



UNIVERSITÀ DI PISA

CORSO DI LAUREA IN FISICA

LABORATORIO DI FISICA 3

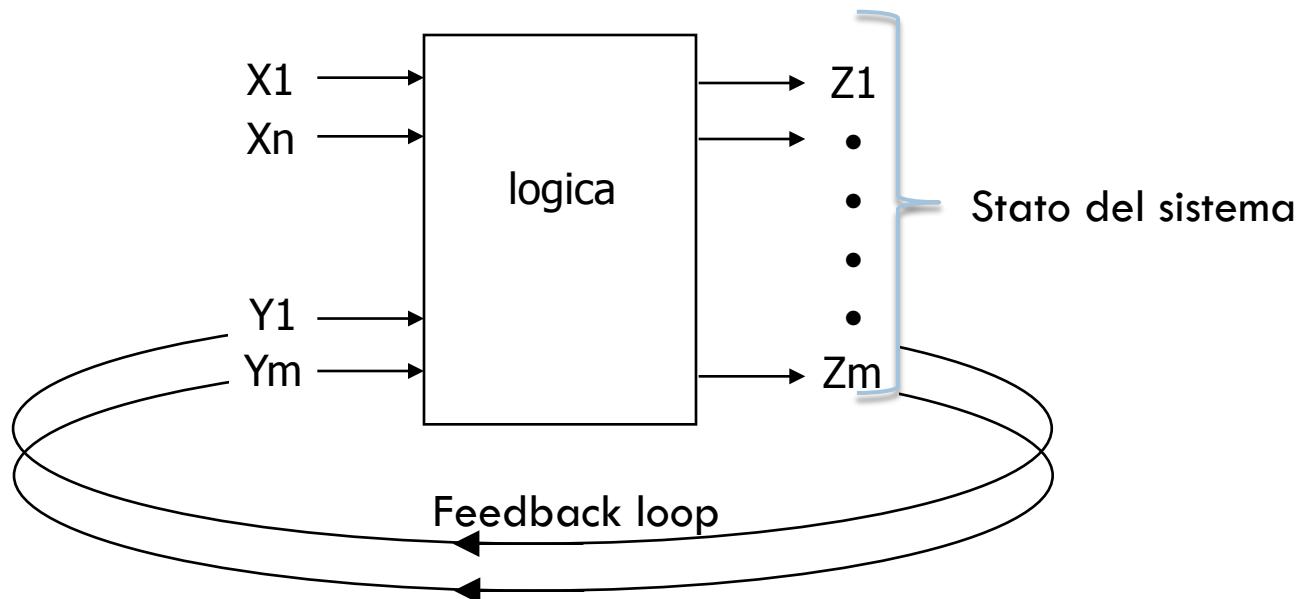
D04 – LOGICA SEQUENZIALE

Prof. F. Forti (con materiale di R. Katz, C. Roda)

Logica sequenziale

2

- Uscita che dipende dalla storia delle entrate
 - Circuiti con memoria: hanno uno stato da cui dipendono le uscite
 - Cioè lo stato del sistema (output) viene collegato in feedback alle entrata (input).



Le entrate vengono continuamente aggiornate dalle uscite → può essere instabile

Equazione caratteristica

3

- Le variazioni su Z si presentano dopo un tempo Δt (ritardo della logica)
 - $Z(t + \Delta t) = f(X(t); Y(t))$
- Considero quale sarà l'uscita a $t + \Delta t$ sapendo gli ingressi e le uscite al tempo t
 - $Z(t + \Delta t) = f(X(t); Z(t))$
- Si chiama equazione caratteristica
- Problema: Δt è piccolo e non ben determinato

Notazione

4

- Per semplificare la scrittura delle slides, utilizzo

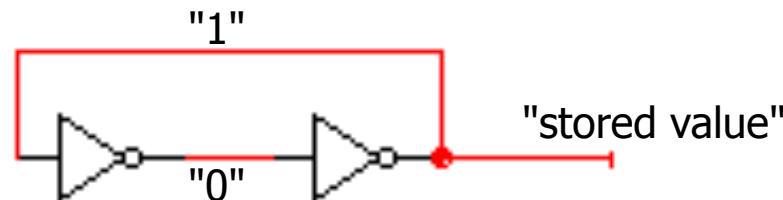
$$X' \equiv \overline{X} \quad A \cdot B \equiv A \cdot B$$

- Per esempio: NOT A \rightarrow A'
- A NAND B = (A.B)'
- A NOR B = (A+B)'
- De Morgan: (A.B)' = A'+B'; (A+B)' = A'.B'

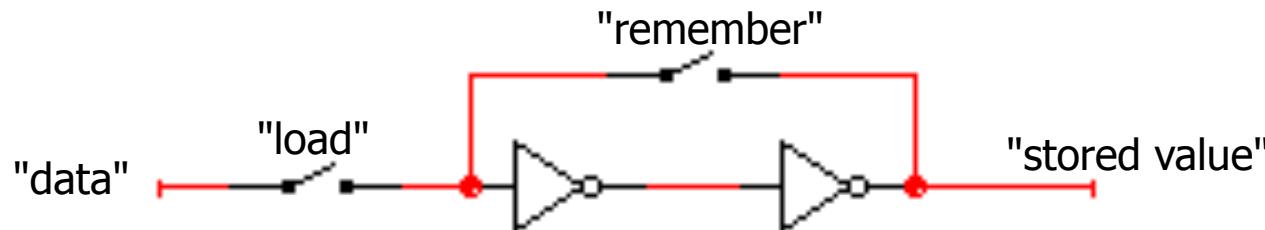
Simplest Circuits with Feedback

5

- Two inverters form a static memory cell
 - Will hold value as long as it has power applied



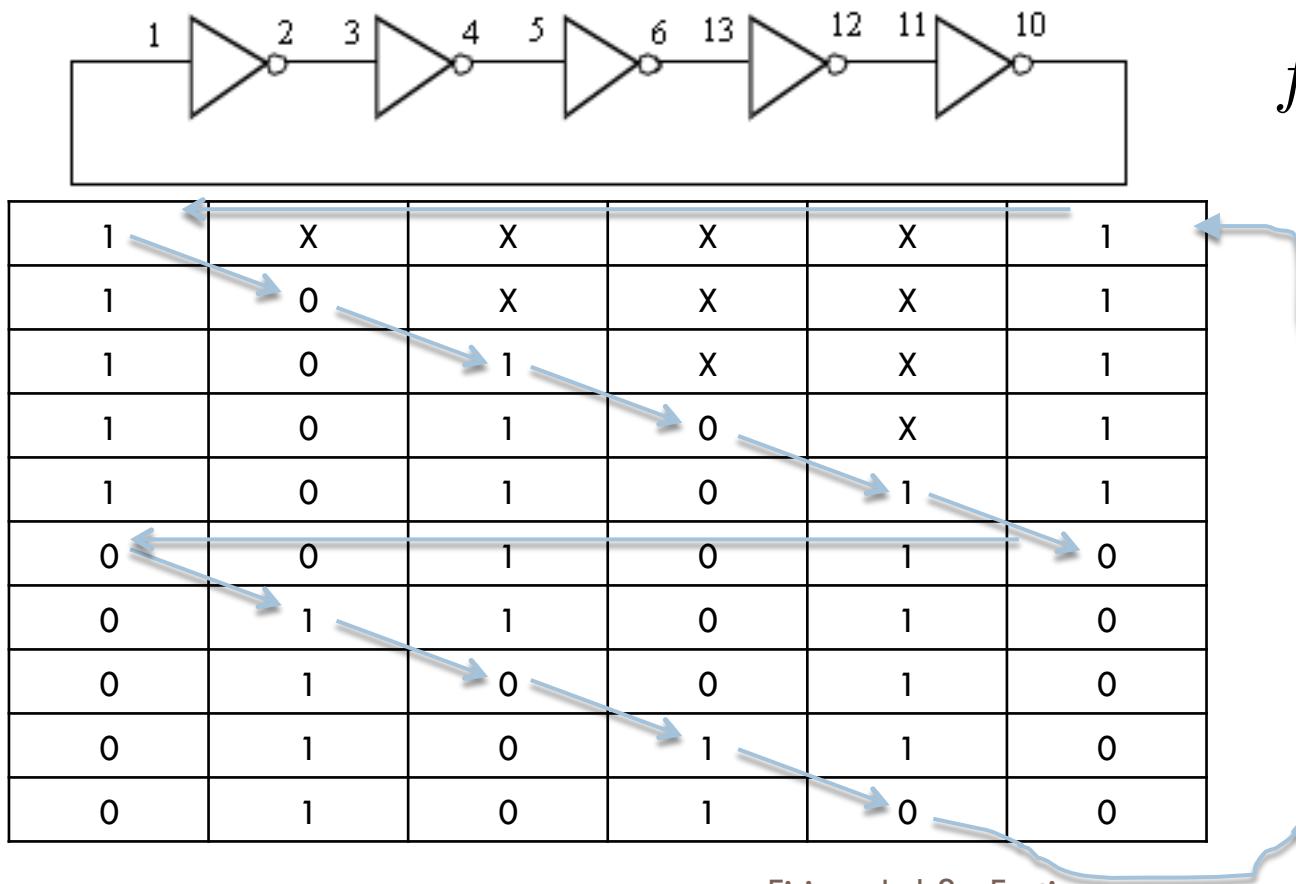
- How to get a new value into the memory cell?
 - Selectively break feedback path
 - Load new value into cell



Ring oscillator

6

□ Feedback con un numero dispari di NOT



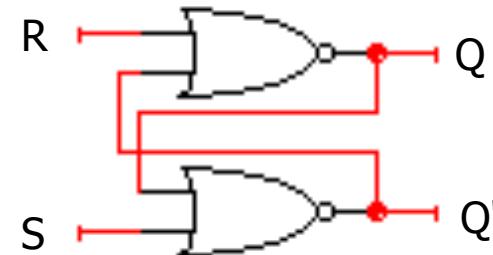
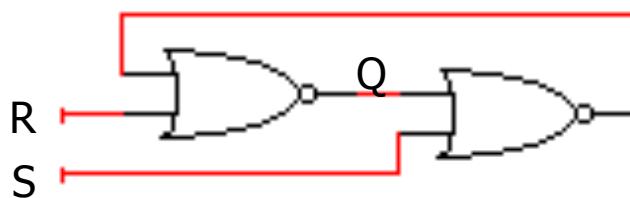
Memory with Cross-coupled Gates

R-S Latch

7

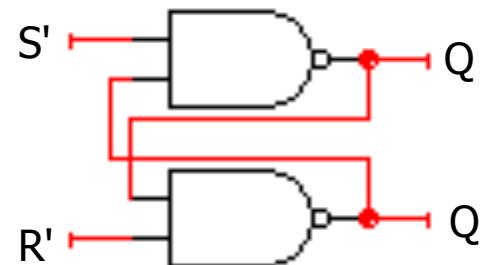
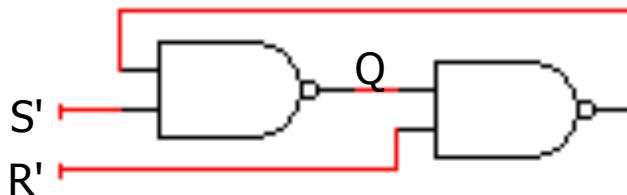
□ Cross-coupled NOR gates

- Similar to inverter pair, with capability to force output to 0 (reset=1) or 1 (set=1)



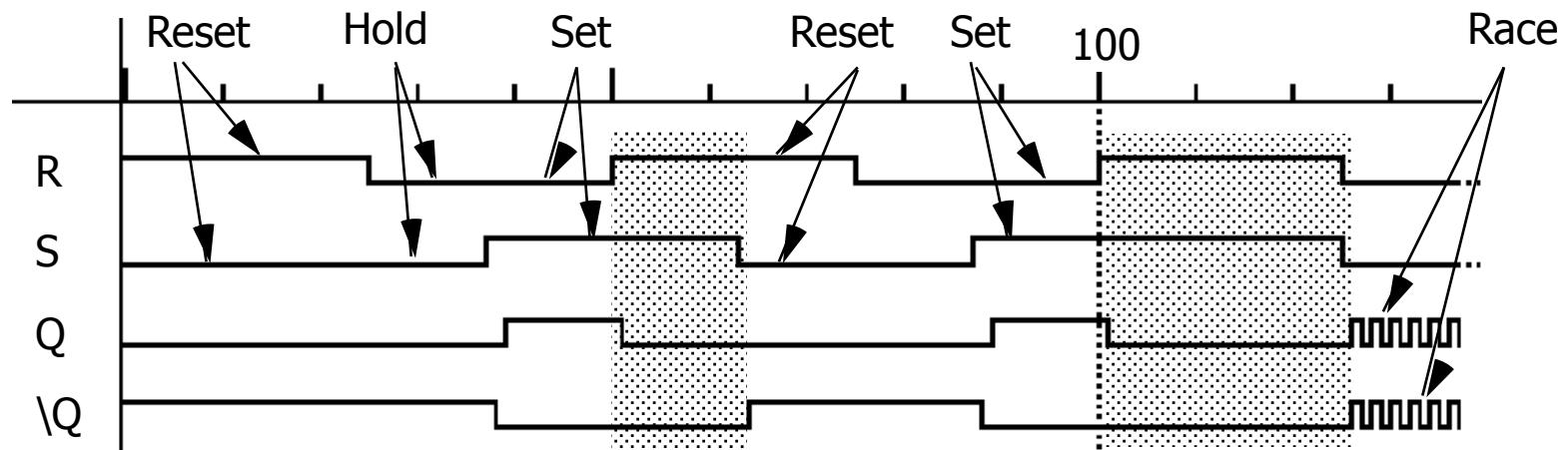
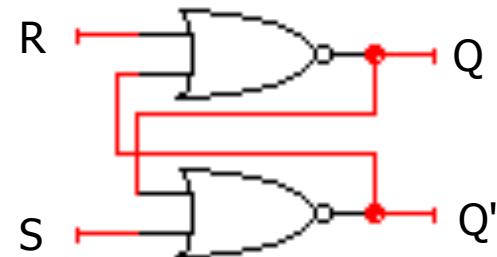
□ Cross-coupled NAND gates

- Similar to inverter pair, with capability to force output to 0 (reset=0) or 1 (set=0)



Timing Behavior

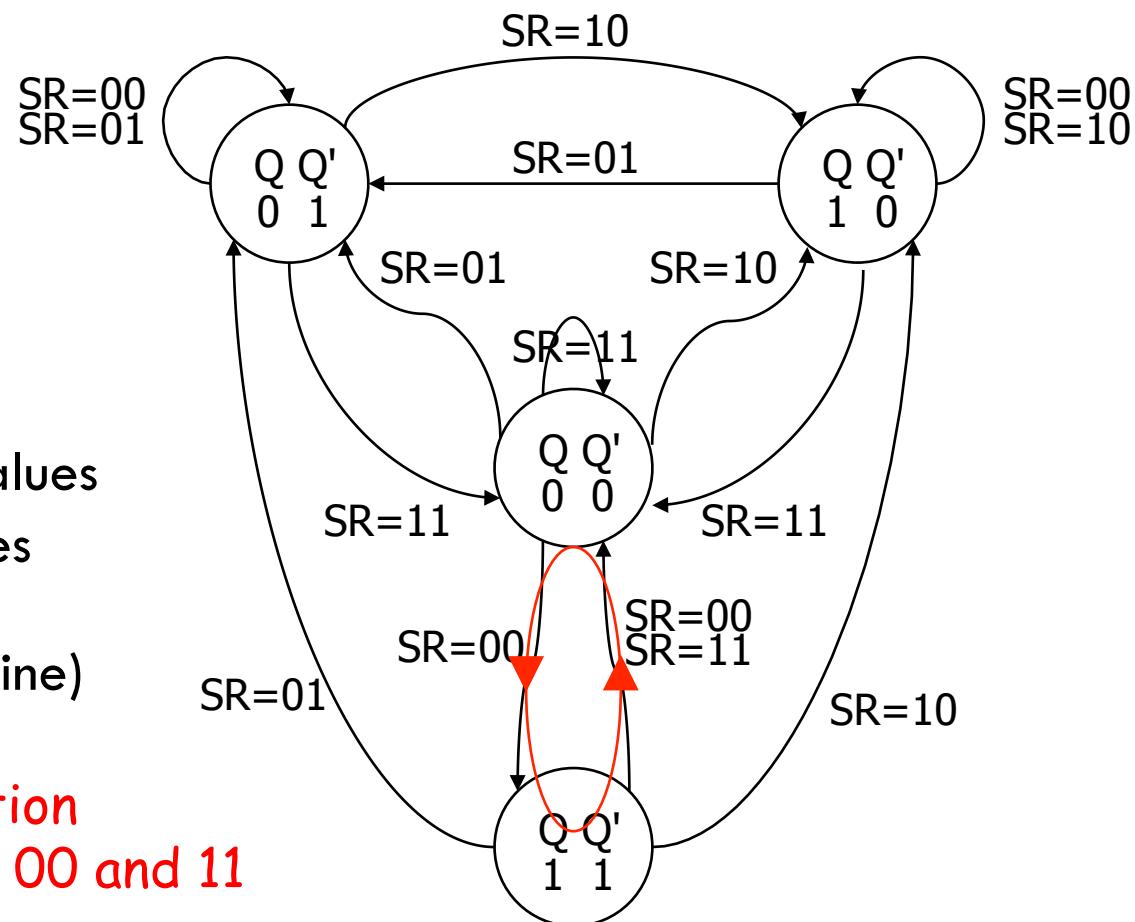
8



Theoretical R-S Latch Behavior

9

S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	unstable

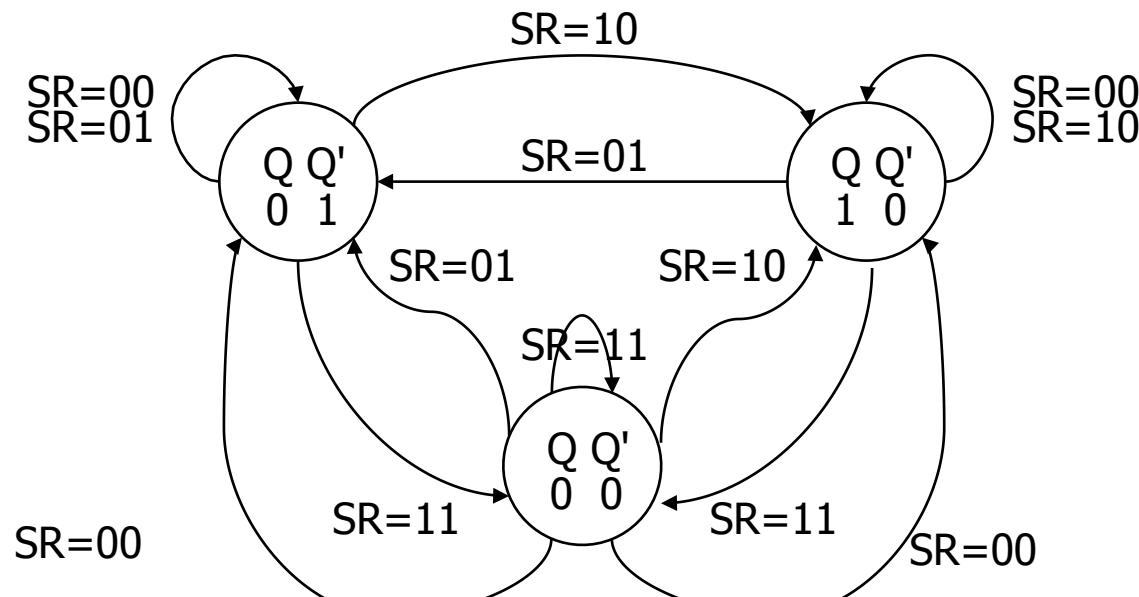


possible oscillation
between states 00 and 11

Observed R-S Latch Behavior

10

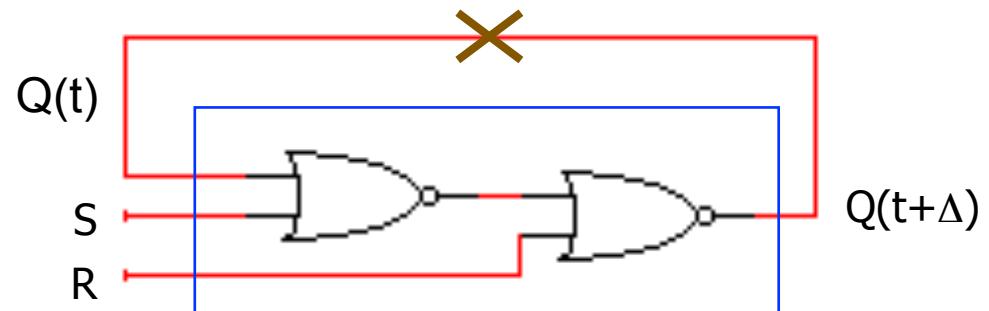
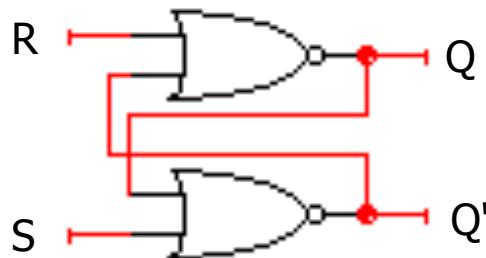
- Very difficult to observe R-S latch in the 1-1 state
 - One of R or S usually changes first
- Ambiguously returns to state 0-1 or 1-0
 - A so-called "race condition"
 - Or non-deterministic transition



