Circuiti sequenziali

Circuiti sequenziali

e

applicazioni

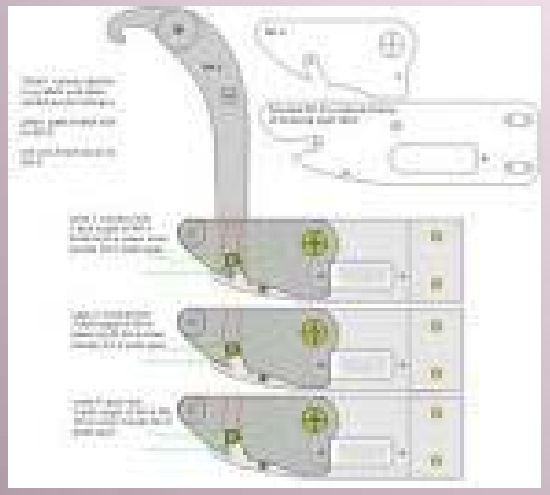
Circuiti sequenziali

Prima di poter parlare delle memorie è utile dare un accenno ai circuiti sequenziali.

Per circuiti sequenziali intendiamo tutti quei circuiti la cui variabile di uscita oltre che a dipendere dall'ingresso, dipende anche dall'uscita o dallo stato precedente.

In elettronica, i circuiti sequenziali sono i latch e i flip flop

Latch



Latch = chiavistello

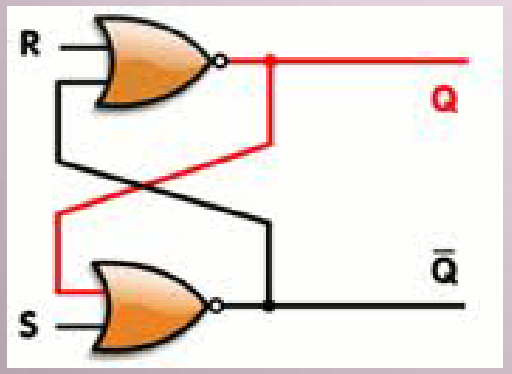
I latch si classificano in:

- S-R
- D

I latch SR sono detti anche bistabili in quanto hanno due possibili uscite

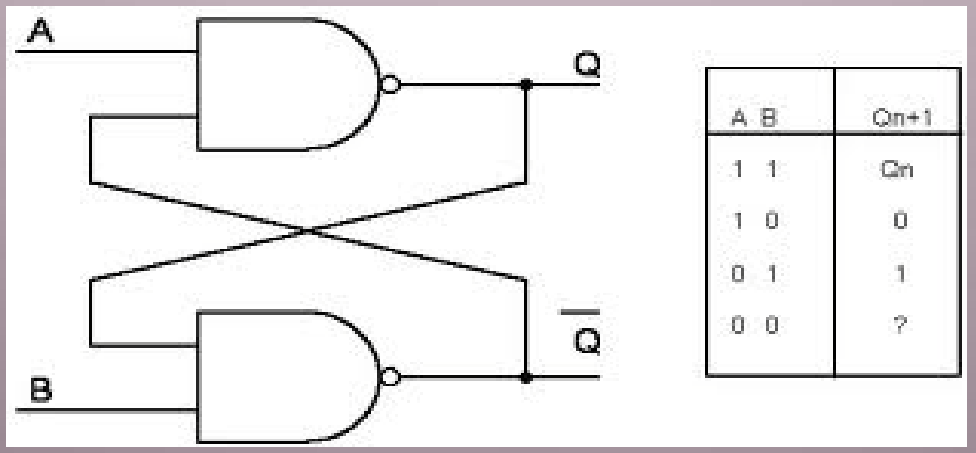
Latch S R

- Latch SR con NOR



S	R	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	?

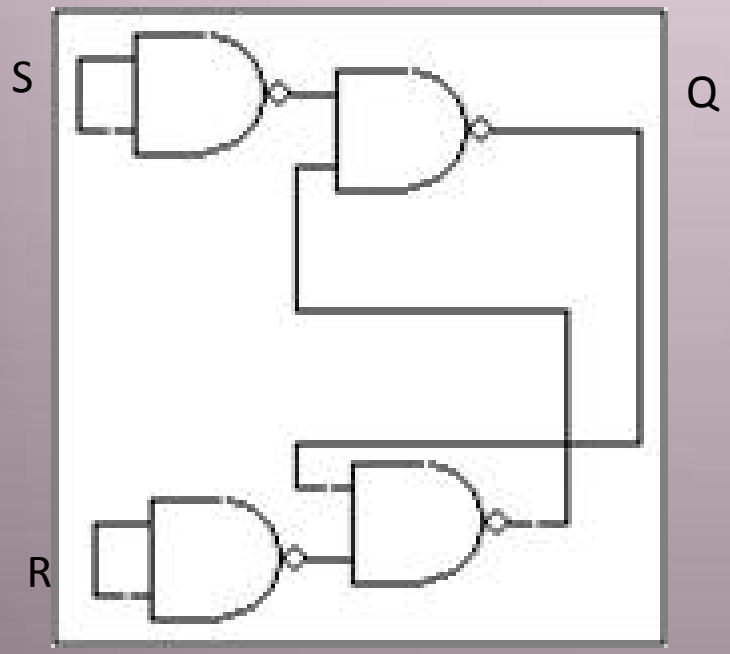
- Latch SR con NAND



A B	Qn+1
1 1	Qn
1 0	0
0 1	1
0 0	?

Latch SR

- Con opportuni accorgimenti, il latch SR sia con sole NOR che con sole NAND ha la seguente tabella della verità



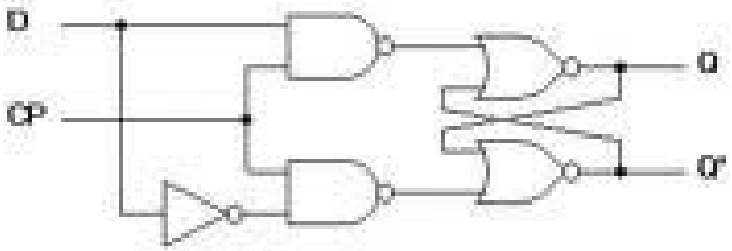
S	R	Q(i+1)
0	0	Q(i)
0	1	0
1	0	1
1	1	?

Flip flop

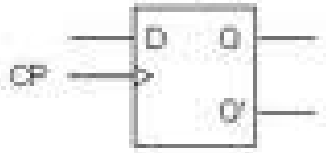
I latch cambiano il loro stato quando sono abilitati ma, se posti in cascata c'è il problema della corsa critica

Il problema della corsa critica viene risolto dai flip flop che commutano solo sul fronte di salita o discesa del ck

Latch D e flip flop D



(a) Logic diagram with NAND gates



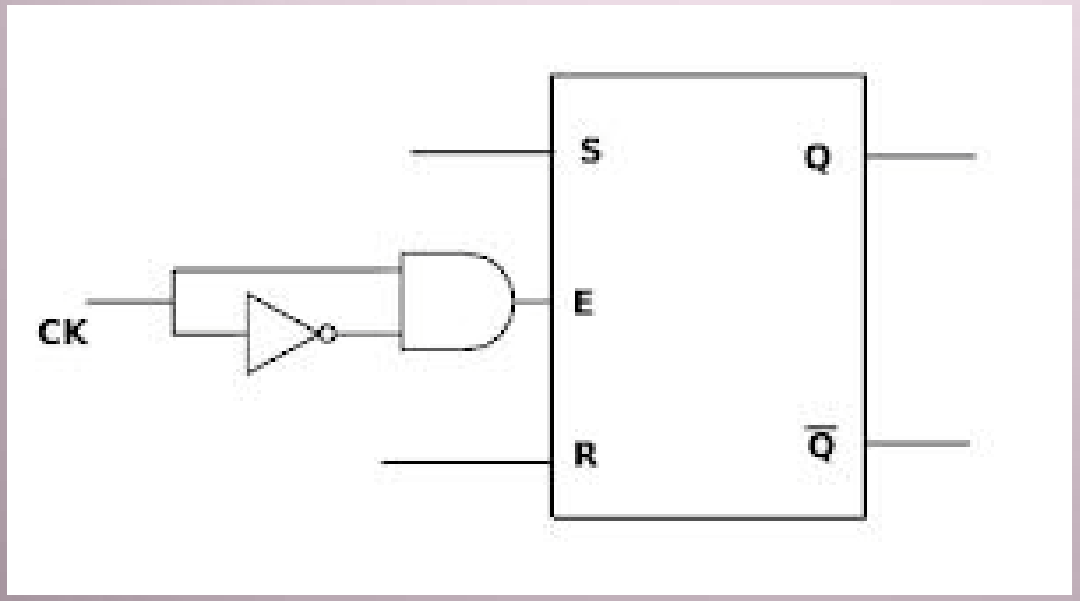
(b) Graphical symbol

Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

(c) Transition table

Clocked D flip-flop

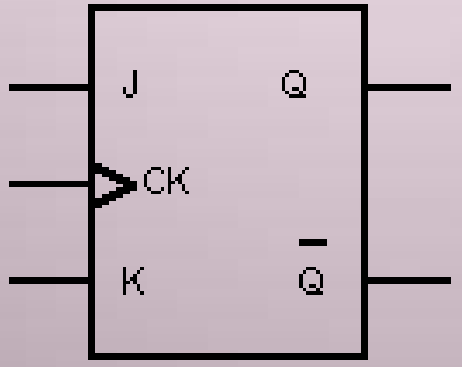
Flip Flop SR



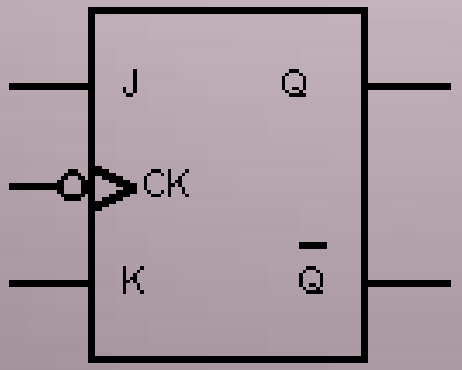
ck	S	R	$Q_{(i+1)}$
0	X	X	$Q_{(i)}$
1	X	X	$Q_{(i)}$
↑	0	0	$Q_{(i)}$
↑	0	1	0
↑	1	0	1

Il flip flop SR sfrutta l'alea statica, il ritardo di propagazione nella porta NOT. In questo modo si evita la corsa critica. Il Flip flop fa sempre da memoria ma commuta sul fronte di salita (può commutare anche sul fronte di discesa se si nega il clock). Resta il problema dell'ingresso 1 1 non accetabile.

