

Laboratorio di Architetture degli Elaboratori I
Corso di Laurea in Informatica, A.A. 2024-2025
Università degli Studi di Milano



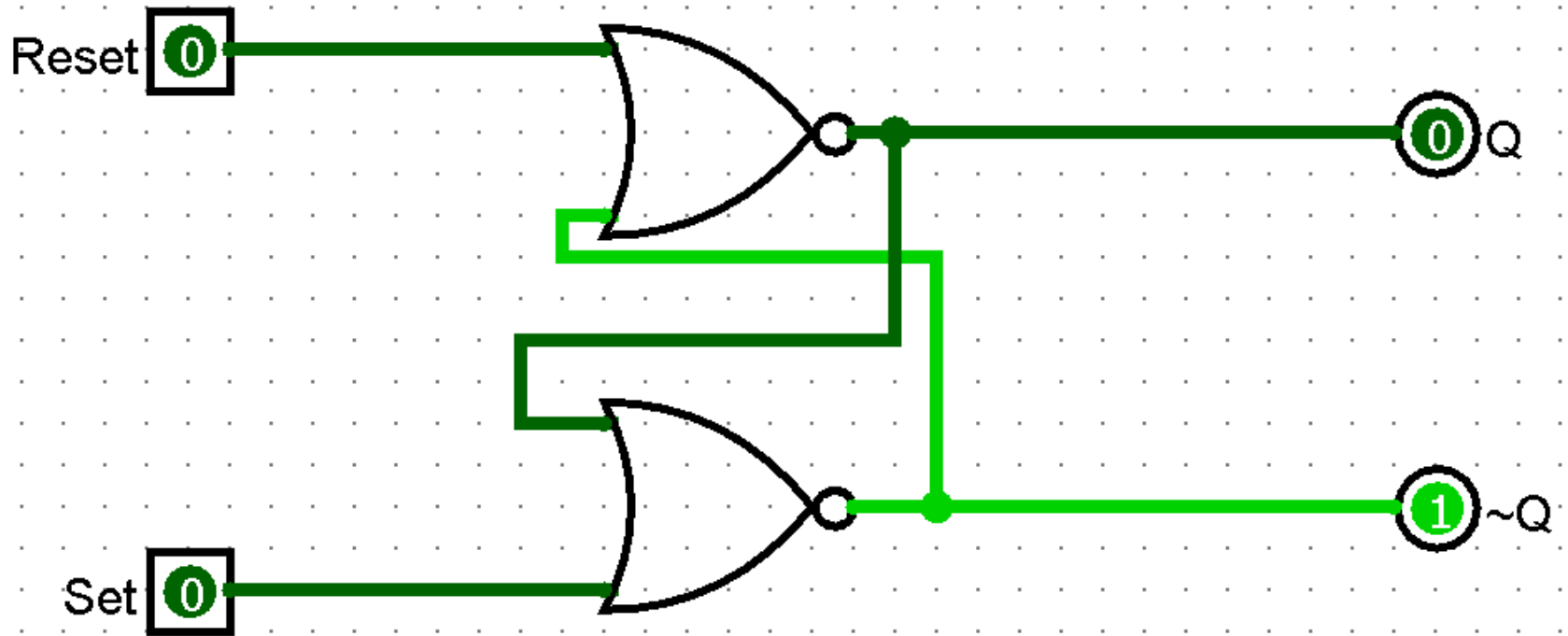
Memorie

Esercizio 1

- Si implementi in Logisim un bistabile Set Reset (SR) asincrono utilizzando due porte NOR retroazionate
- Si definisca la tabella delle transizioni (o stato prossimo) del bistabile
- Si derivi la SOP e la si semplifichi, implementandone il circuito corrispondente (si assuma prima $X=0$, poi $X=1$ per le due uscite indeterminate della tabella di verità)

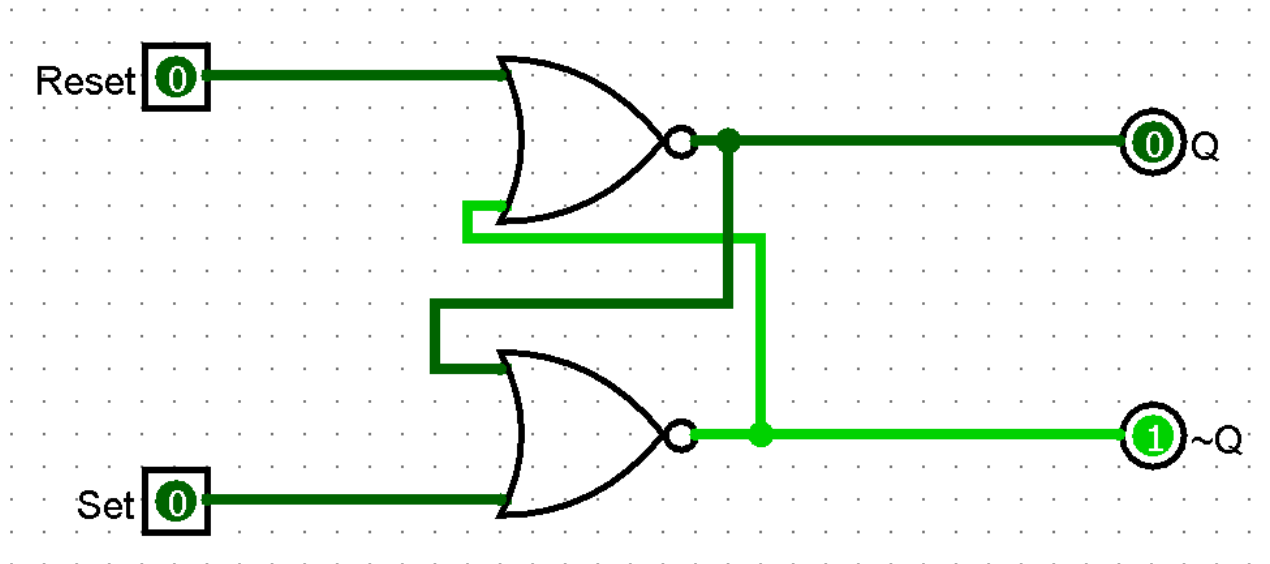
Esercizio 1

Bistabile asincrono SR con due NOR



Esercizio 1

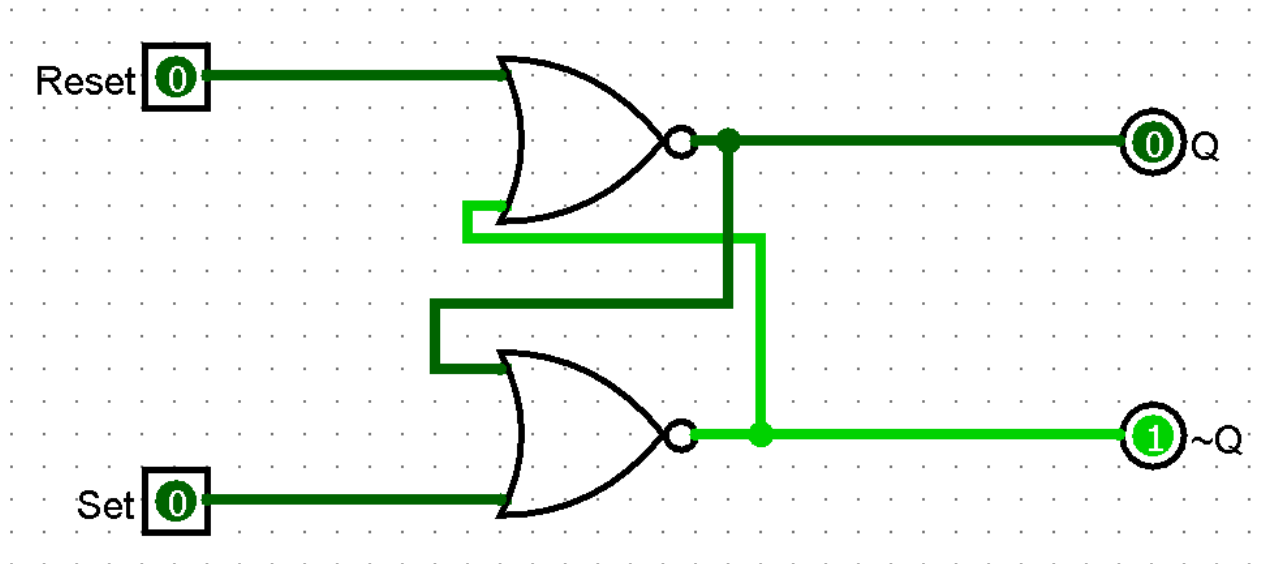
Bistabile asincrono SC (o SR) con due NOR



- **Q** è sia lo **stato** del bistabile che la sua **uscita** ($\sim Q$ è lo stato negato, quindi **Q** e $\sim Q$ hanno sempre valori complementari)
- Lo stato/uscita del circuito è modificato dalle variazioni di due segnali:
 - Quando **Set (S)** passa da **0** a **1** lo stato passa a **1 stabilmente** (cioè **Q** continua a valere 1 anche dopo che **S** è tornato a 0; abbiamo memorizzato un 1)
 - Quando **Reset (R)** passa da **0** a **1** lo stato passa a **0 stabilmente** (cioè **Q** continua a valere 0 anche dopo che **R** è tornato a 0; abbiamo memorizzato uno 0)

Esercizio 1

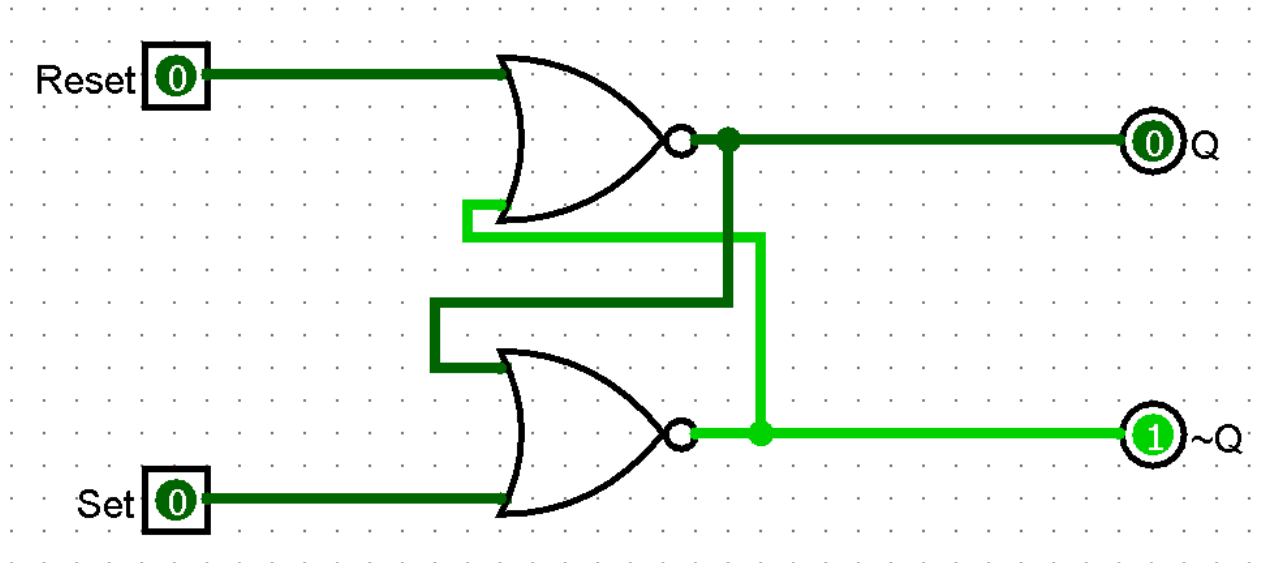
Bistabile asincrono SC (o SR) con due NOR



- S e R non devono trovarsi mai contemporaneamente a 1. Perché?

