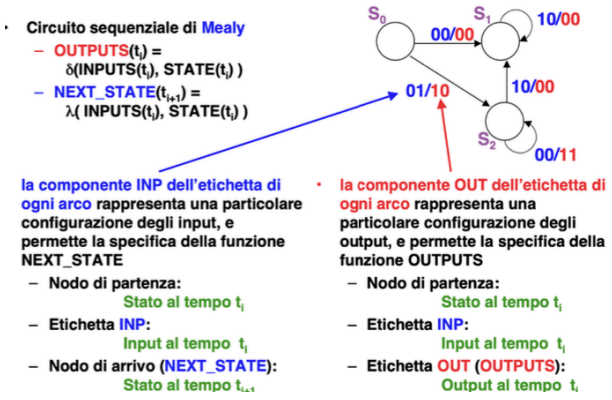
18 Aprile 2023

ARCHITETTURE DI CALCOLO LEZIONE 10

Automi di Mealy e Moore, esercizi

Automa a stati finiti di Mealy

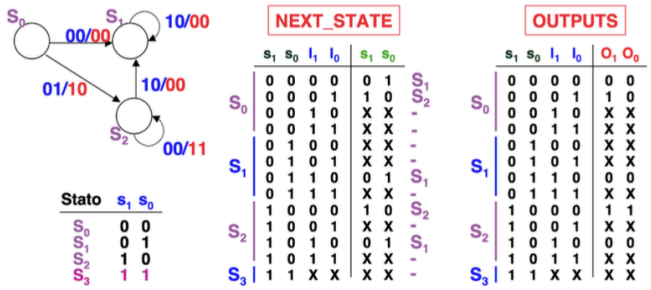


Due aspetti contraddistinguono tale circuito da quello di Moore:
•I valori presenti nei nodi, che rappresentavano gli output, devono essere rimossi. Questo perché non essendoci più un'associazione 1 a 1 tra lo stato e l'output non possiamo inserire in modo diretto l'output.
•Nelle etichette sugli archi possiamo distinguere 2 componenti distinte, input e output, anziché il solo input.

Esempio automa di Mealy a 3 stati

Prendiamo l'automa di Mealy portato nell'immagine a destra, formato da tre stati S0, S1, S2. Sono necessari 2 flip flop (2 bit= 4 configurazioni) per indicare i 3 stati che si vogliono rappresentare nel registro, mentre una configurazione rimane indeterminata. Osserviamo che gli archi sono diversi rispetto a come li abbiamo precedentemente visti nel caso Moore; infatti, sono riportati sia l'input (blu) che l'output prodotto (rosso). Questo perché uno stesso stato può produrre output diversi in base all'input ricevuto.
Ad esempio, se siamo nello stato S0 ed arriva per input 00 andremo nello stato S1 ed avremo output 00, se l'input ricevuto risulta 01 andremo invece nello stato S2 con output prodotto 10.
Ricordiamo: nel caso dell'automa di Mealy, sia l'output che il next state dipendono dall'input.
Nell'esempio riportato abbiamo input, output e stato tutti a due bit ma non è detto che ciò si verifichi sempre e/o che essi coincidano, non c'è un legame diretto. L'unica condizione è che il numero delle etichette devono contenere tutti i possibili stati.
Nella funzione Next_State, il nodo di partenza indica lo stato a tempo ti, l'etichetta blu INP indica l'input al tempo ti e il nodo di arrivo indica lo stato al tempo ti+1. Per quanto riguarda la funzione di output, il nodo di partenza indica lo stato a tempo ti, l'etichetta blu INP indica l'input al tempo ti e l'etichetta rossa OUT indica l'output prodotto al tempo ti.
Una volta costruito il nostro automa è importante derivare le tavole di verità per la creazione del circuito. Da notare che le etichette possono avere invece che un simbolo direttamente la

sequenza di bit che compongono la configurazione, anche se, per una questione di leggibilità, è preferibile in generale usare le etichette.
Partendo dallo stato S0, se l'input è 00 il next state è S1 in quanto i nuovi s1 e s2 (possono anche essere indicati da un asterisco) 01 corrispondono alla configurazione dello stato S1.

