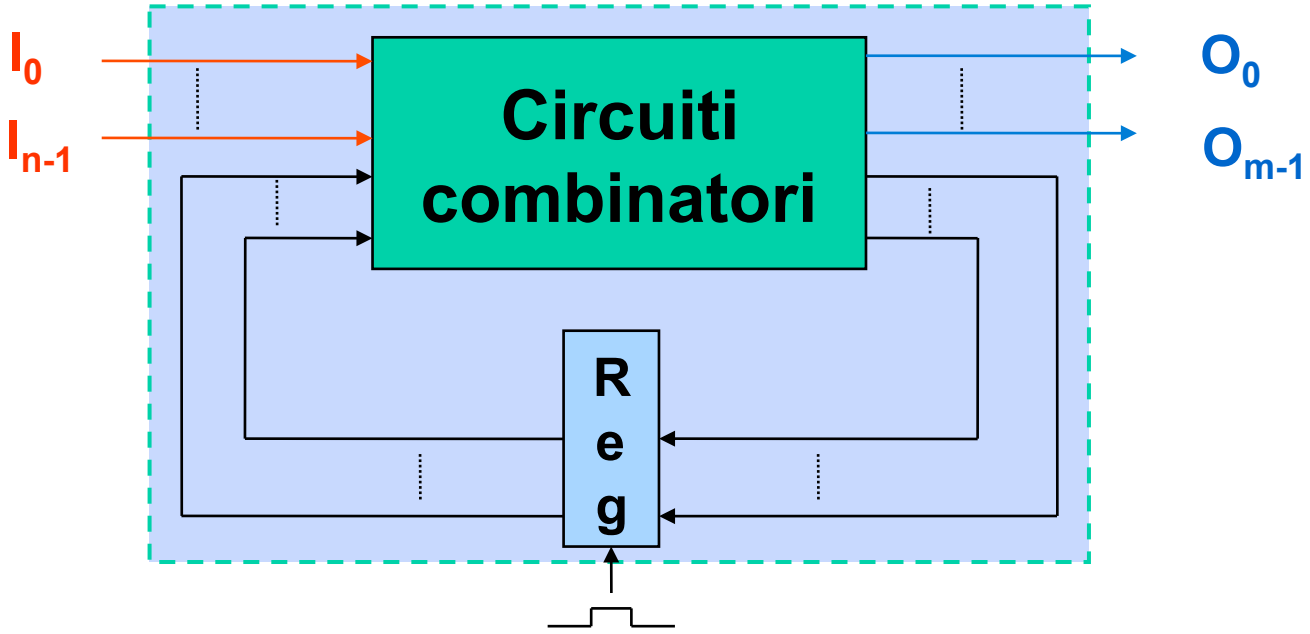
Architettura degli Elaboratori

Sintesi di circuiti sequenziali

slide a cura di Salvatore Orlando, Andrea Torsello, Marta Simeoni

Circuito sequenziale sincrono



Il comportamento di un circuito sequenziale è determinato dai circuiti combinatori, che calcolano funzioni in base al

- valore dello stato contenuto in Reg
- valore degli *n* input $I_0 \ldots I_{n-1}$

Dobbiamo quindi specificare gli output del circuito (OUTPUT & NEXT_STATE) per tutte le combinazioni significative di stato e input

Automi per specificare circuiti di Moore

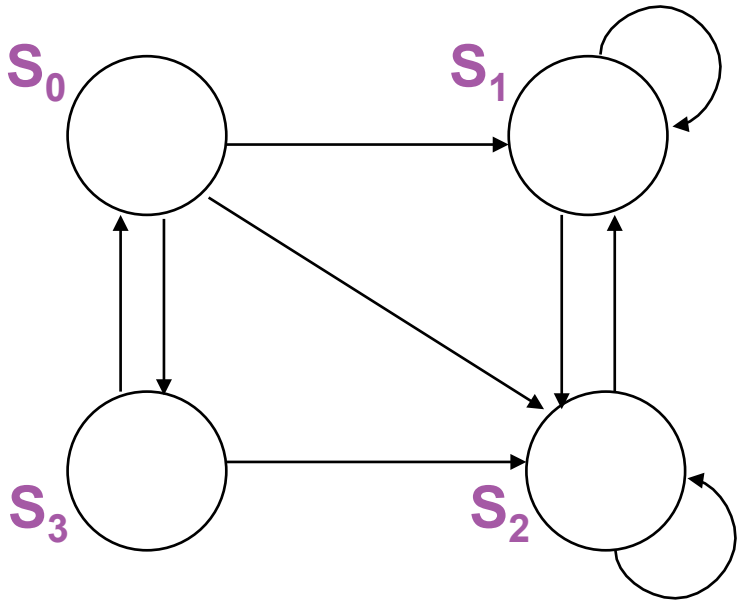
Circuito sequenziale di Moore

- $OUTPUT(t_i) = \delta(STATE(t_i))$
- $NEXT\_STATE(t_{i+1}) = \lambda(INPUT(t_i), STATE(t_i))$

Specifica tramite Automa a stati finiti

- grafo diretto, con un numero finito di nodi
- nodi = stati (configurazioni possibili degli elementi di memoria presenti del circuito)
- archi = transizioni di stato

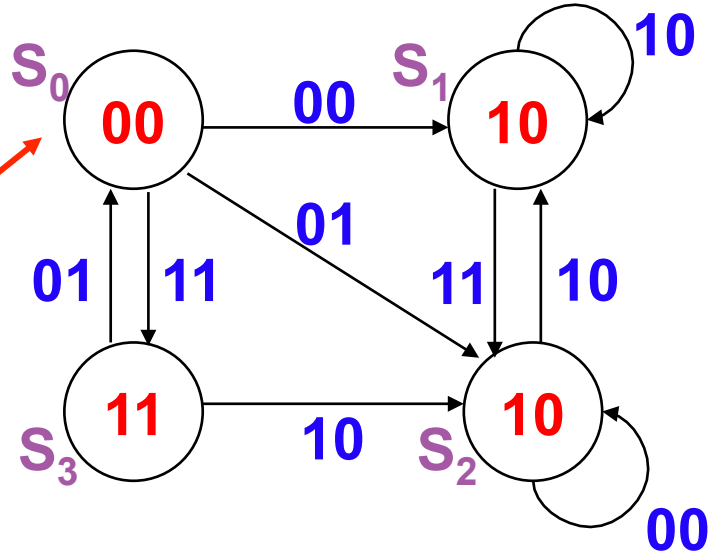
Etichetta che individua una specifica configurazione del registro di stato



Automi per specificare circuiti di Moore

Circuito sequenziale di Moore

- OUTPUT(t_i) = δ(STATE(t_i))
- NEXT_STATE(t_{i+1}) = λ(INPUT(t_i), STATE(t_i))



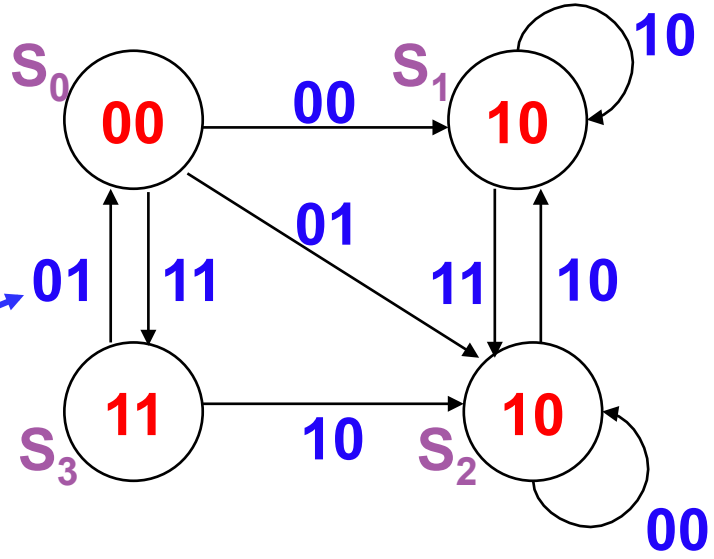
L'etichetta all'interno di ogni nodo definisce l'output del circuito come funzione dello stato corrispondente al nodo

