UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche,
Informatiche e Matematiche

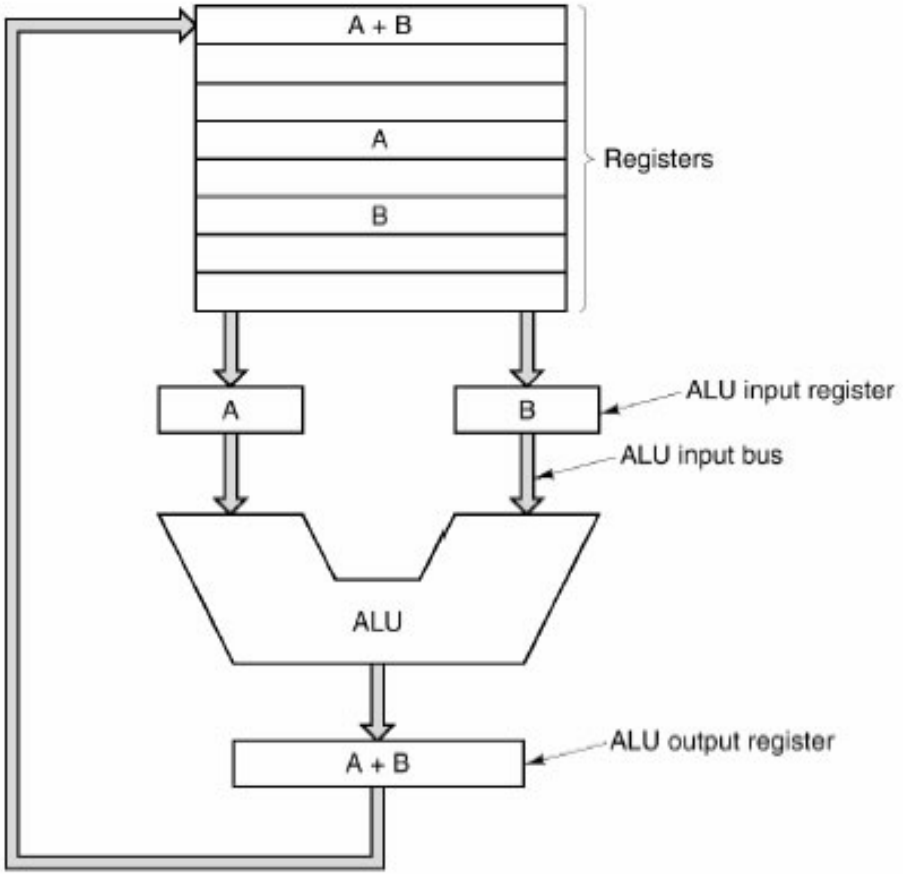
Esercitazione 3: Memorie e Registri

Architettura dei calcolatori [MN1-1143]

Corso di Laurea in Ingegneria Informatica
(D.M.270/04) [16-215]
Anno accademico 2020/2021

Dott. Gianluca Brilli
gianluca.brilli@unimore.it
Prof. Marko Bertogna
Marko.bertogna@unimore.it

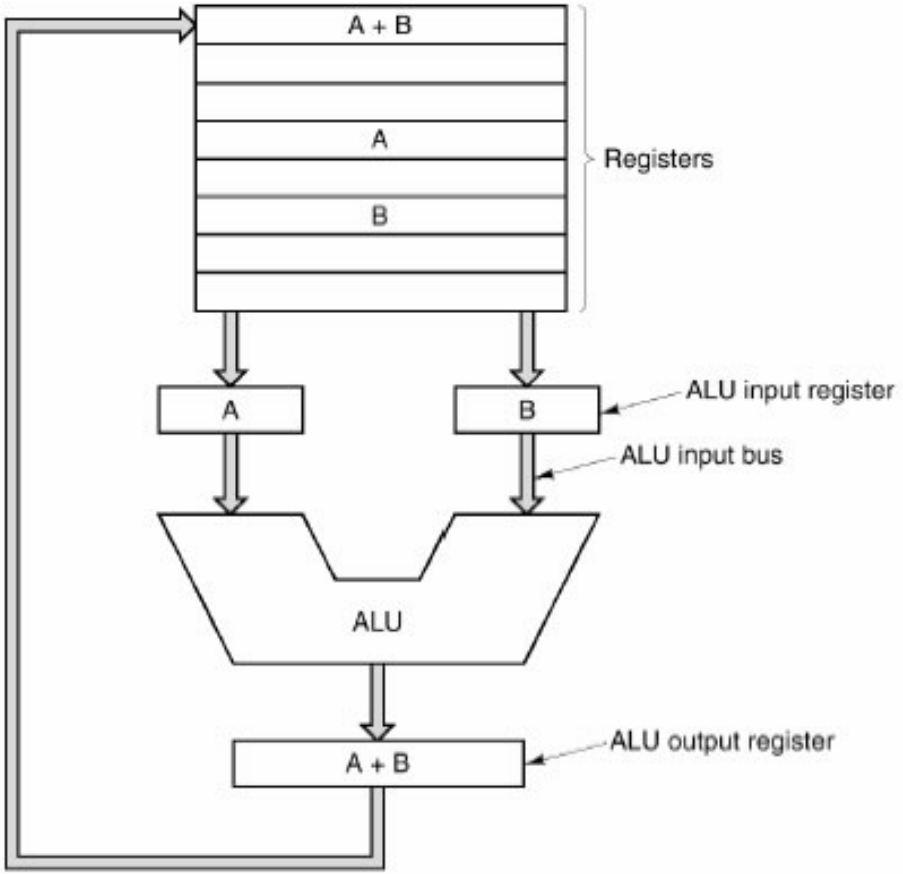
Obiettivi - ALU



›Unità Aritmetico-Logica.

