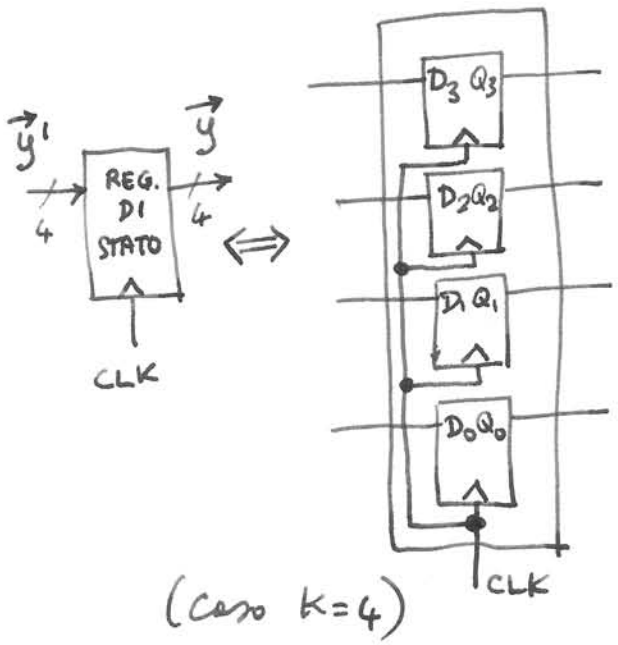
Registri

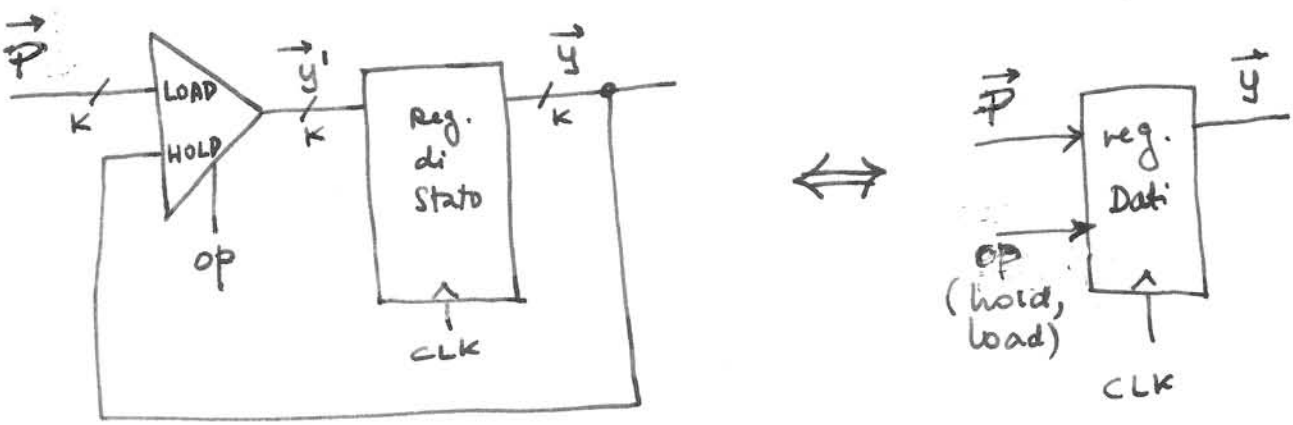
Abbiamo visto che il blocco M di una macchina sequenziale (registro di stato) è composto da K flip-flop D:



(caso K=4)

Notare come i flip-flop siano indipendenti: ognuno amministra un bit di stato. Le sole operazioni che si possono fare con questo tipo di registro sono SET ($Q'_i = D_i = 1$) o RESET ($Q'_i = D_i = 0$). NON C'E' HOLD.

Il <u>registro di stato</u> può essere trasformato in un <u>registro dati</u> se si aggiungono le funzionalità standard di HOLD e CARICAMENTO PARALLELO (o LOAD). Nel caso dell'hold, lo stato viene reimmesso in ingresso.



Nota: op è la versione <u>codificata</u> di LOAD/HOLD. Trattandosi di operazioni mutuamente esclusive, basta un bit. Ad es. op = 0 → hold, op = 1 → load.

