Esercizio 1

Progettare un circuito sequenziale sincrono che regola l'accensione di 3 lampadine a led.

Il circuito integra un contatore (sincrono) modulo k, dove k, specificato in ingresso su 3 bit, è un intero positivo appartenente al range [1, 4]

Assumiamo che possano essere specificati in ingresso solo valori nel range [1, 4]

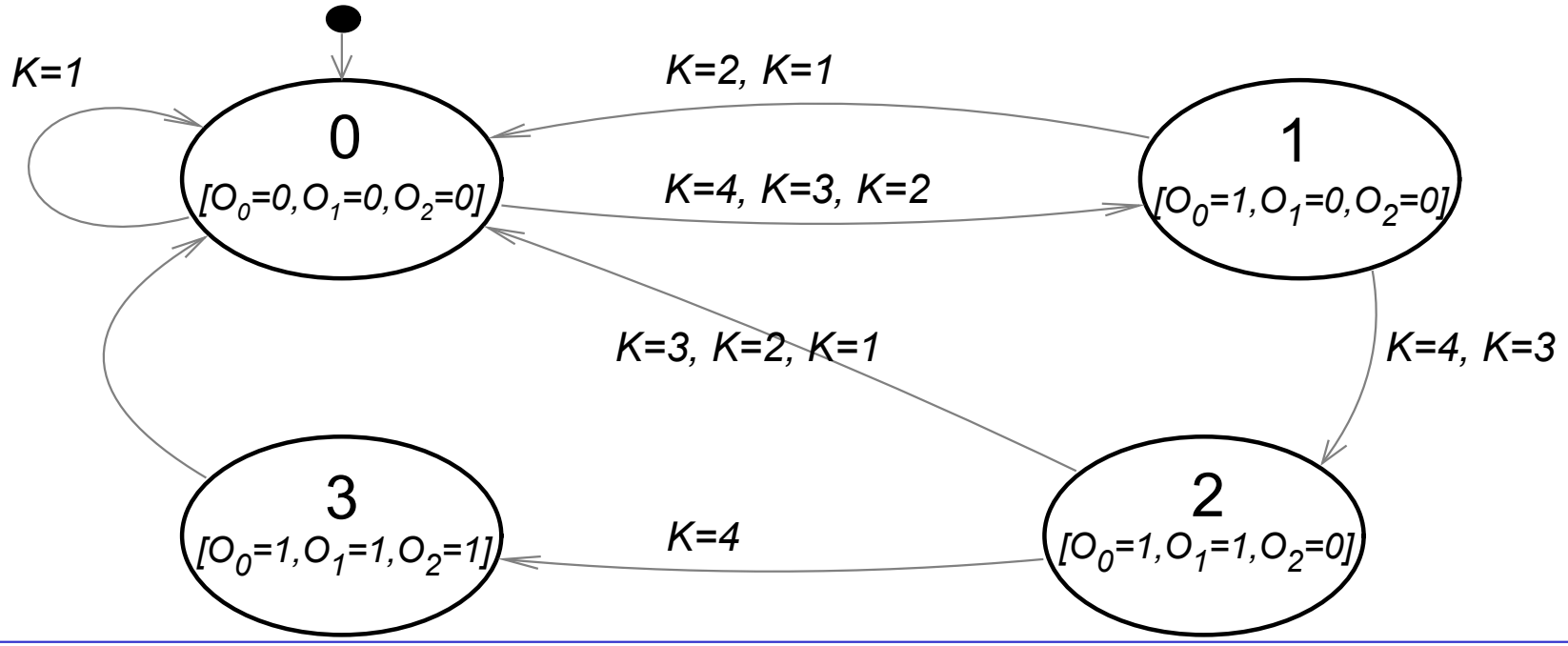
Il circuito opera facendo in modo che in ogni istante il numero di lampadine accese corrisponde al valore memorizzato dal contatore.

Il circuito inizia ad operare partendo da una configurazione in cui tutte le lampadine sono spente.

Esercizio 1: soluzione

Gli stati ammissibili corrispondono ai valori memorizzabili dal contatore
k può valere : 1, 2, 3 o 4
Gli stati denotano i valori 0, 1, 2, 3

Ad ogni ciclo di clock il valore da memorizzare dipende da k:
se S < k allora S'= (S+1) mod k, altrimenti S'=0



Esercizio 1: soluzione
Approccio 2 - un flip-flop per stato

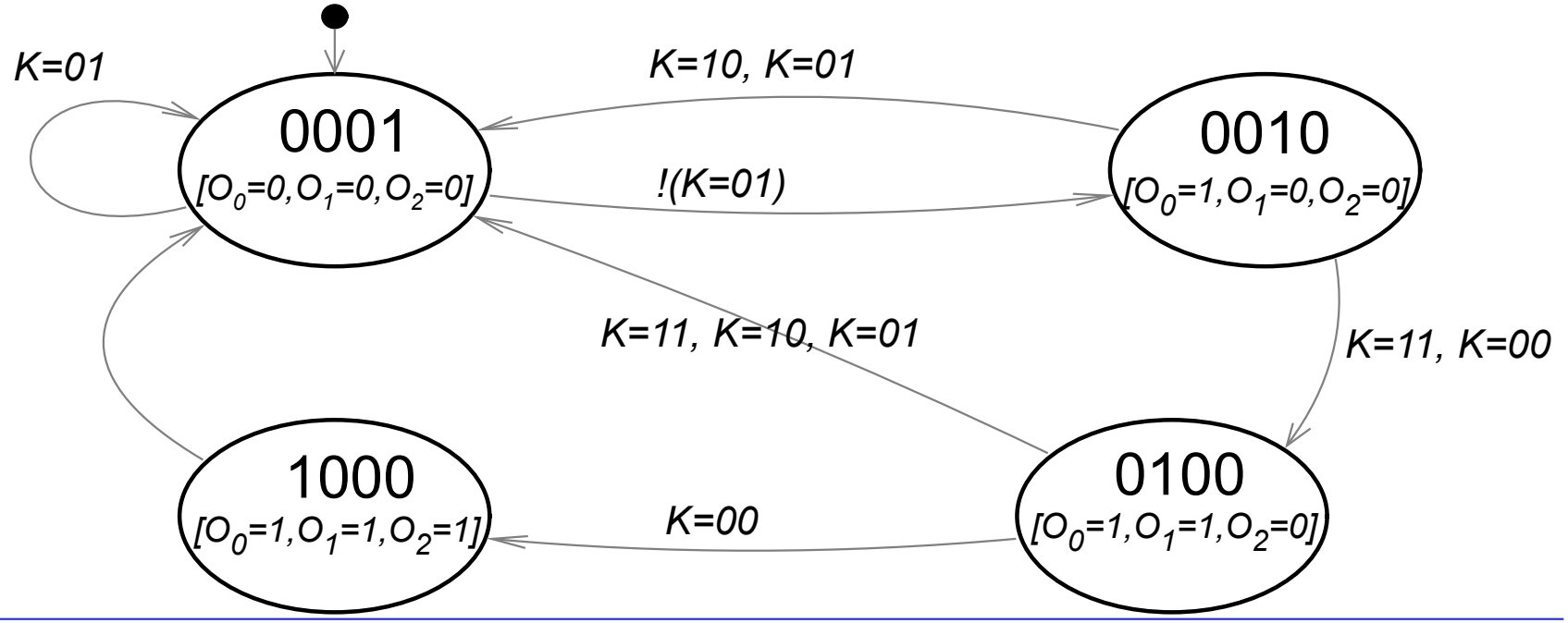
La codifica degli stati specificata su un numero di bit pari al numero di stati