Esercizio 1

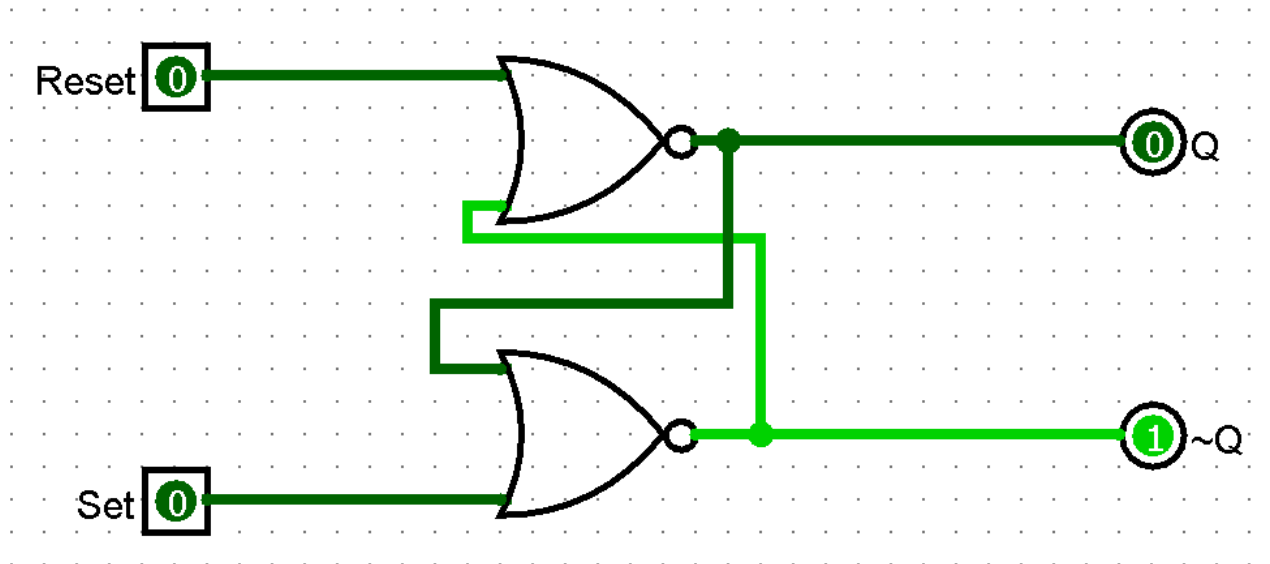
Bistabile asincrono SC (o SR) con due NOR



- S e R non devono trovarsi mai contemporaneamente a 1. Perché?
 - *Dal punto di vista logico: è come se stessi cercando memorizzare nello stato uno 0 e un 1 contemporaneamente, che è contraddittorio*
 - *Dal punto di vista fisico: lo stato diventa dipendente dai ritardi di commutazione e quindi non è prevedibile (non deterministico)*
- Questo circuito è sincrono o asincrono?

Esercizio 1

Bistabile asincrono SC (o SR) con due NOR



- S e R non devono trovarsi mai contemporaneamente a 1. Perché?
 - *Dal punto di vista logico: è come se stessi cercando memorizzare nello stato uno 0 e un 1 contemporaneamente, che è contraddittorio*
 - *Dal punto di vista fisico: lo stato diventa dipendente dai ritardi di commutazione e quindi non è prevedibile (non deterministico)*
- Questo circuito è sincrono o asincrono?
 - *Il circuito è asincrono perchè lo stato cambia ogniqualevolta S o R cambiano (e non se S o R cambiano solo in certi istanti di tempo)*

Esercizio 1

Tabella delle transizioni di stato:

S	C	Q	Q^*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

Esercizio 1

Tabella delle transizioni di stato:

Stato corrente		Stato prossimo	
S	C	Q	Q^*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

Queste sono le configurazioni vietate (stato prossimo non-deterministico).
Possiamo impostare arbitrariamente X per semplificare la funzione.
Proviamo ad implementare alcune forme canoniche.

Esercizio 1

SOP, con X=0

S	C	Q	Q_*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$$\begin{aligned} Q_* &= \neg S \neg C Q + S \neg C \neg Q + S \neg C Q \\ &= \neg S \neg C Q + S \neg C (\neg Q + Q) \\ &= \boxed{\neg S \neg C Q} + \boxed{S \neg C} \end{aligned}$$

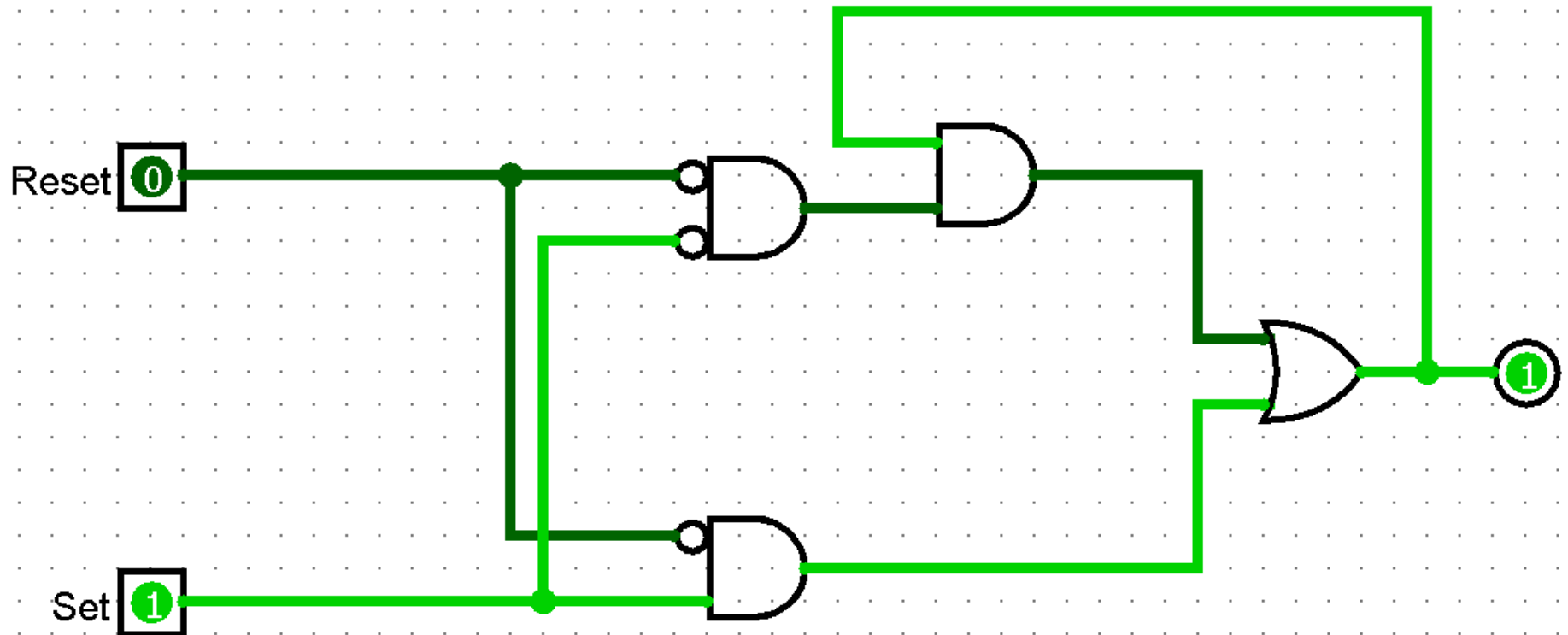
Status quo

Rivoluzione

(Raccoglimento)
(Inverso)

