ARCHITETTURA DEGLI ELABORATORI

A.A. 2020-2021

Università di Napoli Federico II
Corso di Laurea in Informatica

Docenti
Proff. Luigi Sauro gruppo 1 (A-G)
Silvia Rossi gruppo 2 (H-Z)

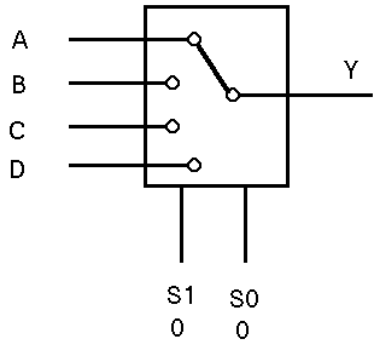
ALGEBRA DI BOOLE E RETI COMBINATORIE

Combinational Building Blocks

- Multiplexers
- Decoders

Multiplexer

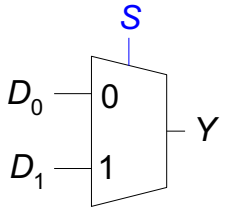
- Un multiplexer è sostanzialmente un selettore di linea



- In generale è costituito da N ingressi (dove N è una potenza di 2), 1 uscita, e $\log_2 N$ linee di selezione che indicano a quale ingresso deve corrispondere l'output

Multiplexer (Mux)

- Selects between one of *N* inputs to connect to output
- $\log_2 N$-bit select input – control input
- **Example:** **2:1 Mux**

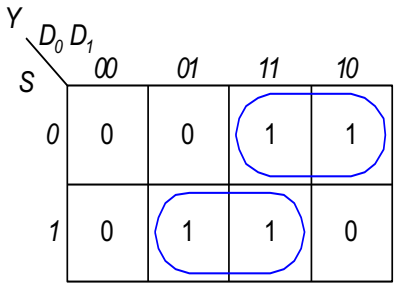


S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

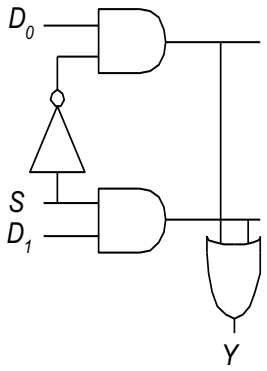
S	Y
0	D_0
1	D_1

Multiplexer Implementations

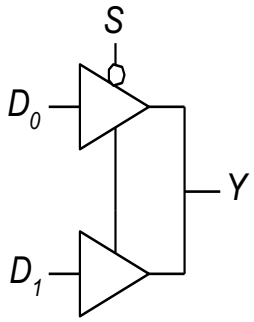
- **Logic gates**
  - Sum-of-products form