›Vista nel blocco di esercitazioni precedente

Obiettivi - ALU



›Vedremo nel dettaglio come progettarli.

›Ne studieremo il funzionamento in maniera approfondita.

Bistabile SR

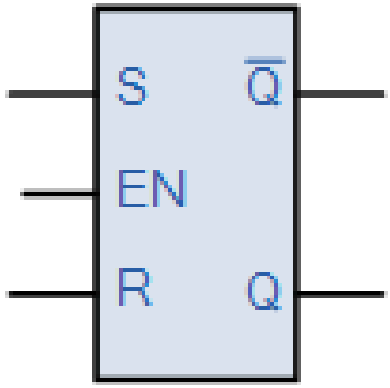
›Creare un nuovo progetto su Logisim e realizzare un **Bistabile SR** come sottocircuito.

S	R	Q	$\bar{Q}$
0	0	Hold state	
0	1	0	1
1	0	1	0
1	1	Prohibited state	

Esercizio 1

Latch SR

›utilizzando il Bistabile SR a NAND, realizzare un **Latch SR** dotato di tre ingressi: **Set** (S), **Reset** (R) ed **Enable** (E). L'ingresso di enable consente di abilitare o meno l'azione imposta sui due ingressi.



Esercizio 3
Latch D

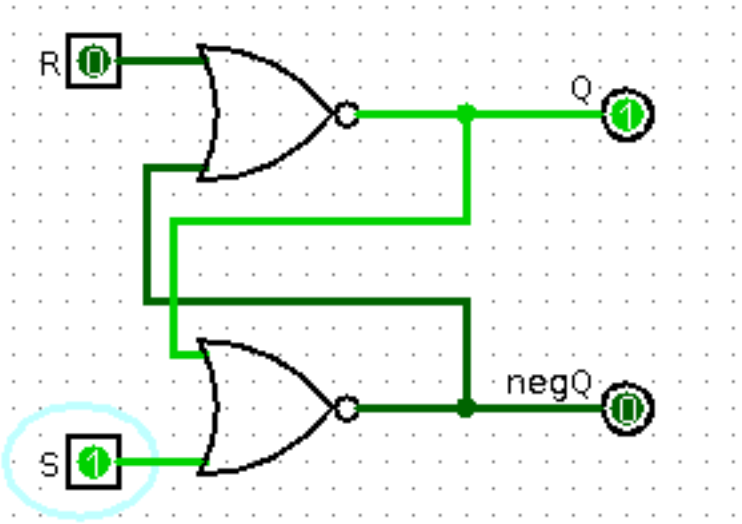
›Partendo dal circuito del Latch SR realizzato nel precedente esercizio, andiamo a costruire un **Latch D**.

›**Suggerimento**: In questo caso a differenza del Latch SR, abbiamo un solo **ingresso D** (D = Data).

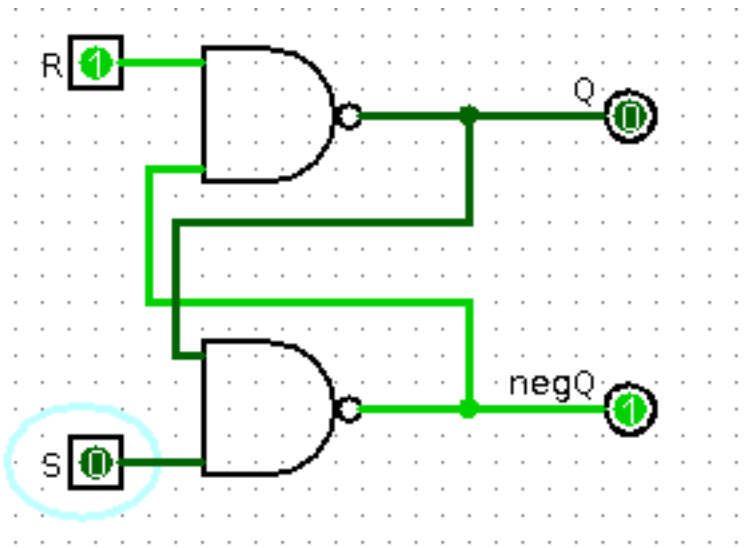
›Se D è settato deve imporre S = 1 e R = 0 e viceversa.

Bistabile SR

›Sintesi a NOR:



›Sintesi a NAND:



Esercizio 2
Latch SR

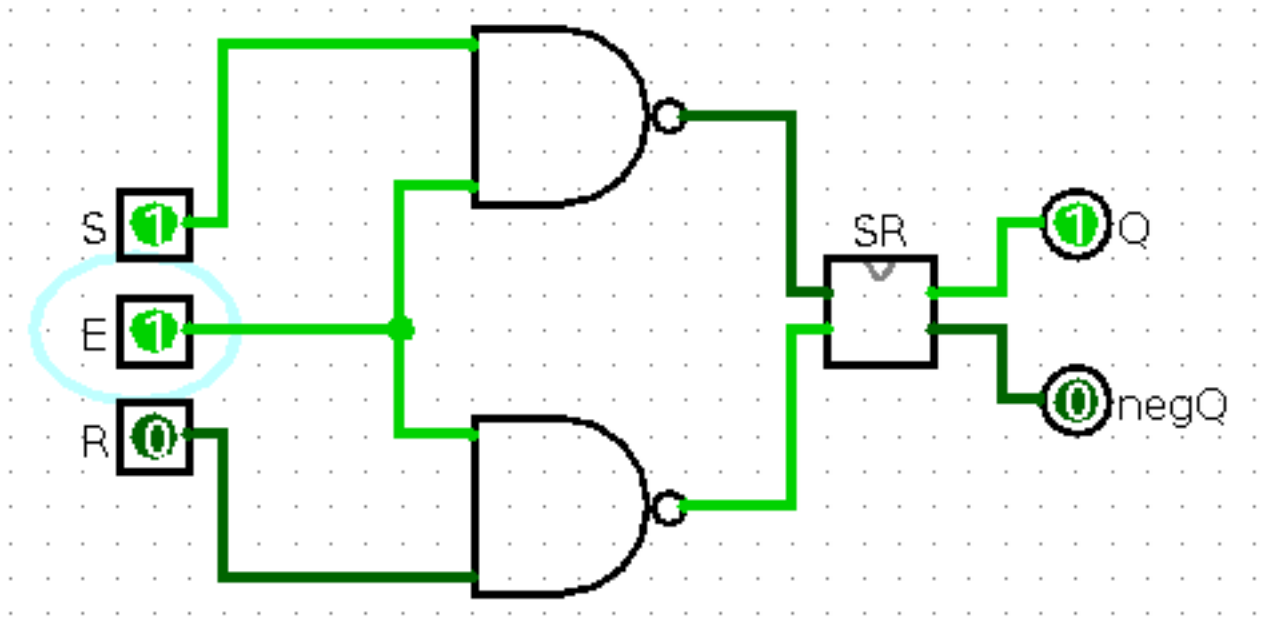
›**Aggiungiamo Funzionalità** al nostro Latch SR:

1) **Clock**: Circuito che fornisce un'oscillazione continua 0 → 1 ad una frequenza definita.
Andiamo a sostituire l'Enable (E) con un ingresso di **Clock** (CLK).

2) **Segnale di Reset**: Segnale fondamentale per ogni tipologia di rete logica sequenziale, permette di portare il circuito ad uno stato noto.

Esercizio 1 - Soluzione

Latch SR

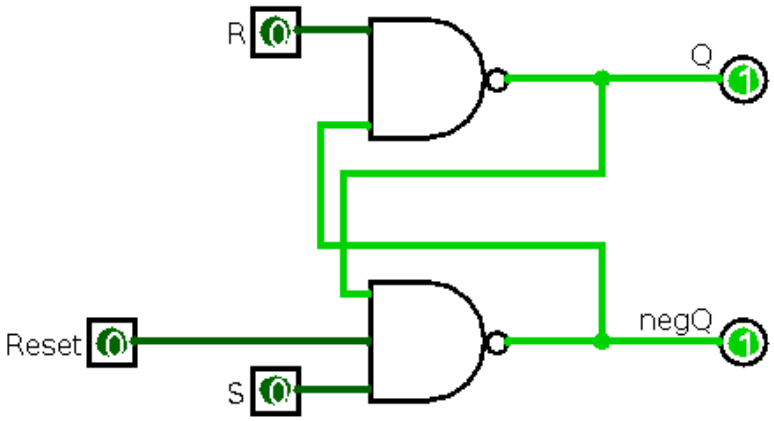


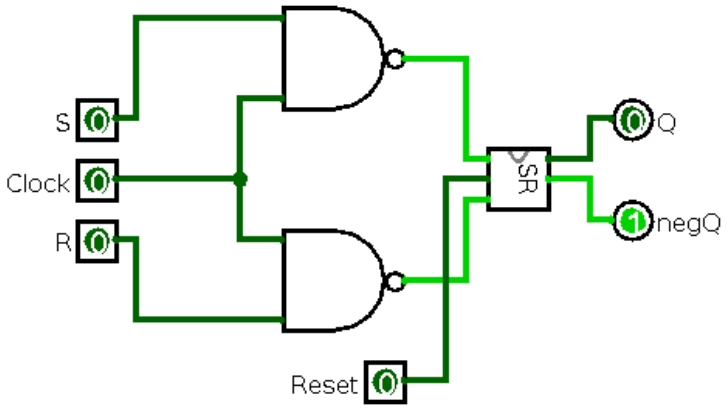
›Notiamo che se E = 0, le modifiche imposte su S e R non hanno effetto.

Esercizio 2 - Soluzione

Latch SR con reset

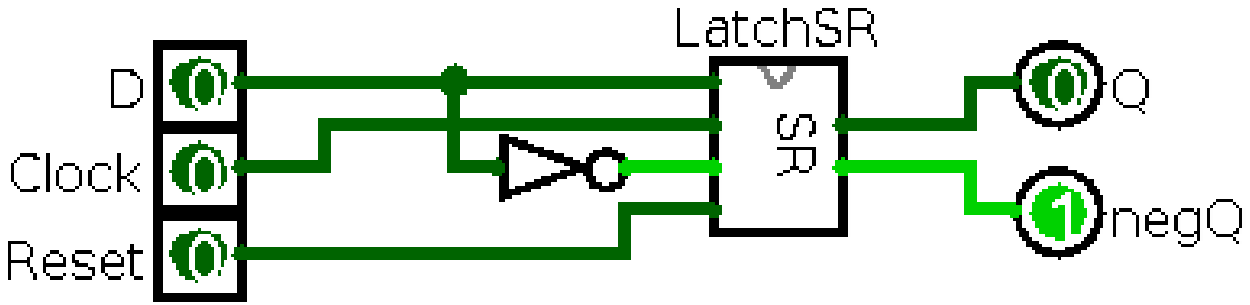
›Testiamo un po il funzionamento e dopo integriamo la modifica del **Reset** nel **sottocircuito del Bistabile SR**, così lavoriamo in maniera più modulare.