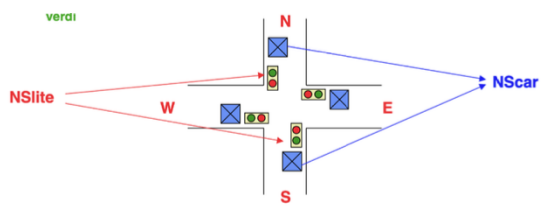


Se ci troviamo in S1 vediamo che tranne nella condizione di input 00 non abbiamo un next state (condizione d'indeterminazione X X); in caso di input 00 si ritorna nello stato S1 (l'automa in questione non ha un'applicazione reale).
Nello stato S2 di fatto gli input significativi sono solo gli input 00 e 11.

Lo stato S3 (che non ci interessa) viene aggiunto per completezza e, per evitare di aggiungere 4 righe nella tabella di verità che avrebbero solo output indifferente, possiede solo la configurazione 11 XX, ad indicare che qualsiasi valore input avrebbe prodotto un qualsiasi valore in output (DON'T CARE).
Nella tabella output vediamo, ad esempio, che nello stato S0 avremo output solo in caso di input 00 e 01, mentre le altre due righe danno DON'T CARE.
Di queste tabelle di verità è possibile creare le mappe di Karnaugh, una per ogni output prodotto; in questo caso sarà necessario creare 4 mappe, 2 per il next state e due per l'output.
Ricordiamo che non è necessaria la costruzione di tali mappe (a meno che non venga richiesto) ma serve a semplificare/minimizzare il circuito.
Notiamo che la X si usa come forma di indeterminazione sia per l'input che per l'output

Esempio di circuito di Moore

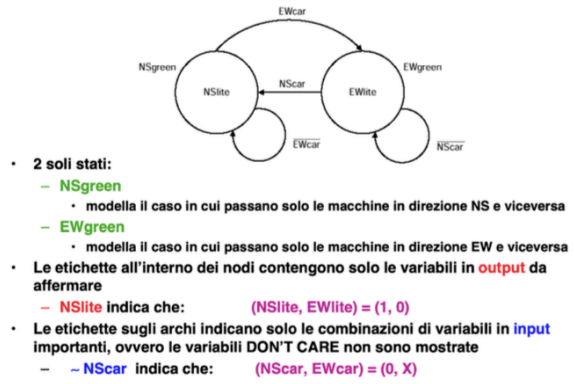
Questo è un circuito molto semplificato che controllo i semafori di un incrocio. Tale sistema consta di input, dati da alcuni sensori sull'asfalto che controllano se sono presenti macchine in attesa, e di output che determinano l'accensione (rosso/verde) dei semafori. Il colore del semaforo è uguale nei semafori opposti (N-S, E-W) e contrario rispetto ai semafori perpendicolari.
Questi semafori risultano intelligenti in quanto il cambio di stato avviene solo se, a fine del ciclo temporale, il sensore avverte la presenza di auto su quella strada; se eventualmente al cambio del clock non passa alcuna auto, il semaforo rimane rosso in favore del semaforo perpendicolare che rimane verde.



Per rappresentare tale circuito sono necessari due bit in input collegati ai sensori, che segnalano l'arrivo delle macchine:
•NScar, avvisa dell'arrivo di auto nella strada Nord Sud;
•EWcar, avvisa dell'arrivo di auto in direzione Est Ovest.

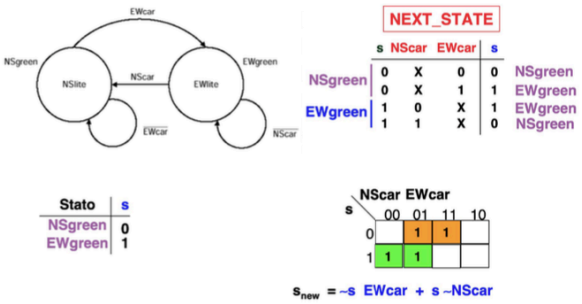
Infine, abbiamo due bit in output:

- NSlite, indica che i semafori sulla strada Nord Sud sono verdi;
- EWlite, indica che i semafori in direzione Est Ovest sono verdi.