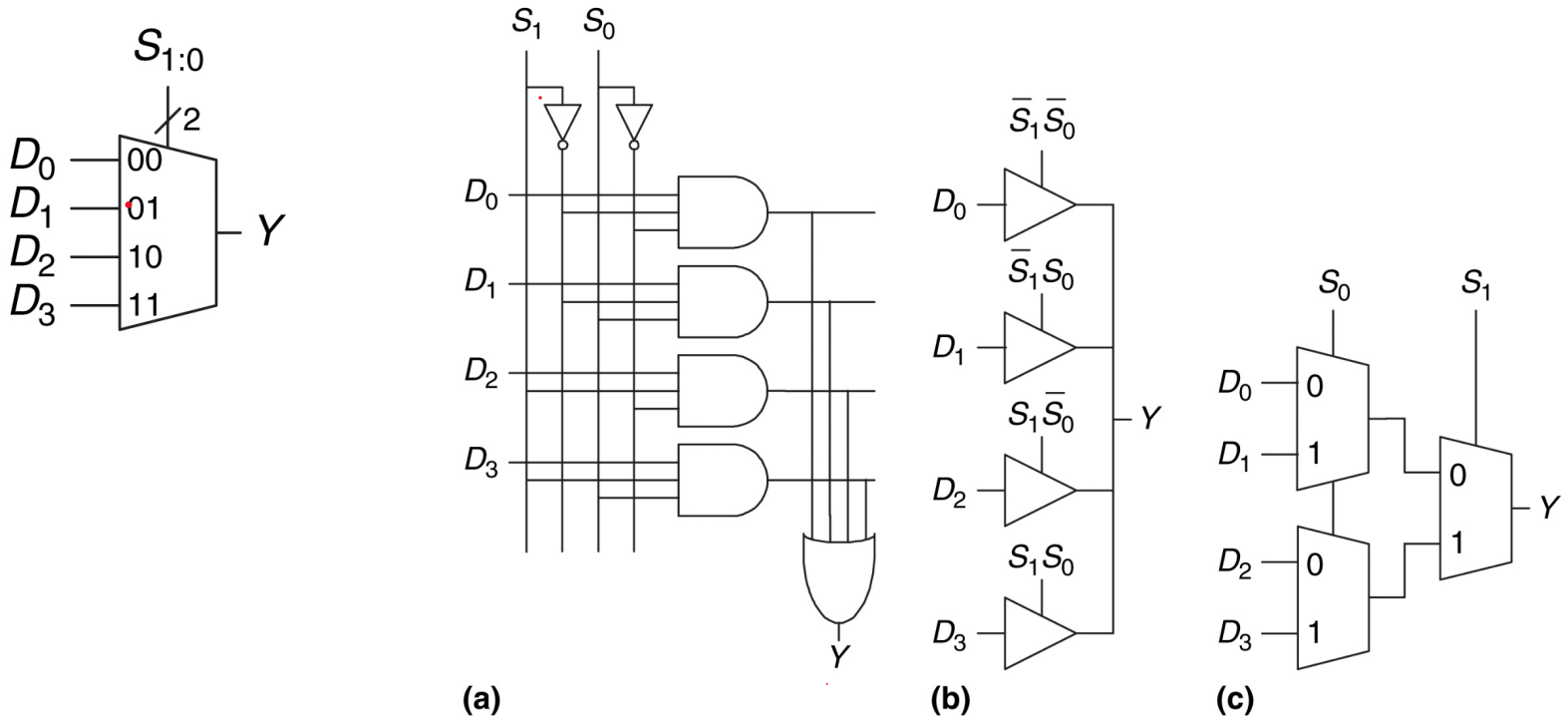
$Y = D_0\overline{S} + D_1S$



- **Tristates**
  - For an N-input mux, use N tristates
  - Turn on exactly one to select the appropriate input



Multiplexer 4:1

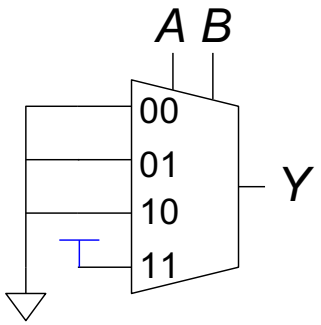


Logic using Multiplexers

Using mux as a lookup table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$

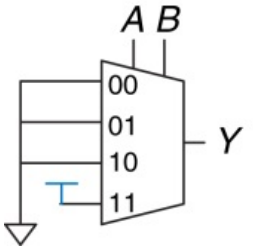


Sintetizzare funzioni booleane con mux

- I multiplexer possono essere usati anche per sintetizzare delle funzioni booleane
- Sintetizzare una funzione di m variabili con un mux a 2^m linee è molto semplice: le variabili saranno linee di selezione. Data una certa configurazione delle variabili, la linea di ingresso corrispondente sarà posta al valore della funzione in quella configurazione
- Di fatto le linee di ingresso riproducono la tabella di verità della funzione

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$

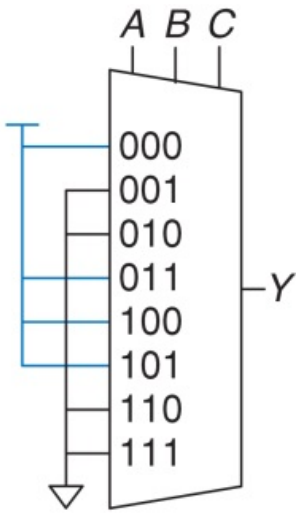


Sintetizzare funzioni booleane con mux

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$Y = A\overline{B} + \overline{B}\overline{C} + \overline{A}BC$

(a)



(b)

Y= Σ(0,3,4,5)

Sintetizzare funzioni booleane con mux

- E’ possibile utilizzare un mux con 2^{m-1} ingressi per sintetizzare un funzione ad m variabili
- Le prime m-1 variabili saranno linee di selezione, mentre le linee di ingresso possono essere poste a 0,1, oppure all’ultima variabile (positiva o negata)