- Nodo: Stato al tempo t_i
- Etichetta del nodo (OUTPUT): Output al tempo t_i

Automi per specificare circuiti di Moore

Circuito sequenziale di Moore

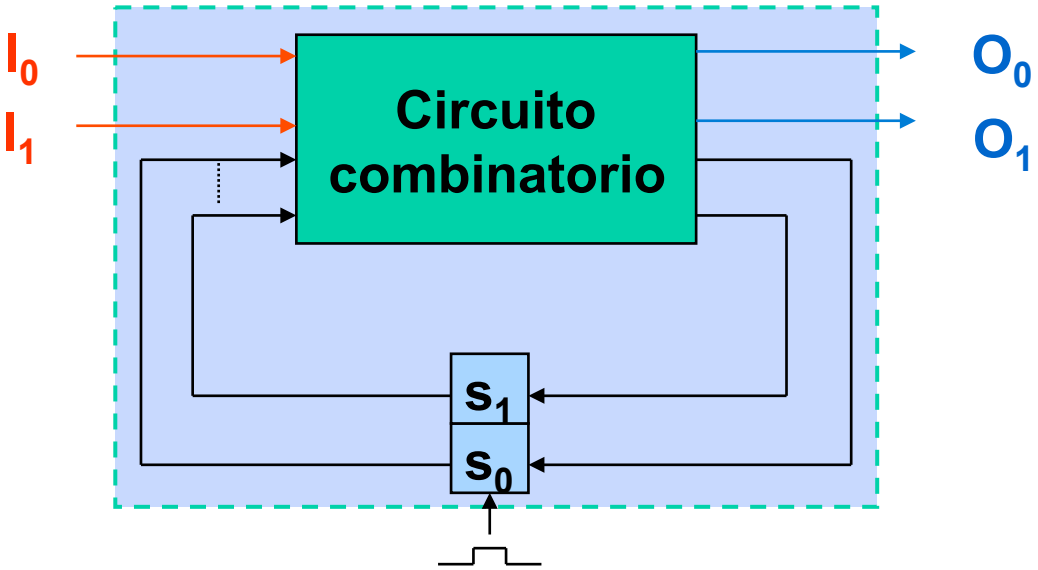
- OUTPUT(t_i) =
  δ(STATE(t_i))
- NEXT_STATE(t_{i+1}) =
  λ(INPUT(t_i), STATE(t_i))

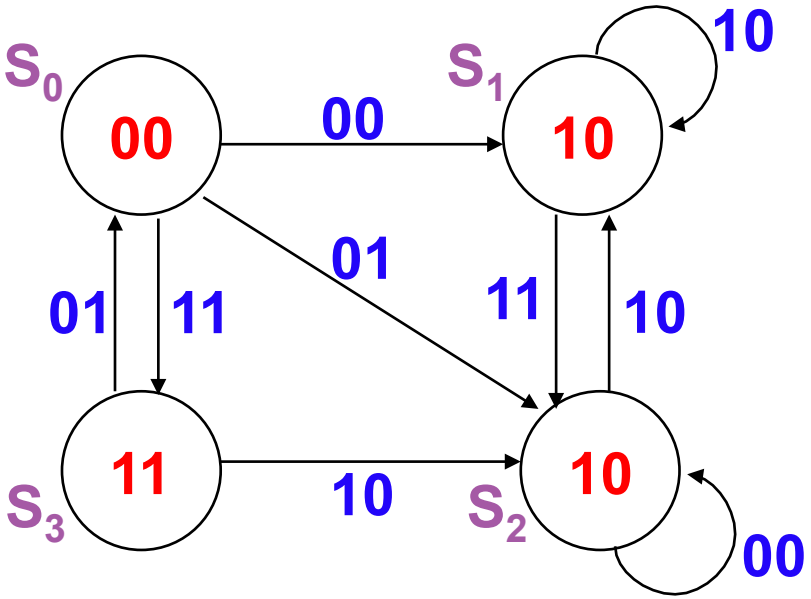


L'etichetta di ogni arco rappresenta una particolare configurazione degli input, e permette la specifica della funzione NEXT_STATE

- Nodo di partenza: Stato al tempo t_i
- Etichetta dell'arco: Input al tempo t_i
- Nodo di arrivo (NEXT_STATE): Stato al tempo t_{i+1}

Esempio di circuito di Moore





2 input e 2 output

4 stati (Reg = 2 bit)

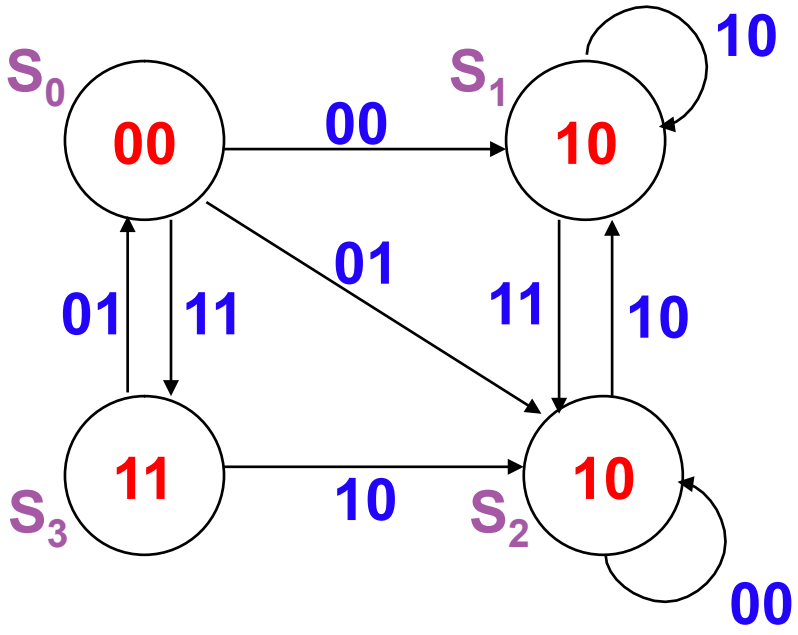
Automa a stati finiti con 4 nodi (stati) etichettati

Al più 4 archi etichettati uscenti da ogni nodo

- uno per ogni possibile configurazione dei 2 input

Le *etichette esterne* ai nodi servono solo per nominare i 4 stati

Automa di Moore e Tabelle di verità



Stato	s_1	s_0
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

NEXT_STATE

	s_1	s_0	I_1	I_0	s_1	s_0	
S_0	0	0	0	0	0	1	S_1
	0	0	0	1	1	0	S_2
	0	0	1	0	X	X	-
	0	0	1	1	1	1	S_3
S_1	0	1	0	0	X	X	-
	0	1	0	1	X	X	-
	0	1	1	0	0	1	S_1
	0	1	1	1	1	0	S_2
S_2	1	0	0	0	1	0	S_2
	1	0	0	1	X	X	-
	1	0	1	0	0	1	S_1
	1	0	1	1	X	X	-
S_3	1	1	0	0	X	X	-
	1	1	0	1	0	0	S_0
	1	1	1	0	1	0	S_2
	1	1	1	1	X	X	-

OUTPUT

	s_1	s_0	O_1	O_0
S_0	0	0	0	0
S_1	0	1	1	0
S_2	1	0	1	0
S_3	1	1	1	1

In NEXT_STATE, (s_1 s_0 = X X) corrispondono a transizioni che non dovrebbero mai verificarsi

Sintesi di un circuito di Moore

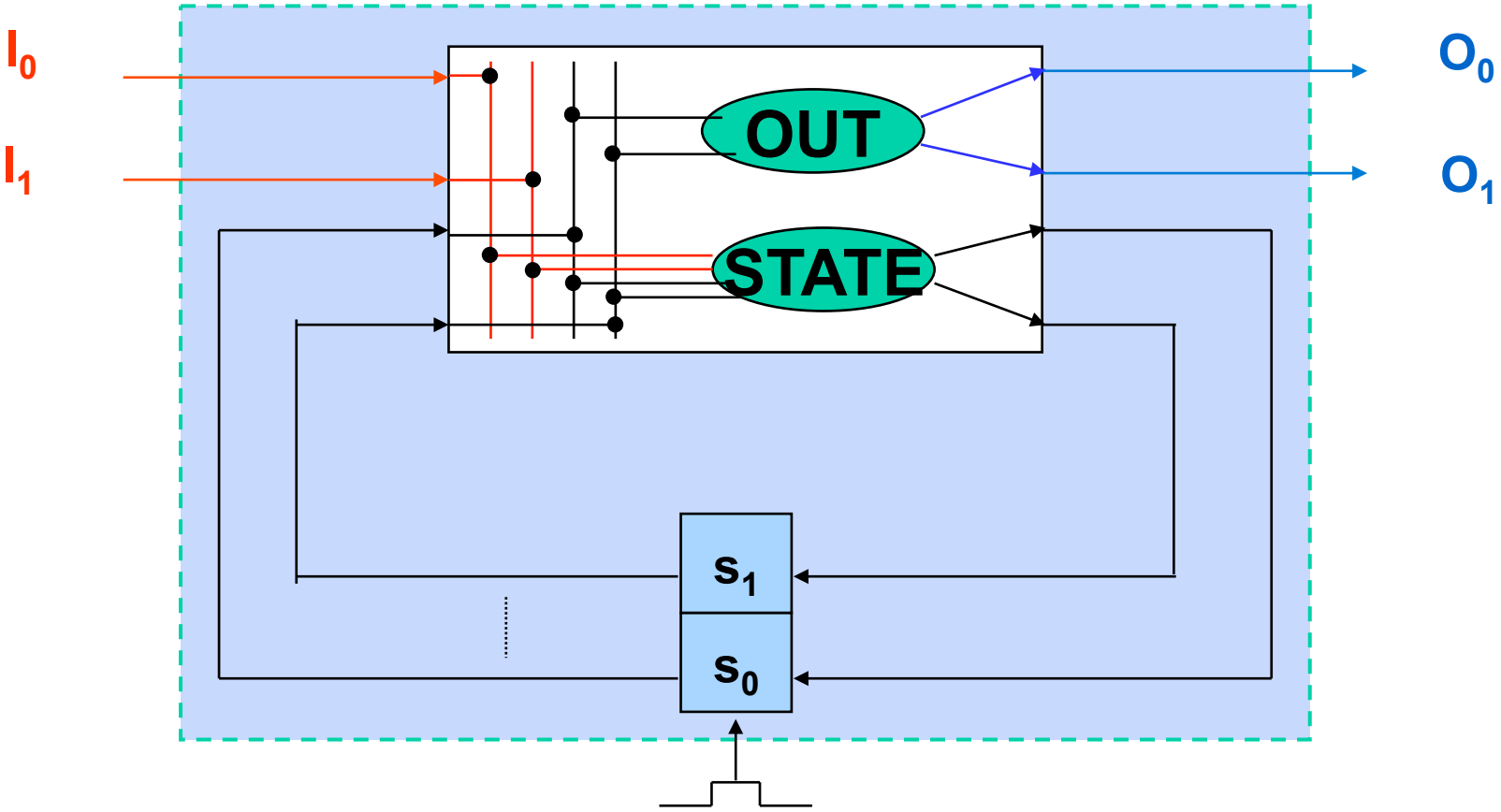
Deriviamo le mappe di Karnaugh che generano NEXT_STATE