- Un solo bit a 1, tutti gli altri a 0

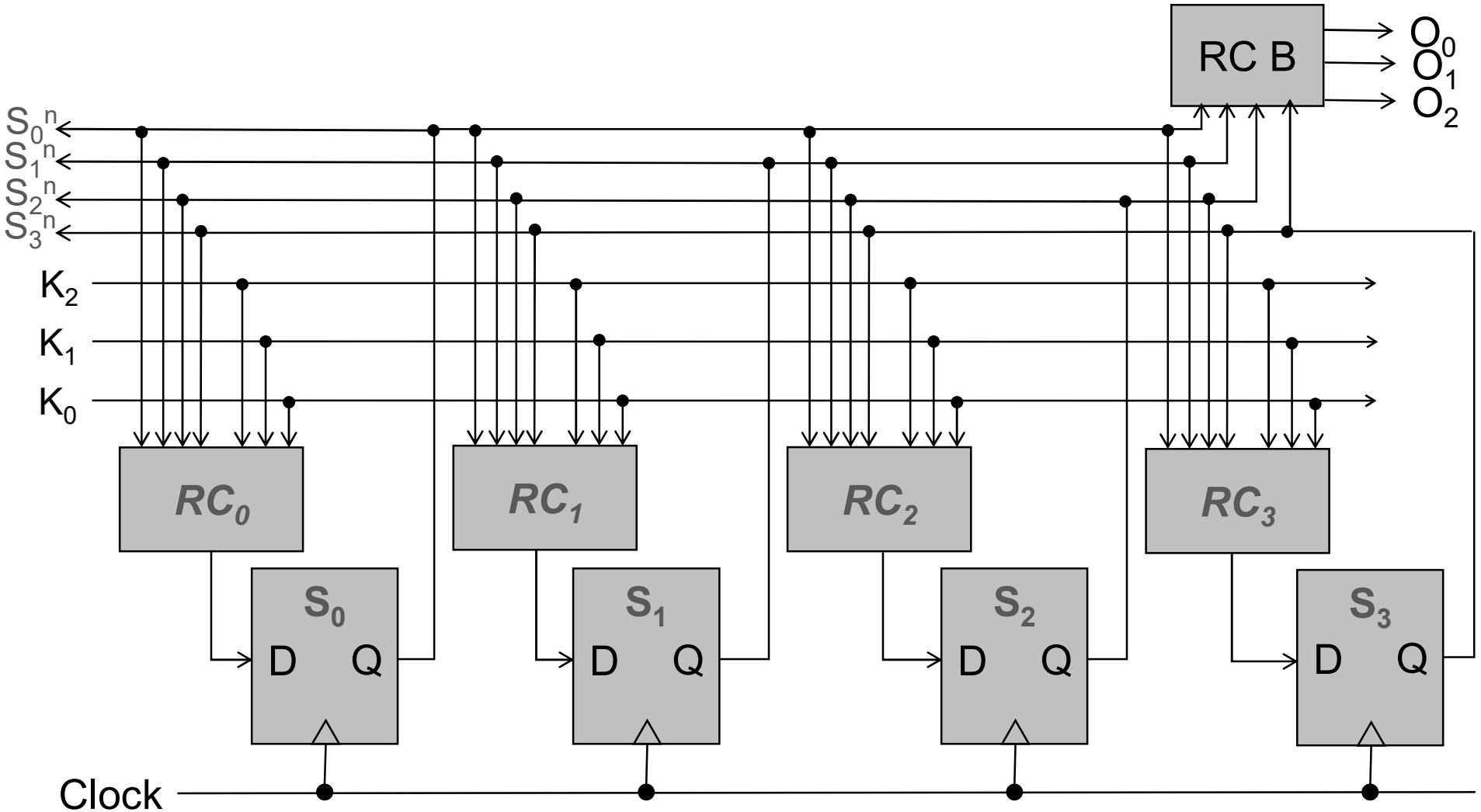
S=0 → 0001, S=1 → 0010, S=2 → 0100, ed S=3 → 1000

I valori di k ammissibili sono 4: 001, 010, 011, 100 ->

- sono sufficienti 2 bit per codificare l'ingresso: uso i 2 meno significativi
  01 denota 1, 10 denota 2, 11 denota 3, 00 denota 4



Esercizio 1: soluzione
schema di riferimento



Esercizio 1: soluzione – *Rete combinatoria B*

S	S_3^n	S_2^n	S_1^n	S_0^n	O_2	O_1	O_0
0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	1
2	0	1	0	0	0	1	1
3	1	0	0	0	1	1	1

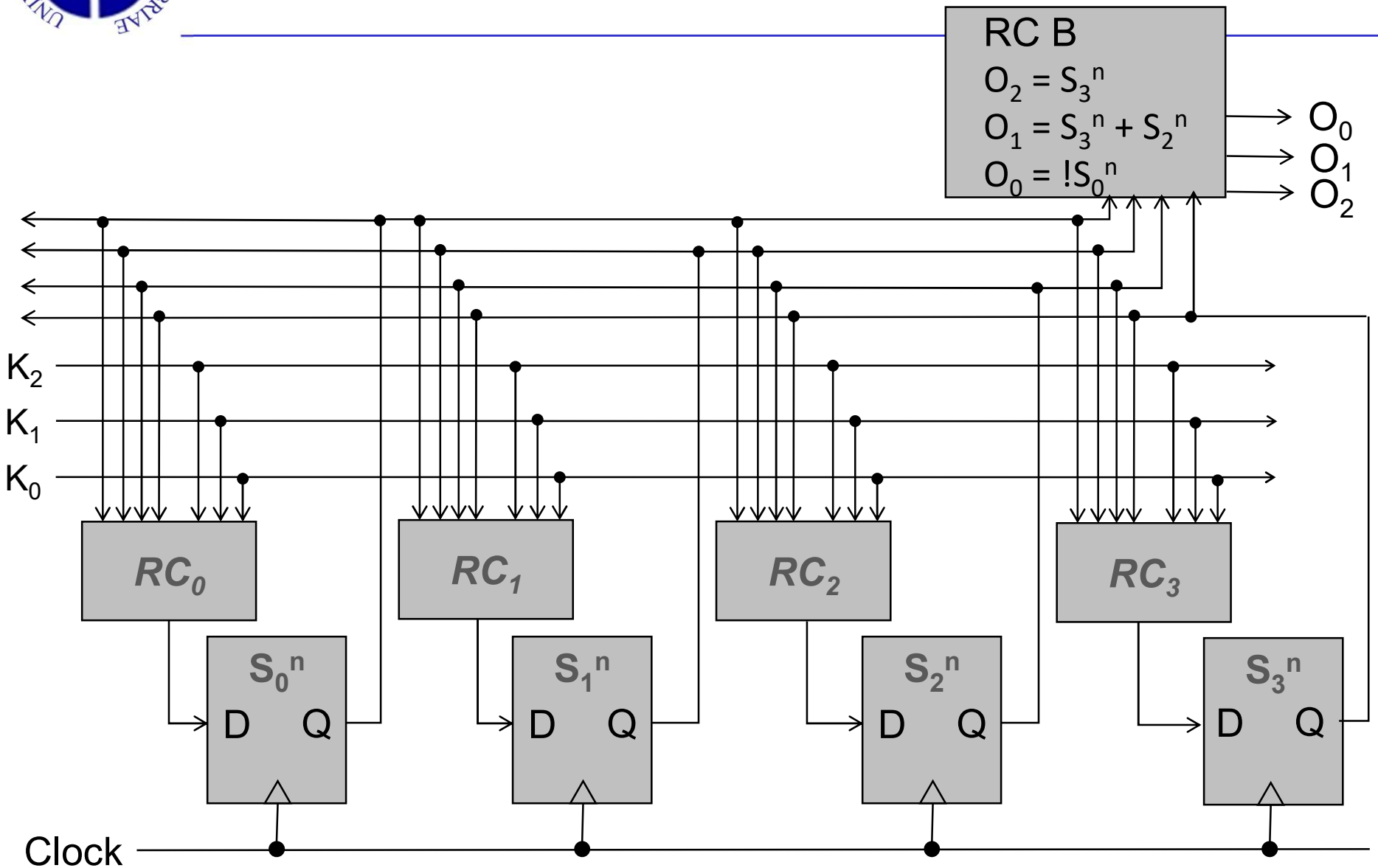
Le uscite dipendono solo dallo stato corrente, non dagli ingressi:

