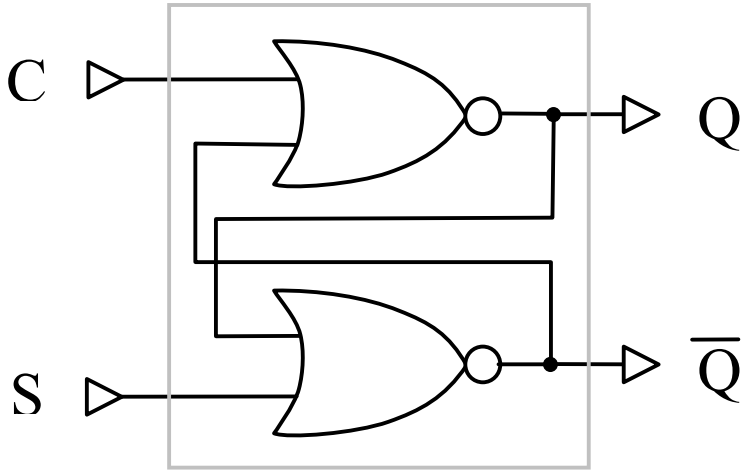
Es. 07

Bistabile asincrono SC, Latch sincrono SC, Latch sincrono tipo D, Flip-Flop sincrono D. Hold Time e Set Time, Flip-flop sincrono J-K, Flip-flop sincrono T, Flip-Flop sincrono D Master-Slave, Reti sequenziali e contatori

Es. 1 Bistabile asincrono SC

- Si implementi in Logisim un bistabile asincrono SC (o SR), utilizzando due porte NOR.
- Si definisca la tabella delle transizioni (o stato prossimo) del bistabile, si derivi la SOP, si semplifichi la SOP e si implementi il circuito corrispondente (si assuma X=0 per le due uscita indeterminate della tabella di verità).
- Si confrontino il primo ed il secondo circuito implementati in termini di tempo di transizione dell'uscita.

Sol. 1



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

*Configurazioni **vietate**:il circuito per queste configurazioni ha un comportamento indeterminato. E' possibile quindi impostare arbitrariamente il valore di queste configurazioni allo scopo di semplificare le funzioni se necessario.*

C -> Clear [annulla lo stato]
S -> Set [Reset dello stato]

Il circuito memorizza 0/1, a seconda che venga impostato ad il bit di Set o di Clear.
Il circuito è asincrono (il funzionamento non è comandato da nessun clock).

Il tempo di transizione totale è 3 [es. C -> Q, Q -> not(Q)].

Sol. 1

- Scriviamo la SOP nel caso in cui X=0.

Funzione stato prossimo: *(sviluppo SOP, X=0)*

$$Q^* = \sim S \sim C Q + S \sim C \sim Q + S \sim C Q$$

$$= \sim S \sim C Q + S \sim C (\sim Q + Q) = \sim S \sim C Q + S \sim C$$

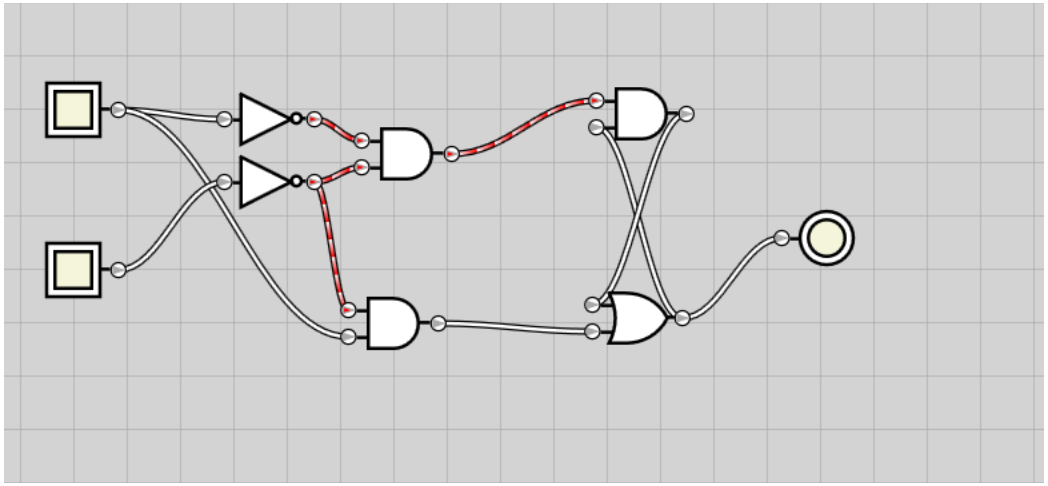
Status quo

Rivoluzione

- Lo stato prossimo va a 1 se
  - Set=0, Clear=0 e lo stato corrente è pari a 1 (nulla cambia nel circuito)
  - Set=1, Clear=0 (set a 1 dello stato).