%

L'espressione caratteristica di un registro dato è dunque

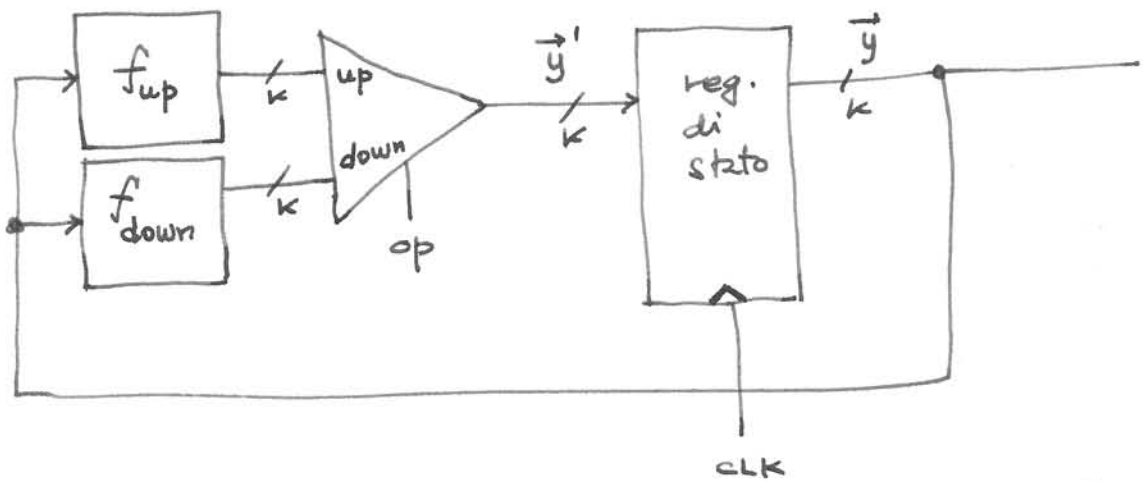
$$\vec{y}' = hold \cdot \vec{y} + load \cdot \vec{p} \quad ,$$

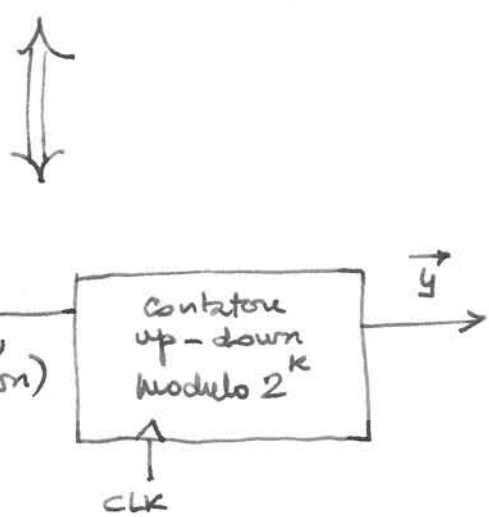
Con $hold = \overline{op}$ e $load = op$.

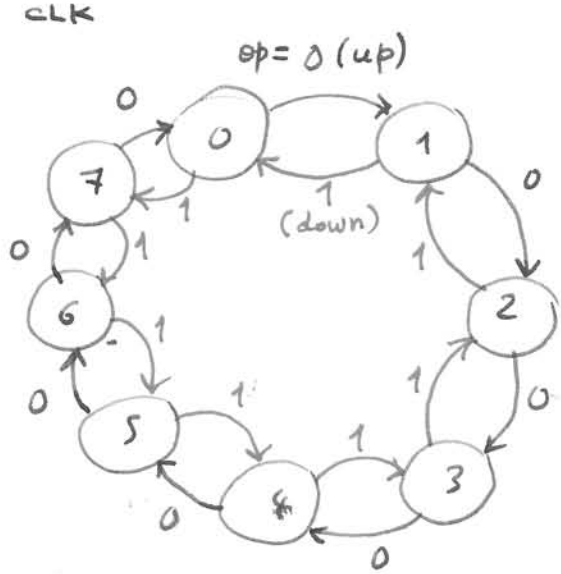
Un altro esempio di registro è il <u>registro contatore</u>.

Abbiamo già fatto il progetto di un contatore up-down.

