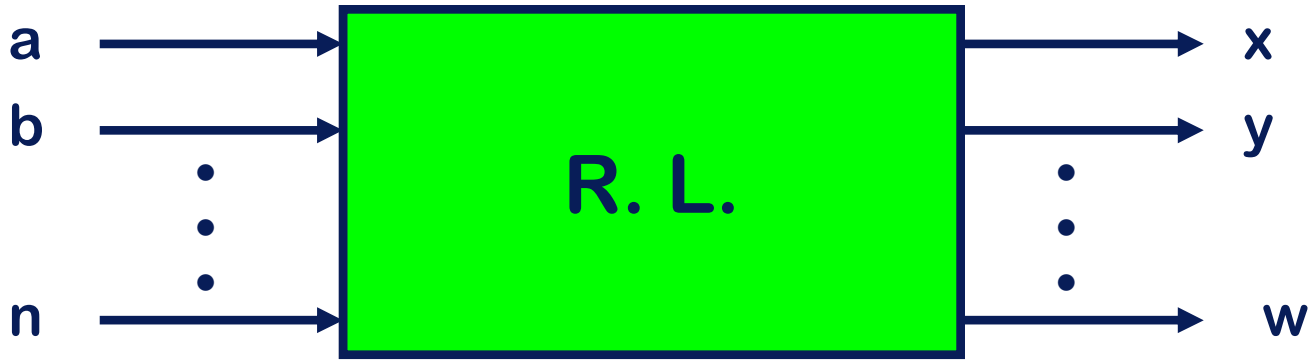
LEZIONE N° 93

Reti Logiche Sequenziali

(parte prima)

Reti Logiche

- Sistema elettronico che ha in ingresso segnali digitali e fornisce in uscita segnali digitali secondo leggi descrivibili con l'algebra Booleana



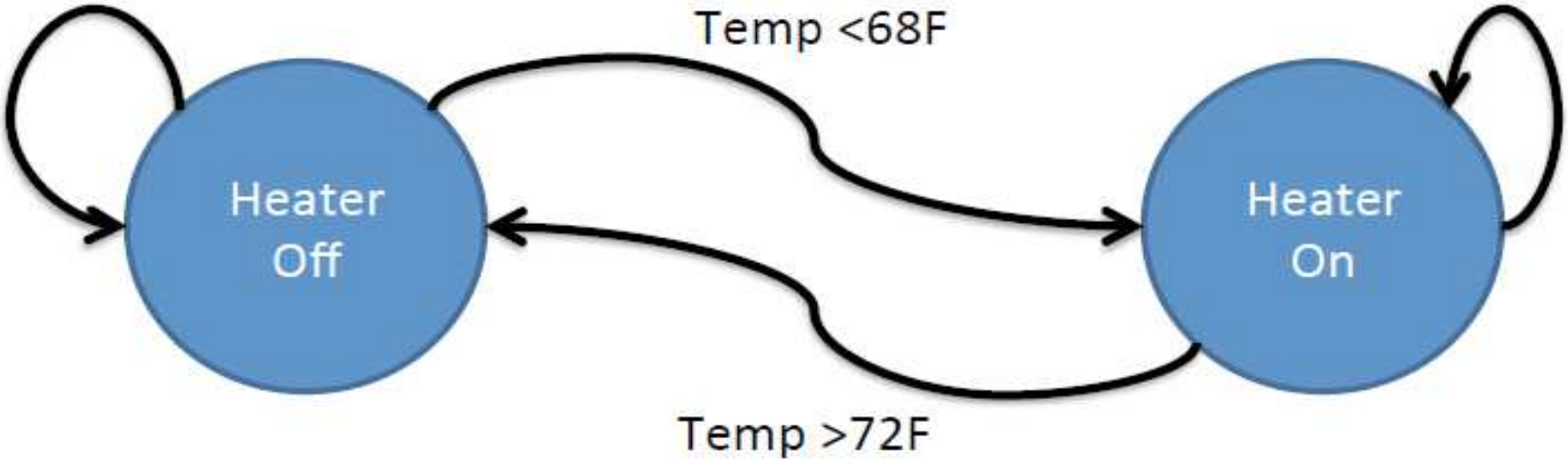
- R.L. è unidirezionale

Tipi di reti

- Reti COMBINATORIE
  - In qualunque istante le uscite sono funzione del valore che gli ingressi hanno in quell'istante
  - Il comportamento (uscite in funzione degli ingressi) è descritto da una tabella
  - Non contengono anelli di richiusura
- Reti SEQUENZIALI
  - In un determinato istante le uscite sono funzione del valore che gli ingressi hanno in quell'istante e i valori che hanno assunto precedentemente
  - La descrizione è più complessa
  - Stati Interni →
  - Reti dotate di *MEMORIA*

Macchine a Stati Finiti

- Un gruppo di stati e di transizioni



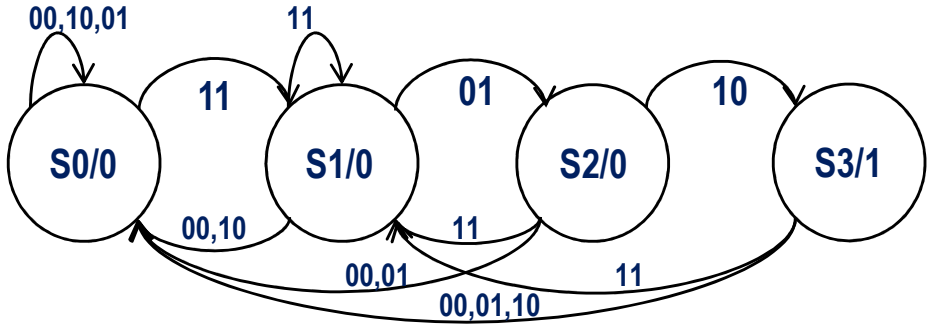
- Ci si sposta da uno stato all'altro seguendo una linea di transazione SE la condizione dalla transizione e' soddisfatta

Macchine a Stati Finiti (2)

- Nel campo dell'architettura dei calcolatori per le FSM:
  - Le transizioni tipicamente avvengono su fronti del clock
  - Sono complete
    - Tutti gli stati definiscono transizioni per tutti gli input
  - Sono deterministiche
- Possono essere trasformate in reti logiche dette Reti Sequenziali
  - Lo vedremo prossimamente.

Teoria delle reti sequenziali

- Negli anni 40-50 si e' consolidata una teoria degli Automi (Chomsky) e in particolare delle cosiddette Macchine a Stati Finiti (FSM)
- Nel 1955 Mealy [2] e nel 1956 Moore [3] propongono dei modelli per implementare una FSM
  - L'idea era che una FSM potesse essere realizzata ricorrendo solo ad elementi di memoria e logica combinatoria opportunamente collegata
  - Pertanto e' nata una teoria di come realizzare delle reti sequenziali che implementino FSM il cui comportamento e' descritto o informalmente (tramite il linguaggio naturale) o formalmente (tramite un diagramma a stati)
    - Esempio: riconoscere una sequenza binaria 11, 01, 10

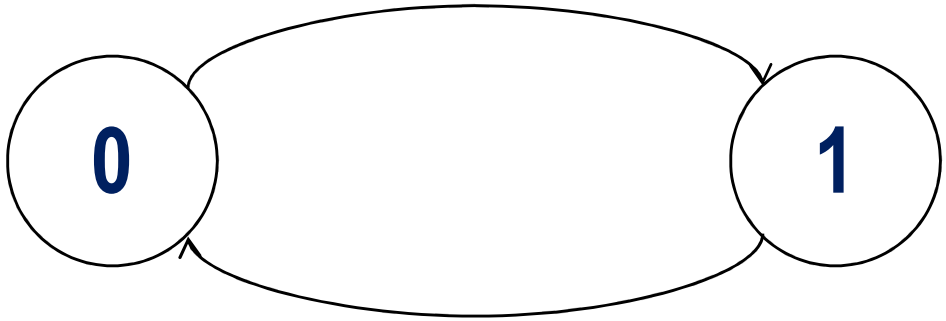


[2] Mealy, George H. (September 1955). "A Method for Synthesizing Sequential Circuits". *Bell System Technical Journal* **34**: 1045–1079

[3] Moore, Edward F (1956). "Gedanken-experiments on Sequential Machines". *Automata Studies, Annals of Mathematical Studies* (Princeton Univ. Press) (34): 129-153.

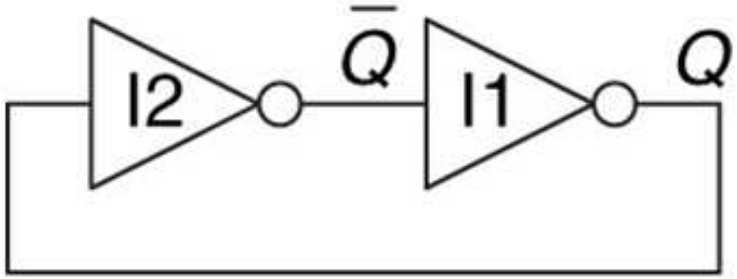
L'elemento di memoria

- La FSM piu' semplice che si puo' immaginare (e che varia il proprio stato) e' una rete a due soli stati
  - Tale rete non rappresenta altro che un bit di informazione e quindi permette di implementare un elemento di memoria



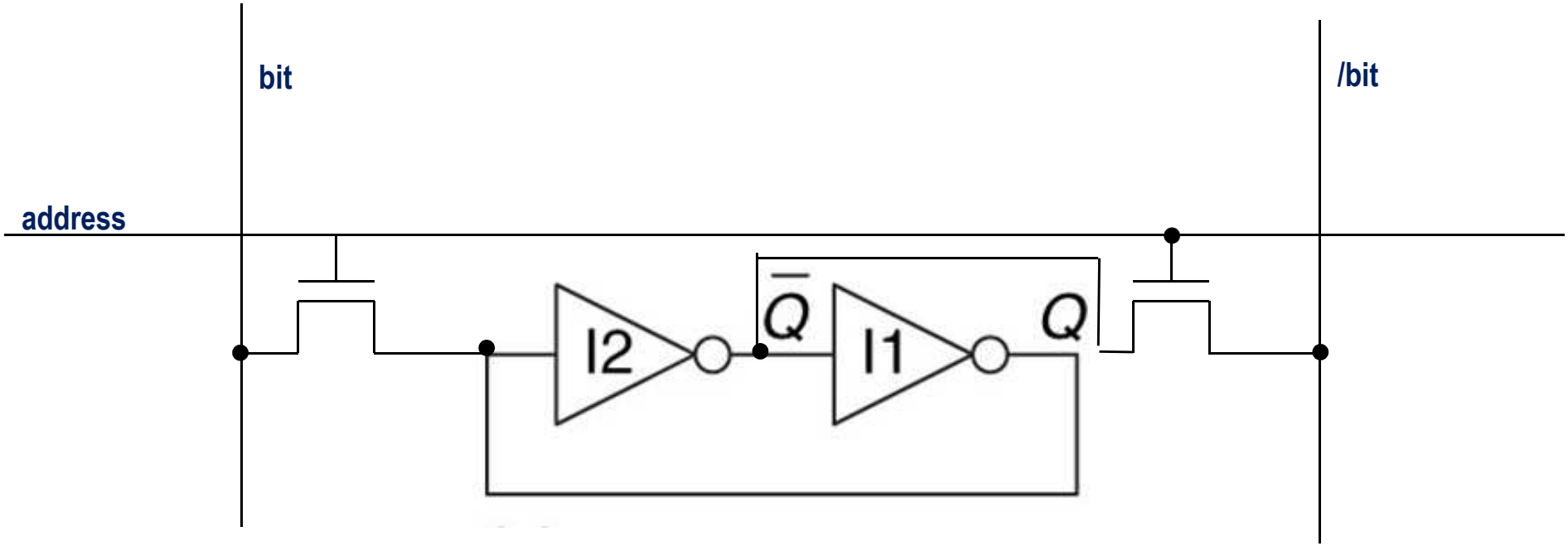
Realizzazione di elementi di memoria con reti combinatorie

