelle? Vediamo.

1 Analisi di circuiti sequenziali

Consideriamo, per esempio, il circuito in Figura 1

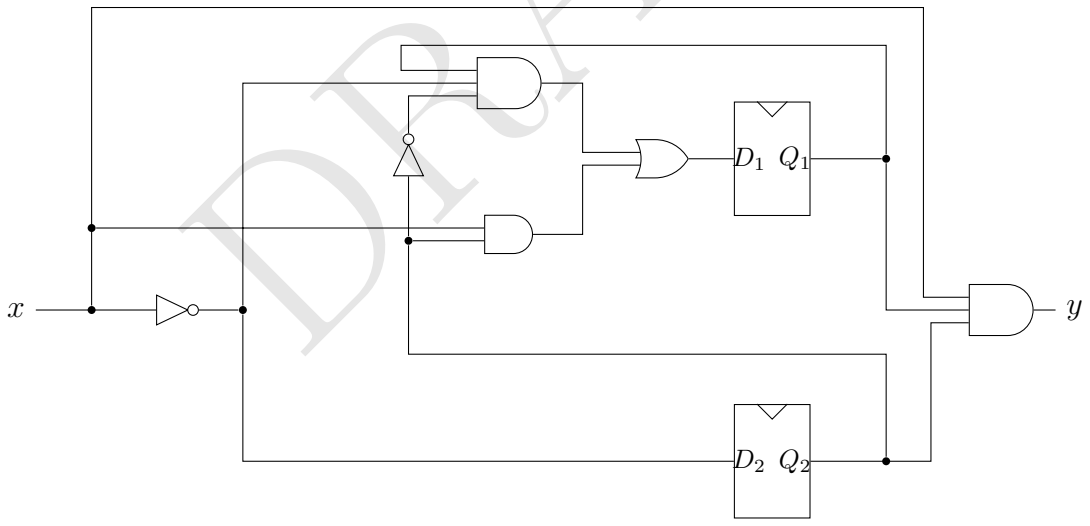


Figura 1: Esempio di circuito sequenziale

Il circuito contiene alcune porte logiche elementari e un registro formato da due Flip-Flop. Ognuno dei due Flip-Flop quindi ha uno *stato attuale* (ossia il bit che sta memorizzando), dato dalla variabile indicata con Q nel disegno, e uno *stato futuro* (ossia il bit che si troverà a memorizzare nel momento in cui il clock passerà dallo stato 0 allo stato 1) dato dalla variabile indicata con D.

Dato lo stato attuale del circuito (la coppia di bit Q_1, Q_2) e l'input che gli passiamo in questo momento (il bit x), risultano univocamente determinati sia l'output attuale del

circuito (il bit y) sia lo stato futuro del circuito (la coppia di bit D_1 e D_2). Quindi il circuito può essere descritto da un sistema di tre equazioni in cui Q_1, Q_2 e x costituiscono le variabili indipendenti e D_1, D_2 e y quelle dipendenti.

Esercizio 1. Verificare che i valori di D_1, D_2 e y del circuito in Figura 1 sono dati dalle formule seguenti

$$\begin{cases} D_1 &= Q_1\overline{Q}_2\overline{x} + Q_2 x \\ D_2 &= \overline{x} \\ y &= Q_1 Q_2 x \end{cases}$$

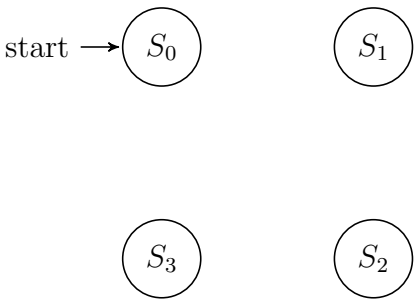
Dalle equazioni dell'esercizio precedente possiamo anche ricavare le tabelle di verità di D_1, D_2 e y. Chiamiamo *tabella di stato* la tabella che le contiene tutte

Q_1	Q_2	x	D_1	D_2	y
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	1	0	0	1	0
1	1	1	1	0	1

Figura 2: Tabella di stato del circuito in Fig. 1

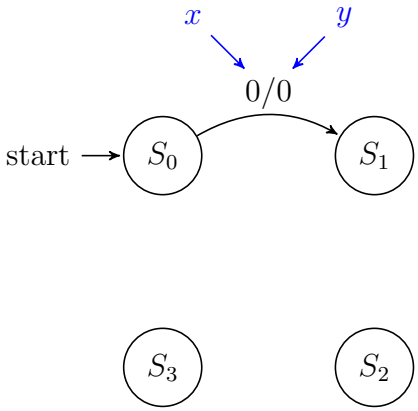
Oltre che con le *equazioni di stato* e la *tabella di stato* c'è anche un altro modo conveniente in cui possiamo rappresentare un circuito sequenziale: con un *grafo diretto* con gli archi *etichettati*. Usiamo i nodi del grafo per rappresentare gli stati del circuito, gli archi per le transizioni da uno stato all'altro e le etichette sugli archi per indicare gli input che determinano le transizioni e gli output del circuito. Il grafo che si ottiene si chiama *diagramma di stato* del circuito.