Esercizio 3 - Soluzione

Latch D



Esercizio 3 - Soluzione

Latch D

- **Funzionamento**: Supponendo di avere

  D = 1

  Q = 0

  CLK = 1Hz.

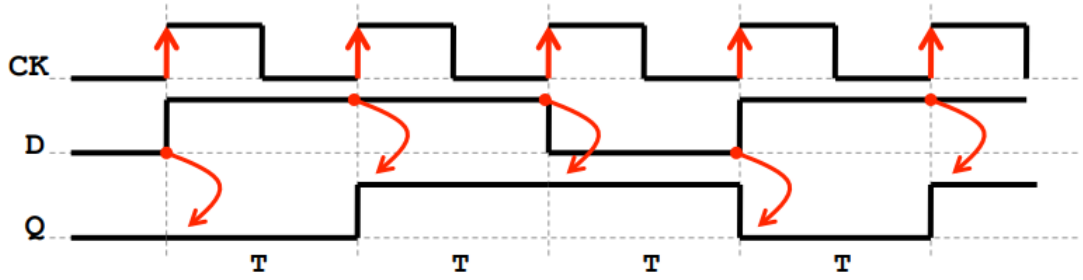
  - Dopo un mezzo periodo di Clock ingresso viene scritto sull'uscita.

  - In questo esempio: supponendo di aver scritto D = 1 all'inizio del fronte basso di clock, dopo mezzo secondo D è trasferito su Q.

Esercizio 4
Flip-Flop D

›**Nota**: Nel Latch D, le modifiche scritte su D agiscono su tutto l'impulso di clock.

›Realizzare un **flip-flop D** con **commutazione sul fronte ascendente di clock**, (Rising edge-triggered).



›**Suggerimento**: Pensate come sfruttare più Latch D per risolvere il problema.

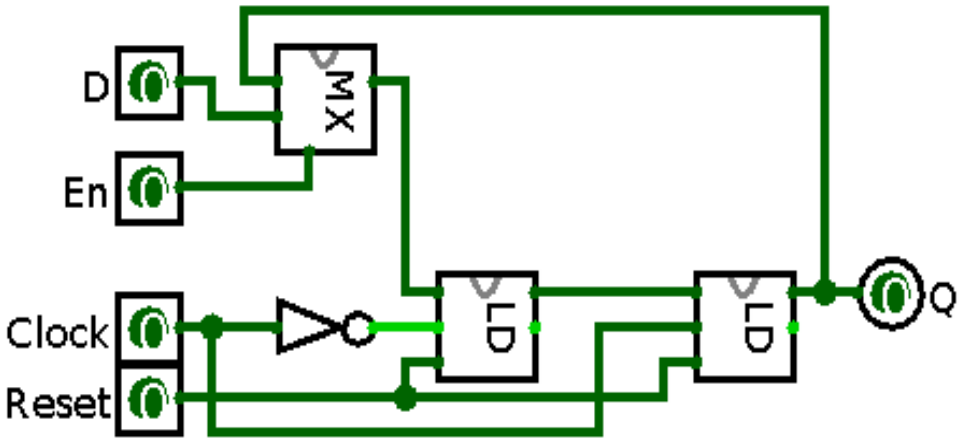
Esercizio 4
Flip-Flop D

›Aggiungiamo al nostro Flip Flop anche un segnale di **Enable** (EN), tale segnale abilita o meno la scrittura sul registro.

›Supponendo di avere più registri collegati sullo stesso Bus dati, senza un segnale di enable, al fronte di salita del clock, rischierei di sovrascrivere tutti i registri!

Esercizio 4 - Soluzione

Flip-Flop D



1. **CLK = 0** : LD Master riceve D e lo porta in Q, LD Slave non trasferisce ancora in quanto vede il clock negato.

2. **CLK = 1** : LD Master non trasferisce più, LD Slave porta D in Q.

Registri (1)

›Con il termine **registri** si comprende tutta una serie di circuiti sincronizzati,

›principalmente deputati alla memorizzazione dell'informazione.

›Più generalmente, un registro a n bit ha la funzione di mantenere un dato,

›detta **parola**, che è in pratica un numero binario ad n cifre.