- Abbiamo visto che l'elemento piu' semplice che riusciamo a realizzare facilmente su silicio e' un inverter…
- Il modo piu' semplice per realizzare un elemento di memoria usando inverter e' di utilizzare due inverter… richiusi ad anello



- Questo circuito digitale presenta DUE STATI stabili (si dice peranto che e' bistabile):
  - i) Q=0 (e di conseguenza /Q=1)
  - ii) Q=1 (e di conseguenza /Q=0)
- Pero' presenta solo due uscite e nessun ingresso
- Se ben dimensionato fisicamente funziona, altrimenti potrebbe anche oscillare (comportamento metastabile)

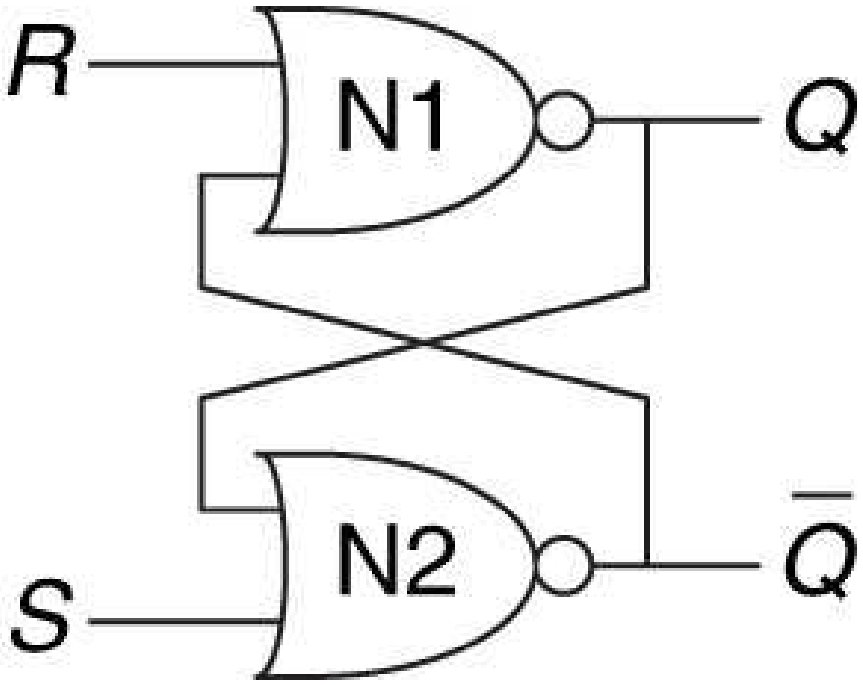
Perfezionando il bistabile usando la tecnologia CMOS



- In questo modo si puo' attivare l'elemento tramite il filo address e
  - imporre un 1 logico (bit=1,/bit=0) oppure uno 0 logico (bit=0,/bit=1)
  - Oppure leggere il valore Q senza distruggere il contenuto
- Questo tipo di cella di memoria viene usata per realizzare le SRAM (Static RAM) → l'informazione rimane memorizzata fintanto che la cella e' alimentata
  - In tecnologia CMOS si realizza con 6 transistor (2 per ogni inverter e due per i due transistor di passo)

Latch SR

- L'elemento di memoria realizzato con due inverter e' in grado di memorizzare l'informazione ma non ha una forma "autocontenuta" per memorizzare il dato.
- Pertanto sono stati sviluppati ulteriori elementi bistabili denominati
  - Latch: elementi che variano lo stato su un livello di un segnale di clock
  - Flip-flop: elementi che variano lo stato su un fronte di un segnale di clock

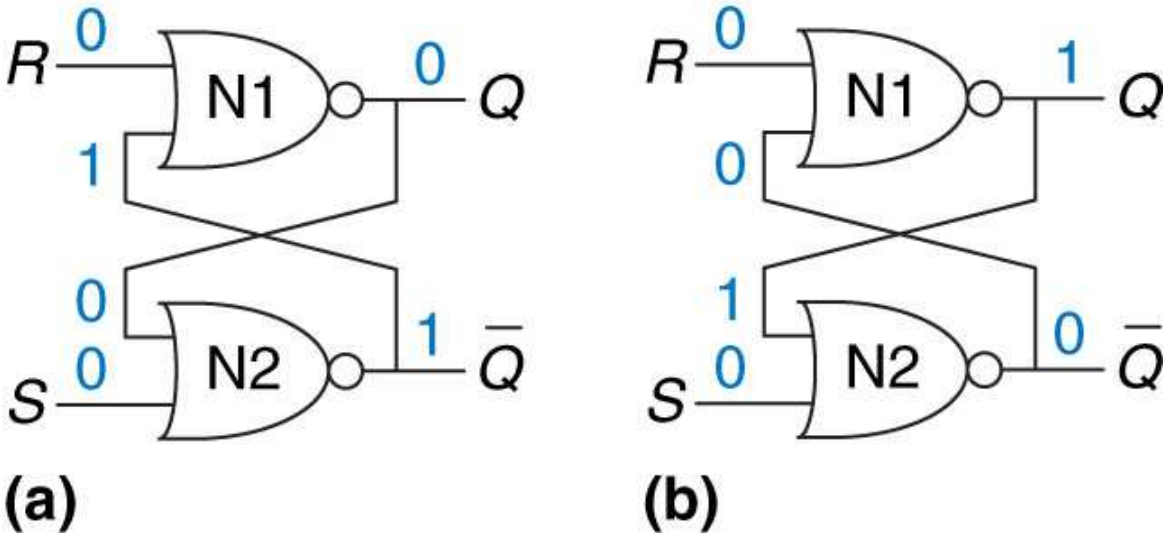


Elemento di memorizzazione in logica sequenziale asincrona

- Il latch SR
  - l'uscita dipende dai segnali presenti sugli ingressi e dai segnali precedentemente presentati in ingresso

S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	---	---

Tabella di Verità

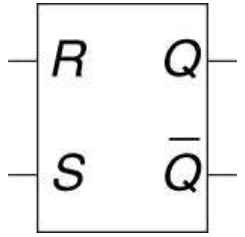


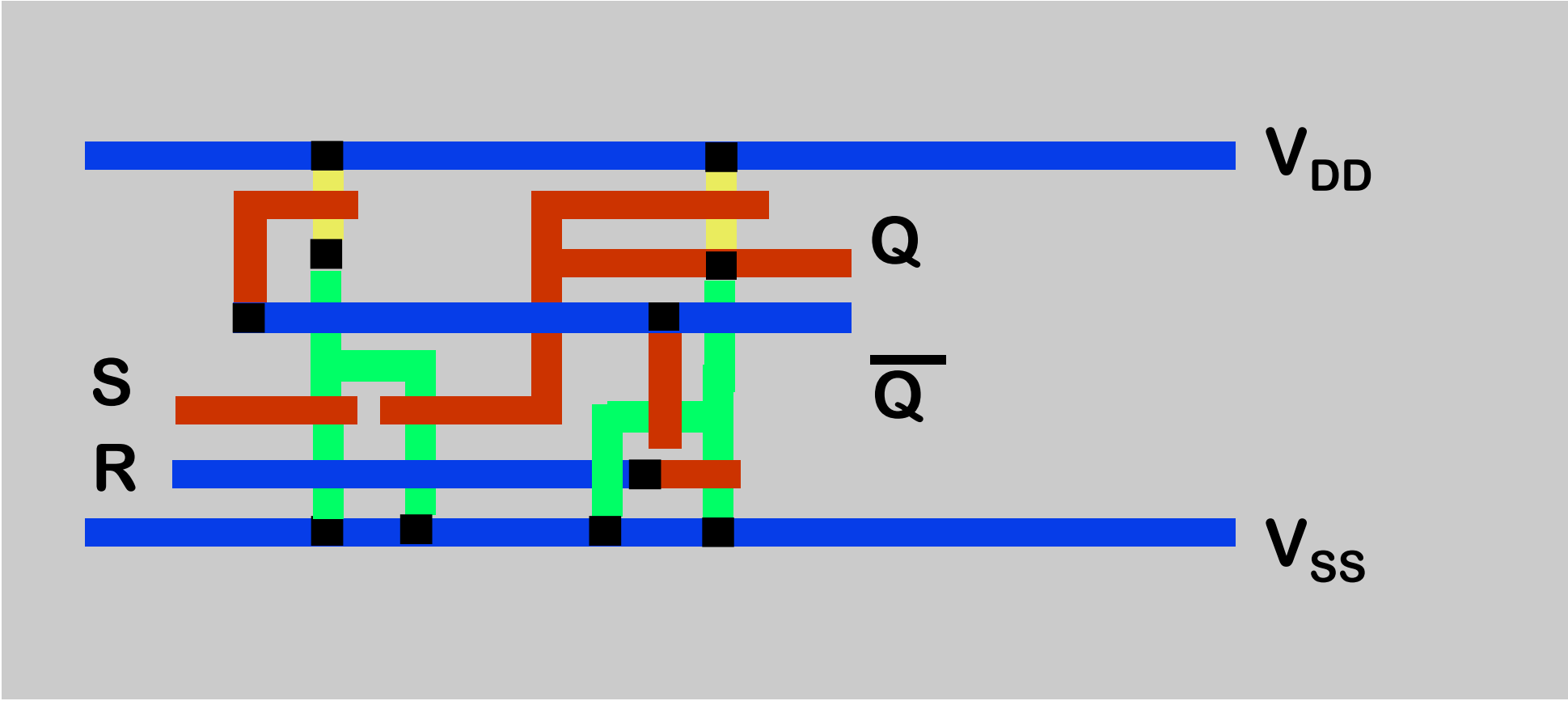
Schema logico

Lay-Out simbolico in tecnologia CMOS (latch SR)