Flip flop JK



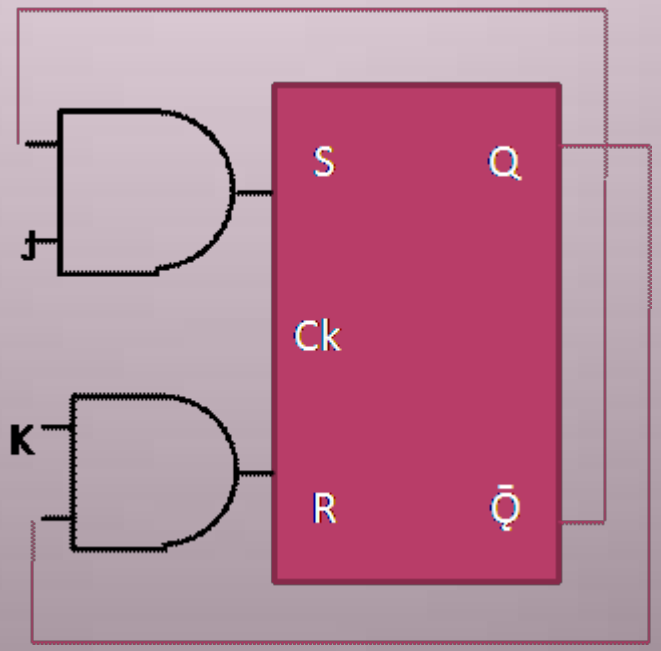
CK	J	K	Q
0	X	X	Q_0
↑	0	0	Q_0
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q_0}$



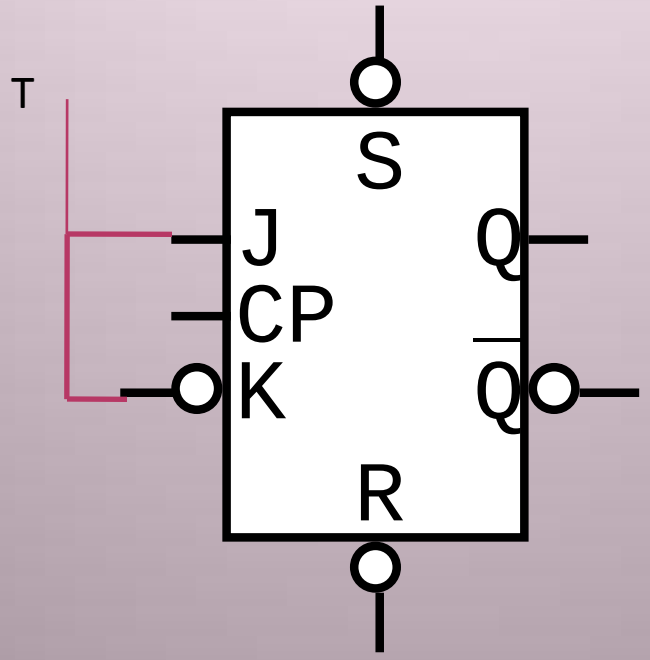
CK	J	K	Q
0	X	X	Q_0
↓	0	0	Q_0
↓	0	1	0
↓	1	0	1
↓	1	1	$\overline{Q_0}$

Da SR A JK

Il flip flop JK risolve il problema degli ingressi 11

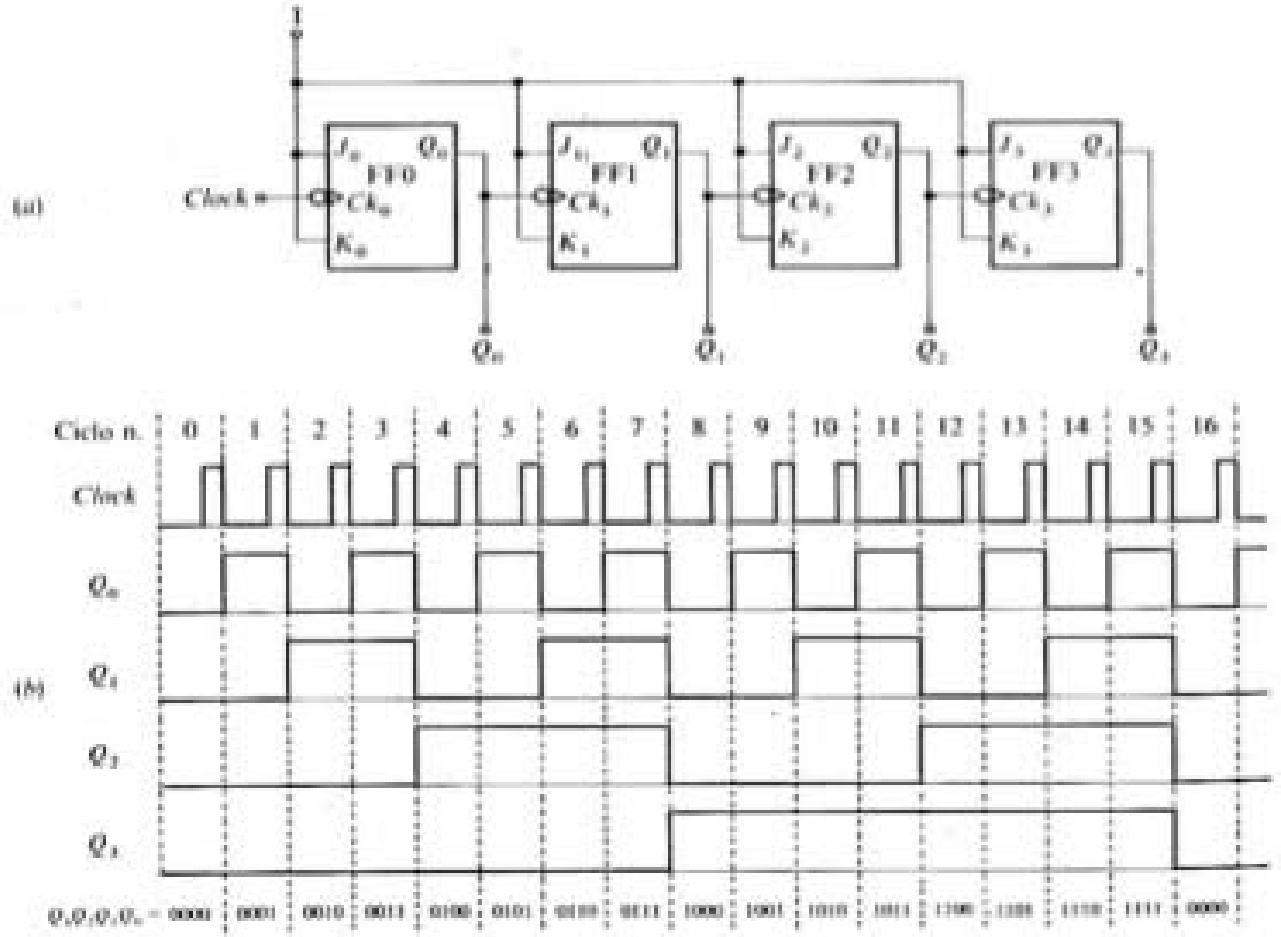


Flip Flop T=Toggle



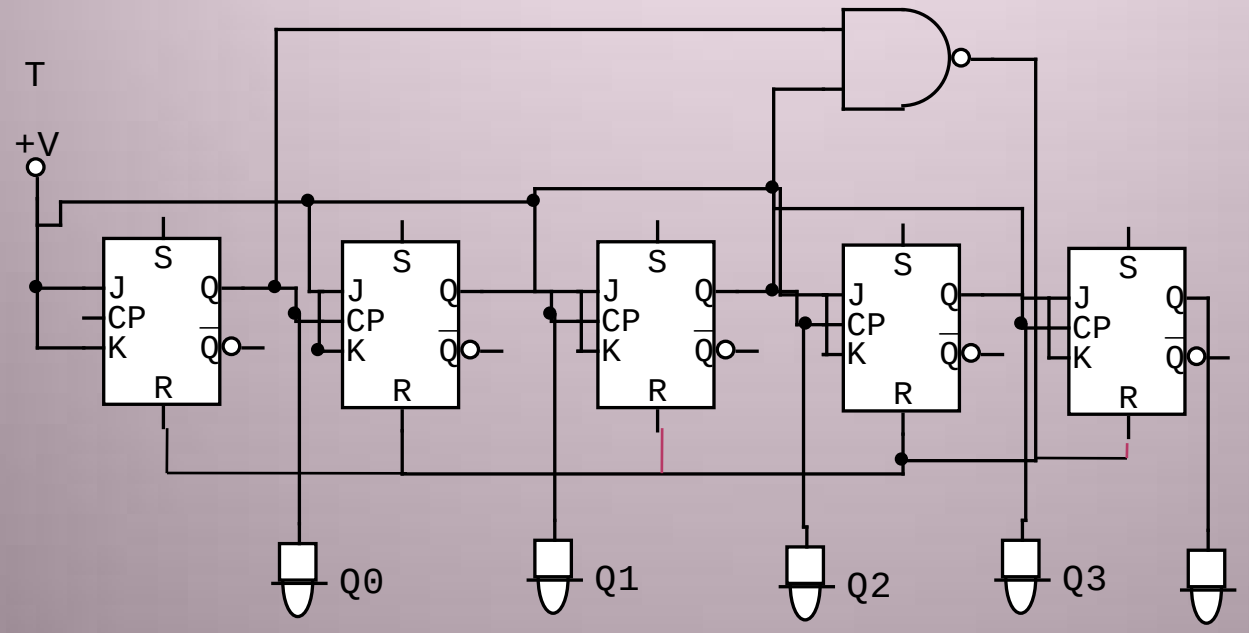
Ck	T	Q(i+1)
0	0	mem
1	1	mem
↑	0	mem
↑	1	$\bar{Q}(i)$

Contatore asincrono in avanti modulo n



Il contatore è formato da **n** flip flop di tipo t in cascata; sono detti asincroni perchè il clock sincronizza solo il primo flip flop.
L'uscita del primo fa da clock al secondo e così via.
Il contatore conta fino a 2^n-1

Contatore modulo qualunque



Il contatore modulo qualunque può contare fino a quando vogliamo con opportuni Accorgimenti. Nella figura la porta Nand ha come ingressi le uscite del primo e terzo flip flop. L'uscita della porta viene posta sul reset

Ciò vuol dire che solo quando i due ingressi sono uguali a 1 l'uscita della NAND è zero e fa da reset al contatore che riprende daccapo.

Contatore modulo qualunque

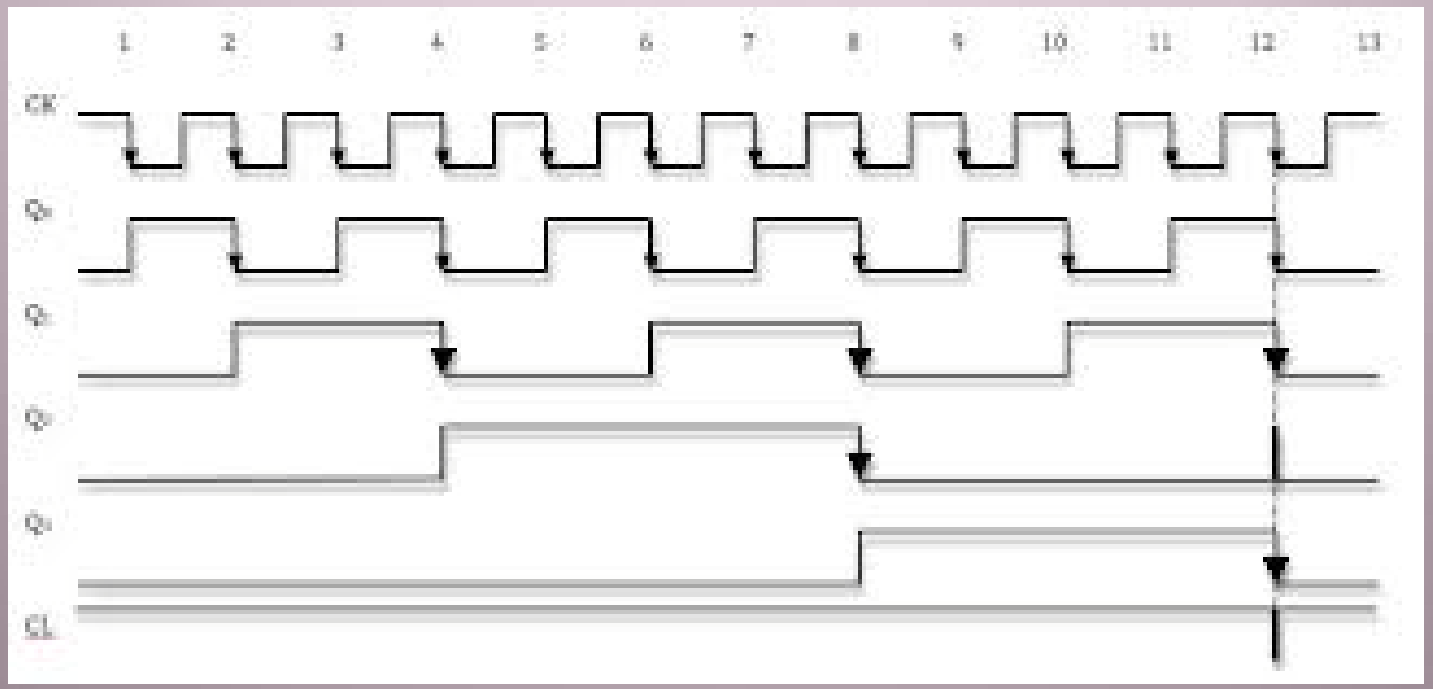
Q3	Q2	Q1	Q0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Il contatore della figura precedente conta fino a 5; la riga corrispondente è colorata in arancione

Il grafico successivo è quello di un contatore fino a 12; la colonna corrispondente è colorata in azzurro.