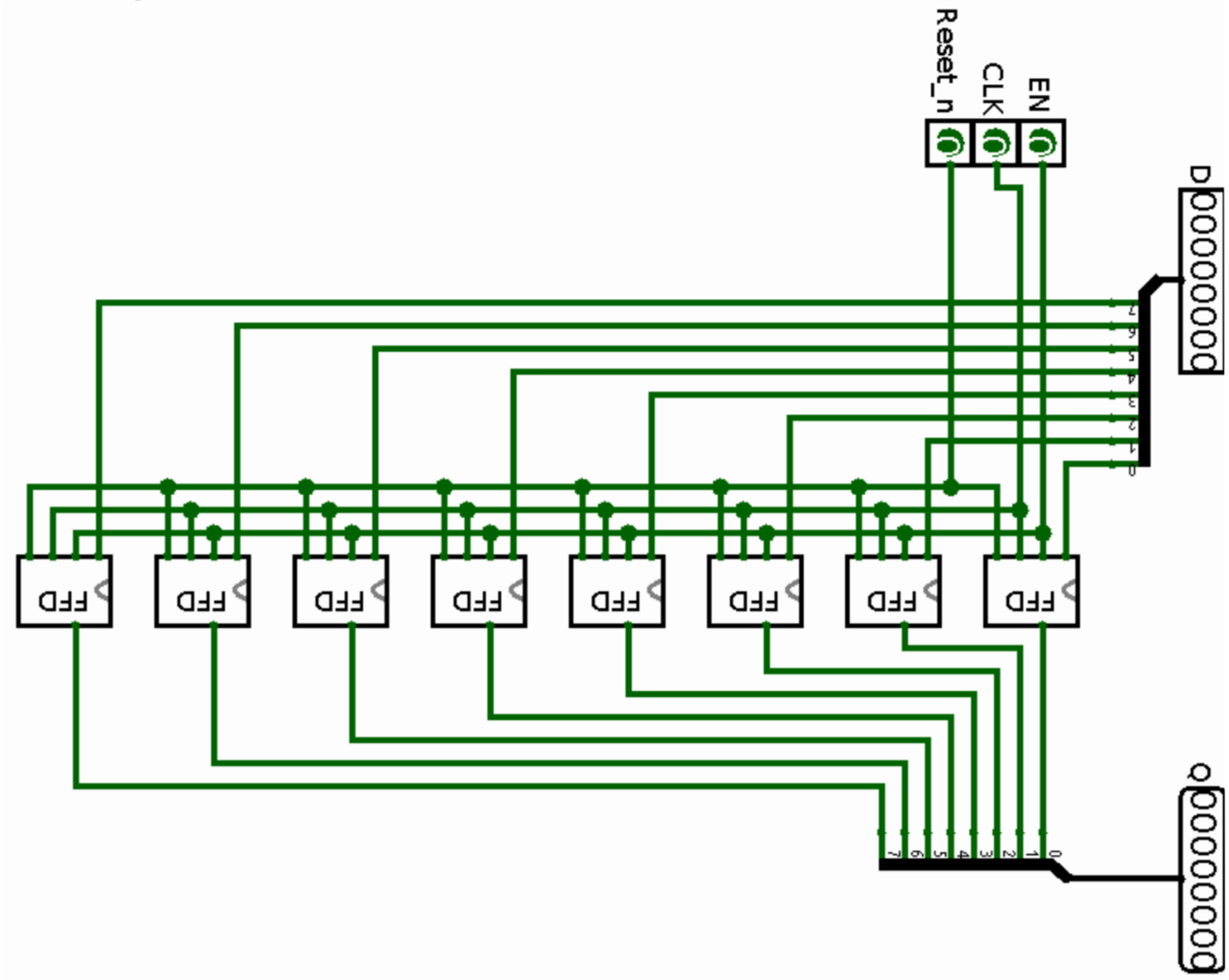
Esercizio 5
Registro

›Realizzare un **registro a 8 bit** utilizzando i Flip-Flop D precedentemente costruiti e avente tutte le caratteristiche viste:

- ›Rising edge-triggered
- ›Segnale di Reset
- ›Segnale di Enable

Esercizio 5 - Soluzione

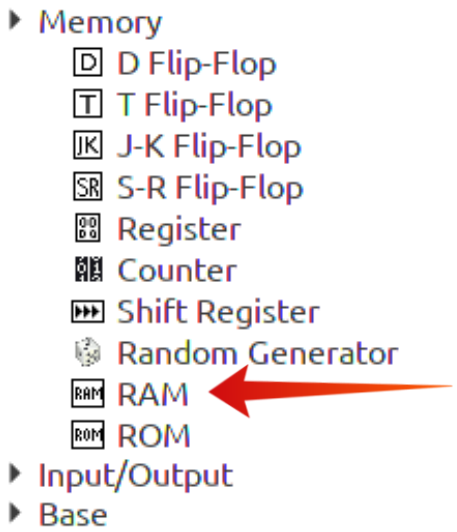
Registro



Esempio
Test con registri

›Per testare il funzionamento dei registri che abbiamo progettato, andiamo a simulare un **caricamento di dati** dalla RAM.

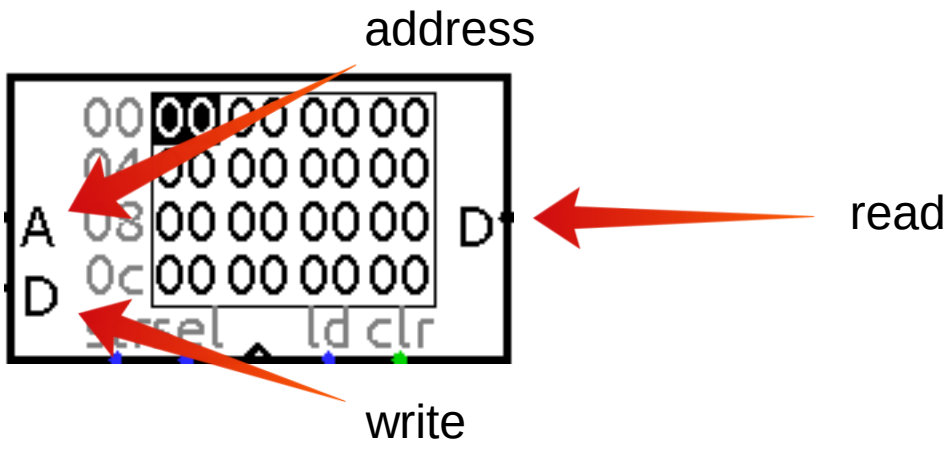
›Utilizziamo il componente RAM di Logisim:



Esempio
Test con registri

›Dopo aver piazzato il componente selezioniamo “Separate Load and Store Ports” nella voce “Data Interface”.