$O_2 = S_3^n$

$O_1 = S_3^n + S_2^n$

$O_0 = !S_0^n$

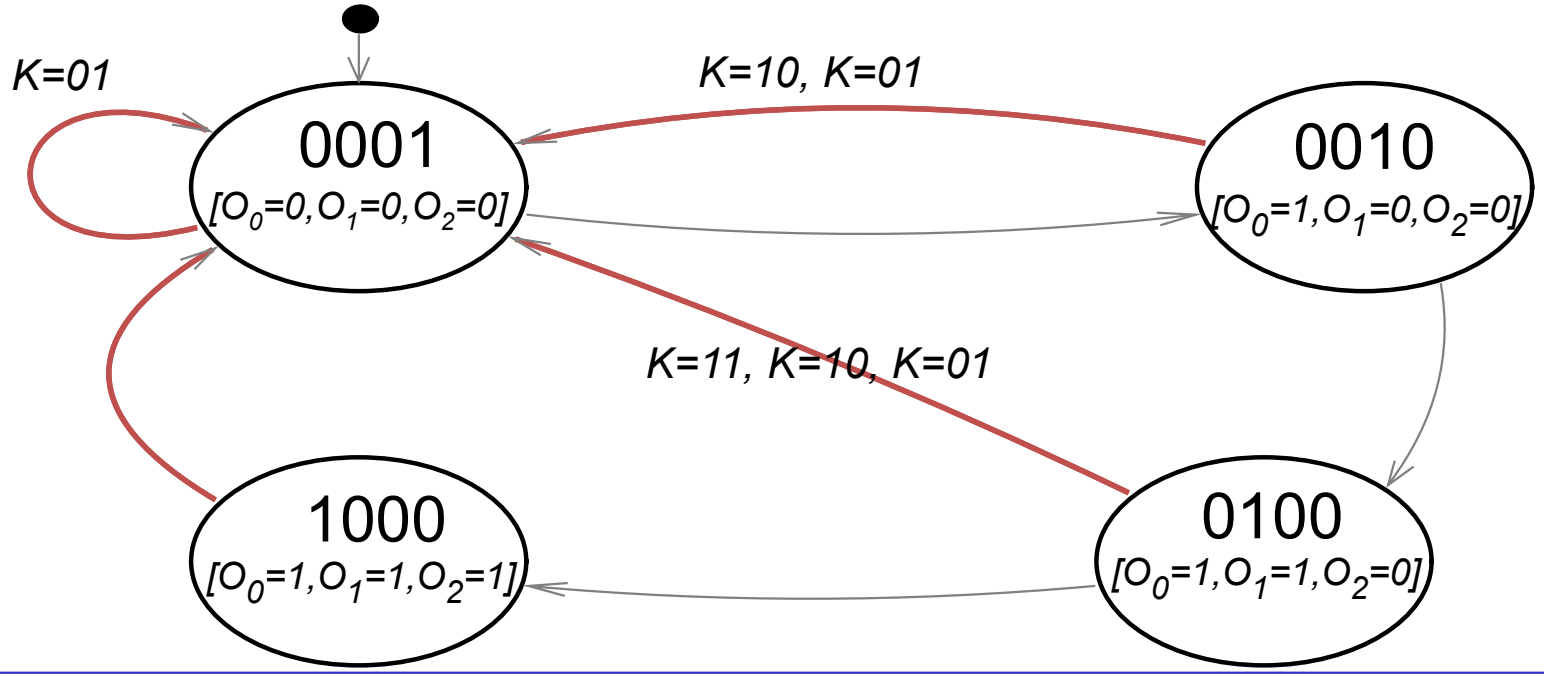
Esercizio 1: soluzione



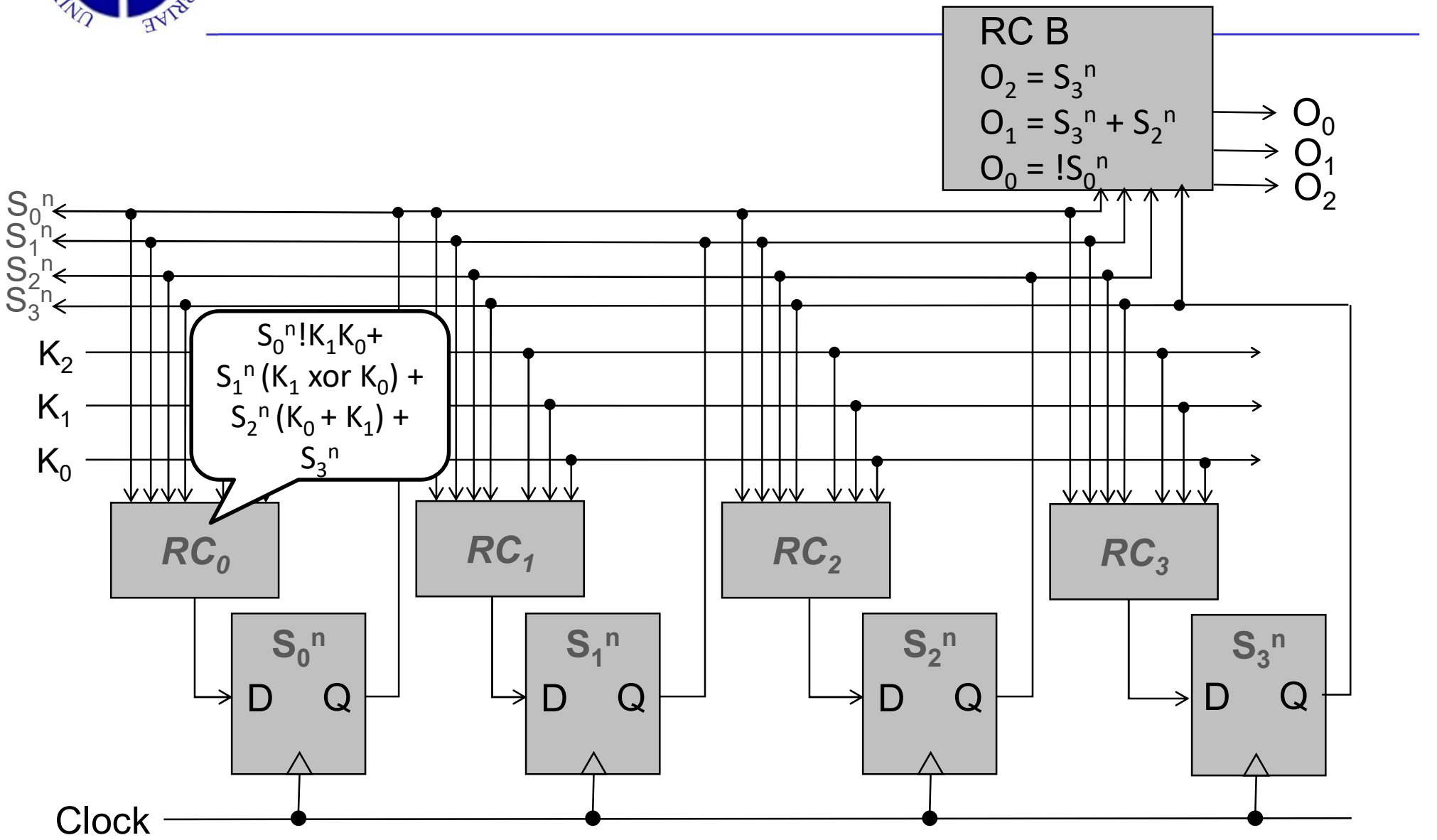
Esercizio 1: soluzione

Calcolo di RC_0 il circuito che memorizza il bit meno significativo della codifica di stato ovvero S_0

$S_0^{n+1}=1$ *(il contatore memorizza 0)*
sse $S_0^n!K_1K_0 + S_1^n(!K_1K_0 + K_1!K_0) + S_2^n(!K_1K_0 + K_1!K_0 + K_1K_0) + S_3^n$
$= S_0^n!K_1K_0 + S_1^n(K_1 \text{ xor } K_0) + S_2^n(!K_1K_0 + K_1(!K_0 + K_0)) + S_3^n$
$= S_0^n!K_1K_0 + S_1^n(K_1 \text{ xor } K_0) + S_2^n(K_0 + K_1) + S_3^n$