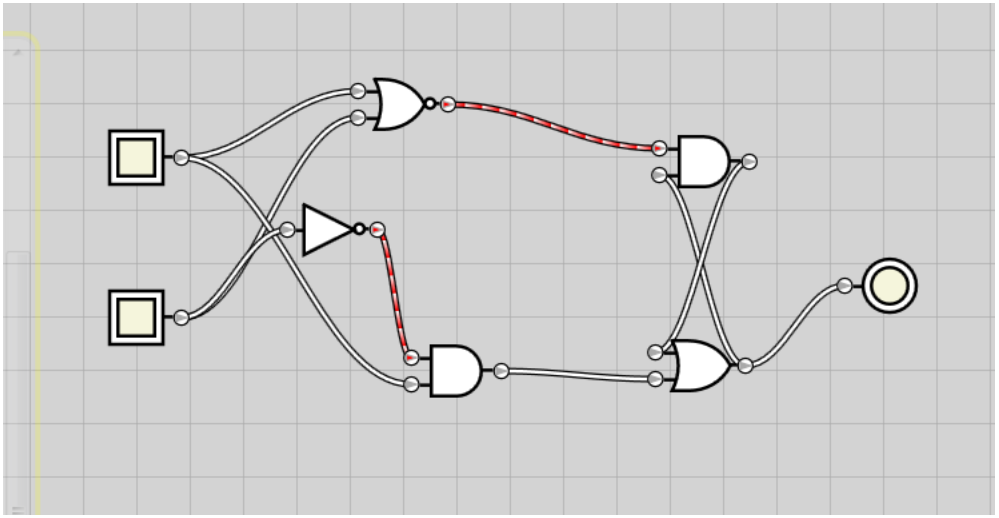
Sol. 1

- Proviamo ad implementare in Logisim il circuito corrispondente... Questo non funziona:



Sol. 1

- Modifichiamo il circuito precedente applicando DeMorgan... Questo funziona:

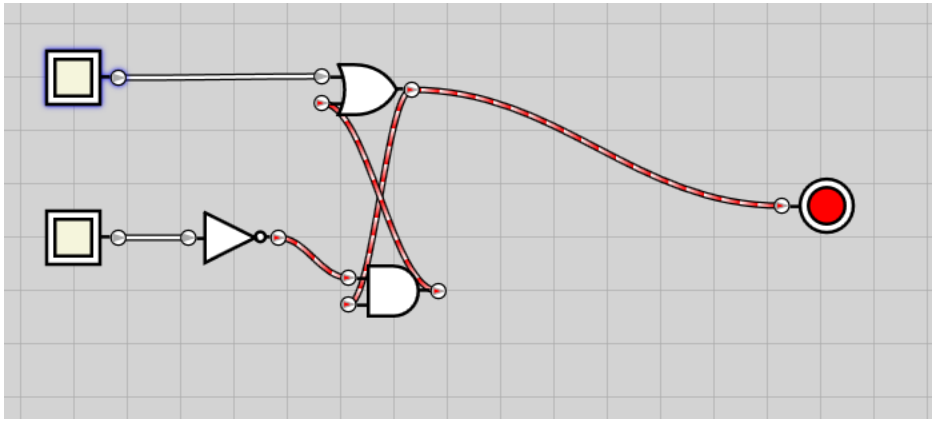


Sol. 1

- Alcune osservazioni:
  - Il circuito “originale” del bistabile asincrono SC utilizza solo due porte;
  - Gli altri due circuiti realizzati utilizzano un maggior numero di porte;
  - A causa dei ritardi di propagazione, il primo dei due circuiti “alternativi” non funziona (questo circuito “funziona in teoria, in pratica non funziona”)!

Sol. 1

- Impostando X=1,1 otteniamo:
  - Q* = C’Q+S
  - 2 porte!

