ARCHITETTURA DEGLI ELABORATORI

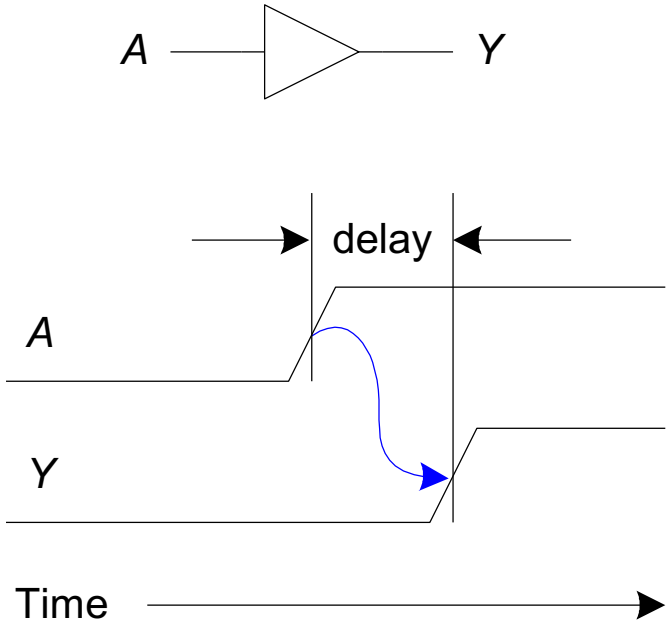
A.A. 2020-2021

Università di Napoli Federico II
Corso di Laurea in Informatica

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

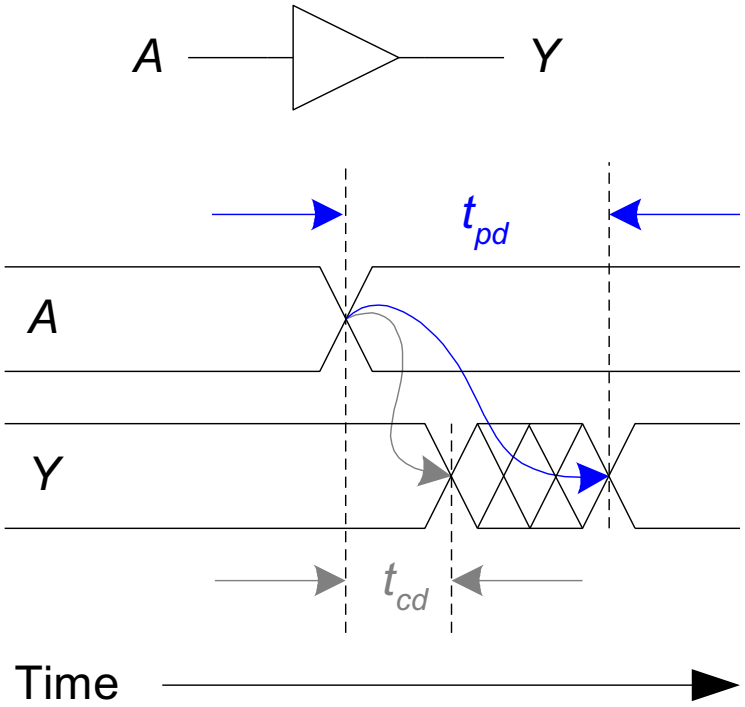
Timing

- **Delay:** time between input change and output changing
- How to build fast circuits?



Propagation & Contamination Delay

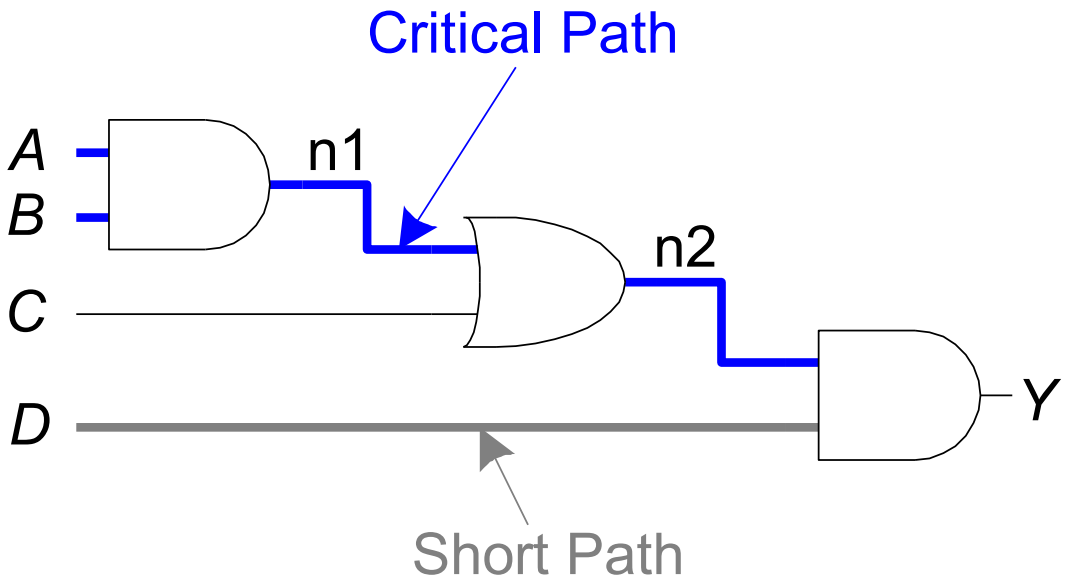
- **Propagation delay:** t_{pd} = max delay from input to output
- **Contamination delay:** t_{cd} = min delay from input to output



Propagation & Contamination Delay

- Delay is caused by
  - Capacitance and resistance in a circuit
  - Speed of light limitation
- Reasons why t_{pd} and t_{cd} may be different:
  - Different rising and falling delays
  - Multiple inputs and outputs, some of which are faster than others
  - Circuits slow down when hot and speed up when cold

Critical (Long) & Short Paths



Critical (Long) Path: $t_{pd} = 2t_{pd\_\mathrm{AND}} + t_{pd\_\mathrm{OR}}$

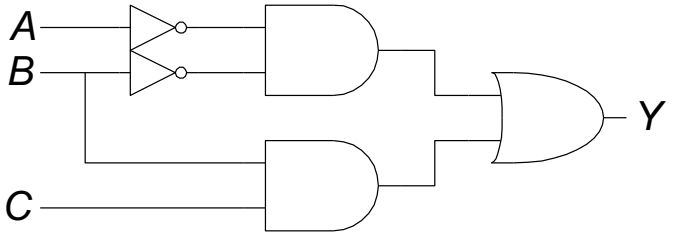
Short Path: $t_{cd} = t_{cd\_\mathrm{AND}}$