Per esempio, nel caso del nostro circuito abbiamo quattro possibili stati (i valori che la coppia di bit (Q_1, Q_2) può assumere). Indichiamoli con $S_0 = (0,0)$, $S_1 = (0,1)$, $S_2 = (1,0)$ e $S_3 = (1,1)$, e assumiamo che inizialmente il circuito si trovi nello stato $(Q_1, Q_2) = (0,0)$ (osservate che possiamo sempre farlo partire da quello stato se i nostri Flip-Flop sono *resettabili*)

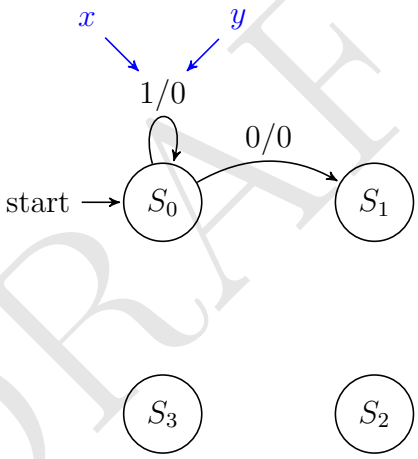


Cosa succede quando il clock del nostro circuito comincia a ticchettare e passa per la prima volta da 0 a 1? Beh, dipende da qual è l'input x del circuito: in accordo alla tabella in Figura 2, quando il circuito è nello stato $(Q_1, Q_2) = (0,0)$ abbiamo che

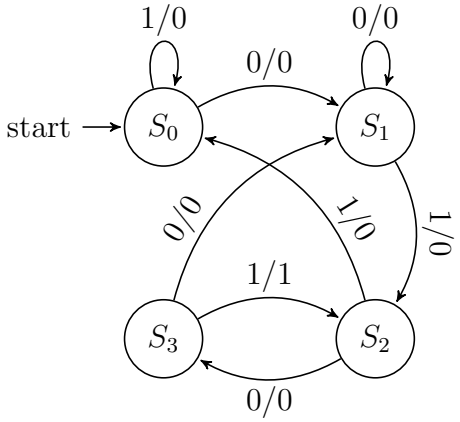
- se l'input è $x = 0$, allora l'output è $y = 0$, inoltre quando il clock passa da 0 a 1 lo stato del primo Flip-Flop resterà 0 mentre quello del secondo Flip-Flop andrà a 1, $((D_1, D_2) = (0, 1))$, quindi al ciclo di clock successivo il circuito sarà nello stato S_1



- se invece l'input è $x = 1$, l'output sarà ancora $y = 0$, e quando il clock passa da 0 a 1 lo stato di entrambi i Flip-Flop resterà a 0, $((D_1, D2) = (0, 0))$, quindi al ciclo di clock successivo il circuito sarà ancora nello stato S_0



Esercizio 2. Verificare che il diagramma di stato completo che si ottiene dalla tabella in Fig. 2 è



Adesso tocca a voi fare un po' di allenamento: se avete capito il contenuto di questa sezione dovrebbe essere facile svolgere il prossimo esercizio.

Esercizio 3. Un circuito sequenziale con un ingresso x, un'uscita y, e due D-Flip-Flop è descritto dalle seguenti equazioni:

$$D_1 = Q_1\overline{Q_2} + x \qquad D_2 = (Q_1 \oplus Q_2)\overline{x} \qquad y = Q_1 Q_2 x$$

dove Q_i e D_i indicano rispettivamente lo stato corrente e lo stato futuro dell'i-esimo Flip-Flop, per $i = 1, 2$.

1. Derivare la tabella di stato e il diagramma di stato;
2. Disegnare il circuito.

2 Progetto di circuiti sequenziali

Nella sezione precedente siamo partiti da un circuito e ne abbiamo "analizzato" il funzionamento, descrivendolo prima con delle equazioni, poi con una tabella, infine con un diagramma.