- due mappe, una per ognuna delle 2 variabili s_0 e s_1
- equazione logica in forma SP ottimizzata, e circuito logico (STATE) a 2 livelli

Deriviamo le mappe di Karnaugh che generano OUTPUT

- due mappe, una per ognuna delle 2 variabili O_0 e O_1
- equazione logica in forma SP ottimizzata, e circuito logico (OUT) a 2 livelli

Sintesi di un circuito di Moore



Esempio di circuito di Moore

Circuito *molto semplificato,* che controlla i semafori di un incrocio tra una strada North/South e una East/West

- input: sensori sull'asfalto che controllano se sono presenti macchine in attesa
- output: segnali che determinano l'accensione (rosso/verde) dei semafori

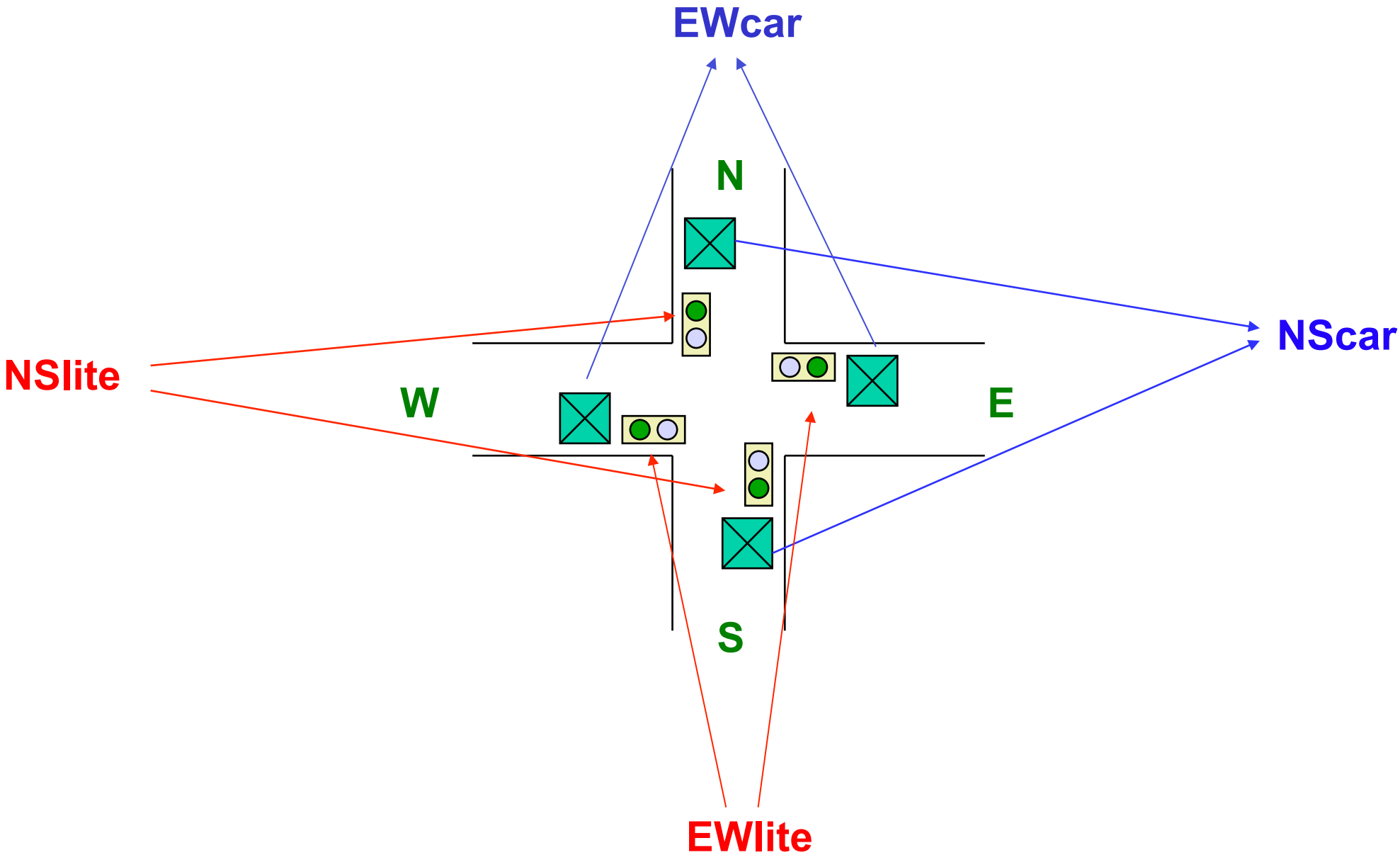
2 bit in input che sono collegati ai sensori, e segnalano l'arrivo delle macchine

- NScar e EWcar, che quando affermati segnalano al circuito la presenza di macchine nella direzione N/S e viceversa, e nella direzione E/W e viceversa

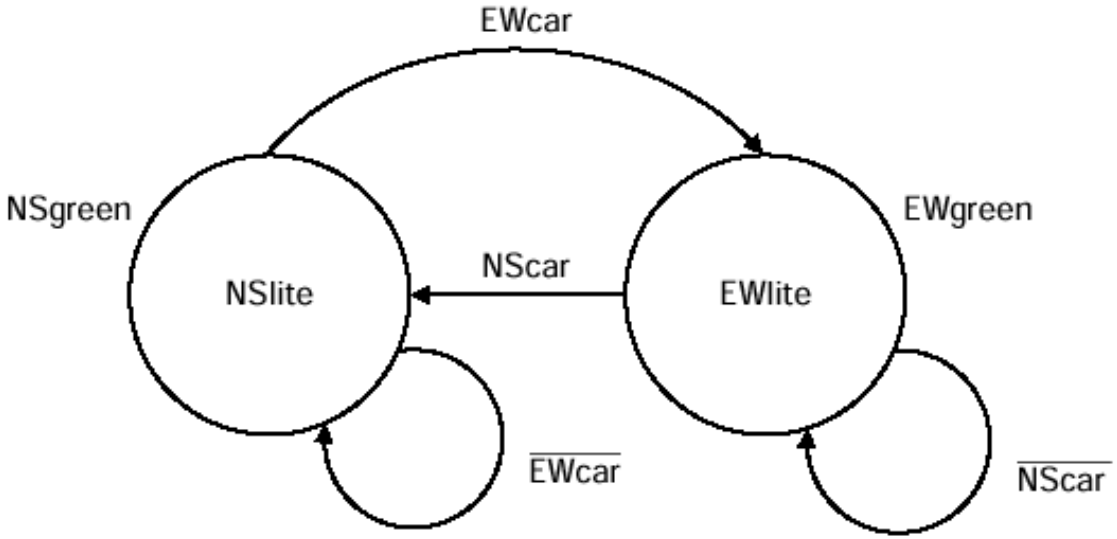
2 bit in output

- NSlite e EWlite, che quando affermati indicano che i corrispondenti semafori sono verdi

Esempio di circuito di Moore



Esempio di circuito di Moore: automa



2 soli stati:

- NSgreen
  - modella passaggio delle macchine in direzione NS (e viceversa)
- EWgreen
  - modella passaggio delle macchine in direzione EW (e viceversa)

