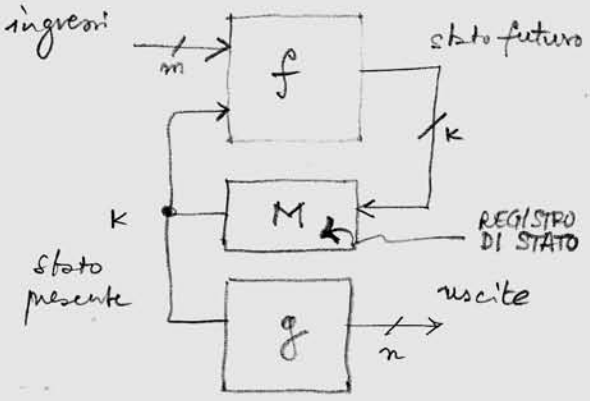
6/11/14 **Il progetto pratico di ASF**

Concentriamoci sulla parte di controllo, che tipicamente è specificata attraverso un diagramma degli stati. Nella pratica, bisogna cercare di sfruttare le peculiarità dell'automa da realizzare.

Lo schema realizzativo "cieco" (ovvero che prescinde dalle caratteristiche specifiche della macchina da progettare) è



Con questo schema, in ogni stato presente viene calcolato lo stato futuro a cui saltare. Inoltre, il salto viene calcolato sulla base di tutti gli ingressi della macchina, anche se solo una parte di essi dovrebbero essere campionati nello stato presente in questione.

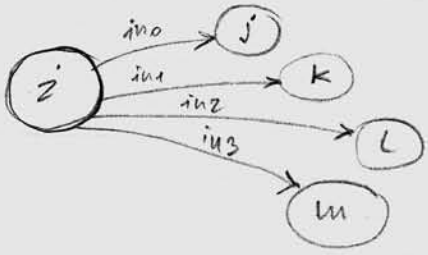
Ora: le peculiarità di un automa reale si possono riassumere come segue

(1) La gran parte delle transizioni di stato avviene in maniera OBBLIGATA, cioè senza dover campionare alcun ingresso esterno. In particolare, molte delle transizioni obbligate sono tra due stati con CODIFICHE BINARIE successive, ossia

Queste speciali transizioni potrebbero essere calcolate in modo del tutto naturale con un CONTATORE al posto del REGISTRO DI STATO.

Riguardo alle transizioni non successive, ossia queste sono equivalenti a ciò che in programmazione si chiamano SALTI INCONDIZIONATI ("goto").

Gli altri tipi di salto li chiameremo SALTI CONDIZIONATI: essi avvengono sulla base del valore di <u>alcuni</u> ingressi esterni:

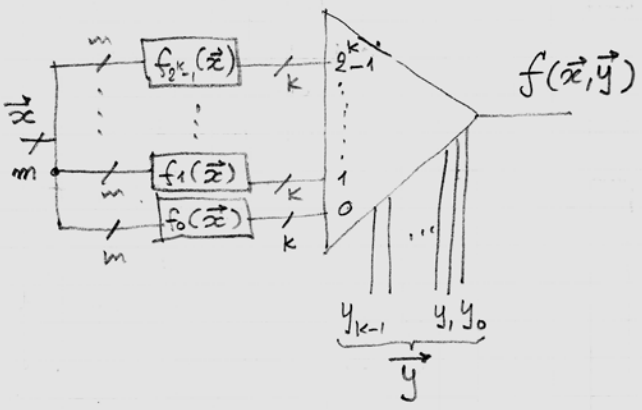


Nota: il # di ingressi binari da campionare per eseguire una scelta tra P stati di salto possibili è $\lceil \log_2 P \rceil$ (nell'es. è 2)

(2) Dato che, come già osservato sopra, non tutti gli m ingressi di una macchina devono essere valutati contemporaneamente (cosa nel moderno stato presente) per determinare la transizione di stato, la funzione

$$f(x_{m-1}, x_{m-2}, \dots x_2, x_1, x_0; y_{k-1}, \dots y_1, y_0)$$

può essere convenientemente progettata con un multiplexer vettoriale che abbia come ingressi di selezione i bit di stato:



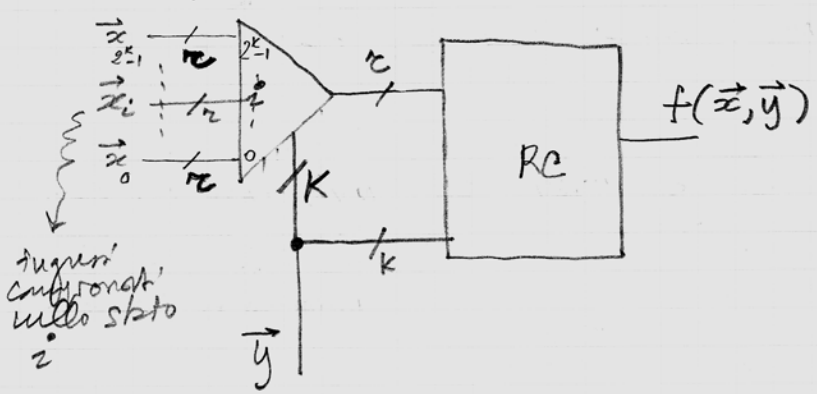
In questo modo:

a) si devono progettare 2^k funzioni combinatorie che hanno al massimo m variabili, e non $m+k$ variabili

b) per quanto sopra detto, le funzioni $f_i(\vec{x})$ in realtà dipenderanno spesso da un piccolo sottoinsieme degli ingressi $(x_{m-1}, \dots x_0)$, con ciò semplificando il progetto. Addirittura, nel caso di TRANSIZIONI OBBLIGATE, le funzioni saranno costanti, valendo lo stato di salto.

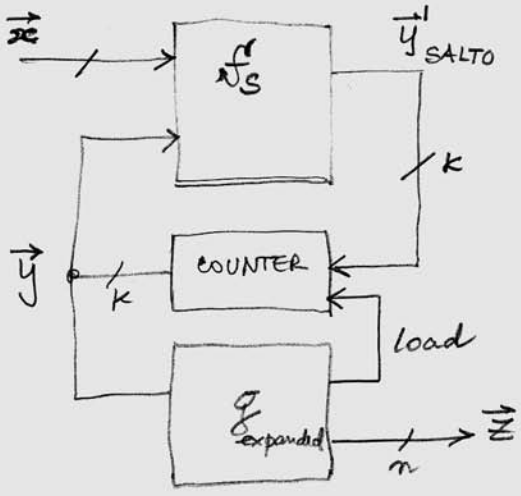
i = stato presente

Ora: supponiamo che il massimo numero di bit di ingresso campionati in uno stato sia $0 \le r \le m$. Allora lo schema sopra si potrebbe ridisegnare come:



la semplificazione è che il blocco combinatorio è funzione di solo $r+k$ (anziché $m+k$) ingressi.

Vediamo ora un primo schema che sfrutta il contatore al posto del registro di stato:



La funzione f_S dipende sempre da $\vec{x}$ e $\vec{y}$ (come ξ), ma ha in realtà moltissimi DON'T CARE, corrispondenti a tutti gli stati presenti in cui

(a) non si compromette ingresso

(b) lo stato futuro è incrementato di 1 rispetto a quello presente

Quindi

$$\vec{y}' = \begin{cases} \vec{y}'_{COUNTER} \equiv \vec{y} + 1 \pmod{2^k} \\ \vec{y}'_{SALTO} \end{cases}$$

Se $\vec{y} \to \vec{y}'$ è una TRANSIZIONE OBBLIGATA INCREMENTALE

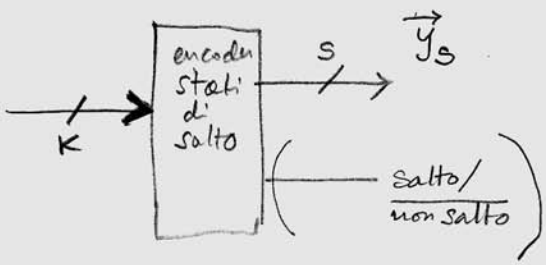
Se $\vec{y} \to \vec{y}'$ è una TRANSIZIONE DI SALTO, ovvero una TRANSIZIONE OBBLIGATA NON INCREMENTALE (Salto incondizionato) oppure una TRANSIZIONE CONDIZIONATA (o salto condizionato) (incrementale o non incrementale)

(N.B. Assumiamo che, se da un certo SP si può andare in p stati futuri, uno dei quali è SP+1, allora anche quest'ultimo è considerato un salto). (*)

(*) per saltare solo quando gli stati non sono successivi, bisogna far dipendere il LOAD da $\vec{x}$, e dunque realizzarlo in f_S e non in $g_{expanded}$.