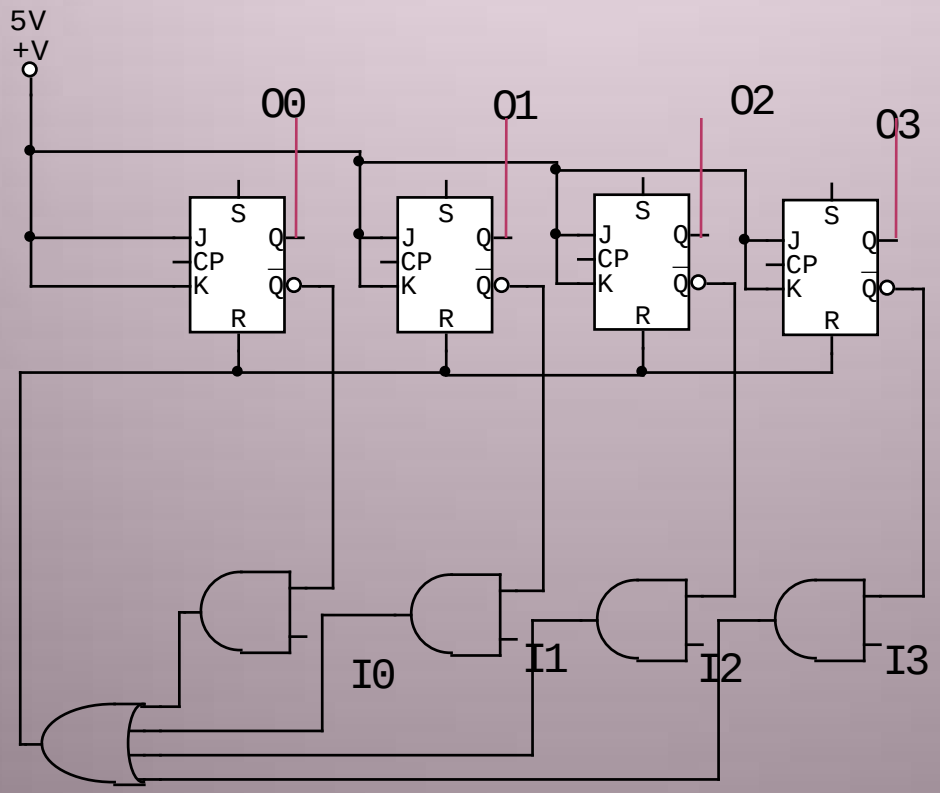
Per fare ciò, bisogna mettere gli ingressi della porta NAND sulle uscite del terzo e quarto flip flop

Contatore in modulo qualunque



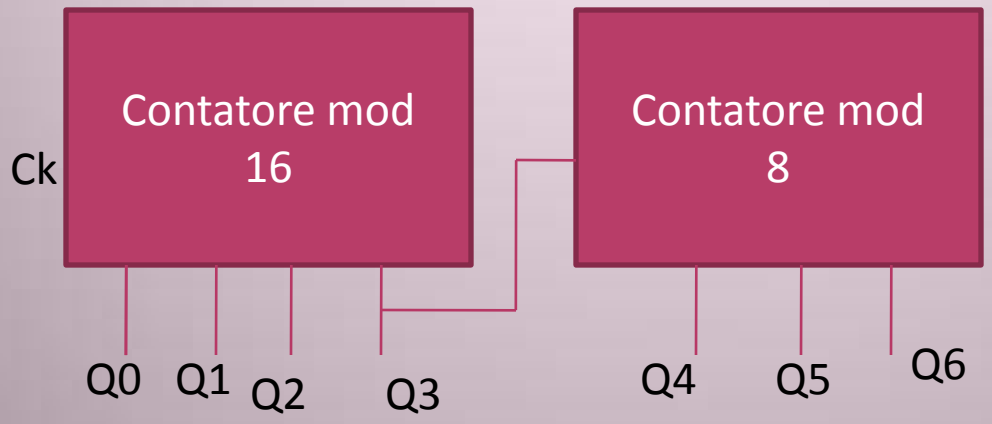
Contatore asincrono con modulo da 1 a n a piacimento

- Supponiamo che sia da 1 a 15 a piacimento



Se vogliamo contare fino a n, dove n è un numero da 1 a 15, basta inserire il numero n in binario da 1 a 15 su I3I2I1I0

Contatori in cascata



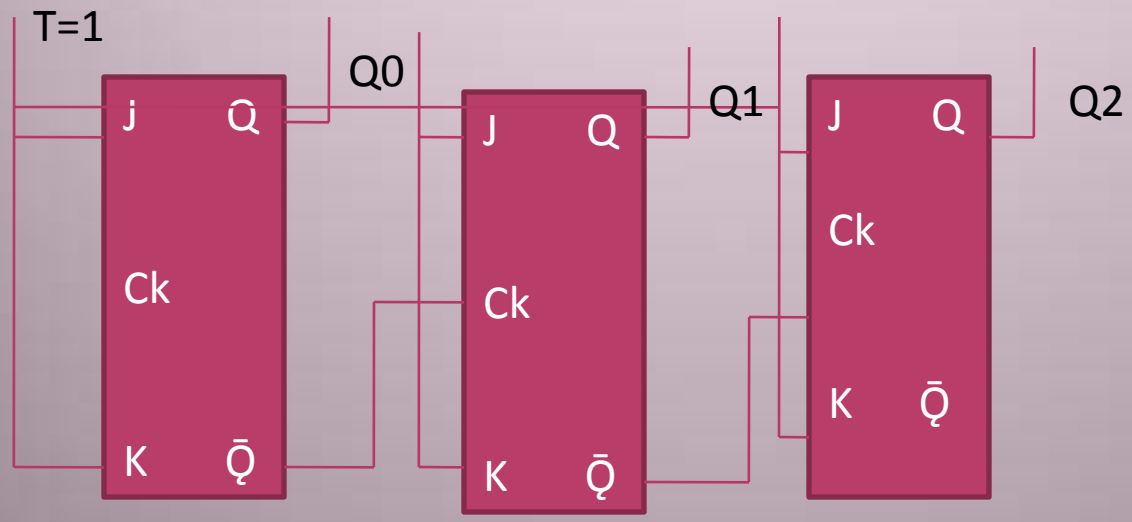
Contatore
modulo 16X8

Problema: progetta un contatore modulo 100

Contatore a decremento

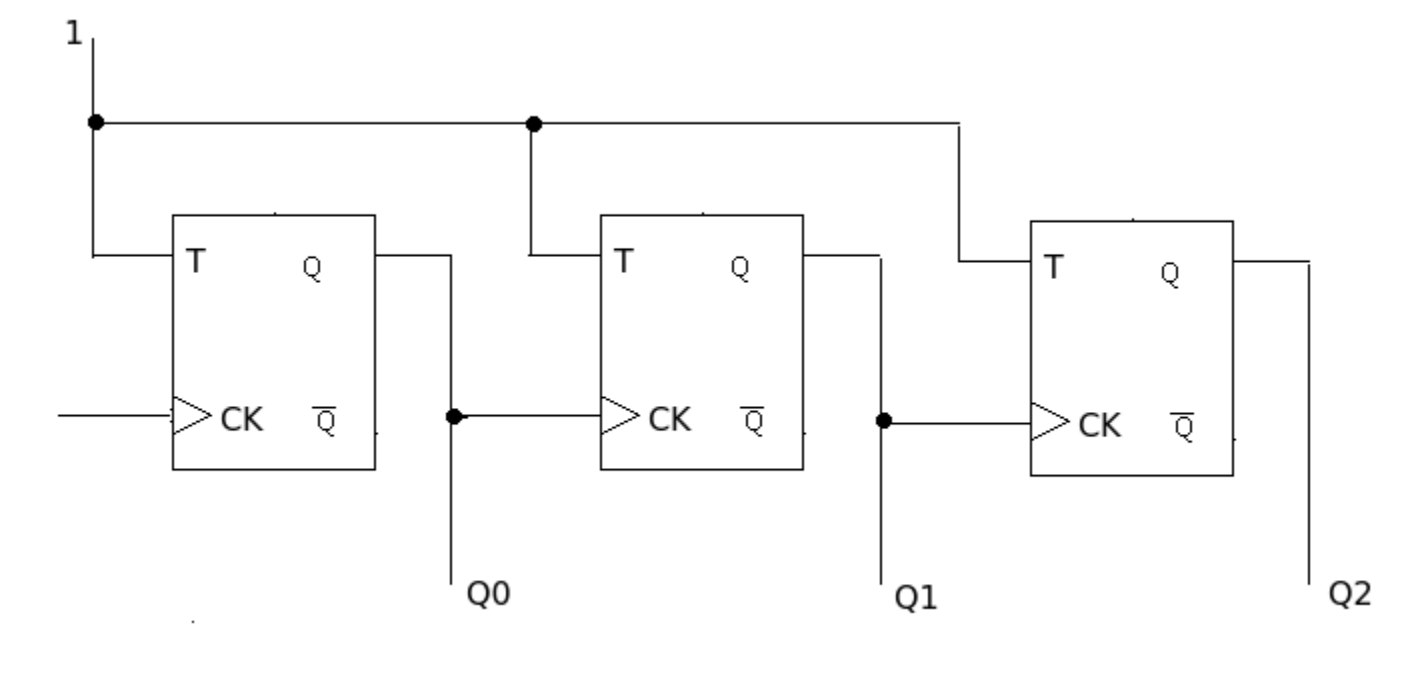
Vengono di seguito riportati vari modi per realizzare un contatore a decremento