Iniziamo la costruzione del grafo partendo dal disegno degli stati, ossia 2 nodi, a cui applichiamo le rispettive etichette:
•NSgreen, verde sulla strada Nord Sud
•EWgreen, verde in direzione Est Ovest.
All'interno dei nodi inseriamo i possibili output prodotti, mentre creaiamo degli archi che collegano i nodi in base agli input.
Ad esempio, se si è nello stato NSgreen, si ha output NSlite. Allo scadere del ciclo dle clock viene controllato l'input (presenza di auto nella direzione perpendicolare):

- Se EWcar è 1 (positivo, passano auto in direzione perpendicolare), avremo il passaggio allo stato EWgreen e l'output prodotto è EWlite, ossia semaforo verde in direzione Est Ovest.
- Se, invece, EWcar è zero (negativo, non ci sono auto in direzione EW), allora il semaforo rimane verde in direzione NS fino al prossimo ciclo di clock.

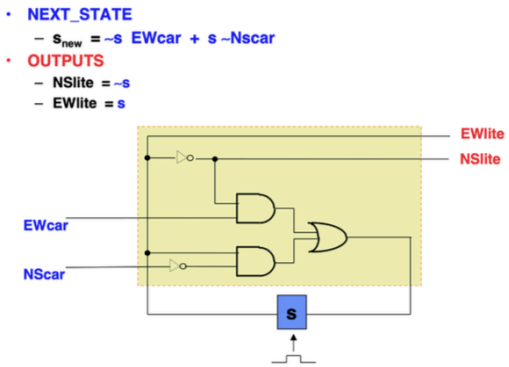


Passiamo, una volta costruito il grafo, alla creazione delle tabelle di verità, una per lo stato (etichette), una per il Next_State ed una per gli output.
La tabella Next_State qui riportata presenta delle configurazioni in meno (dovrebbero essere 8) poiché si riducono tramite l'uso del don't care.

s	NScar	EWcar	S*
0	0	0	0
0	1	0	0
0	0	1	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Ad esempio, nella prima riga avremmo al posto della riga con X due righe che hanno al suo posto 0 ed 1 ma si può omettere in quanto il fatto che tale valore sia 0 o 1 non influisce sul next state, il quale risulta dipendere solo dal EWcar. Qui a sinistra la tabella di verità senza omissioni.

Si passa poi alla costruzione della mappa di Karnaugh e la ricerca della prima forma canonica per il S(new).
Lo stesso processo vale per gli outputs.
In questo caso risulta inutile costruire la tavola di Karnaugh e si passa subito alla forma canonica.
A questo punto, infine, si costruisce il circuito. Esso presenterà due input, EWcar e NScar, due output, EWlite e NSlite, ed un flip flop a 2 bit, contente i due stati.



Essendo un circuito sincrono il flip flop è ovviamente dotato di clock. Dalla formula next state vediamo che risulta necessario mettere in OR NScar negata (porta NOT) in AND con EWcar con EWcar in AND con s negata, che ritorna ad s ad indicare il nuovo stato da memorizzare nel registro. L'output prevede banalmente EWlite nel caso di s e NSlite quando si ha in uscita NOT s.

Scelta della frequenza del clock

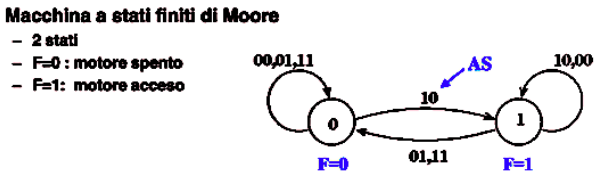
La frequenza del clock determina il momento in cui il valore del prossimo stato viene memorizzato. Ciò significa che se ho, ad esempio, un clock con periodo di 10 minuti, qualunque siano gli input lo stato rimarrà sempre lo stesso; per questo tale periodo deve essere scelto in modo adeguato alle esigenze. Una volta scelto il periodo adeguato, la frequenza del clock è data dall'inverso del periodo (esempio 1/60 secondi= 0,17 Hz)