Glitches

- When a single input change causes an output to change multiple times

Glitch Example

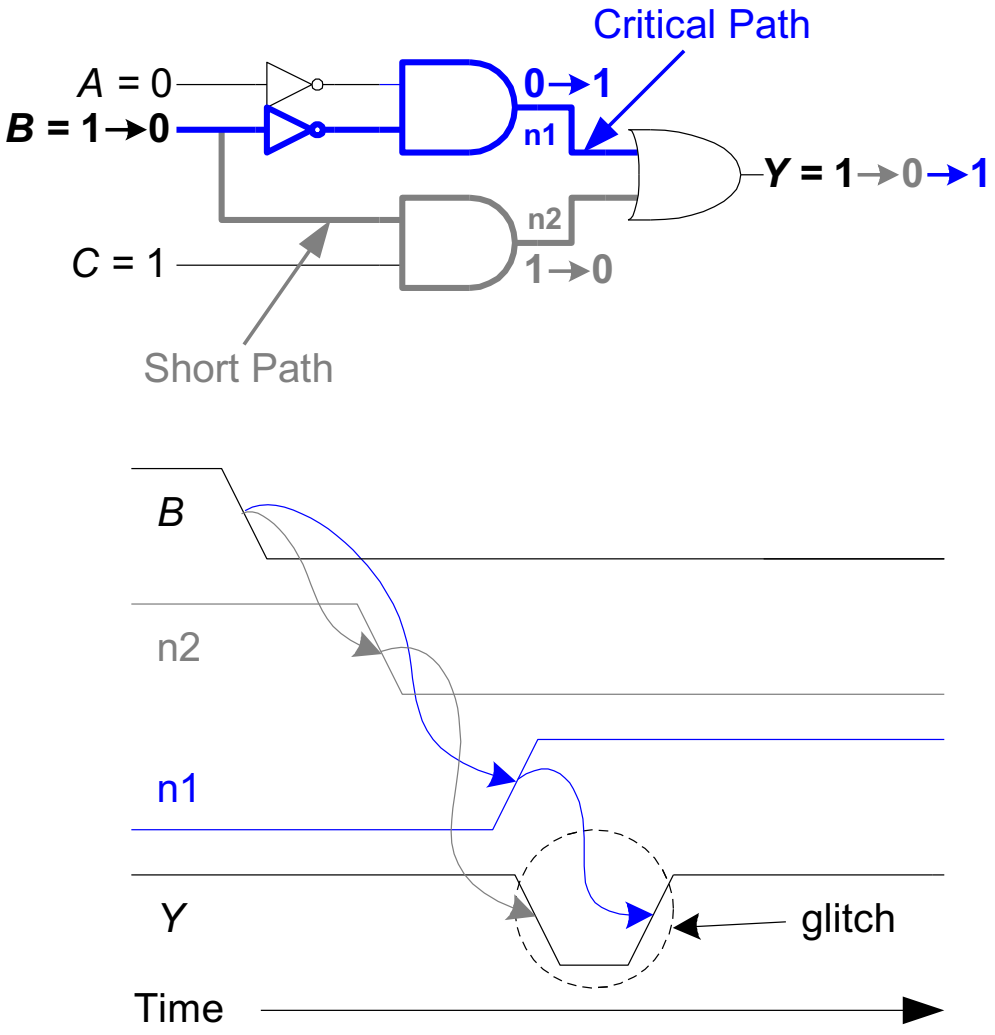
- What happens when A = 0, C = 1, B falls?



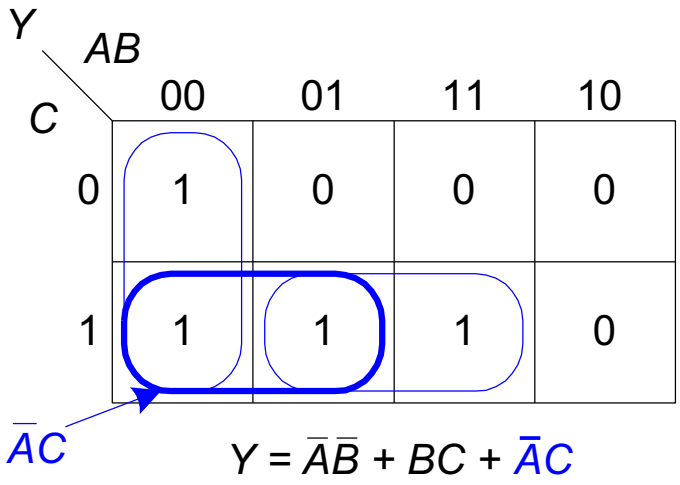
Y	AB = 00	AB = 01	AB = 11	AB = 10
C = 0	1	0	0	0
C = 1	1	1	1	0

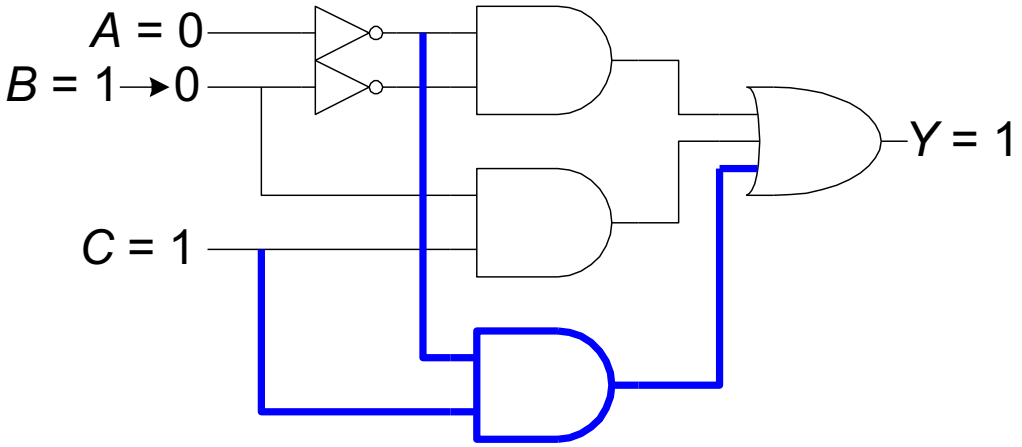
$Y = \bar{A}\bar{B} + BC$

Glitch Example (cont.)



Fixing the Glitch



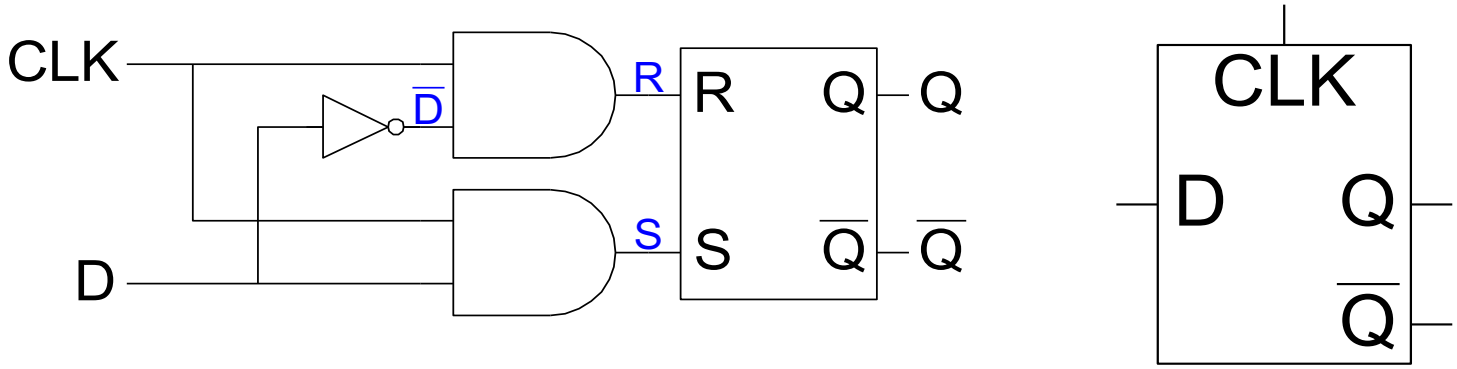


Why Understand Glitches?