Ad ogni modo, è evidente che f_S potrebbe essere fatta dipendere da un numero di variabili inferiore - infatti, se Y_S è il numero di stati di salto, allora questi ultimi sono codificabili con $0 \le S \le k$ bit, con un encoder $k \to S$:

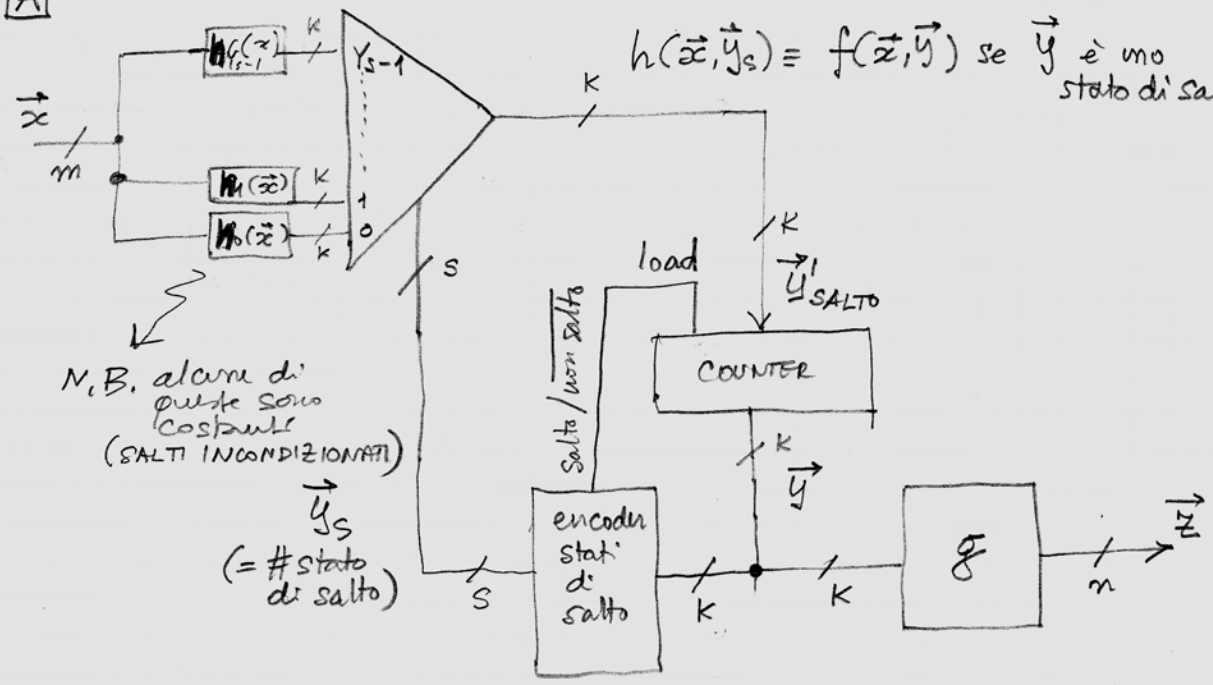
$= \lceil \log_2 Y_S \rceil$



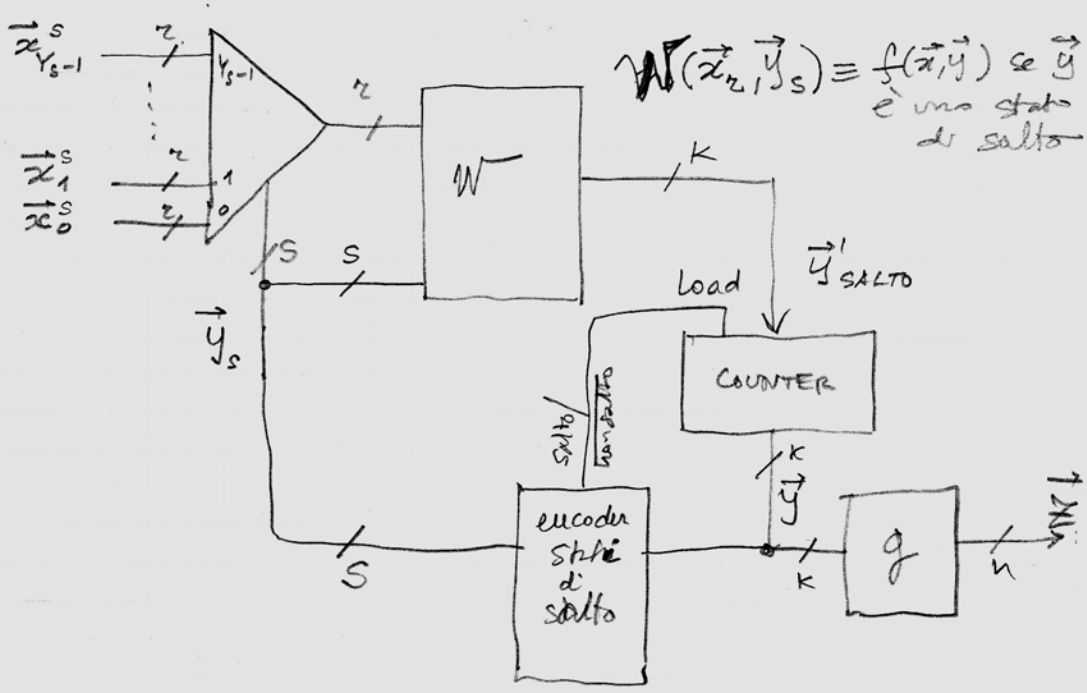
ciò equivale a NUMERARE GLI STATI DI SALTO, ASSOCIANDO A CIASCUNO DI ESSI UN NUMERO PROGRESSIVO