L'uscita negata di un Ti viene posta sul clock di Ti+1



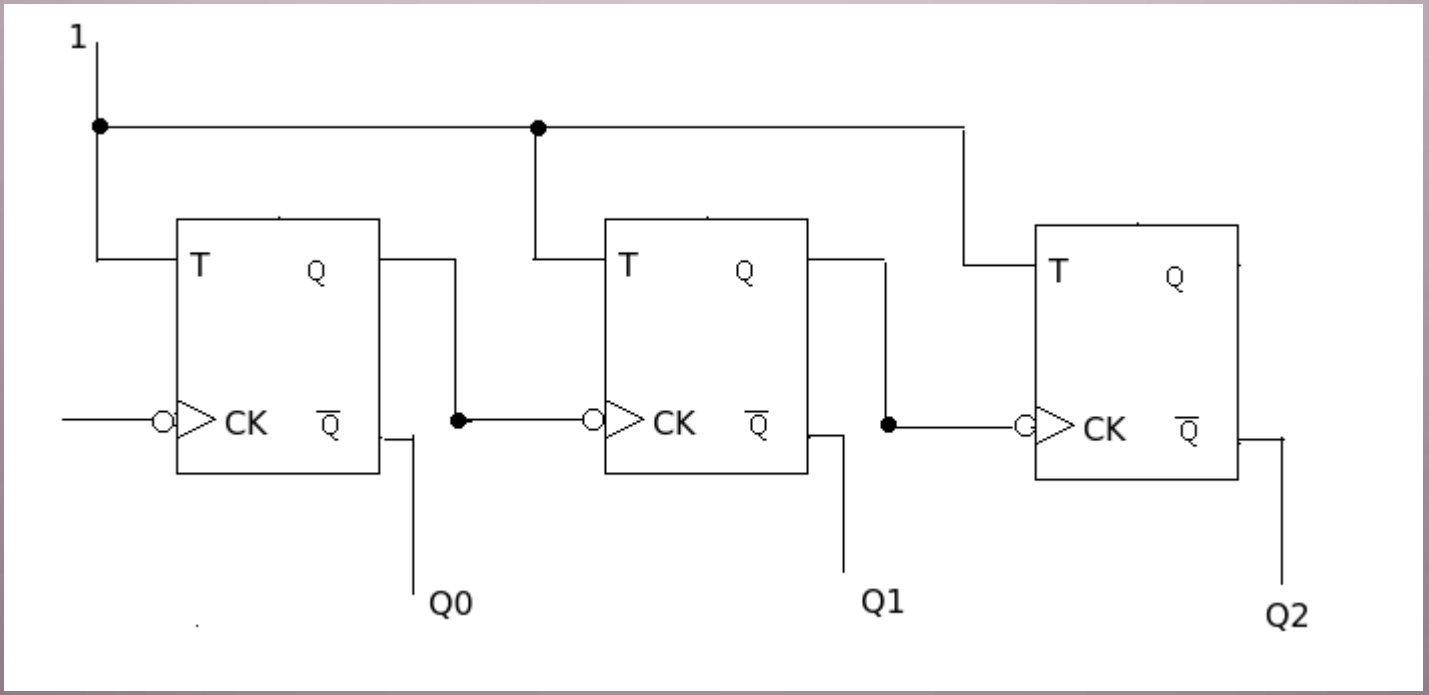
Contatore a decremento

In questo caso l'uscita commuta sul fronte di salita

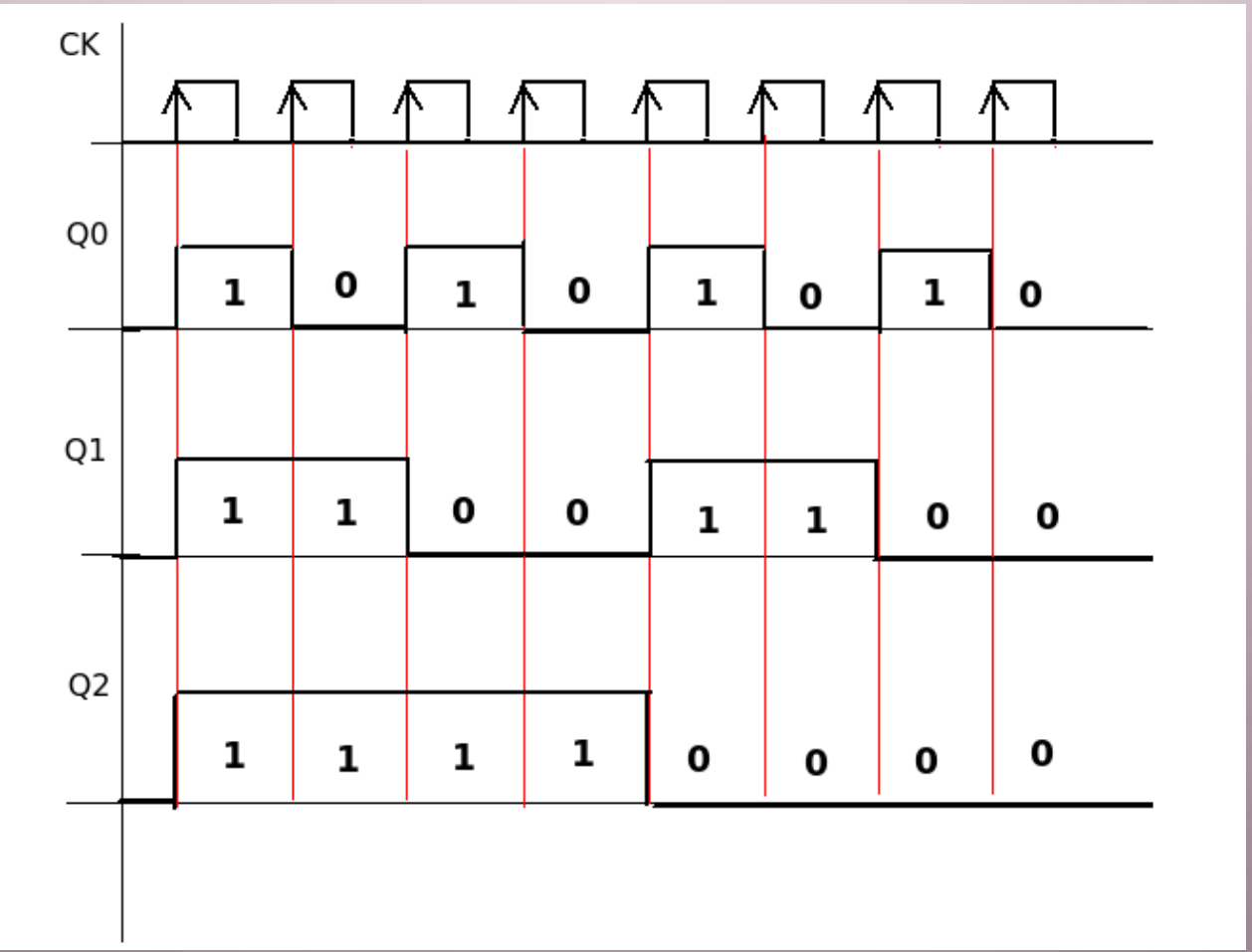


Contatore a decremento

L’uscita viene presa su Q negato



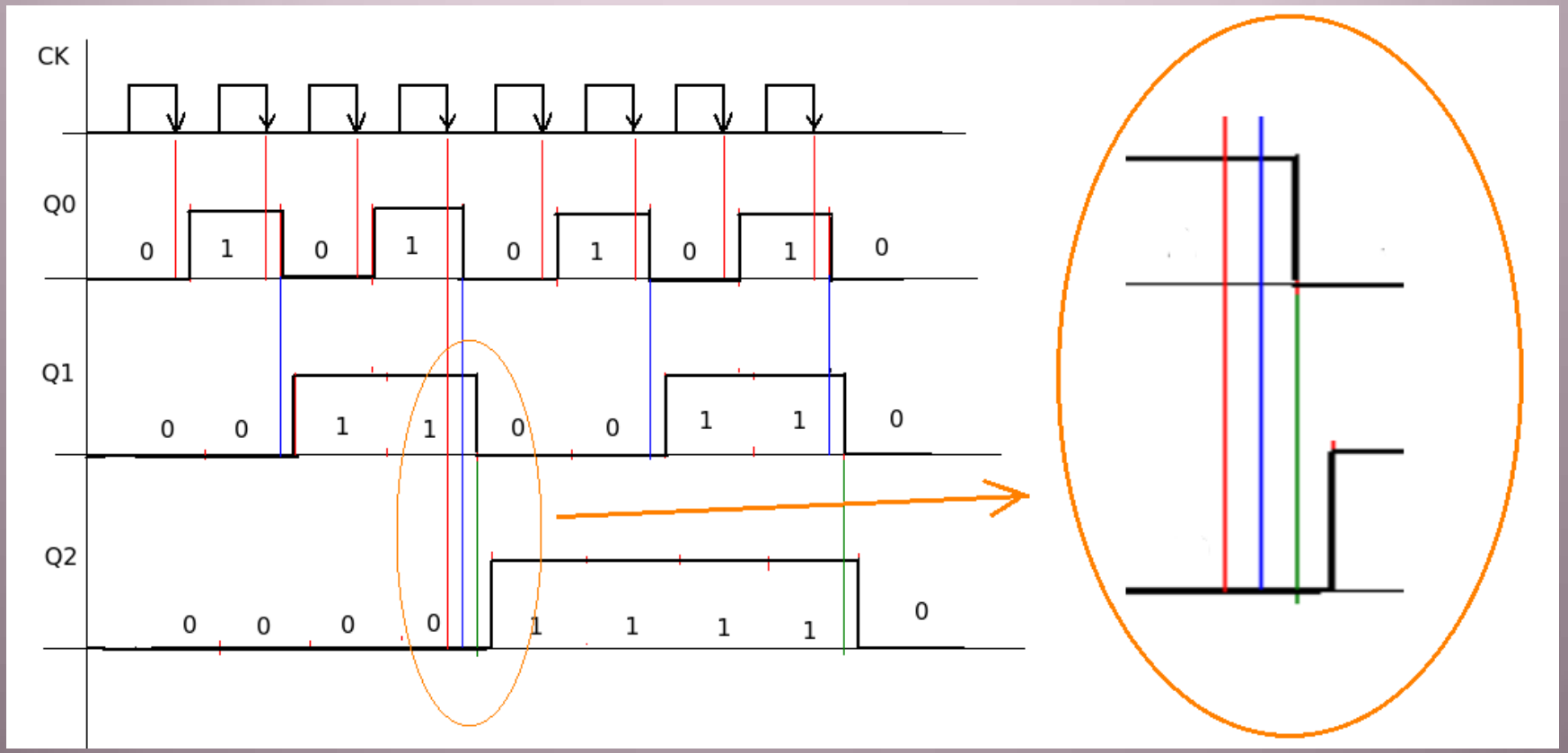
Temporizzazione di un contatore a decremento modulo 8



Limiti dei contatori asincroni

- Nel grafico successivo viene riportato l'effetto del ritardo di propagazione di un segnale attraverso i flip flop. Si noti l'effetto cumulativo del ritardo di propagazione: l'ultimo flip flop commuta in generale dopo un tempo pari a n x T_p, dove n è il numero di bit e T_p è il ritardo di propagazione di un singolo flip flop.
- Se il numero dei flip flop collegati in cascata aumenta tale tempo di ritardo cresce di conseguenza. Ciò può creare dei problemi, in particolare quando la frequenza f del segnale di clock è elevata, ovvero quando il suo periodo T=1/f è piccolo. Infatti potrebbe accadere che il primo ff della serie commuti prima che l'ultimo abbia fatto in tempo a sua volta a commutare. Il problema si verifica se il periodo del clock è più breve della somma dei ritardi e cioè se: $T_{CLK} < n \times T_p$
- In tali condizioni il funzionamento del contatore asincrono non è più garantito, in quanto le combinazioni binarie presenti in uscita potrebbero non essere più corrette.