- Glitches don't cause problems because of **synchronous design** conventions (see Chapter 3)
- It's important to **recognize** a glitch: in simulations or on oscilloscope
- Can't get rid of all glitches – simultaneous transitions on multiple inputs can also cause glitches

CIRCUITI SEQUENZIALI

D Latch Internal Circuit

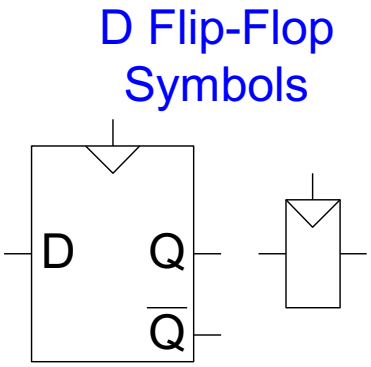


CLK	*D*	$\overline{D}$	*S*	*R*	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

D Flip-Flop

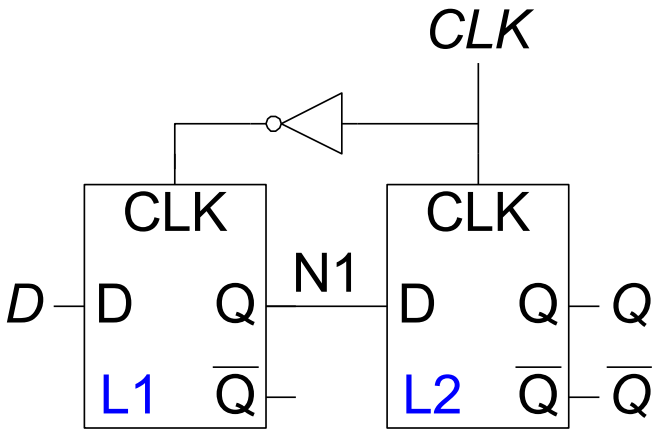
- Inputs: *CLK*, *D*
- Funzione:
  - Quando *CLK* passa da 0 a 1, *D* passa fino a *Q*
  - Altrimenti, *Q* mantiene il suo valore precedente
- *Q* cambia solo durante la transizione di *CLK* da 0 a 1

Queste tipologie di componenti sono dette edge-triggered perché sono pilotate non da un valore ma da una transizione (di CLK)



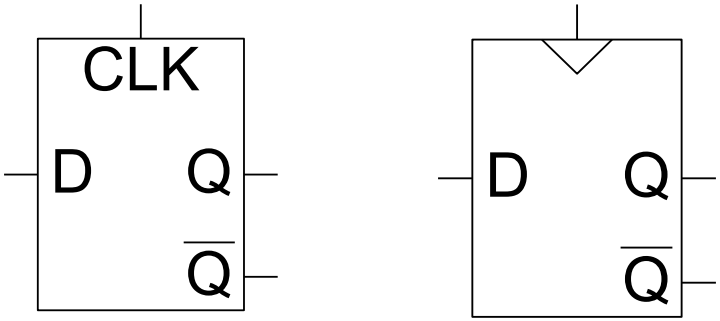
D Flip-Flop

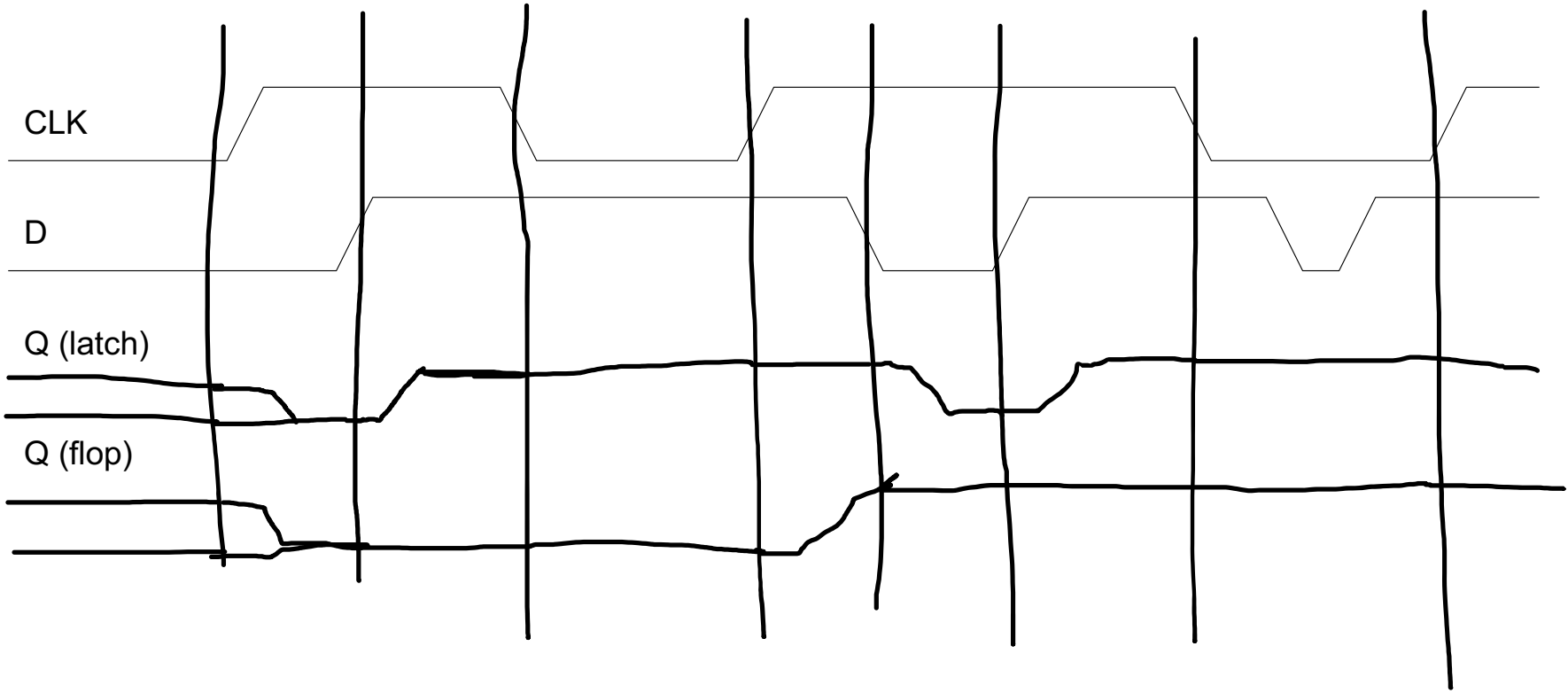
- 2 D latch (L1 e L2) controllati da clock complementari
- Quando ***CLK* = 0**
  - L1 è trasparente
  - L2 è opaco
  - *D* passa fino a N1
- Quando ***CLK* = 1**
  - L2 è trasparente
  - L1 è opaco
  - N1 passa fino a *Q*