In questa sezione vediamo come il diagramma e le equazioni risultano molto utili anche quando invece vogliamo "progettare" un circuito sequenziale che svolga una determinata operazione.

Supponiamo di voler progettare un circuito con un input, x, e un output, y che si comporti come segue:

- Ad ogni ciclo di clock noi passiamo all'input x un valore Booleano;
- L'output y del circuito deve essere 1 ogni volta che gli ultimi tre bit che abbiamo dato in input formano la sequenza 101.

Per esempio, se gli input x passati al circuito nei primi $t = 1, 2, \ldots, 10$ cicli di clock sono

t	1	2	3	4	5	6	7	8	9	10	...
x	1	1	0	1	0	1	1	0	0	1	...

allora i valori restituiti in output dal circuito nei rispettivi cicli di clock devono essere

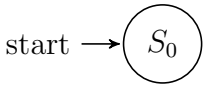
t	1	2	3	4	5	6	7	8	9	10	...
y	0	0	0	1	0	1	0	0	0	0	...

Infatti y deve essere 1 al quarto ciclo di clock, perché nei cicli $2, 3$ e 4 la sequenza di bit in input è stata 101, e deve essere 1 anche al sesto ciclo di clock perché nei cicli $4, 5, 6$ la sequenza in input è stata 101. Per tutti gli altri cicli t l'output y deve essere 0 perché la sequenza di bit in input nei cicli $(t-2,\ t-1,\ t)$ è diversa da 101 per ogni $t \neq 4, 6$.

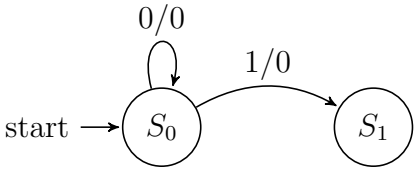
Esercizio 4. Possiamo costruire un circuito che si comporti così? E se sì come procediamo? Provate a pensarci un attimo prima di andare avanti.

Mentre progettare il circuito pensando direttamente a quali porte logiche usare e come connetterle ai Flip-Flop può sembrare un'impresa abbastanza ardua, se partiamo dall'astrazione che ci fornisce il diagramma di stato, il progetto del circuito diventa molto più facile. Proviamo.

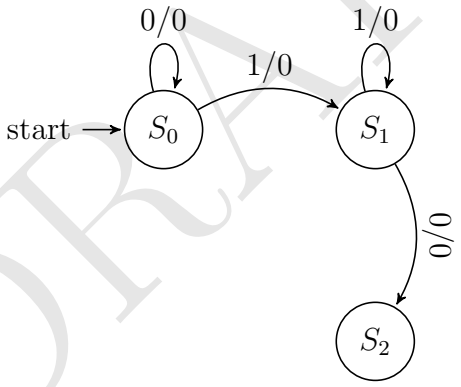
Il nostro circuito dovrà partire da uno stato iniziale S_0.



Il primo input x sarà o 0 o 1. Cosa deve fare il circuito nei due casi? Sicuramente in entrambi i casi deve restituire 0 in output, ma se l'input è $x = 0$ allora è come se fossimo ancora nello stato iniziale, mentre se $x = 1$, quello potrebbe essere il primo bit della sequenza 101 che il circuito deve "riconoscere", quindi mandiamo il circuito in un nuovo stato S_1. Potete pensare allo stato S_1 informalmente come quello che si "ricorda" che l'ultimo bit letto è 1.



Cosa deve fare il circuito quando si trova nello stato S_1? Se l'input è $x = 0$ allora quello potrebbe essere il secondo bit della nostra sequenza 101, quindi mandiamo il circuito in un nuovo stato, S_2, che deve "ricordarsi" che gli ultimi due bit letti sono 10. Se invece $x = 1$ allora vuol dire che gli ultimi due bit letti sono 11, ma quindi ai fini di riconoscere la sequenza 101, il circuito deve semplicemente ricordarsi che l'ultimo bit letto è 1. Abbiamo già uno stato che si ricorda che l'ultimo bit letto è 1: S_1.



Se siamo nello stato S_2 vuol dire che gli ultimi due bit letti sono 10. Quindi se ora leggiamo $x = 0$ vuol dire che gli ultimi tre bit letti saranno 100: restituiamo in output 0 e mandiamo il nostro circuito nello stato iniziale S_0. Se invece $x = 1$ abbiamo completato la nostra sequenza 101 e quindi restituiamo in output $y = 1$. In quale stato mandiamo il circuito? Beh, l'ultimo 1 letto potrebbe essere il primo bit di una nuova sequenza 101, quindi mandiamo il circuito nello stato che si ricorda che l'ultimo bit letto è 1.