Esercizio 1: soluzione



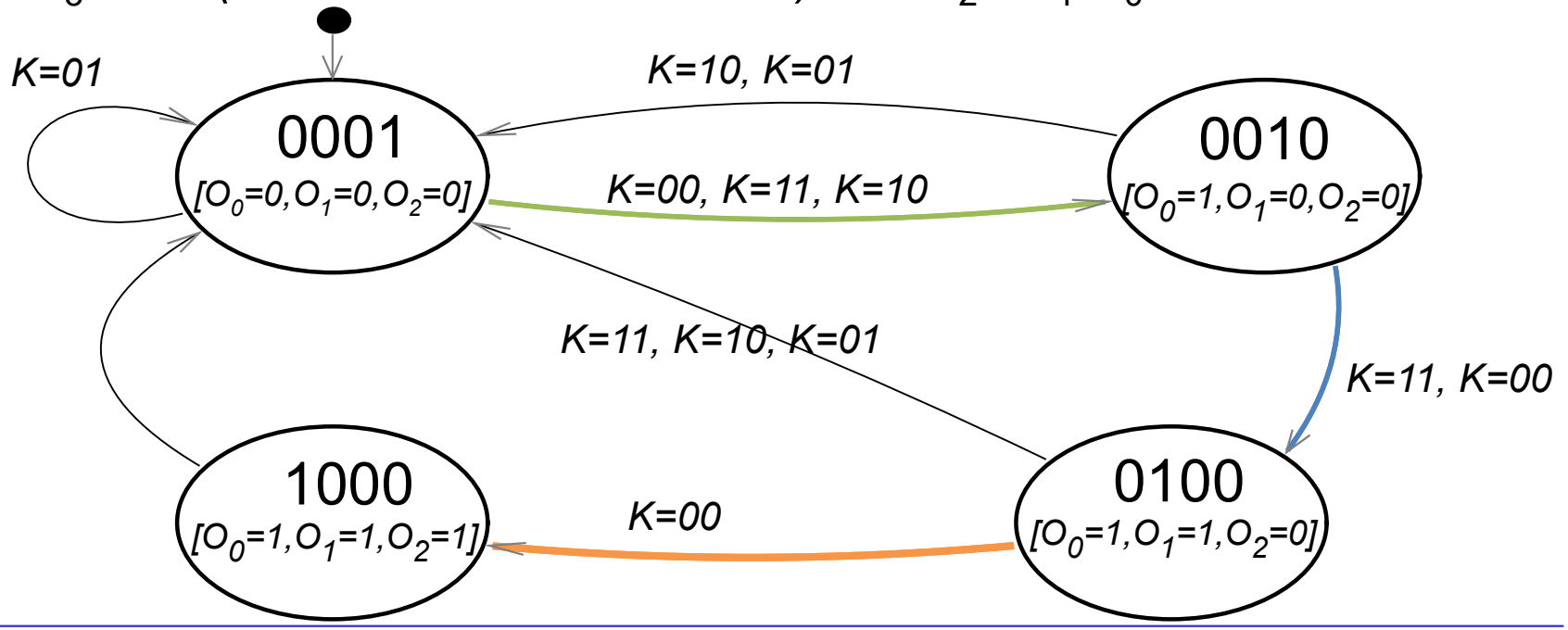
Esercizio 1: soluzione

Il calcolo di RC_1,RC_2 ed RC_3 , che generano rispettivamente i bit S_1, S_2 ed S_3 della codifica di stato è molto più semplice (una sola transizione)

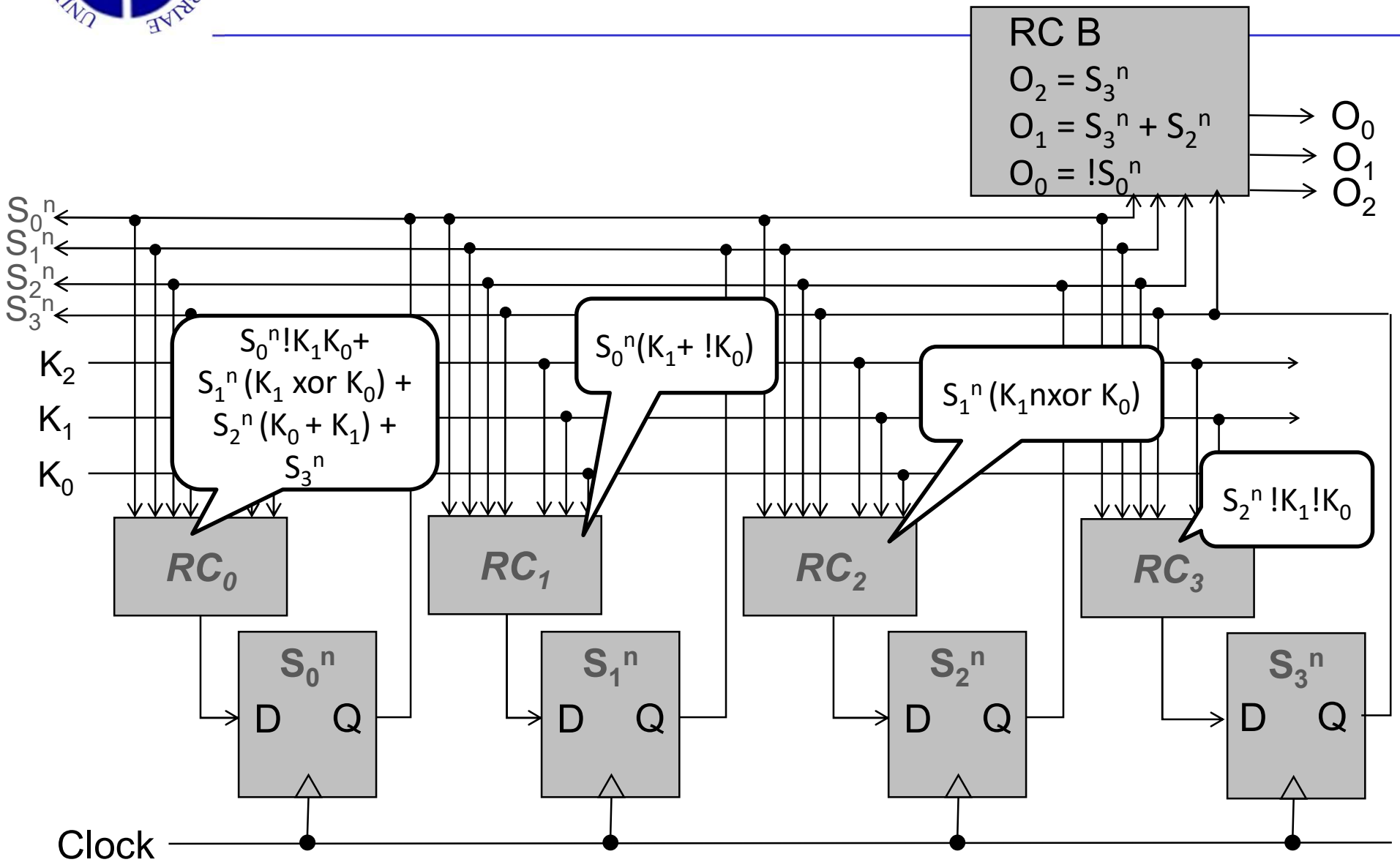
RC_1: $S_1^{n+1}=1$ *(il contatore memorizza 1)* sse $S_0^n(K_1 + !K_0)$

RC_2: $S_2^{n+1}=1$ *(il contatore memorizza 2)* sse S_1^n (K_1nxor K_0)

RC_3: $S_3^{n+1}=1$ *(il contatore memorizza 3)* sse S_2^n $!K_1!K_0$



Esercizio 1: soluzione

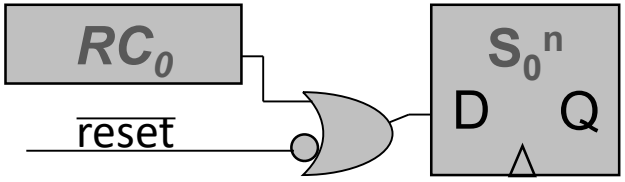


Esercizio 1: soluzione

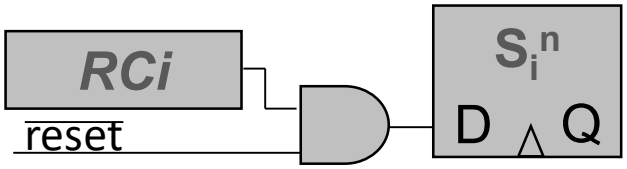
Funziona a regime solo se un solo flip-flop memorizza 1 e gli altri 0…
E' necessario inizializzare il circuito allo stato 0:

- Serve un segnale esterno di *reset* per forzare la scrittura di 1 in S_0^n e 0 in S_1^n ,S_2^n, ed S_3^n
- Meglio sfruttare un segnale attivo basso ($\overline{reset}$), che ha effetto quando tale vale 0 (l'esecuzione parte automaticamente dallo stato 0)