›In questo modo abbiamo tre Bus separati, uno di scrittura, uno di lettura e un Bus per gli indirizzi:

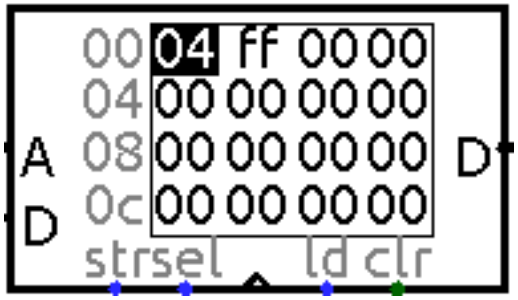


Esempio
Test con registri

›In modalità simulazione possiamo poi settare manualmente i bytes all'interno della memoria RAM, settiamo i primi due bytes, ad esempio:

›Byte 0x00 → al valore 0x04

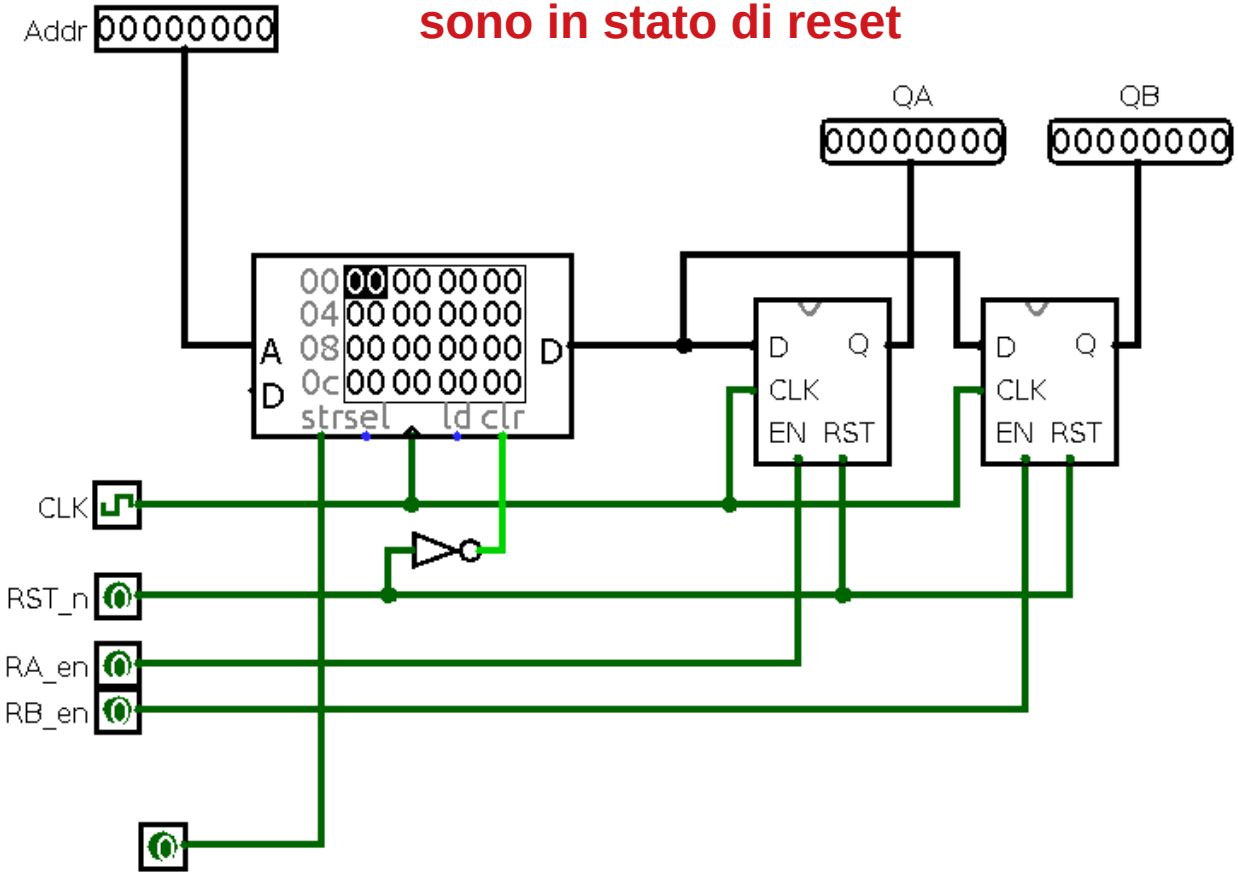
›Byte 0x01 → al valore 0xFF



Esempio