Rivediamolo:







Ep. caratteristica:

$$\vec{y}' = up \cdot f_{up}(\vec{y}) + down \cdot f_{down}(\vec{y})$$

Con $f_{up}(\vec{y}) = \vec{y}$ "+" $1 \pmod{2^k}$ e $f_{down}(\vec{y}) = \vec{y}$ "−" $1 \pmod{2^k}$

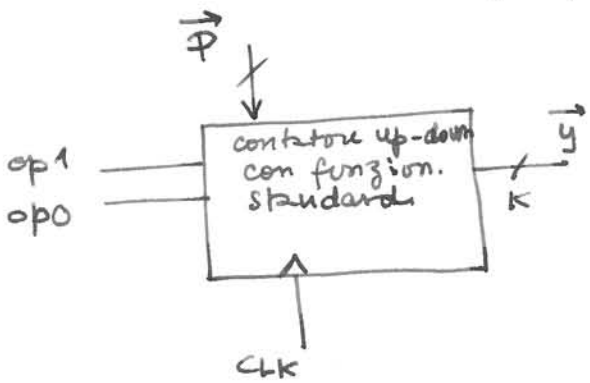
e $up = \overline{op}$, $down = op$.

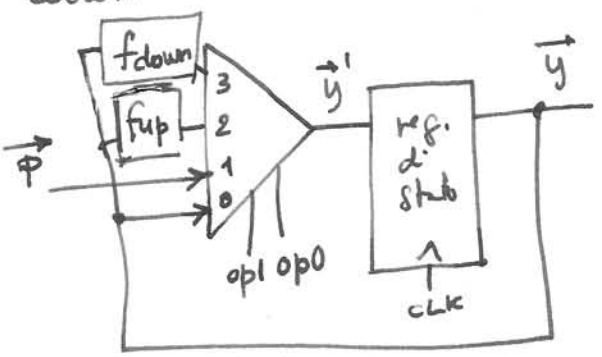
Possiamo estendere la funzionalità del contatore up-down, dotandolo delle funzionalità standard di HOLD e LOAD:

$$\vec{y}' = hold \cdot \vec{y} + load \cdot \vec{p} + up \cdot f_{up}(\vec{y}) + down\ f_{down}(\vec{y})$$

con (ad es.)

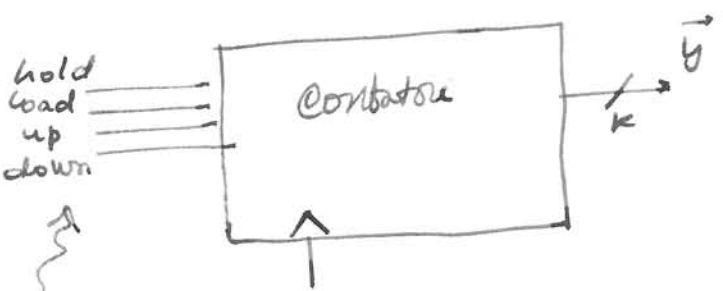
op_1	op_0	funzionalità (tutte mutuamente esclusive)
0	0	Hold
0	1	Load
1	0	up
1	1	down





NOTA sulla codifica delle operazioni.

Molto spesso, per chiarezza, eviteremo di usare la versione codificata dei controlli, preferendo invece usare la notazione esplicita sulle "scatole" dei registri. Ad es. più avremmo potuto scrivere

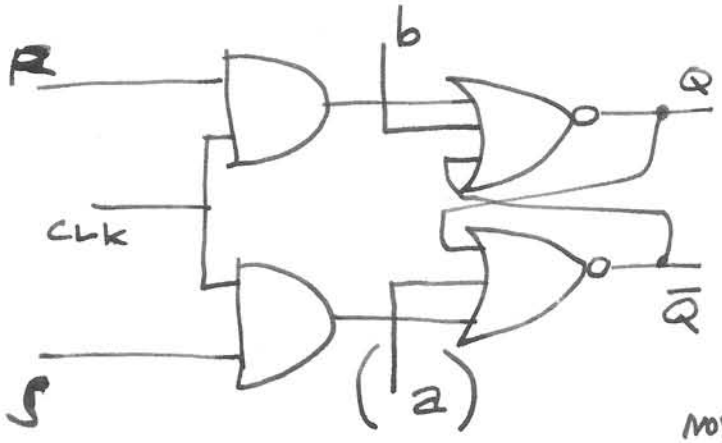


controlli esterni per il contatore: questi segnali si intendono MUTUAMENTE ESCLUSIVI (solo 1 asserito per volta)

Inizializzazione dei registri.