R-S Latch Analysis NOR

11

□ Break feedback path



S	R	Q(t)	Q(t+Δ)
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

hold
reset
set
not allowed

S		R	
Q(t)	0	0	X
1	0	X	1

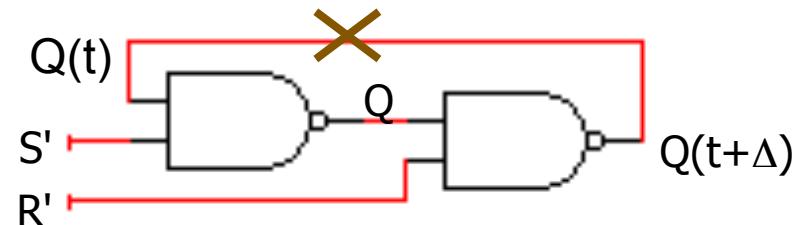
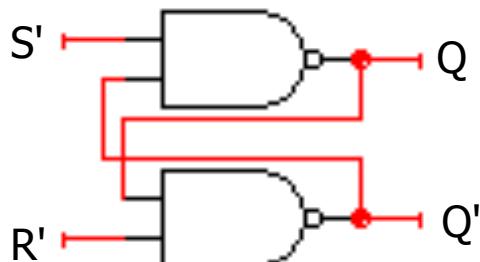
characteristic equation
$$Q(t+\Delta) = S + R' \cdot Q(t)$$

R-S Latch Analysis NAND

12

□ Break feedback path

Note SR are swapped and negated



S	R	Q(t)	Q(t+Δ)
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

hold
reset
set
not allowed

		S	
		0	1
Q(t)	0	X	1
	1	0	X

R

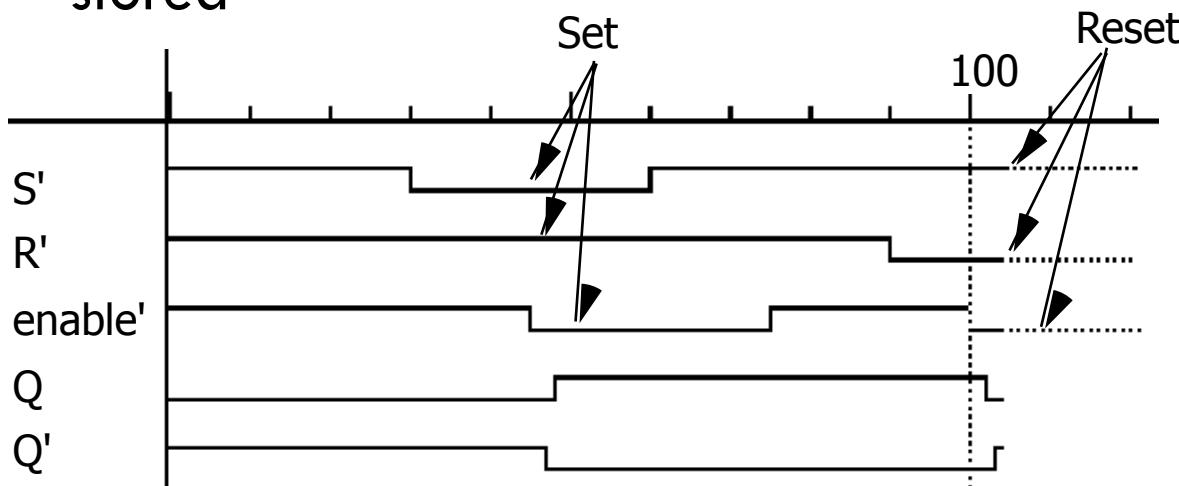
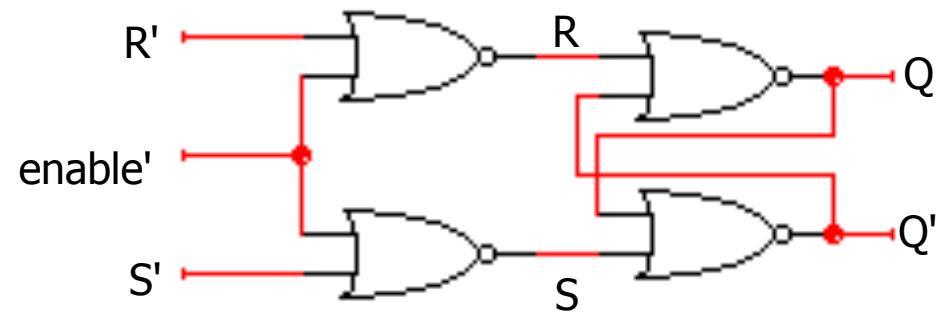
$$\text{characteristic equation}$$
$$Q(t+\Delta) = S + R' \cdot Q(t)$$

Gated R-S Latch

13

- Control when R and S inputs matter

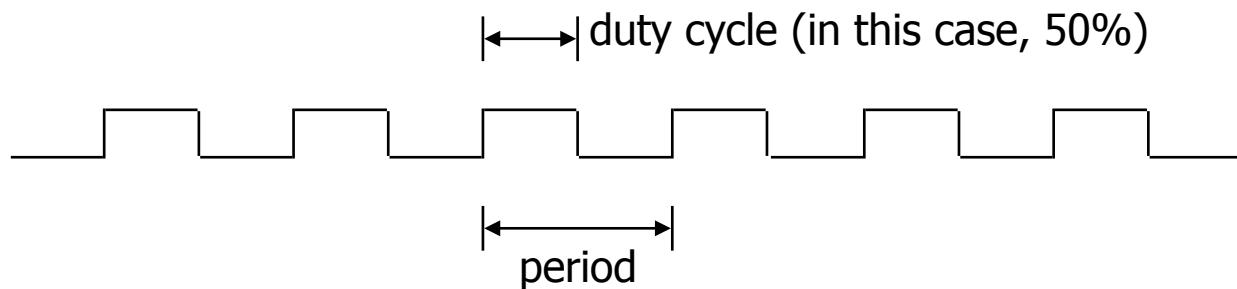
- Otherwise, the slightest glitch on R or S while enable is low could cause change in value stored



Clocks

14

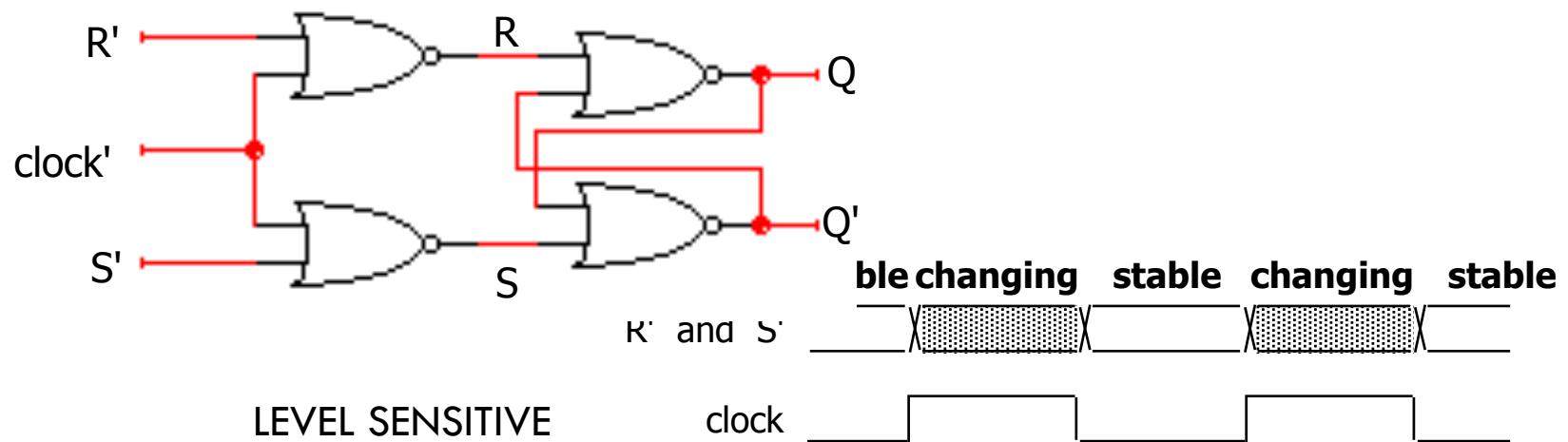
- Used to keep time
 - Wait long enough for inputs (R' and S') to settle
 - Then allow to have effect on value stored
- Clocks are regular periodic signals
 - Period (time between ticks)
 - Duty-cycle (time clock is high between ticks - expressed as % of period)



Clocks (cont'd)

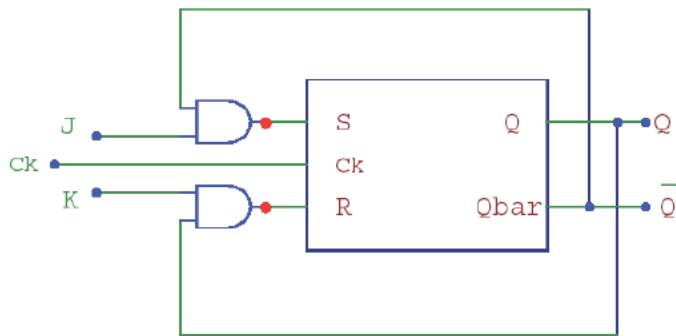
15

- Controlling an R-S latch with a clock
 - Can't let R and S change while clock is active (allowing R and S to pass)
 - Only have half of clock period for signal changes to propagate
 - Signals must be stable for the other half of clock period



J-K latch

Non esiste più lo stato proibito che viene anzi trasformato per implementare una funzione molto utile: se $J=K=1$ le uscite vengono complementate



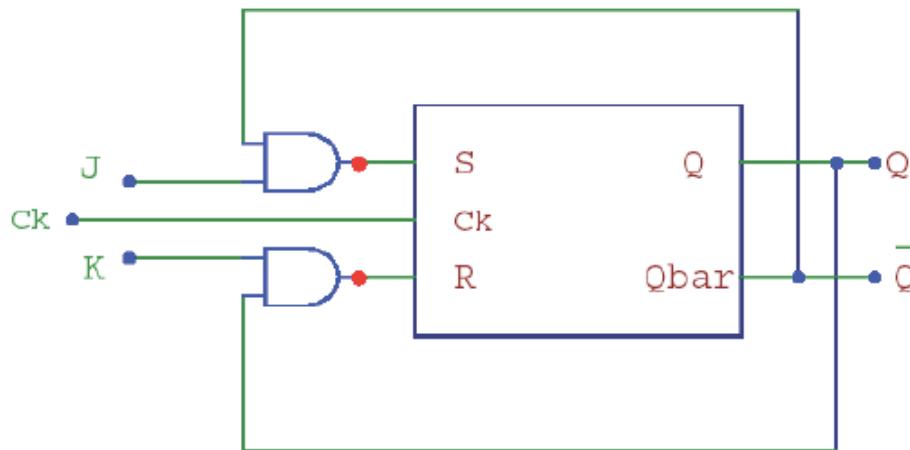
K	J	Q_n	\overline{Q}_n
0	0	Q_{n-1}	\overline{Q}_{n-1}
0	1	1	0
1	0	0	1
1	1	\overline{Q}_{n-1}	Q_{n-1}

E	S	R	Q_n	\overline{Q}_n
0	x	x	Q_{n-1}	\overline{Q}_{n-1}
1	0	0	Q_{n-1}	\overline{Q}_{n-1}
1	0	1	0	1
1	1	0	1	0
1	1	1	\overline{Q}_{n-1}	Q_{n-1}

Da evitare

- $J=K=0 \rightarrow$ Entrambe le porte in ingresso disabilitate $\rightarrow S=R=0$ e quindi mantengo lo stato
- $J=1 K=0 \rightarrow R=0$ comunque e $S=1$ solo se avevo $Q_{\text{bar}}=1$ quindi: o Q era già alto o lo diventa se non lo era.

J-K latch: J=K=1 il toggle



K	J	Q_n	\bar{Q}_n
0	0	Q_{n-1}	\bar{Q}_{n-1}
0	1	1	0
1	0	0	1
1	1	\bar{Q}_{n-1}	Q_{n-1}

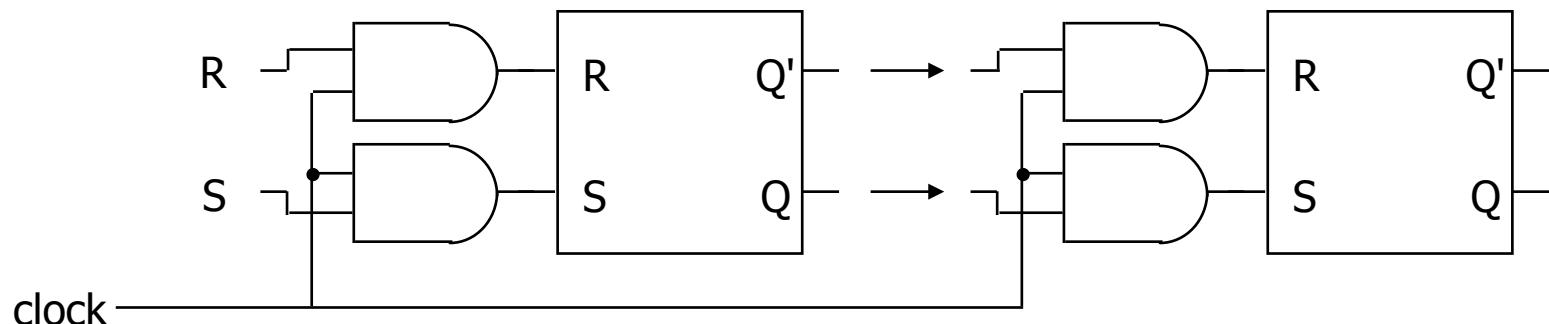
$J=K=1 \rightarrow$ Abilito solo la porta che aveva Q o Qbar alto. ES. Se $Q=1 \rightarrow$ Solo K e' abilitata e quindi ho Reset.

Di conseguenza $J=K=1$ manda in uscita lo stato complementare a quello precedente ... ma come faccio ad impedire una continua oscillazione ? Devo trovare una soluzione per interrompere il circuito di feedback.

Cascading Latches

18

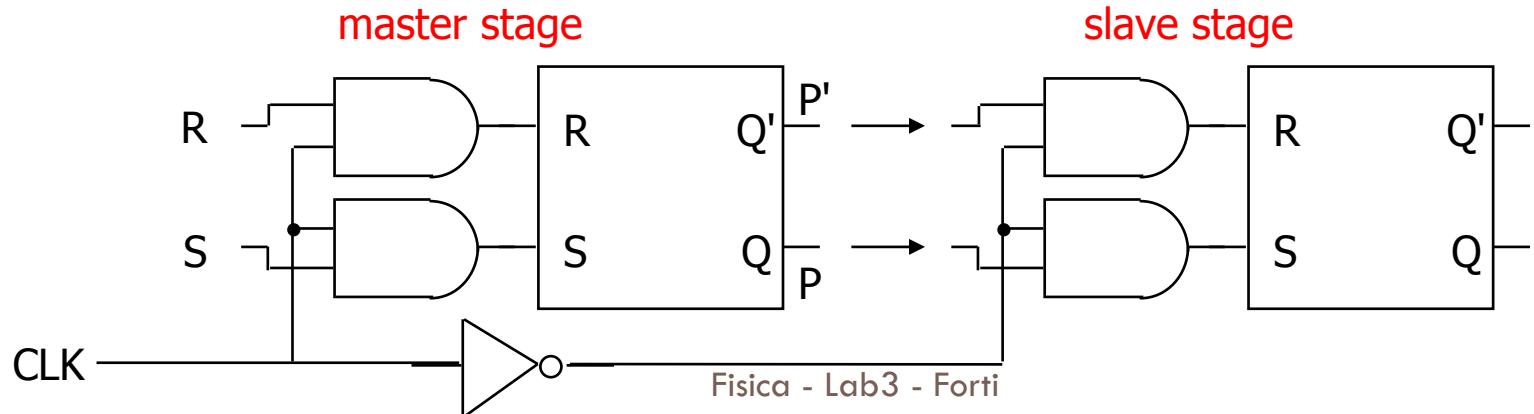
- Connect output of one latch to input of another
- How to stop changes from racing through chain?
 - Need to control flow of data from one latch to the next
 - Advance from one latch per clock period
 - Worry about logic between latches (arrows) that is too fast
 - Clock enable time must be shorter than propagation time



Master-Slave Structure

19

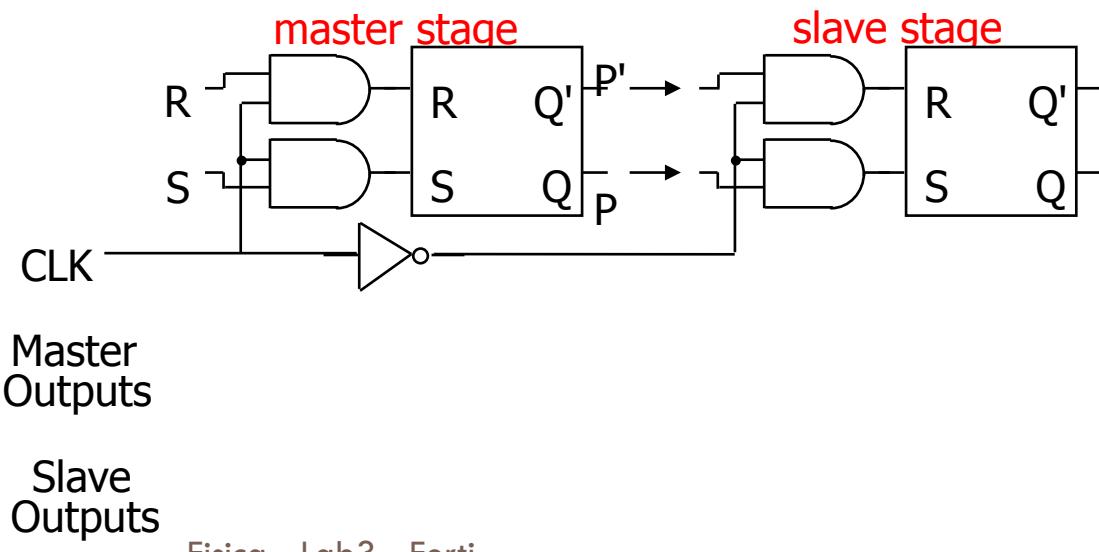
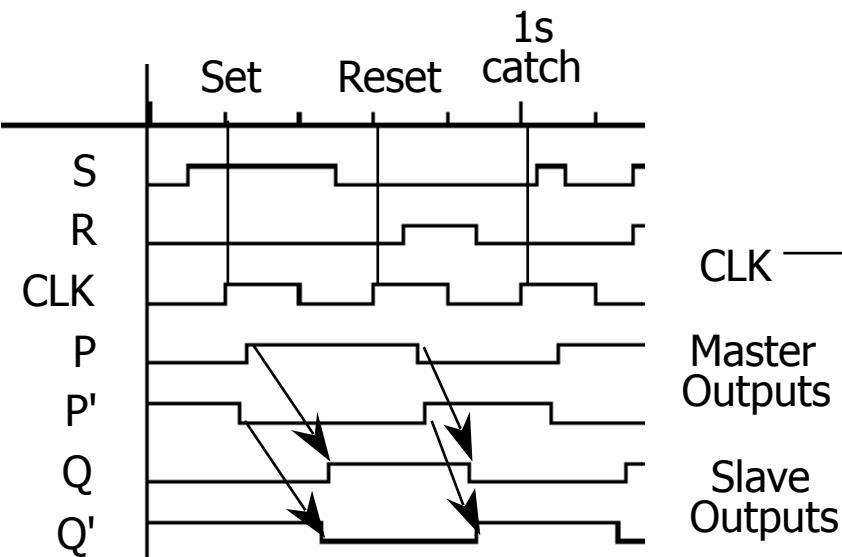
- Break flow by alternating clocks (like an air-lock)
 - Use positive clock to latch inputs into one R-S latch
 - Use negative clock to change outputs with another R-S latch
- View pair as one basic unit
 - master-slave flip-flop
 - twice as much logic
 - output changes a few gate delays after the falling edge of clock but does not affect any cascaded flip-flops