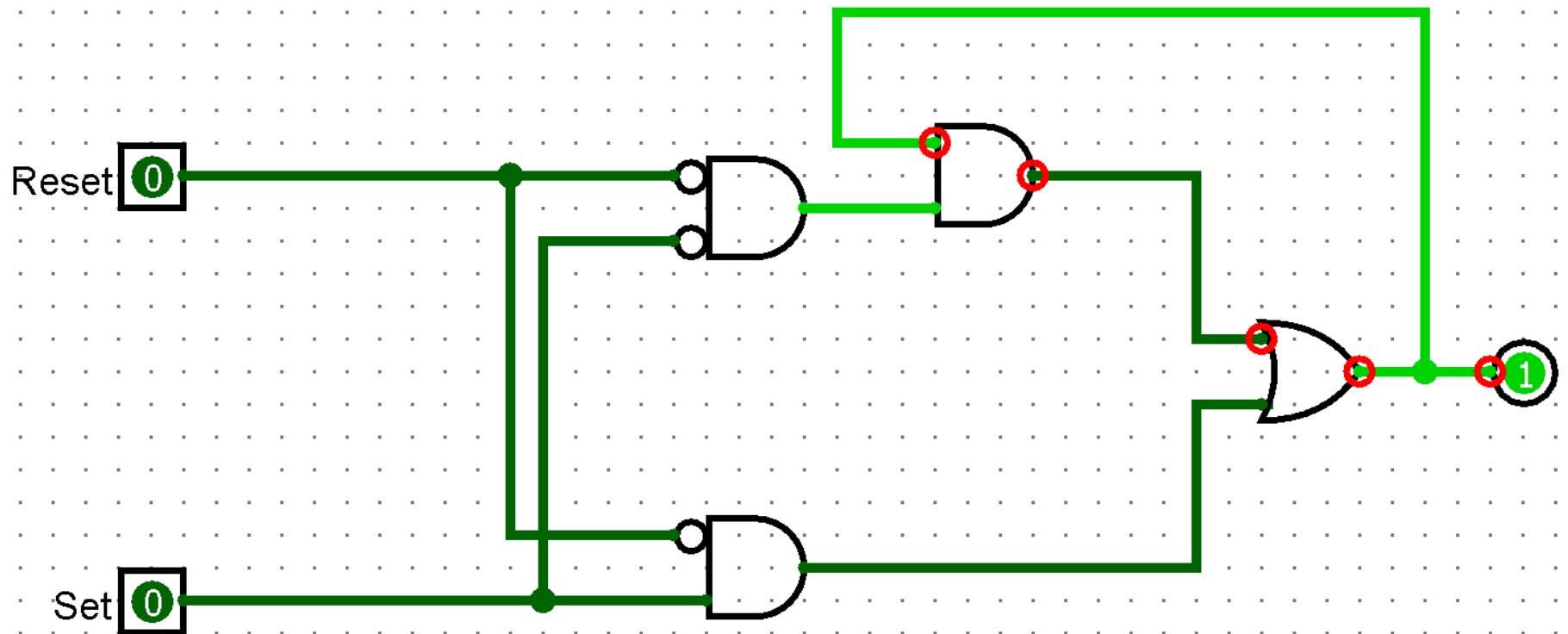
Esercizio 1

Circuito derivante dalla SOP



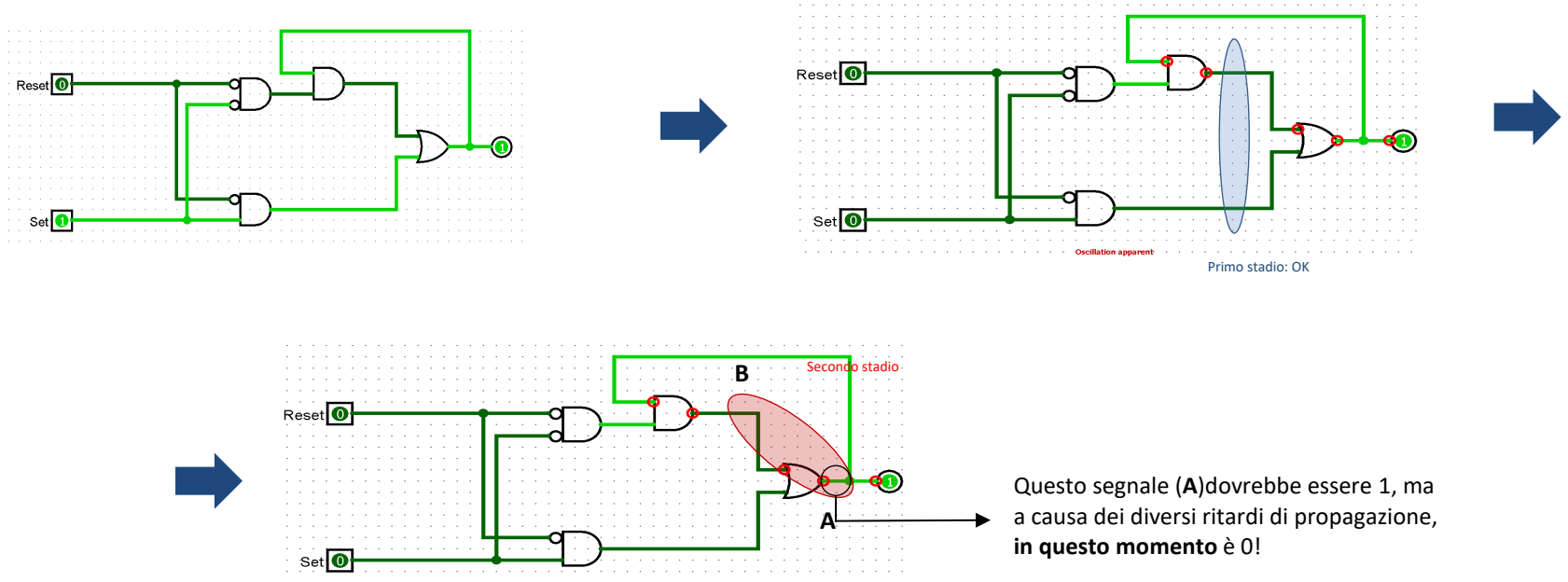
Cosa succede se lo simuliamo?

Esercizio 1



Oscillation apparent

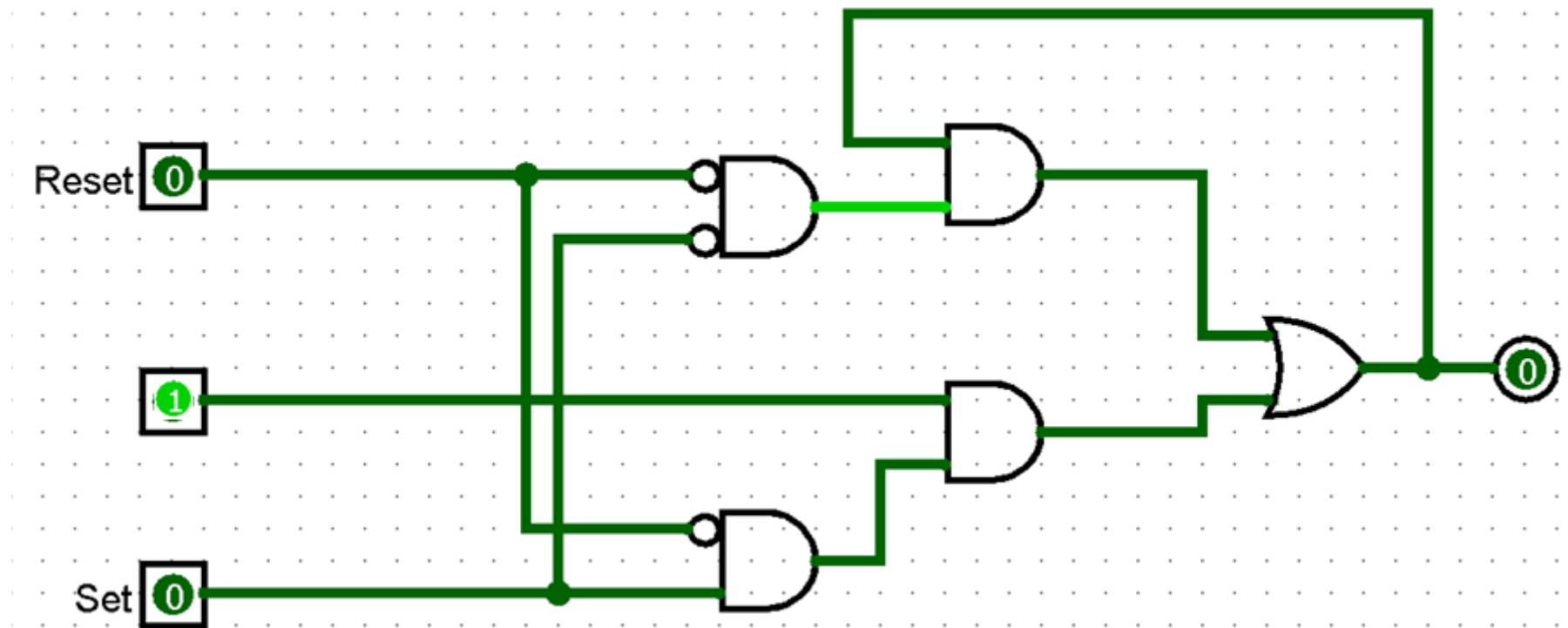
Esercizio 1



- Nella stessa iterazione in cui Logisim determina che **A** è 0, determina anche che **B** è 1
- Nell'iterazione successiva Logisim determina quindi che **A** è 1, ma anche che **B** è 0
- Il valore di A oscilla all'infinito! Come risolverlo?

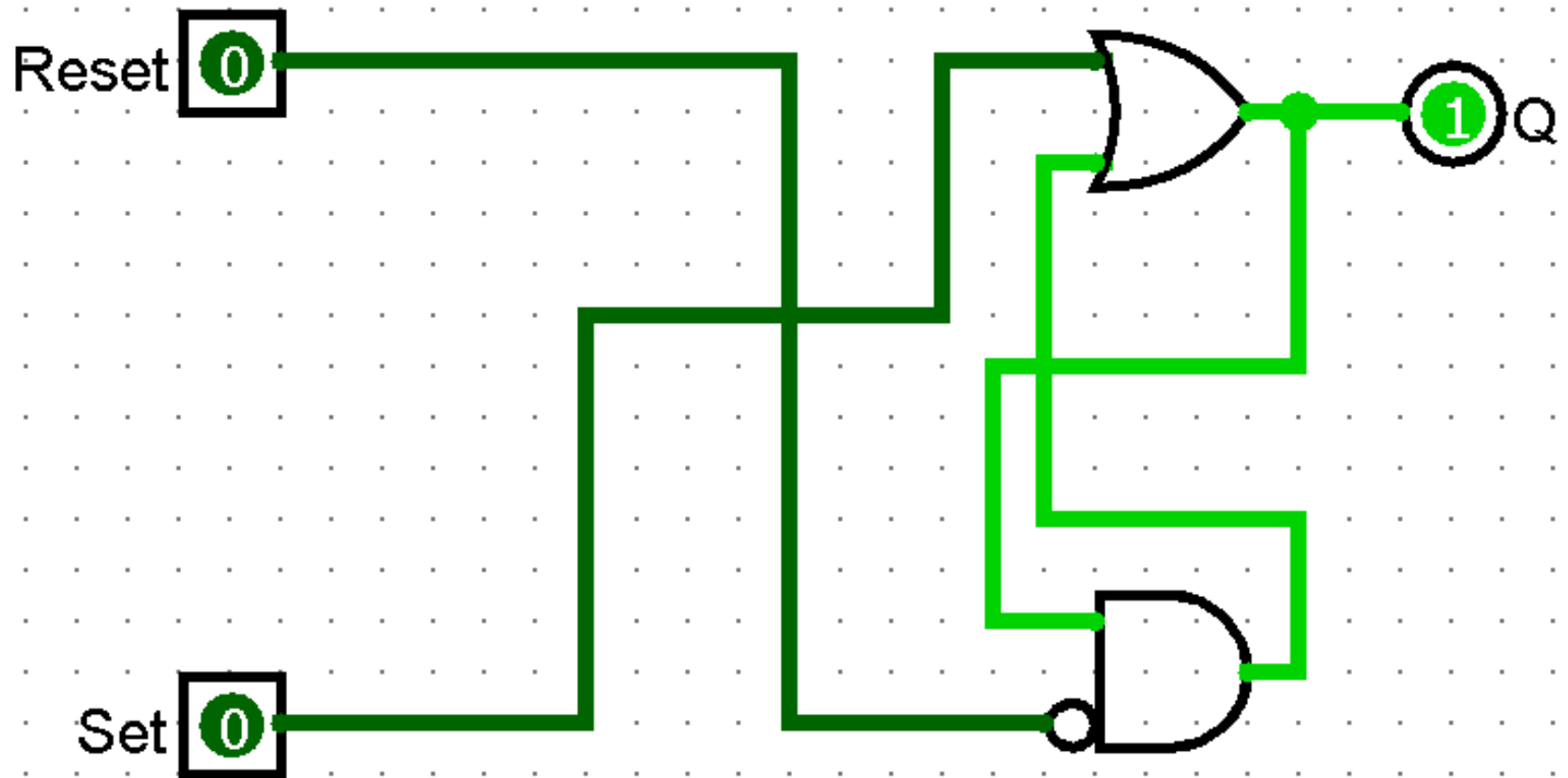
Esercizio 1

Soluzione:

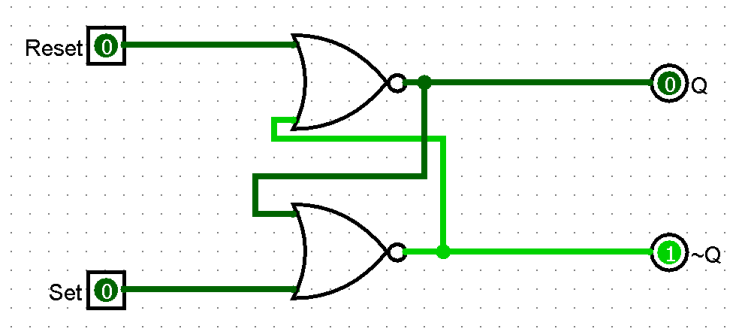


Esercizio 1

SOP, con $X = 1$

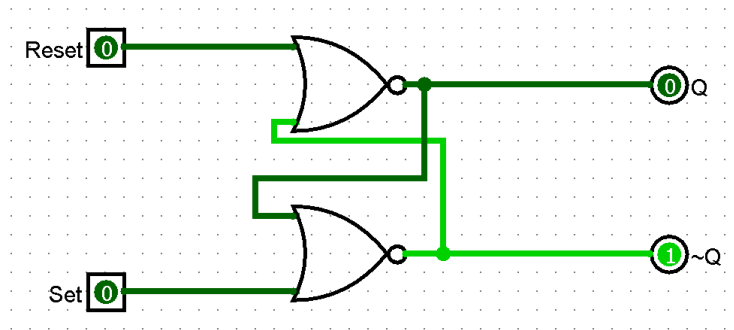


Esercizio 1

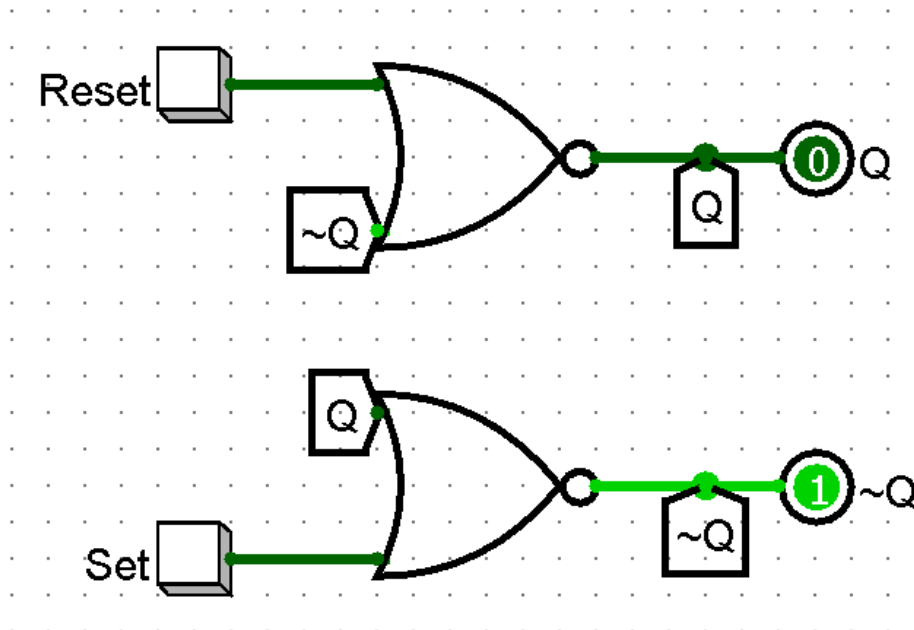


Realizziamone una versione che fa uso di bottoni e tunnel

Esercizio 1



Realizziamone una versione che fa uso di bottoni e tunnel



Esercizio 2

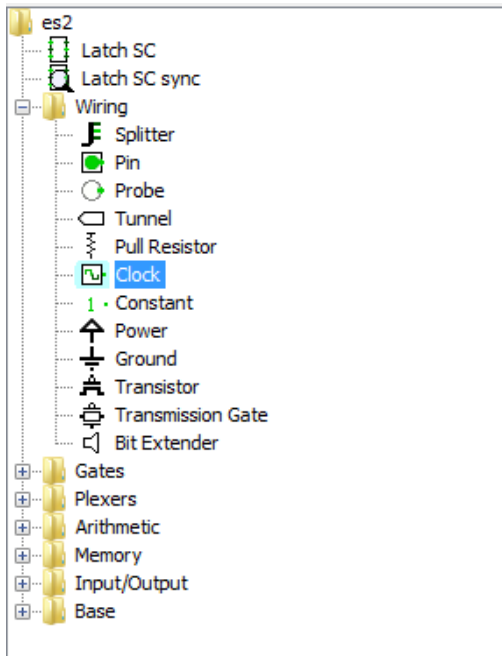
- Si aggiunga un clock (frequenza 0.5Hz) e due porte AND al circuito realizzato nell'esercizio 1 per ottenere un latch sincrono SR

Suggerimento: un latch sincrono SR è un bistabile che può commutare il suo stato solo quando il clock è nello stato alto

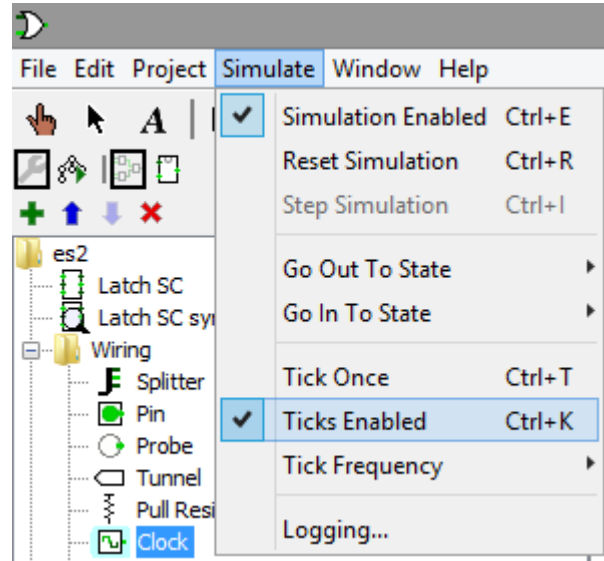
- Si osservi come cambia la risposta del circuito con diverse frequenze di clock

Esercizio 2

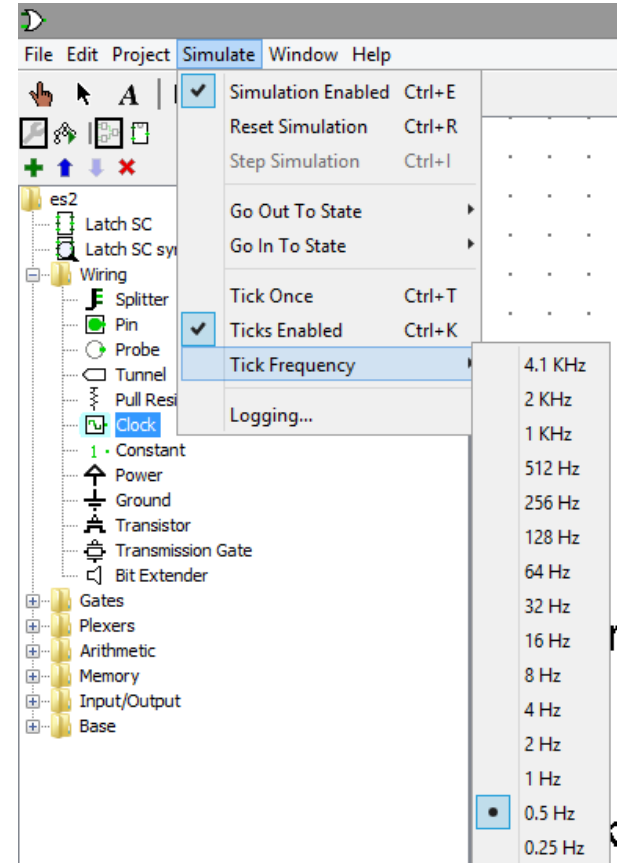
Per aggiungere il clock:



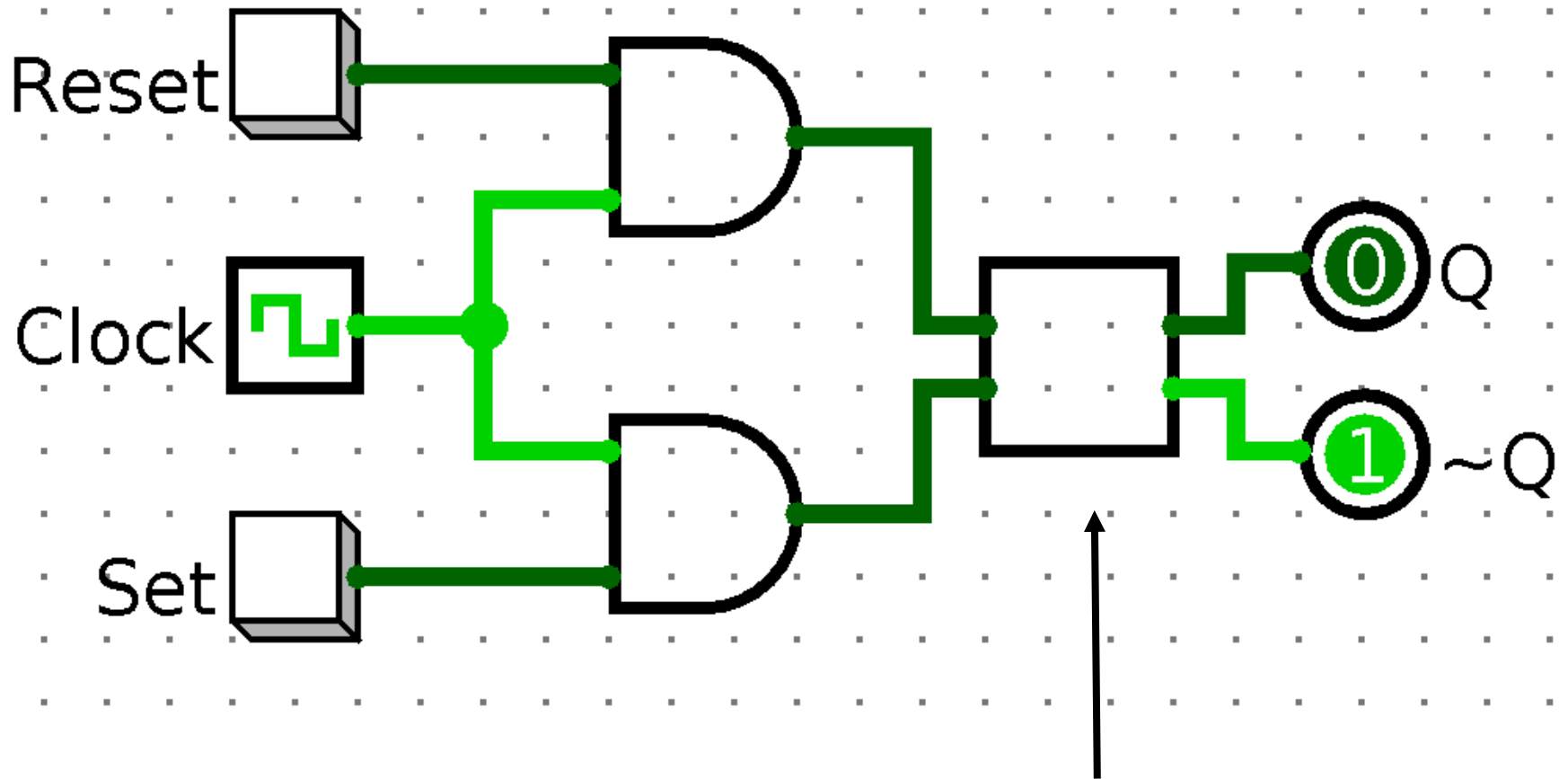
Per attivare il clock:



Per settare la frequenza di clock:

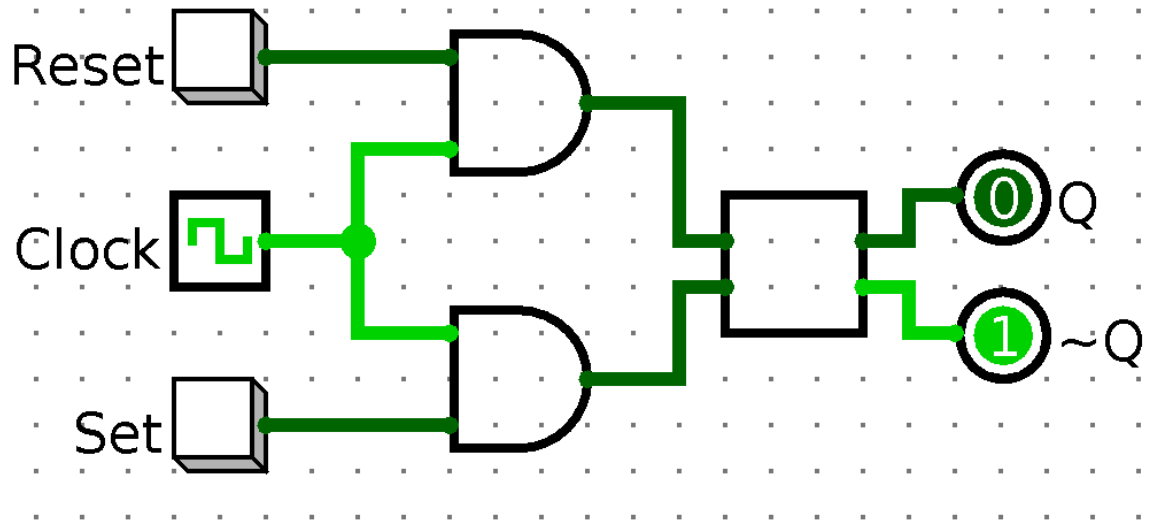


Esercizio 2



E' il bistabile SR implementato
precedentemente (versione senza bottoni)

Esercizio 2



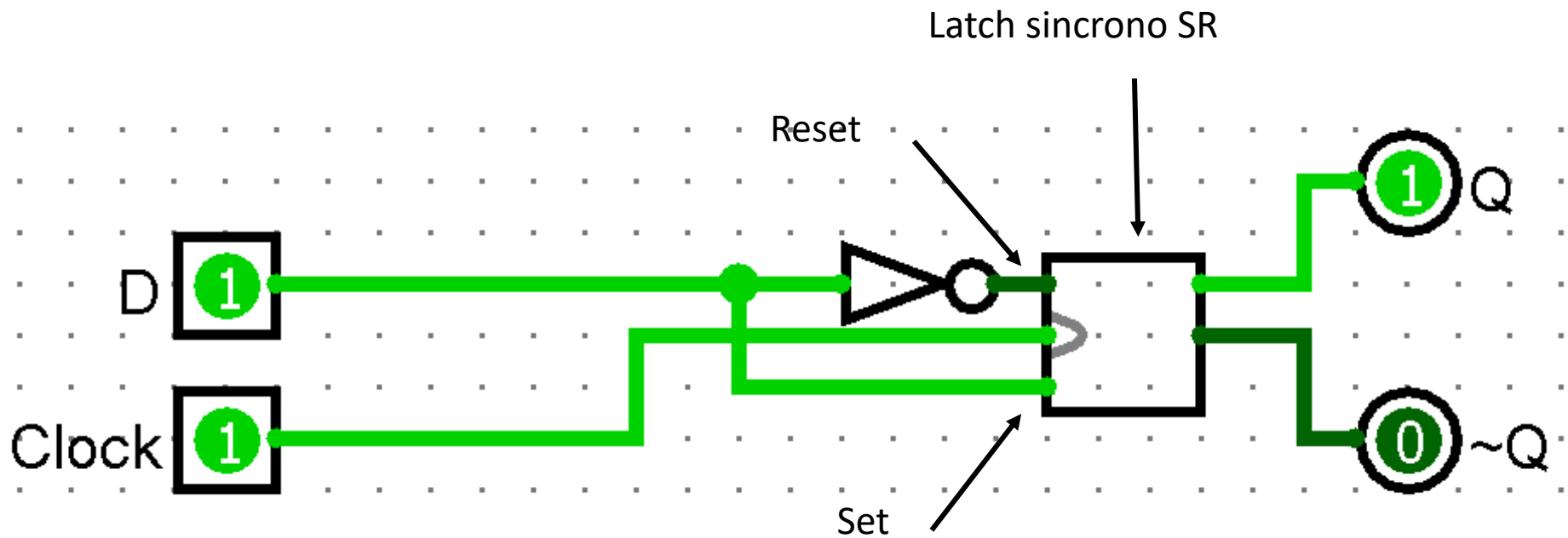
- I segnali di **Set** e **Clear** possono raggiungere il bistabile solo in certi istanti di tempo: quelli in cui il clock è alto (le porte AND agiscono da “cancelli”)
- Se aumentiamo la frequenza di clock il circuito sarà molto più sensibile ai nostri input, ma alte frequenze di clock richiedono circuiterie combinatorie asincrone veloci (i segnali devono riuscire a stabilizzarsi entro un ciclo di clock)

Esercizio 3

- Utilizzando il circuito del bistabile sincrono SR realizzato nell'esercizio precedente, si crei un latch sincrono D

Suggerimento: è presente un clock – il latch sincrono D memorizza il valore di D quando il clock è alto