- Transistori necessari = 6

Simbolo →



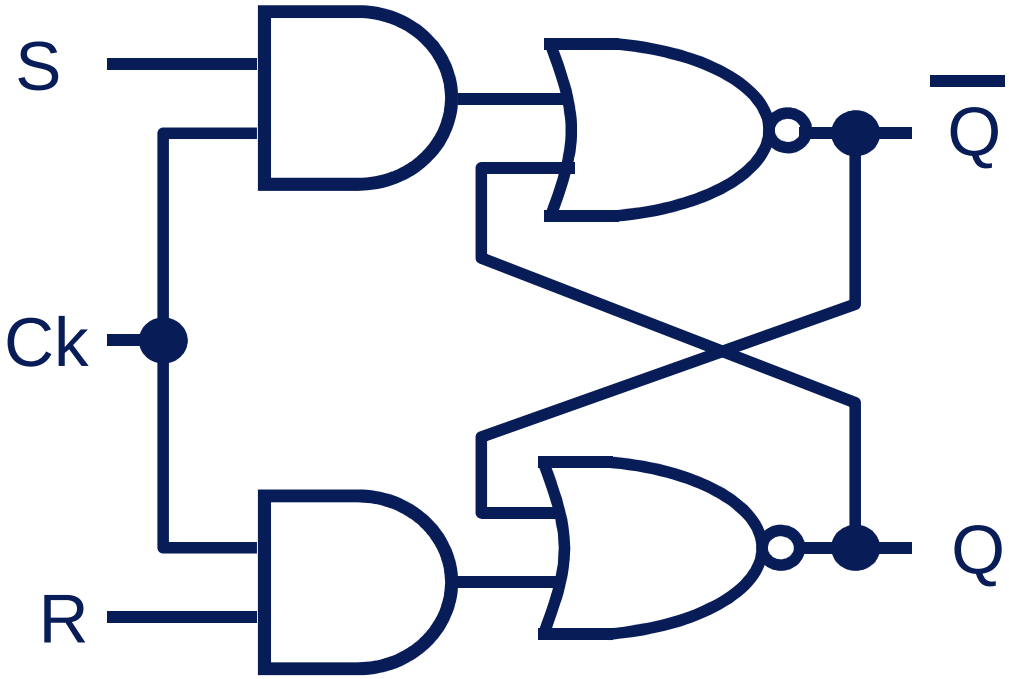


S-R cloccato

Sente S e R solo quando il clock vale 1

Ck	S	R	Q	$\overline{Q}$
1	0	0	Q	$\overline{Q}$
1	0	1	0	1
1	1	0	1	0
1	1	1	---	---
0	X	X	Q	$\overline{Q}$

Tabella di Verità

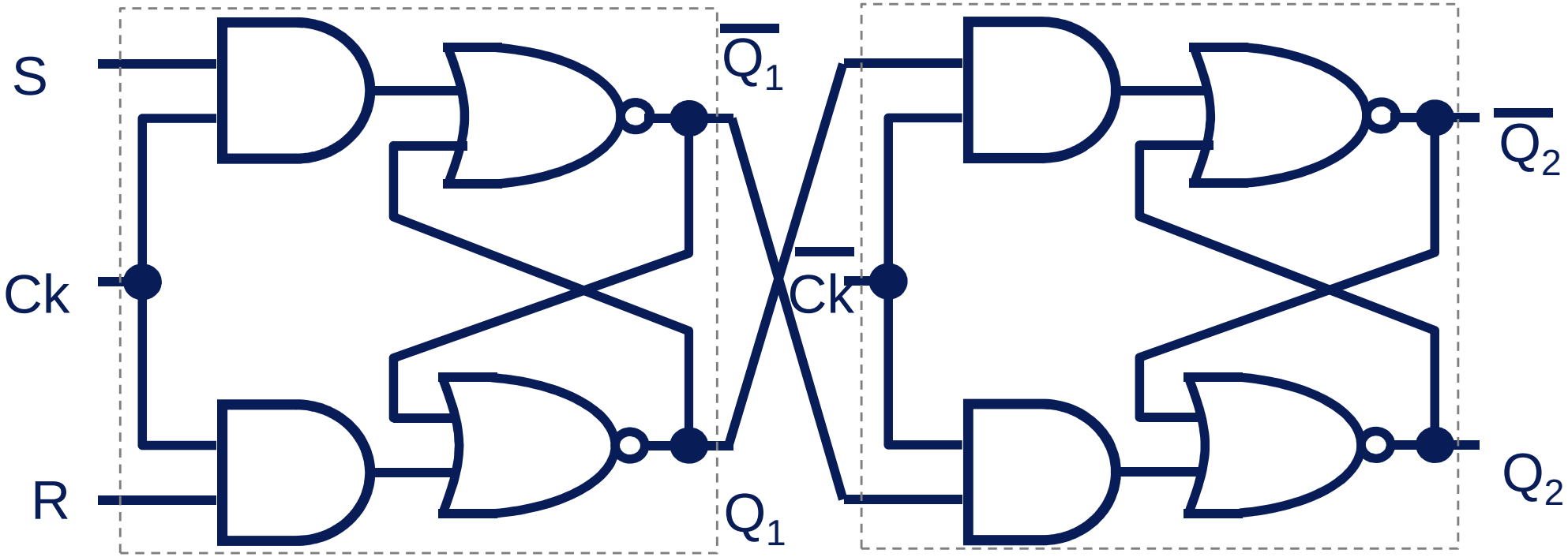


Schema logico

Layout simbolico: (12 transistor)

S-R edge triggered

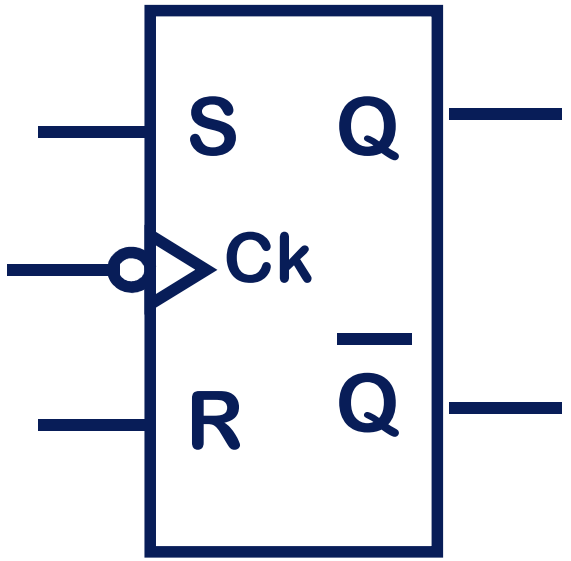
In questo caso vogliamo sensibilita' solo sul fronte in discesa



Layout simbolico: (24 transistor)

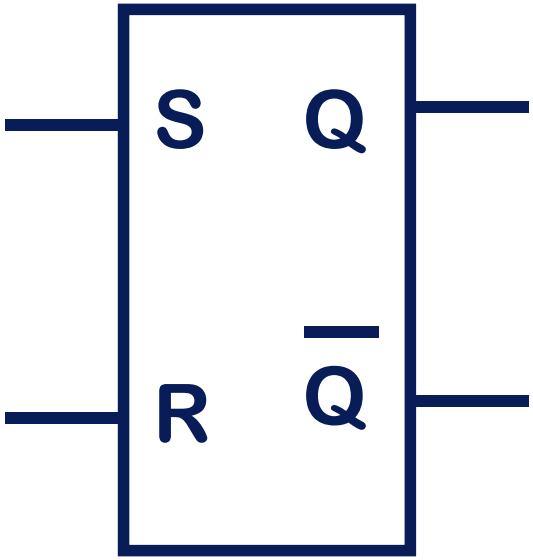
Tabella verita' S-R edge-triggered

Ck	S	R	Q	$\overline{Q}$
⌐	0	0	Q	$\overline{Q}$
⌐	0	1	0	1
⌐	1	0	1	0
⌐	1	1	---	---
0	X	X	Q	$\overline{Q}$
1	X	X	Q	$\overline{Q}$
⌙	X	X	Q	$\overline{Q}$

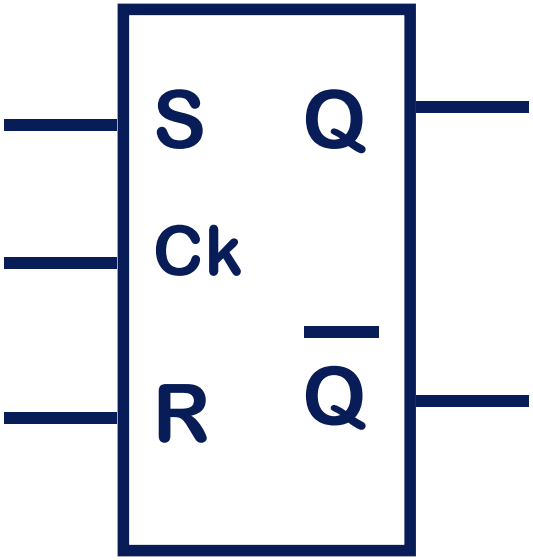


Simboli

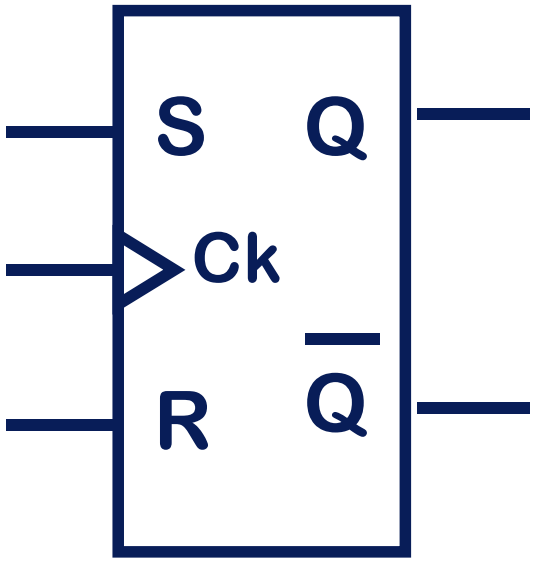
S-R



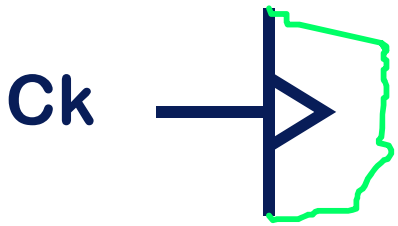
S-R cloccato



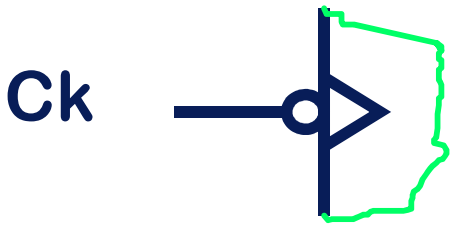
S-R edge triggered



Sensibili sul:



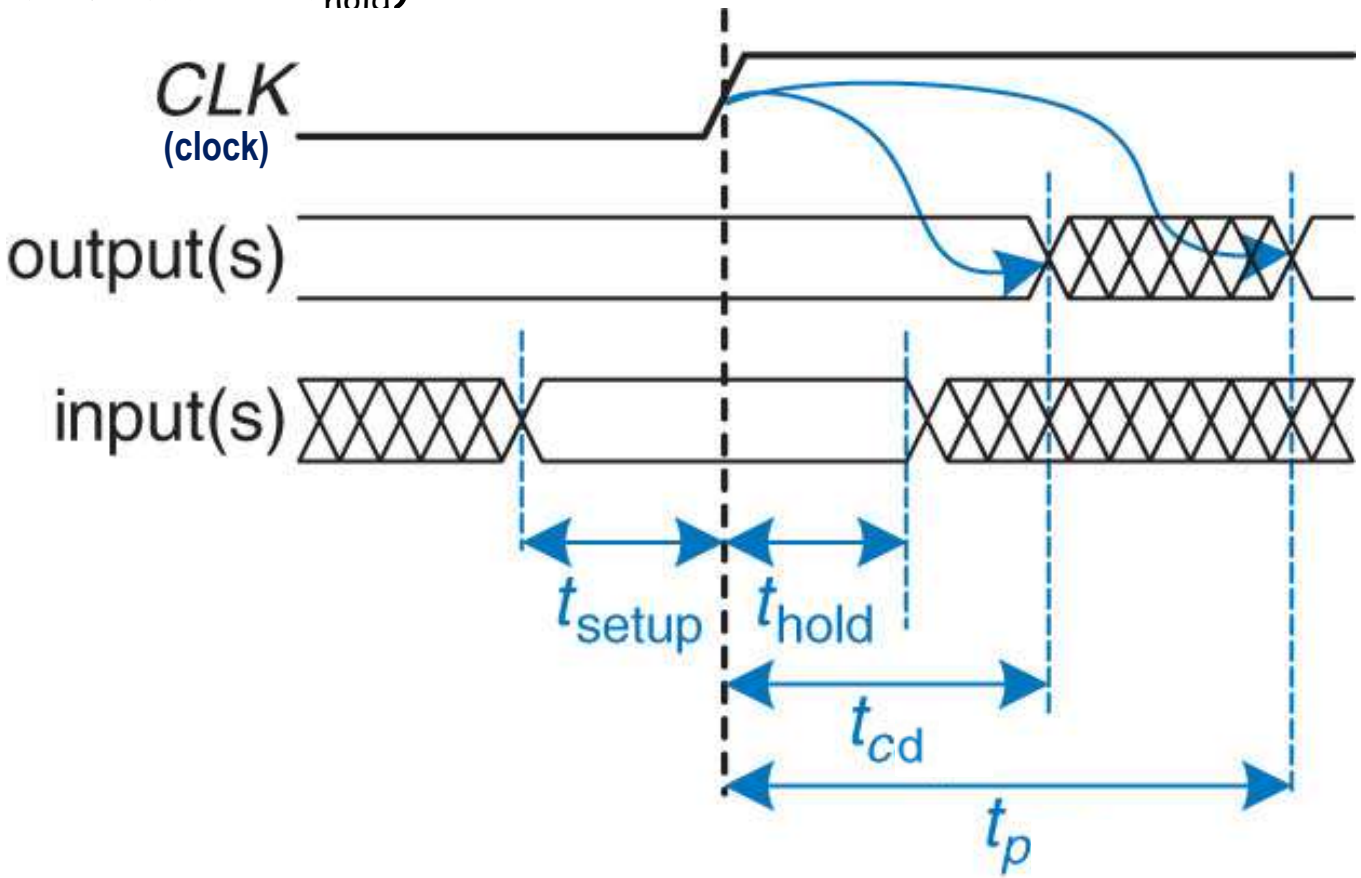
Fronte in salita



Fronte in discesa

tsetup e thold

- I segnali di ingresso SR devono essere stabili durante un fronte del clock (vanno preparati un po' prima → t_{setup})
- Dopo il fronte si possono tranquillamente variare i segnali (dopo aver atteso t_{hold})



- Il tempo di propagazione tp e' il tempo necessario affinche' l'uscita sia stabile dal momento in cui ho variato l'ingresso
- Si definisce invece tempo di contaminazione tcd il tempo che intercorre da quando ho variato l'ingresso a quando inizia a variare l'uscita

D-Latch

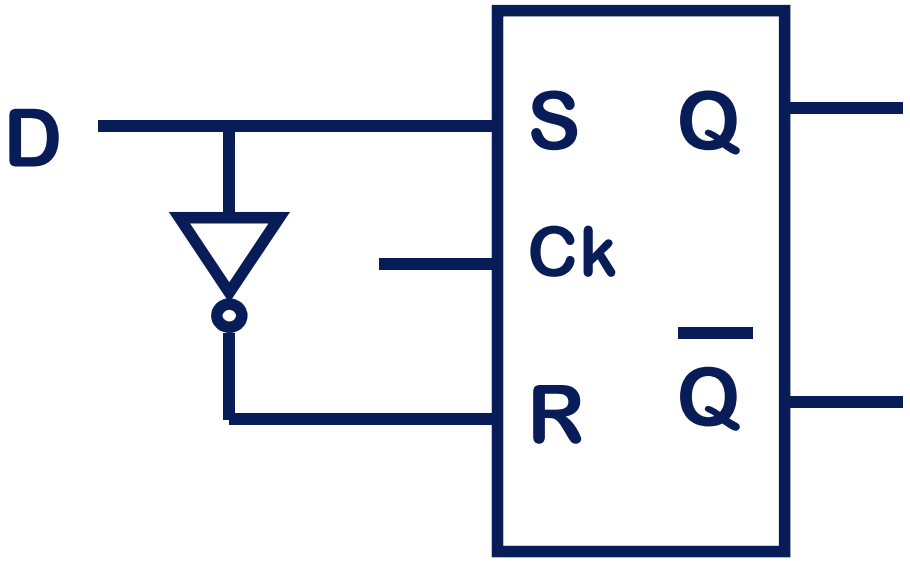
- IL latch SR e' problematico perche' si comporta in modo impredicibile quando SR=11
- SR definiscono anche *cosa* e *quando* le uscite vengono cambiate
  - Il progetto dei circuiti digitali e' piu' semplice se si definiscono separatamente cosa cambiare e quando cambiarlo
  - Questo e' esattamente cosa fa il D-latch
- D-latch
  - Il segnale D controlla il dato ovvero *cosa* deve essere l'uscita
  - Il segnale CLK controlla *quando* lo stato deve essere cambiato

D-Latch o D-Transparent

- Due ingressi:
  - il dato da memorizzare (D)
  - il segnale di clock (Ck) che indica quando leggere o memorizzare (D)

D	Ck	Q
X	0	Q
0	1	0
1	1	1

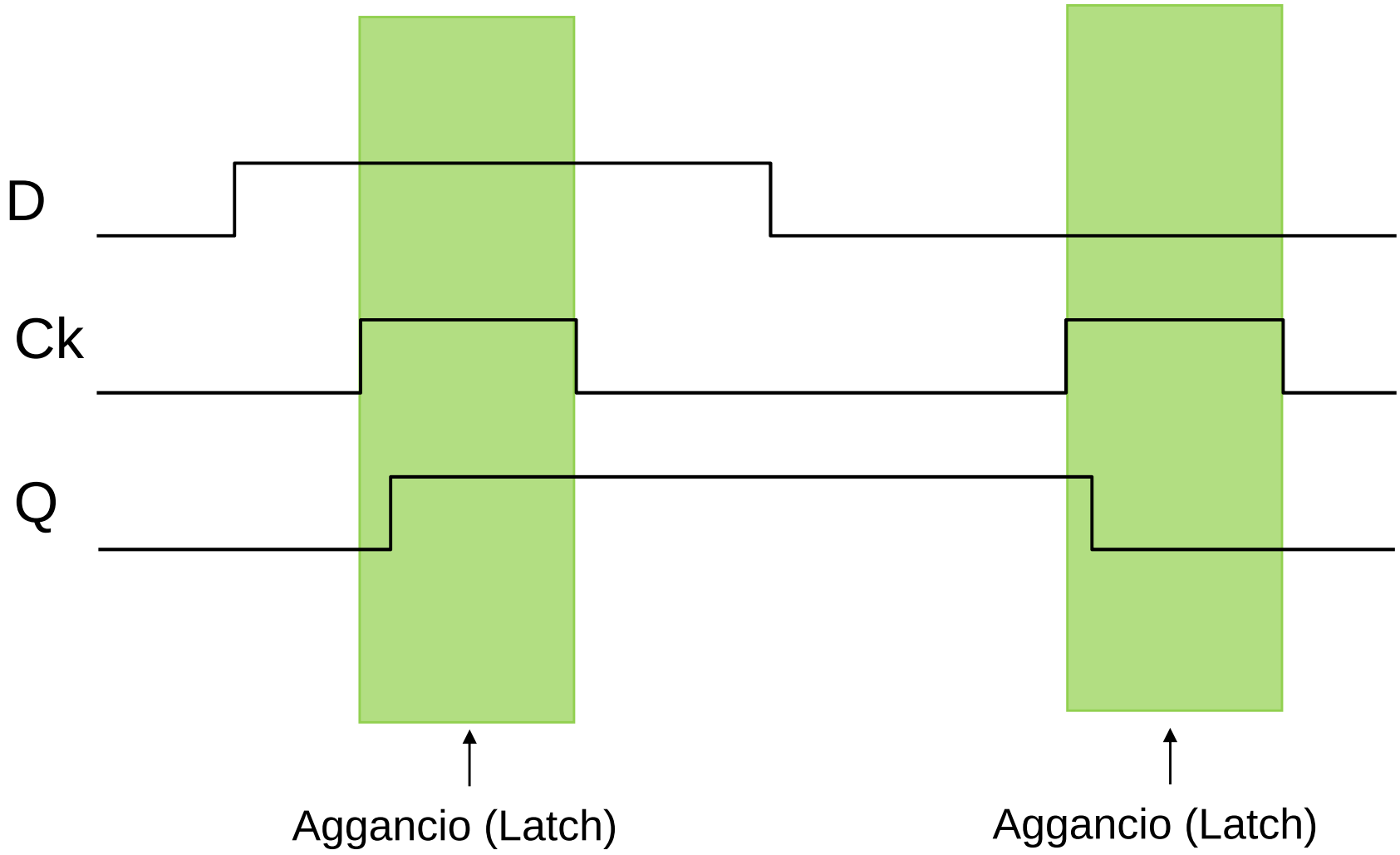
Tabella di Verità



Schema logico

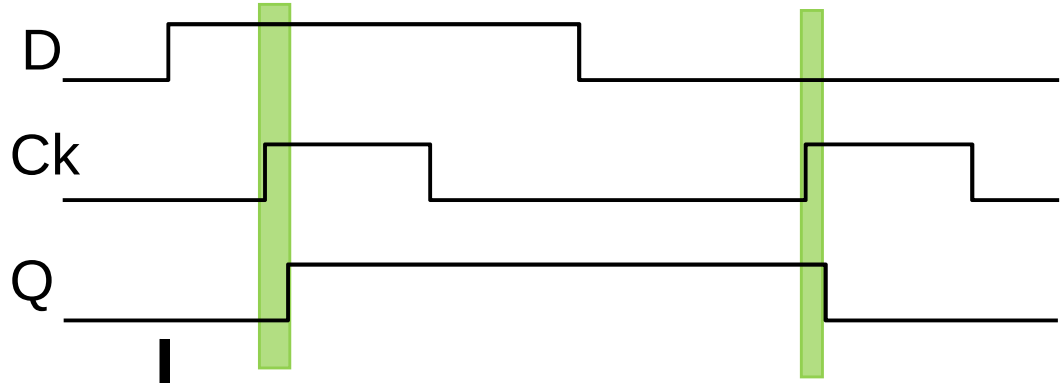
Layout simbolico: (14 transistor)