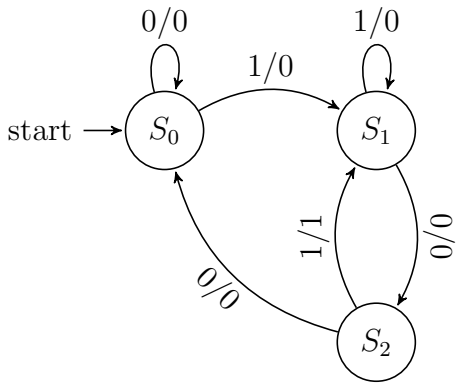


Figura 3: Diagramma di stato di un circuito che restituisce 1 se e solo se gli ultimi tre bit in input sono 101

A questo punto tutto il lavoro concettuale di progettazione è finito. Non ci resta che tradurre il diagramma di stato in Figura 3 in un circuito, ripercorrendo a ritroso i passaggi che abbiamo svolto nella sezione precedente.

Siccome il nostro diagramma ha tre stati, S_0, S_1 e S_2, avremo bisogno di almeno due Flip-Flop. Se codifichiamo i tre stati del nostro diagramma con le coppie di bit $S_0 = (0, 0)$, $S_1 = (0, 1)$ e $S_2 = (1, 0)$ e indichiamo come al solito con Q_1 e Q_2 gli stati attuali dei due Flip-Flop e con D_1 e D_2 gli stati futuri, dal diagramma di stato in Figura 3 otteniamo la seguente tabella di stato

Q_1	Q_2	x	D_1	D_2	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	X	X	X
1	1	1	X	X	X

Nella tabella ho inserito delle `X` sulle righe corrispondenti allo stato $(Q_1, Q_2) = (1, 1)$, perché se il nostro circuito parte dallo stato $(Q_1, Q_2) = (0, 0)$ non andrà *mai* nello stato $(1, 1)$, quindi "non importa" quali valori assegnamo a D_1, D_2 e y in corrispondenza di quelle righe.

Sappiamo già come possiamo ricava una formula da una tabella di verità: per esempio, la mappa di Karnaugh per la colonna D_1 è

$x \backslash {}^{Q_1}{}_{Q_2}$	00	01	11	10
0	0	1	X	0
1	0	0	X	0

quindi una formula per D_1 è $Q_2\overline{x}$.

Esercizio 5. Verificare che dalla tabella di stato si ottengono le equazioni seguenti

$$\begin{cases} D_1 &= Q_2\overline{x} \\ D_2 &= x \\ y &= Q_1 x \end{cases}$$

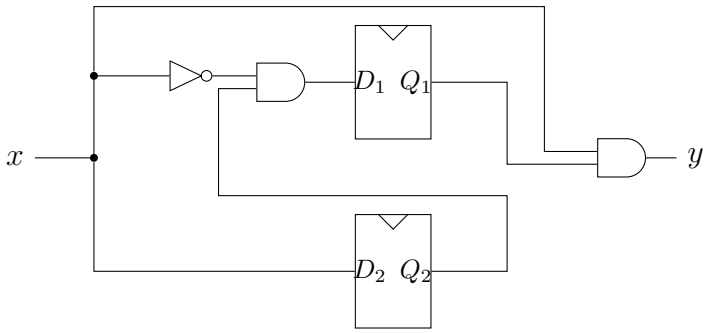


Figura 4: Un circuito che implementa il diagramma in Figura 3

A questo punto non ci resta che disegnare il circuito corrispondente alle equazioni dell'esercizio precedente.

Nella costruzione del nostro circuito abbiamo scelto di codificare gli stati del diagramma in Figura 3 con le sequenze binarie $S_0 = (0, 0)$, $S_1 = (0, 1)$ e $S_2 = (1, 0)$. È utile soffermarsi un attimo a riflettere che questa scelta è arbitraria: se avessimo scelto un'altra codifica avremmo ottenuto un circuito diverso ma equivalente.

Esercizio 6. Che circuito ottenete se codificate gli stati del diagramma in Figura 3 con $S_0 = (0, 0)$, $S_1 = (1, 1)$ e $S_2 = (0, 1)$? E se li codificate con $S_0 = (1, 0, 0)$, $S_1 = (0, 1, 0)$, e $S_2 = (0, 0, 1)$?[1]

Esercizio 7. Alla luce di quanto visto in questa sezione, riuscite a descrivere a parole cosa fa il circuito in Figura 1?