The 1s Catching Problem

20

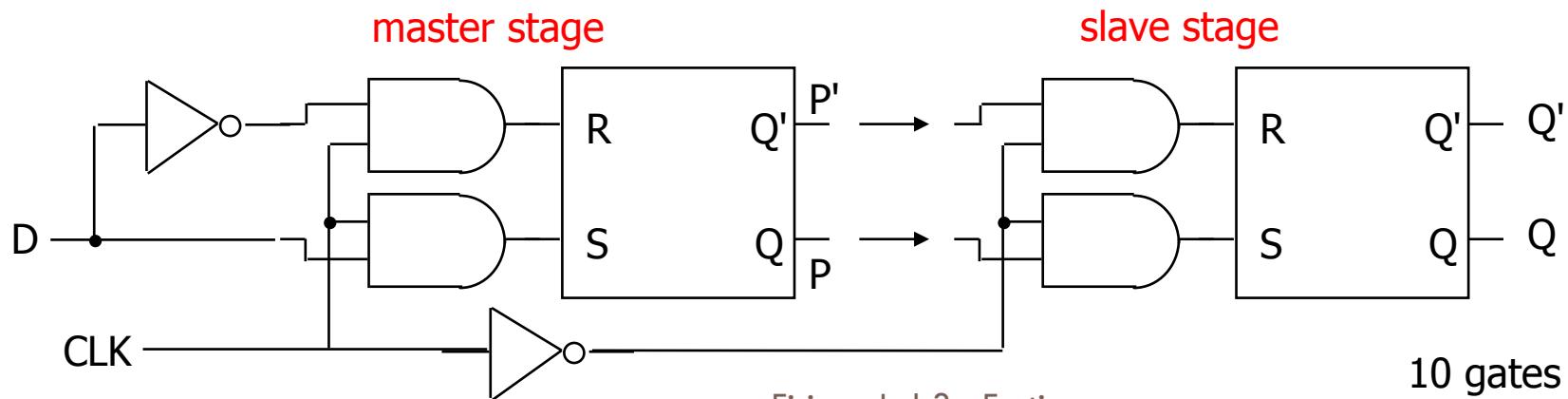
- In first R-S stage of master-slave FF
 - 0-1-0 glitch on R or S while clock is high "caught" by master stage
 - Leads to constraints on logic to be hazard-free



D Flip-Flop

21

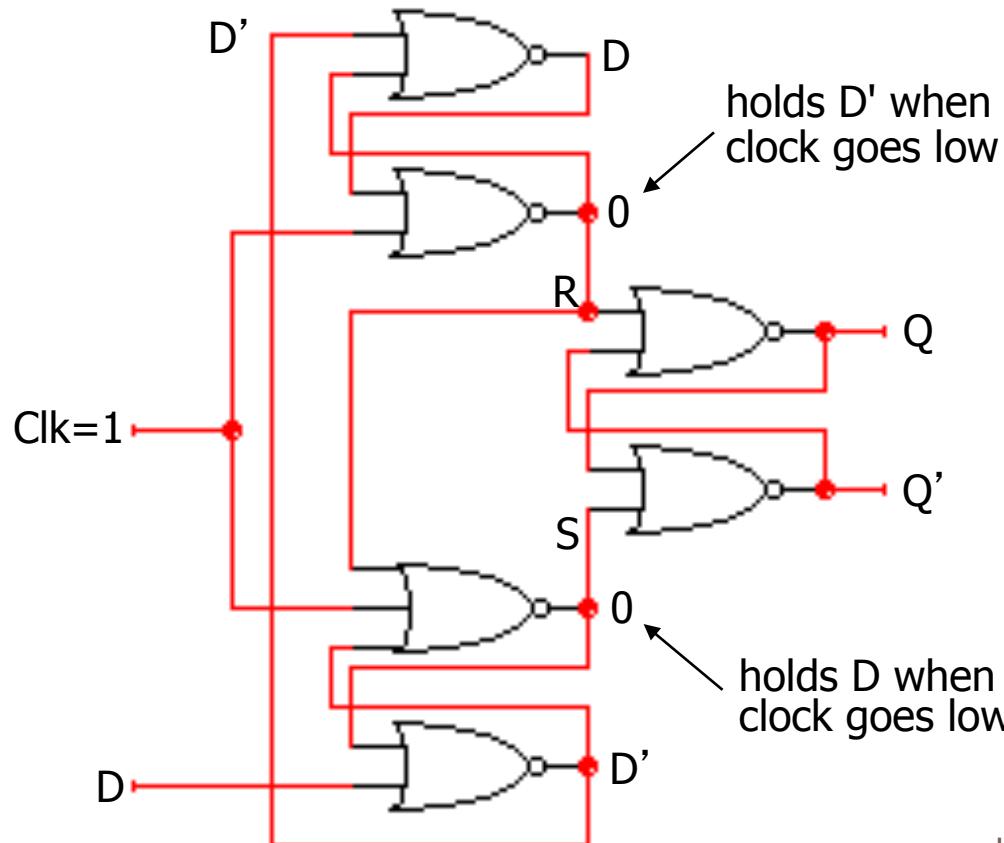
- Make S and R complements of each other
 - Eliminates 1s catching problem
 - Can't just hold previous value (must have new value ready every clock period)
 - Value of D just before clock goes low is what is stored in flip-flop
 - Can make R-S flip-flop by adding logic to make $D = S + R' \cdot Q$



Edge-Triggered Flip-Flops

22

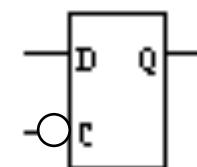
- More efficient solution: only 6 gates
 - sensitive to inputs only near edge of clock signal (not while high)



negative edge-triggered D flip-flop (D-FF)

4-5 gate delays

must respect setup and hold time constraints to successfully capture input

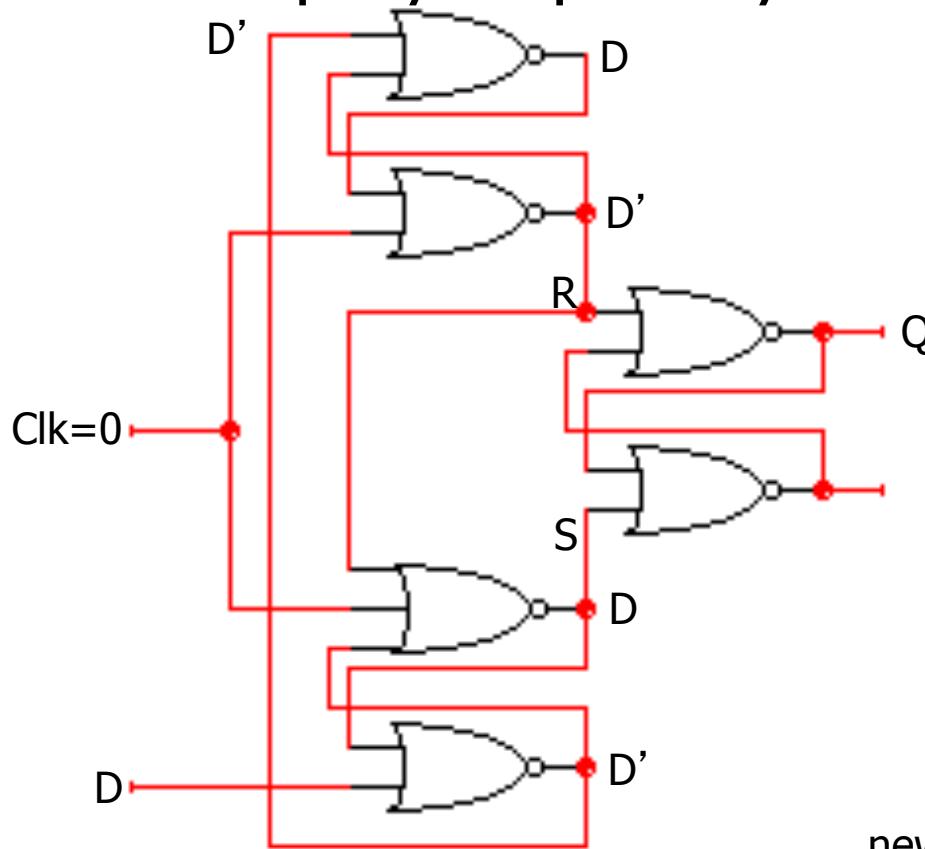


characteristic equation
 $Q(t+1) = D$

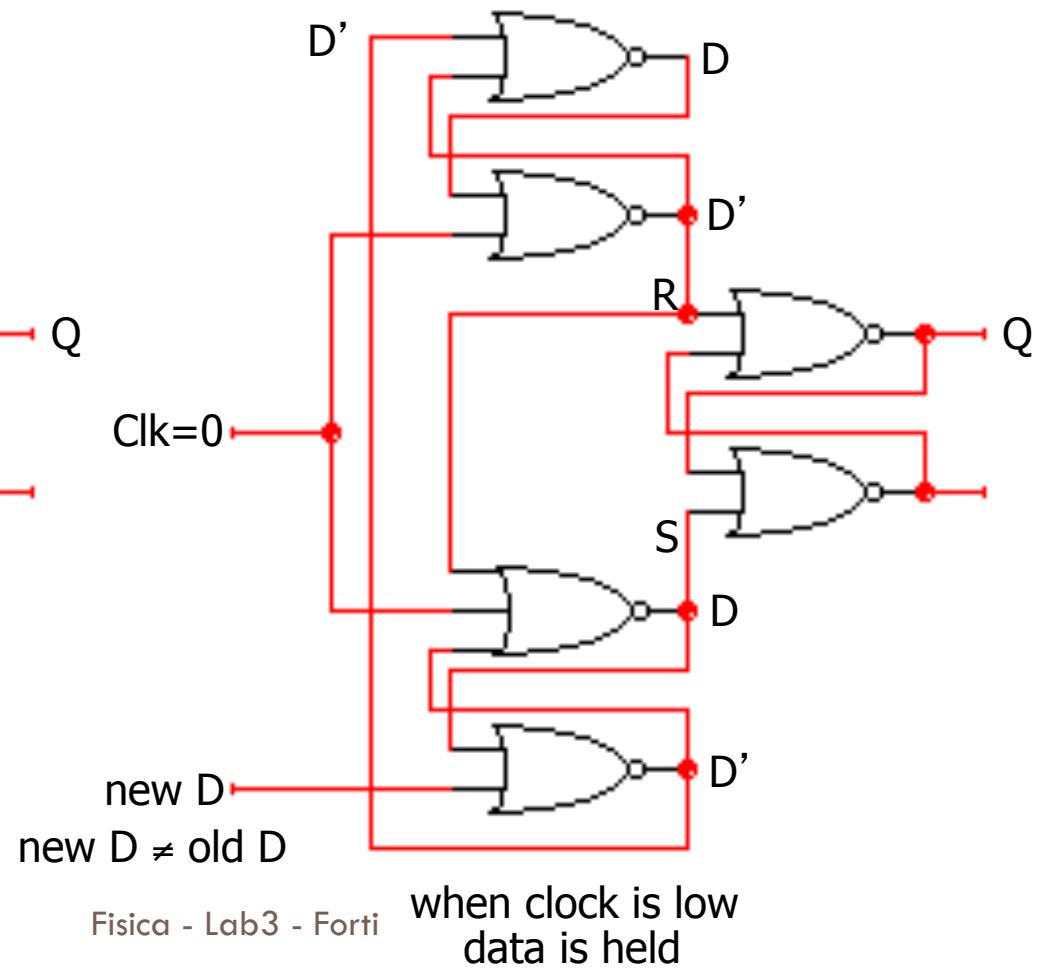
Edge-Triggered Flip-Flops (cont'd)

23

□ Step-by-step analysis



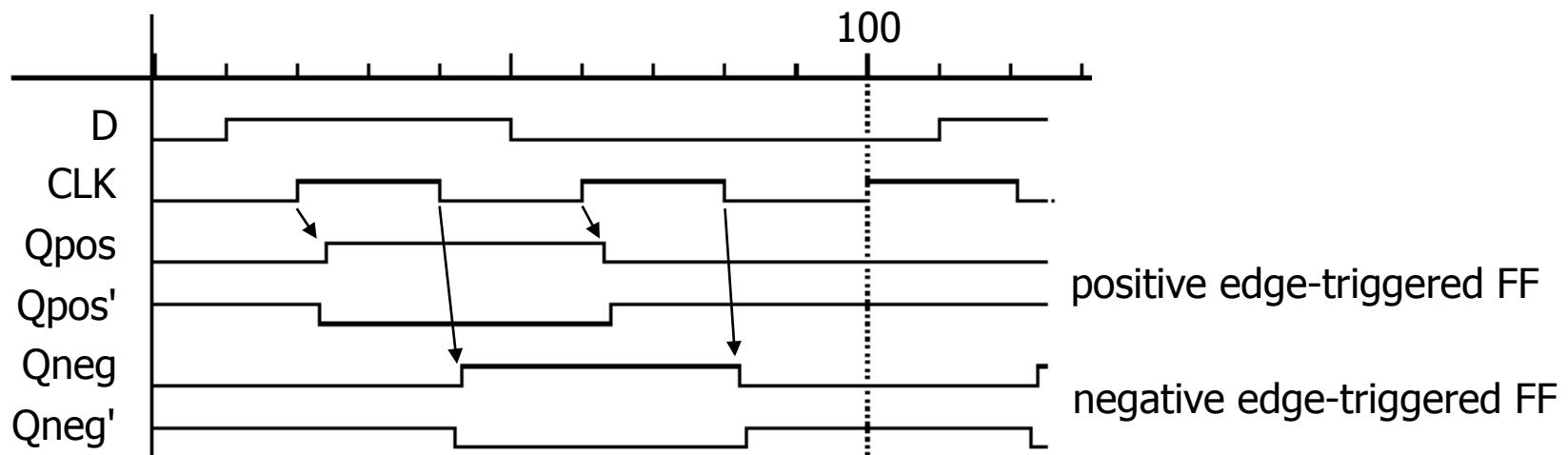
when clock goes high-to-low
data is latched



Edge-Triggered Flip-Flops (cont'd)

24

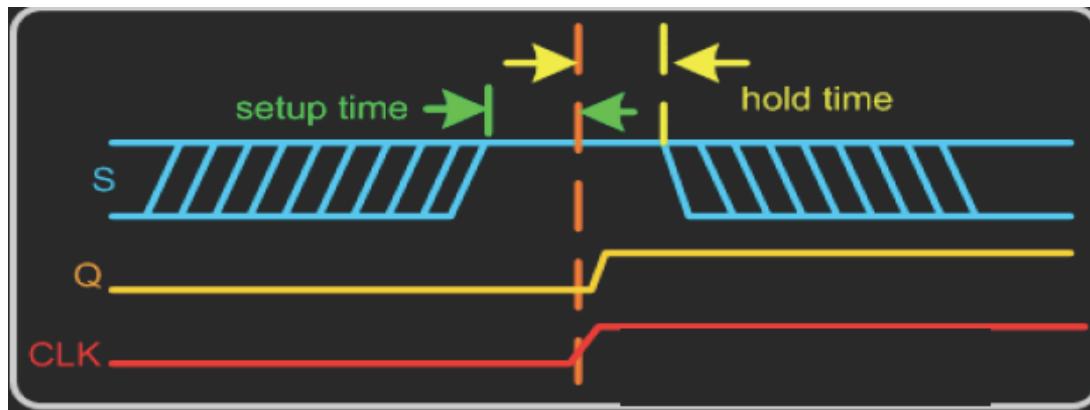
- Positive edge-triggered
 - Inputs sampled on rising edge; outputs change after rising edge
- Negative edge-triggered flip-flops
 - Inputs sampled on falling edge; outputs change after falling edge



Tempi nel Flip-Flop: Setup time, Hold Time

Tempo di Setup (T_{su}) = minimo Δt **prima** dell'arrivo dell'evento di Clock durante il quale gli ingressi devono essere stabili

Tempo di Hold (T_h) = minimo Δt **dopo** l'arrivo dell'evento di Clock durante il quale gli ingressi devono essere stabili



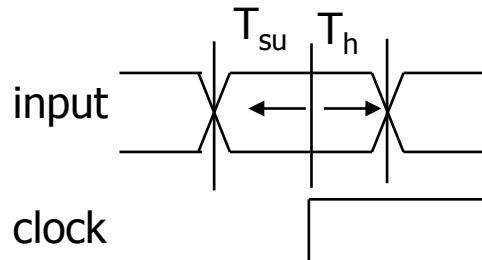
Gli ingressi devono essere stabili per un lasso di tempo = $T_{su} + T_h$ per essere riconosciuti e trasmessi correttamente
Anche il Clock ha un tempo minimo in cui deve essere costante per avere il corretto funzionamento del circuito.

Timing Methodologies (cont' d)

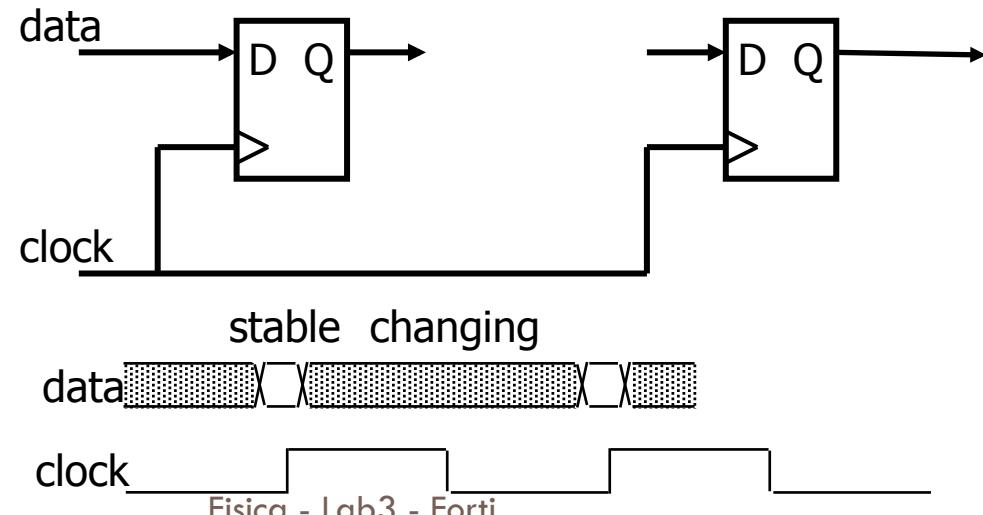
26

□ Definition of terms

- clock: periodic event, causes state of memory element to change; can be rising or falling edge, or high or low level
- setup time: minimum time before the clocking event by which the input must be stable (T_{su})
- hold time: minimum time after the clocking event until which the input must remain stable (T_h)

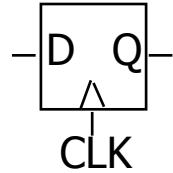


there is a timing "window" around the clocking event during which the input must remain stable and unchanged in order to be recognized

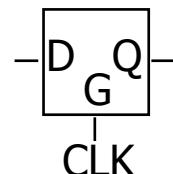


Comparison of Latches and Flip-Flops

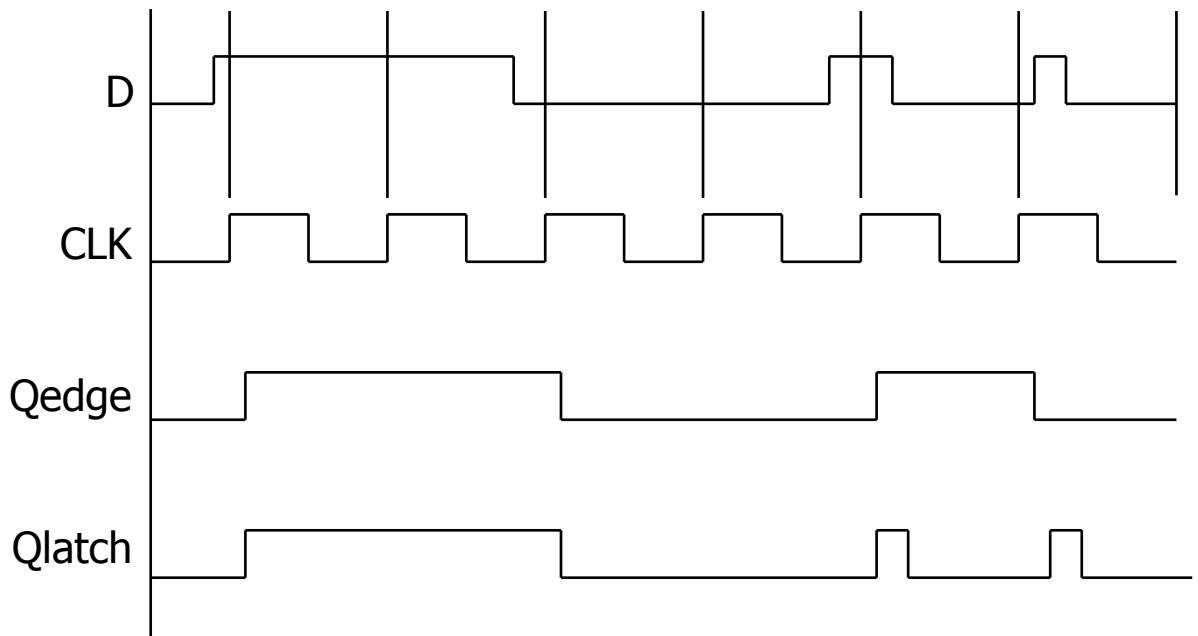
27



positive
edge-triggered
flip-flop



transparent
(level-sensitive)
latch



behavior is the same unless input changes
while the clock is high

Comparison of Latches and Flip-Flops (cont'd)