D-Latch o D-Transparent (esempio)



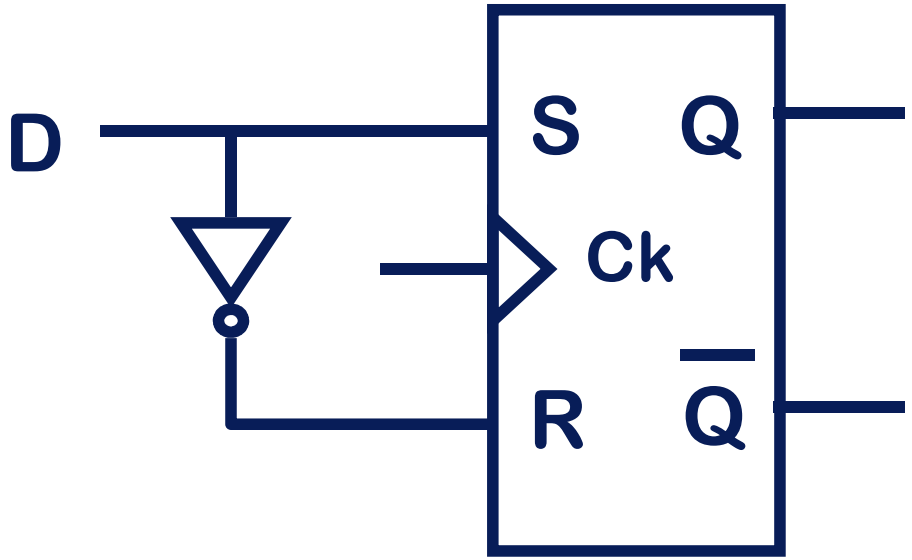
D-edge triggered

- L'uscita cambia solo sul fronte in salita del clock



D	Ck	Q
X	1	Q
X	0	Q
X	↑	Q
0	↑	0
1	↑	1

Tabella di Verità



Schema logico

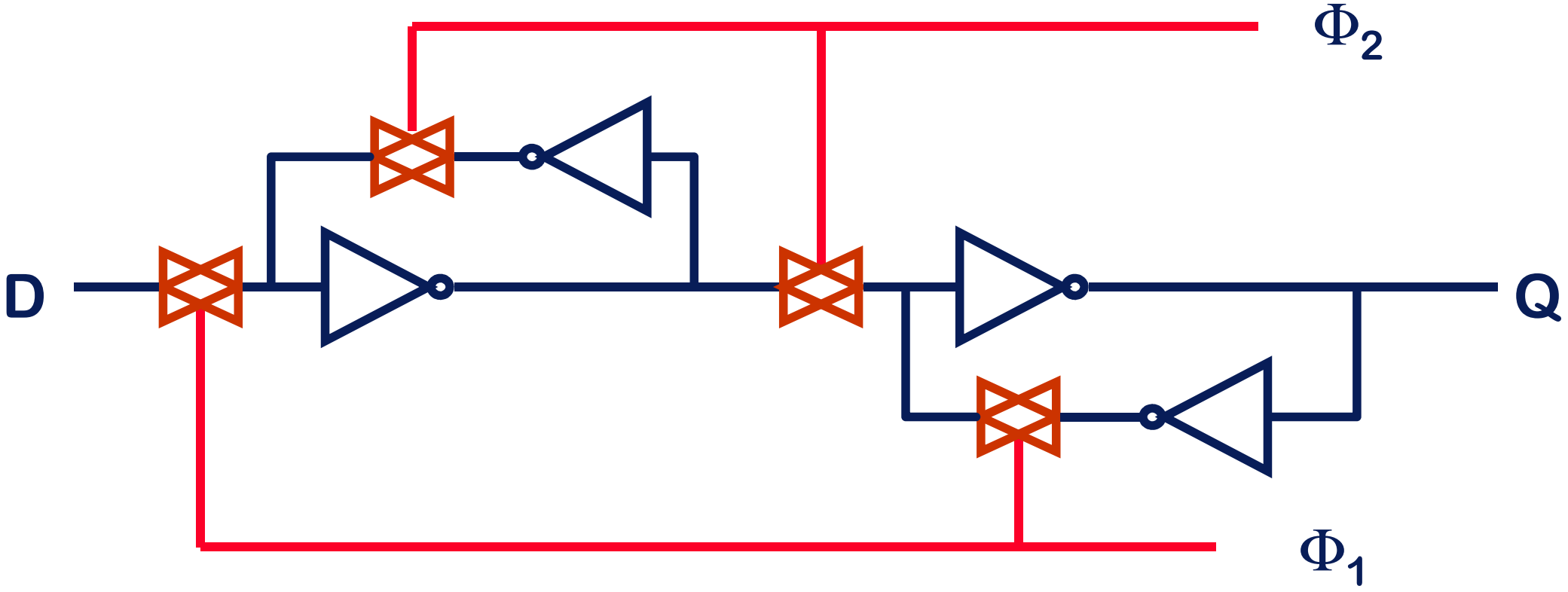
Layout simbolico: (26 transistor)

Latch vs. Flip-flops

- Latch:
  l'uscita cambia non appena cambia l'ingresso
  E
  il clock ha valore alto
- Flip-flop:
  l'uscita cambia solo su un fronte del clock
  (metodologia edge-triggered)
- Elementi in comune:
  - l'uscita e' pari al valore memorizzato nell'elemento
    (non e' necessario fare azioni particolari per leggere il valore)
  - il cambiamento dell'uscita avviene rispetto a un segnale di clock
- Nota:
  nella logica sincrona si definisce l'istante
  in cui un valore e' pronto per essere letto sugli ingressi del flip-flop
  - non si vuole leggere un dato D mentre la rete che genera D sta ancora "calcolandone" il valore

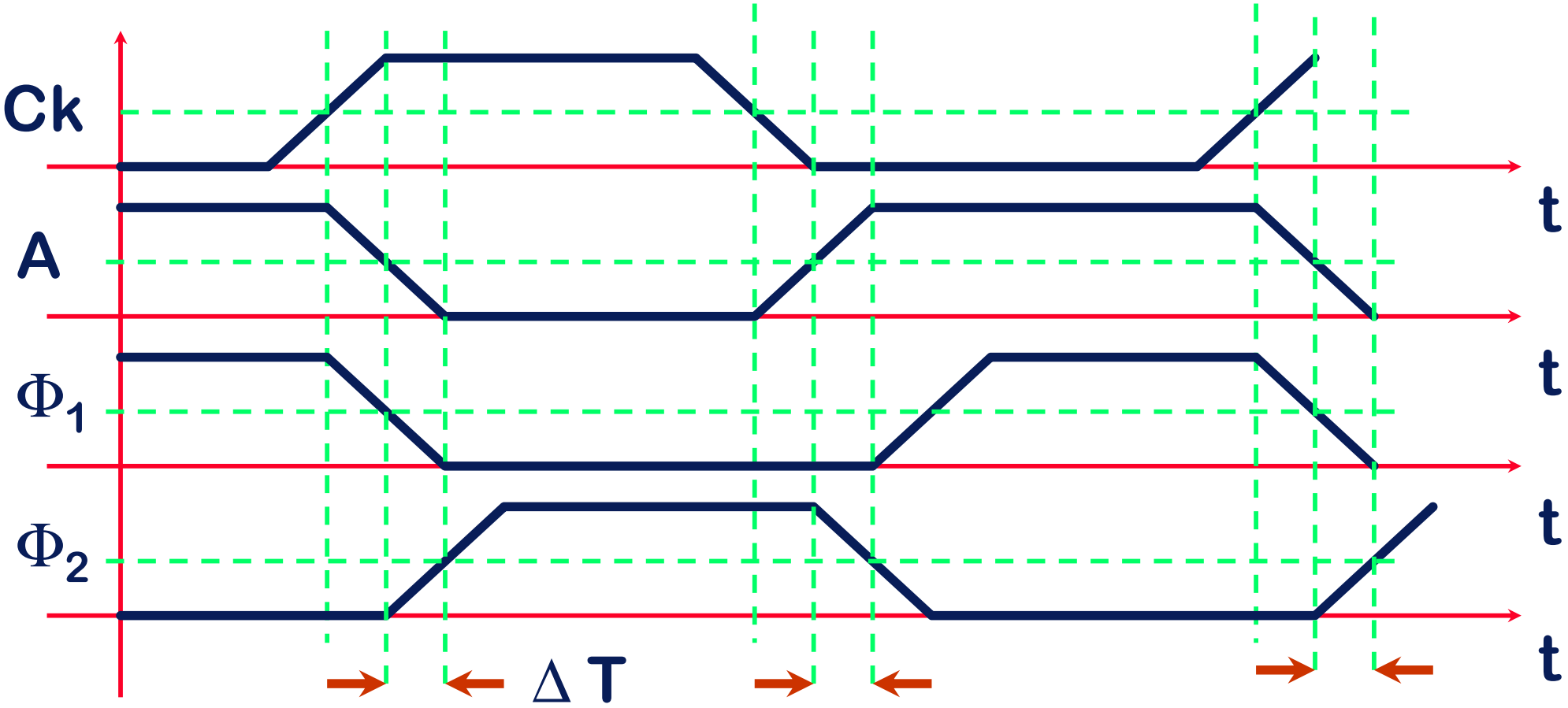
D Edge Triggered

Disponendo di un “clock a due fasi” si puo’ utilizzare una forma piu’ compatta del D-edge-triggerd



Layout simbolico: (16 transistor)