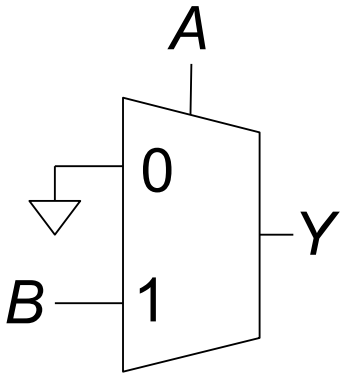
Logic using Multiplexers

Reducing the size of the mux

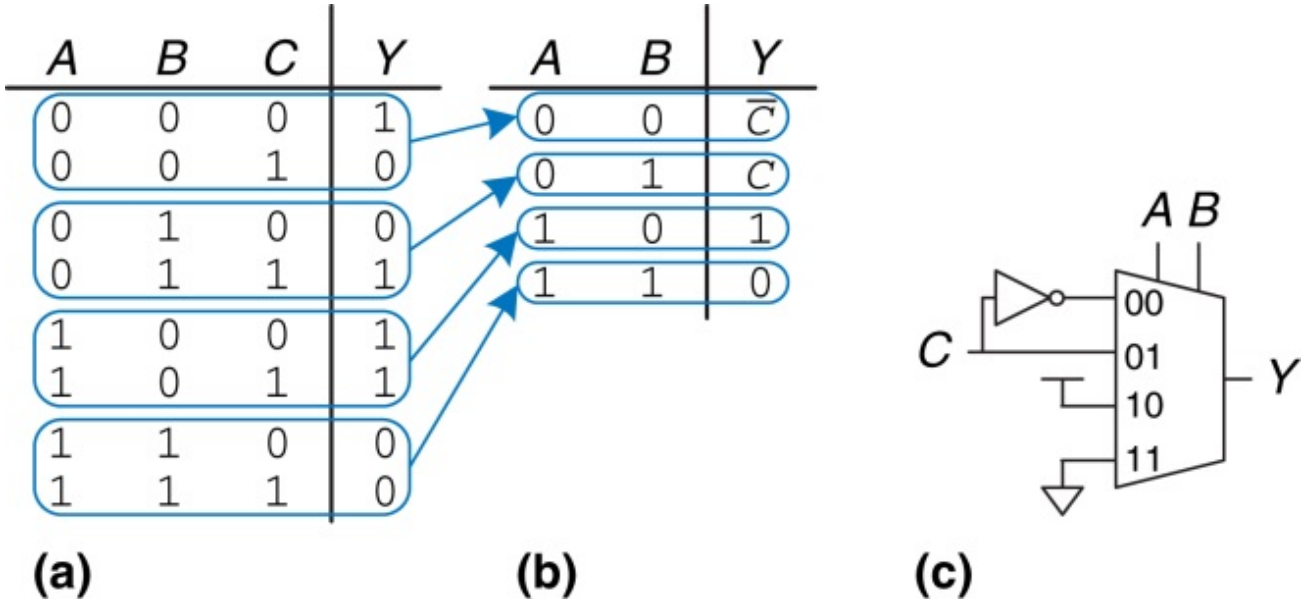
$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	Y
0	0
1	B

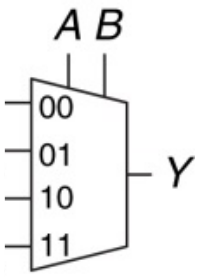


Sintetizzare funzioni booleane con mux



Sintetizzare funzioni booleane con mux

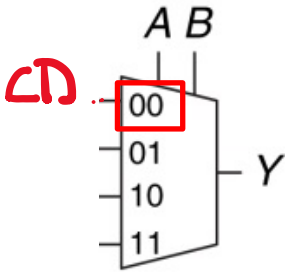
Supponiamo di avere mux 4:1 e di voler sintetizzare una funzione di 4 variabili



A B C D	Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	0
1 0 0 0	0
1 0 0 1	1
1 0 1 0	1
1 0 1 1	0
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

Sintetizzare funzioni booleane con mux

Supponiamo di avere mux 4:1 e di voler sintetizzare una funzione di 4 variabili

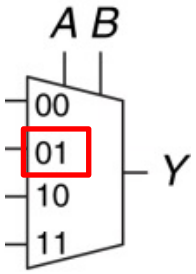


A B C D	Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	0
1 0 0 0	1
1 0 0 1	1
1 0 1 0	0
1 0 1 1	0
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

CD

Sintetizzare funzioni booleane con mux

Supponiamo di avere mux 4:1 e di voler sintetizzare una funzione di 4 variabili

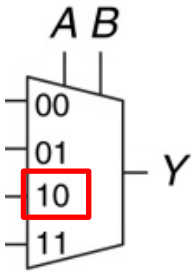


A B C D	Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	0
1 0 0 0	1
1 0 0 1	1
1 0 1 0	0
1 0 1 1	0
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

$\bar{C}$

Sintetizzare funzioni booleane con mux

Supponiamo di avere mux 4:1 e di voler sintetizzare una funzione di 4 variabili

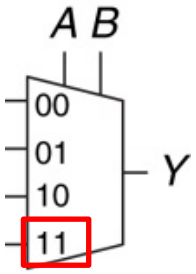


A B C D	Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	0
1 0 0 0	0
1 0 0 1	1
1 0 1 0	1
1 0 1 1	0
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