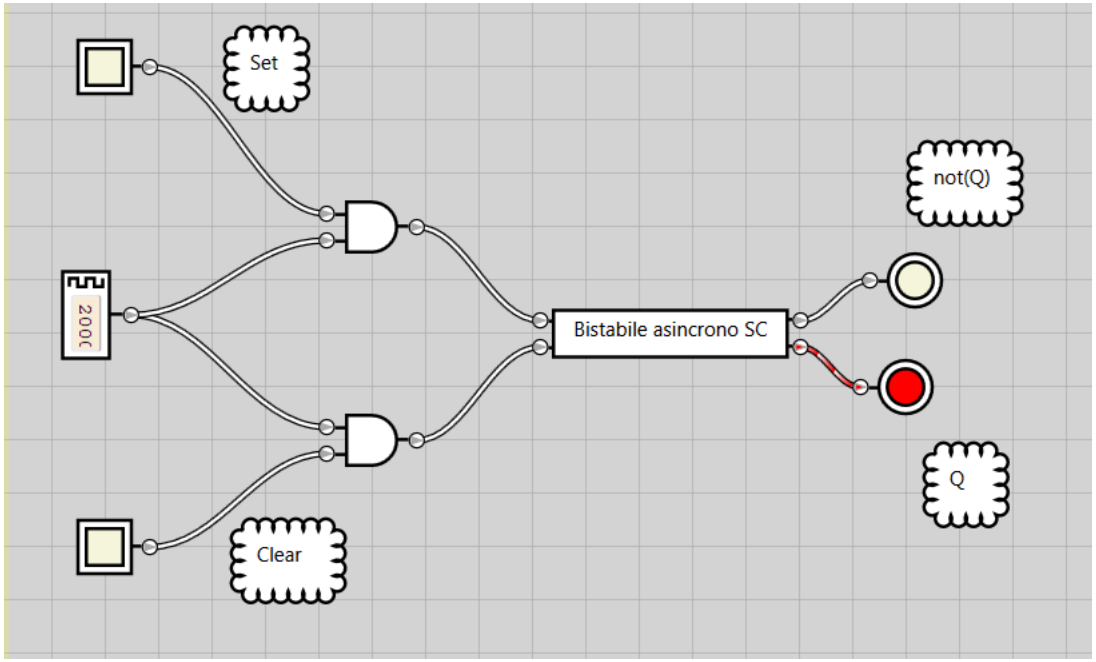


Es. 2

- Si aggiunga un clock (frequenza 0.5Hz) e due porte AND al circuito realizzato nell'esercizio 1 per ottenere un latch sincrono SC [-> ovvero un bistabile che possa commutare il suo stato solo quando il clock è nello stato alto].
- Si utilizzi l'oscilloscopio di Logisim per capire come variano i segnali S e C prima e dopo le porte AND.
- Quale è la differenza tra il bistabile SC e il latch sincrono SC?
- Si provi a lavorare in modalità veloce ed in modalità lenta (=> in questo secondo caso, il ritardo di propagazione delle porte è non trascurabile). Cosa succede nel secondo caso alle uscite del circuito? In che relazione è tale fenomeno con la frequenza di clock?

Sol. 2

- I segnali di Set e Clear possono passare attraverso le porte AND solo quando il clock è alto (le porte agiscono da cancelli);
- Di conseguenza, lo stato Q può essere aggiornato solo quando il clock è alto.
- Per il bistabile asincrono SC, invece, lo stato può essere aggiornato in qualsiasi momento.