Test con registri

<u>Stato iniziale</u>: tutte le memorie sono in stato di reset



Esempio

Test con registri

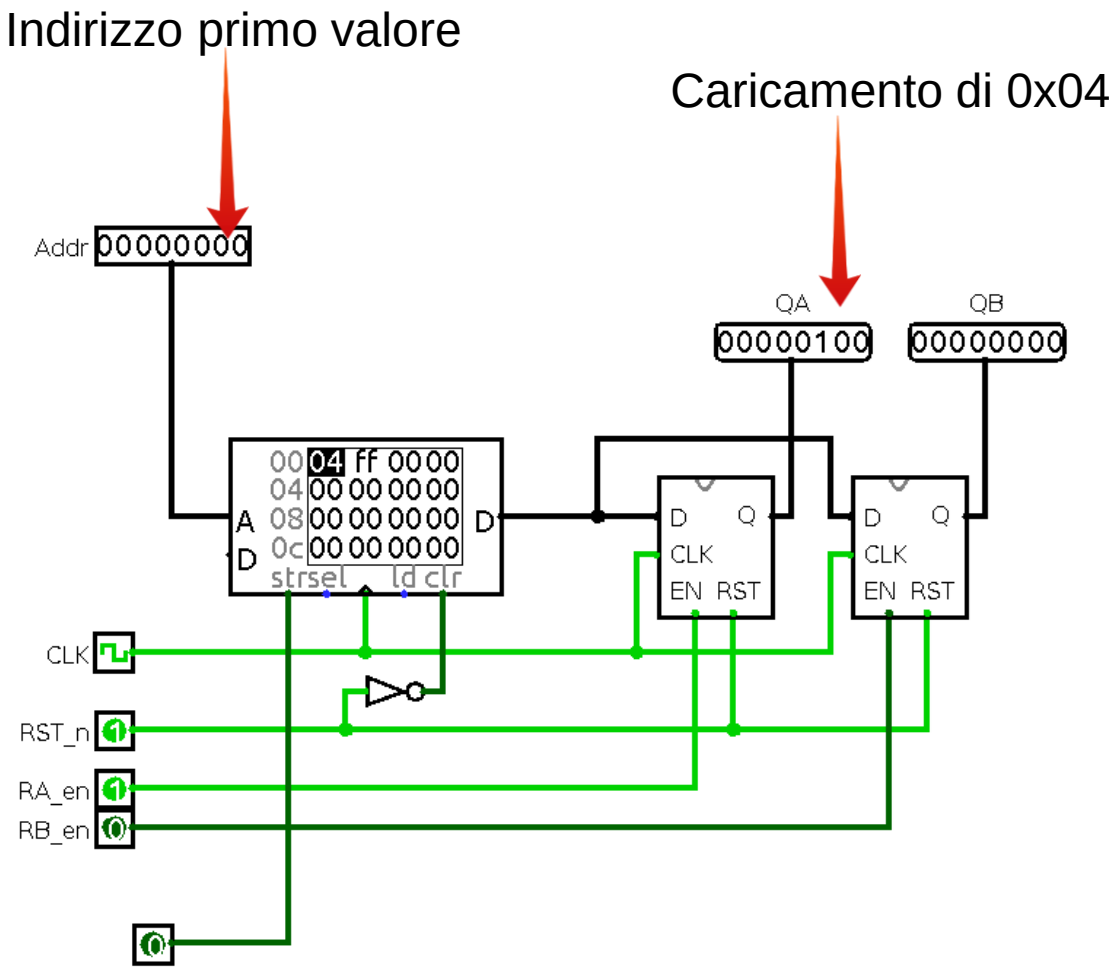
›Supponiamo di voler realizzare il seguente comportamento:

›Ciclo 0 → caricamento del dato all'indirizzo 0x00 nel registro RA.

›Ciclo 1 → caricamento del dato all'indirizzo 0x01 nel registro RB.

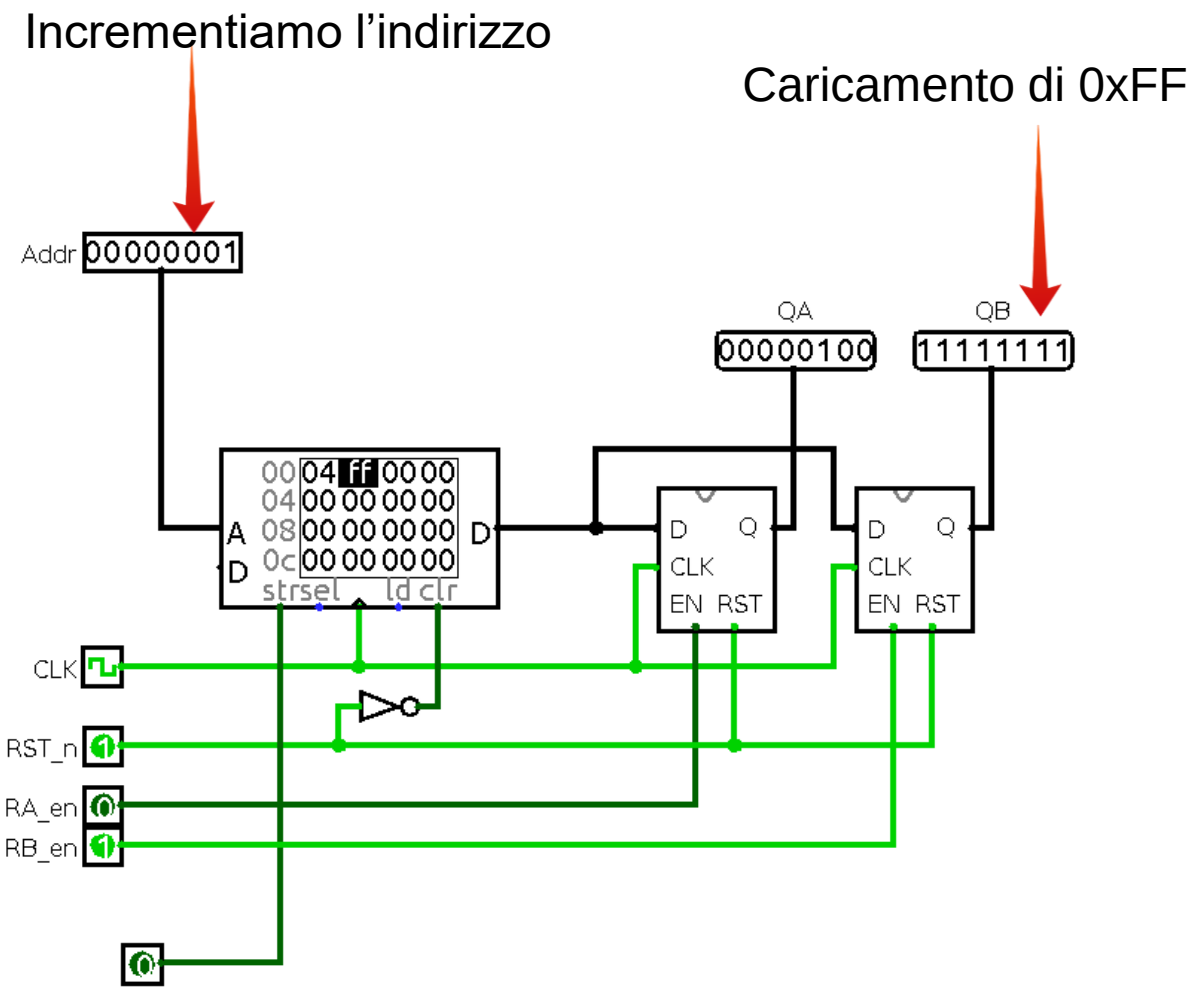
›Reset delle memorie.