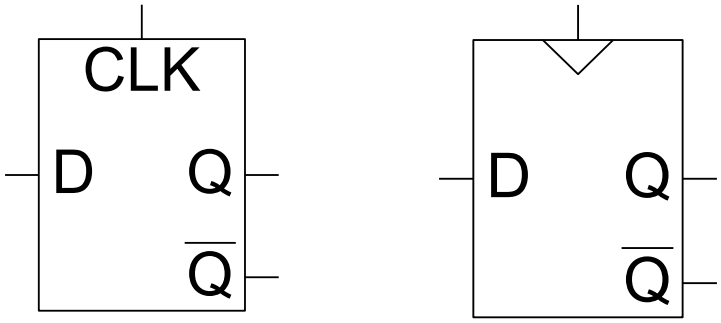
- Quindi, D passa fino a Q *sulla transizione di CLK da 0 a 1*
- Ulteriori variazioni di D quando CLK=1 (risp. CLK=0) non passano a Q perché L1 (risp. L2) è opaco

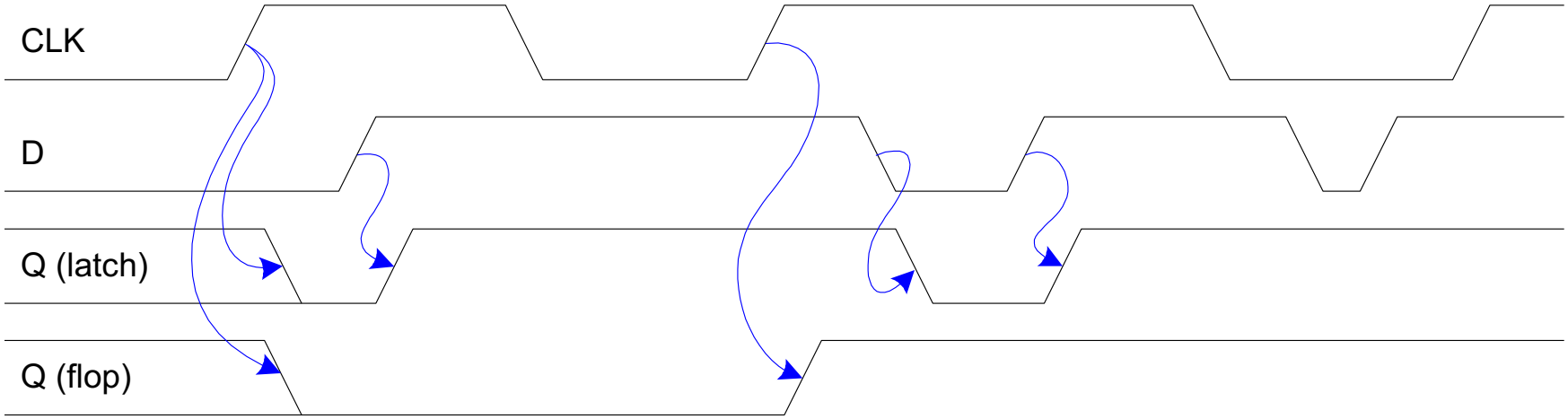
D Latch vs. D Flip-Flop



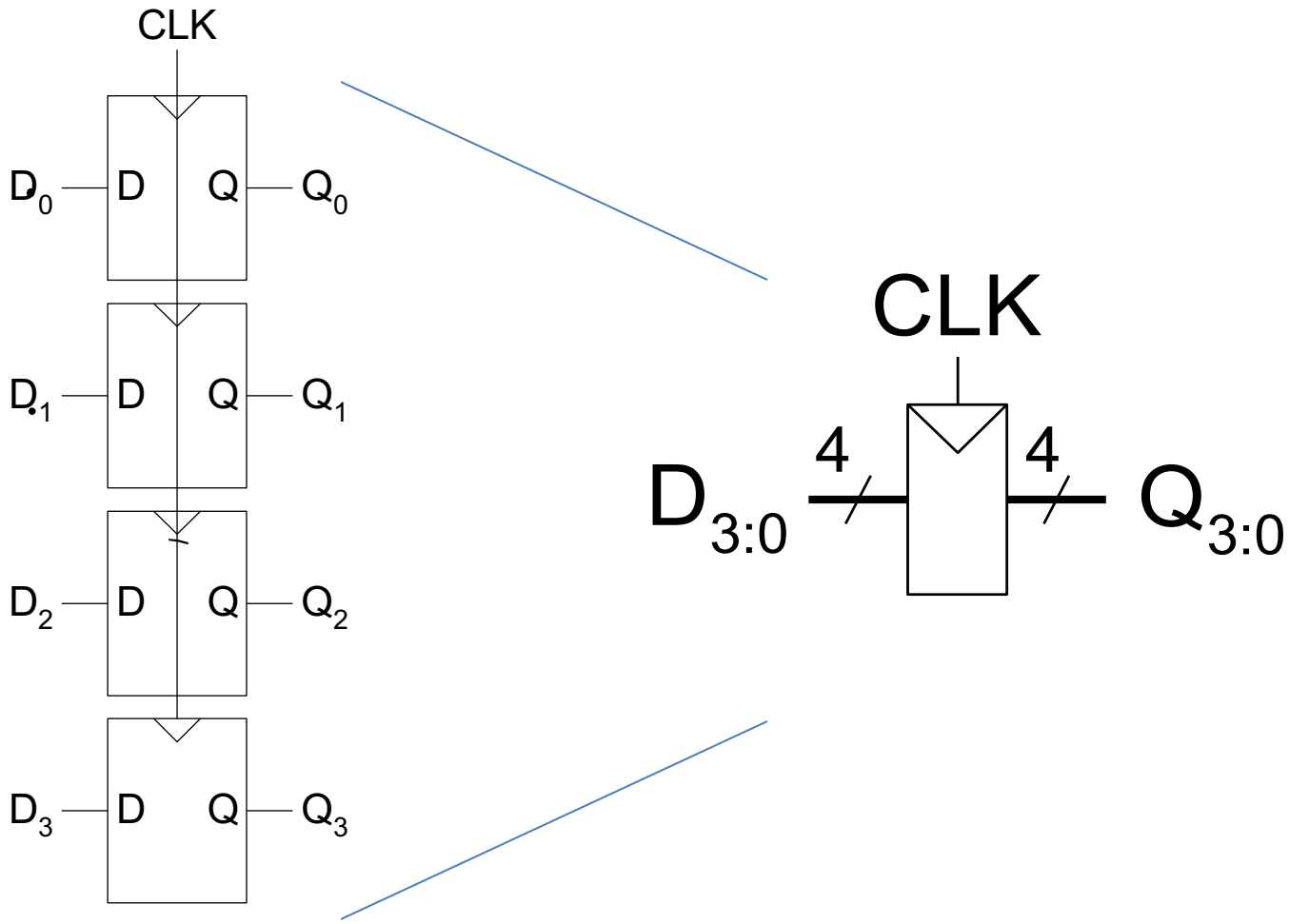


D Latch vs. D Flip-Flop



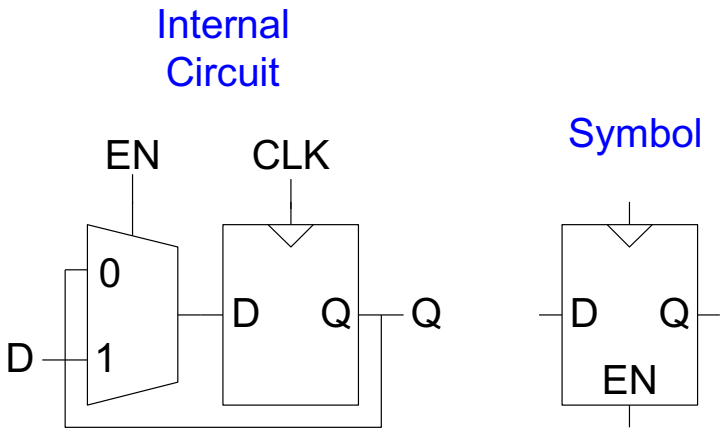


Registri: Multi-bit Flip-Flop



Flip-Flops “enabled”