Esericizi

Esercizio 1

- Progettare una rete sequenziale per il controllo di un motore elettrico.
- La rete riceve in input i segnali relativi a due pulsanti A e S
  - A=1 ⇒ accendi
  - S=1 ⇒ spegni
  - In caso di pressione simultanea, S prevale.
- Se il motore è acceso (o spento) e arriva un altro segnale di accensione (o spegnimento), la rete deve ignorare il segnale.
- La rete deve dare in output il segnale O:
  - O = 0 ⇒ motore spento
  - O = 1 ⇒ motore acceso
- Si richiede di:
  1. Disegnare la macchina a stati finiti
  2. Scrivere la tabella di verità
  3. Trovare le forme SP minime
  4. Disegnare il circuito

1. Si parte disegnando i nodi, ossia gli stati, che in questo caso sono 2 come gli output (Acceso e spento), a cui diamo un'etichetta (nell'esempio F); all'interno dei due nodi andiamo ad inserire l'output prodotto, 1 e 0. Si passa poi agli archi, ossia gli input; gli archi sono 4, due permettono il passaggio da uno stato all'altro mentre gli altri due



ritornano nello stesso nodo di partenza. Sopra gli archi inseriamo le possibili configurazioni AS che portano a tale stato. Soluzione a sinistra.

2. Si passa poi alla costruzione delle tavole di verità, una per il Next State ed una per gli output.

NEXT_STATE

F	A	S	F*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

OUTPUTS

F	O
0	0
1	1

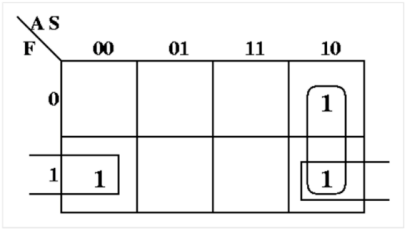
O = F
F* = ~FA~S + F~A~S + FA~S

3. Cerchiamo e minimizziamo le forme SP o direttamente tramite la ricerca delle equazioni che portano a valore 1 o, se serve, tramite costruzione della mappa di Karnaugh; in questo caso la mappa si fa solo per il next state, mentre per l'output si va direttamente.

- **Mimimizzazione**

F	A	S	F*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

F	O
0	0
1	1

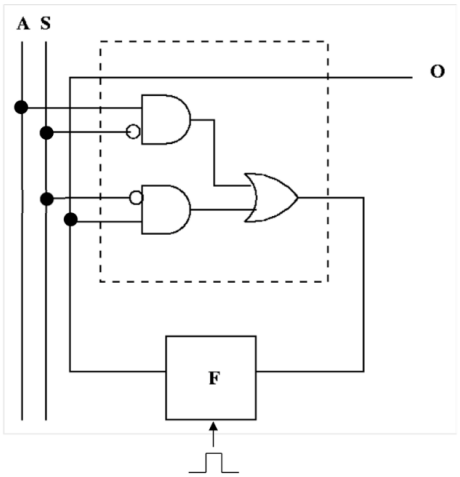


O = F
F* = A~S + F~S

4. Infine, si costruisce il circuito.

Circuito

O = F
F* = A~S + F~S



N.B.(da leggere da sinistra verso destra !!!)

Esercizio 2

- Progettare una rete sequenziale che comanda l'accensione e lo spegnimento di tre lampadine ($Lamp_S$,$Lamp_D$,$Lamp_C$) in sequenza
- L'output del circuito sono tre bit che per comodità chiamiamo: S,C,D.
  - Quando questi sono affermati, le lampadine corrispondenti sono accese
- Il ritmo del circuito è determinato dal periodo di clock
- La rete riceve un segnale di ingresso I tale che:
  - se I = 0 ⇒ le lampadine devono accendersi in sequenza, una alla volta, partendo (la prima volta) da S
    100→ 010→ 001→ 100→ …
  - se I = 1 ⇒ le lampadine devono accendersi in sequenza, due alla volta, partendo (la prima volta) da S e C
    110→ 011→ 101→ 110→ …
- Determinare: Macchina a stati di Moore + Tabelle + Equazioni minime