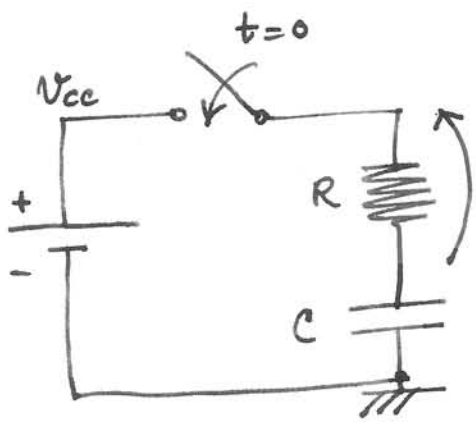
→ a qualsiasi valore $\vec{P}$

I registri possono essere inizializzati in modo sincrono con la funzionalità LOAD. Essi possono anche essere azzerati in modo asincrono (cioè, a valle del clock, e dunque indipendentemente dal livello alto o basso dello stesso attraverso il segnale CLEAR (CLR):



a	b	mode
0	0	transparent
0	1	CLEAR
1	0	PRESET
1	1	forbidden

NOTA: Per il solo azzeramento, ignorando dunque la possibilità di PRESET a 1, si può omettere a e ~~Si fatto~~ porre b = CLEAR. A funzionamento normale, b = 0 non influisce sullo stato.

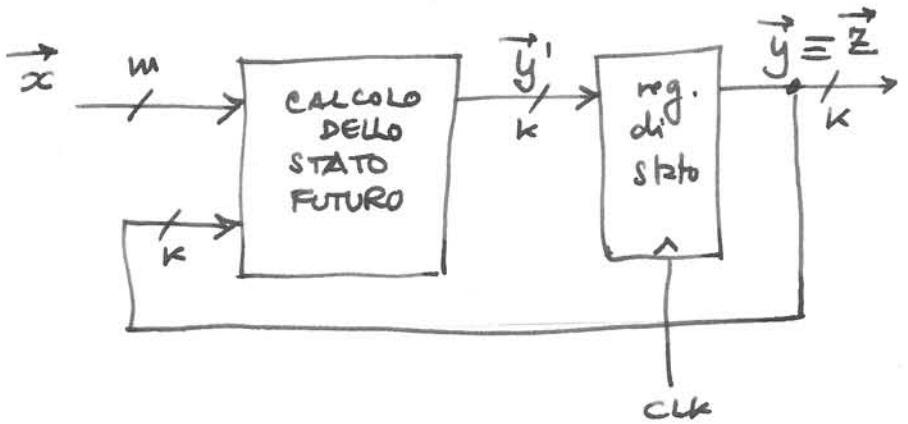


$$b(t) = V_{cc}\, e^{-t/RC}$$

$$b(0) = V_{cc}$$

$$b(t \to \infty) = 0$$

NOTA. I registri visti (e anche quelli che vedremo in seguito) hanno tutti la caratteristica (come i flip-flop) di avere $g(\vec{y}) = \vec{y}$, ossia lo stato si vede direttamente in uscita. Lo schema di un qualsiasi registro è dunque, in generale:

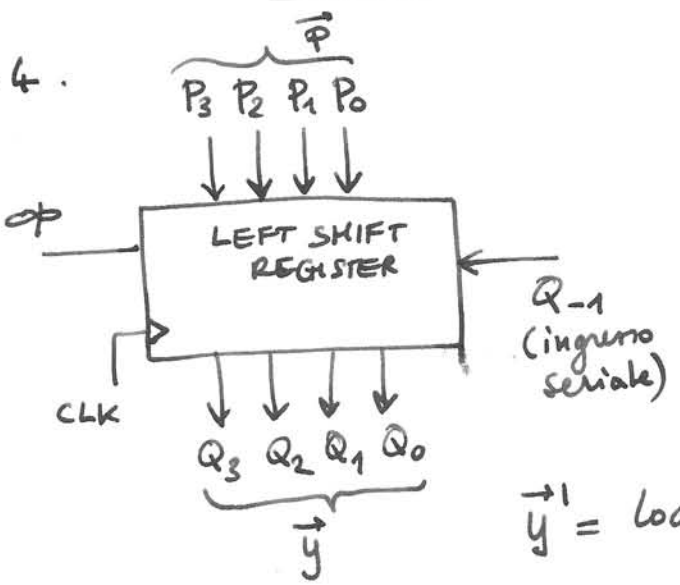


Abbiamo visto:

- registro di stato
- registro dati
- registro contatore (modulo 2^k)

Vediamo ora il Registro a scorrimento (shift register).