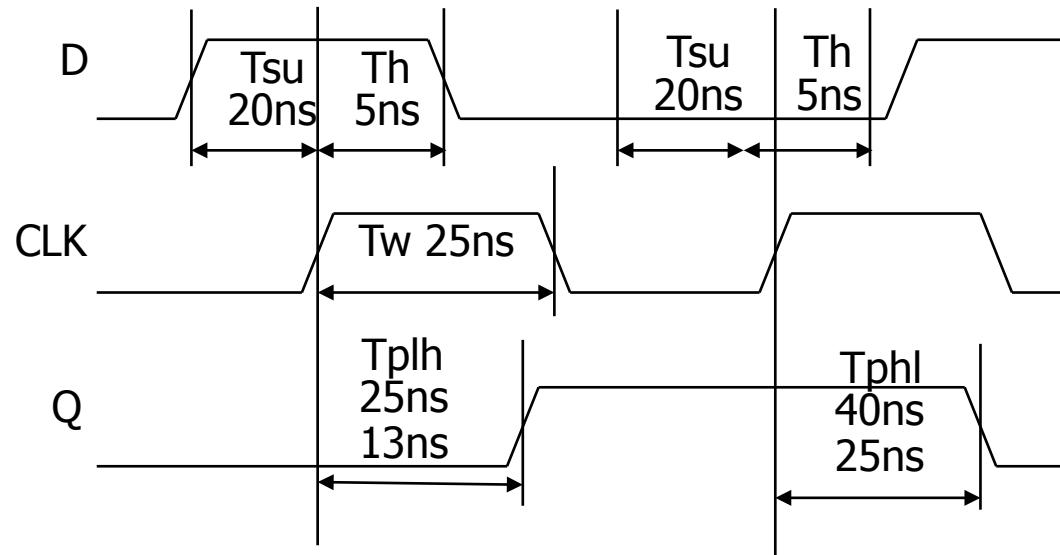
28

Type	When inputs are sampled	When output is valid
unclocked latch	always	propagation delay from input change
level-sensitive latch	clock high (T_{su}/T_h around falling edge of clock)	propagation delay from input change or clock edge (whichever is later)
master-slave flip-flop	clock high (T_{su}/T_h around falling edge of clock)	propagation delay from falling edge of clock
negative edge-triggered flip-flop	clock hi-to-lo transition (T_{su}/T_h around falling edge of clock)	propagation delay from falling edge of clock

Typical Timing Specifications

29

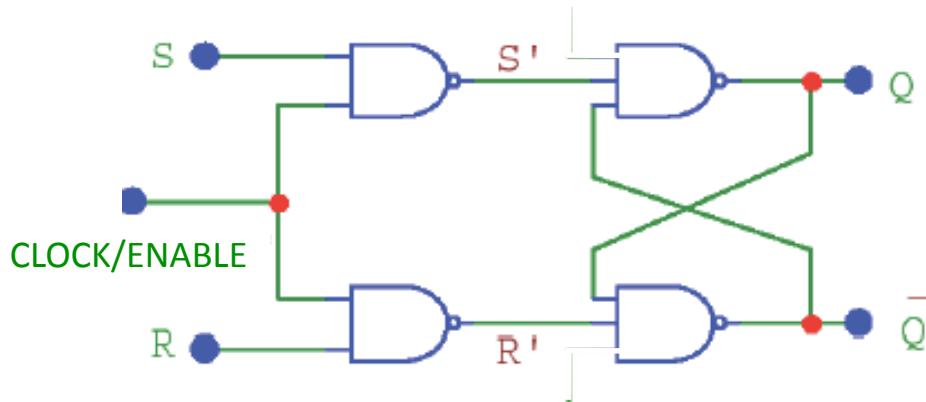
- Positive edge-triggered D flip-flop
 - Setup and hold times
 - Minimum clock width
 - Propagation delays (low to high, high to low, max and typical)



all measurements are made from the clocking event that is,
the rising edge of the clock.

FF e Contatori

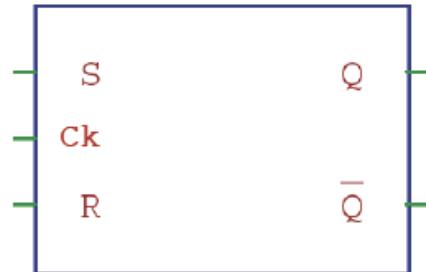
Dalla lezione scorsa: Latch RS Sincrono



E	S	R	Q_n	\bar{Q}_n
0	x	x	Q_{n-1}	\bar{Q}_{n-1}
1	0	0	Q_{n-1}	\bar{Q}_{n-1}
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

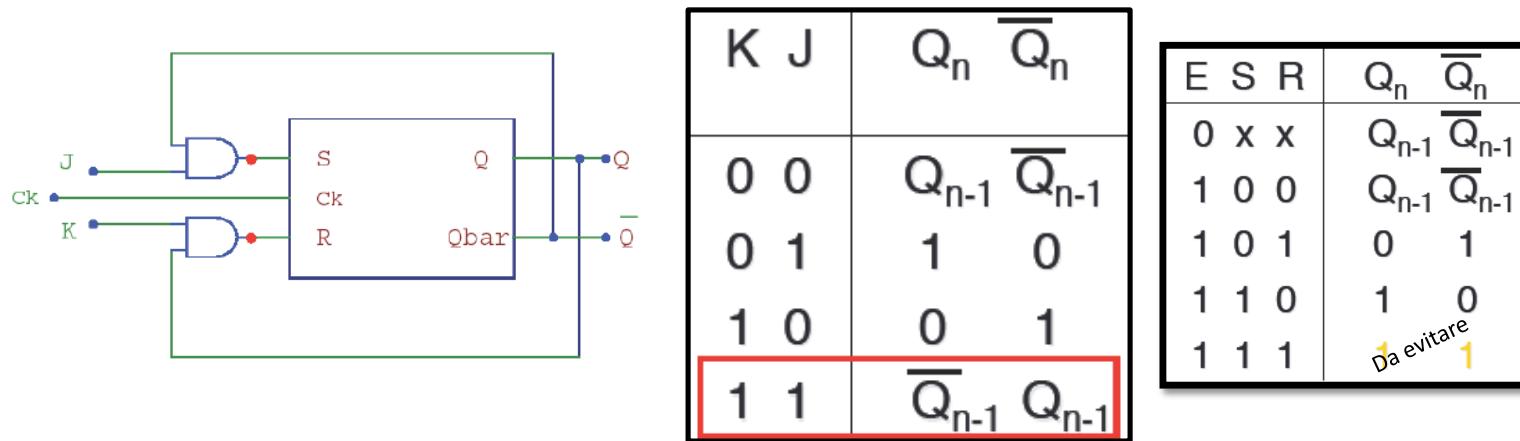
Da evitare

Questo circuito e` abilitato quando
ENABLE=1 -> e` sensibile al **livello
del CLOCK - LEVEL TRIGGERED** un
qualsiasi cambiamento di R ed S
mentre E=1 viene "visto". Con E=1
il latch e` trasparente.



J-K latch

Non esiste più lo stato proibito che viene anzi trasformato per implementare una funzione molto utile: se $J=K=1$ le uscite vengono complementate

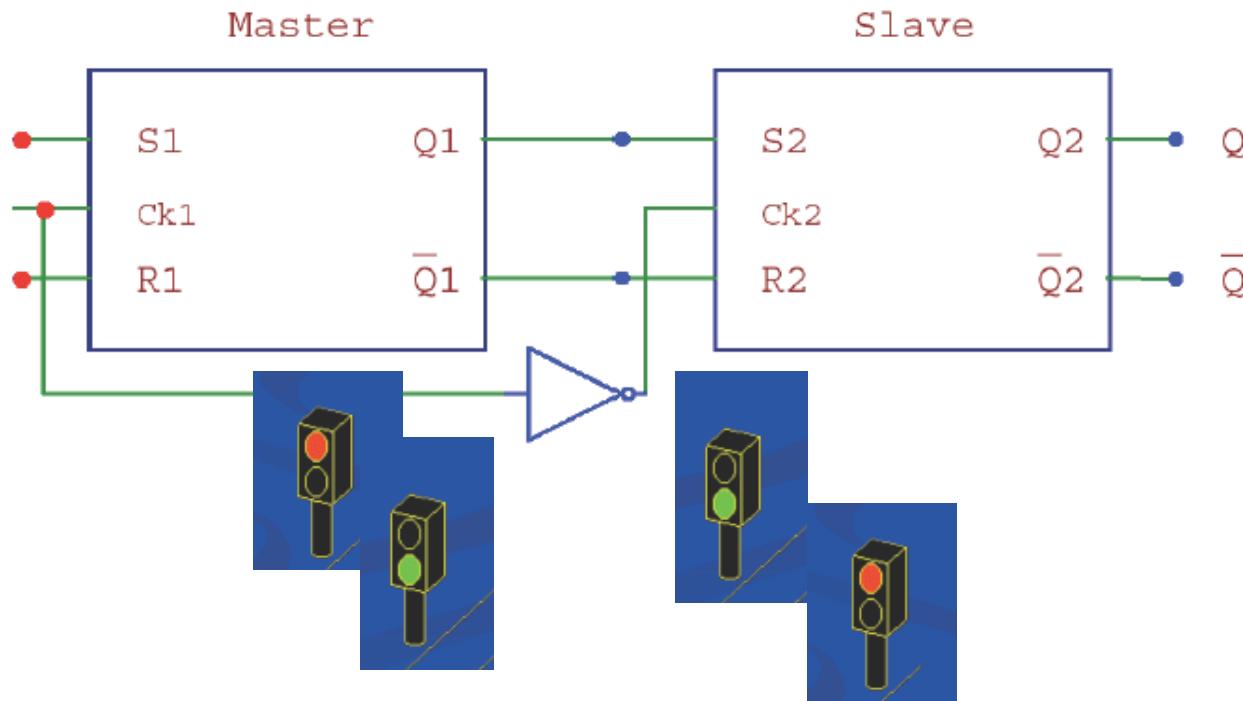


- $J=K=0 \rightarrow$ Entrambe le porte in ingresso disabilitate $\rightarrow S=R=0$ e quindi mantengo lo stato
- $J=1 \ K=0 \rightarrow R=0$ comunque e $S=1$ solo se avevo $Q_{\text{bar}}=1$ quindi: o Q era già alto o lo diventa se non lo era.

$J=K=1 \rightarrow$ Abilito solo la porta che aveva Q o Q_{bar} alto. ES. Se $Q=1 \rightarrow$ Solo K è abilitata e quindi ho Reset.

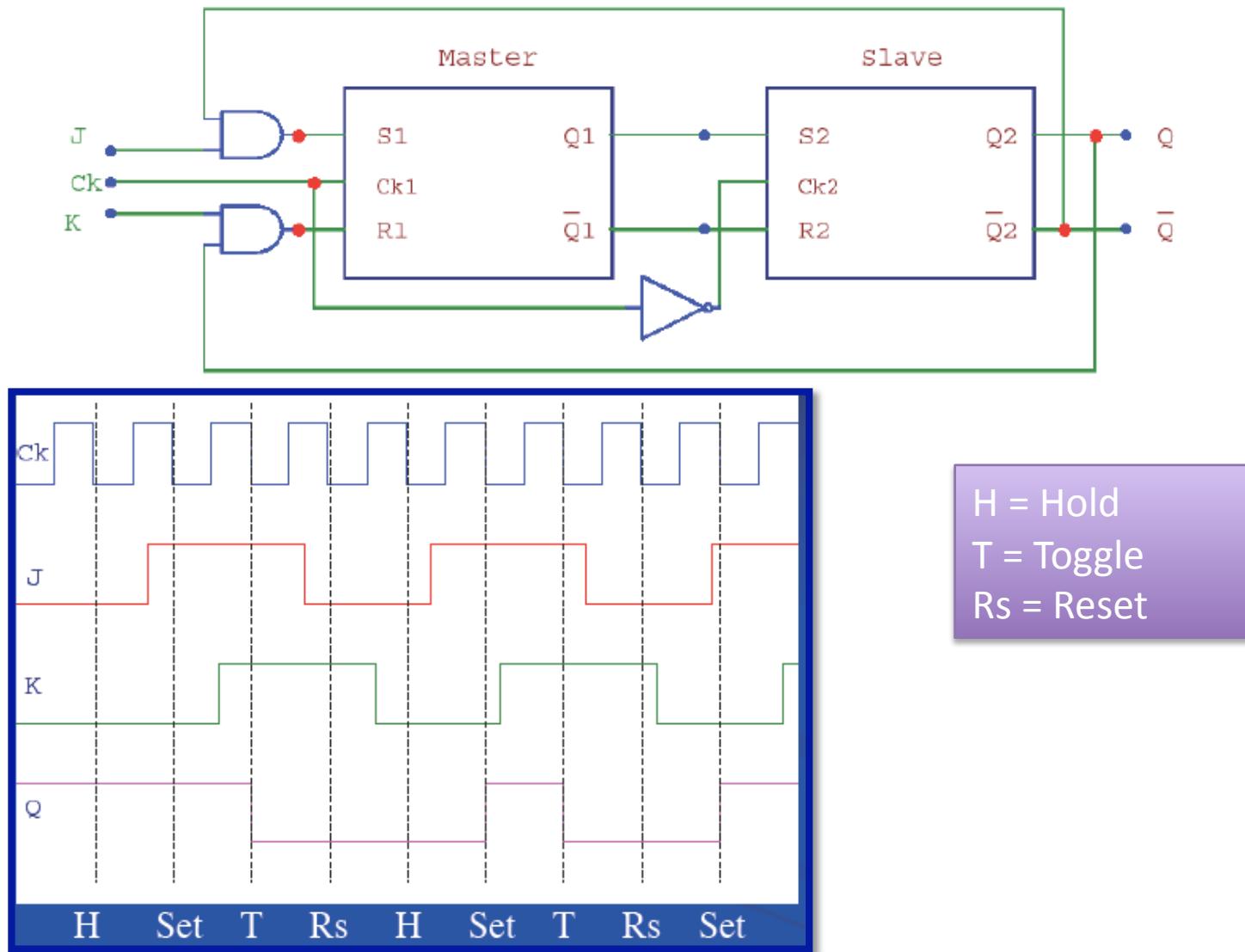
Di conseguenza $J=K=1$ manda in uscita lo stato complementare a quello precedente ... ma come faccio ad impedire una continua oscillazione ? Devo trovare una soluzione per interrompere il circuito di feedback.

Clock sul fronte: Master-Slave



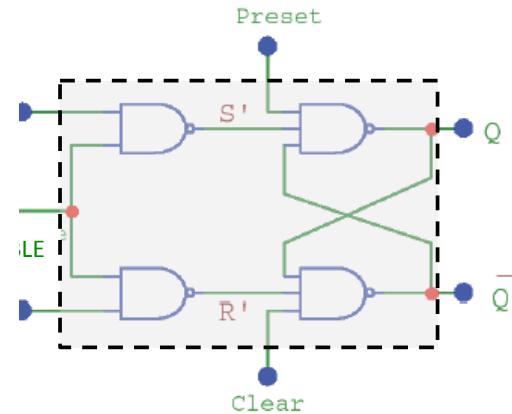
- Quando il *Master* riceve il segnale di CLOCK alto lo *Slave* riceve il segnale di CLOCK basso quindi affinche` le uscite dello *Slave* “ricevano” il segnale dovuto a S1 e R1 si deve aspettare il fronte di discesa del CLOCK
- In modo analogo posso ottenere un circuito sensibile al fronte di salita

J-K Flip-Flop: implementazione e funzionamento



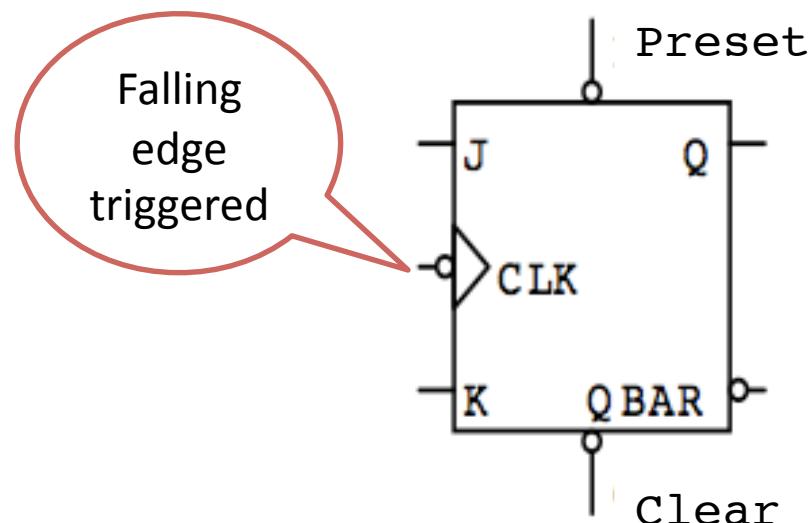
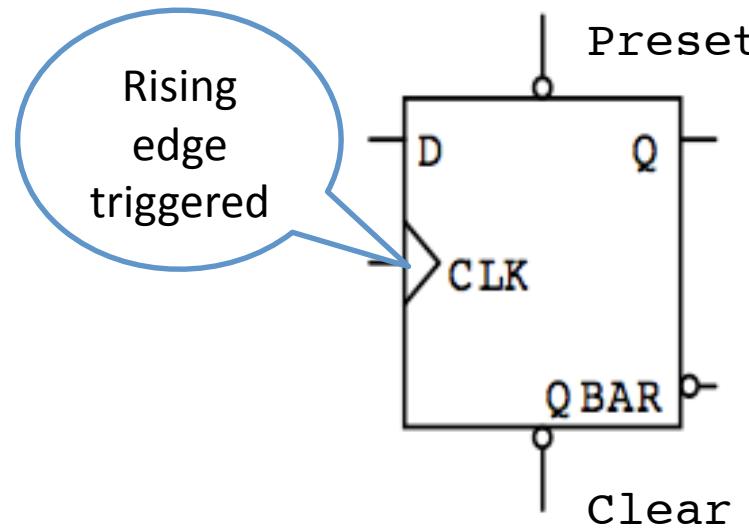
Ingressi sincroni, ingressi asincroni

- In generale oltre agli ingressi S,R o J,K o D lo stato delle uscite del Flip-Flop puo` essere definito con degli ingressi normalmente indicati con Preset e Clear
- S,R (J,K e D) sono ingressi che sono sensibili al segnale di Clock e si dicono per questo SINCRONI
- Preset e Clear sono invece indipendenti dal Clock e si dicono ASINCRONI
- Preset e Clear vengono utilizzati per definire lo stato delle uscite nell'inizializzazione del circuito oppure per un reset in condizioni speciali...