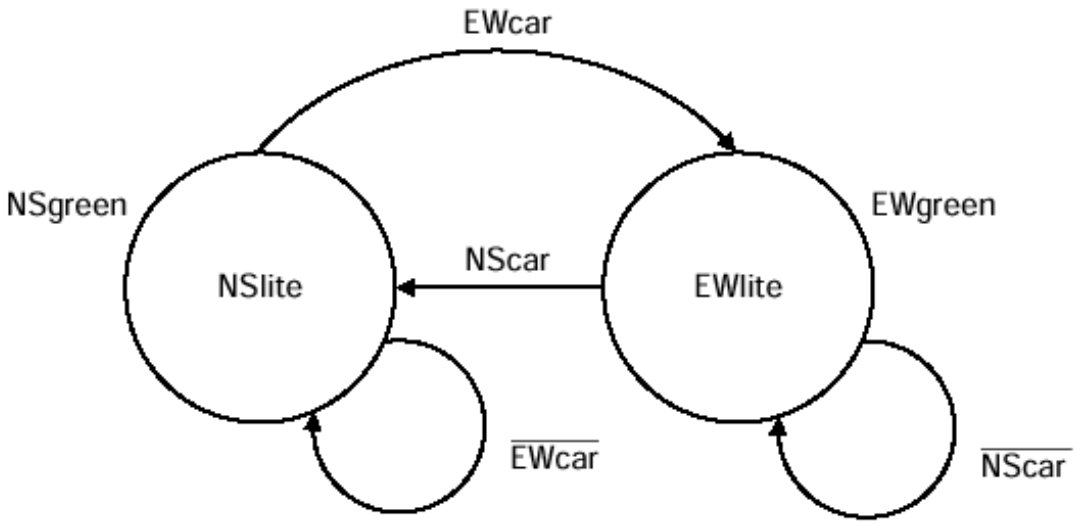
Etichette all'interno dei nodi contengono solo variabili in output da affermare

- NSlite indica che: (NSlite, EWlite) = (1, 0)
- EWlite indica che: (NSlite, EWlite) = (0, 1)

Etichette sugli archi indicano solo combinazioni di variabili in input importanti (variabili DON'T CARE non mostrate)

- ~ NScar indica che: (NScar, EWcar) = (0, X)
- ~ EWcar indica che: (NScar, EWcar) = (X, 0)

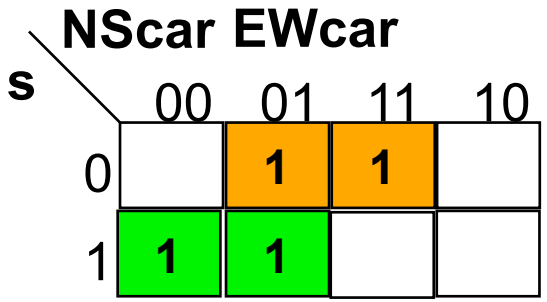
Esempio di circuito di Moore: tabelle di verità



NEXT_STATE

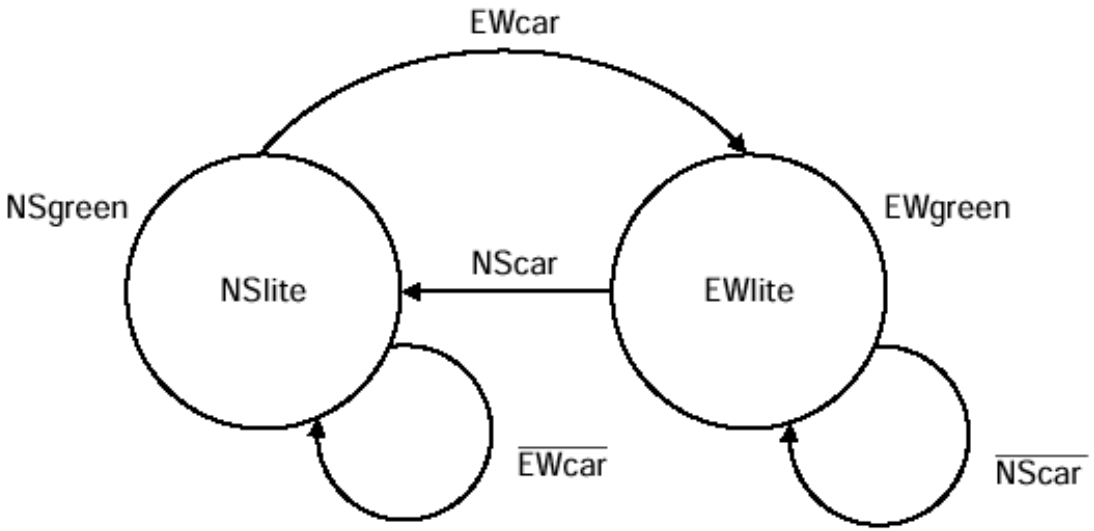
	s	NScar	EWcar	s	
NSgreen	0	X	0	0	NSgreen
	0	X	1	1	EWgreen
EWgreen	1	0	X	1	EWgreen
	1	1	X	0	NSgreen

Stato	s
NSgreen	0
EWgreen	1



$s_{new} = \sim s\ EWcar + s \sim NScar$

Esempio di circuito di Moore: tabelle di verità



OUTPUT

	s	NSlite	EWlite
NSgreen	0	1	0
EWgreen	1	0	1

NSlite = ~s

EWlite = s

Stato	s
NSgreen	0
EWgreen	1

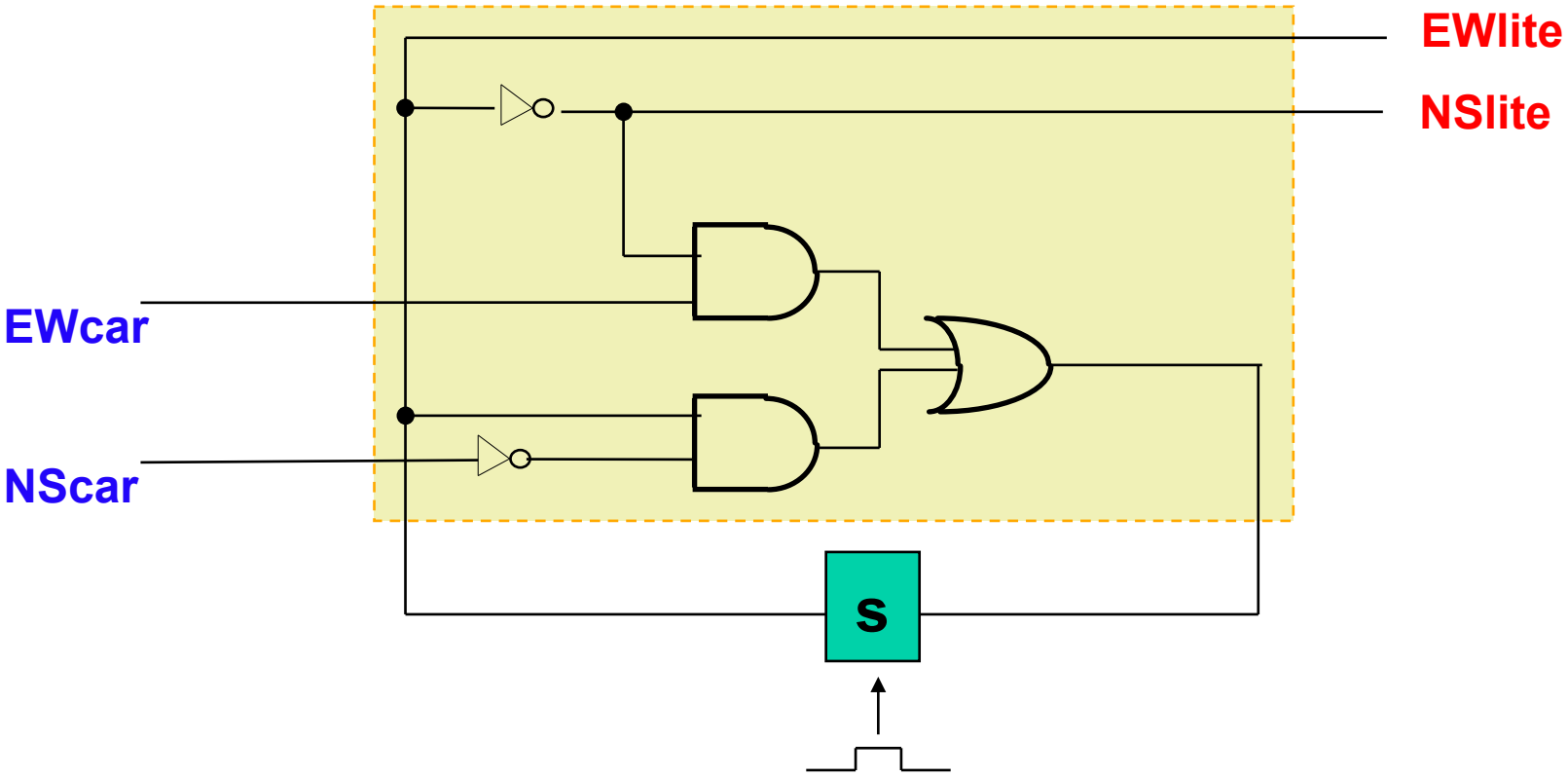
Esempio di circuito di Moore: circuito finale

NEXT_STATE

- s_{new} = ~s EWcar + s ~Nscar

OUTPUT

- NSlite = ~s
- EWlite = s



Commenti

La *frequenza del clock* determina il momento in cui il valore del prossimo stato viene memorizzato

Durante il periodo t_i del clock, OUTPUT non può cambiare

- dipende solo dallo stato

Durante il periodo t_i del clock, NEXT_STATE può cambiare man mano che cambiano gli input

- ma il nuovo stato viene memorizzato solo sul fronte di discesa (o salita) del segnale di clock

Nel circuito precedente, se vogliamo controllare una volta al minuto se dobbiamo (o meno) invertire i colori dei 2 semafori