Es.: $k = 4$.

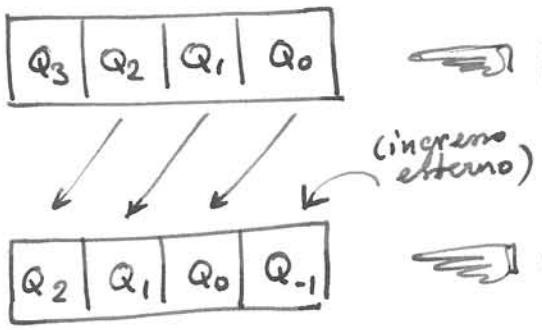


questo è un LEFT SHIFT REGISTER, con funzionalità di LOAD e SHIFT:

$$\vec{y}' = load \cdot \vec{P} + shift\, f_{SHL}(\vec{y}, Q_{-1})$$

%

Funzionamento:



prima dello scorrimento a sinistra

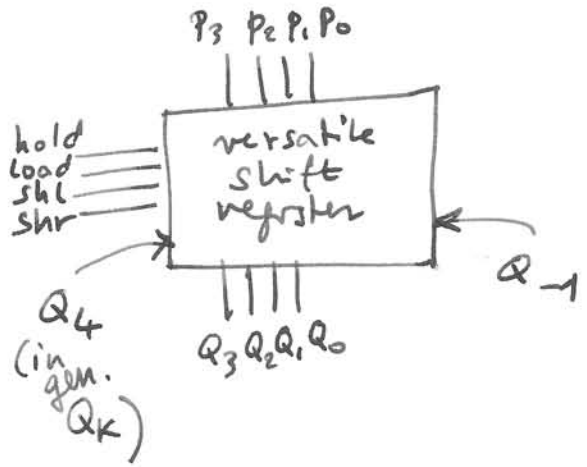
dopo lo scorrimento.

Il valore Q_3 si perde, i valori Q_2, Q_1 e Q_0 cambiano posto, il bit Q_{-1} entra da destra

op	Q'_3	Q'_2	Q'_1	Q'_0
load	P_3	P_2	P_1	P_0
shift (left)	Q_2	Q_1	Q_0	Q_{-1}

Stessa cosa per lo scorrimento a destra.
Costruiamo ora uno shift register che scorre sia a ~~sx~~ che a ~~dx~~, e ha in più load e hold:

op	Q'_3	Q'_2	Q'_1	Q'_0
hold	Q_3	Q_2	Q_1	Q_0
load	P_3	P_2	P_1	P_0
shL	Q_2	Q_1	Q_0	Q_{-1}
shr	Q_4	Q_3	Q_2	Q_1



NOTA: Lo shift a sin realizza (a meno di overflow) la moltiplicazione per 2, mentre lo shift a destra realizza la divisione INTERA (%) per 2.

Ancora sui contatori

Consideriamo ora un contatore modulo $n \neq 2^k$.
Ad es. $n = 5$. Dovremo comunque usare $k = 3$ bit di stato.
Facciamo per esercizio il progetto di un contatore up modulo 5, usando la sintesi "monoblocco".

automa:

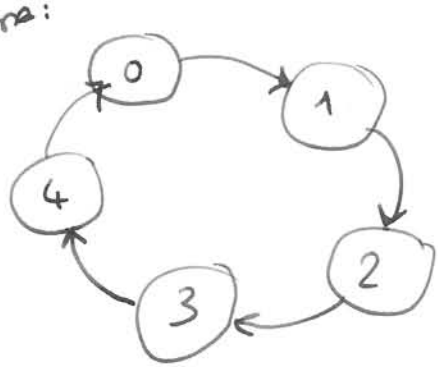
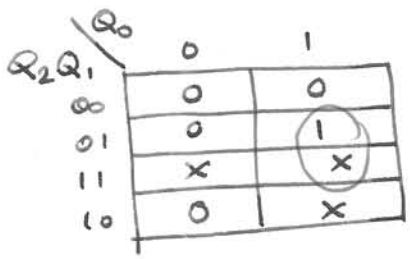