Due Schemi pratici per il progetto sono dunque i seguenti =

A

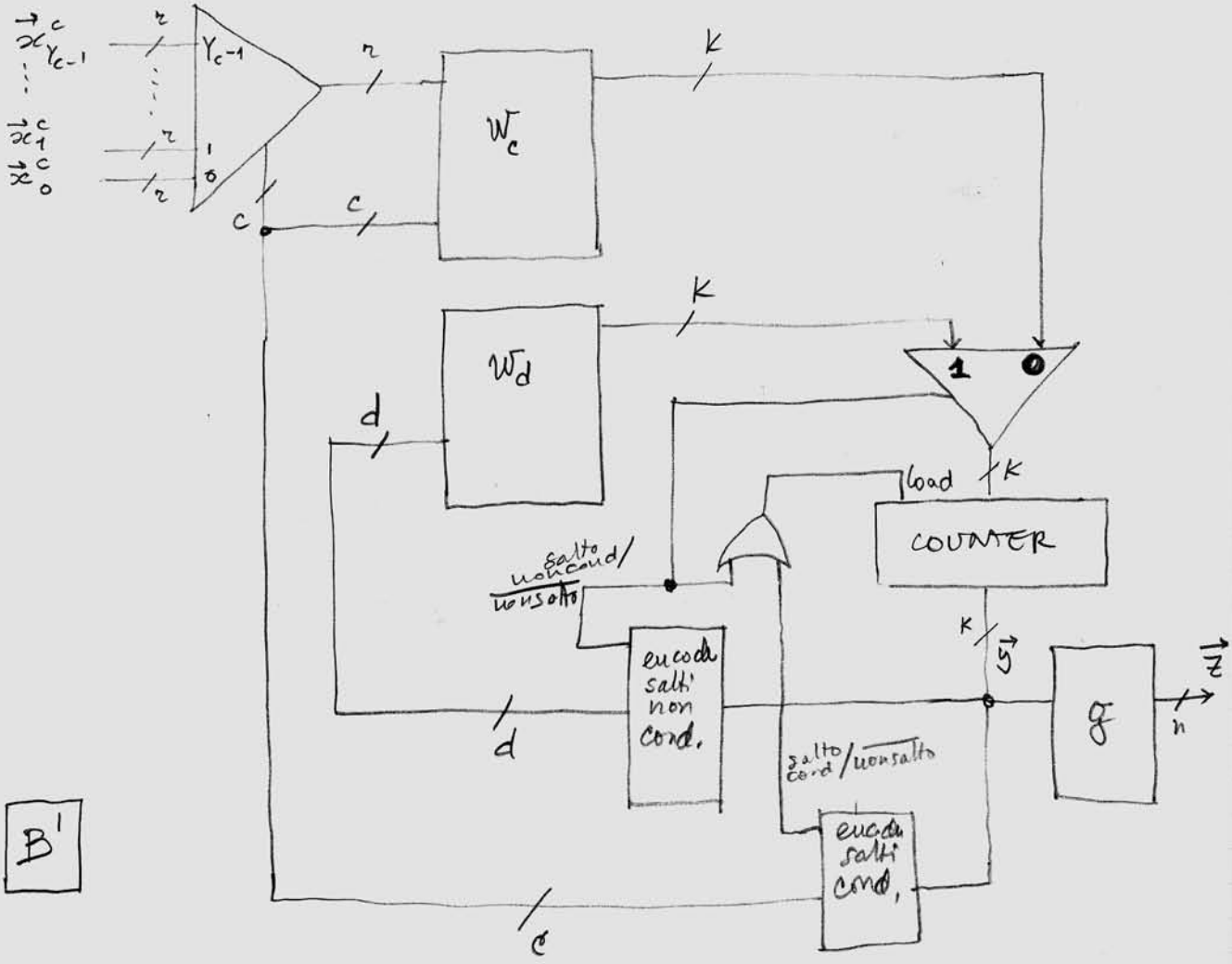


B



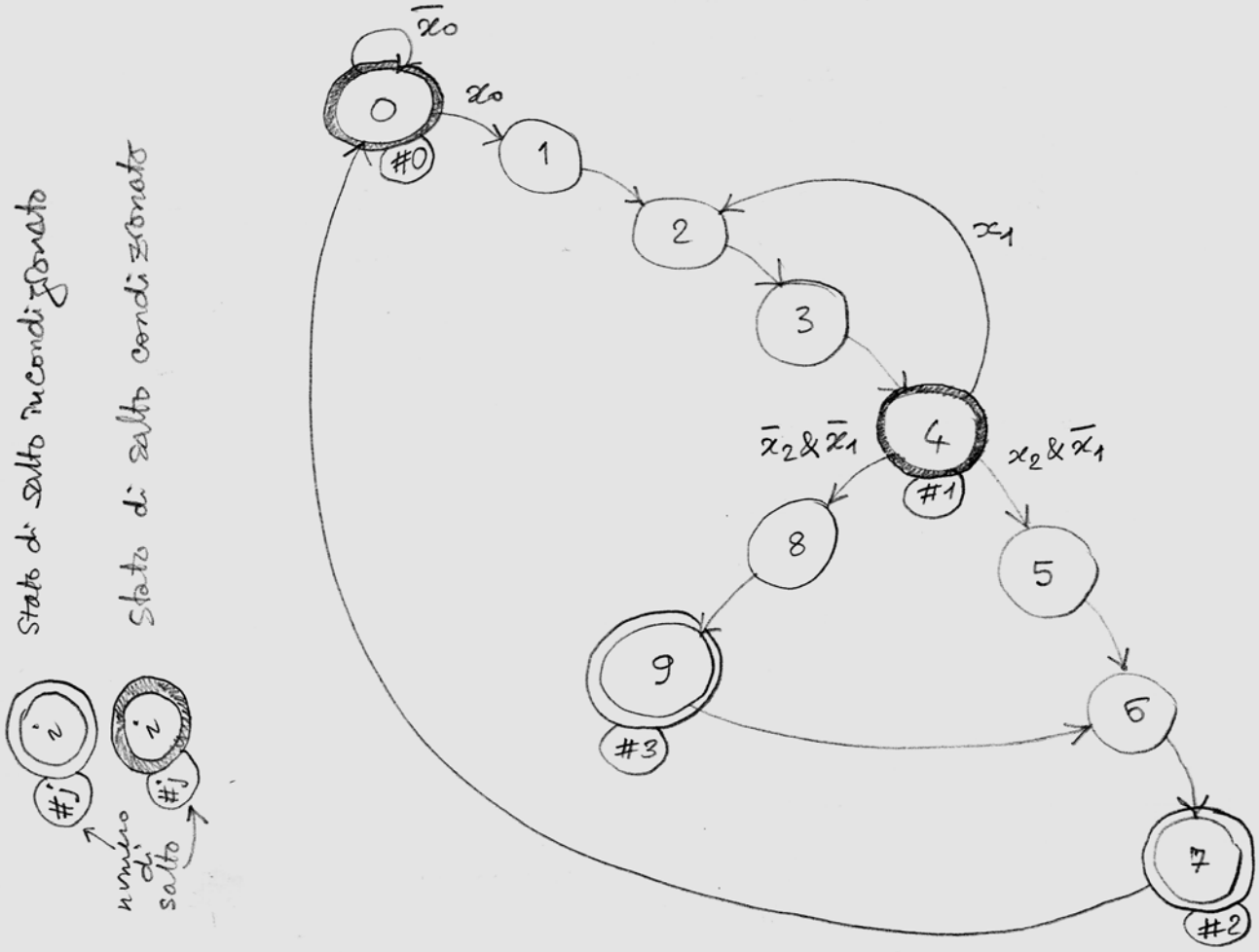
B

NOTA. Lo schema ha ancora un difetto: non si fa distruzione tra salti CONDIZIONATI e INCONDIZIONATI, col risultato che fra questi ultimi gli ingressi della tabella della funzione W sono in sostanza ripetuti - Questo è uno spreco, soprattutto se la W è realizzata con una ROM (la cui dimensione RADDOPPIA con ogni nuovo ingresso). Per rimediare a questo inconveniente, si può fare una CODIFICA SEPARATA degli stati di salto condizionati e incondizionati