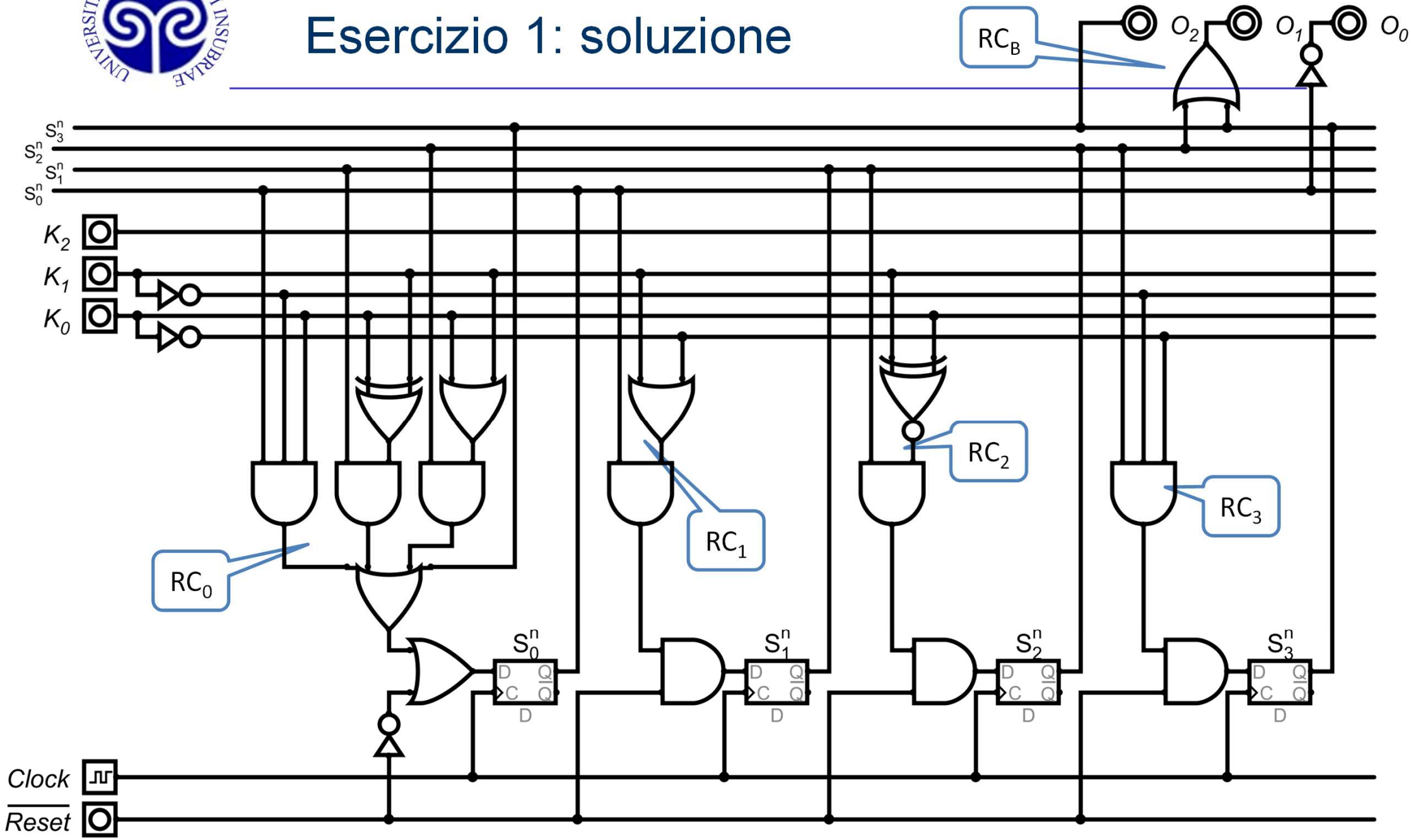
- Lo stato 0 è codificato 0001,
  - per forzare la scrittura di 1 in S_0^n bisogna agire tra RC_0 ed il flip-flop S_0^n: inverto $\overline{reset}$ e lo metto in OR con l'output di RC_0 e invio il segnale a S_0^n



  - per scrivere 0 in S_1^n ,S_2^n ed S_3^n bisogna agire tra RC_1 e S_1^n, RC_2 e S_2^n, e tra RC_3 e S_3^n : metto in AND $\overline{reset}$ con l'output di RC_i e invio il segnale a S_i^n (i vale 1, 2 e 3)