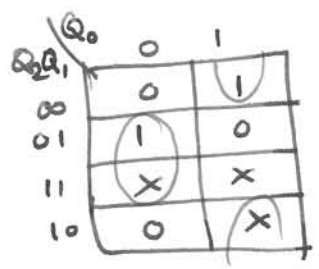
tabella:

$Q_2 Q_1 Q_0$	$Q_2^+ Q_1^+ Q_0^+$
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	0 0 0
1 0 1	X X X
1 1 0	X X X
1 1 1	X X X

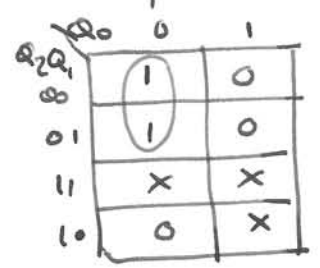
Karnaugh:



$Q_2^+ = Q_1 Q_0$

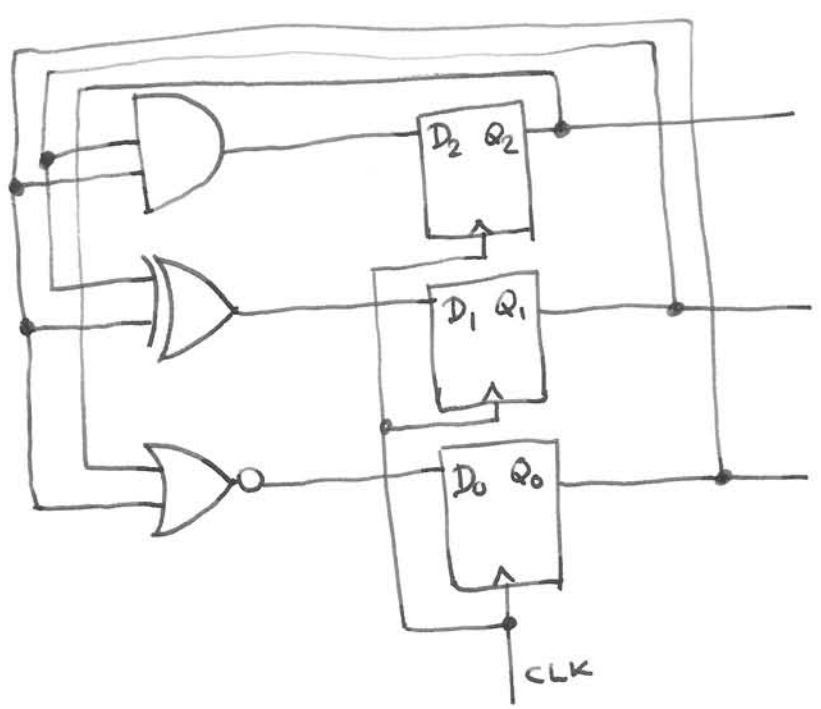


$Q_1^+ = Q_1 \bar{Q}_0 + \bar{Q}_1 Q_0$
$= Q_1 \oplus Q_0$



$Q_0^+ = \bar{Q}_2 \bar{Q}_0 =$
$= \overline{Q_2 + Q_0}$

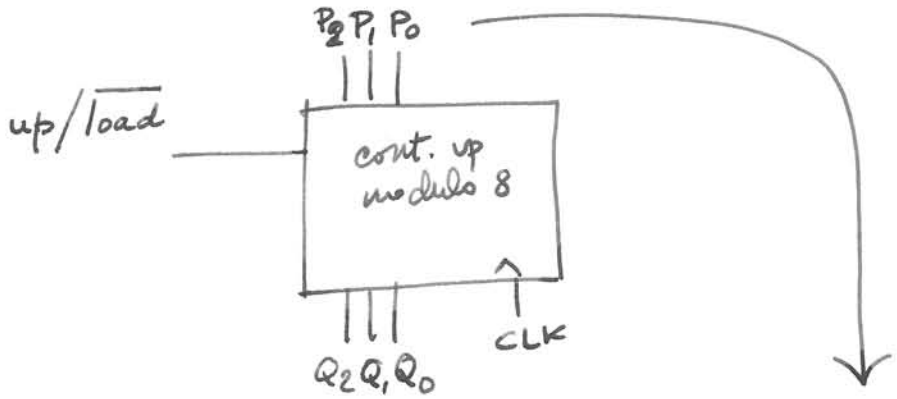
Circuito:



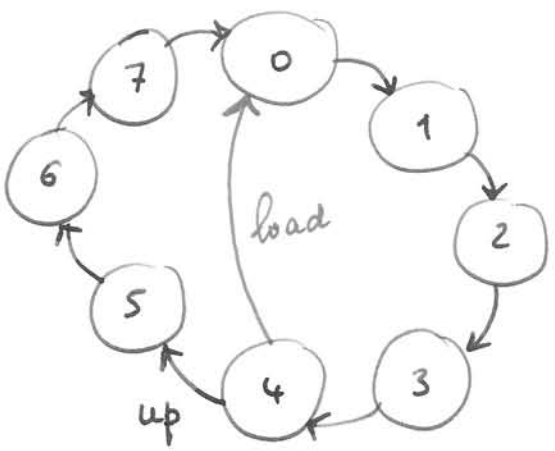
Notiamo ancora una volta che il contatore senza funzionalità addizionali è una rete autonoma, dunque senza ingressi, e per questo ha un'uscita periodica.

Supponiamo ora di voler realizzare la stessa funzione precedente (conteggio up modulo 5) ma a partire da un contatore up con funzionalità addizionale load:

→ modulo $8 = 2^3$.



Con automa:



Nota: l'ingresso di caricamento parallelo è posto qui a $P_2P_1P_0 = 000$, quindi se up/$\overline{load}$ = 0, lo stato futuro è 0.

Come fare a contare modulo 5? Dovremo prevedere un secondo arco uscente da 4 ed entrante in 0 (vedi freccia rossa in fig.). Ora dallo stato 4 partono due possibili archi, e questo richiede il comportamento di un ingresso esterno: up/$\overline{load}$, per l'appunto.

%

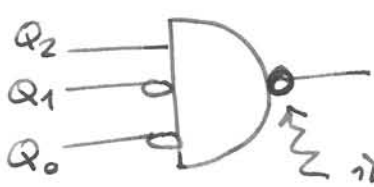
Tuttavia, se ipotizziamo che l'ingresso nello stato 4 sia esso stesso funzione del solo stato presente (4, appunto), allora il valore dell'ingresso, poiché funzione combinatoria di un valore costante, potrà avere solo uno dei due possibili valori: 0 oppure 1. Ciò significa che potremo fare in modo da evitare che dallo stato 4 il contatore prosegua ~~nello~~ con lo stato 5: basterà porre

$$\text{ingresso nello stato 4} = \text{load} \quad \text{(con riferimento in 0)}$$

mentre

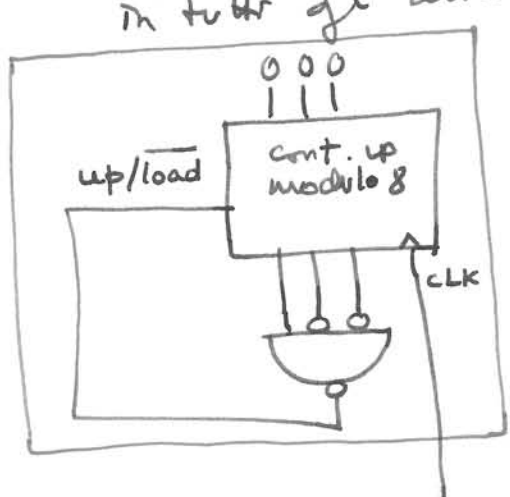
$$\text{ingresso in q/s stato} \neq \text{stato 4} = \text{up}$$

Realizzazione dell'ingresso desiderato: basta un riconoscitore della configurazione $Q_2Q_1Q_0 = 100$ (=4).