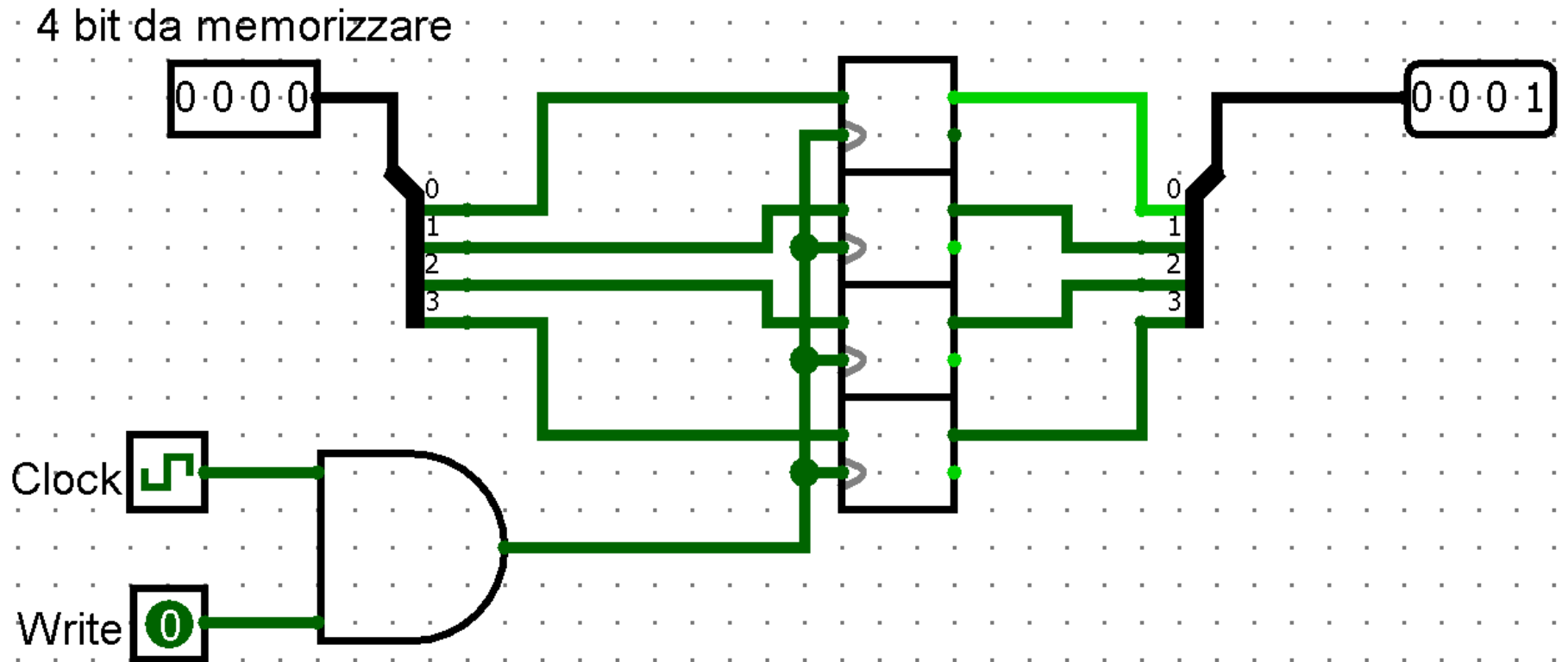
Esercizio 3



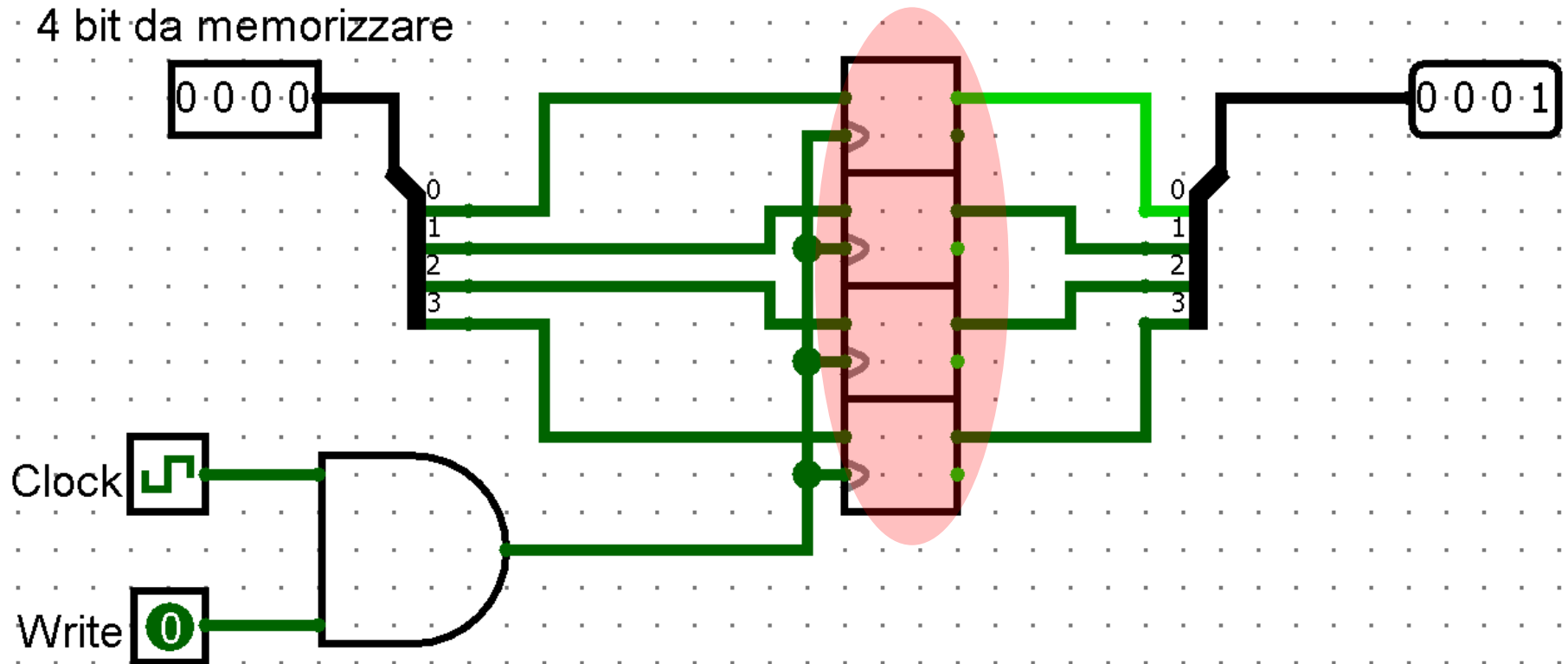
Esercizio 4

- Si utilizzino 4 latch D sincroni per realizzare un banco di memoria a 4 bit
- Tale circuito sarà caratterizzato da:
 - 4 bit in ingresso da memorizzare
 - 1 ingresso per il clock
 - 1 ingresso per il bit di write: questo bit abilita (inibisce) la scrittura nel banco di memoria quando è posto a 1 (0)

Esercizio 4



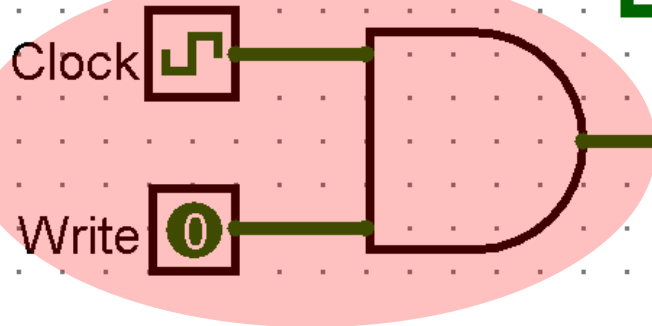
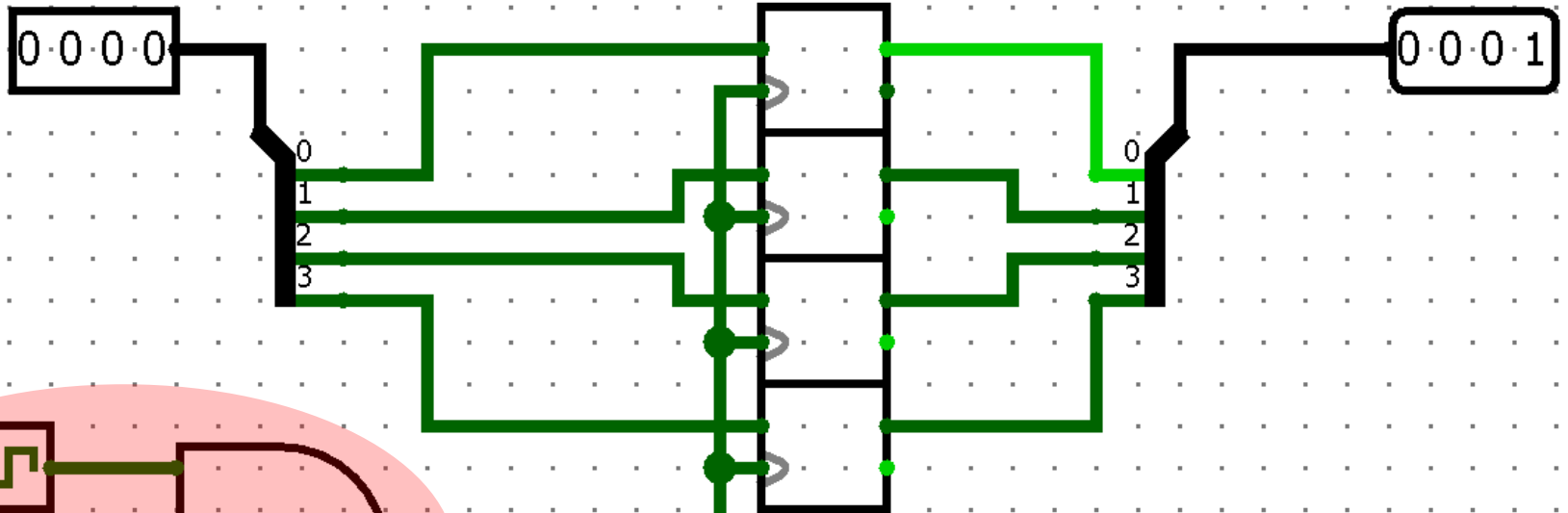
Esercizio 4



Utilizziamo 4 latch D, che permettono
l'aggiornamento dello stato quando il clock è alto e il
bit di write è pari a 1

Esercizio 4

4 bit da memorizzare

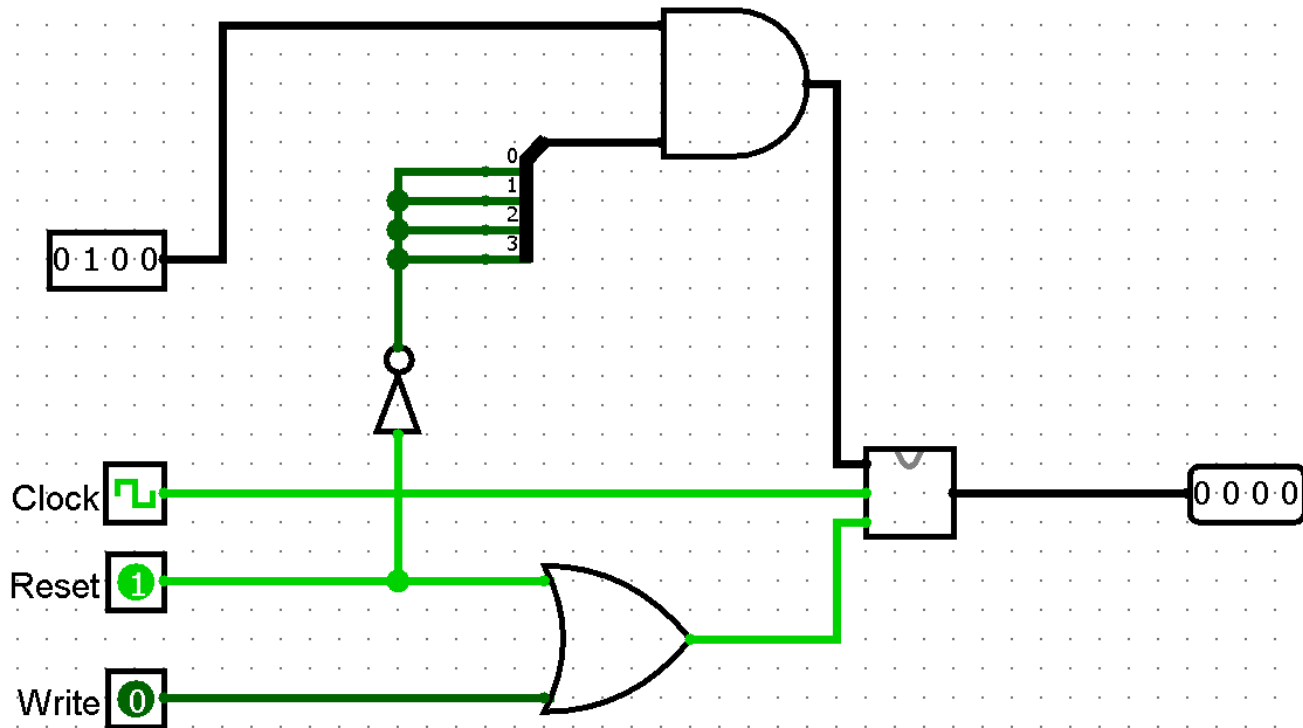


Il bit Write (che abilita la scrittura) è in AND con il clock e agisce sui latch sincroni D, per abilitare/bloccare la scrittura