Esercizio 1: soluzione

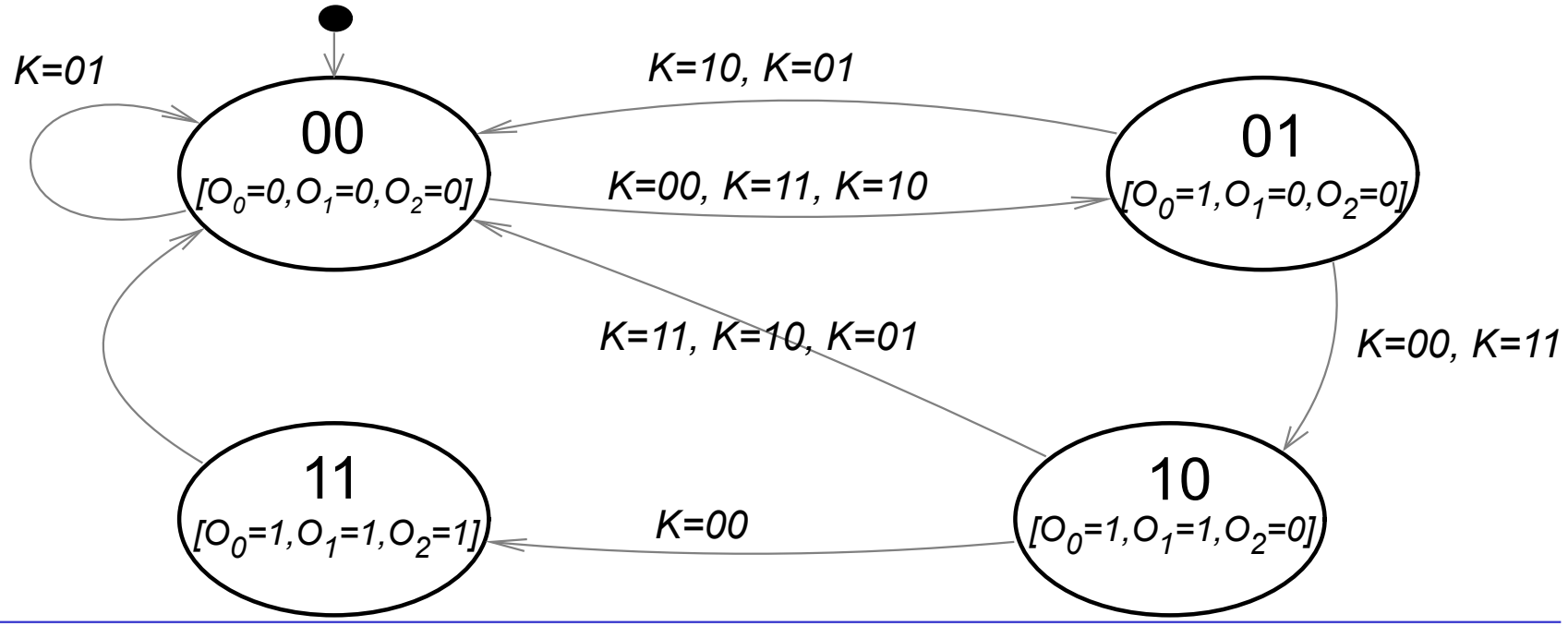


Esercizio 1: soluzione
approccio minimizzando i flip flop

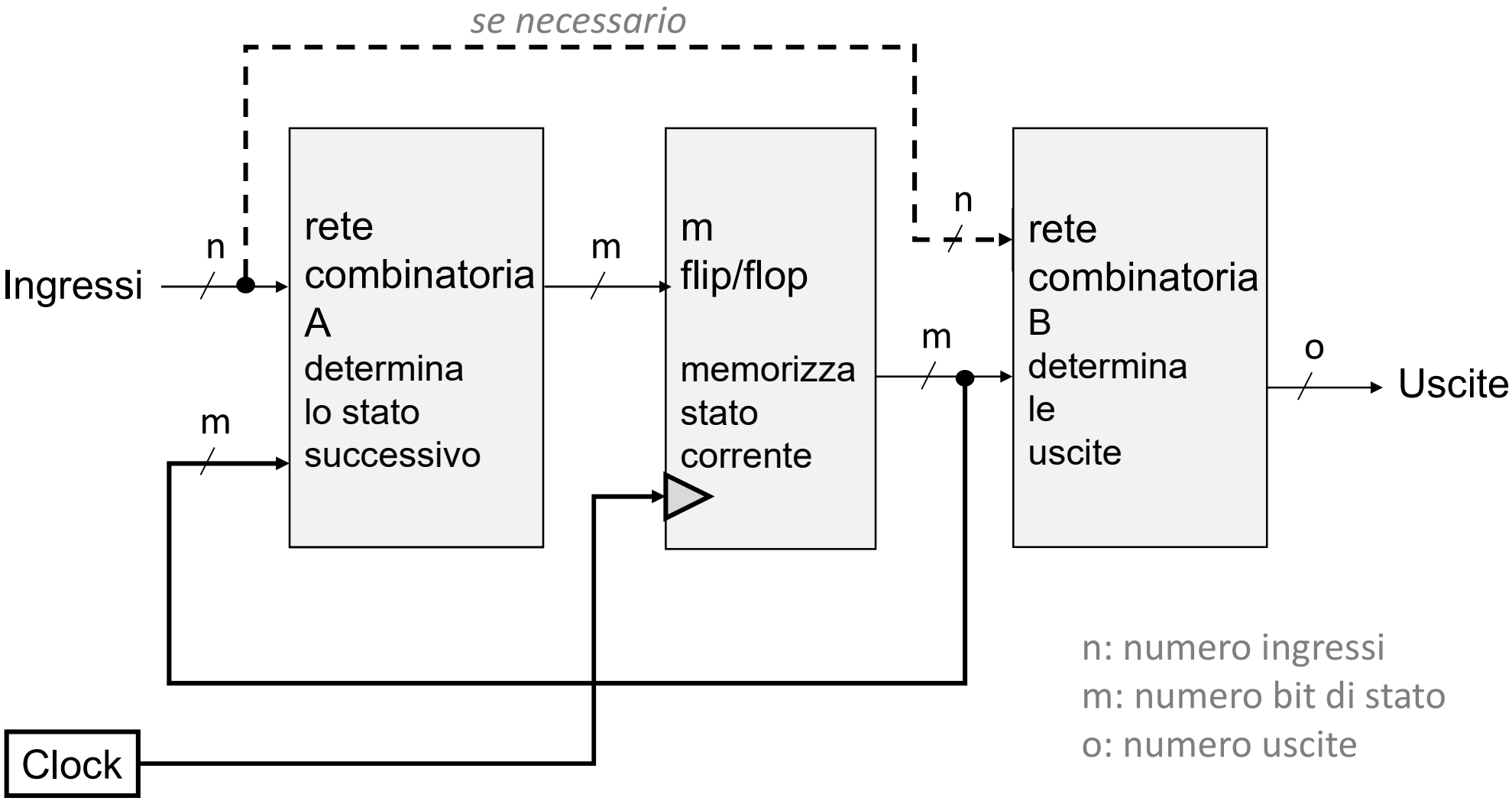
La codifica degli stati richiede 2 bit (2= $\log_2 4$, dove 4 è il numero di stati)
S può valere 00, 01, 10, 11

I valori di k ammissibili sono 4: 001, 010, 011, 100 ->

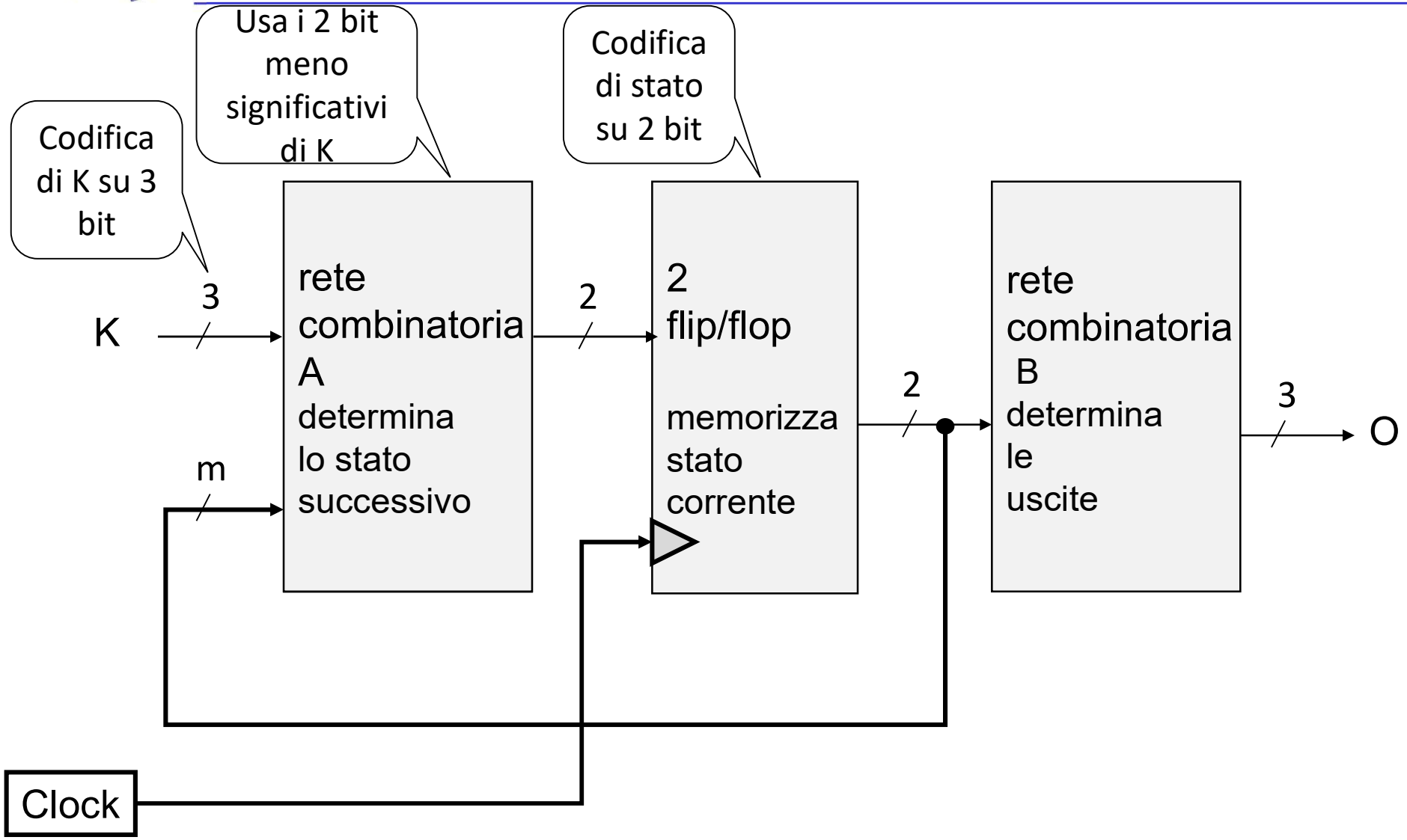
- sono sufficienti 2 bit per codificare l'ingresso: uso i 2 meno significativi
  01 denota 1, 10 denota 2, 11 denota 3, 00 denota 4



Esercizio 1: soluzione
schema di riferimento



Esercizio 1: soluzione
schema di riferimento



Esercizio 1: soluzione – *Rete combinatoria B*

S	S_1^n	S_0^n	O_2	O_1	O_0
0	0	0	0	0	0
1	0	1	0	0	1
2	1	0	0	1	1
3	1	1	1	1	1

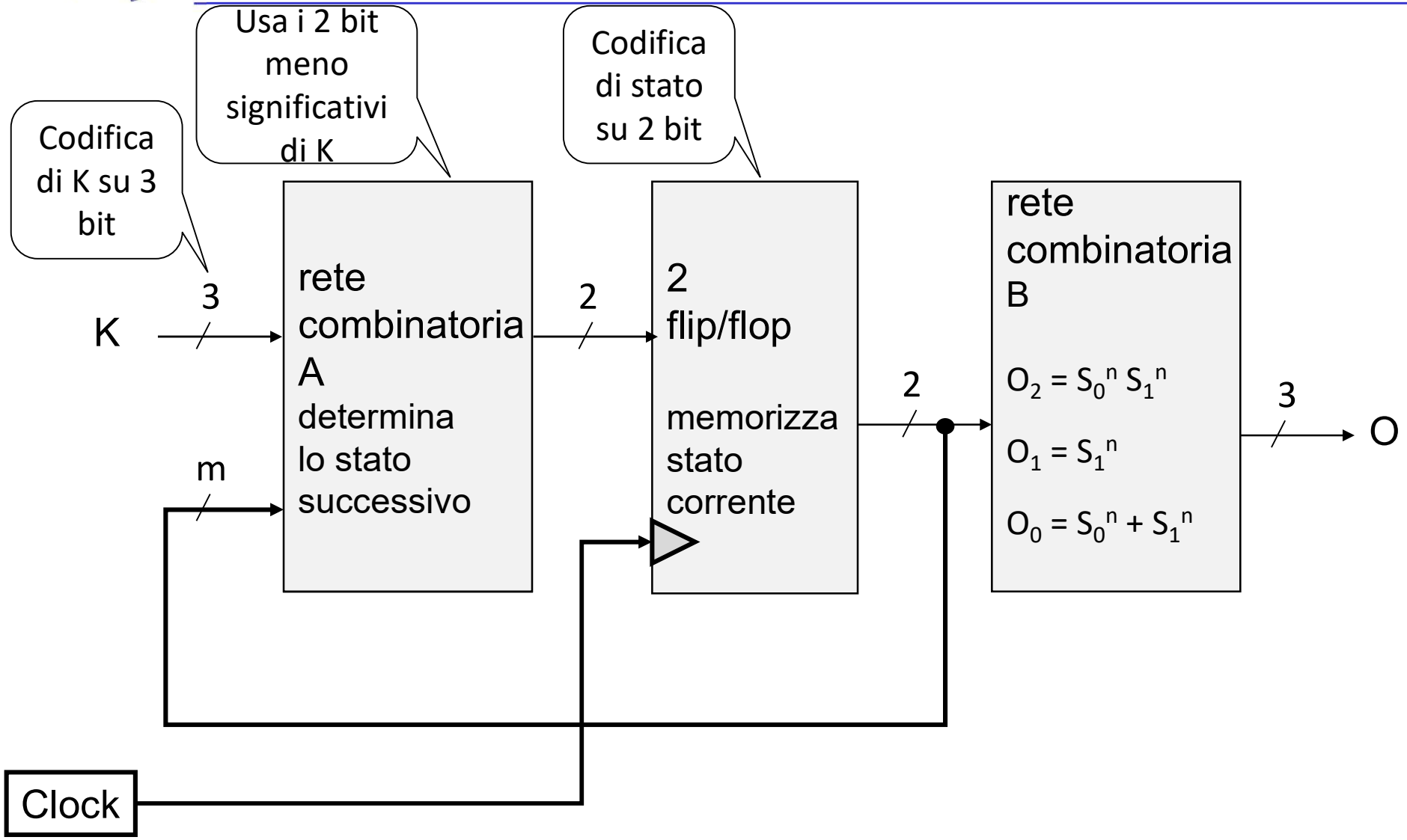
Le uscite dipendono solo dallo stato corrente, non dagli ingressi:

$$O_2 = S_0^n \, S_1^n$$

$$O_1 = S_1^n$$

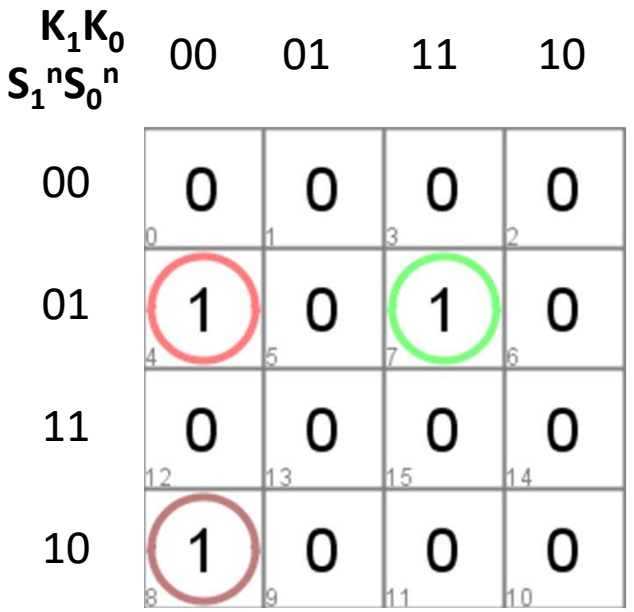
$$O_0 = S_0^n + S_1^n$$

Esercizio 1: soluzione

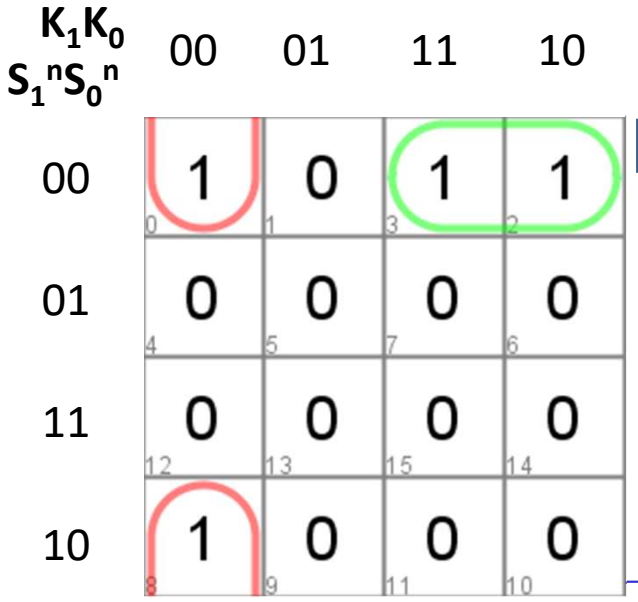


Esercizio 1: soluzione – *Rete combinatoria A*

S_1^n	S_0^n	K_1	K_0	S_1^{n+1}	S_0^{n+1}
0	0	0	0	0	1
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