Sol. 2

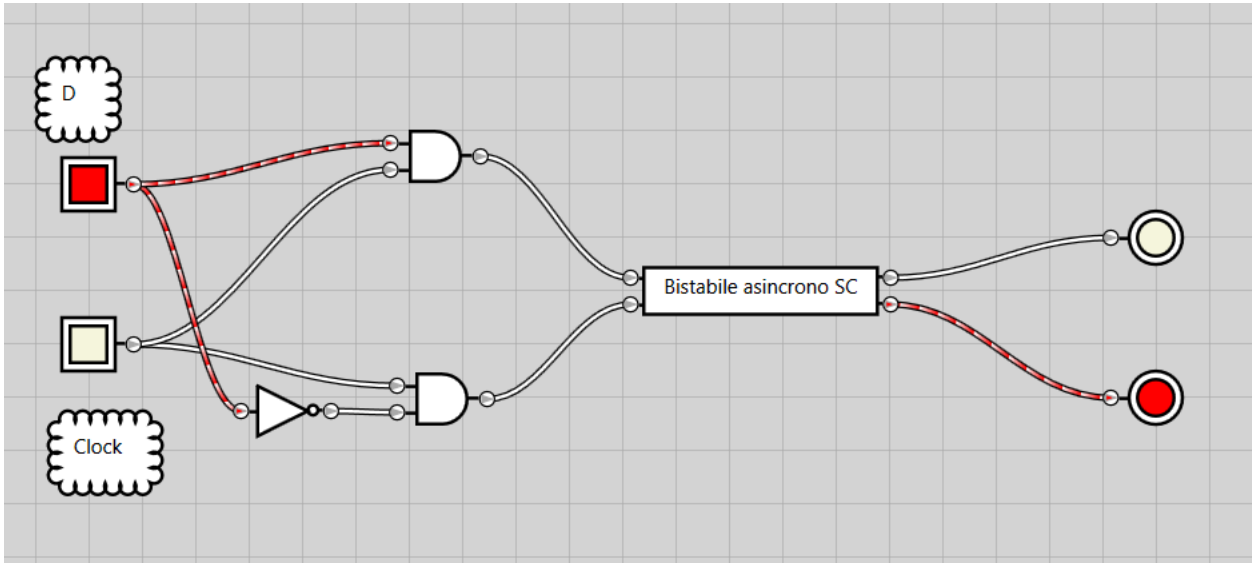
- Se aumentiamo significativamente il ritardo di propagazione delle porte (modalità tartaruga di Logisim), osserviamo delle oscillazioni non predicibili sullo stato di uscita.
- In pratica, i segnali non hanno il tempo di propagarsi dagli ingressi alle uscite (o, nel caso delle retroazioni, dalle uscite alle parti interne del circuito) prima che il clock si abbassi.
- Se la circuiteria asincrona non è abbastanza veloce -> è necessario abbassare la frequenza di clock per garantire il corretto funzionamento del circuito!

Es. 3

- Utilizzando il circuito del bistabile sincrono SC, si crei un latch sincrono D (-> è presente un clock – il latch sincrono D memorizza il valore di D quando il clock è alto).

Sol. 3

- Utilizziamo ancora due porte AND come gates. Se il valore da registrare (D) è pari a 1, si apre la porta SET. Se D è pari a 0, si apre la porta CLEAR. Le porte si aprono solo se il clock è alto.