$C \oplus D$

Sintetizzare funzioni booleane con mux

Supponiamo di avere mux 4:1 e di voler sintetizzare una funzione di 4 variabili

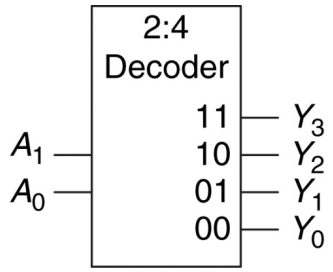


A B C D	Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	0
1 0 0 0	0
1 0 0 1	1
1 0 1 0	1
1 0 1 1	0
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

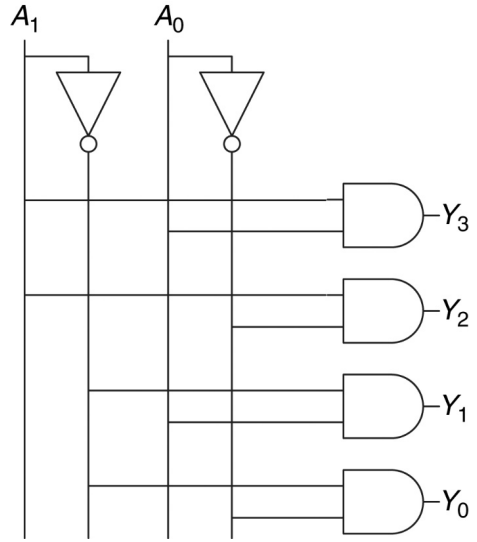
1

Decoder

- Un decoder ha N linee di ingresso e 2^N linee di uscita
- se m è numero rappresentato dagli input allora solo l'm-esima linea di uscita è pari a 1 mentre tutte le altre sono a 0

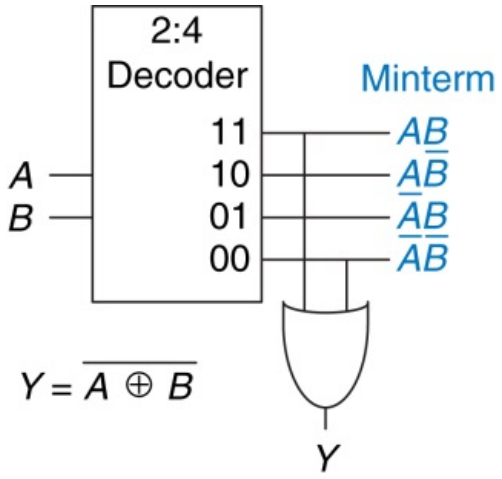


A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



Sintetizzare funzioni booleane con decoder

- Anche i decoder possono essere usati per sintetizzare funzioni booleane
- Basta mettere in OR tutte e solo le linee di uscita che occorrono nella sigma-espressione della funzione da sintetizzare



$Y = \Sigma(0, 3)$

CIRCUITI SEQUENZIALI

Logica sequenziale

- Nei sistemi sequenziali l'output dipende sia dal valore corrente sia da valori precedenti dell'input. In tal senso si dice che il sistema ha *memoria*
- *Stato interno:* rappresenta l'informazione che mantiene la *storia* di un circuito sequenziale ed è necessaria per prevedere il suo comportamento futuro
  - Come vedremo lo stato di un sistema sarà memorizzato in componenti come i latches e flip-flop

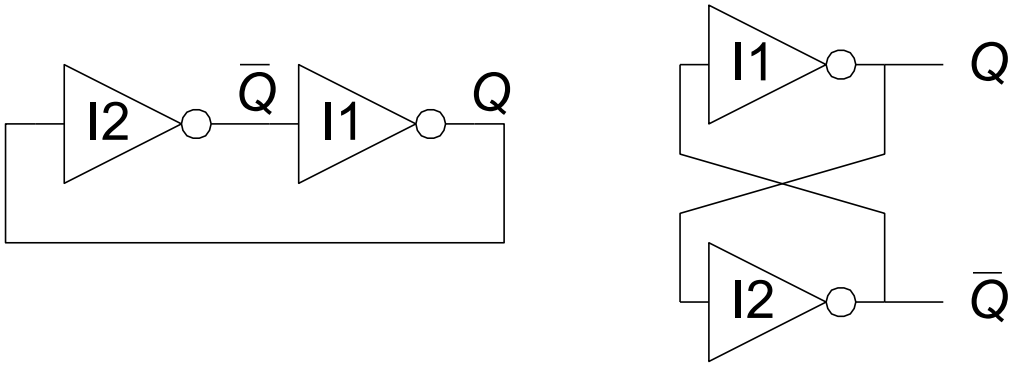
- Un circuito sequenziale (*sincrono*) avrà una topologia ben precisa:
  - logica combinatoria: definisce l'evoluzione del sistema
  - Banchi di flip-flop: servono a memorizzare gli stati del sistema
- Un aspetto peculiare dei sistemi sequenziali è quello della retroazione (*feedback*), ovvero il segnali di output vengono riportati in input