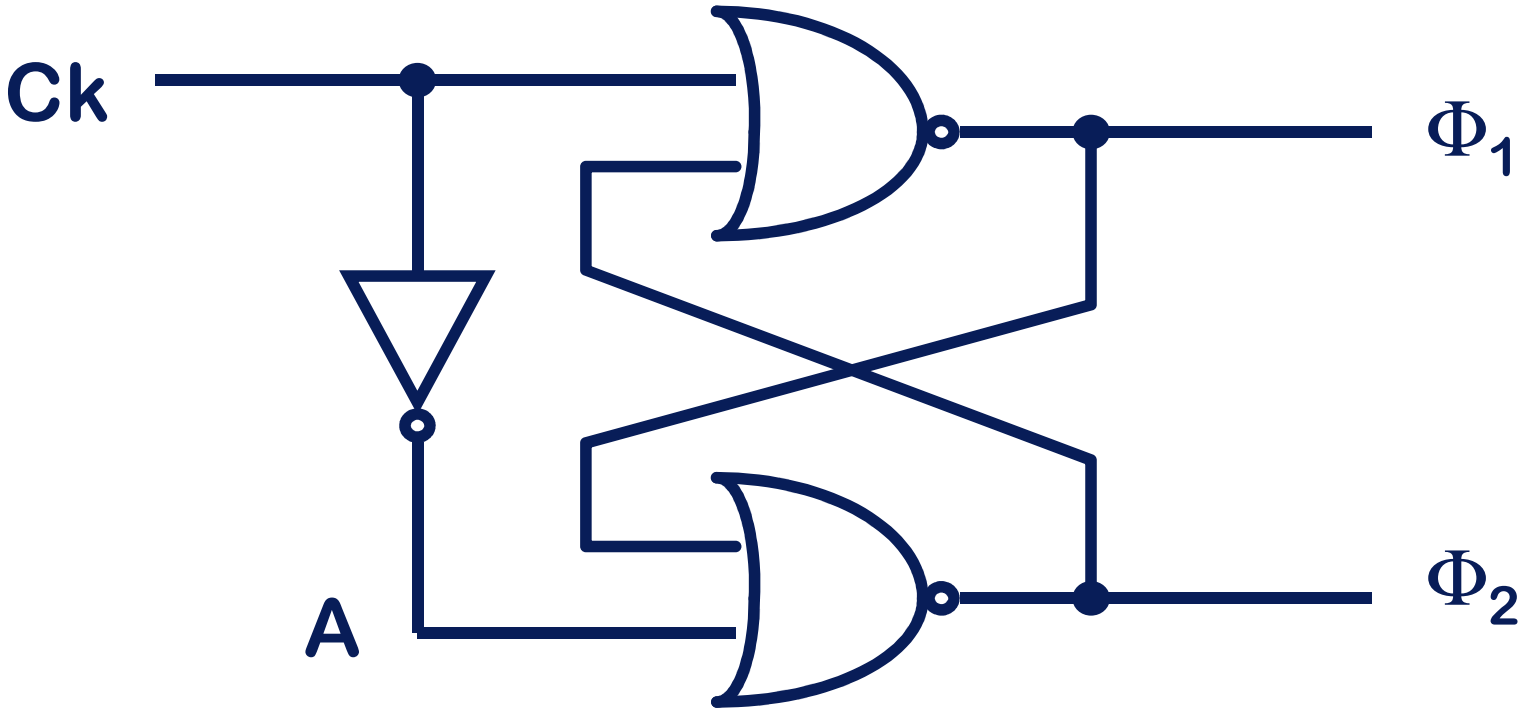
Forme d'Onda del Clock a due fasi



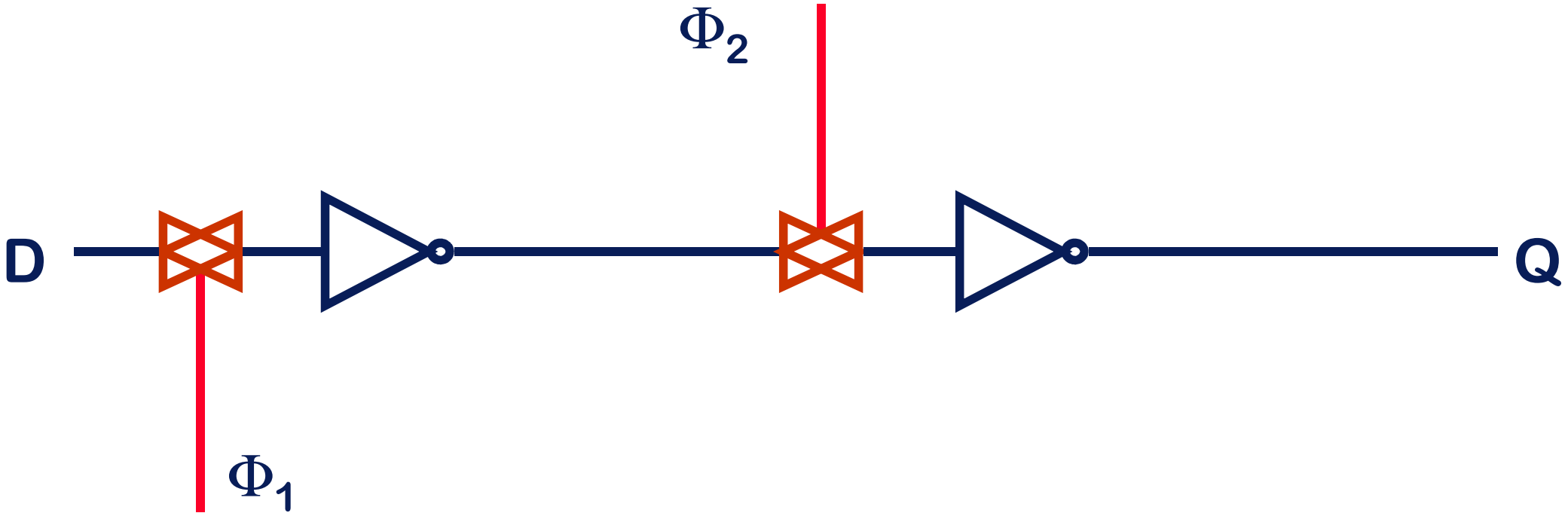
Generatore di Clock a due fasi

- Φ e $\overline{\Phi}$ non sono l'uno la negazione dell'altro !
  - Un opportuna spaziatura garantisce che non siano mai entrambi alti (neanche durante un breve intervallo di tempo)

- Ad es. Φ_1 e Φ_2 si possono generare col seguente circuito

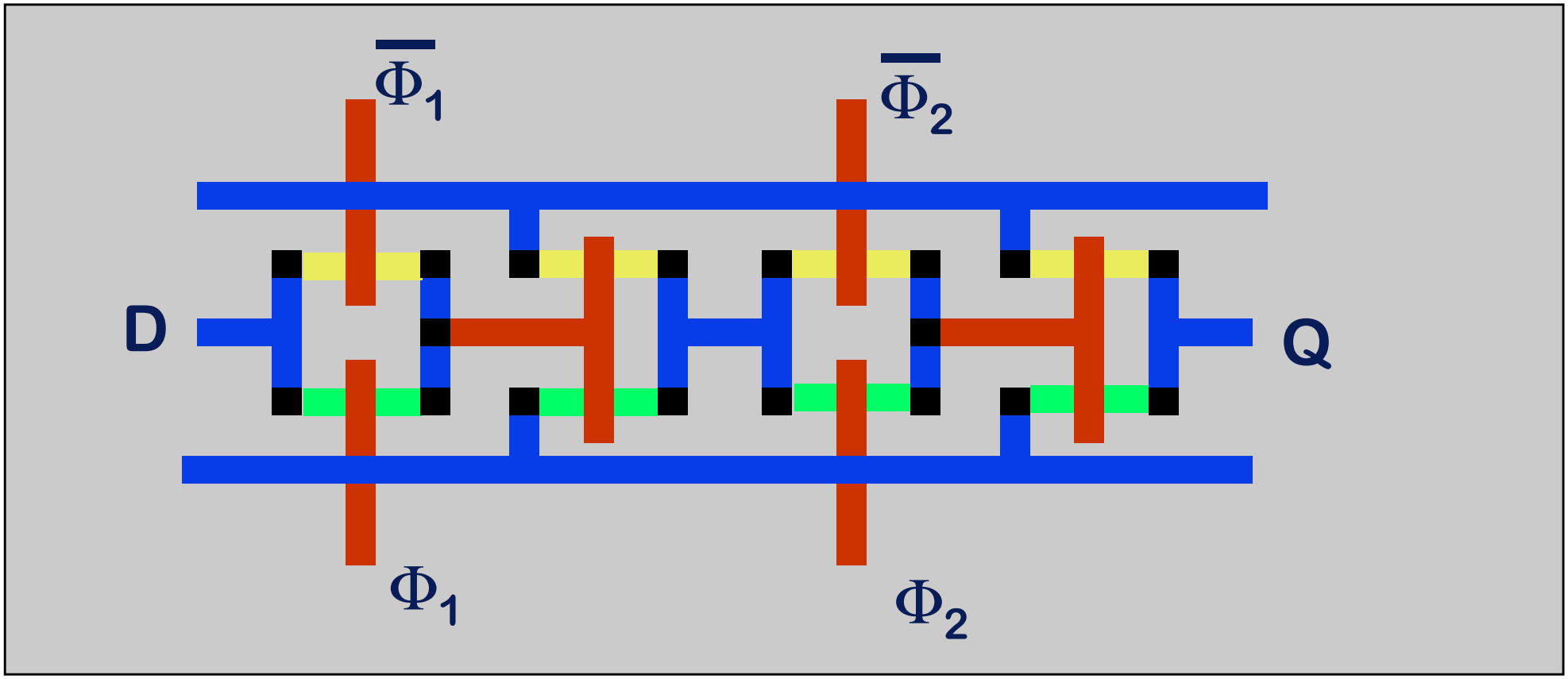


D-Edge Triggered DINAMICO



Layout simbolico: (8 transistor !)

Lay-Out Simbolico D-Edge Triggered DINAMICO



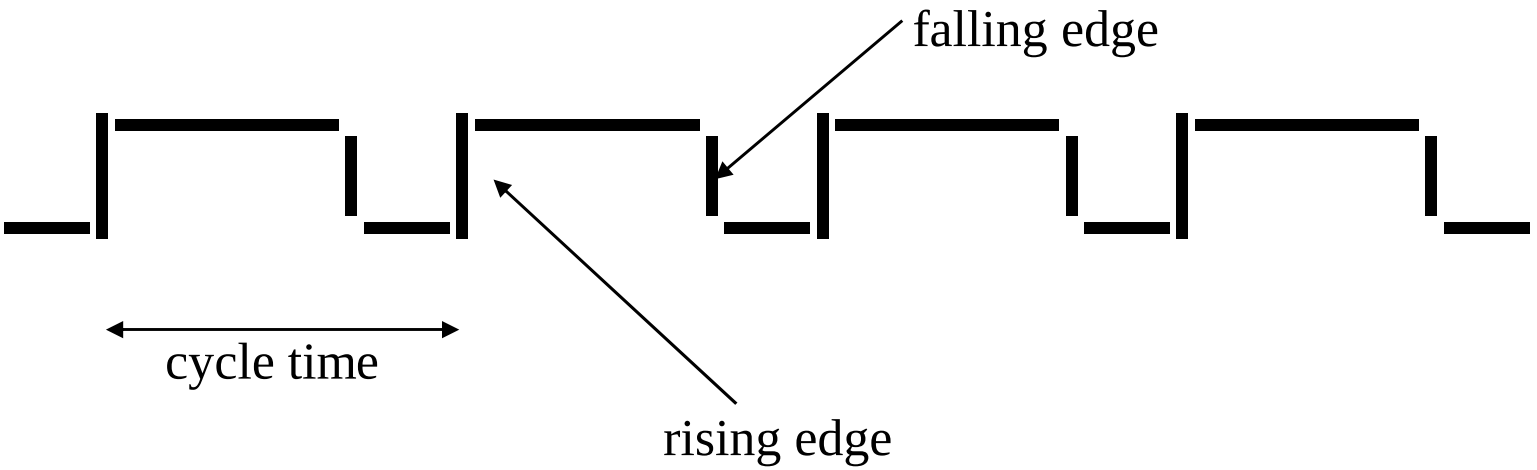
Realizzazione di elementi di memoria

- Un elemento di memoria si puo' anche realizzare SENZA ricorrere a reti combinatorie
  - DRAM (RAM Dinamica) → usa un condensatore
    - La carica deve essere rinnovata periodicamente o viene persa
  - STTRAM (Spin Transfer Torque RAM) → usa l'allineamento degli spin degli elettroni in modo da definire il suo stato attraverso una giunzione a tunnel magnetico (MJT)
    - Molto veloce e con durata illimitata ma non mantiene a lungo l'informazione e ha una bassa densita'
  - PCM (Phase Change Memory) → usa una lega calcogenura (Germanio, Antimonio, Tellurio = $Ge_2Sb_2Te_5$) interposta fra due elettrodi: il passaggio di corrente riscalda il materiale e induce un cambiamente di fase da amorfo a cristallino o viceversa
    - Moderatamente veloce e moderatamente densa, ma richiede abbastanza energia per memorizzare il bit, il materiale e' soggetto a invecchiamento e non e' semplice da fabbricare
  - RRAM (Resistive RAM o Memristor) → opera attraverso il movimento di ioni di ossigeno all'interno di una giunzione metallo-ossido (diossido di Titanio [1])
    - Ha stabilita' e durata molto lunghe ed e' sia molto veloce che densa essendo possibile stratificarle; migliora il comportamento piu' e' piccola