- basta fissare un segnale di clock il cui ciclo (periodo) è di 60 s.
- Frequenza del clock = 1/60 Hz = 0,017 Hz

Automi per specificare circuiti di Mealy

Circuito sequenziale di Mealy

- OUTPUT(t_i) = δ(INPUT(t_i), STATE(t_i))
- NEXT_STATE(t_{i+1}) = λ(INPUT(t_i), STATE(t_i))

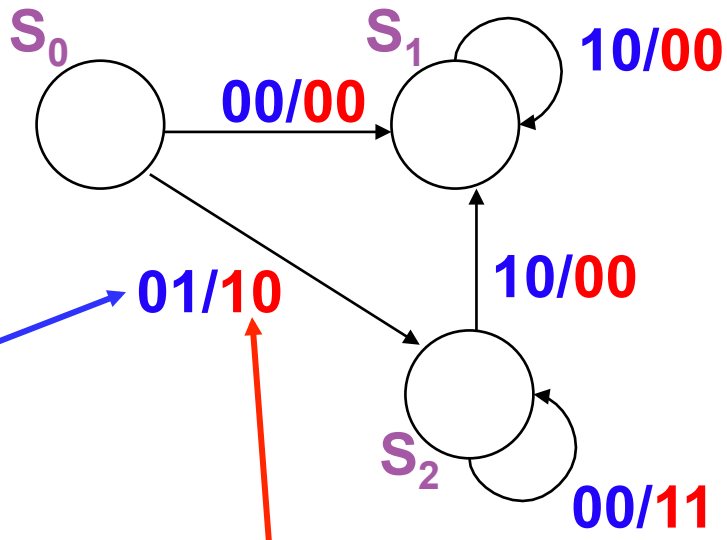
Rispetto all' Automa a stati finiti di Moore, le differenze solo le seguenti

- le etichette all'interno dei vari nodi, che modellavano l'output del circuito, devono essere eliminate
  - nei circuiti di Mealy gli output dipendono infatti non solo dallo stato ma anche dall'input
- nelle etichette sugli archi possiamo distinguere 2 componenti distinte: INP / OUT
  - INP : configurazione dell'input al tempo t_i
  - OUT : configurazione dell'output del circuito al tempo t_i

Automi per specificare circuiti di Mealy

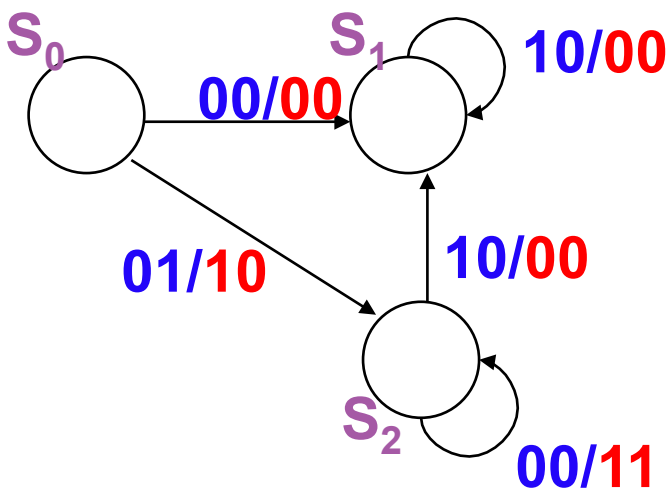
Componente INP dell'etichetta di ogni arco rappresenta una particolare configurazione degli input, e permette la specifica della funzione NEXT_STATE

- Nodo di partenza:
  Stato al tempo t_i
- Etichetta INP:
  Input al tempo t_i
- Nodo di arrivo (NEXT_STATE):
  Stato al tempo t_{i+1}



Componente OUT dell'etichetta di ogni arco rappresenta una particolare configurazione degli output, e permette la specifica della funzione
OUTPUT al tempo t_i

Automa di Mealy e Tabelle di verità



Stato	s_1	s_0
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

NEXT_STATE

	s_1	s_0	I_1	I_0	s_1	s_0	
S_0	0	0	0	0	0	1	S_1
	0	0	0	1	1	0	S_2
	0	0	1	0	X	X	-
	0	0	1	1	X	X	-
S_1	0	1	0	0	X	X	-
	0	1	0	1	X	X	-
	0	1	1	0	0	1	S_1
	0	1	1	1	X	X	-
S_2	1	0	0	0	1	0	S_2
	1	0	0	1	X	X	-
	1	0	1	0	0	1	S_1
	1	0	1	1	X	X	-
S_3	1	1	X	X	X	X	-

OUTPUT

	s_1	s_0	I_1	I_0	O_1	O_0
S_0	0	0	0	0	0	0
	0	0	0	1	1	0
	0	0	1	0	X	X
	0	0	1	1	X	X
S_1	0	1	0	0	X	X
	0	1	0	1	X	X
	0	1	1	0	0	0
	0	1	1	1	X	X
S_2	1	0	0	0	1	1
	1	0	0	1	X	X
	1	0	1	0	0	0
	1	0	1	1	X	X
S_3	1	1	X	X	X	X

Stato S_3 non significativo

- nelle tabelle NEXT_STATE e OUTPUT, la configurazione (s_1 s_0 = 1 1) può essere combinata con qualsiasi altro valore in input (DON'T CARE), per produrre qualsiasi valore in output

Sintesi di un circuito di Mealy

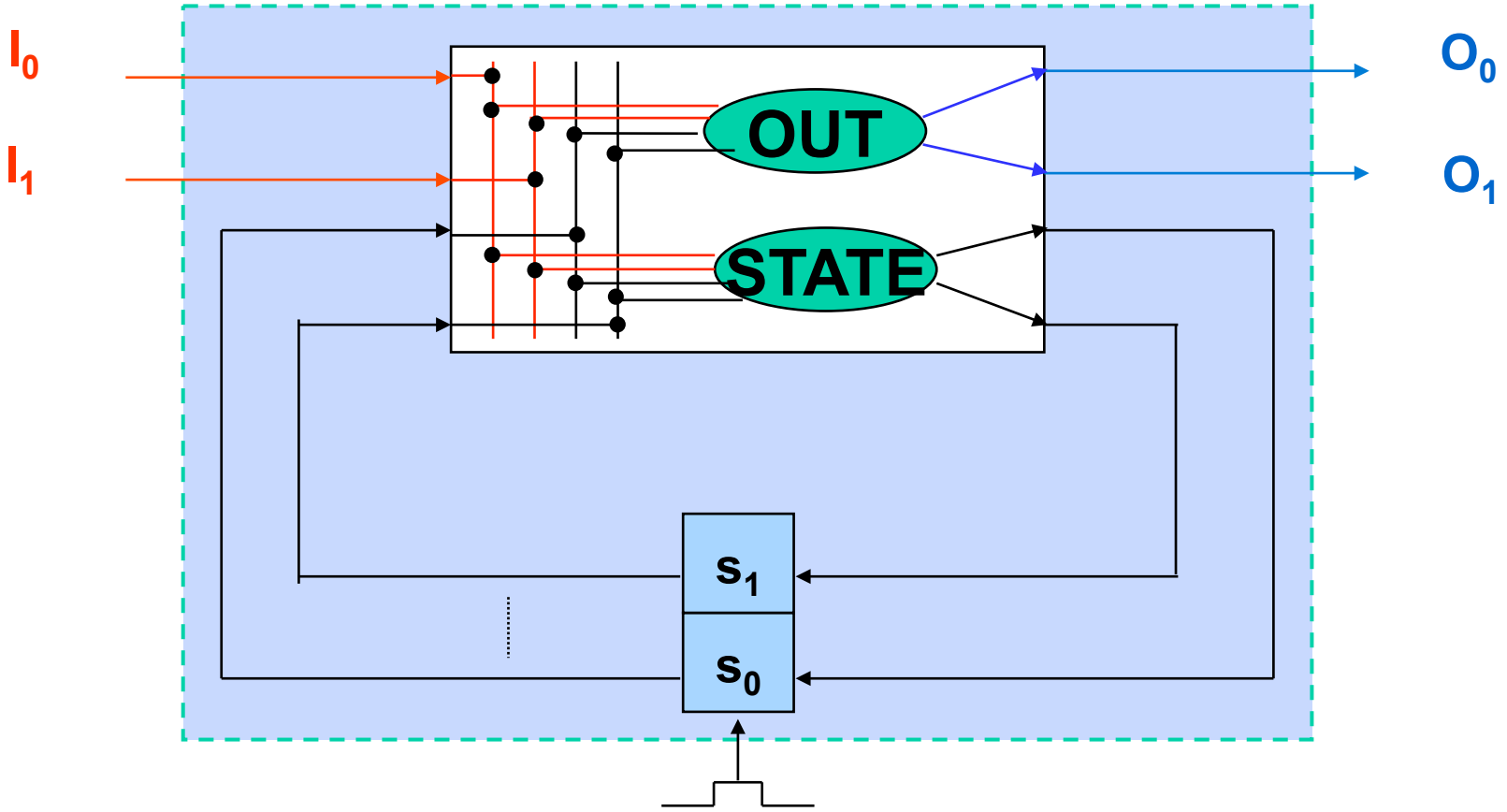
Deriviamo mappe di Karnaugh che generano NEXT_STATE

- due mappe, una per ognuna delle 2 variabili s_0 e s_1
- equazione logica in forma SP ottimizzata, e circuito logico (STATE) a 2 livelli

Deriviamo le mappe di Karnaugh che generano OUTPUT

- due mappe, una per ognuna delle 2 variabili O_0 e O_1
- equazione logica in forma SP ottimizzata, e circuito logico (OUT) a 2 livelli

Sintesi di un circuito di Mealy



Esempio di circuito di Mealy

Si consideri un circuito sequenziale di Mealy che riceve in ingresso una sequenza di bit, all'interno della quale deve riconoscere se le varie sottosequenze di 3 bit hanno un numero pari o dispari di bit uguali ad 1.