Preset	Clear	Q	Qbar
H	H	Definiti dagli ingressi sincroni	
L	H	H	L
H	L	L	H
L	L	Da evitare	

Simboli e significato



7474

74112

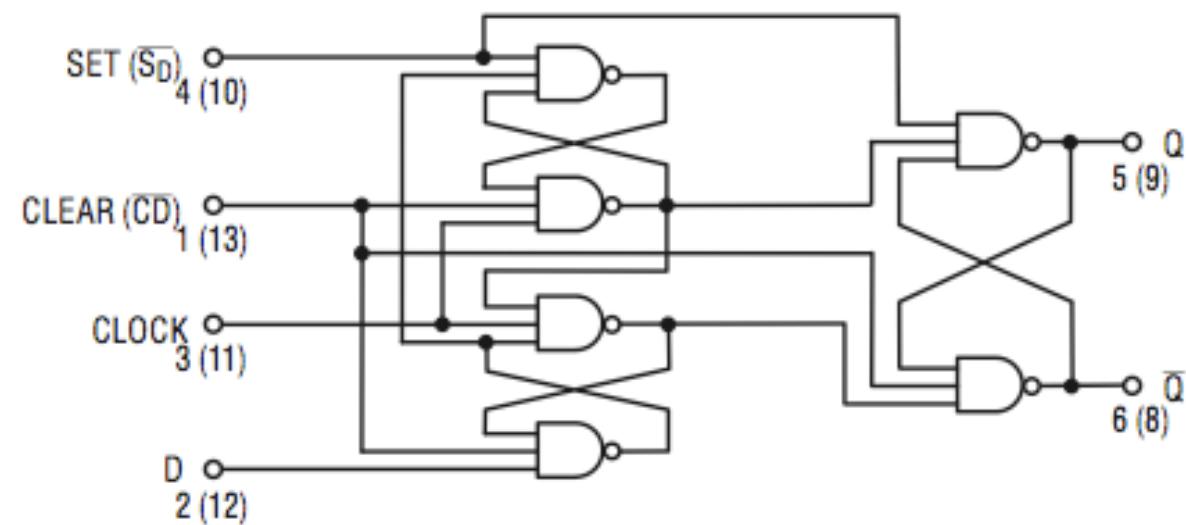
Tipo di integrato

Preset e Clear sono gli ingressi Asincroni e sono attivi BASSI (negazione in ingresso):
Preset=1 Clear=1 Non Attivi
Preset=0 Clear=1 Q=1 Qbar=0
Preset=1 Clear=0 Q=0 Qbar=1

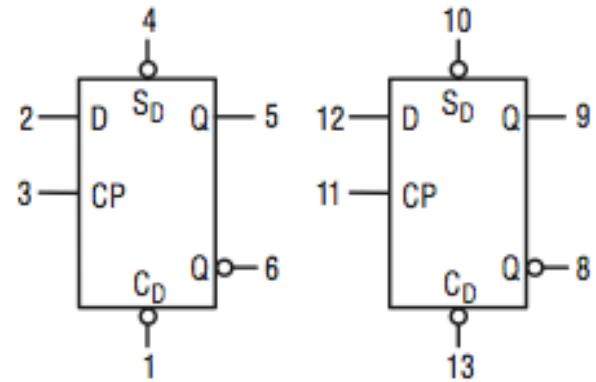
Un'occhiata al data-sheet del 74LS74

SN74LS74A

LOGIC DIAGRAM (Each Flip-Flop)



LOGIC SYMBOL



V_{CC} = PIN 14
GND = PIN 7

MODE SELECT – TRUTH TABLE

OPERATING MODE	INPUTS			OUTPUTS	
	$\bar{S_D}$	$\bar{S_D}$	D	Q	\bar{Q}
Set	L	H	X	H	L
Reset (Clear)	H	L	X	L	H
*Undetermined	L	L	X	H	H
Load "1" (Set)	H	H	h	H	L
Load "0" (Reset)	H	H	I	L	H

- * Both outputs will be HIGH while both $\bar{S_D}$ and $\bar{C_D}$ are LOW, but the output states are unpredictable if $\bar{S_D}$ and $\bar{C_D}$ go HIGH simultaneously. If the levels at the set and clear are near V_{IL} maximum then we cannot guarantee to meet the minimum level for V_{OH} .

H, h = HIGH Voltage Level

L, I = LOW Voltage Level

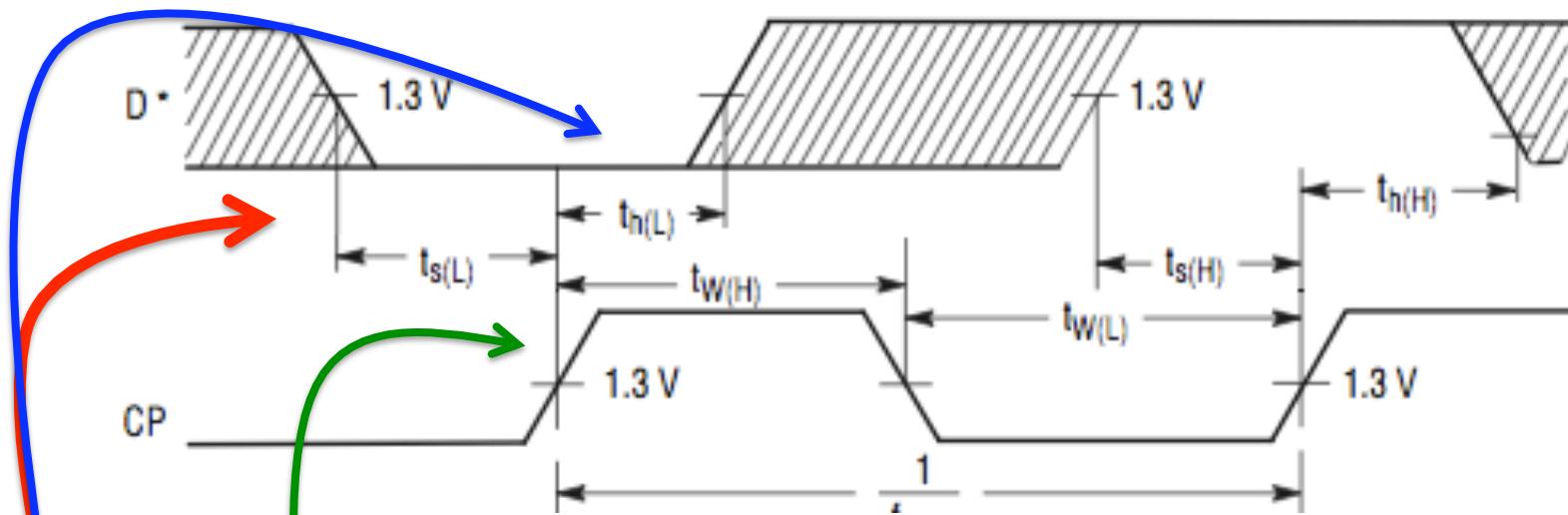
X = Don't Care

I, h (q) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the HIGH to LOW clock transition.

C_D bar



Un'occhiata al data-sheet del 74LS74

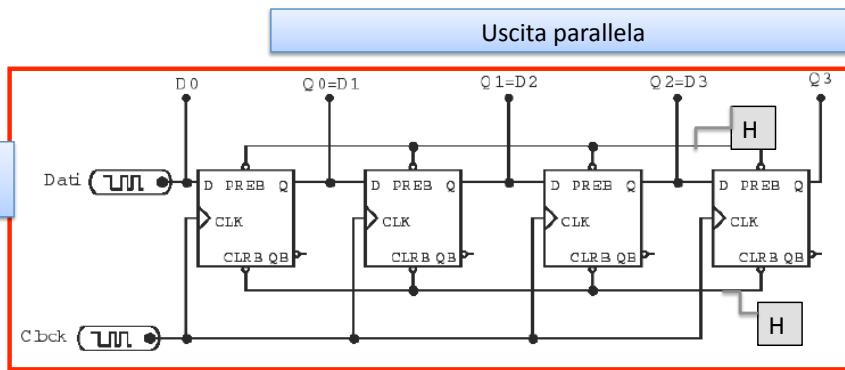


Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
$t_{w(H)}$	Clock	25			ns
$t_{w(L)}$	Clear, Set	25			ns
t_s	Data Setup Time – HIGH				ns
	LOW	20			ns
t_h	Hold Time	5.0			ns

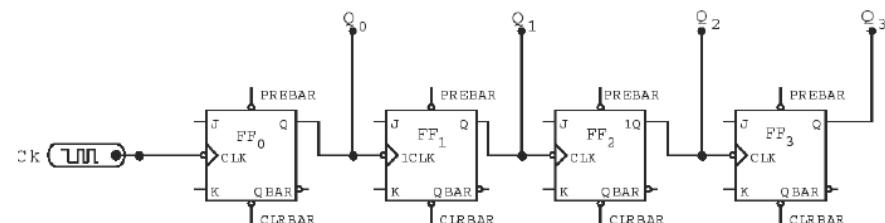
Contatori e registri

40

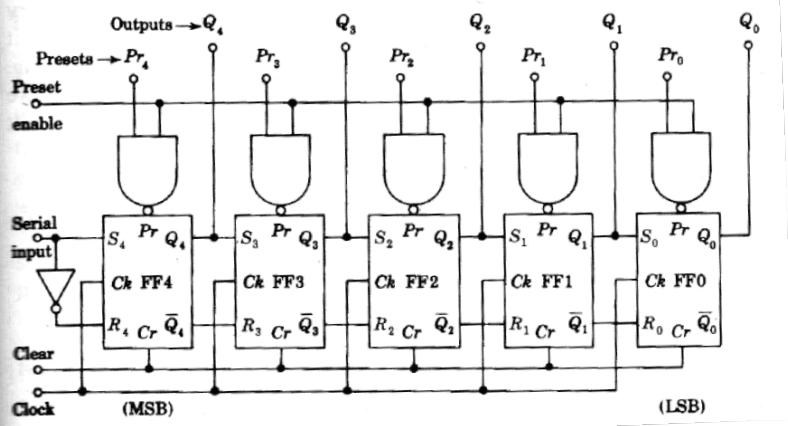
Registri a scorrimento (shift registers)



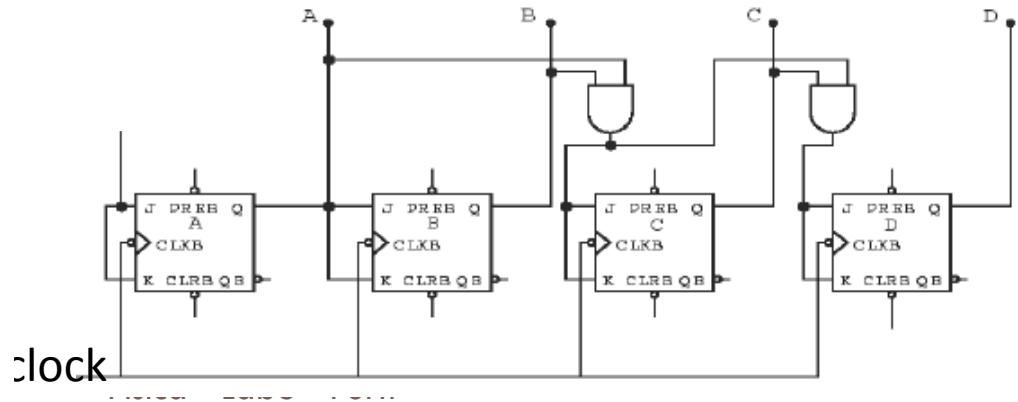
Contatore asincrono con FF JK in toggle



Registro a scorrimento con FF D o RS



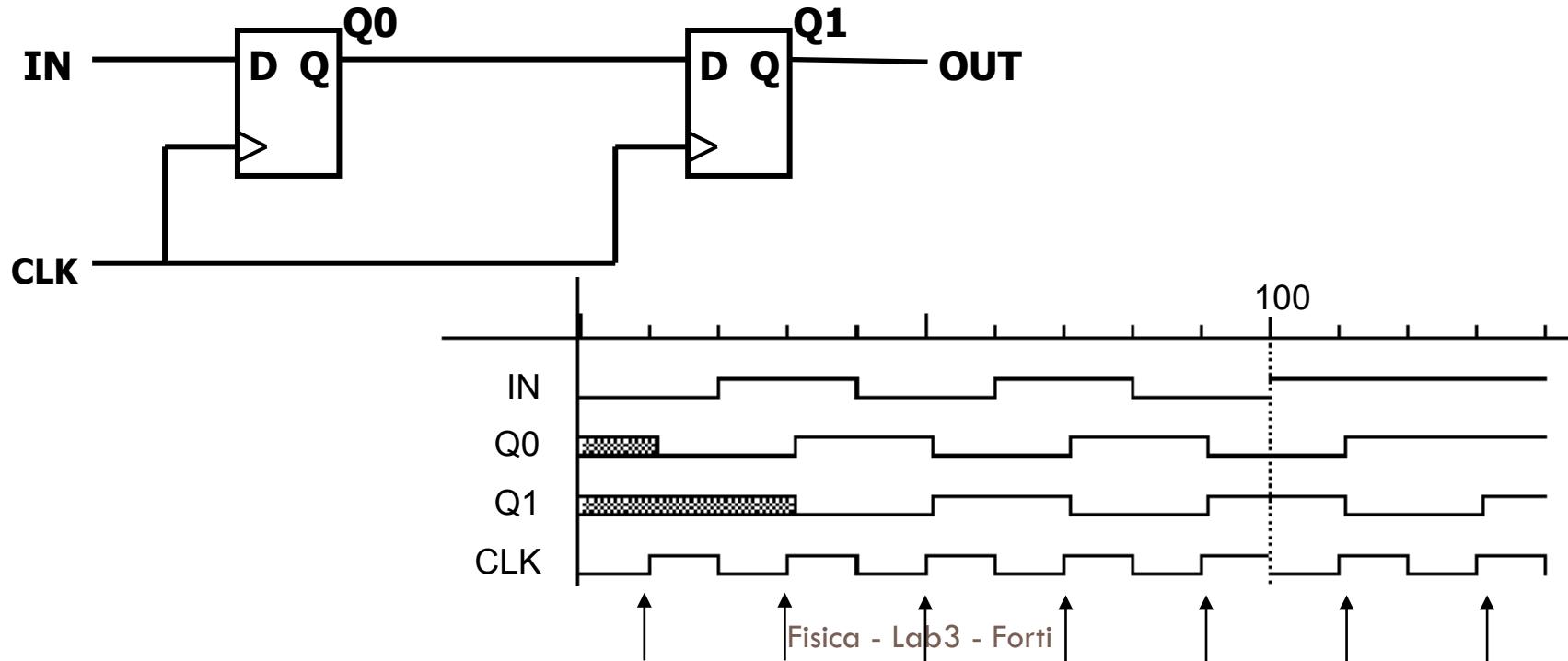
Contatore sincrono con FF JK in toggle



Cascading Edge-triggered Flip-Flops

41

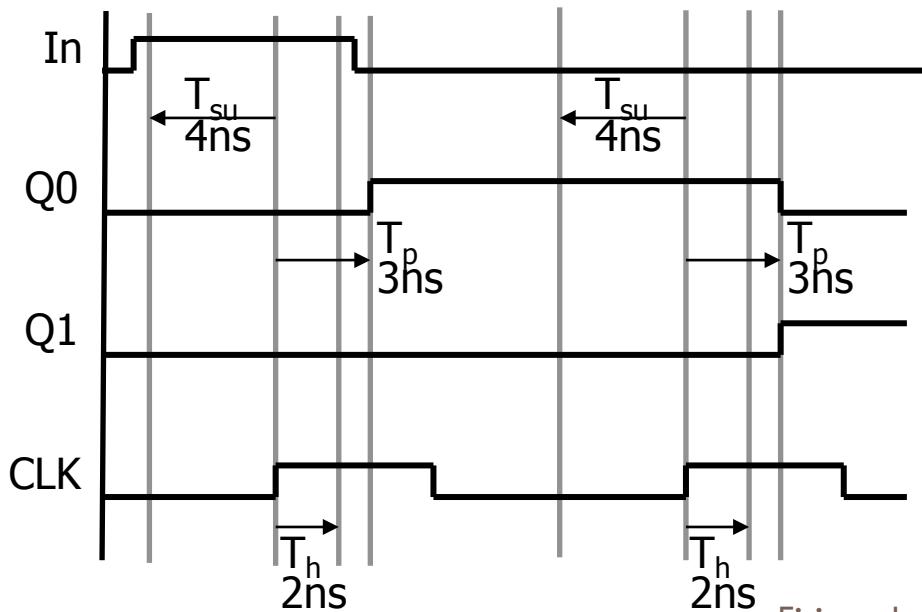
- Shift register
 - New value goes into first stage
 - While previous value of first stage goes into second stage
 - Consider setup/hold/propagation delays (prop must be > hold)



Cascading Edge-triggered Flip-Flops (cont'd)

42

- Why this works
 - Propagation delays exceed hold times
 - Clock width constraint exceeds setup time
 - This guarantees following stage will latch current value before it changes to new value

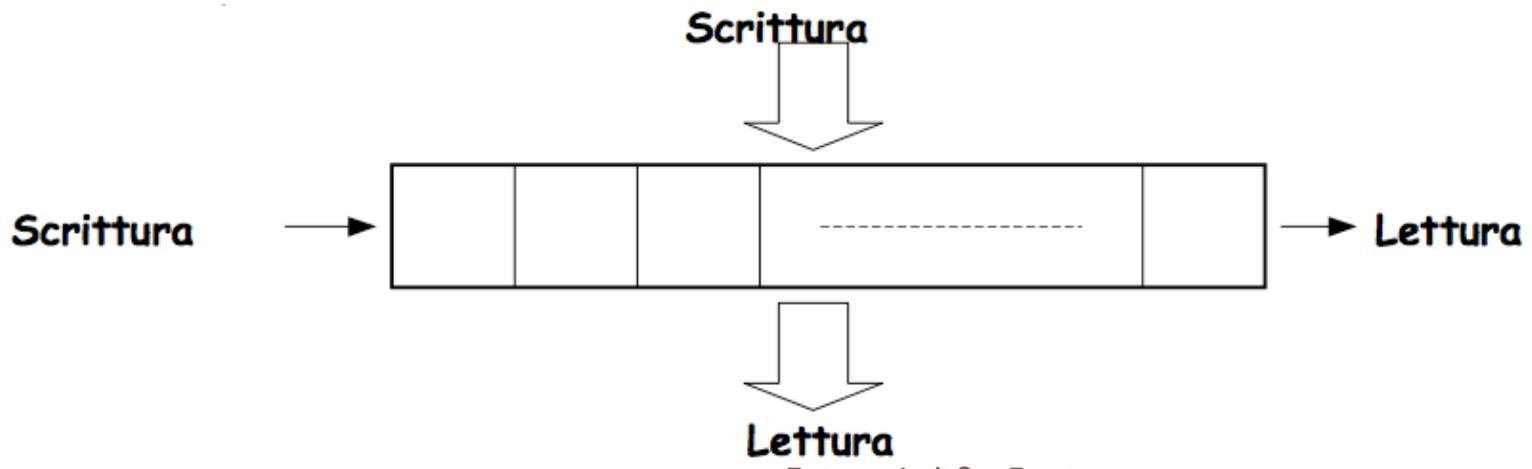


timing constraints
guarantee proper
operation of
cascaded components

assumes infinitely fast
distribution of the clock

Shift Register – Registri a Scorrimento

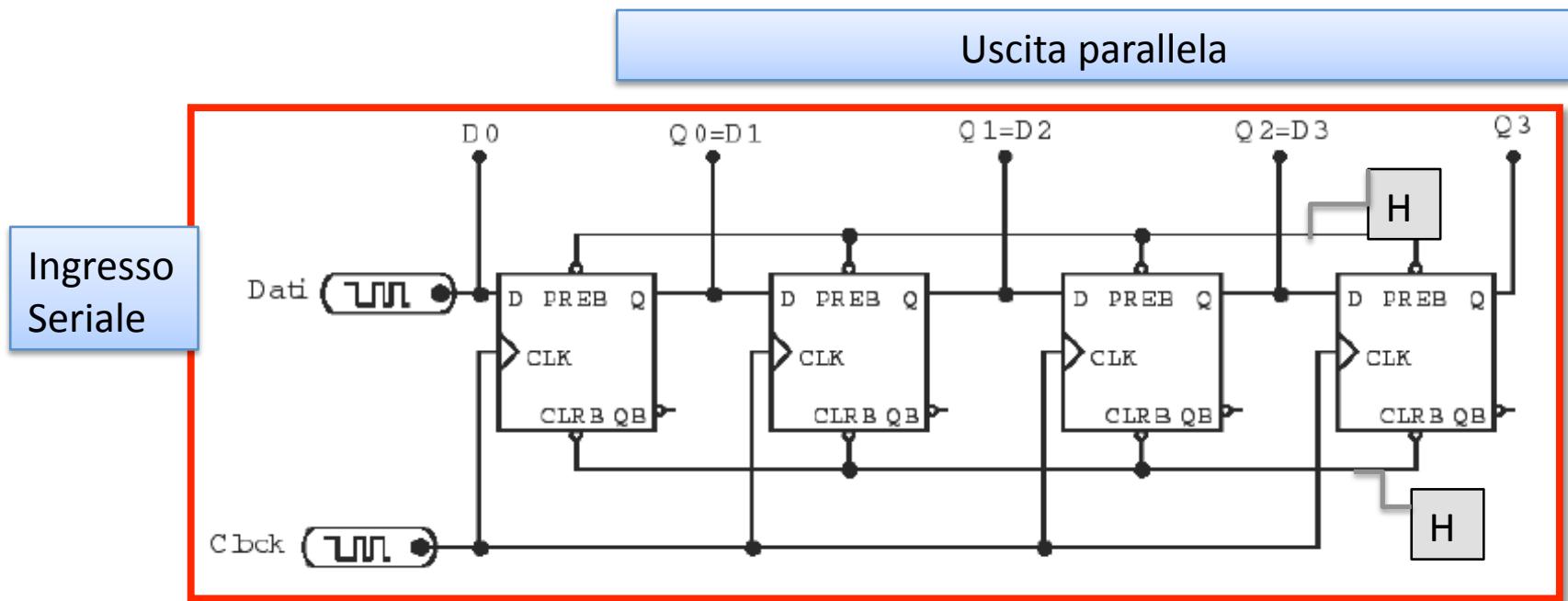
- Shift Register (SR) o Registri a Scorrimento sono catene di Flip-Flop collegate tra loro in modo da memorizzare una stringa di bits
- Le stringhe di bit possono essere immesse nel Registro in parallelo o in serie
- Le stringhe di bit possono essere estratta dal registro in parallelo o in serie