Esempio
Test con registri



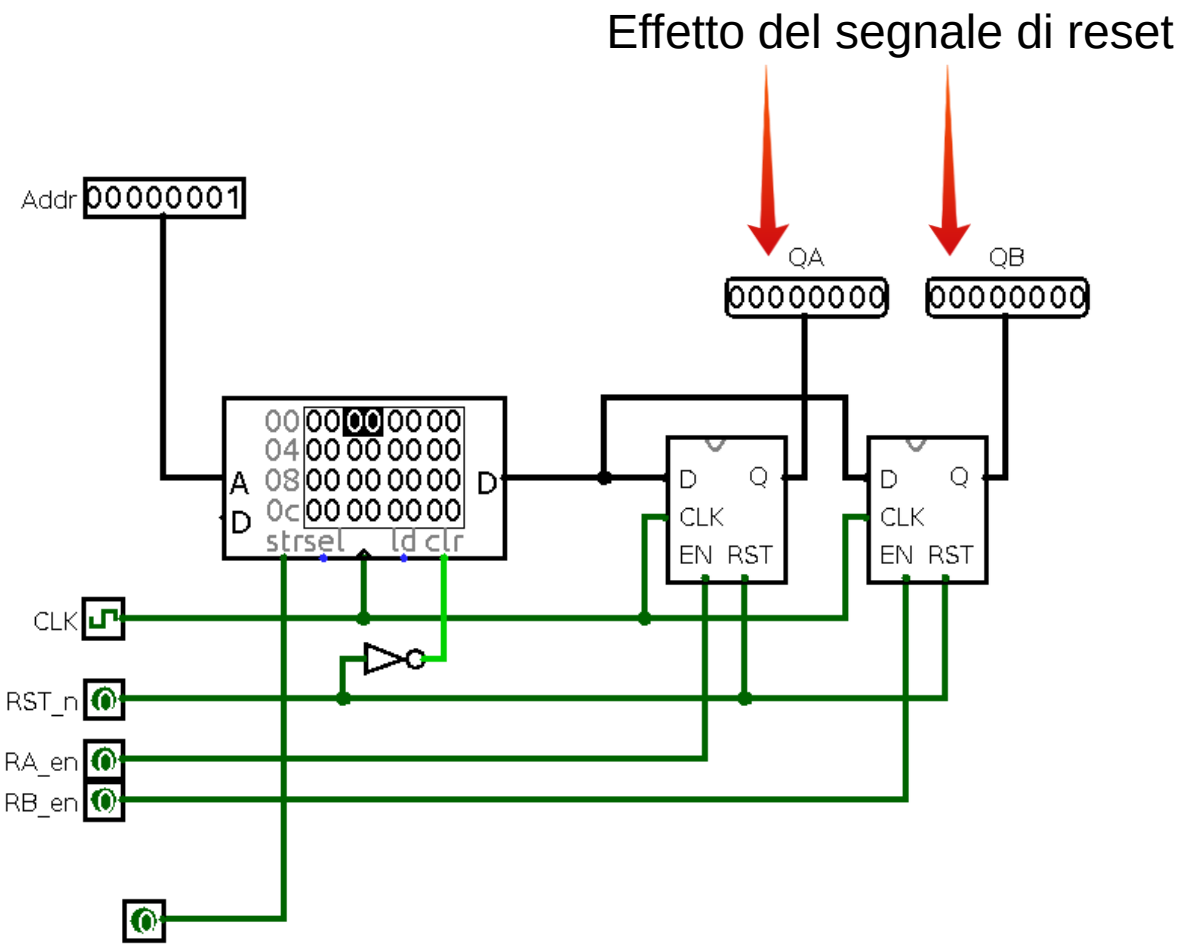
Esempio

Test con registri



Esempio

Test con registri



Esercizio 6

Register File

›Circuito digitale che mantiene al suo interno un certo numero di registri.

›E’ dotato di due linee di uscita alla quale sono multiplexati tutti i registri disponibili. Queste linee di uscita sono i due operandi in ingresso all’Unità Aritmetico-Logica.

›E’ inoltre dotato di alcune logiche che permettono la scrittura su un particolare registro del regfile.

Esercizio 6
Register File

›Schema a blocchi: per chiarire meglio cosa dobbiamo andare a progettare.