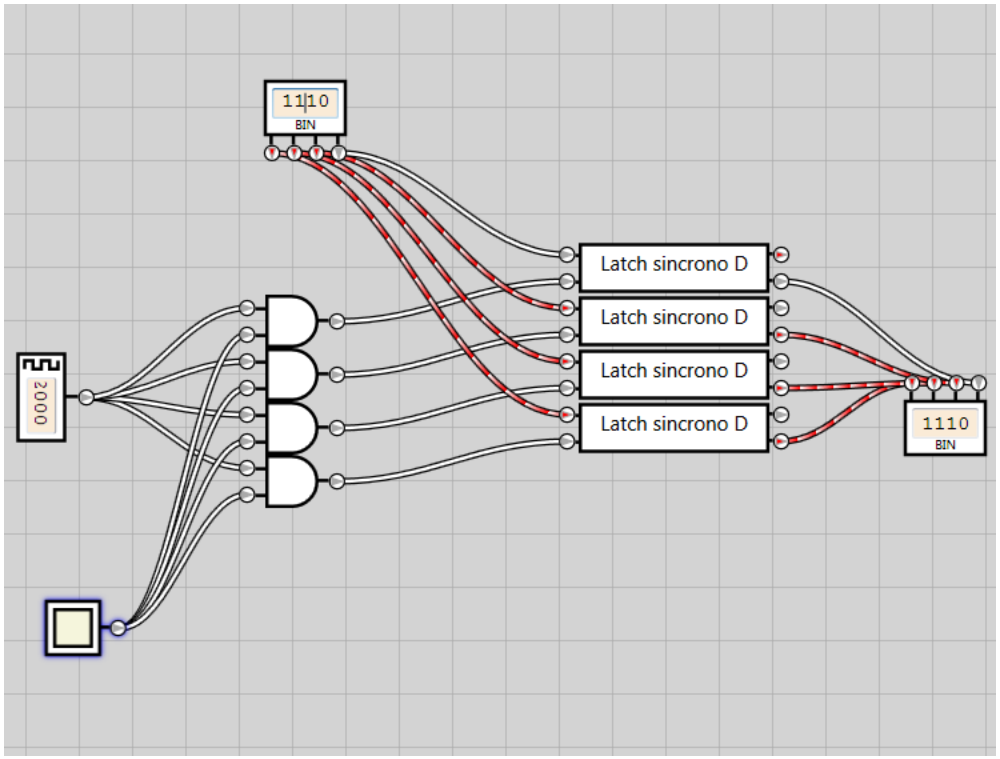


Es. 4

- Si utilizzino 4 latch sincroni D per realizzare un banco di memoria a 4 bit, che lavori alla frequenza di 0.5Hz.
- In particolare, il circuito da realizzare sia caratterizzato da 5 ingressi:
  - 4 bit da memorizzare;
  - 1 bit che abiliti la scrittura della memoria (se il bit è a zero, la memoria non può essere scritta;; se il bit è a 1, la memoria può essere scritta).

Sol. 4

- Utilizziamo 4 latch D.
- I latch D permettono l'aggiornamento dello stato quando il clock è alto...
- Utilizziamo il bit che abilita la scrittura per bloccare il clock mediante 4 porte AND.
- Il clock entra in 4 porte AND con il bit di abilitazione della scrittura (in realtà basta un'unica porta AND...).
- Le porte AND agiscono da cancelli sul clock...
- Se il clock è bloccato, la memoria non può essere scritta!