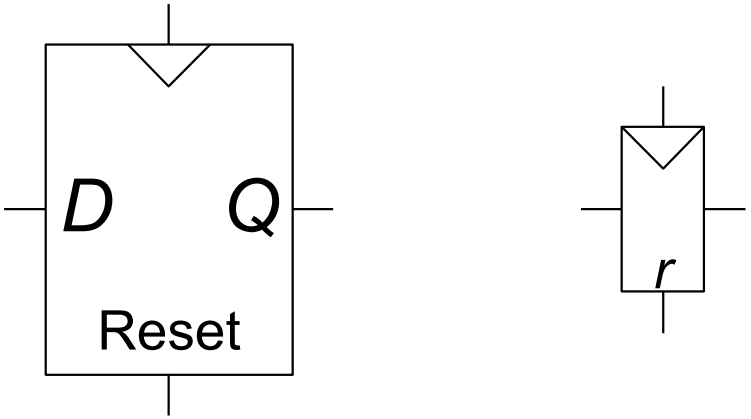
- *Inputs: CLK, D, EN*
- L’input enable (*EN*) stabilisce quando un nuovo valore di *D* è memorizzato
- ***EN* = 1:** *D* passa fino a *Q* (clock: 0→1)
- ***EN* = 0:** il flip-flop mantiene il suo stato precedente



Flip-Flops “resettabili”

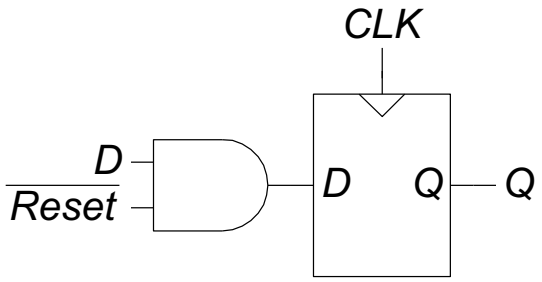
- *Inputs: CLK, D, Reset*
- ***Reset* = 1:** $Q = 0$
- ***Reset* = 0:** il flip-flop si comporta “normalmente” come un D flip-flop

Symbols



Flip-Flops “resettabili”

- Vi sono due tipi di flip-flop resettabili:
  - **Sincroni:** il reset è pilotato dal clock
  - **Asincroni:** il reset avviene non appena *Reset* = 1
- Flip-flop sincroni:

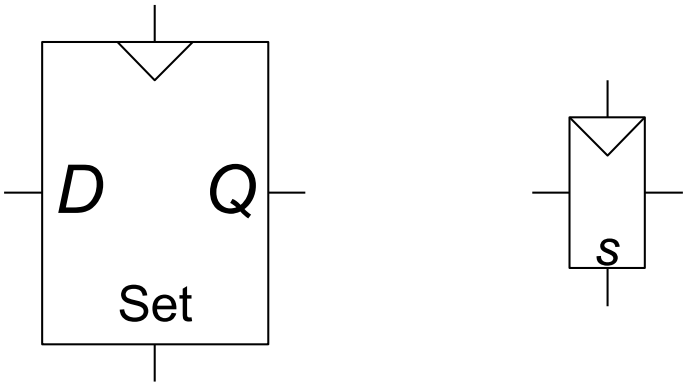


- Per i flip-flop asincroni occorre modificare il circuito interno del flip-flop

Flip-Flops “settabili”

- *Inputs: CLK, D, Set*
- ***Set* = 1:** *Q*=1
- ***Set* = 0:** il flip-flop si comporta “normalmente” come un D flip-flop

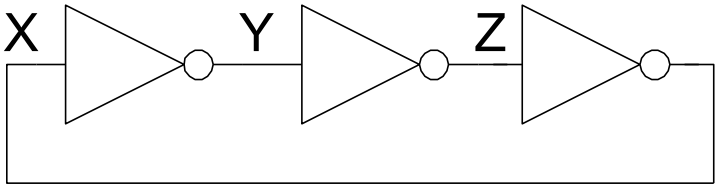
Symbols

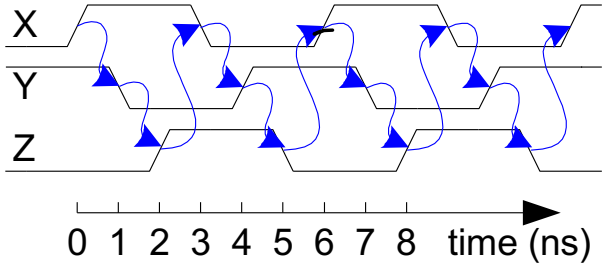


Esercizi

- Esercizi 3.1, 3.3, 3.5, 3.7, 3.8, 3.13, 3.15

Criticità nella logica sequenziale

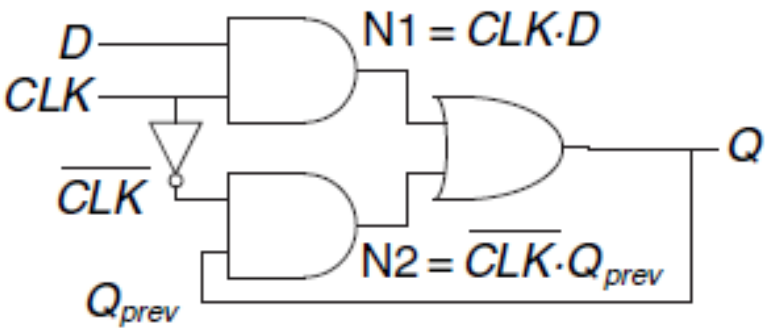




- Questi circuiti vengono detti “astabili” poiché hanno un comportamento oscillante
- Il periodo di oscillazione dipende dai ritardi degli inverter
- Idealmente è di 6 ns tuttavia può variare a causa di diversi fattori
  - differenze nella manifattura
  - temperatura
- Circuito *asincrono*: l’output è retroazionato in maniera diretta

Criticità nella logica sequenziale

- In casi più complessi che comprendono l'uso di più porte AND, NOT, OR il comportamento di una rete asincrona può dipendere fortemente dai ritardi accumulati sui singoli cammini