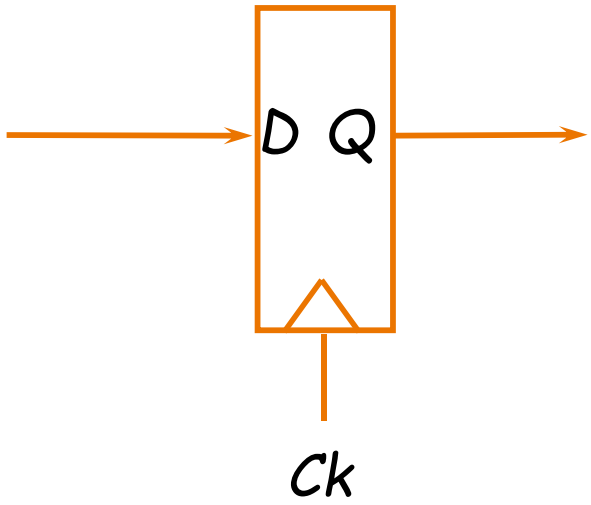
[1] Strukov, D. B.; Snider, G. S.; Stewart, D. R.; Williams, S. R. (2008), "The missing memristor found", *Nature* **453** (7191): 80–83, doi:10.1038/nature06932

Elementi che memorizzano uno stato

- Possono essere di tipo
  - asincrono (non richiede un segnale di clock)
  - sincrono (richiede un segnale di clock)
- Nella logica sequenziale di tipo sincrono
  - Il cambiamento dell'uscita avviene rispetto a un segnale di clock (dopo un certo tempo rispetto a un riferimento sul segnale di clock)
  - Gli ingressi sono sensibili in corrispondenza di un qualche riferimento sul segnale di clock:

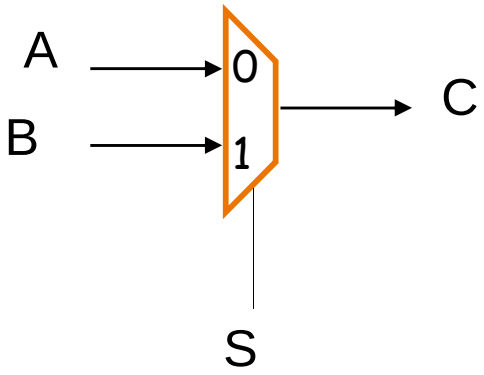


Simbolo del flip-flop D-Edge Triggered (flip-flop D)



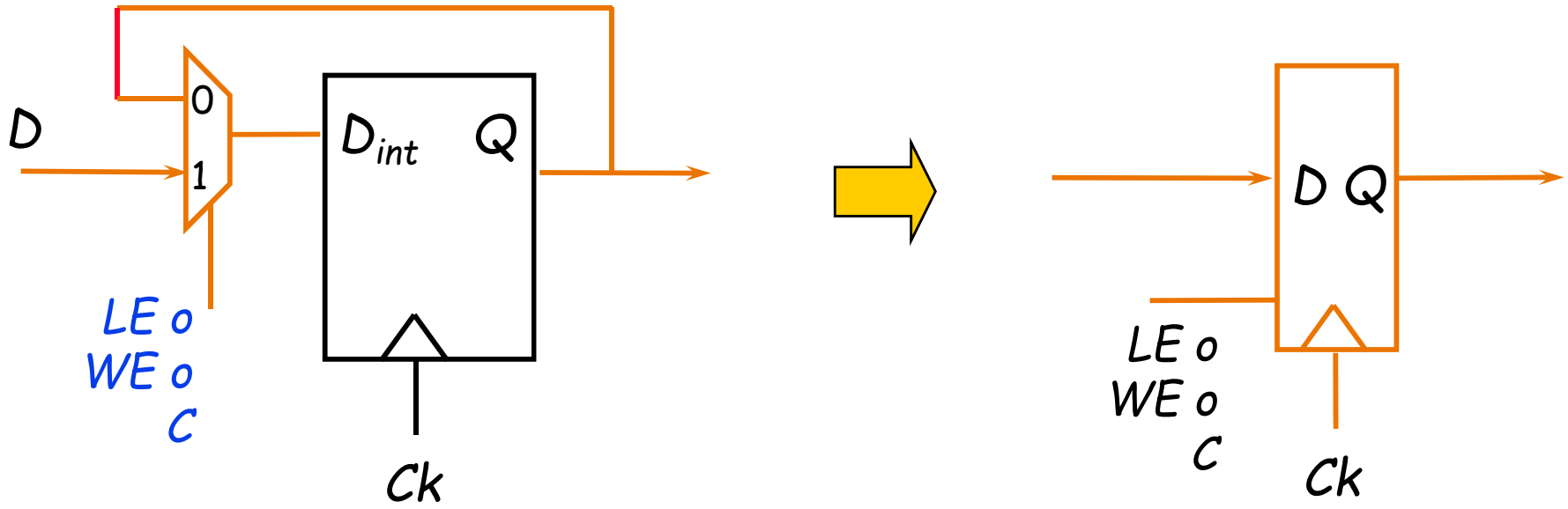
Il Multiplexer

- **Seleziona uno degli ingressi, in base a un segnale di controllo, e lo invia in uscita**



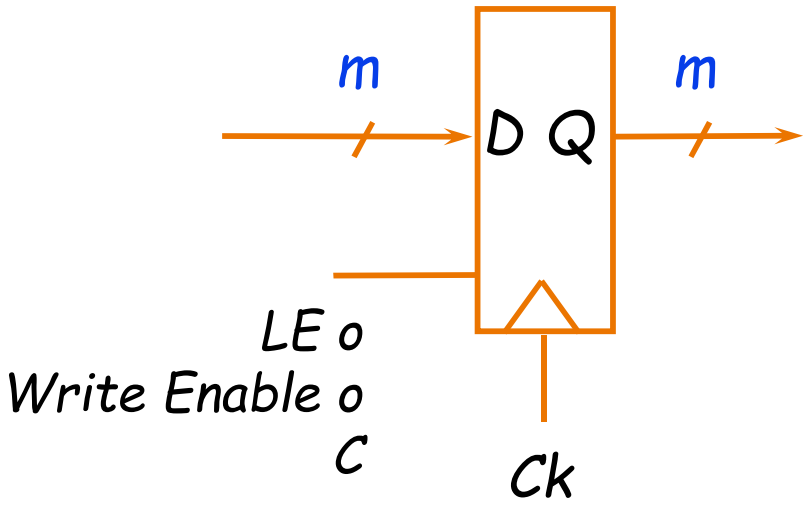
nota: chiamiamo questo "multiplexer a 2-ingressi" anche se in realta' ne ha 3

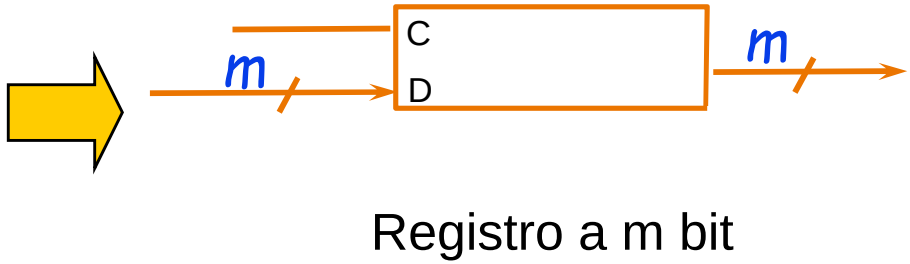
Cella di registro



- Aggiungo un multiplexer il cui controllo si chiama
  - LE = Latch Enable o anche
    WE = Write Enable o anche
    C = Control

Registro a m bit

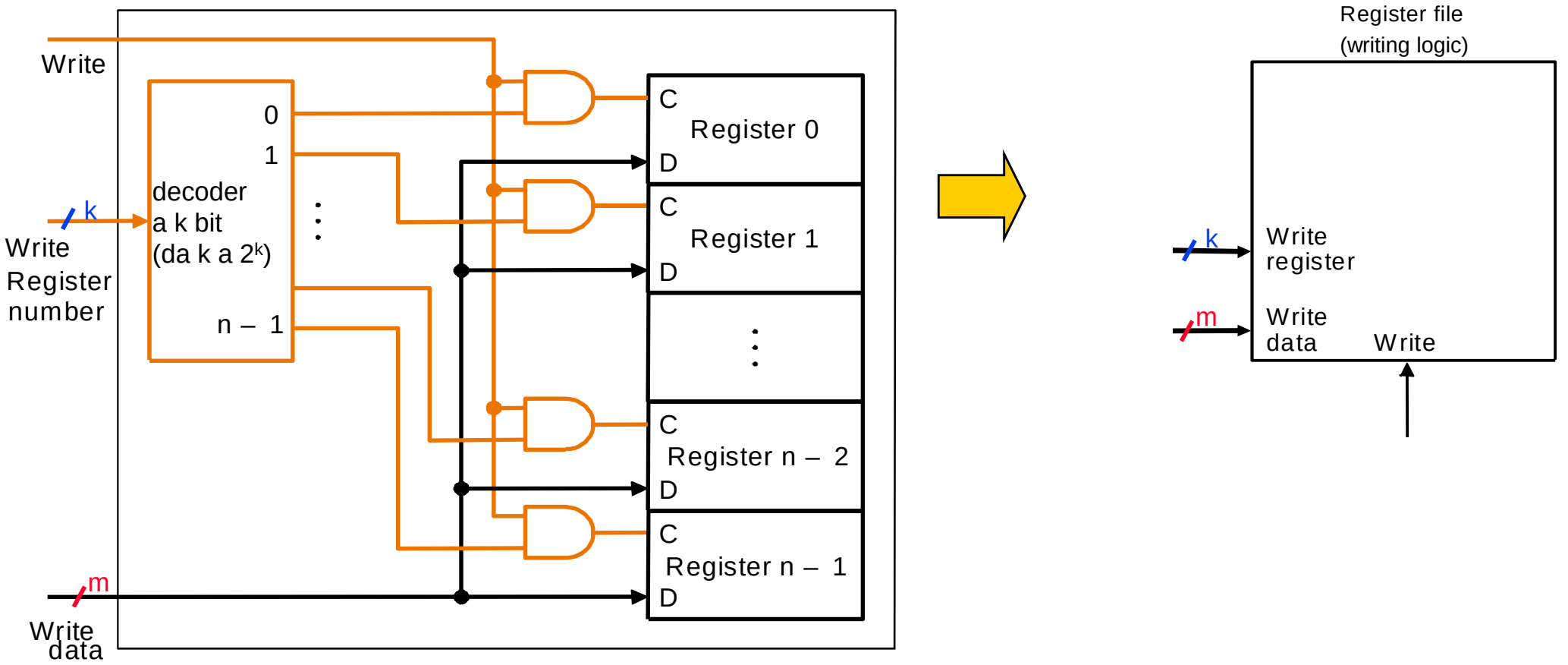




- Nell'ultima rappresentazione si intende implicitamente che sia presente un segnale Ck per il clock