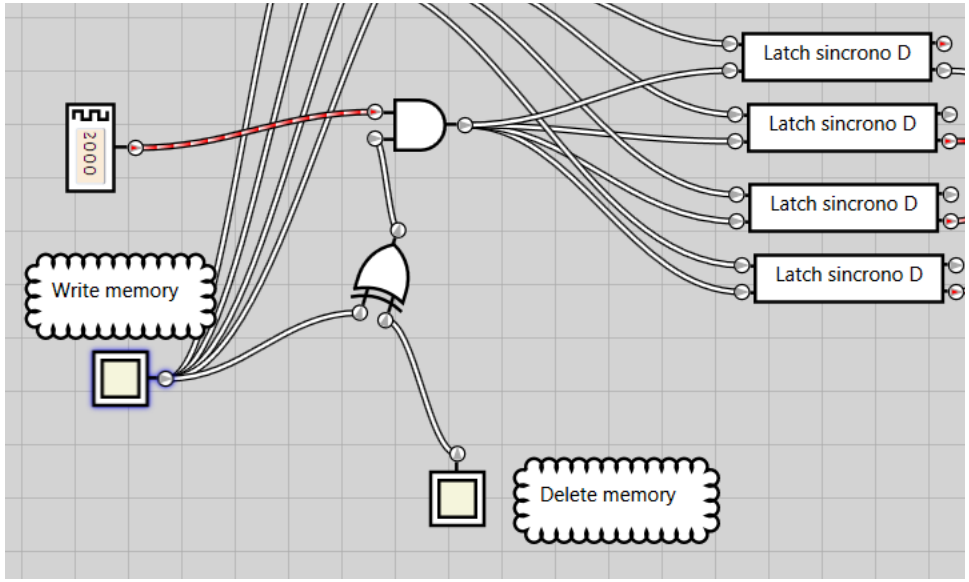


Es. 5

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

Sol. 5

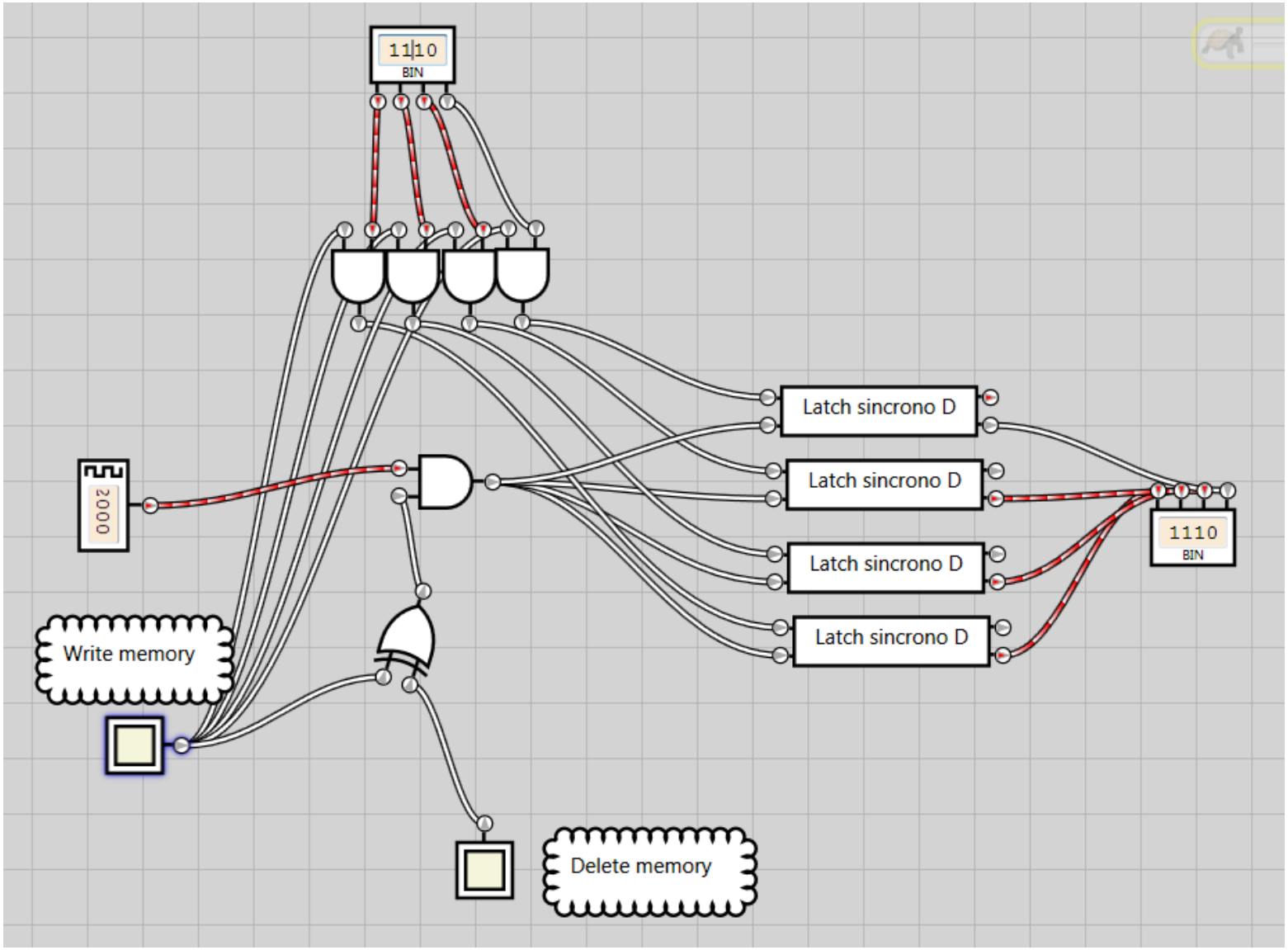
- Abbiamo due bit di selezione: write & delete.
- I due bit non possono essere attivi contemporaneamente. Usiamo una porta XOR per capire quando uno ed uno solo dei due bit write o delete è attivo. Quando questa situazione si verifica, attiviamo i clock sui 4 latch D.



Sol. 5

- Se è attivo il bit di write, nei latch D deve entrare la parola di 4 bit da scrivere. Se invece attivo delete (ovvero non è attivo il bit di write), nei latch D deve entrare una parola di 4 bit nulla.
- Se nessuno dei due bit write o delete è attivo, nei latch D può entrare qualsiasi valore (tanto il clock non arriva sui latch, quindi lo stato non verrà aggiornato).