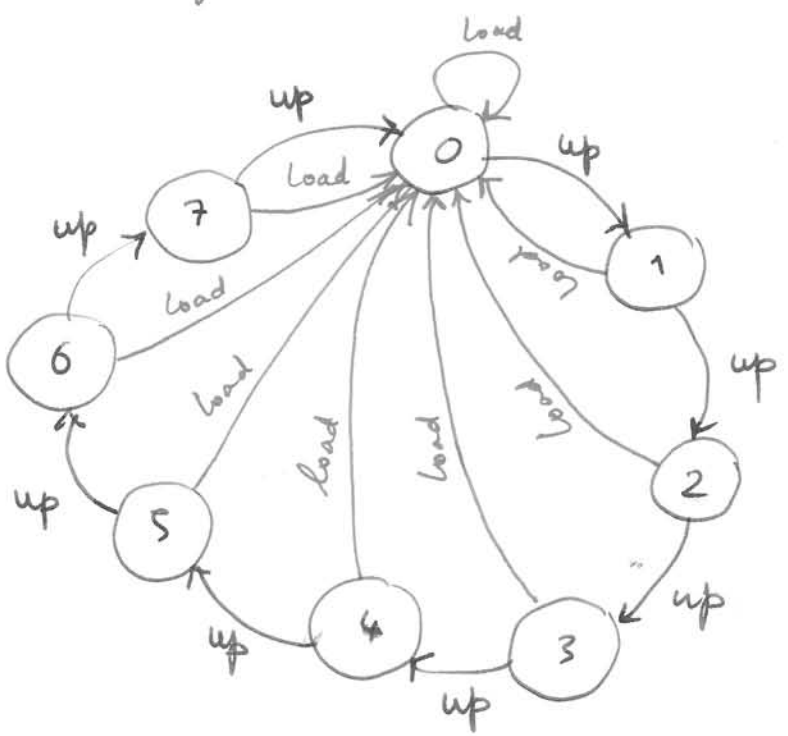


invece della consueta porta AND, qui usiamo una NAND in modo da avere up/$\overline{\text{load}}$ = 0 nello stato 4 (quello riconosciuto) e up/$\overline{\text{load}}$ = 1 in tutti gli altri casi.

Schema circuito:



Vista da fuori, il circuito NON ha ingressi, e si comporta dunque da macchina AUTONOMA (non fa niente realizza esattamente il contatore modulo 5 già visto prima). Vale la pena di osservare che il contatore up modulo 8 con funzionalità aggiunta di load avesse in realtà un automa in cui da ogni stato 0.....7 si sarebbe potuto raggiungere lo stato futuro 0 posto nell'ingresso di caricamento. L'automa completo era dunque il seguente:



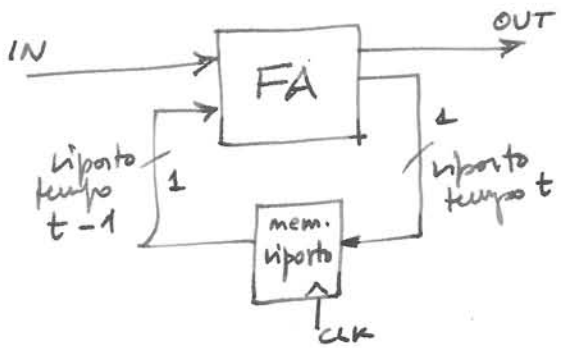
In realtà, poiché

$$load = Q_2 \bar{Q}_1 \bar{Q}_0 ,$$

in qualunque stato diverso da 100 si ha $load = 0 \Rightarrow up = 1$, quindi il caricamento parallelo si fa solo nello stato 4.

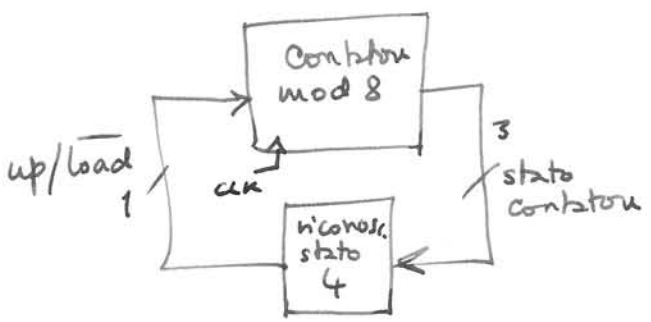
Notiamo infine come alcune delle macchine sequenziali viste finora siano casi particolari della rappresentazione PARTE OPERATIVA e PARTE DI CONTROLLO. Secondo questa rappresentazione, qualsiasi macchina è riguardabile (e progettabile) come il complesso di 2 macchine sequenziali che comunicano tra loro.

Ad es., il SOMMATORE SERIALE aveva lo schema:



Qui la parte operativa è il FA (combinatorio) e quella di controllo è il FF che memorizza il riporto (sequenziale)

Nell'ultimo schema del contatore modulo 5 avevamo invece



Qui la PO è sequenziale (contatore), mentre la PC è combinatoria (riconosc. stato 4).

Nel Caso generale:

con PO e PC entrambe sequenziali