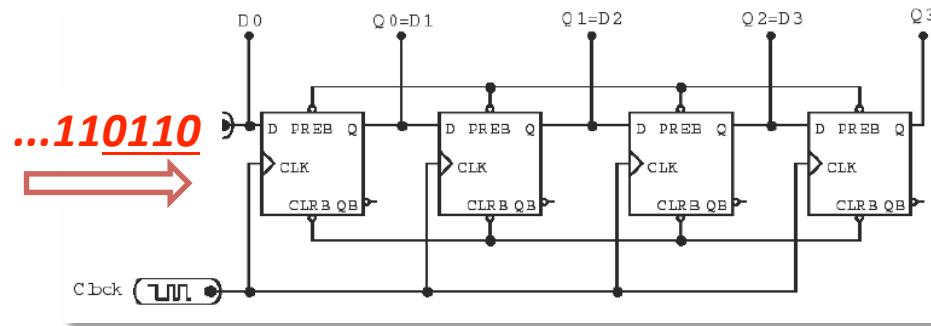
Shift Register – Registri a Scorrimento

Esempio: Registro a scorrimento realizzato con flip-flop di tipo D con ingresso seriale e uscita parallela

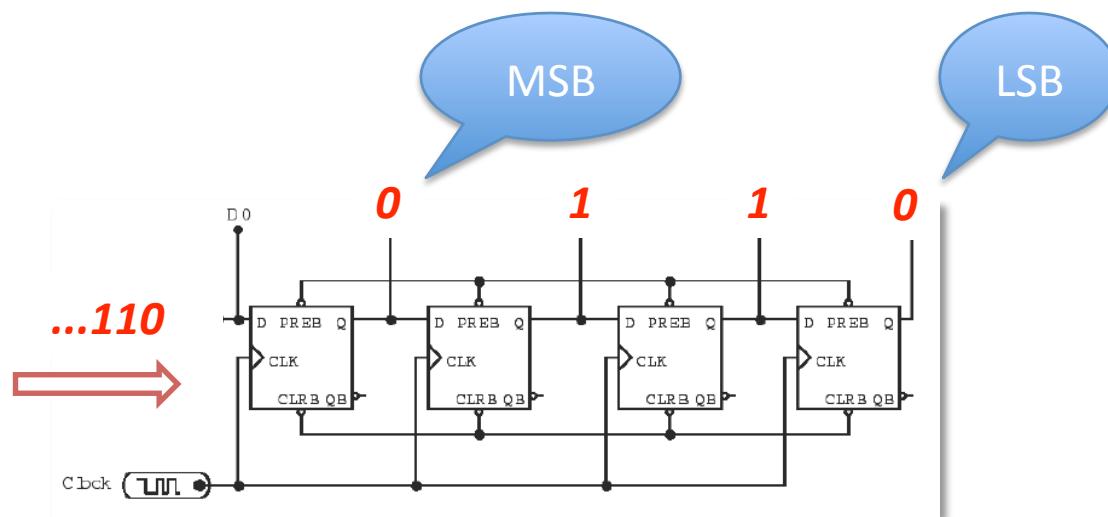
Permette la conversione seriale/parallela



Shift Register con flip-flop D: funzionamento



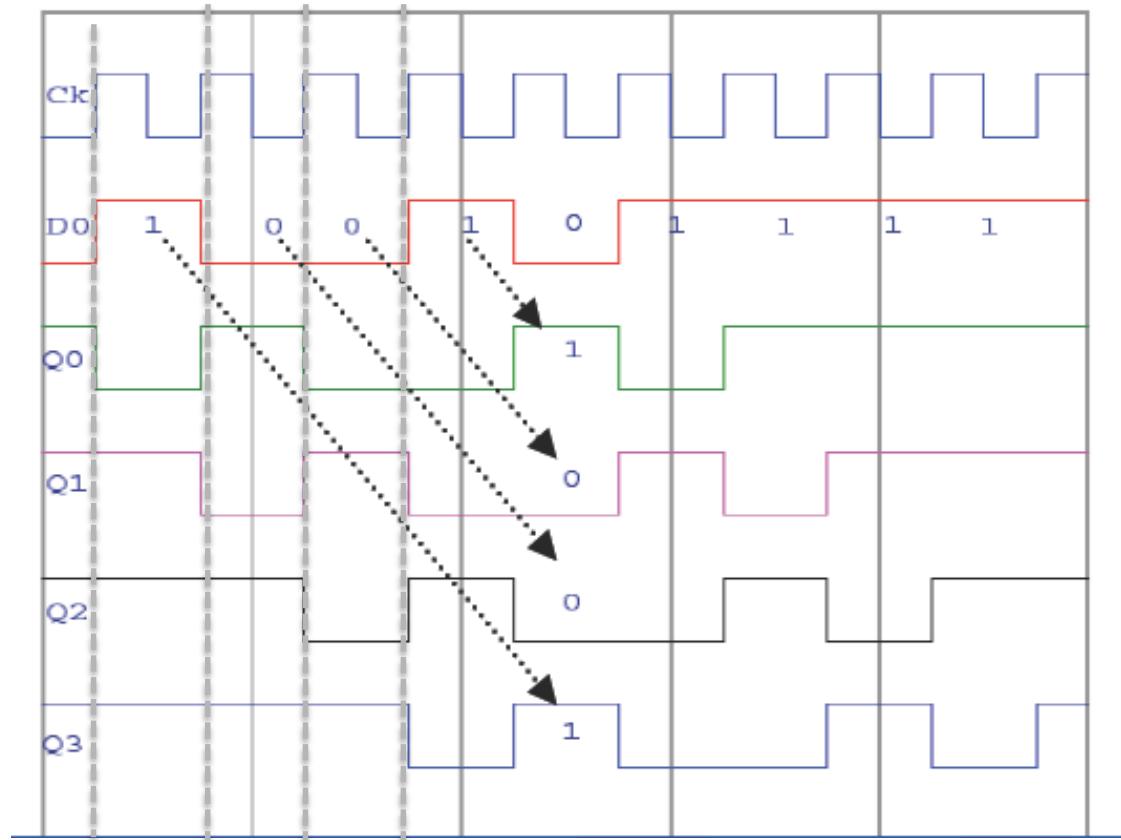
Dopo 4 impulsi di Clock ...



Shift Register con flip-flop D: funzionamento

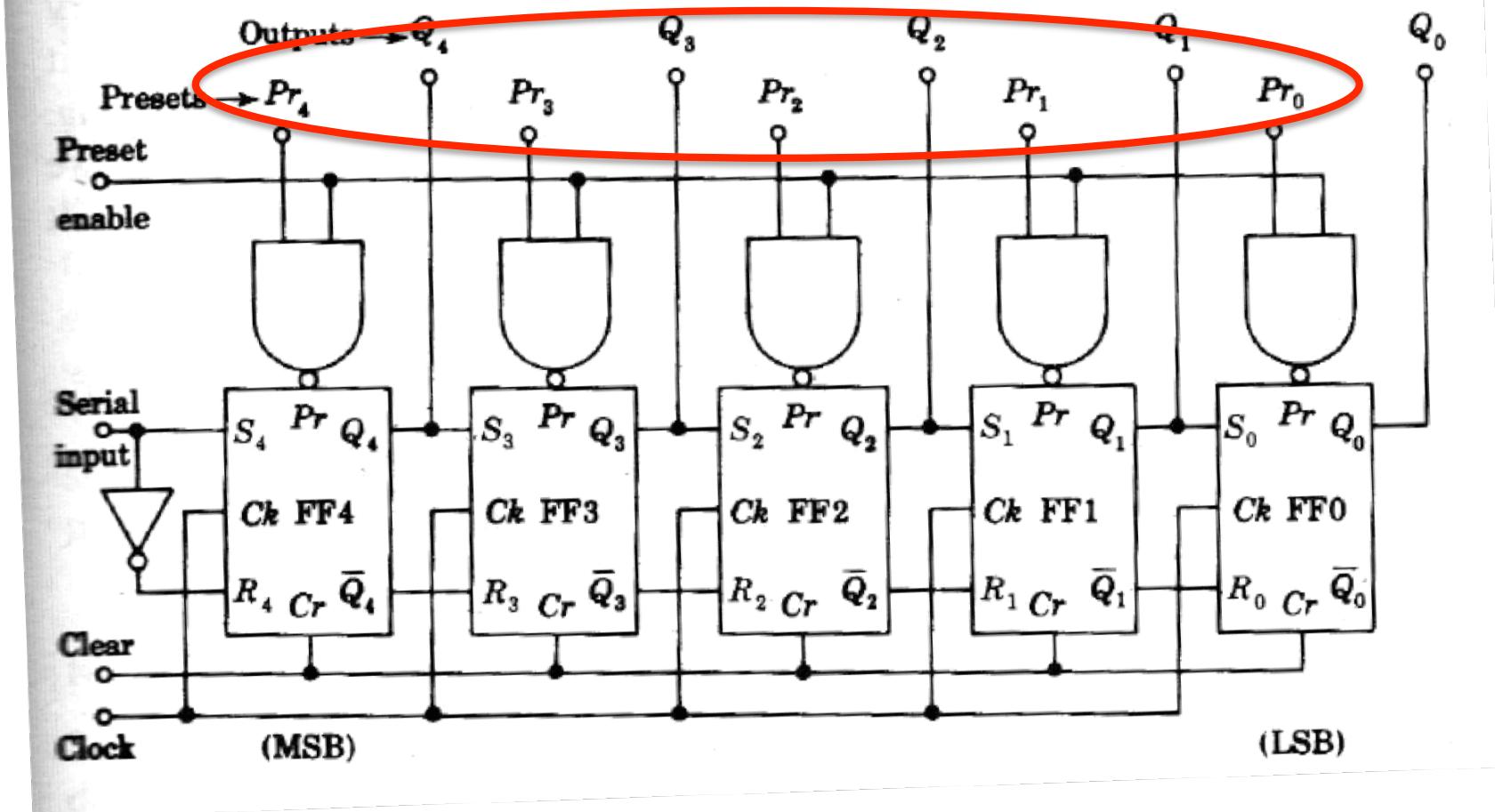
Supponiamo di avere un nro a 4 bit che viene trasmesso in serie sulla linea dei dati. Al primo fronte di salita del Clock il primo bit (b_0) viene trasmesso a Q_0 , al secondo fronte di salita b_0 viene trasmesso a Q_1 mentre b_1 viene trasmesso a Q_0 e così via. Dopo 4 fronti di salita avrò $Q_3=b_0$ $Q_2=b_1$ $Q_1=b_2$ $Q_0=b_4$.

Questo tipo di SR che ha un ingresso in serie ed un'uscita in parallelo si dice: SIPO = Serial In Parallel Output.



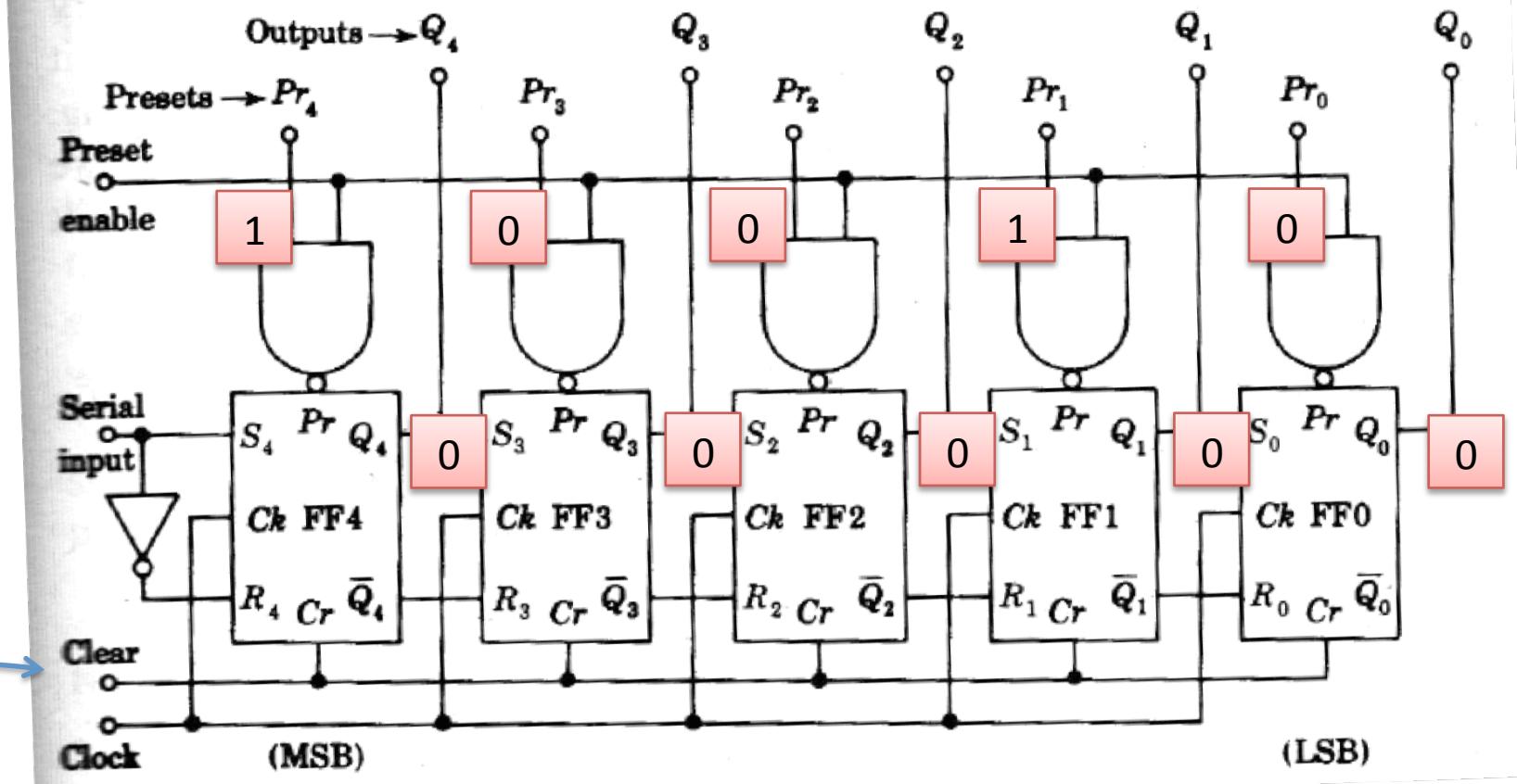
Registro a scorrimento universale: Parallel IN

Pr_i Definizione del nro a 5bit da immettere in modo parallelo



Registro a scorrimento universale: Parallel IN

Definizione del nro a 5bit da immettere in modo parallelo

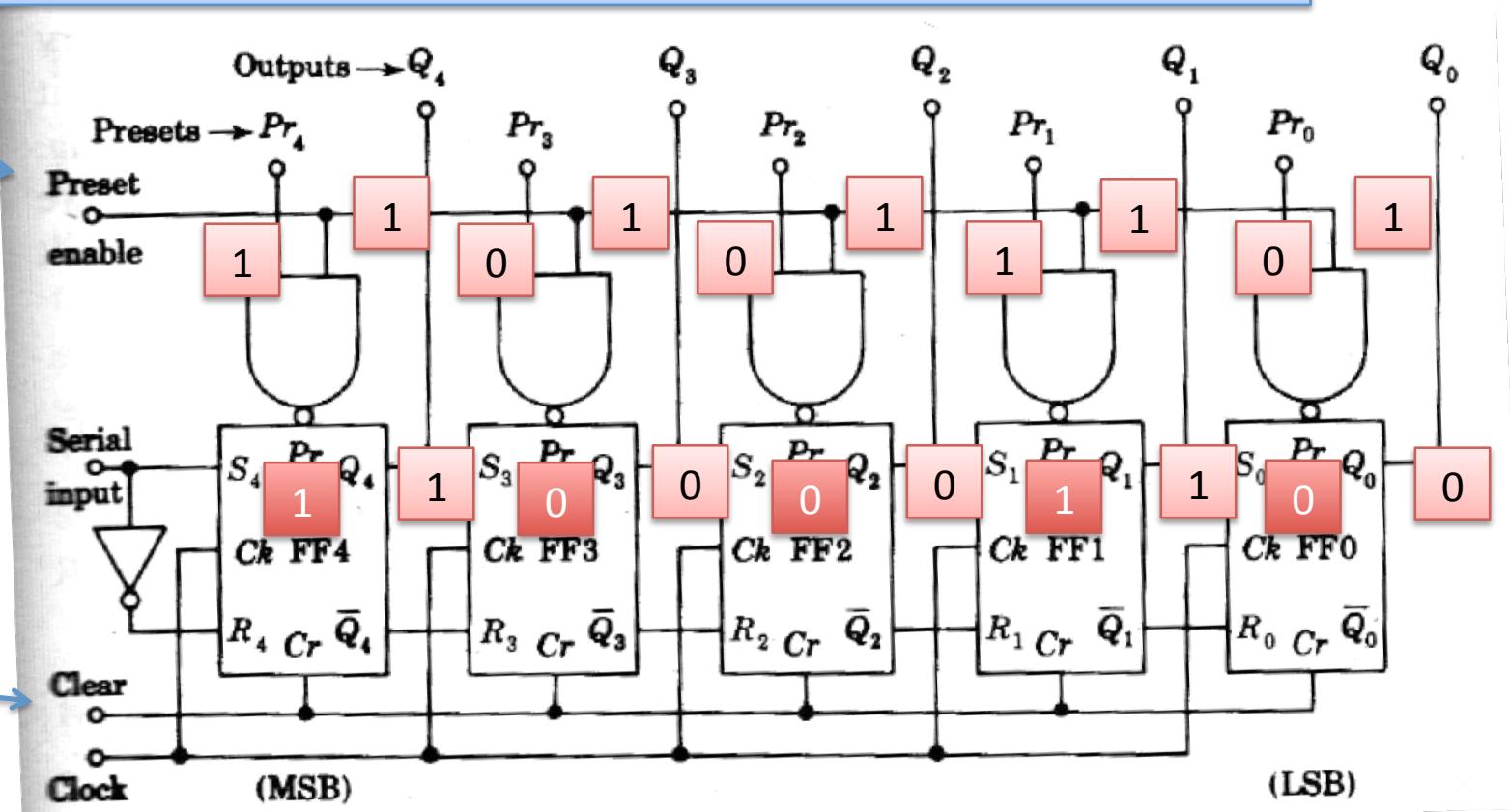


Prima operazione: CLEAR=0 -> Q_i=0

Seconda operazione: CLEAR=1 -> disabilitato

Registro a scorrimento universale: Parallel IN

Terza operazione: Preset Enable=0 -> le uscite delle porte NAND corrispondenti a bit=1 vanno basse e portano Q=1 per i corrispondenti FF

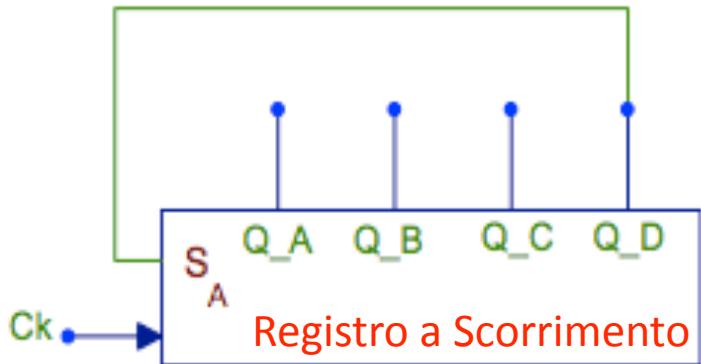


Prima operazione: CLEAR=0 -> Qi=0

Seconda operazione: CLEAR=1

Applicazioni shift registers

Generatore di Sequenze o Contatore ad Anello



impulsi di Ck	Q_A	Q_B	Q_C	Q_D
0	1	0	0	1
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