Le sottosequenze considerate non si sovrappongono.

Ogni qualvolta il circuito arriva a leggere il terzo bit di ogni sottosequenza deve restituire il valore P o D, in base al numero di bit uguali ad 1 all'interno sottosequenza appena letta (P per pari, D per dispari).

L'output in corrispondenza di tutte le altre posizioni della sequenza deve essere N.

Determinare l'automa a stati finiti che modella il funzionamento del circuito, usando i valori simbolici per gli output.

Codificare poi l'output e gli stati, determinare le tabelle di verità, minimizzarle e ricavare il circuito finale.

Esempio di circuito di Mealy

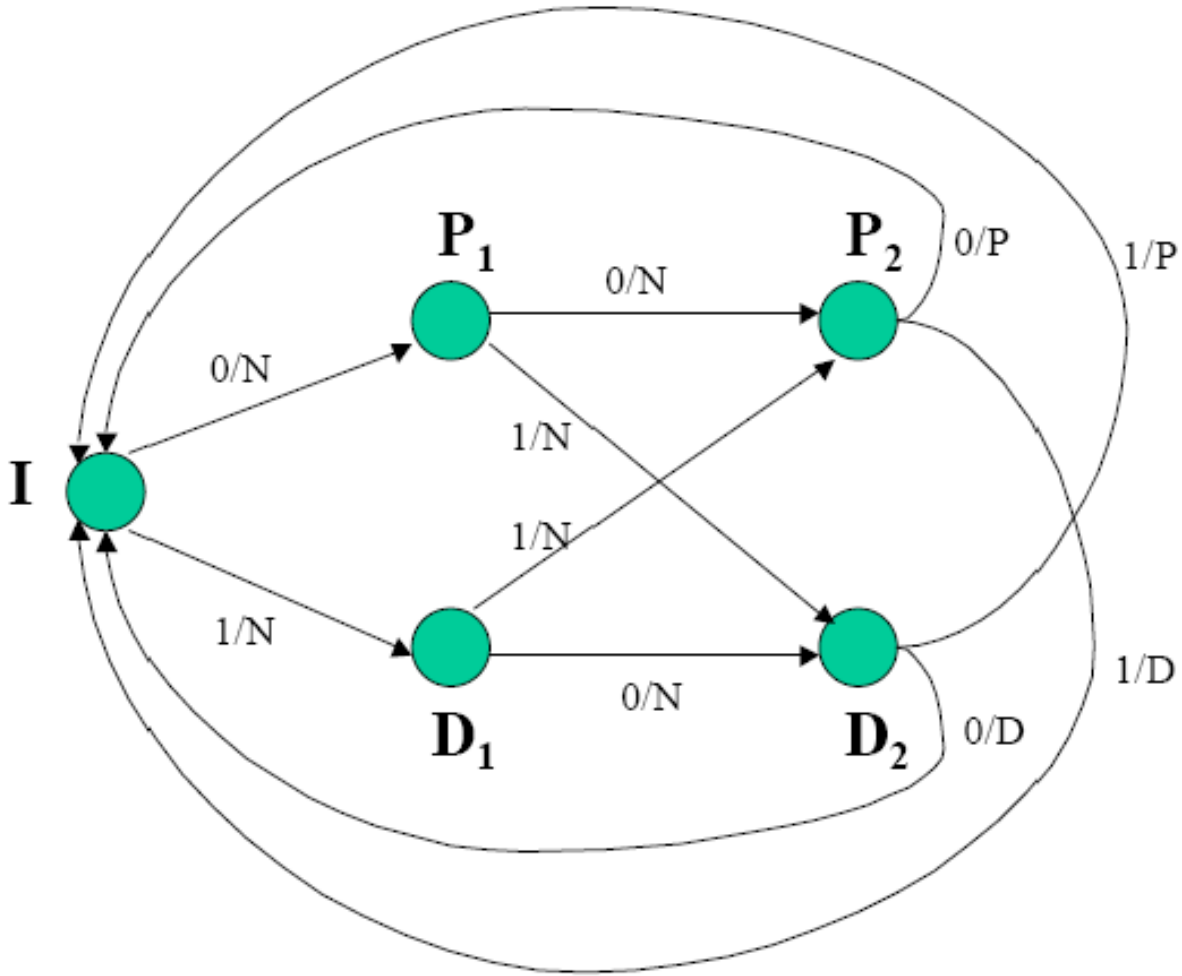
L'automa ha:

- uno stato iniziale I, che viene raggiunto ogni qualvolta si termina la lettura di una sottosequenza di 3 bit.

- 2 stati, P1 e D1, che corrispondono alla lettura di una sottosequenza lunga 1 bit (rispettivamente con un numero pari o dispari di bit affermati)

- 2 stati, P2 e D2, che corrispondono alla lettura di una sottosequenza lunga 2 bit (rispettivamente con un numero pari o dispari di bit affermati)

Esempio di circuito di Mealy: automa



Esempio di circuito di Mealy: codifica

Stato	s_2 s_1 s_0
I	0 0 0
P1	0 0 1
P2	0 1 0
D1	0 1 1
D2	1 0 0
-	1 0 1
-	1 1 0
-	1 1 1

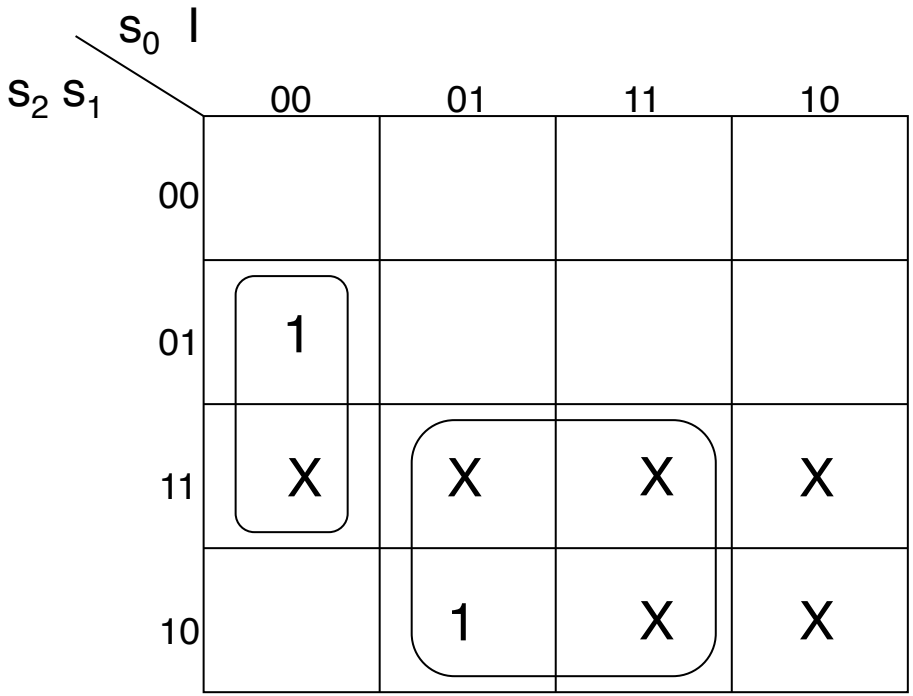
Output	O_1 O_2
N	0 0
D	0 1
P	1 0
-	1 1

Esempio di circuito di Mealy: tabelle

s_2	s_1	s_0	I	O_1	O_2	s_2'	s_1'	s_0'
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	1	0	0
0	1	0	0	1	0	0	0	0
0	1	0	1	0	1	0	0	0
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	0	1	0
1	0	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0	0
1	0	1	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X