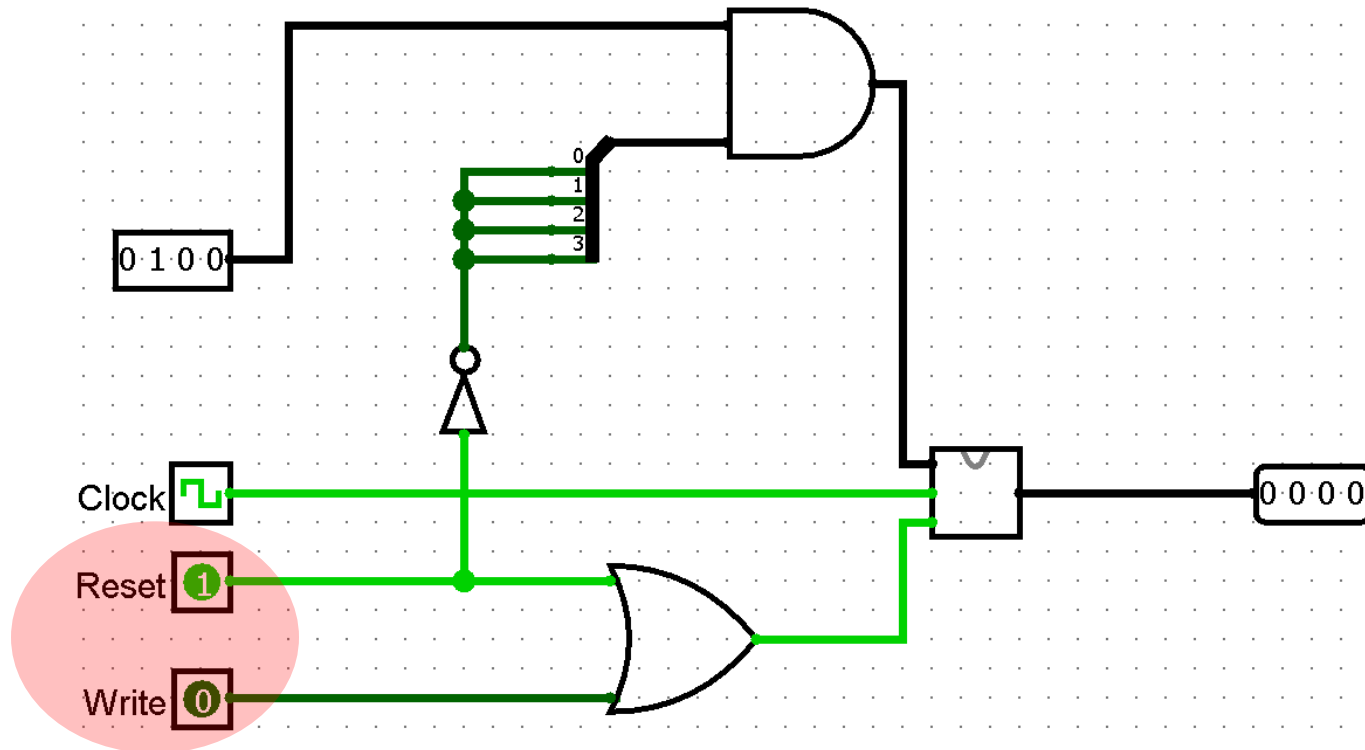
Esercizio 5

- Si modifichi il circuito realizzato nel corso dell'esercizio 4 aggiungendo 1 bit di input per l'azzeramento del contenuto della memoria.

Esercizio 5



Esercizio 5



Abbiamo due bit di selezione, Write e Reset, in OR:

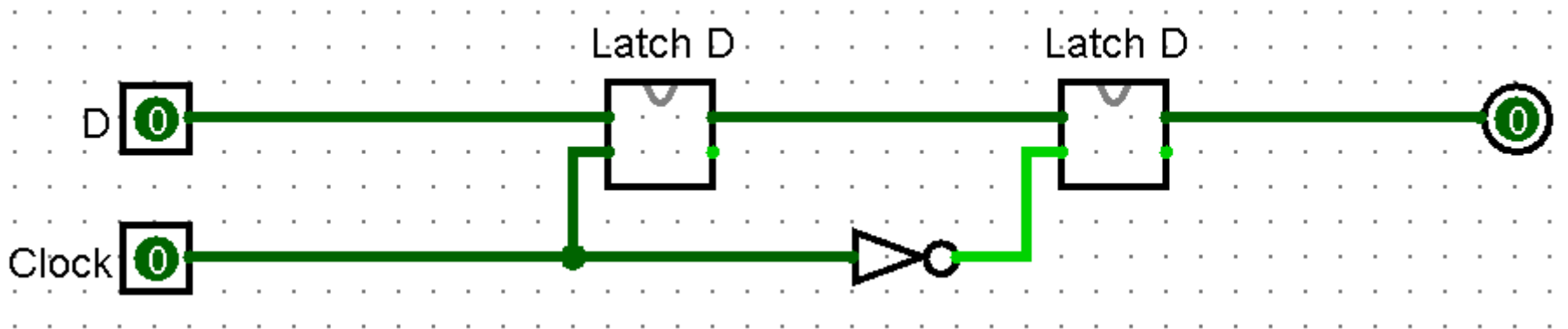
- Quando Reset è a 1 (e anche il clock è alto), allora lo stato viene azzerato
- Altrimenti, quando Write è a 1 (e anche il clock è alto) allora lo stato viene aggiornato
- Altrimenti, lo stato non viene aggiornato

Esercizio 6

- Si realizzi un flip-flop a partire da due latch D.

Esercizio 6

- Si realizzi un flip-flop a partire da due latch D.



Esercizio 7

- Si realizzi un **registro a scorrimento** (verso sinistra) con input seriale e output parallelo.

Esercizio 7

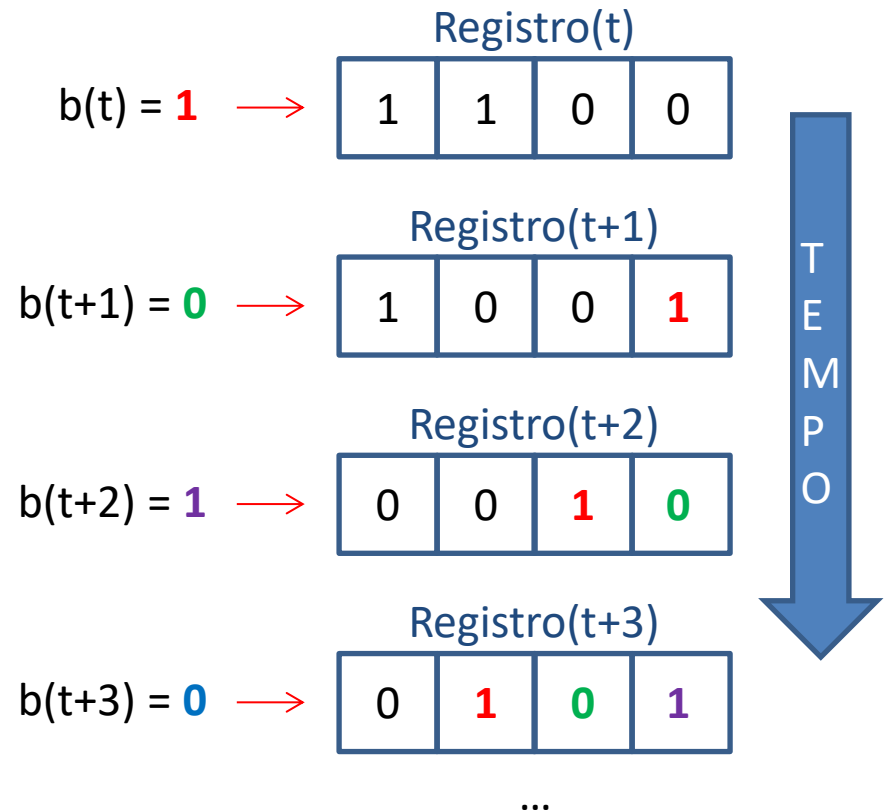
- Si realizzi un **registro a scorrimento** (verso sinistra) con input seriale e output parallelo.

Come funziona?

Ad ogni istante di tempo diamo in input un singolo bit **$b(t)$** (input seriale)

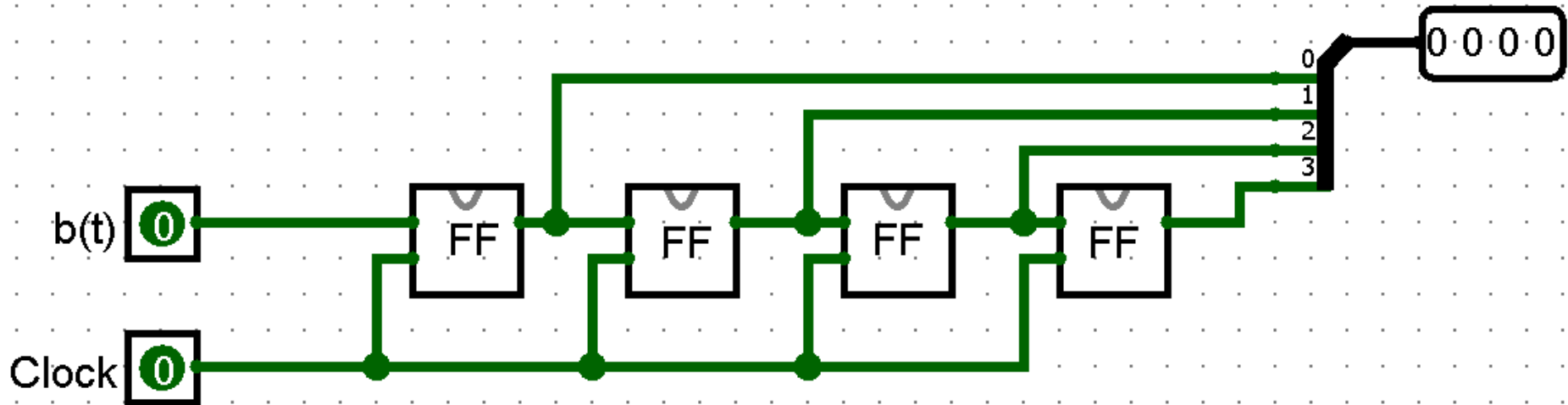
Il registro shifta a sinistra tutto il suo contenuto per fare posto a $b(t)$ e lo memorizza nel bit più a destra

I quattro bit del registro possono essere letti contemporaneamente ad ogni t (output parallelo)