3 Macchine a stati finiti

I circuiti visti in questo episodio si chiamano *macchine* (o *automi*) a stati finiti e possiamo schematizzarli come in Figura 5: ogni blocco del circuito è o un circuito combinatorio o un registro; tutti i registri sono *sincronizzati* allo stesso clock e in ogni ciclo contenuto nel circuito c'è almeno un registro.

Questi circuiti si distinguono in macchine *alla Mealy* (quelli in cui gli output dipendono anche dagli input *attuali*) e le macchine *alla Moore* (quelli in cui gli output dipendono soltanto dallo stato del circuito). Osservate che gli esempi visti in questo episodio sono macchine alla Mealy.

Esercizio 8. Come possiamo costruire una macchina alla Moore equivalente alla macchina alla Mealy in Figura 4?

Esercizio 9. Progettare una macchina a stati finiti che legge una sequenza infinita di bit in input e restituisce 1 ogni volta che gli ultimi quattro bit letti sono 1001 e restituisce 0 in tutti gli altri casi.

[1] Quest'ultimo tipo di codifica si chiama *one-hot*: ad ogni ciclo di clock uno e uno solo dei Flip-Flop si trova nello stato 1

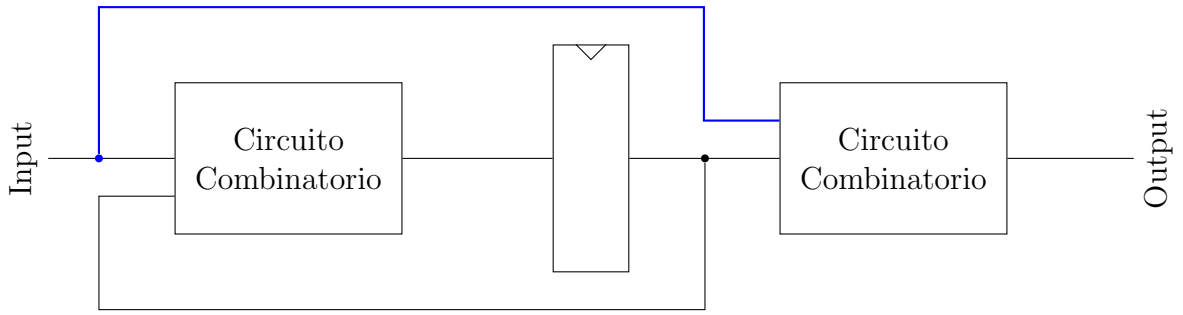


Figura 5: Schema di una macchina a stati finiti. Alla Mealy (con la linea blu: gli output dipendono anche dagli input attuali) o alla Moore (senza la linea blu: gli output dipendono solo dallo stato del registro)

Esercizio 10. Un *contatore binario* a n bit è un circuito sequenziale con un ingresso di `reset` e n output, che rappresentano un numero fra 0 e $2^n - 1$ espresso in binario. Quando il `reset` viene attivato, tutti gli n bit in output vengono inizializzati a 0; successivamente, ad ogni ciclo di clock gli n bit in output devono rappresentare un numero incrementato di 1 rispetto al precedente.

1. Progettare un contatore binario a n bit usando un sommatore a n bit e un registro a n bit;
2. Progettare un contatore a quattro bit usando solo blocchi Half-Adder e Flip-Flop.

Esercizio 11. È possibile progettare un automa a stati finiti che legga una sequenza di bit in input $(x_0, x_1, x_2, \dots)$ e restituisca in output una sequenza di bit $(y_0, y_1, y_2, \dots)$ che rappresenti il complemento a due del numero in input (l'ordine con cui vengono letti e restituiti i bit va dal bit meno significativo al più significativo[2])? Se pensate che sia possibile progettate l'automa, altrimenti motivate la risposta.

[2]Per esempio, se i primi quattro bit che riceve l'automa sono $x_0 = 1$, $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, questi rappresentano il numero $0011_{\overline{2}} = x_0 2^0 + x_1 2^1 + x_2 2^2 - x_3 2^3 = 3_{10}$. L'automa dovrà restituire il complemento a due di questo numero, cioè $1101_{\overline{2}} = 1 * 2^0 + 0 * 2^1 + 1 * 2^2 - 1 * 2^3 = -3_{10}$, quindi la sequenza di bit $y_0 = 1$, $y_1 = 0$, $y_2 = 1$, $y_3 = 1$. Che sequenza dovrebbe restituire se il quinto bit che legge è $x_4 = 0$? e se invece il quinto bit è $x_4 = 1$?