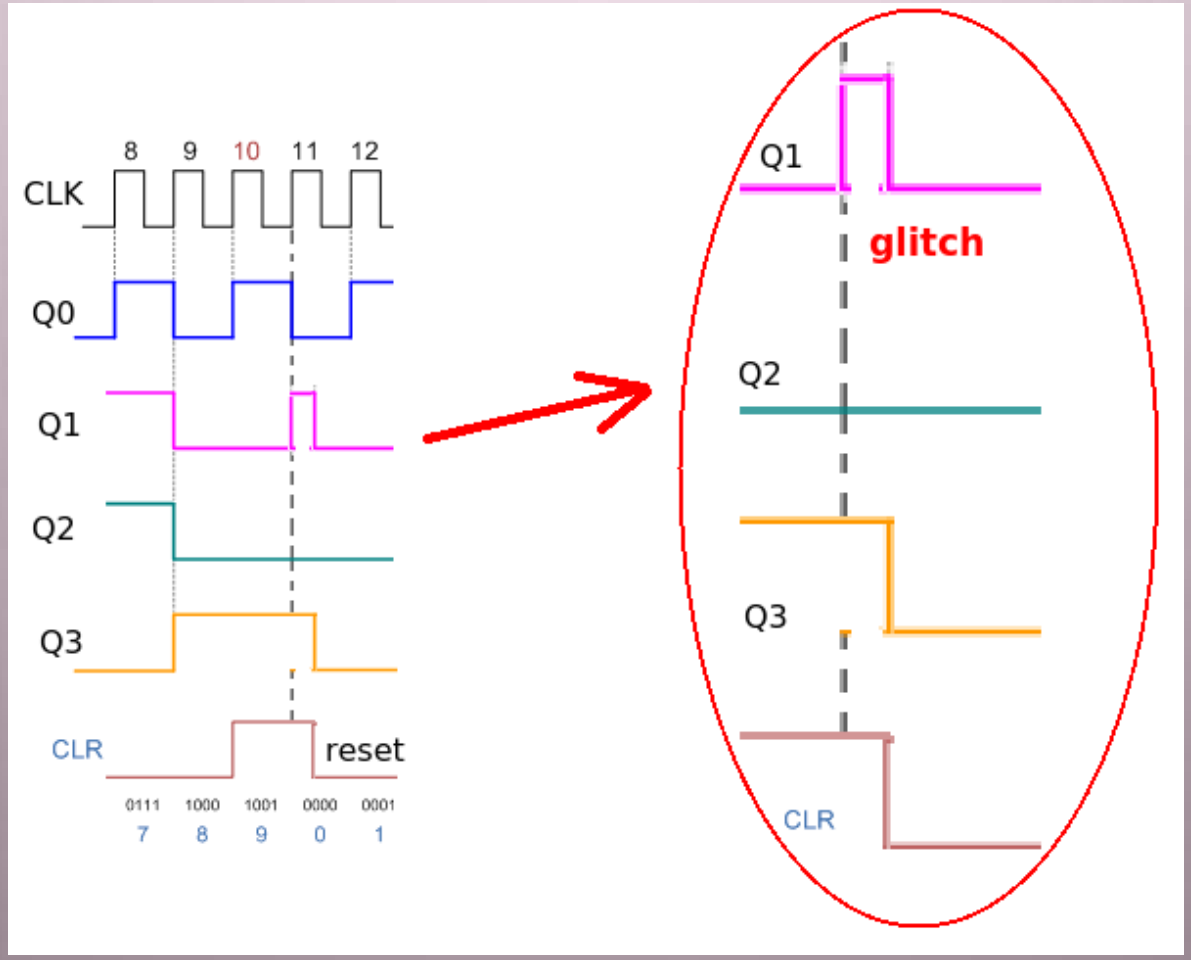
Limiti dei contatori asincroni



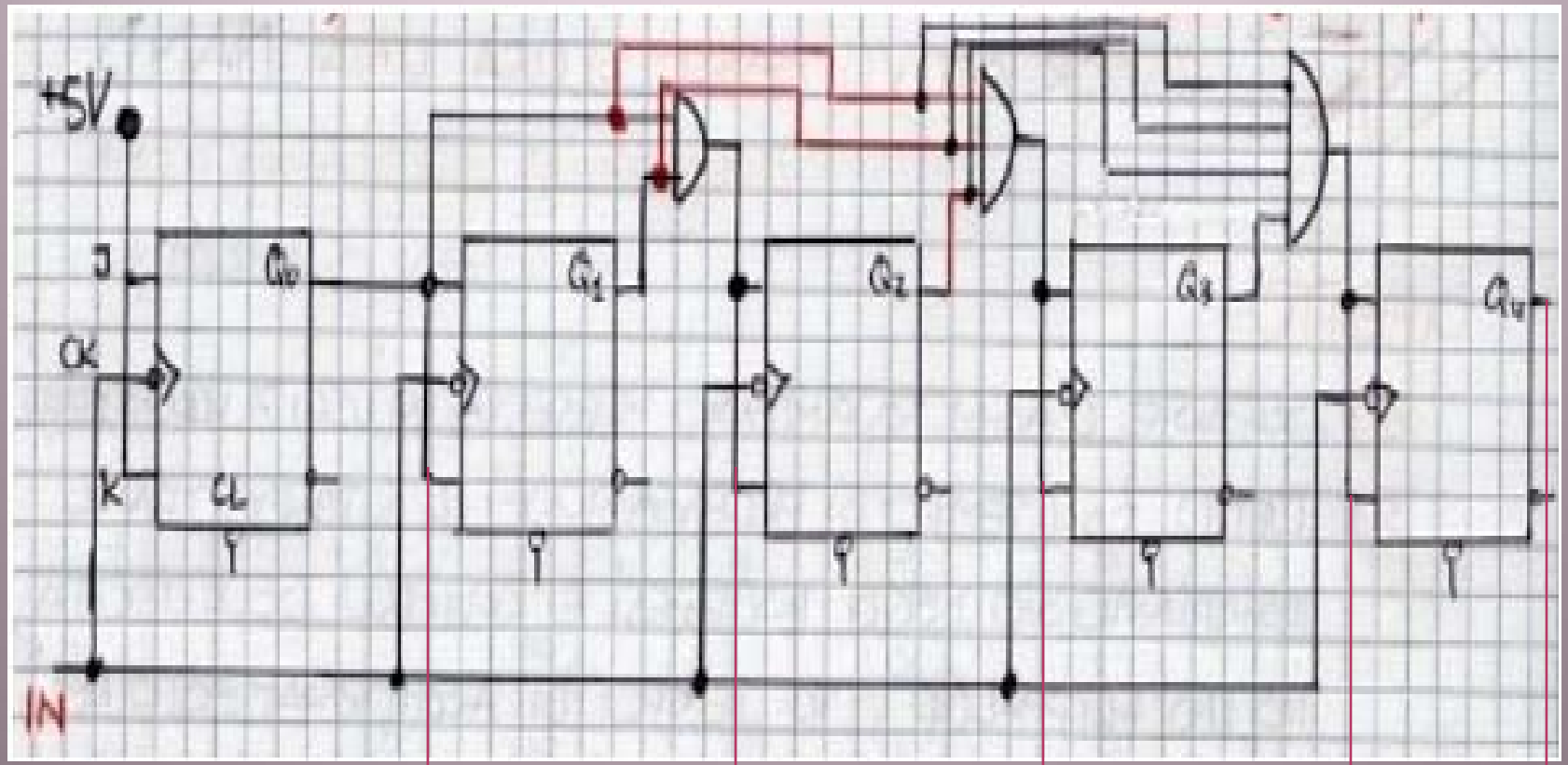
Glitch

- Nei contatori asincroni anche l'impulso di clear non ha durata zero. La sua durata effettiva dipende dai ritardi di propagazione dei flip flop e della porta NAND ed è comunque molto piccola. Nel grafico successivo si osservi in particolare il piccolo impulso spurio (cioè indesiderato) presente sull'uscita Q1. Tale breve impuslo (detto **glitch**) non dovrebbe in teoria essere presente sulle uscite del contatore e corrisponde alla combinazione di reset 1010.
- La presenza di *glitch* in un circuito elettronico è in alcuni casi trascurabile, mentre in altri può dare luogo a malfunzionamenti. Se per esempio il nostro contatore BCD viene collegato in uscita a un display a 7 segmenti per visualizzare la sequenza di conteggio, a causa dei ritardi e delle inerzie intrinseche al display, la breve apparizione della combinazione 1010 non produce nessuna visualizzazione e dunque non crea problemi. Viceversa ci potrebbero essere problemi nel caso in cui il contatore fosse collegato a un dispositivo "veloce", in grado di leggere la sequenza spuria prima che essa scompaia dalle uscite.

Glitch


8
9
10
11
12
CLK
Q0
Q1
Q2
Q3
CLR
reset
0111
1000
1001
0000
0001
7
8
9
0
1
Q1
glitch
Q2
Q3
CLR

Contatore sincrono



Contatore sincrono

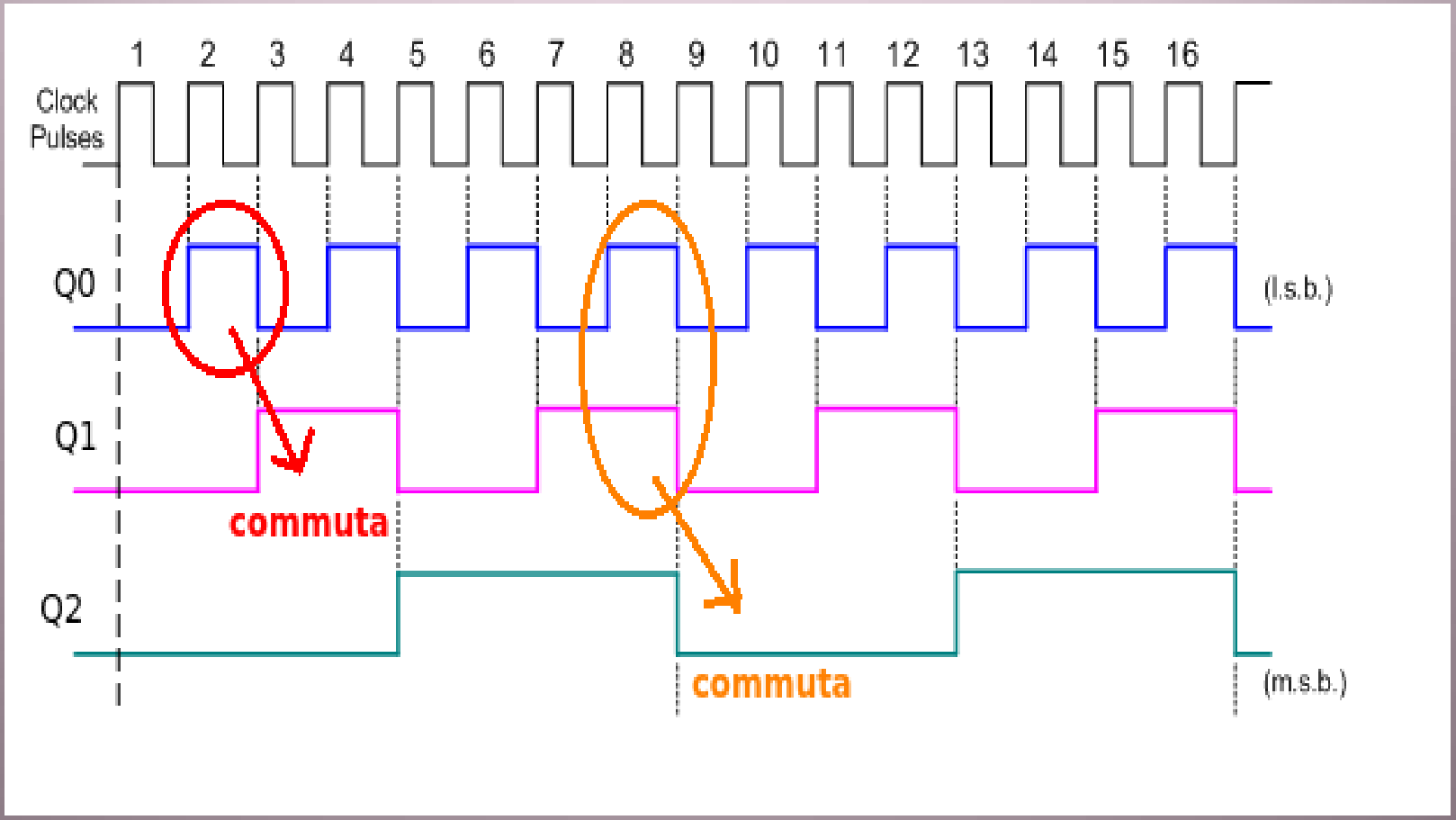
- Il contatore è sincrono se il segnale di clock viene dallo stesso impulso
- L'uscita del flip flop T0 commuta sul fronte di discesa del clock;
- L'uscita del flip flop T0 finisce sull'ingresso di T1 che non commuta per il ritardo di propagazione del segnale attraverso T0;
- Affinchè il contatore funzioni bene, l'uscita di ogni flip flop T(i) è ingresso di una porta And insieme all'uscita del flip flop T(i+1);
- L'uscita di ogni AND pilota il flip flop T(i+2)
- Questo sistema sfrutta sempre il ritardo di propagazione del segnale attraverso la porta