Generatore di sequenze

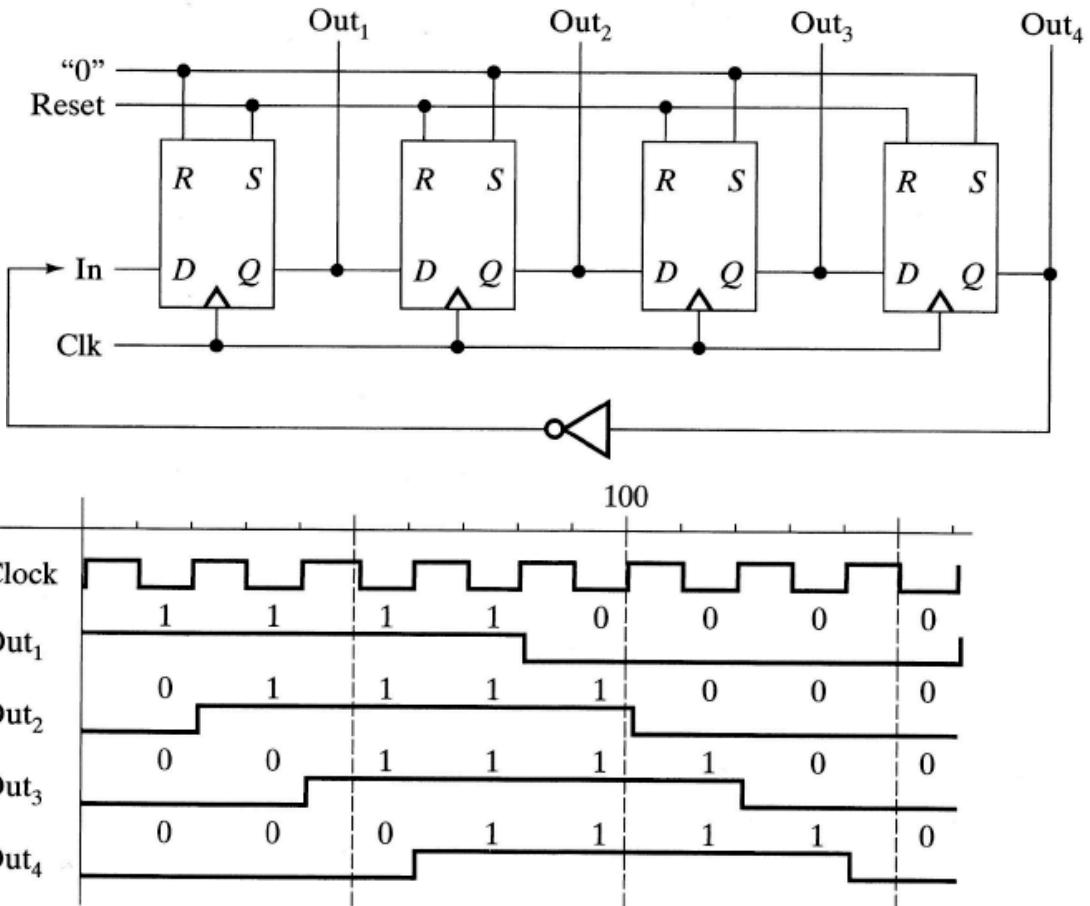
- Collego Q_D (ultimo FF della catena) con l'ingresso seriale (S_A)
- Utilizzando l'ingresso parallelo carico una qualsiasi sequenza (es. 1001) nel Registro e poi abilito il Clock che la fa scorrere nel registro
- Se carico la sequenza 0001 la sequenza che si genera con lo scorrimento puo` essere utilizzato in due modi:

- Contatore il Q che e` alto indica il conteggio (senza bisogno di decoding)
- Divisore di frequenza per il nro di FF presenti (N): 1 si presenta ogni 4 periodi di clock

impulsi di Ck	Q_A	Q_B	Q_C	Q_D
0	0	0	0	1
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

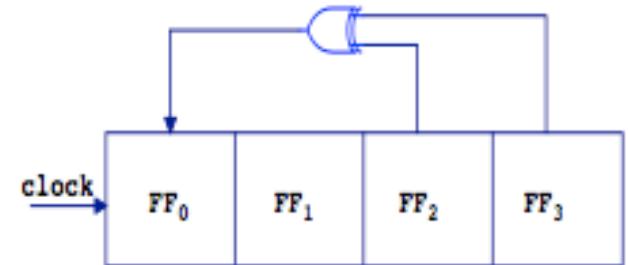
Twisted-ring Johnson counter

Se il bit piu` significativo e` collegato all'ingresso dello SR attraverso un NOT si ottiene un divisore per $2N$: twisted-ring, Johnson counter



Generatore di sequenze (pseudo)casuali

- Una applicazione degli SR che ha trovato larghissima diffusione e` quella che permette la generazione di sequenze di 1,0 con ordine casuale
- In effetti la larghezza finita del registro fa sì che la sequenza sia pseudocasuale
- In un registro di lunghezza k la sequenza si ripete dopo 2^k eventi
- La sequenza dei valori assunti da un qualsiasi FF e` pseudocasuale
- Un generatore di nri casuali a 4 bit di massima lunghezza si ottiene collegando le uscite dei FF2 e FF3 con uno XOR all'ingresso del FF0
- La successione di stati e` mostrata nella tabella delle verita`



clock	FF ₀	FF ₁	FF ₂	FF ₃
-	1	1	1	1
↑	0	1	1	1
↑	0	0	1	1
↑	0	0	0	1
↑	1	0	0	0
↑	0	1	0	0
↑	0	0	1	0
↑	1	0	0	1
↑	1	1	0	0
↑	0	1	1	0
↑	1	0	1	1
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	0	1
↑	1	1	1	0
↑	1	1	1	1

Generatore di sequenze (pseudo)casuali

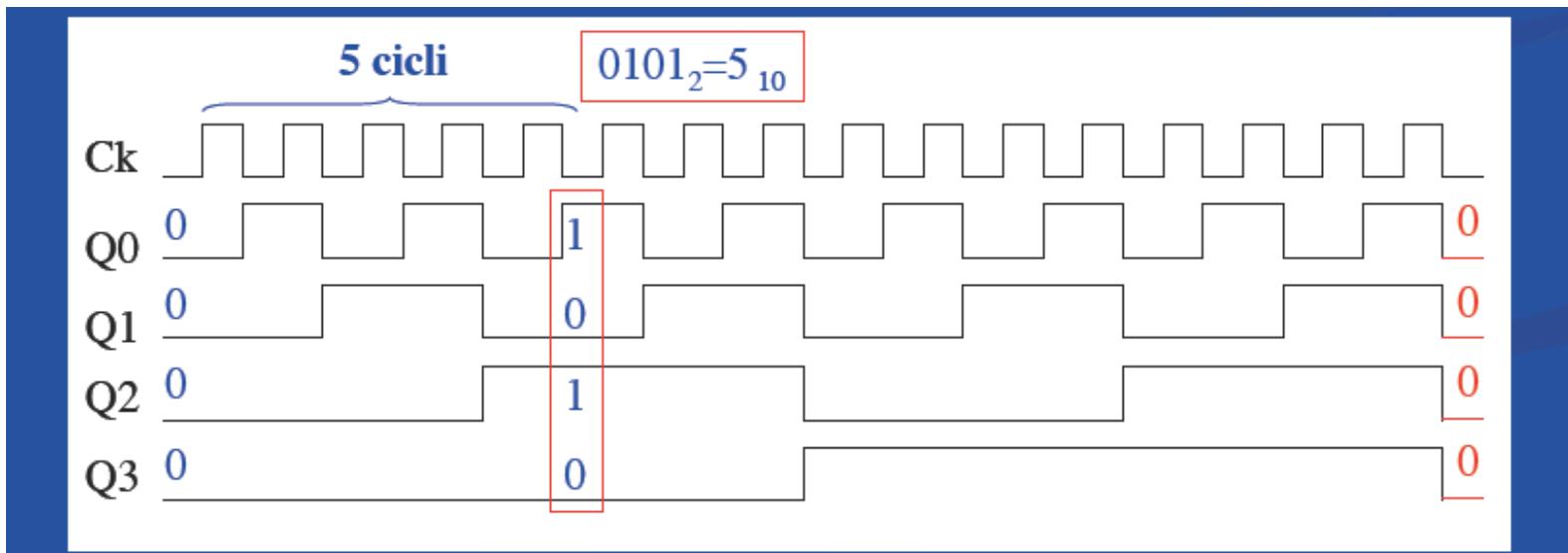
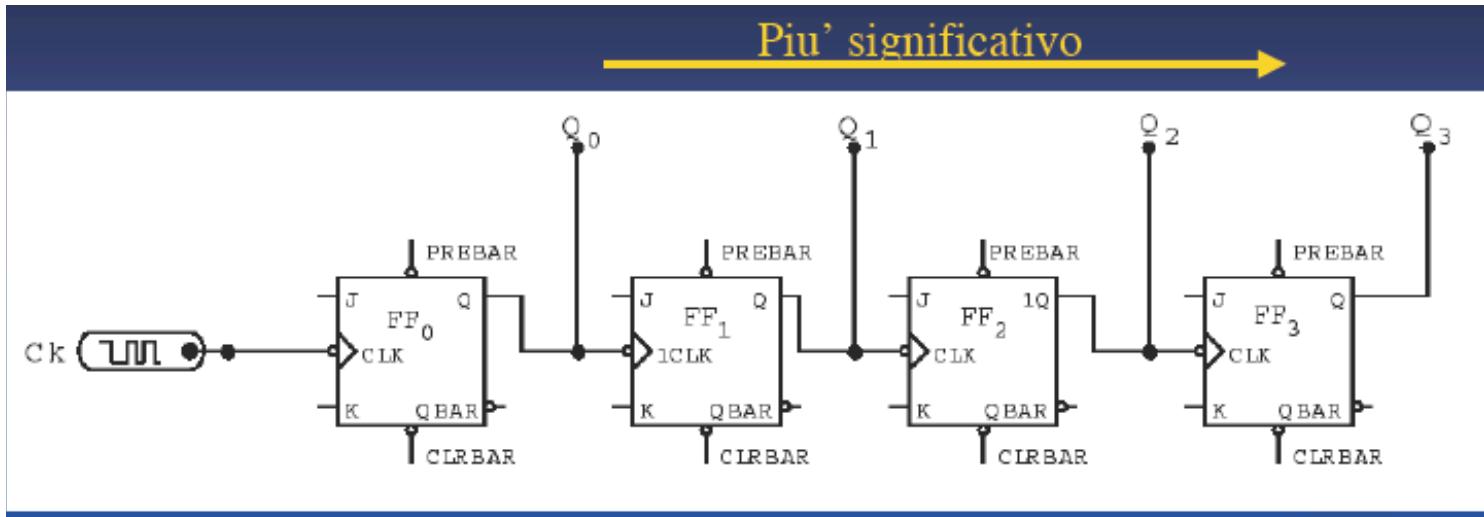
- E` possibile generare sequenze casuali anche collegano il FF in modo diverso tuttavia non si ha la certezza di ottenere la lunghezza massima della sequenza
- Esistono alcune tecniche che permetto di trovare quali porte collegare per essere sicuri di ottenere sequenza di massima lunghezza

k	lunghezza della sequenza $2^k - 1$	ingressi all'XOR
5	31	2,4
6	63	4,5
7	127	5,6
8	255	1,2,6,7
9	511	4,8
10	1023	6,9
11	2047	8,10
12	4095	1,9,10,11
13	8191	0,10,11,12
14	16383	1,11,12,13
15	32767	13,14
16	65535	3,12,14,15

Tabella 1.47: Generatori di sequenze pseudocasuali. La tabella fornisce, per tutti i registri di lunghezza inferiore a 17, la massima lunghezza della sequenza e gli ingressi all'XOR. Si noti che tali ingressi sono numerati a partire dallo zero. Gli ingressi indicati non sono gli unici possibili: ad esempio, nel caso di un registro di lunghezza 7, una sequenza di lunghezza massima si ottiene anche adoperando come ingressi all'XOR le uscite 4 e 7.

Contatori

Contatore binario



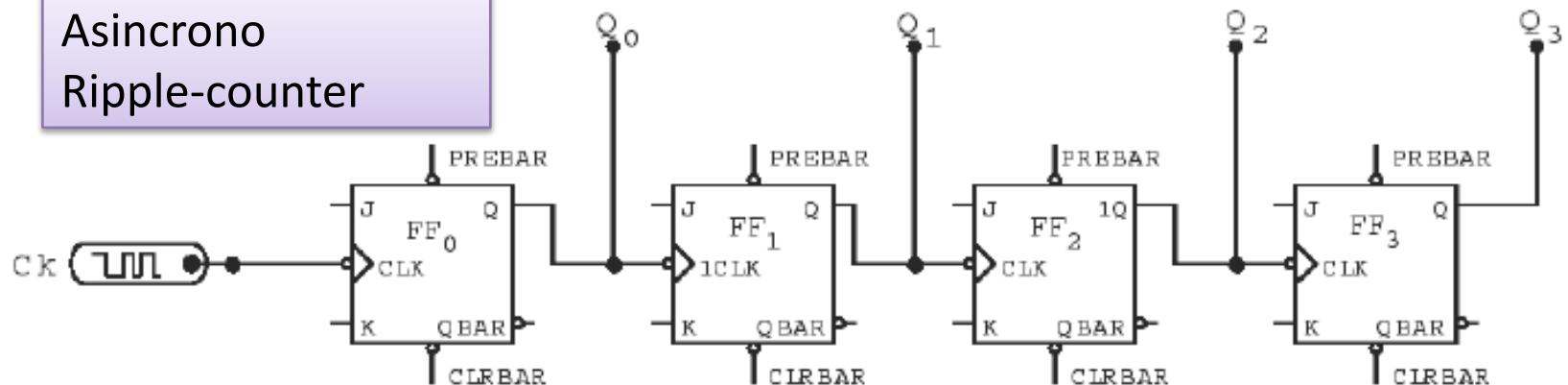
Contatori asincroni/sincroni

- I contatori che abbiamo visto fino ad ora sono detti ASINCRONI poiche` la transizione dei FF avviene in cascata: la transizione del FF_N deve aspettare la transizione del FF_{N-1}
- Questo fa sì che il ritardo di propagazione del segnale di Clock (fronte HL o LH) a tutte le uscite commutate sia nel caso peggiore (tutti i FF=1) $N\Delta t$ dove Δt è il tempo di propagazione di un singolo FF e N è il nro totale di FF.
- Teniamo presente che abbiamo visto che il tempo di propagazione di un singolo FF è $\approx 20-30$ ns
- **Contatori SINCRONI:** sono implementati modo che il segnale di Clock arrivi a tutti i FF contemporaneamente -> più veloce

Contatori asincroni/sincroni

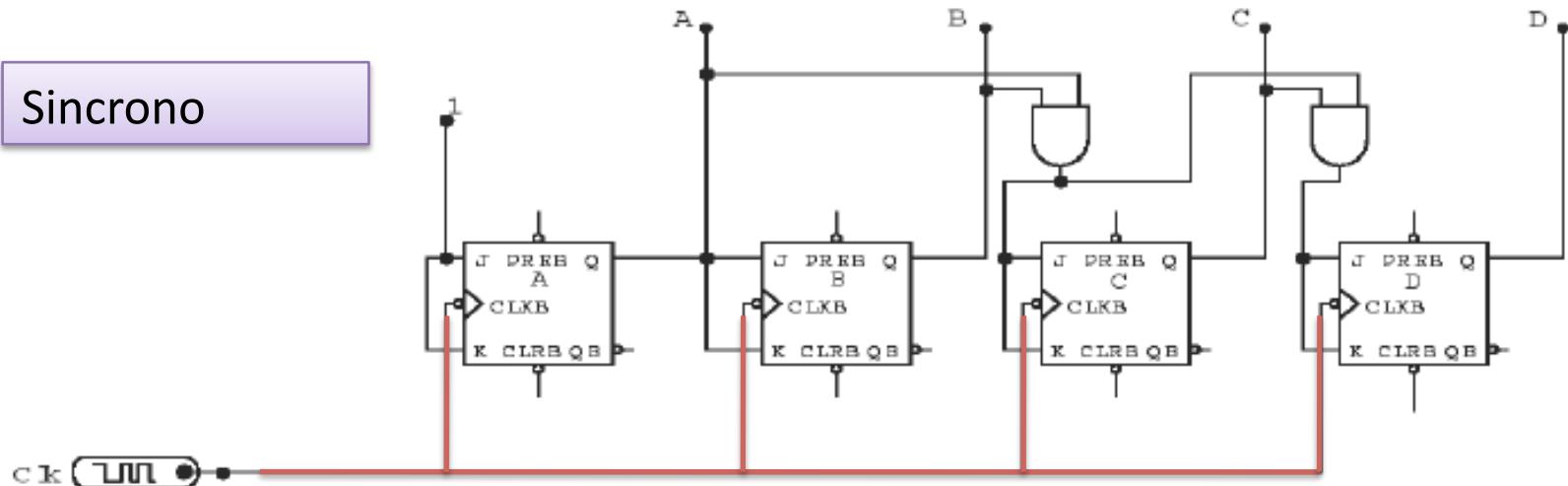
Asincrono

Ripple-counter



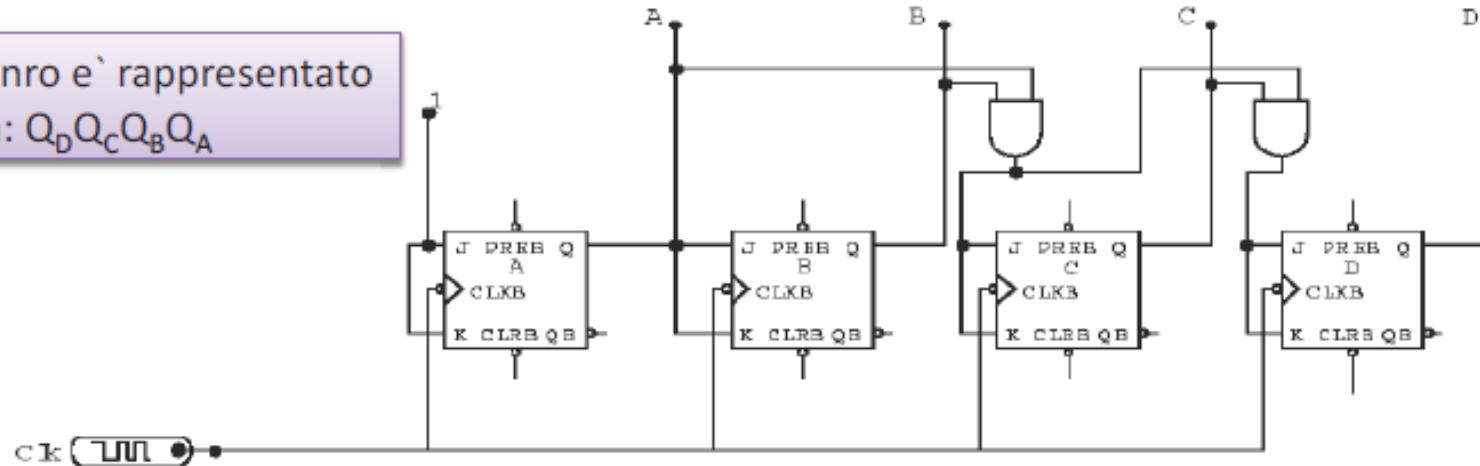
Ripple = increspatura, onda

Sincrono



Contatore sincrono con carry in serie

Il nro e` rappresentato da: $Q_D Q_C Q_B Q_A$



Nro clock	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0

FFB commuta solo se $Q_A=1 \rightarrow JK_B=Q_A$

FFC commuta solo se $Q_A=Q_B=1 \rightarrow JK_C=Q_A Q_B$

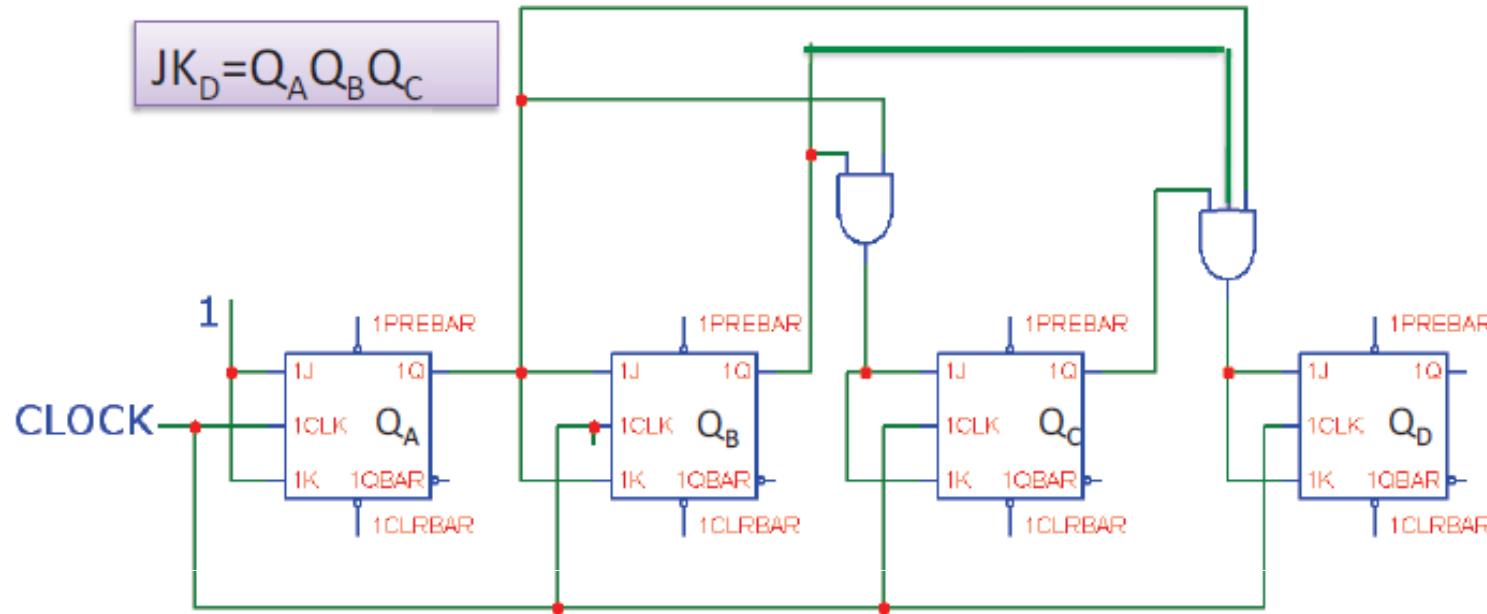
FFD commuta solo se $Q_C=Q_A=Q_B=1$
 $\rightarrow JK_D=Q_A Q_B Q_C = JK_C Q_C$