Esempio di circuito di Mealy: tabelle

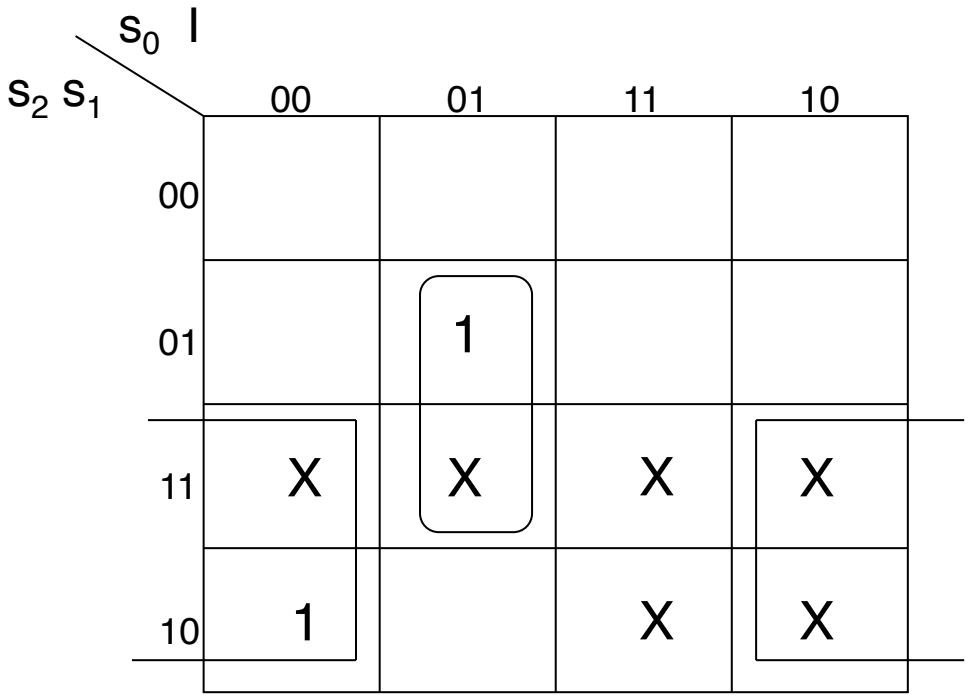
s_2	s_1	s_0	I	O_1	O_2	s_2'	s_1'	s_0'
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	1	0	0
0	1	0	0	1	0	0	0	0
0	1	0	1	0	1	0	0	0
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	0	1	0
1	0	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0	0
1	0	1	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X



$O_1 = s_1 \sim s_0 \sim I + s_2 I$

Esempio di circuito di Mealy: tabelle

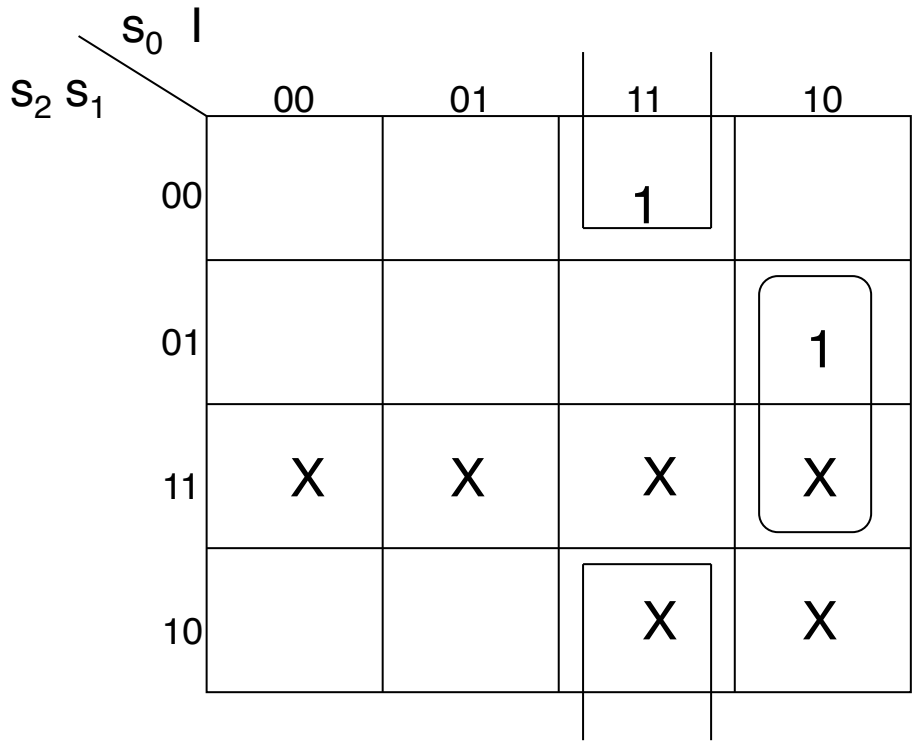
s_2	s_1	s_0	I	O_1	O_2	s_2'	s_1'	s_0'
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	1	0	0
0	1	0	0	1	0	0	0	0
0	1	0	1	0	1	0	0	0
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	0	1	0
1	0	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0	0
1	0	1	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X



$O_2 = s_1 \sim s_0 I + s_2 \sim I$

Esempio di circuito di Mealy: tabelle

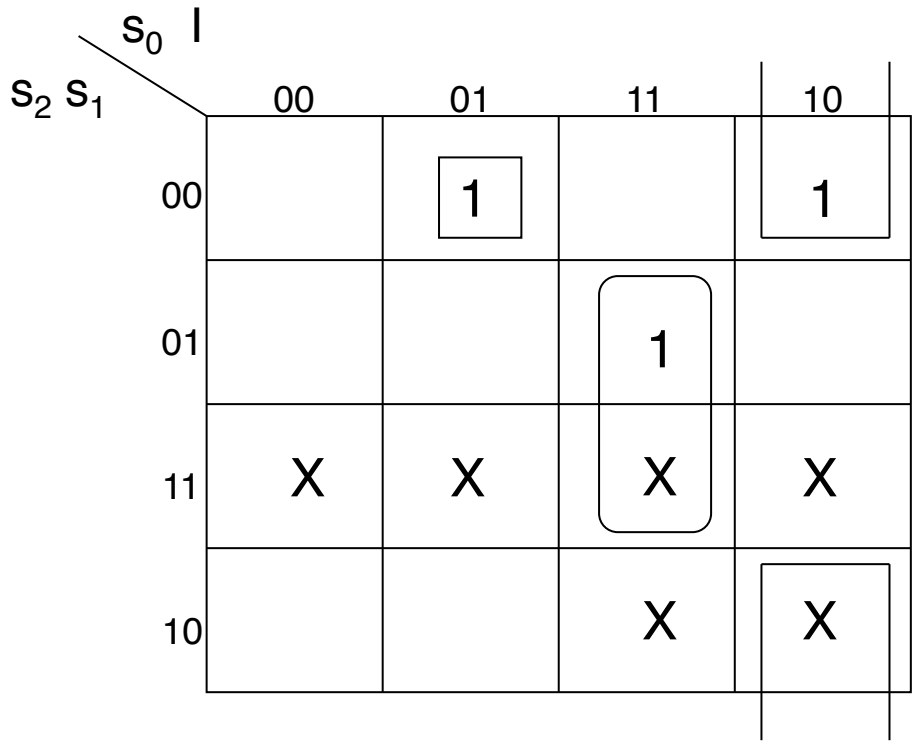
s_2	s_1	s_0	I	O_1	O_2	s_2'	s_1'	s_0'
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	1	0	0
0	1	0	0	1	0	0	0	0
0	1	0	1	0	1	0	0	0
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	0	1	0
1	0	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0	0
1	0	1	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X



$s_2' = s_1 s_0 \sim I + \sim s_1 s_0 I$

Esempio di circuito di Mealy: tabelle

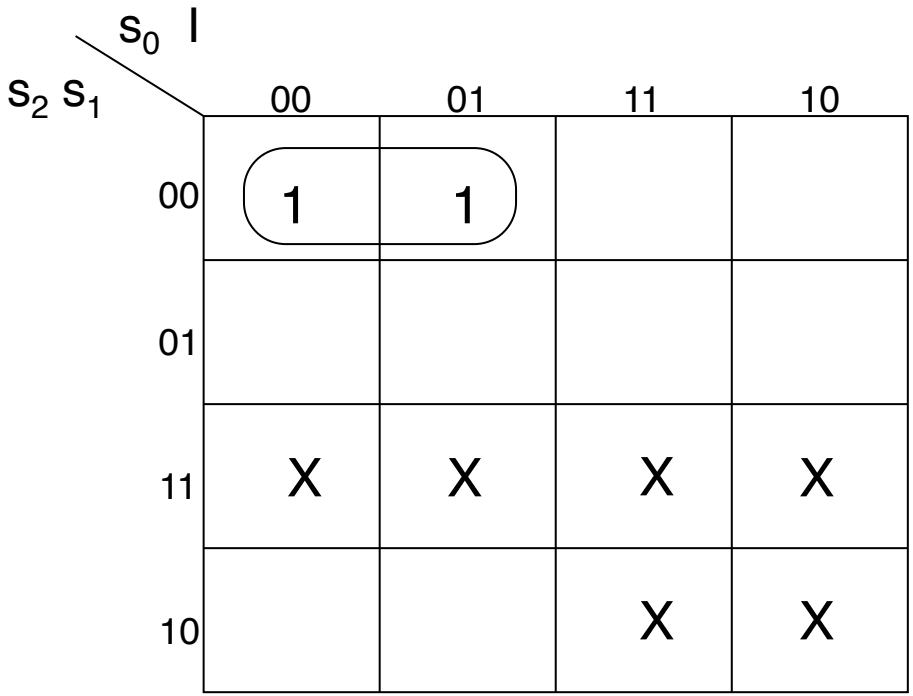
s_2	s_1	s_0	I	O_1	O_2	s_2'	s_1'	s_0'
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	1	0	0
0	1	0	0	1	0	0	0	0
0	1	0	1	0	1	0	0	0
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	0	1	0
1	0	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0	0
1	0	1	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X