State elements

- Lo stato di un circuito influenzerà l'evoluzione del sistema
- Gli *state elements* sono tutte quelle componenti circuitali che vengono adoperate per memorizzare lo stato di un circuito
  - Circuiti bistabili
  - SR Latch
  - D Latch
  - D Flip-flop

Circuito bistabile

- *building block* per altri state elements
- Two outputs: Q, $\overline{Q}$
- No inputs

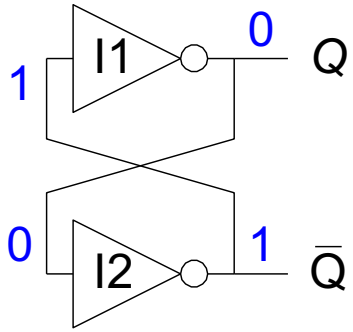


Analisi del circuito bistabile

- Considera i due possibili casi:

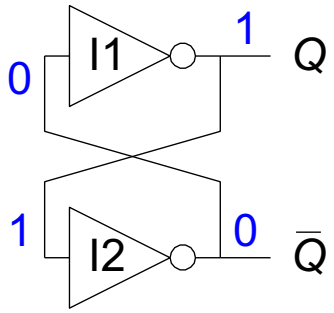
  Q = 0:

  allora $Q = 0$, $\bar{Q} = 1$ (consistente)



  Q = 1:

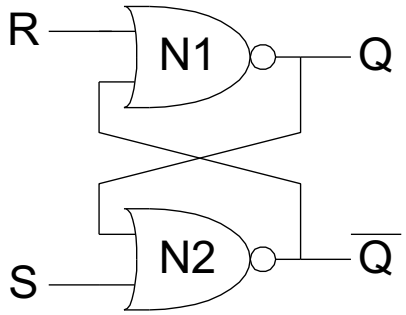
  allora $Q = 1$, $\bar{Q} = 0$ (consistente)



- Memorizza 1 bit nella variabile di stato Q (or $\bar{Q}$)
- Ma non ci sono input per controllare questo stato!

SR (Set/Reset) Latch

- SR Latch



- Consideriamo i 4 possibili stati:

  $S = 1, R = 0$

  $S = 0, R = 1$

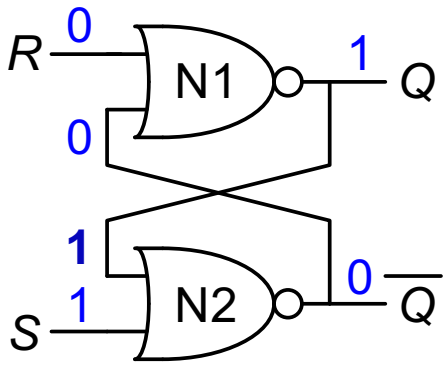
  $S = 0, R = 0$

  $S = 1, R = 1$

Analisi di un SR Latch

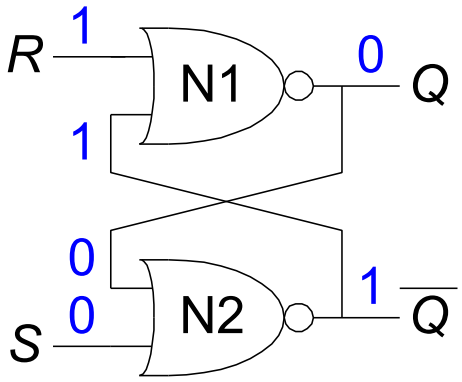
$S = 1, R = 0$:

allora $Q = 1$ e $\overline{Q} = 0$



$S = 0, R = 1$:

allora $Q = 0$ e $\overline{Q} = 1$

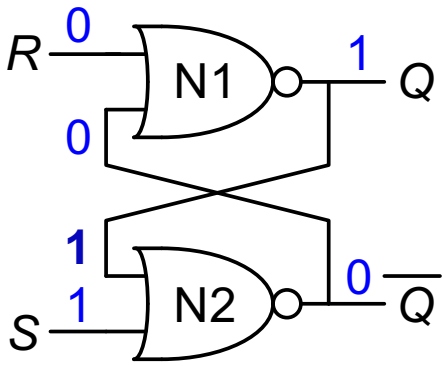


Analisi di un SR Latch

***S* = 1, *R* = 0:**

allora $Q = 1$ e $\overline{Q} = 0$

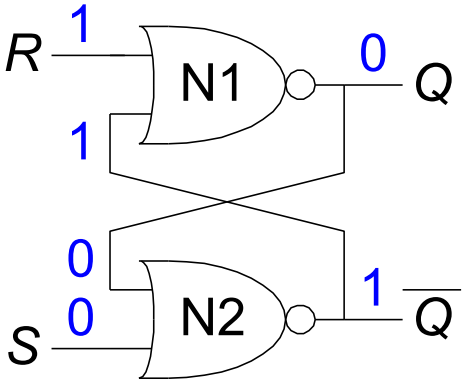
operazione Set



***S* = 0, *R* = 1:**

allora $Q = 0$ e $\overline{Q} = 1$

operazione Reset

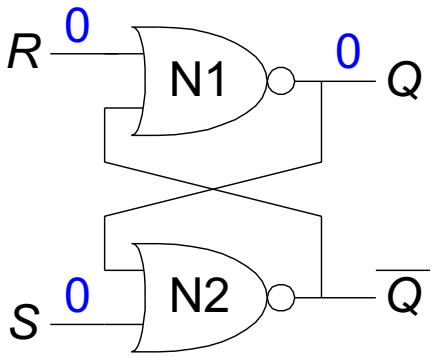


Analisi di un SR Latch

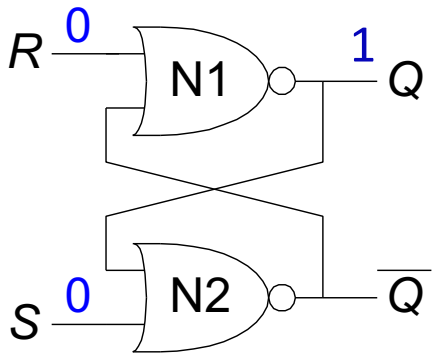
$S = 0, R = 0$:

allora $Q = Q_{prev}$

$Q_{prev} = 0$

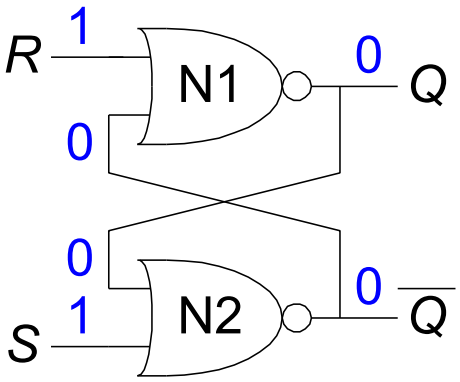


$Q_{prev} = 1$



$S = 1, R = 1$:

allora $Q = 0$, $\bar{Q} = 0$

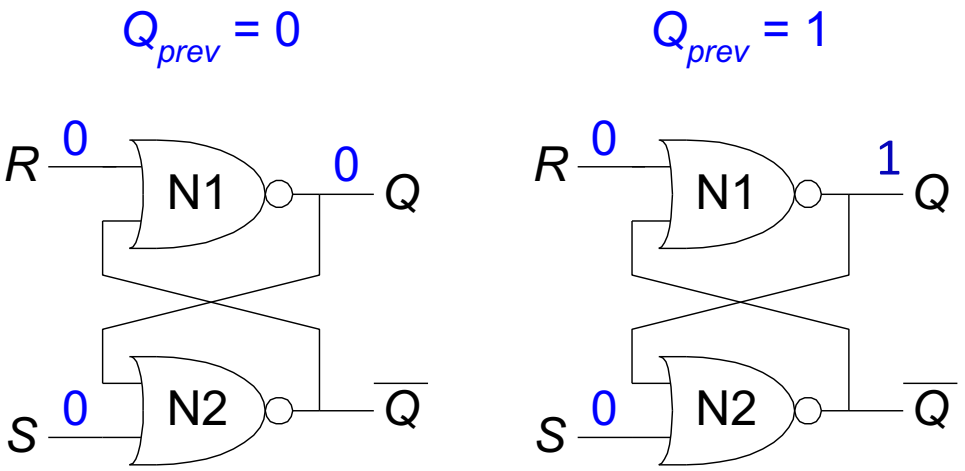


Analisi di un SR Latch

$S = 0, R = 0$:

allora $Q = Q_{prev}$

Memoria

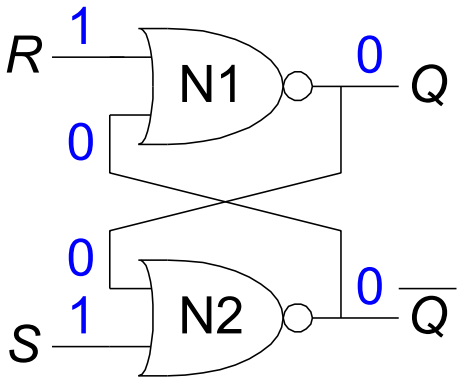


$S = 1, R = 1$:

allora $Q = 0, \bar{Q} = 0$

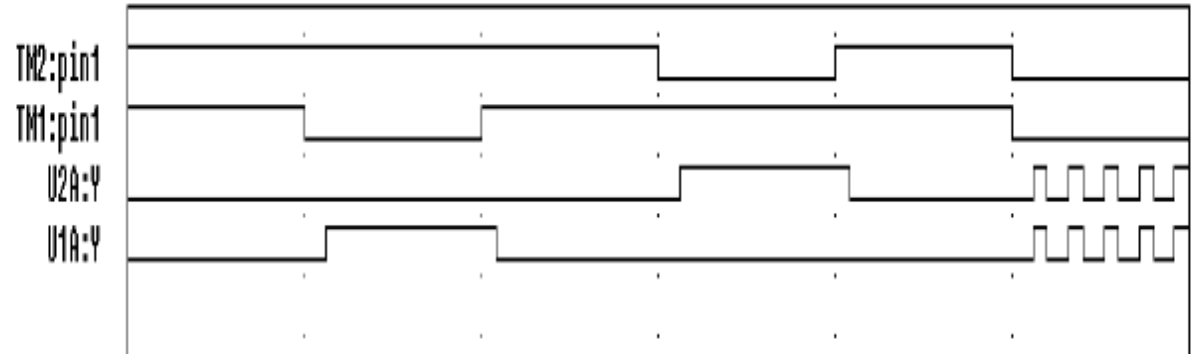
Stato non valido

$Q \neq$ NOT $\bar{Q}$



Analisi di un SR Latch

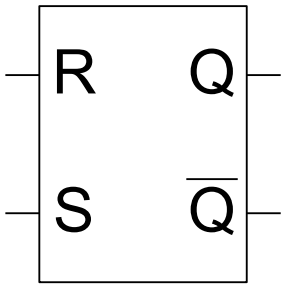
- Se dalla condizione S=R=1 si passa alla condizione S=R=0 allora
  - Se i tempi di propagazione sono uguali allora il circuito va in oscillazione
  - Nell'ipotesi, più realistica che le porte abbiano ritardi anche lievemente differenti, il circuito si mette in uno dei due stati possibili. Anche in questo caso, però, lo stato finale non è predicibile



Simbolo per un SR Latch

- SR sta per Set/Reset
  - Memorizza un bit (*Q*)
- **Set:** Pone l'output a 1
  (*S* = 1, *R* = 0, *Q* = **1**)
- **Reset:** Pone l'output a 0
  (*S* = 0, *R* = 1, *Q* = **0**)
- **Memoria:** mantiene memoria dell'output
  (*S* = 0, *R* = 0, *Q* = Q_{prev})