- Register file:
  - Composto da n registri di questo tipo
  - Occorre della logica di scrittura
  - Occorre della logica di lettura

Register File: logica di scrittura



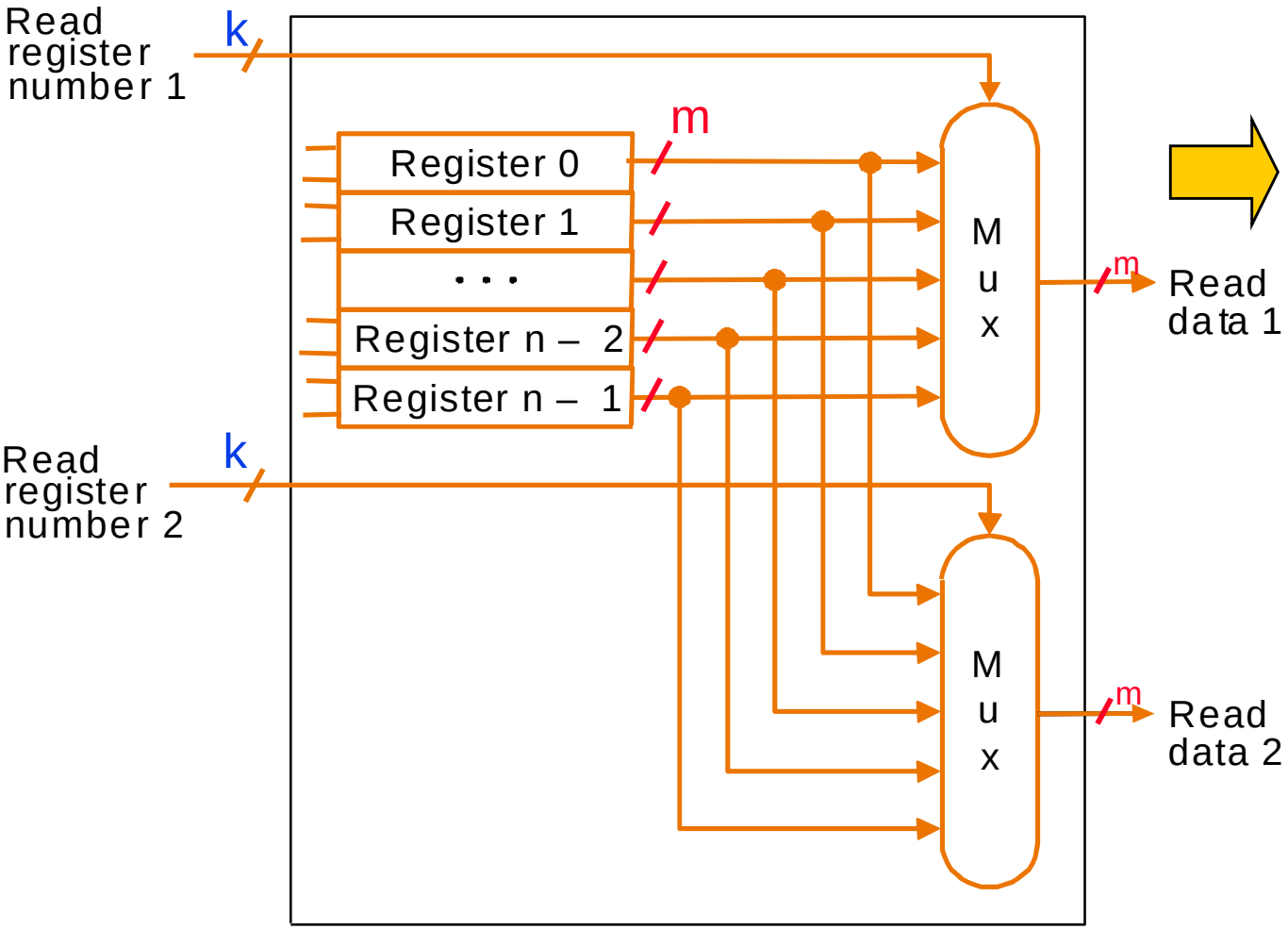
n = numero di registri disponibili (e.g. 32)

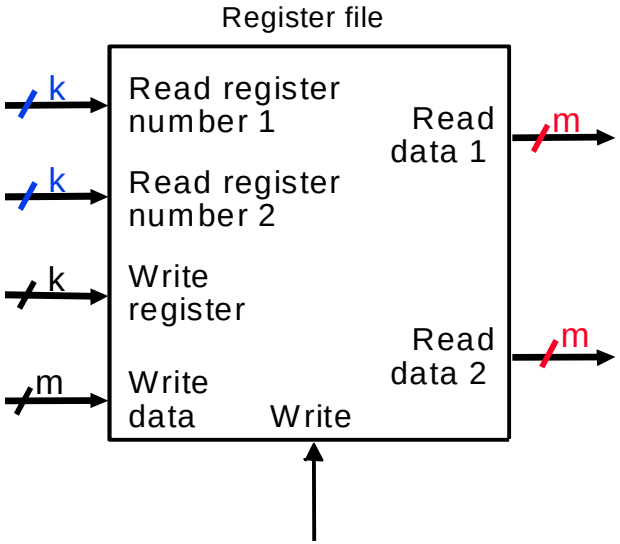
k = $\log_2(n)$ (e.g. 5)

m = larghezza dei registri (e.g. 32)

Register File: logica di lettura

- Si usano i flip-flop di tipo D

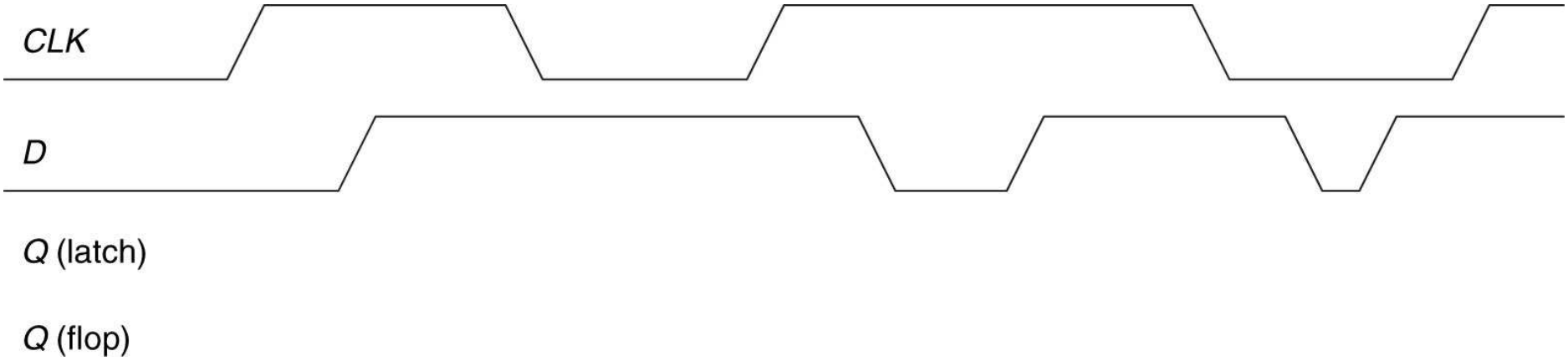




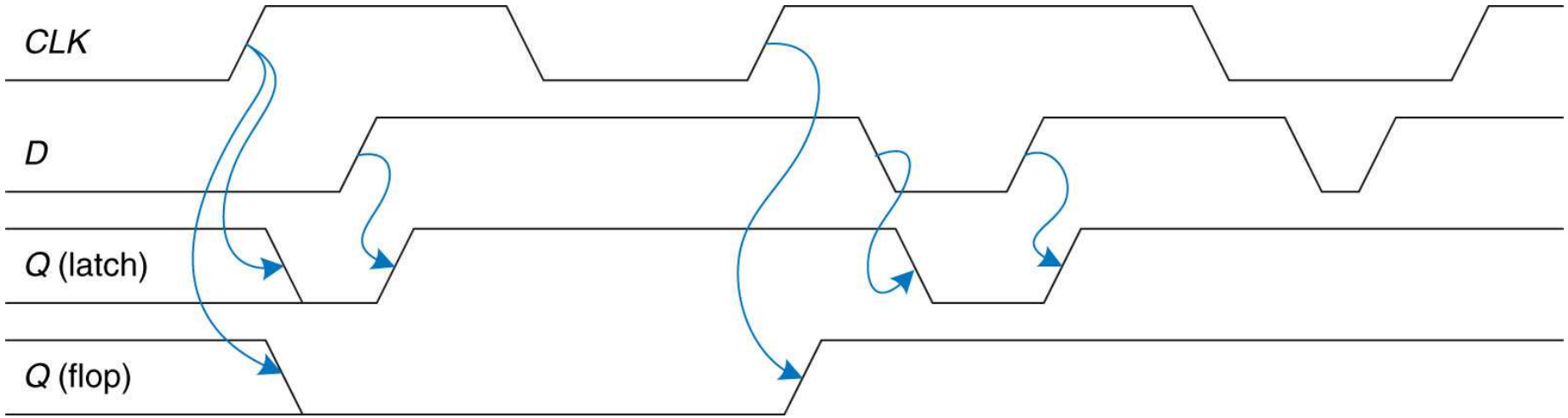
Riepilogo

- Latch e Flip-flop sono i blocchi architetturali fondamentali delle reti logiche sequenziali

- Esercizio
  - Vengono applicati i seguenti segnali D e CLK al D-latch e al D-flip-flop: disegnare le forme d'onda di uscita (Q) in entrambi i casi



Esercizio – D-latch e D-FF



Esercizio2

- **Descrivere un D-FF in Verilog**

```
module DFF(clock,D,Q,Qbar);
  input clock, D;
  output reg Q; // Q is a reg since it is assigned in an always block
  output Qbar;

  assign Qbar = ~ Q; // Qbar is always just the inverse of Q

  always @(posedge clock) // perform actions whenever the clock rises
    Q <= D;
endmodule
```