Contatori sincroni

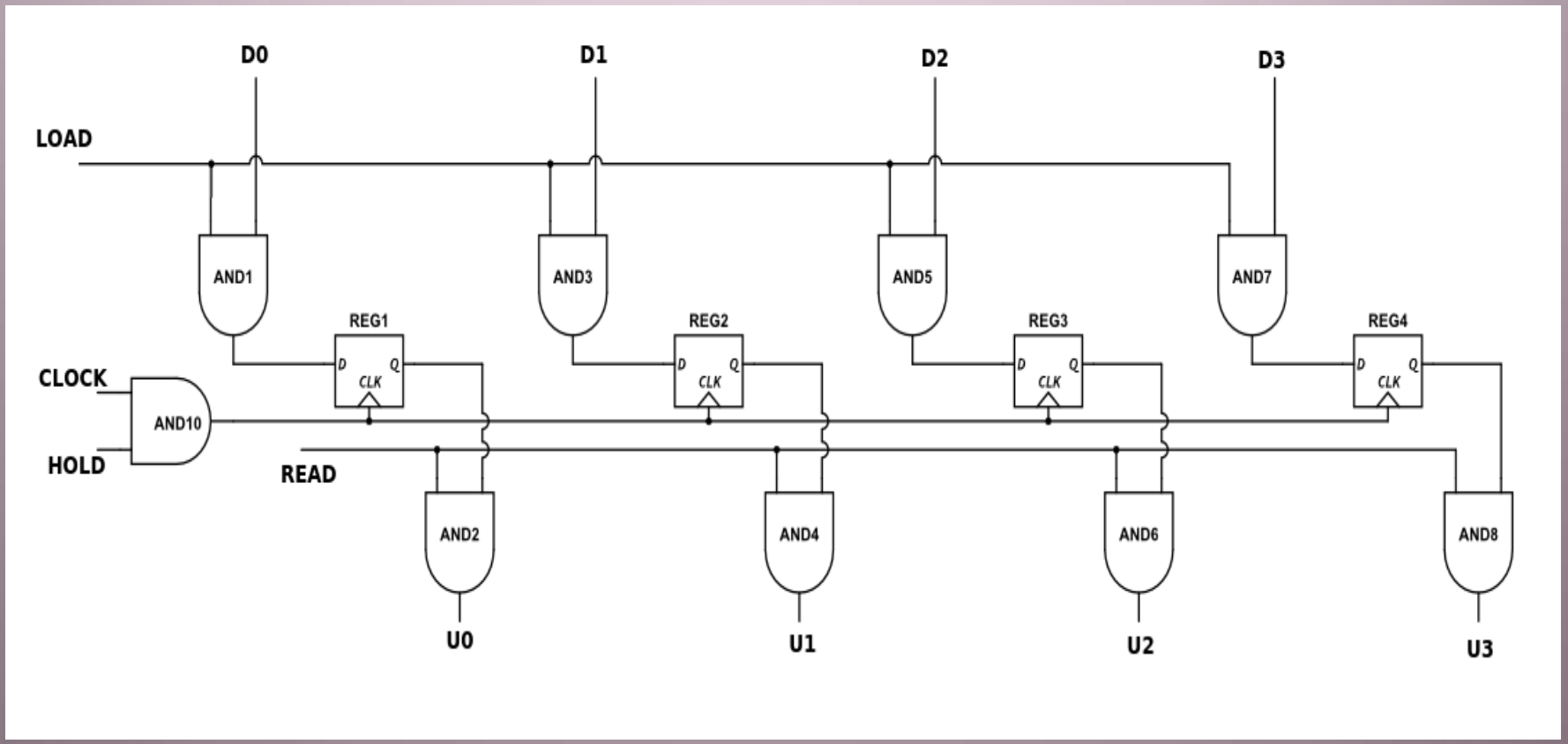
Riassumendo

il primo flip flop T commuta sempre, ad ogni fronte di salita del clock. Il secondo flip flop commuta solo quando l'uscita del primo è a 1, mentre il terzo commuta quando le uscite dei primi due sono a 1.

Contatori sincroni



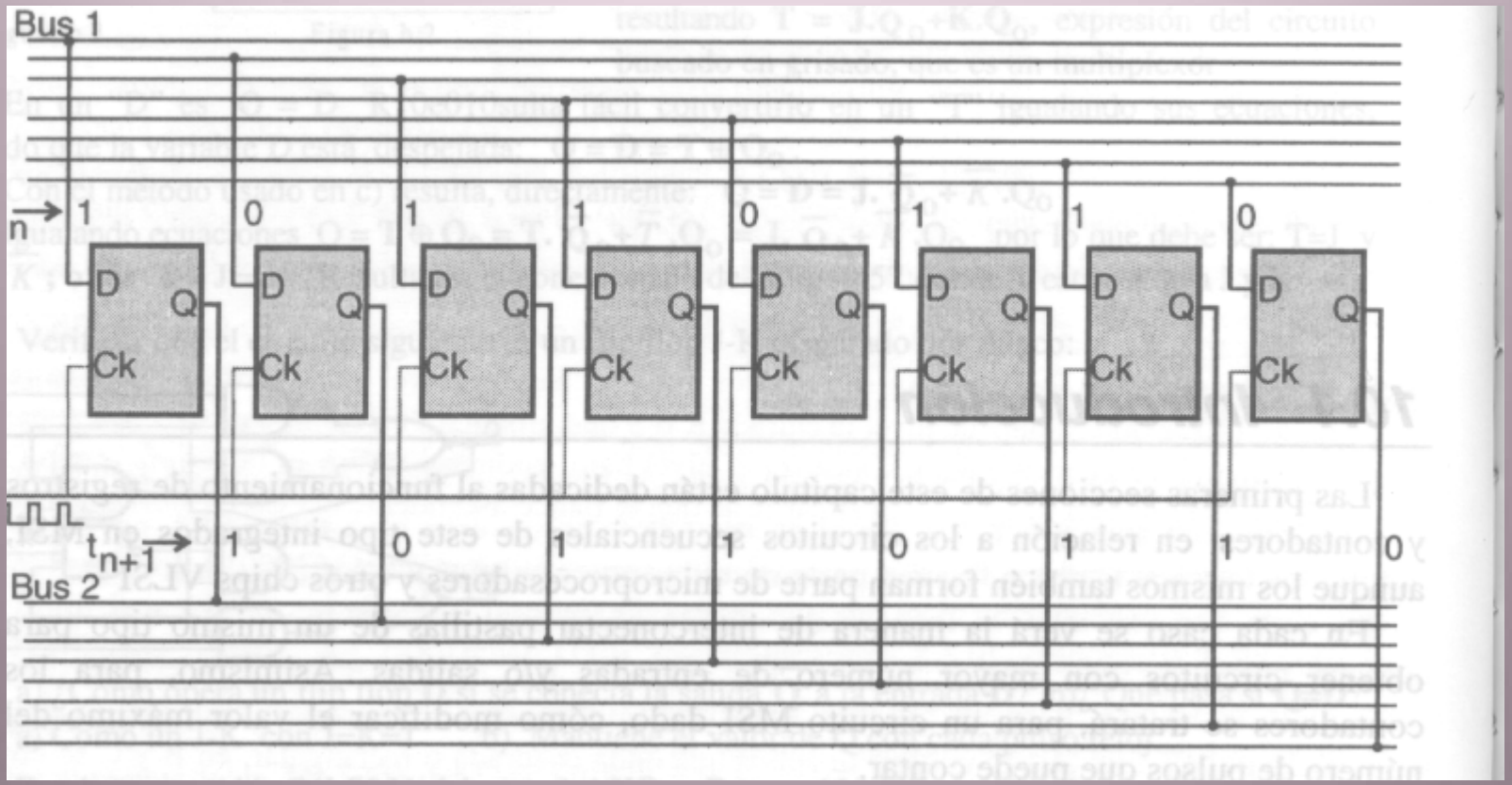
Registri a scorrimento
PIPO (Parallel Input Parallel Output)



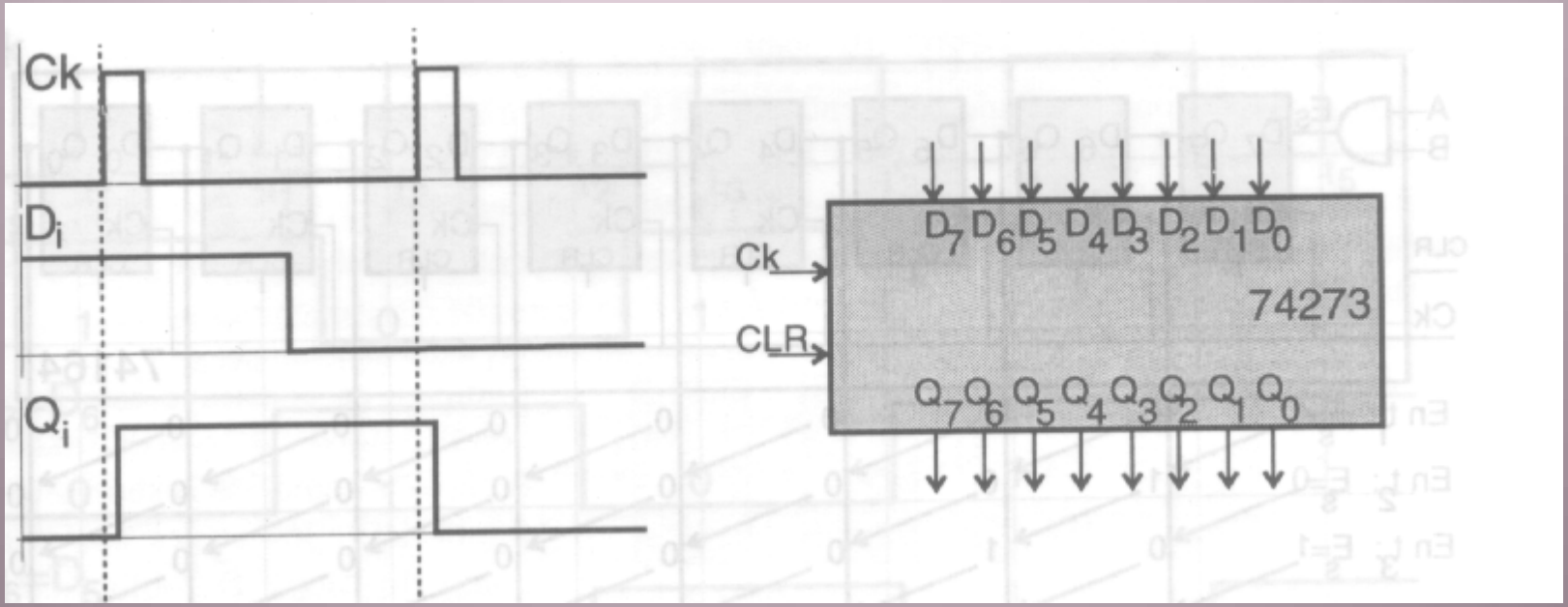
Registro PIPO

- **SCRITTURA**
  Per scrivere un dato binario (4 bit) nel registro, occorre presentare il dato sui piedini D0,..., D3 e porre LOAD=H e HOLD=H. All'arrivo del fronte di salita del CLOCK il dato viene memorizzato nei 4 ff D.
- **MANTENIMENTO**
  Per mantenere il dato in memoria bisogna porre HOLD=L. In questo modo i successivi fronti del CLOCK non producono ulteriori caricamenti, in quando la porta AND10 blocca il segnale di CLOCK.
- **LETTURA**
  Per avere il dato sui piedini di uscita U0,..., U3 bisogna porre READ=H.Il diagramma temporale nella figura seguente mostra il funzionamento del registro:

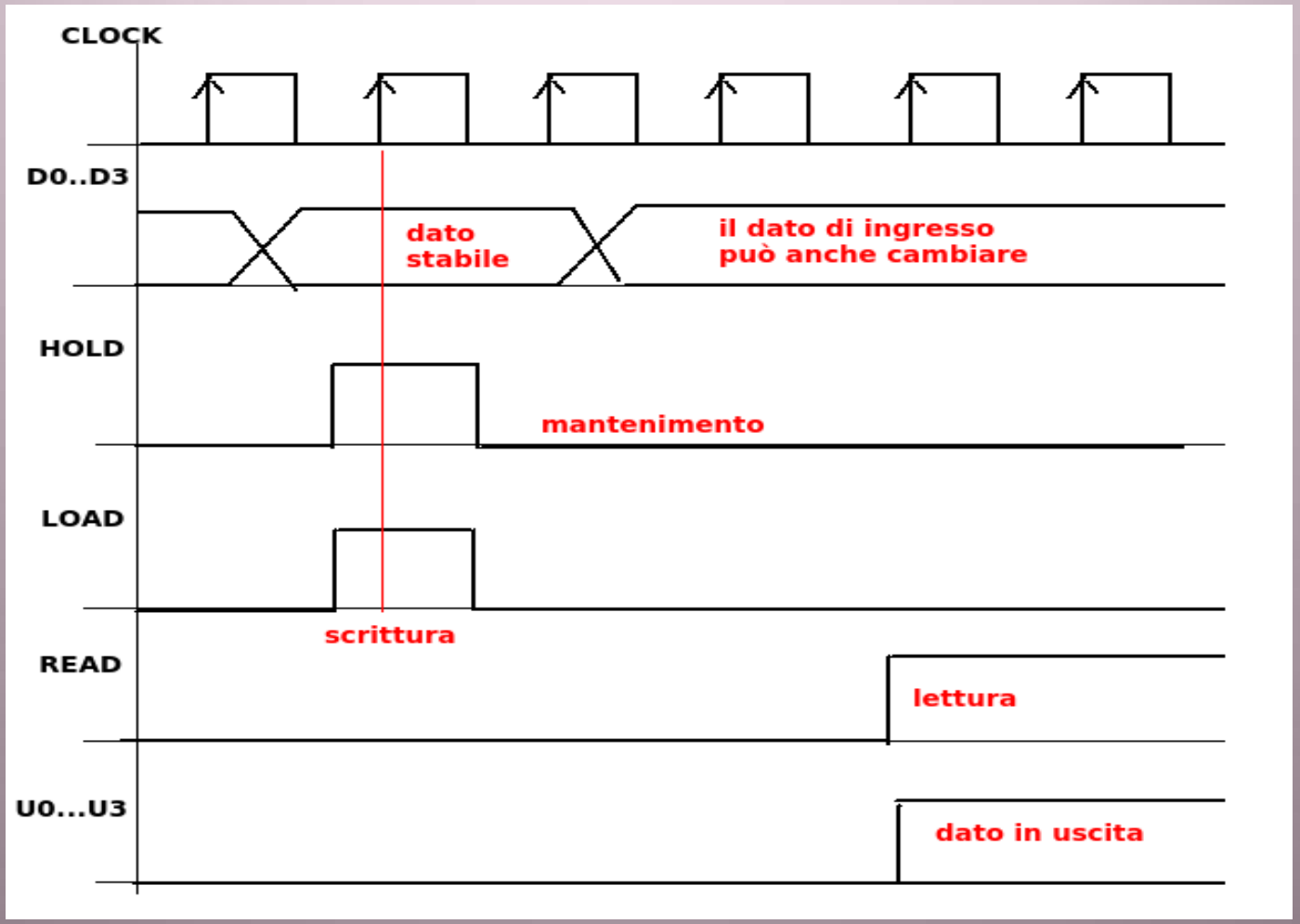
Registro PIPO e relativi bus



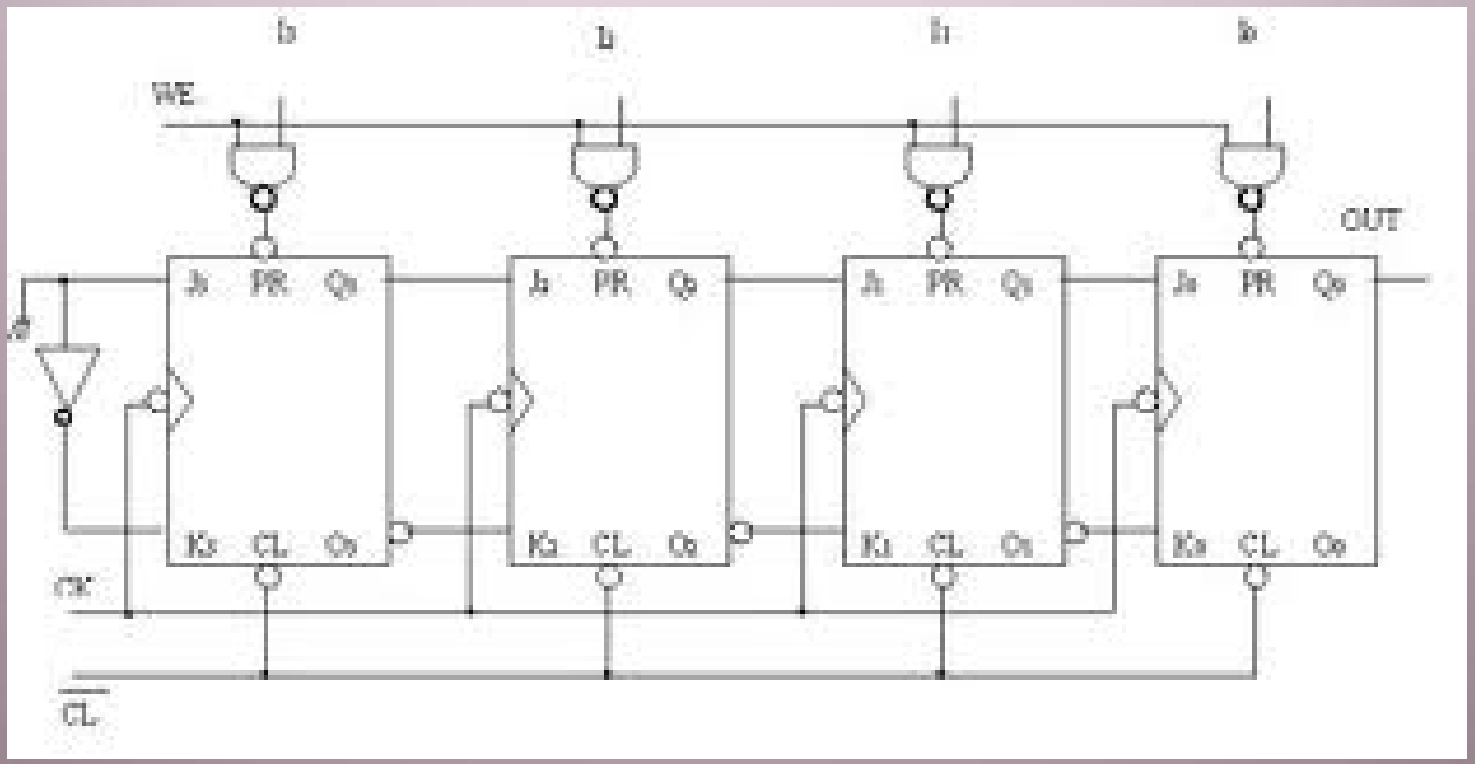
Integrato di un PIPO



Temporizzazione

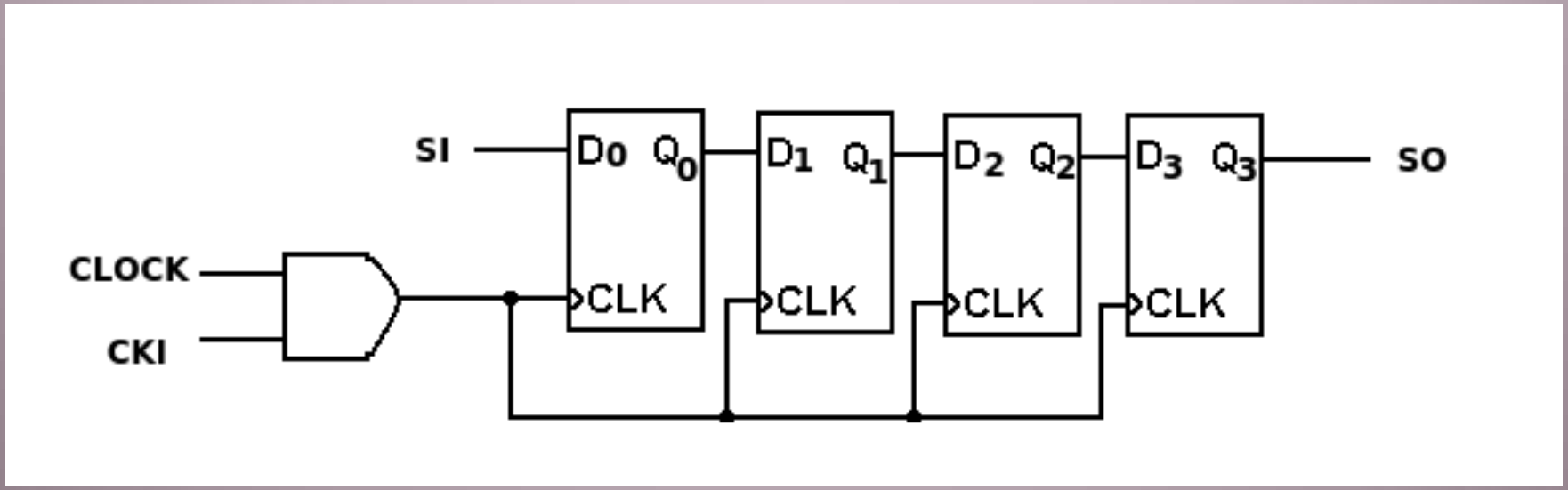


Registri a scorrimento
PISO (Parallel Input Serial Output)