Sol. 5

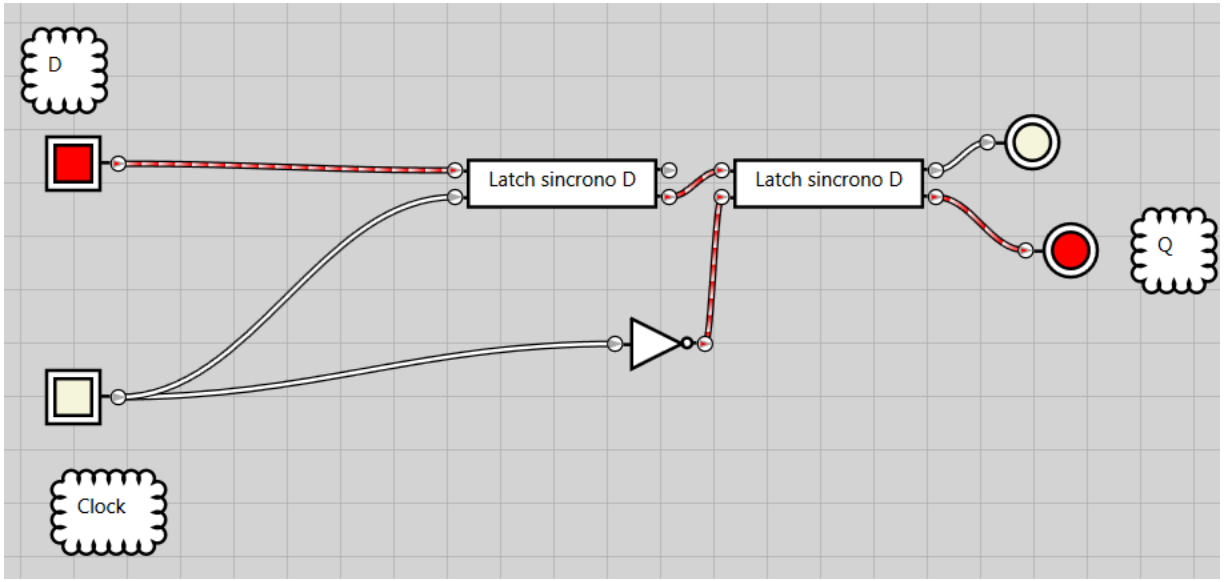

11|10
BIN
2000
Write memory
Latch sincrono D
Latch sincrono D
Latch sincrono D
Latch sincrono D
1110
BIN
Delete memory

Es. 6

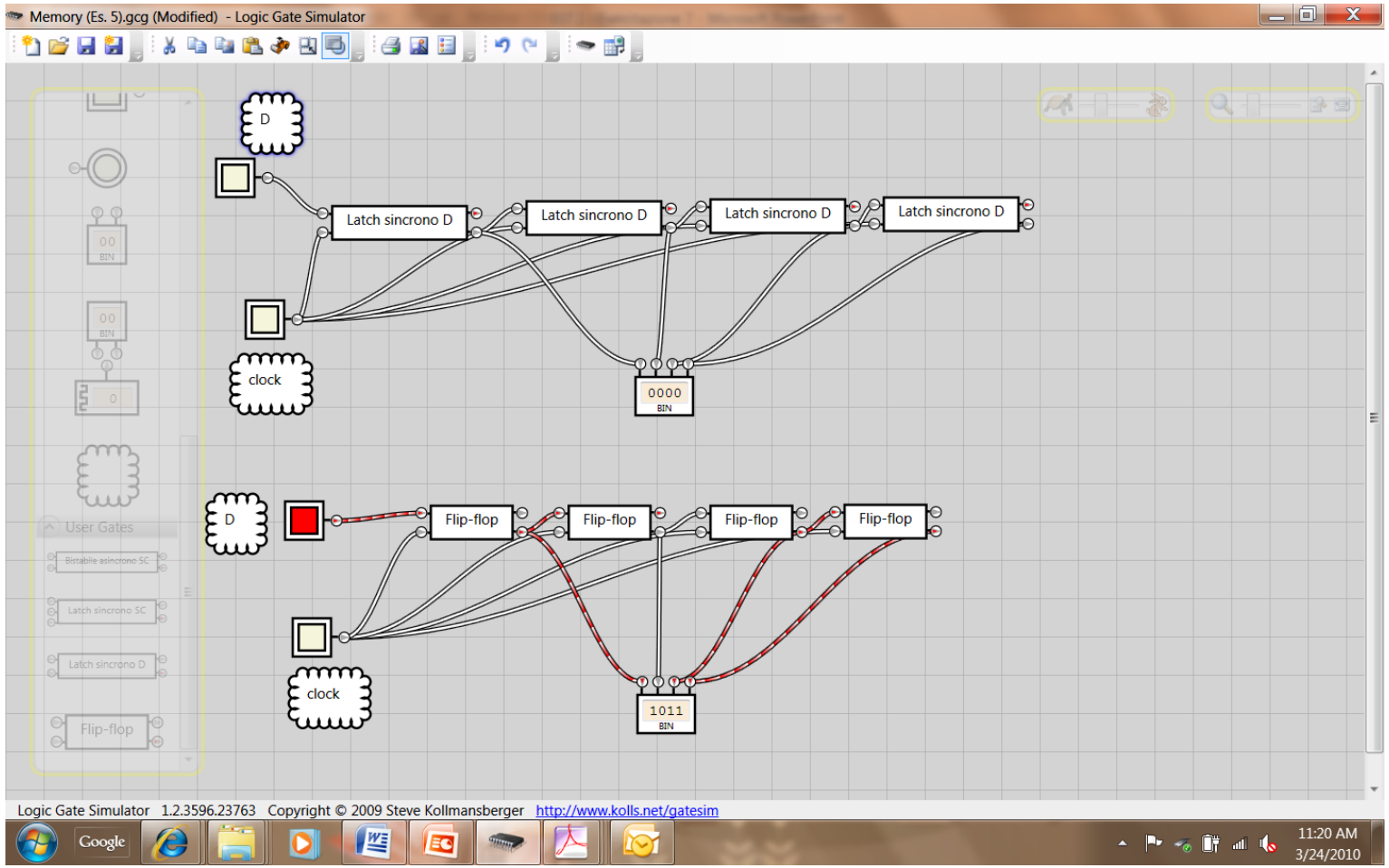
- Si realizzi un flip-flop a partire da due latch D.
- Si realizzi un circuito per lo shift a dx di una parola di 4 bit, utilizzando:
  - 4 latch D;
  - 4 flip-flop.
- Lo shift verso dx viene comandato dal clock.
- Si analizzi quindi il comportamento dei due circuiti realizzati e si spieghi perchè il secondo è preferibile al primo.