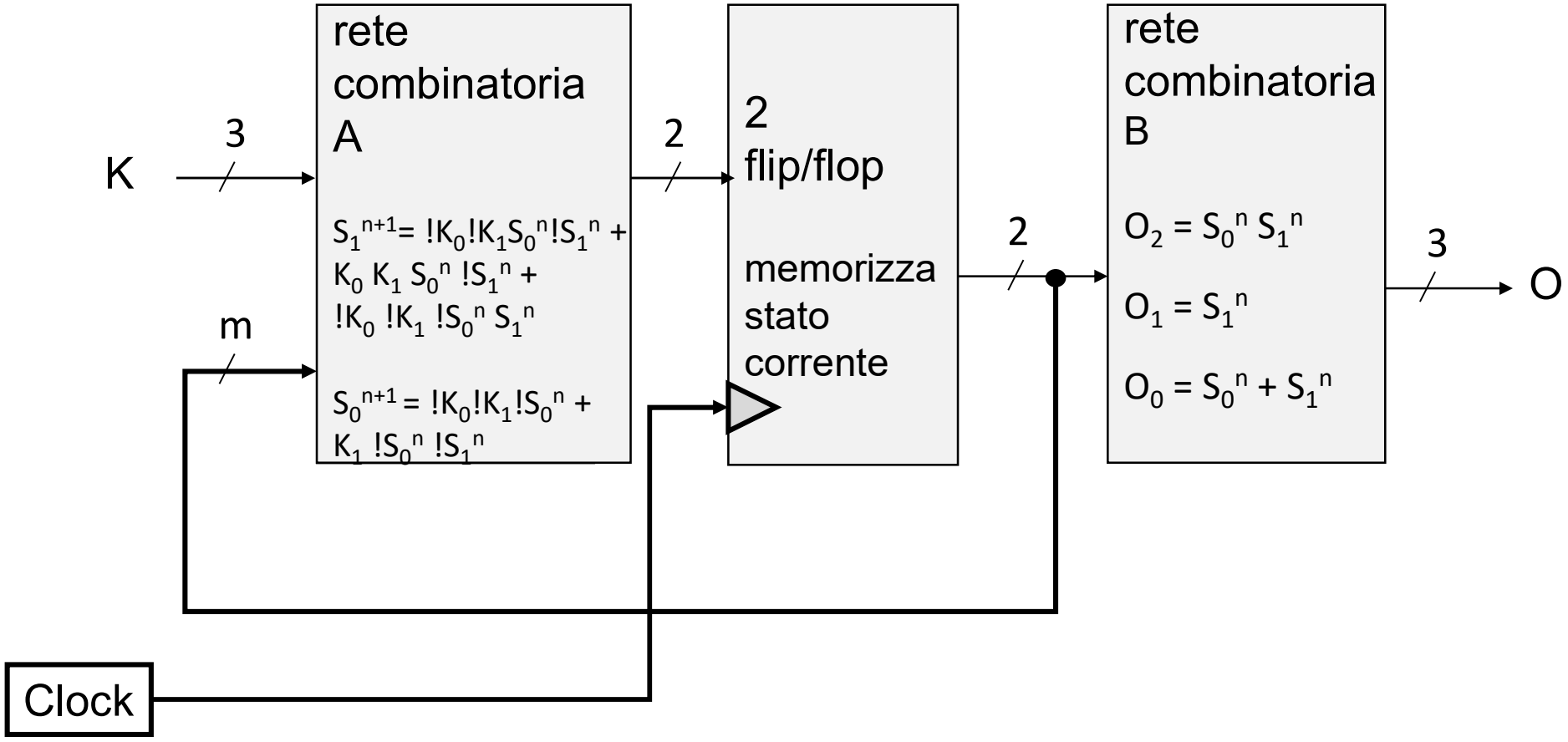


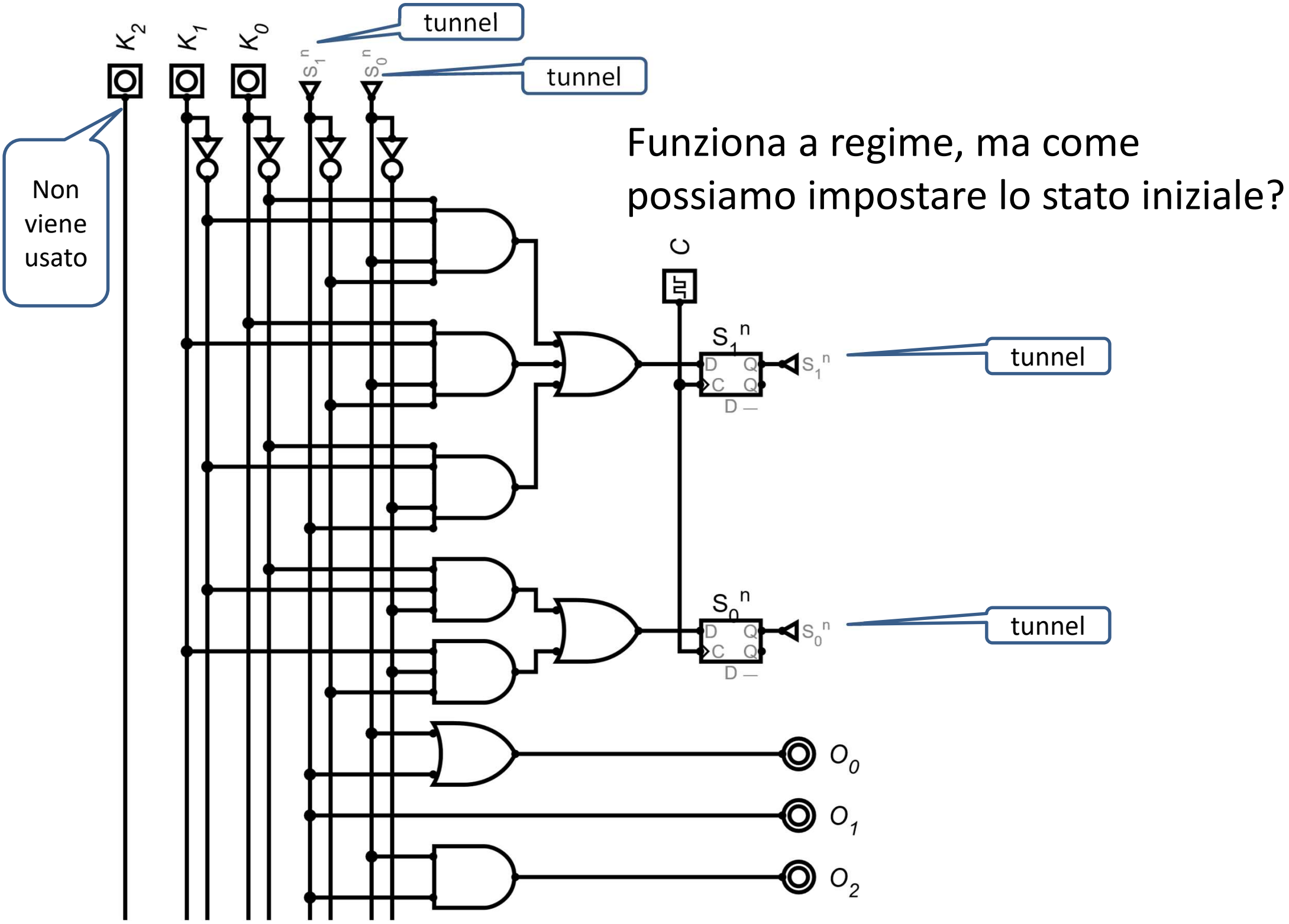
$$S_1^{n+1} = !K_0 !K_1 S_0^n !S_1^n + K_0 K_1 S_0^n !S_1^n + !K_0 !K_1 !S_0^n S_1^n$$



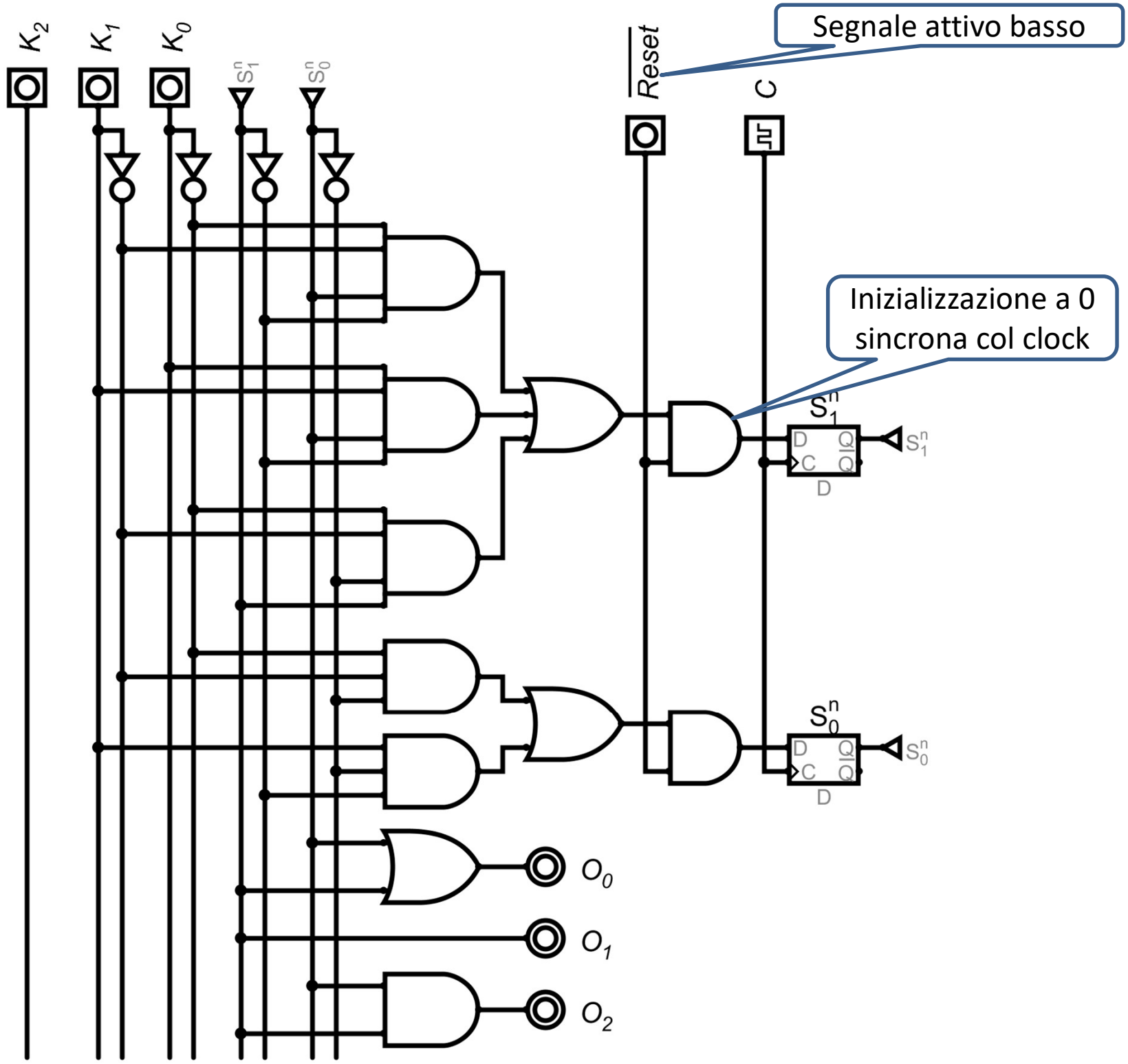
$$S_0^{n+1} = !K_0 !K_1 !S_0^n + K_1 !S_0^n !S_1^n$$

Esercizio 1: soluzione





Funziona a regime, ma come possiamo impostare lo stato iniziale?


Segnale attivo basso
Inizializzazione a 0
sincrona col clock
K_2
K_1
K_0
S_1^n
S_0^n
Reset
C
D
Q
O_0
O_1
O_2

Esercizio 2

- Progettare un circuito contatore binario sincrono modulo 8 impiegando un registro parallelo a 3 bit.
  - Il contatore ad ogni ciclo di clock aggiorna il valore memorizzato nel registro
  - Il valore memorizzato varia da 0 a 7
  - Si può trascurare lo stato iniziale. Basta che il circuito funzioni correttamente a regime