Il ritardo massimo rispetto al fronte del segnale di clock del tempo in cui è stabile il valore su tutti gli ingressi JK dei FF è:

$$\Delta t_{max} = \Delta t_{FF} + 2 \times \Delta t_{AND} \quad \text{e quindi}$$

$$f_{clock_max} = 1 / (\Delta t_{FF} + 2 \times \Delta t_{AND})$$

Contatore sincrono con carry parallelo



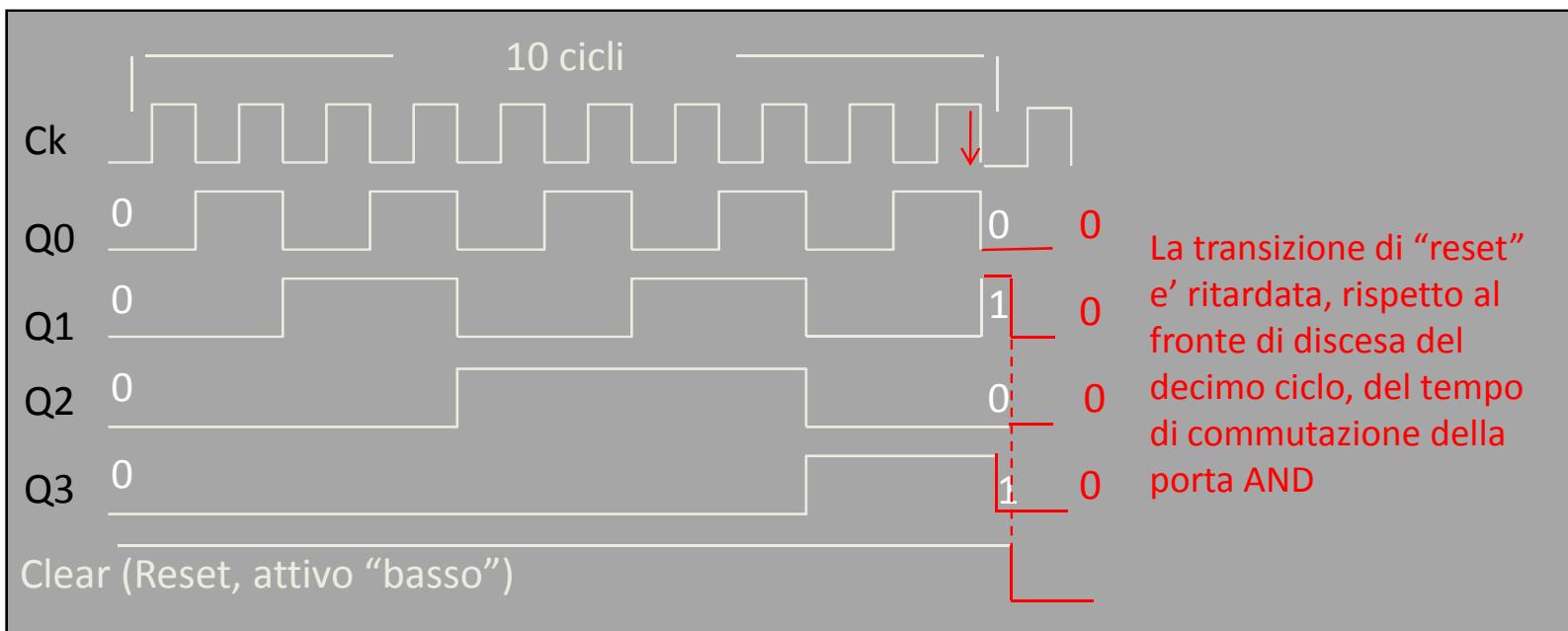
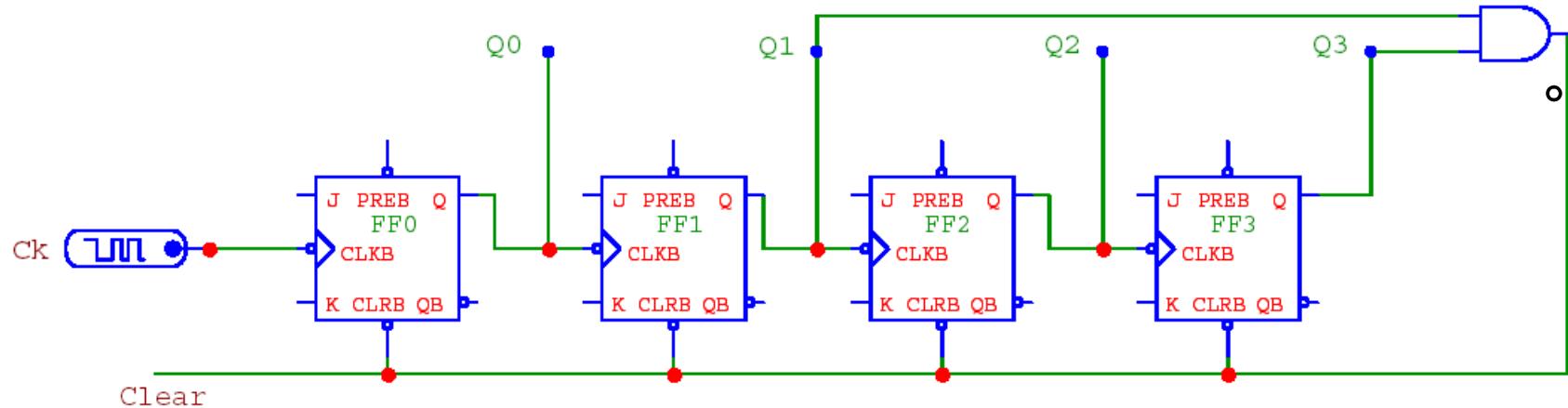
Il ritardo massimo rispetto al fronte del segnale di clock del tempo in cui è stabile il valore su tutti gli ingressi JK dei FF è: $\Delta t_{max} = \Delta t_{FF} + \Delta t_{AND}$ e quindi

$$f_{clock_max}(\text{carry paral.}) = 1 / (\Delta t_{FF} + \Delta t_{AND}) < f_{clock_max}(\text{carry serie}) = 1 / (\Delta t_{FF} + 2 \times \Delta t_{AND})$$

Problemi con il carry parallelo:

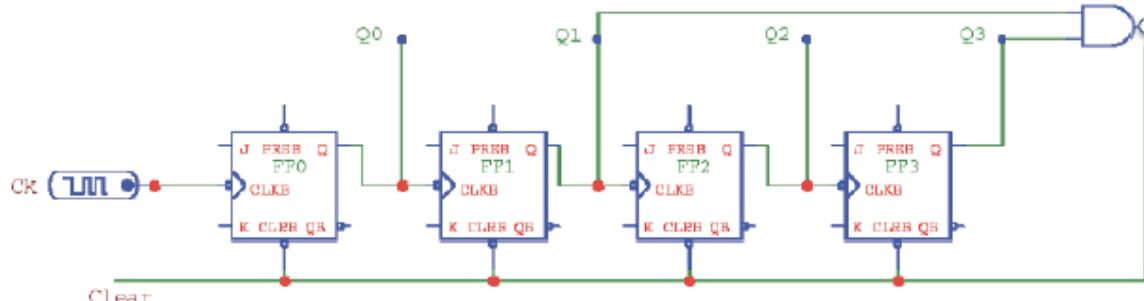
- il nro di ingressi richiesti nelle porte che guidano gli ingressi JK
- fan-out molto alto che si richiede ai FF, FFO deve mandare il segnale in ingresso a tutte le porte che guidano JK.

Contatore decadico/BCD

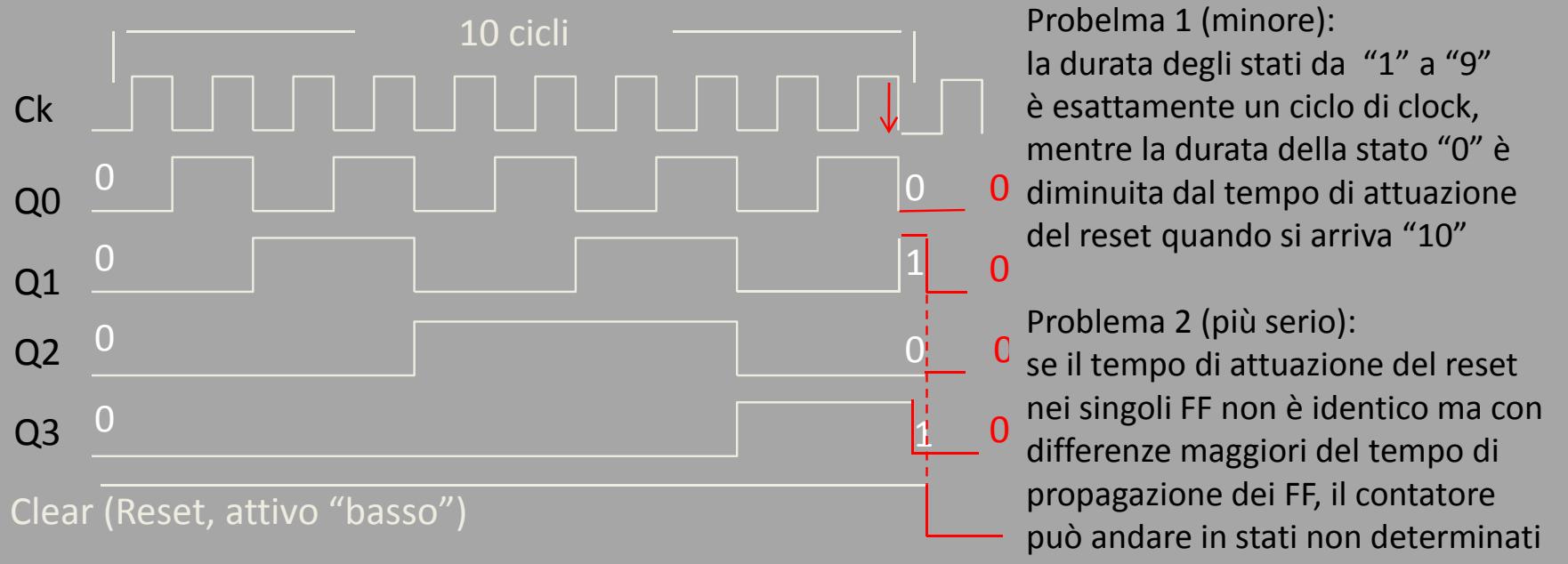


Problemi di temporizzazione con il contatore decadico

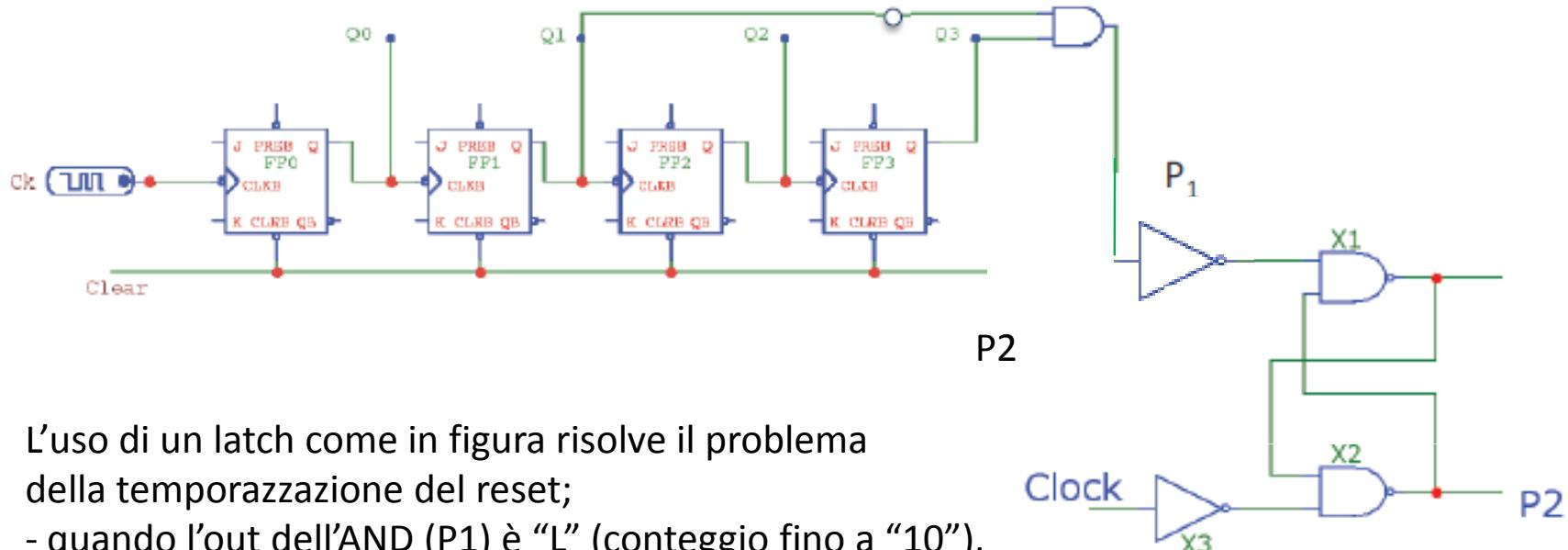
(e, in generale, dei contatori azzerati con reazione sul reset)



La durata di ogni conteggio è = 1 periodo di CLOCK. Vediamo cosa succede sul “clear”:



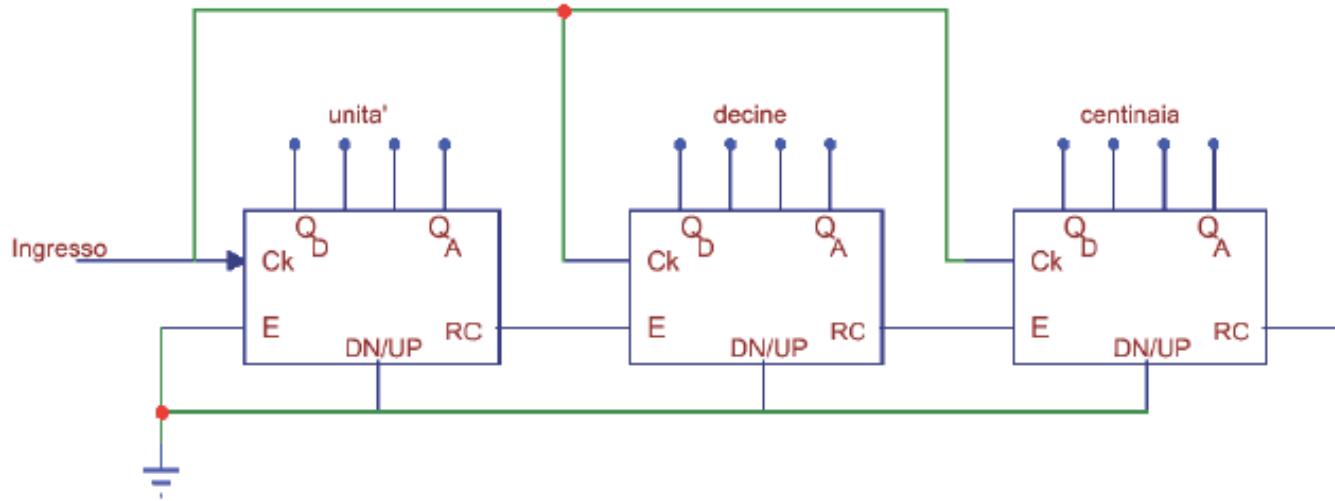
Contatore decadico migliore: una possibile soluzione



L'uso di un latch come in figura risolve il problema della temporizzazione del reset;

- quando l'out dell'AND (P1) è "L" (conteggio fino a "10"), entrambi gli ingessi di X1 sono "H1" (P2 deve manterenere inattivo il "Clear"; l'uscita di X1 è "L" e quella di X2 "H"; il latch mantiene la configurazione).
- quando il contatore raggiunge il "10", l'uscita di X1 commuta in "H", ma quella di X2 commuta in "L" (clear attivo) solo quando NOT(clock) diventa "H", ovvero un semi-periodo dopo la commutazione dei FF; dunque il segnale di reset arriva quando i FF non possono fare una transizione

Contatori sincroni in serie



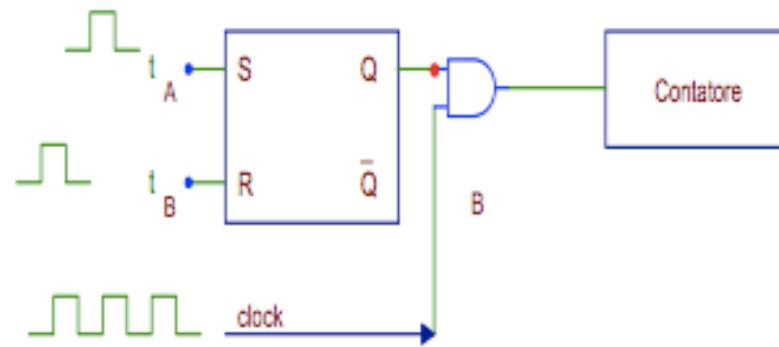
E=Enable, attivo basso
DN/UP= conteggio:
0-UP 1-DOWN
RC=Segnale di Overflow

- Mettendo in serie piu` contatori posso aumentare l'intervallo di conteggio
- Es.: mettendo in serie 3 contatori decadici ottengo un contatore modulo 1000
- E' necessario in questo caso avere un segnale di OVERFLOW (superamento nro massimo, in questo caso 9). Nell'esempio mostrato RC da` un segnale basso quando ho la transizione 9->0 che quindi abilita il FF successivo a contare per un conteggio... finche` non si verifica il prossimo Overflow.

Applicazioni dei contatori

- Misurazione dell'intervallo temporale tra due eventi

t_A abilita la porta AND facendo partire il contatore
 t_B disabilita la porta AND fermando il contatore
Il nro di conteggi \times Tclock =
Tempo (ES.: celle fotelettriche – misurazione velocità)



Se voglio misurare la distanza temporale tra due segnali generati dallo stesso apparato
-> stessa linea: FF di tipo JK funzione Toggle.

Primo impulso abilita la porta AND e fa partire il contatore
Il secondo impulso lo ferma.