B'

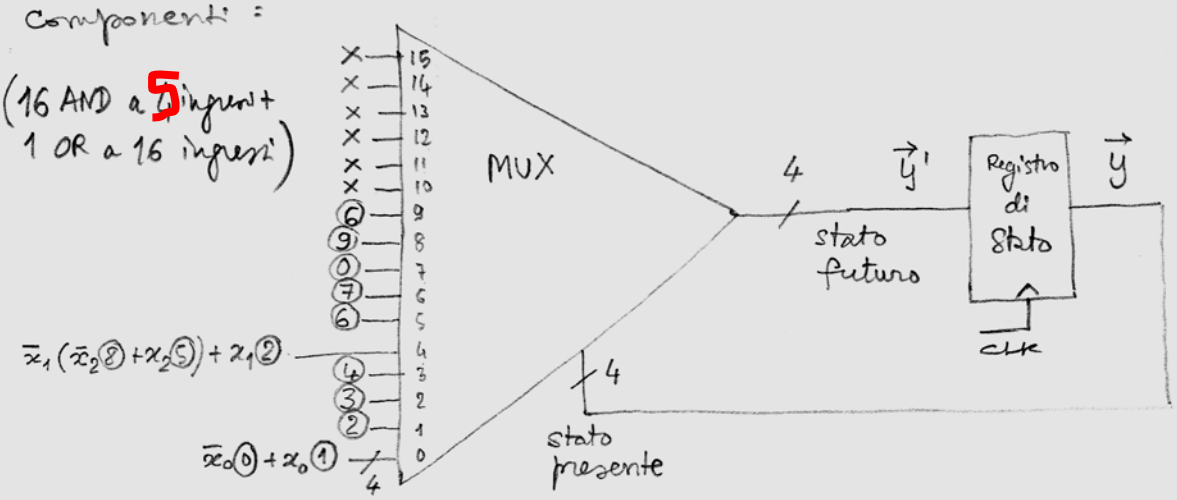
Esempio. Sintetizziamo il seguente automa:



La sintesi monoblocco standard richiederebbe di costruire una tabella con 3 (ingressi) + 4 (bit di stato) ⇒ 128 righe.

Usando lo stato come ingresso di selezione del multiplexer dovremmo calcolare $2^4 = 16$ diverse funzioni vettoriali di 4 componenti:

4 × (16 AND a 5 ingressi + 1 OR a 16 ingressi)



Codificando gli stati di salto (e usando un contatore al posto del registro di stato) abbiamo:

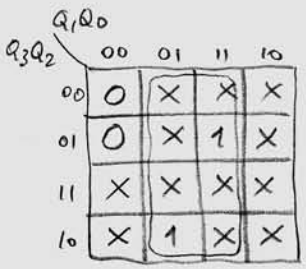
STATO DI SALTO	CODIFICA $S_1\ S_0$	INGRESSI CAMPIONATI	LEGGE DI TRANSIZIONE
(0)	0 0	x_0	$\bar{x}_0$ (0) + x_0 (1)
(4)	0 1	x_1, x_2	$\bar{x}_1(\bar{x}_2$ (8) + x_2 (5)) + x_1 (2)
(7)	1 0	—	(0)
(9)	1 1	—	(6)

x_0	$Q'_3\ Q'_2\ Q'_1\ Q'_0$
0	0 0 0 0
1	0 0 0 1
	↓ ↓ ↓ ↓
	0 0 0 x_0

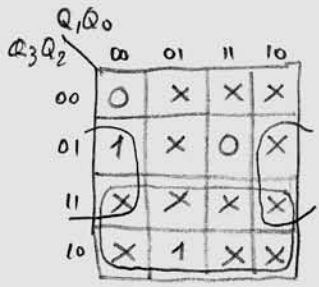
$x_2\ x_1$	$Q'_3\ Q'_2\ Q'_1\ Q'_0$
0 0	1 0 0 0
0 1	0 0 1 0
1 0	0 1 0 1
1 1	0 0 1 0
	↓ | ↓ ↓
	$\bar{x}_2\bar{x}_1$ $x_2\bar{x}_1$ x_1 $x_2\bar{x}_1$

Vediamo ora l'encoder:

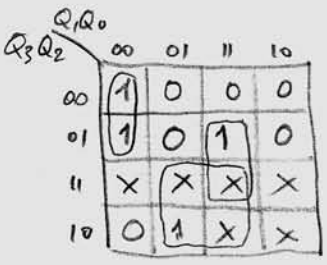
$Q_3\ Q_2\ Q_1\ Q_0$	S_1	S_0	Load
0 0 0 0	0	0	1
0 0 0 1	×	×	0
0 0 1 0	×	×	0
0 0 1 1	×	×	0
0 1 0 0	0	1	1
0 1 0 1	×	×	0
0 1 1 0	×	×	0
0 1 1 1	1	0	1
1 0 0 0	×	×	0
1 0 0 1	1	1	1
1 0 1 0	×	×	×
1 0 1 1	×	×	×
1 1 0 0	×	×	×
1 1 0 1	×	×	×
1 1 1 0	×	×	×
1 1 1 1	×	×	×