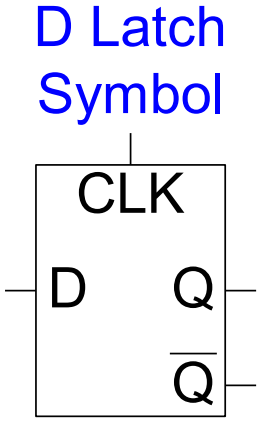
SR Latch Symbol



Occorre evitare lo stato non valido S = R = 1

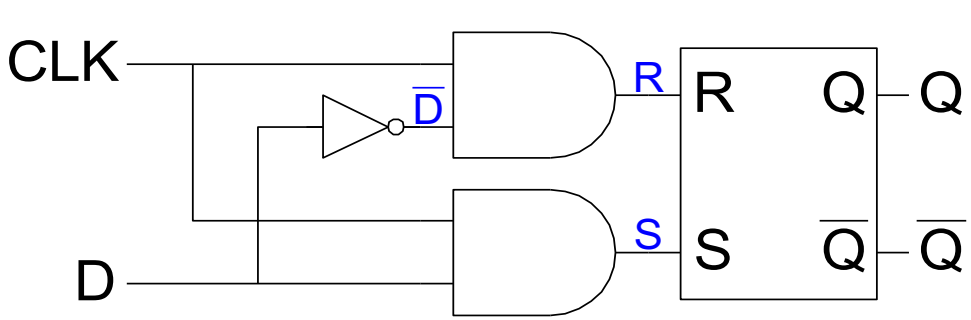
D Latch

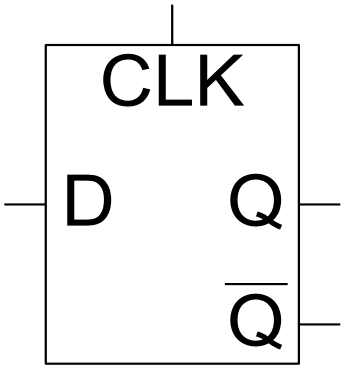
- 2 input: *CLK*, *D*
- ***CLK*:** controlla *quando* l'output cambia
- ***D*** (data input): controlla *in che cosa* l'output cambia
- Se ***CLK* = 1**,
  D passa fino a Q (*transparente*)
- Se ***CLK* = 0**,
  Q mantiene il suo valore precedente (*opaco*)



Evita lo stato non valido in cui $Q \neq \text{NOT } \overline{Q}$

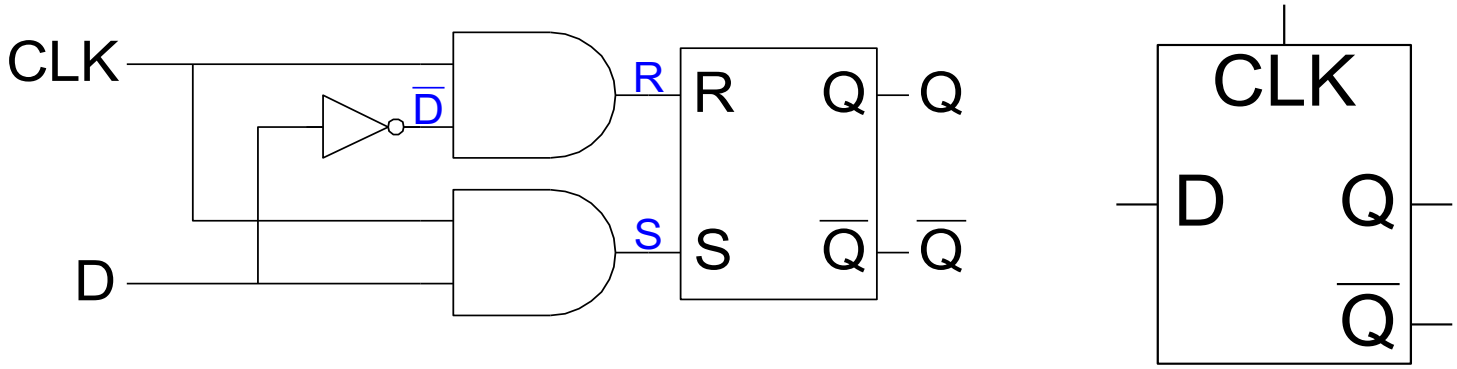
D Latch Internal Circuit





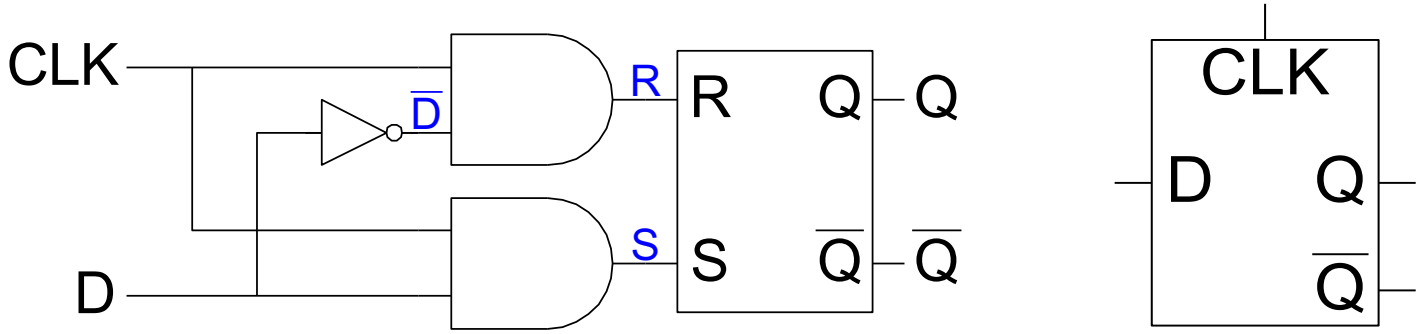
CLK	*D*	$\overline{D}$	*S*	*R*	*Q*	$\overline{Q}$
0	X					
1	0					
1	1					

D Latch Internal Circuit



CLK	*D*	$\overline{D}$	*S*	*R*	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

D Latch



CLK	*D*	$\overline{D}$	*S*	*R*	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0