$$s_1' = \sim s_2 \sim s_1 \sim s_0 I + s_1 s_0 I + \sim s_1 s_0 \sim I$$

Esempio di circuito di Mealy: tabelle

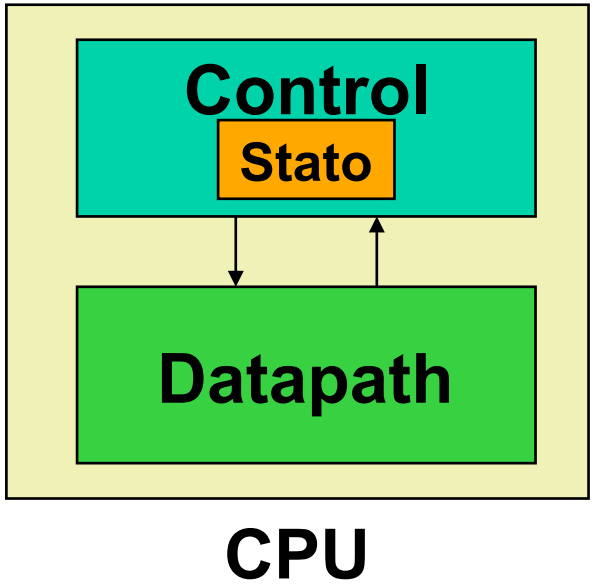
s_2	s_1	s_0	I	O_1	O_2	s_2'	s_1'	s_0'
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	1	0	0
0	1	0	0	1	0	0	0	0
0	1	0	1	0	1	0	0	0
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	0	1	0
1	0	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0	0
1	0	1	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X



$s_0' = \sim s_2 \sim s_1 \sim s_0$

Parte Controllo = Circuito sequenziale

Nel seguito vedremo che la *Parte Controllo* (Control) della CPU è un circuito sequenziale



- istruzioni eseguite in più cicli
  - ad ogni ciclo, si esegue uno *micropasso* (microistruzione) dell'istruzione
  - lo stato interno al circuito sequenziale determina lo specifico micropasso da eseguire
- output della Parte Controllo inviati alla *parte operativa* (Datapath), che li interpreta come *comandi*
  - es.: controlli dei multiplexer, controlli per le ALU, segnali per abilitare la scrittura in registri, ecc.
- gli input della Parte Controllo giungono dal Datapath
  - es.: campi del registro che contiene l'istruzione corrente (IR), risultati di operazioni di confronto, ecc.

Automi vs Microistruzioni

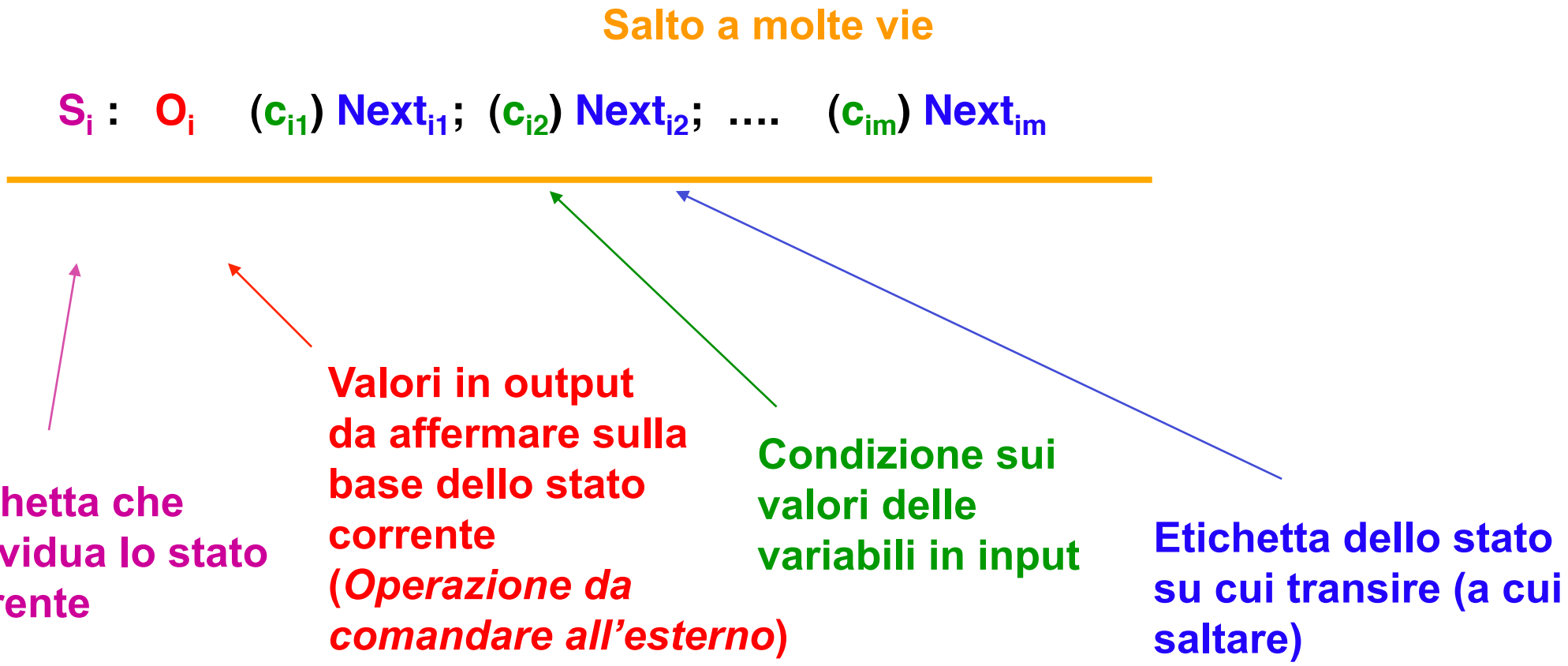
Gli automi a stati finiti costituiscono solo una particolare rappresentazione grafica per descrivere il comportamento, passo per passo, di un circuito sequenziale

Se gli stati sono in numero considerevole =>
diventa *difficile disegnare l'automa*

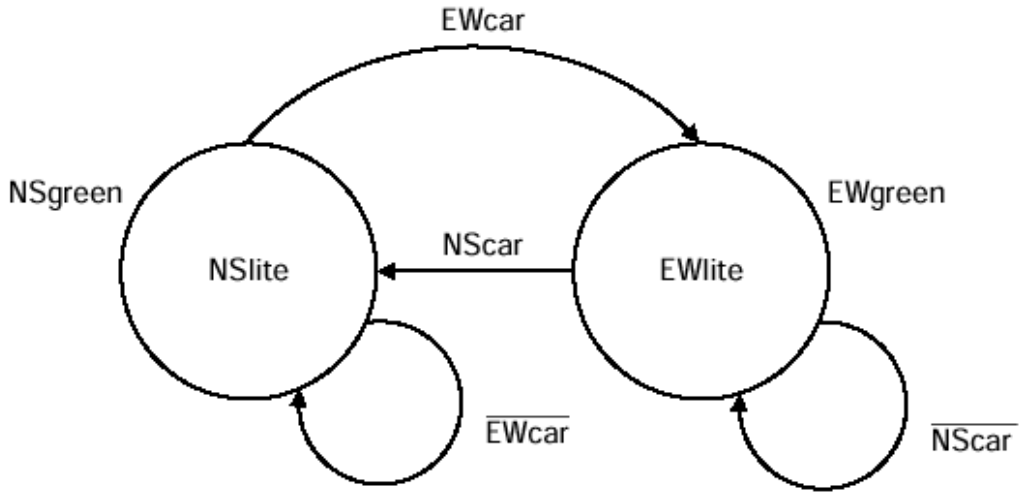
È utile quindi usare un programma (*microprogramma*) scritto con un *linguaggio testuale,* composto da un set definito di istruzioni (*microistruzioni*)

Automi vs Microistruzioni

Sintassi di una *microistruzione* di tipo TS (Transizione di Stato), usata per modellare circuiti di Moore:



Automa vs Microprogramma



NSgreen: NSlite←1, EWlite ←0 case (NScar, EWcar) of

(_,0) : NSgreen ;

(_,1) : EWgreen ;

EWgreen: NSlite←0, EWlite ←1 case (NScar, EWcar) of

(0,_) : EWgreen ;

(1,_) : NSgreen ;

Implementazione alternativa

Per realizzare i circuiti sequenziali, abbiamo già visto la tecnica cosiddetta *cablata*, basata su circuiti combinatori a 2 livelli di logica che determinano

- *NEXT_STATE* e *OUTPUT*

L'uso di *microprogrammi* suggerisce un'implementazione *differente* dei circuiti sequenziali…

Implementazione alternativa

Memorizzare le varie microistruzioni (con formato ben definito) in una ROM

Usare un registro (Stato del circuito), chiamato *sequenzializzatore (micro Program Counter),* per indirizzare la *microistruzione corrente*

Necessaria una rete per determinare il prossimo valore del sequenzializzatore (*logica di selezione* o *sequenzializzazione*), che a sua volta può far uso di ROM per l'implementazione