$S_1 = Q_0$



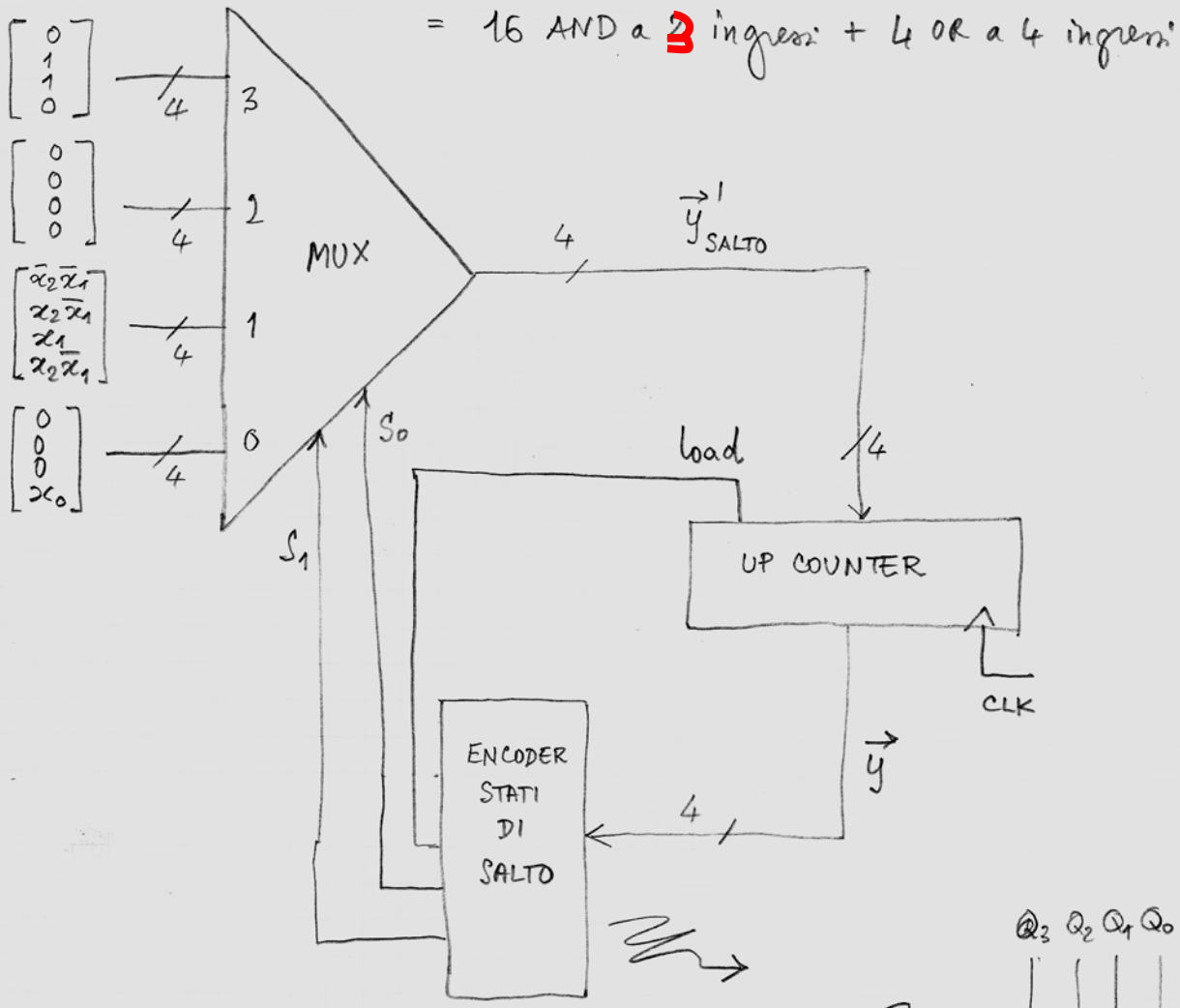
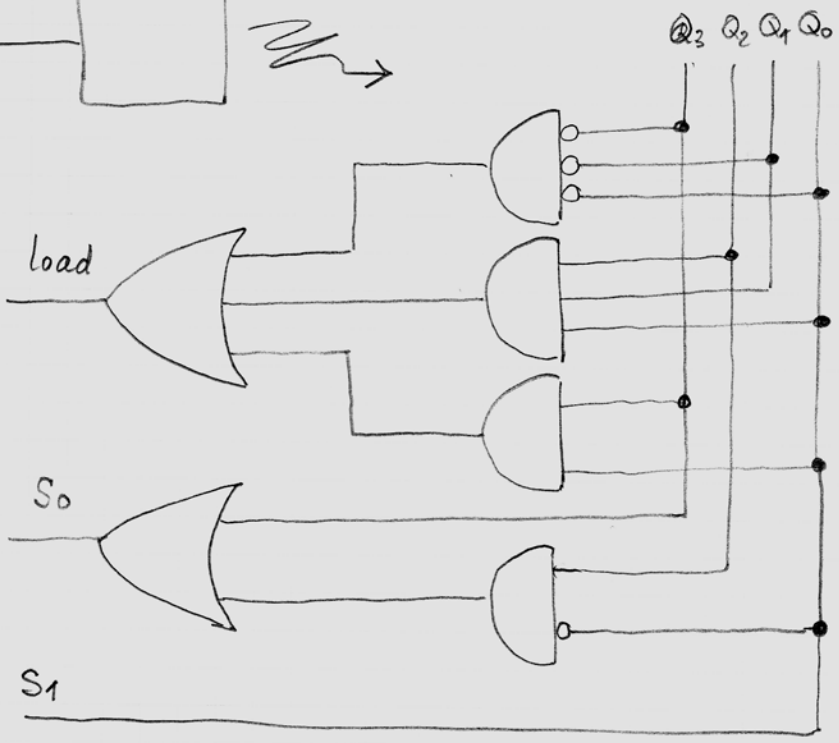
$S_0 = Q_3 + Q_2\bar{Q}_0$



$Load = \bar{Q}_3\bar{Q}_1\bar{Q}_0 + Q_2Q_1Q_0 + Q_3Q_0$

$4 \times (4$ AND a 3 ingressi $+ 1$ OR a 4 ingressi$) =$

$= 16$ AND a 2 ingressi $+ 4$ OR a 4 ingressi



(2 AND a 3 ingressi +
2 AND a 2 ingressi +
1 OR a 3 ingressi +
1 OR a 2 ingressi)