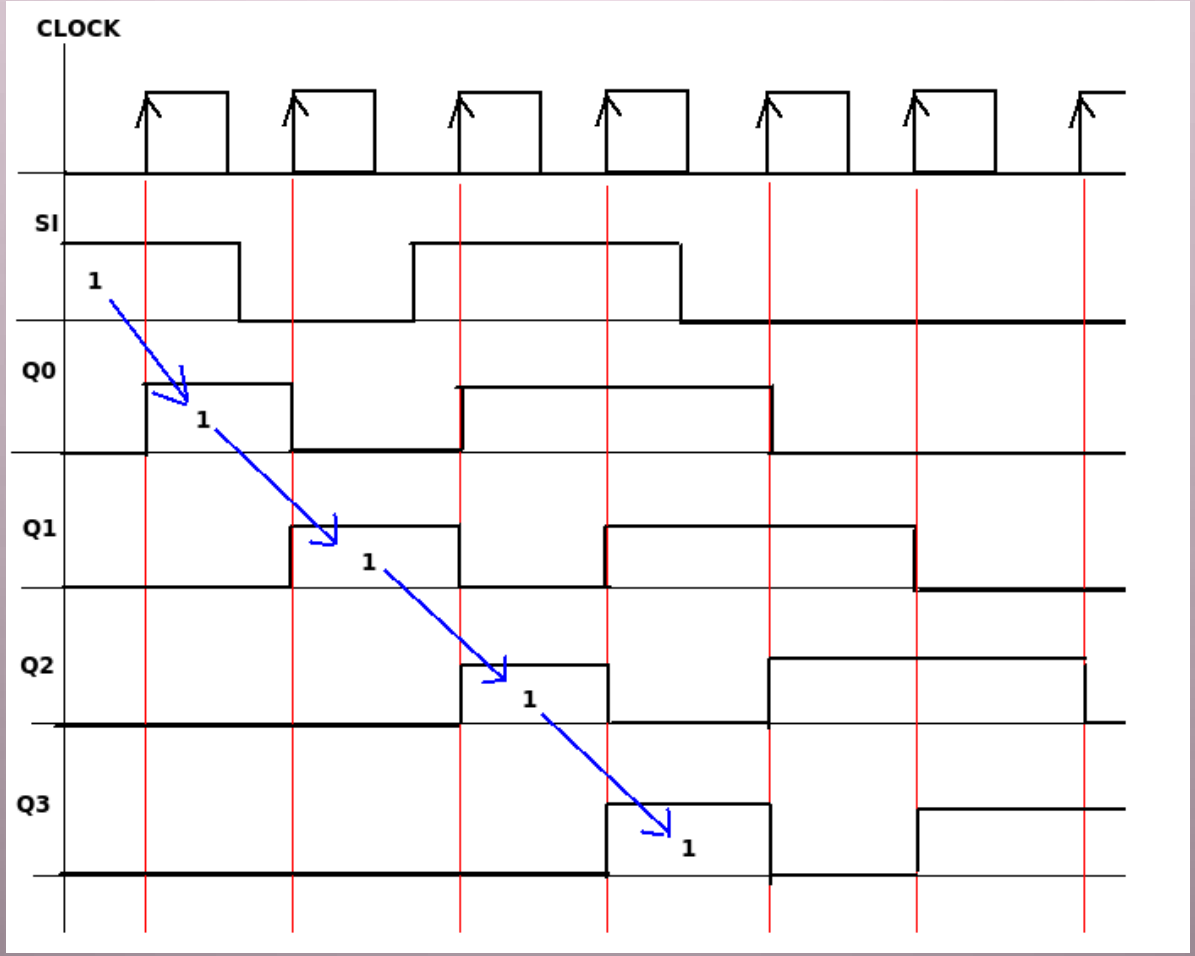


WE= Write Enable

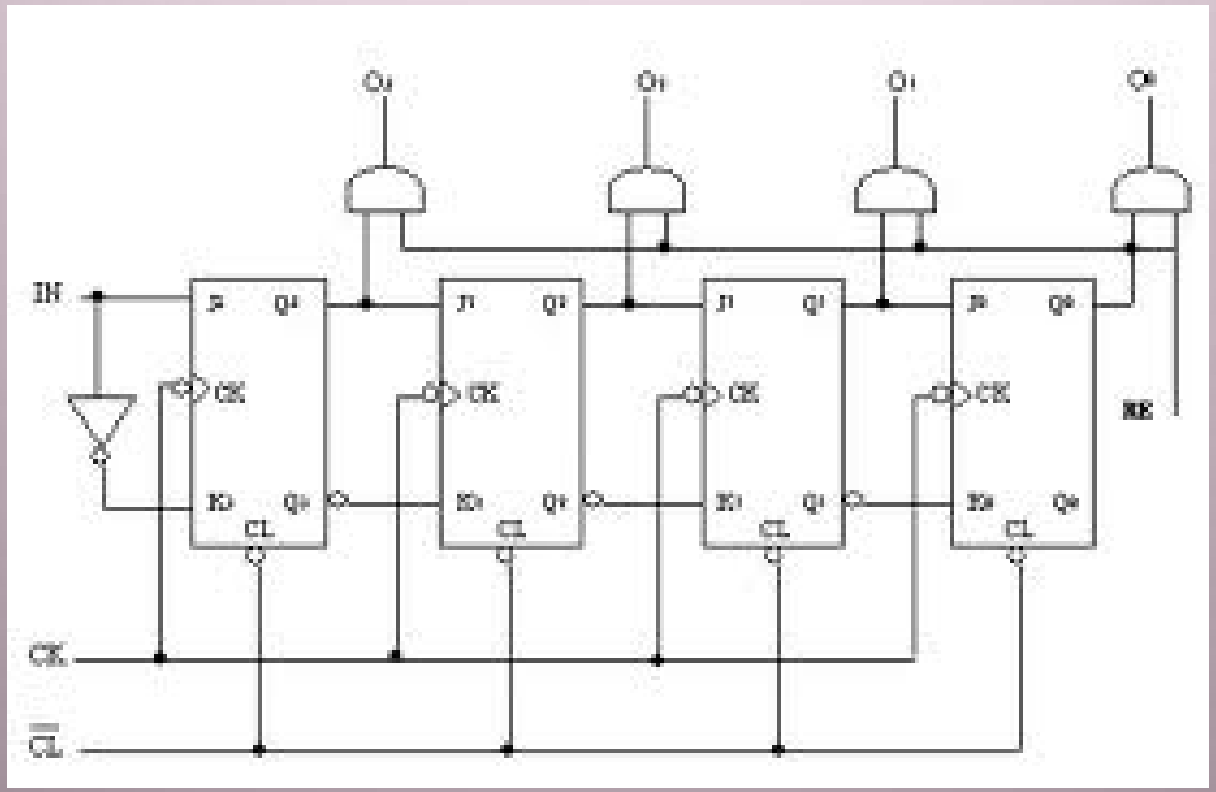
Registri a scorrimento SISO (Serial Input Serial Output)



Registri SISO



Registri a scorrimento
SIPO (Serial Input Parallel Output)

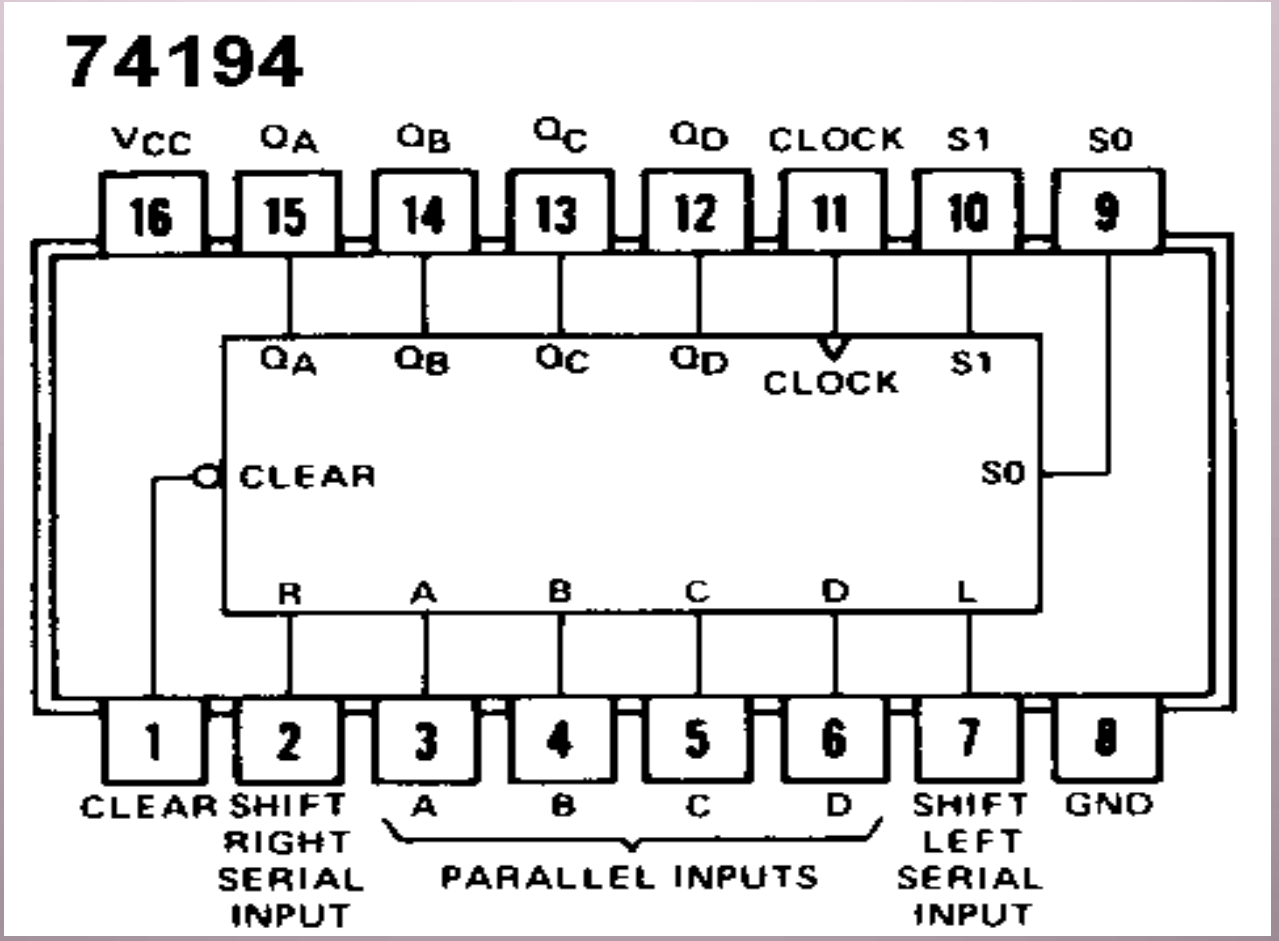


RE = read enable

Applicazioni

- Un semplice vantaggio dei registri seriali rispetto ai registri paralleli è il risparmio nel numero di piedini necessario.
- Un' applicazione dei registri seriali consiste nella realizzazione di *linee di ritardo*, cioè di circuiti in grado di ritardare la trasmissione di un dato segnale digitale. Usando un SISO a n bit infatti il segnale di ingresso viene trasferito in uscita solo dopo n fronti di clock, realizzando in tale modo un ritardo.
- Una delle applicazioni più importanti dei registri a scorrimento è la conversione da parallelo a seriale e viceversa. Si consideri a titolo di semplice esempio il collegamento fra un computer e una stampante. Ogni carattere da stampare è rappresentato all'interno del computer per mezzo di un byte (8 bit). Supponendo di voler stampare una pagina di testo, il computer dovrà quindi inviare alla stampante una sequenza di byte che rappresentano tutti i caratteri da stampare.

74194: un registro universale



74194: un registro universale

- Gli ingressi A, B, C e D sono gli ingressi paralleli. SHIFT RIGHT SERIAL INPUT, come suggerisce il nome, è l'ingresso seriale per lo scorrimento verso destra. Analogamente SHIFT LEFT SERIAL INPUT è l'ingresso seriale per lo scorrimento verso sinistra. QA, QB, QC e QD sono le uscite parallele. QA funge anche da uscita seriale per lo scorrimento verso sinistra e QD è anche l'uscita seriale per lo scorrimento verso destra. CLEAR è un ingresso asincrono che permette di resettare tutti i bit del registro.
- Gli ingressi SO e S1 sono due ingressi di selezione che consentono di selezionare il funzionamento dell'integrato, in base alla seguente tabella di verità:

S0	S1	Funzione
L	L	Hold
L	H	Shift Left
H	L	Shift Right
H	H	Parallel Load