Esercizio 7

- 4 Flip Flop

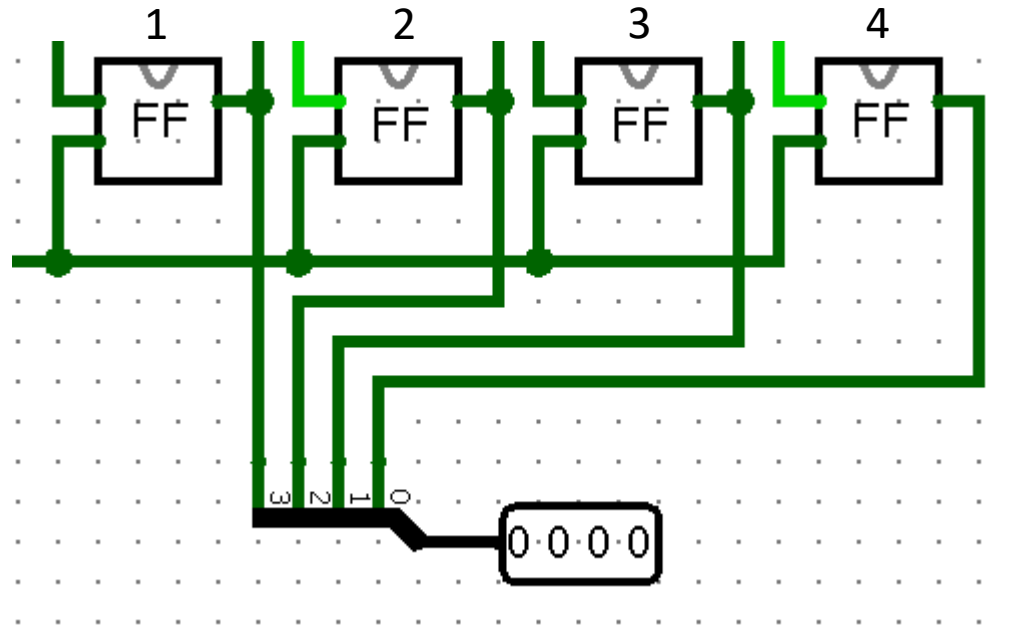


Esercizio 8

- Si realizzi un banco di memoria di 4 bit per il quale siano possibili, per ogni ciclo di clock, le seguenti operazioni:
 - Scrittura di una parola di 4 bit;
 - Reset dei 4 bit;
 - Inserimento di un bit a sinistra (con shift della parola verso destra).

Esercizio 8

- Suggerimenti:

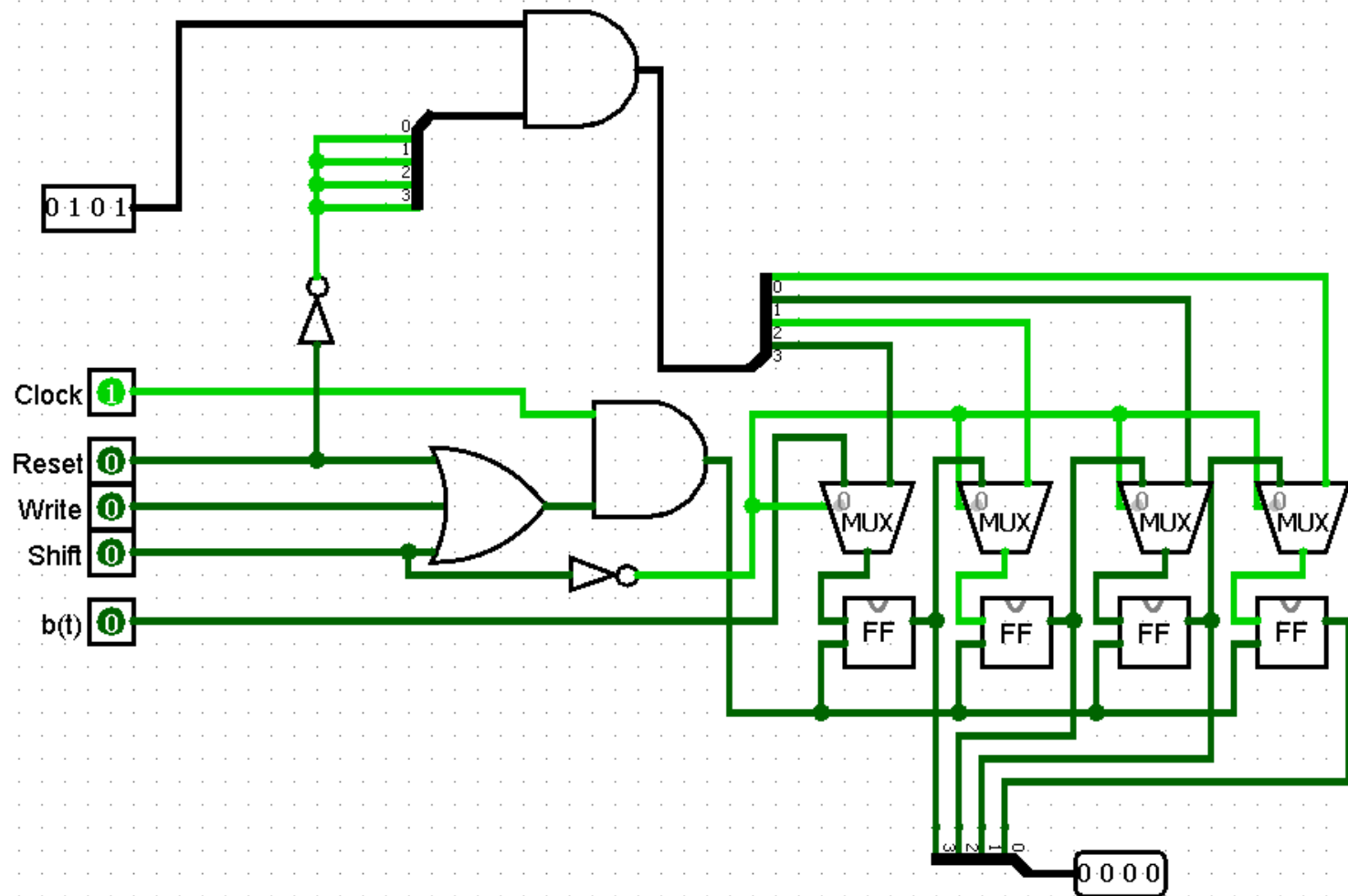


Useremo 4 Flip Flop:

- se RESET=1, memorizzeranno 0 0 0 0
- se WRITE=1, memorizzeranno la parola di quattro bit data in input
- se inseriamo a sinistra un bit **b**: FF1 memorizza **b**, FF2 memorizza FF1, FF3 memorizza FF2, FF4 memorizza FF3

Esercizio 8

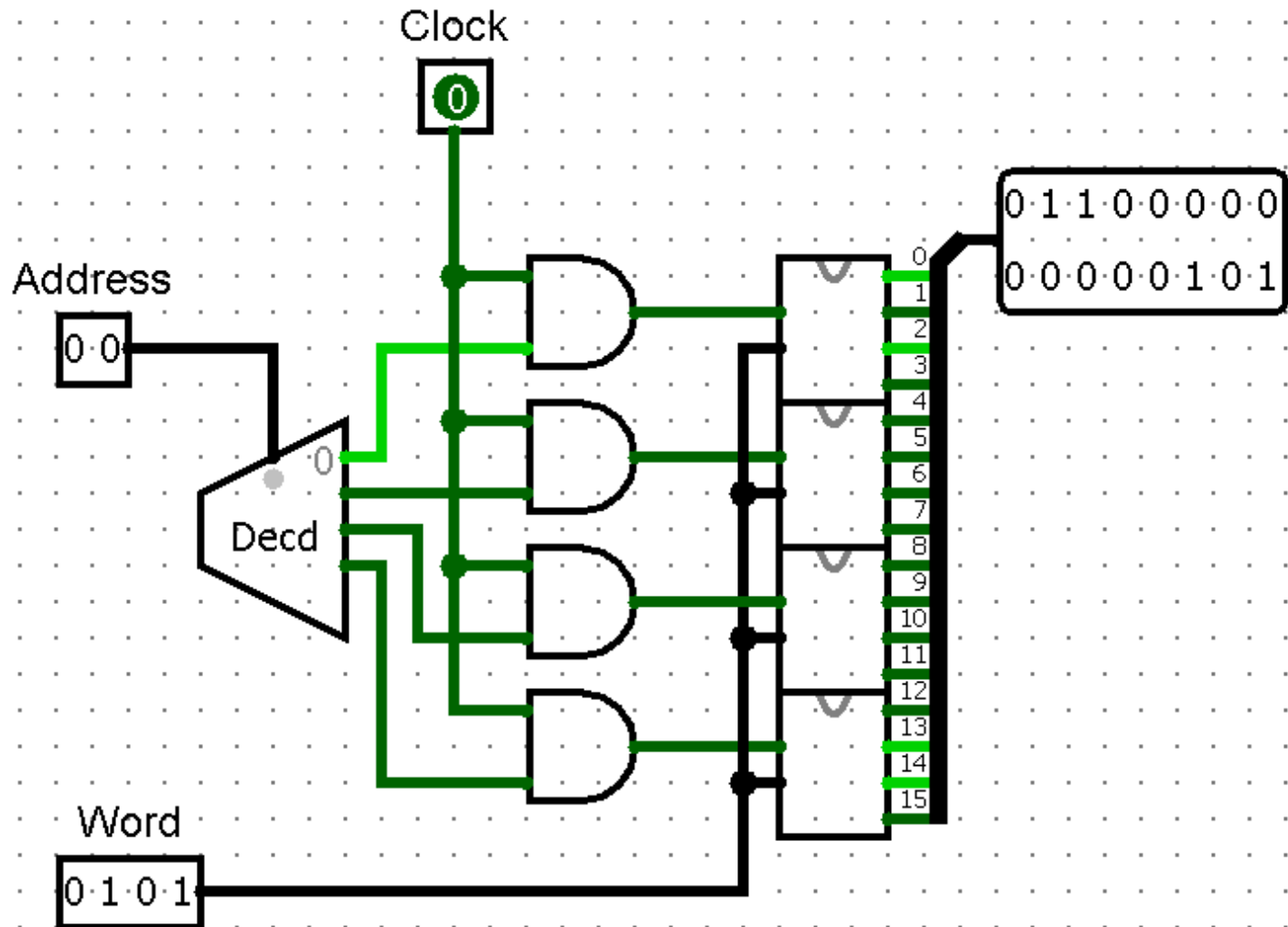
- Soluzione:



Esercizio 9

- Si realizzi un registro nel quale sia possibile scrivere una parola di 4 bit;
- Si crei poi una memoria composta da 4 parole, nella quale sia possibile scrivere una delle 4 parole a seconda dell'indirizzo dato dall'utente.

Esercizio 9



Esercizio 9