- Macchina a stati finiti di Moore
  - 6 stati, corrispondenti alle possibili combinazione degli output

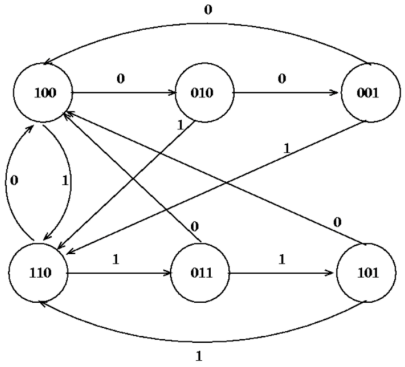
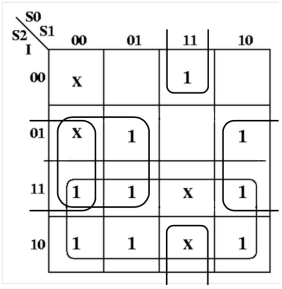


Tabella di verità per *NextState*

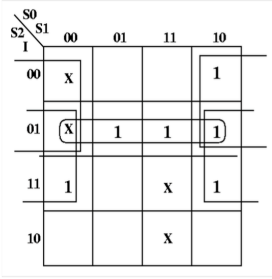
S0	S1	S2	I	S0*	S1*	S2*
0	0	0	0	X	X	X
0	0	1	0	1	0	0
0	1	0	0	0	0	1
0	1	1	0	1	0	0
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	0	0	X	X	X
0	0	0	1	X	X	X
0	0	1	1	1	1	0
0	1	0	1	1	1	0
0	1	1	1	1	0	1
1	0	0	1	1	1	0
1	0	1	1	1	1	0
1	1	0	1	0	1	1
1	1	1	1	X	X	X



S0*=S0 S1 ~I + S2 + ~S0 I + ~S1 I

Tabella di verità per *NextState*

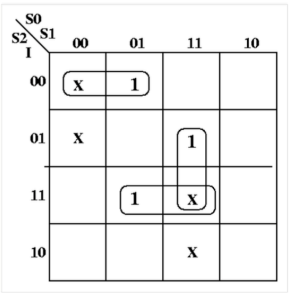
S0	S1	S2	I	S0*	S1*	S2*
0	0	0	0	X	X	X
0	0	1	0	1	0	0
0	1	0	0	0	0	1
0	1	1	0	1	0	0
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	0	X	X	X
0	0	0	1	X	X	X
0	0	1	1	1	1	0
0	1	0	1	1	1	0
0	1	1	1	1	0	1
1	0	0	1	1	1	0
1	0	1	1	1	1	0
1	1	0	1	0	1	1
1	1	1	1	X	X	X



S1* = ~S2 · I + ~S1 · ~S2 + ~S1 · I

Tabella di verità per *NextState*

S0	S1	S2	I	S0*	S1*	S2*
0	0	0	0	X	X	X
0	0	1	0	1	0	0
0	1	0	0	0	0	1
0	1	1	0	1	0	0
1	0	0	0	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	0	X	X	X
0	0	0	1	X	X	X
0	0	1	1	1	1	0
0	1	0	1	1	1	0
0	1	1	1	1	0	1
1	0	0	1	1	1	0
1	0	1	1	1	1	0
1	1	0	1	0	1	1
1	1	1	1	X	X	X



S2*=~S0 ~S2 ~I + S0 S1 I + S1 S2 I

- **Tabella di verità per *Output***

S0	S1	S2	S	C	D
0	0	0	X	X	X
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	X	X	X

S = S0
C = S1
D = S2

S_1	C_1	D_1	I	S	C	D
1	0	0	1	1	1	0
1	0	0	0	0	1	0
0	1	0	1	1	1	0
0	1	0	0	0	0	1
0	0	1	1	1	1	0
0	0	1	0	1	0	0
1	1	0	1	0	1	1
1	1	0	0	1	0	0
0	1	1	1	1	0	1
0	1	1	0	1	0	0
1	0	1	1	1	1	0
1	0	1	0	1	0	0
0	0	0	X	X	X	X
1	1	1	X	X	X	X

Preso dalle loro sbobine non so a cosa corrisponda