Sol. 6

- Il flip-flop si realizza collegando in serie due latch D, negando il clock in ingresso al secondo latch.
- Possiamo osservare il segnale che si propaga dall'ingresso all'uscita del primo latch D quando il clock è alto. Durante questa fase, il secondo latch D ha in ingresso un clock basso, quindi nessun segnale si progaga attraverso di esso.
- Quando il clock si abbassa, il primo latch "si chiude" mentre si apre il secondo. Il valore memorizzato si propaga sino all'uscita del circuito.
- Per la memorizzazione di un valore nel flip-flop, abbiamo quindi bisogno di un intero ciclo di clock.



Sol. 6


Memory (Es. 5).gcg (Modified) - Logic Gate Simulator
D
Latch sincrono D
Latch sincrono D
Latch sincrono D
Latch sincrono D
clock
0000
BIN
D
Flip-flop
Flip-flop
Flip-flop
Flip-flop
clock
1011
BIN
User Gates
Logic Gate Simulator 1.2.3596.23763 Copyright © 2009 Steve Kollmansberger http://www.kolls.net/gatesim
11:20 AM
3/24/2010

Sol. 6

- Nel caso dei latch D, il segnale D in ingresso si propaga verso dx alla massima velocità.
- Nel caso dei flip-flop, il segnale D si propaga verso dx alla velocità di 1 bit per ciclo di clock.

Es. 7

- Utilizzando 4 flip-flop si realizzi un banco di memoria per il quale sono possibili, per ogni ciclo di clock, le operazioni di:
  - Scrittura;
  - Messa a zero;
  - Shift verso dx di una posizione.

Sol. 7 (hint)

- Nei flip-flop deve entrare:
  - Il contenuto del flip-flop precedente, se è attivo lo shift;
  - Il contenuto della parola da scrivere, se è attivo il write;
  - Zero, se è attivo il delete.
- Possiamo creare un sottocircuito che fa passare la linea desiderata (SOP, multiplexer, ...???)
- Il clock deve essere controllato per fare in modo che arrivi sui flip-flop se e solo se uno tra write, delete o shift è attivo => circuito per il controllo che uno solo dei tre bit sia uno.

Sol. 7

- …

Es. 8

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