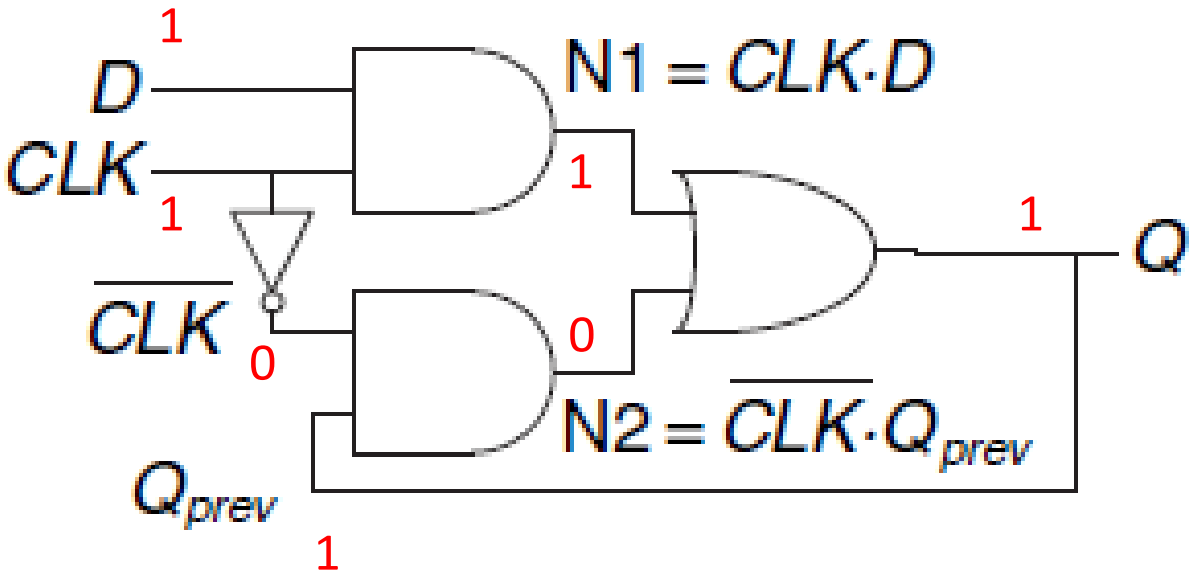


$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$

CLK	D	Q_{prev}	Q
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

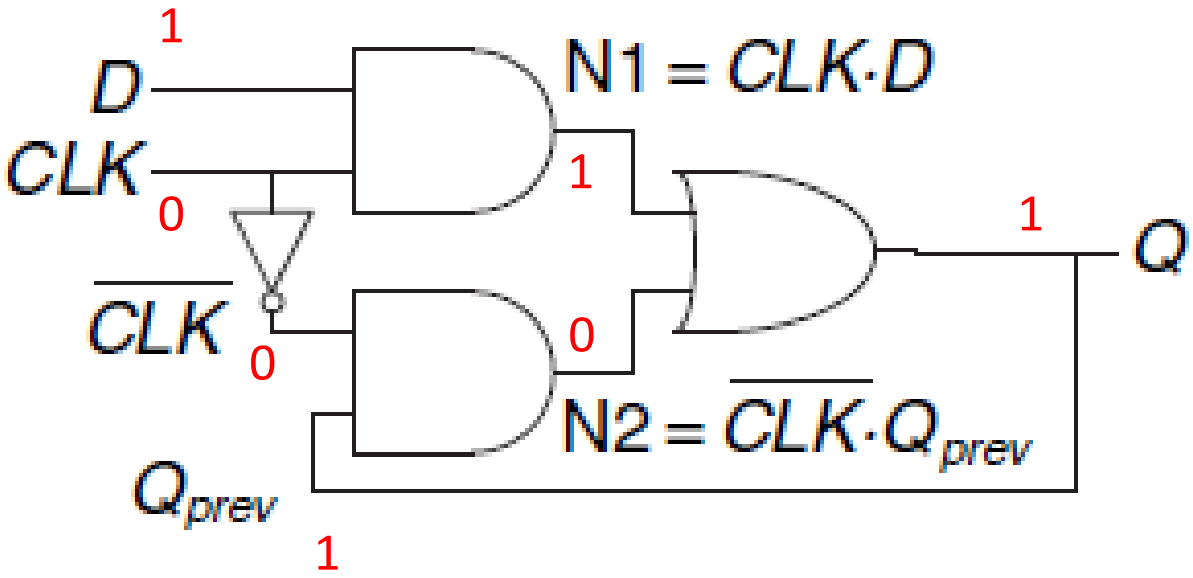
Criticità nella logica sequenziale

- D=1, CLK=1 ➡ Q=1



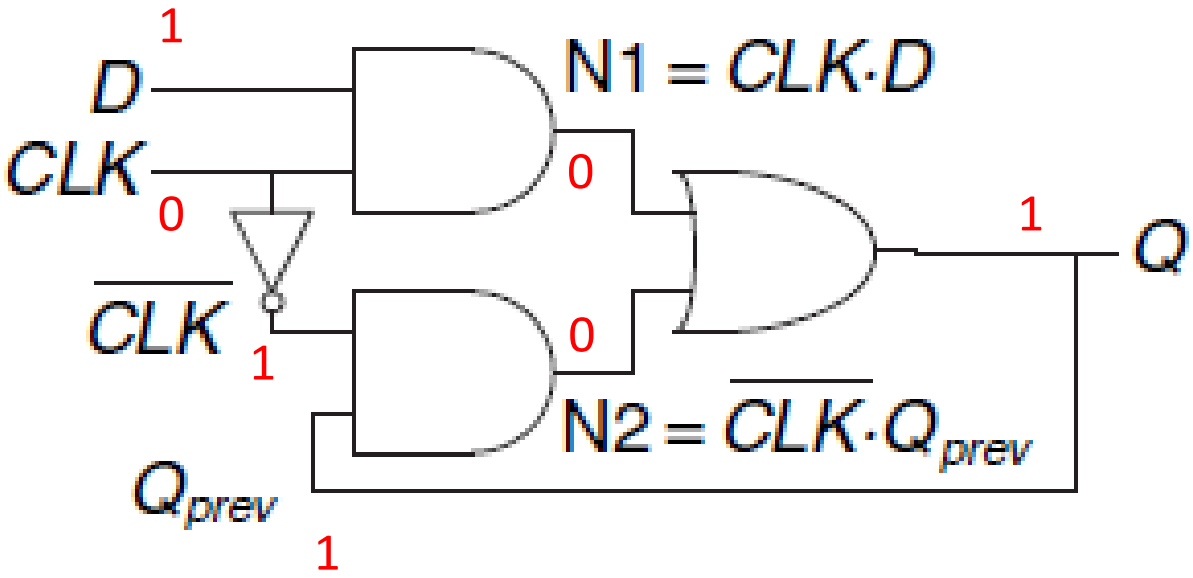
Criticità nella logica sequenziale

t_0 CLK 1→0



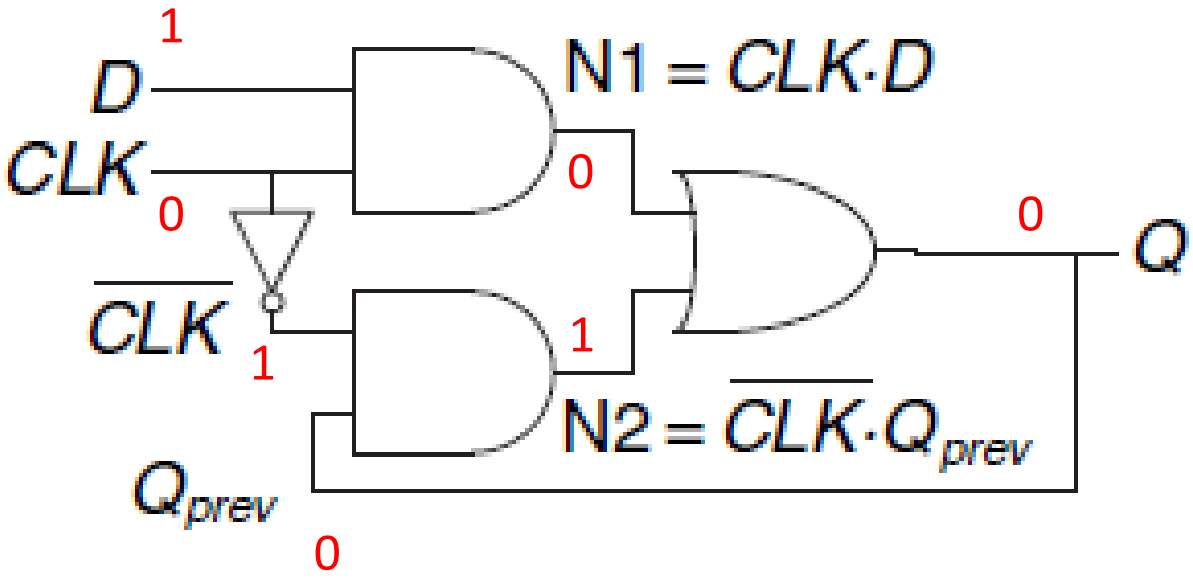
Criticità nella logica sequenziale

t_1



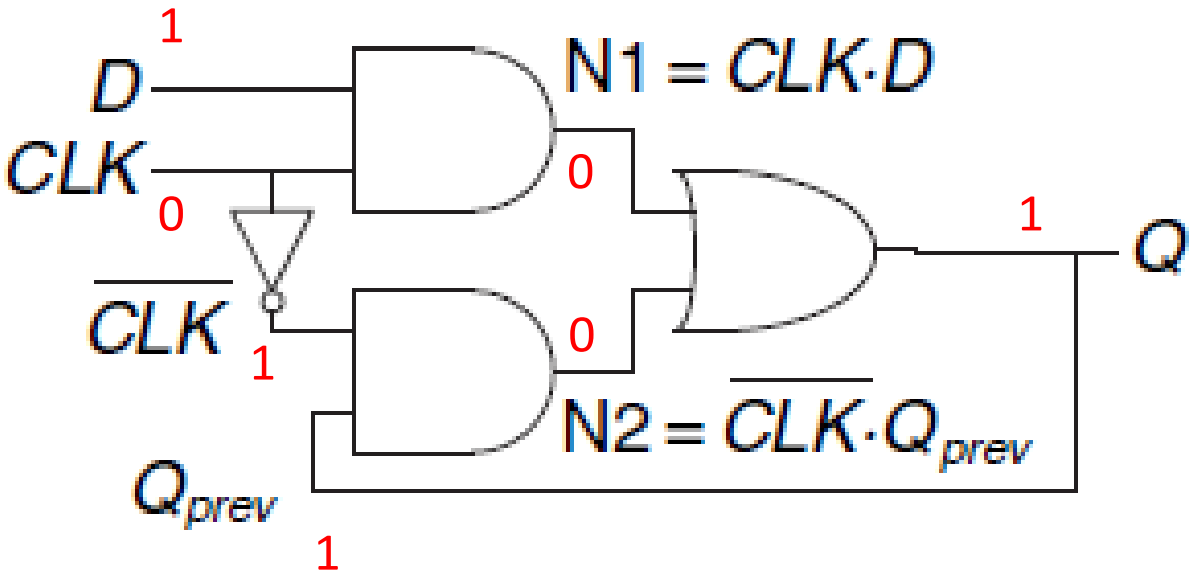
Criticità nella logica sequenziale

t_2



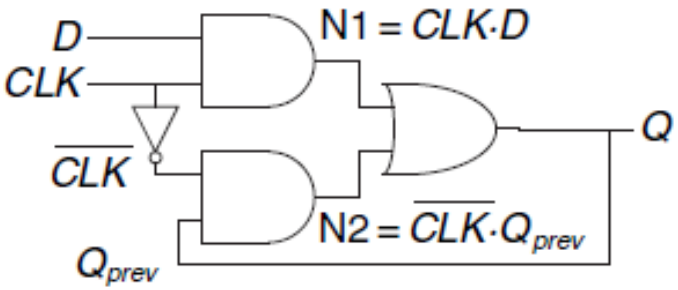
Criticità nella logica sequenziale

t_3 (= t_1)



Criticità nella logica sequenziale

- In casi più complessi che comprendono l'uso di più porte AND, NOT, OR il comportamento di una rete asincrona può dipendere fortemente dai ritardi accumulati sui singoli cammini



- D=1, CLK=1 ➡ Q=1
- CLK=0 ➡ Q oscilla (Q=Q_{prev}=1)

Logiche sequenziali sincrone

- I circuiti asincroni presentano delle criticità a volte difficilmente analizzabili
  - Dipendono dalla struttura fisica dei componenti
- Per questo si cerca di evitare di retroazionare l'output in maniera diretta e si interpone un registro nel ciclo di retroazione
- *Nell'ipotesi che il clock sia più lento del ritardo accumulato sul cammino*, il registro consente al sistema di essere sincronizzato col clock: circuito *sincrono*

Logiche sequenziali sincrone

- In generale un circuito sequenziale sincrono ha un insieme finito di stati $\{S_0,\ldots, S_{k-1}\}$
- Logica combinatoria:

  $out=f(in)$

- Logica sequenziale sincrona:

  $out=f(in,s_c)$

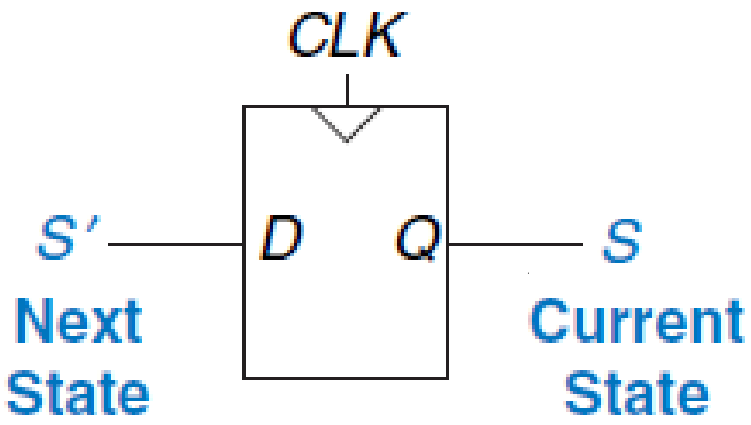
  $s_n = g(in,s_c)$

Design di logiche sequenziali sincrone

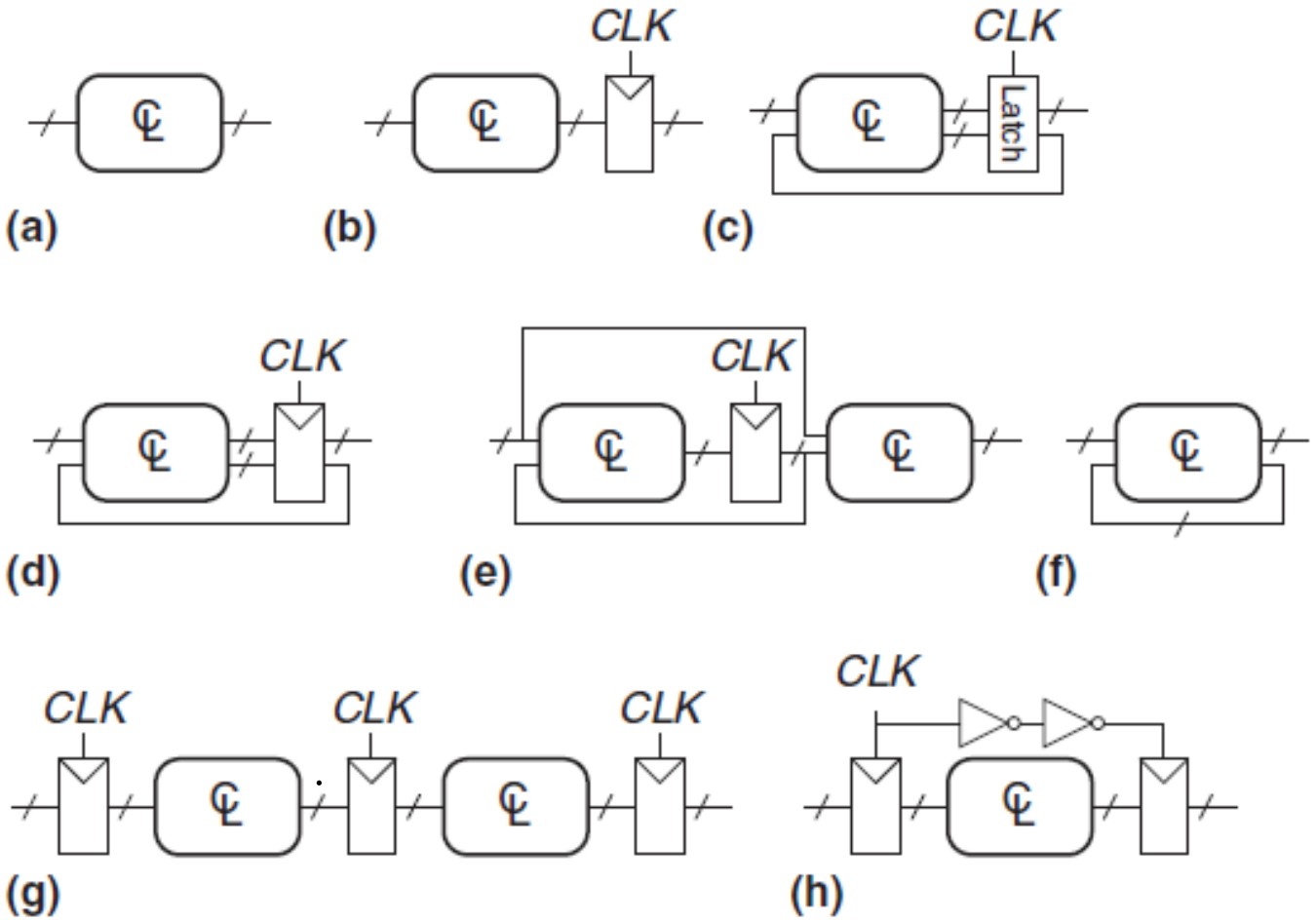
- Inserire registri nei cammini ciclici
- I registri determinano lo **stato** $S_0,\ldots, S_{k-1}$ del sistema
- I cambiamenti di stato sono determinati dalle transizioni del clock: il sistema è sincronizzato con il clock
- *Regole* di composizione:
  - Ogni componente è un registro o un circuito combinatorio
  - Almeno un componente è un registro
  - Tutti i registri sono sincronizzati con un unico clock
  - Ogni ciclo contiene almeno un registro
- Due tipici circuiti sequenziali sincroni
  - Finite State Machines (FSMs)
  - Pipelines

Current state /Next state

- Un flip-flop D è il più semplice circuito sequenziale sincrono
  - Q= s_c
  - D= s_n



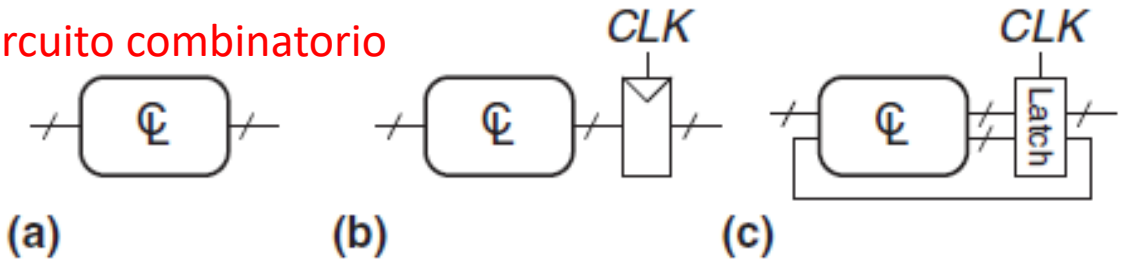
Esempi



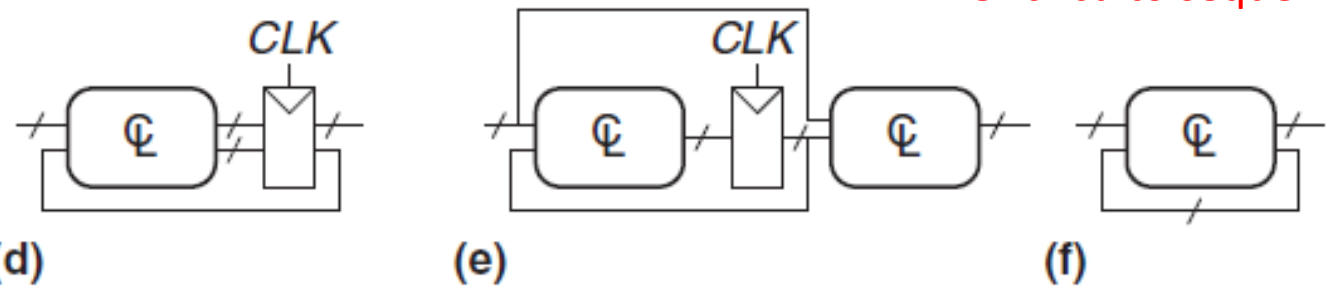
Esempi

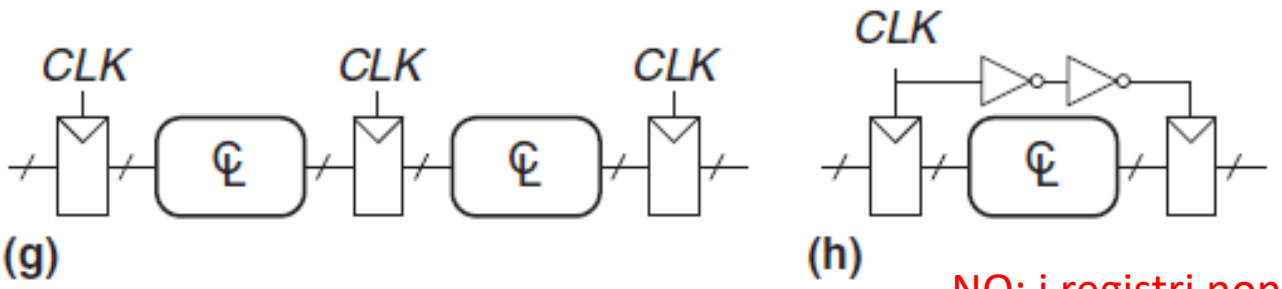
NO: circuito combinatorio

NO: latch e non flip-flop



NO: circuito sequenziale asincrono

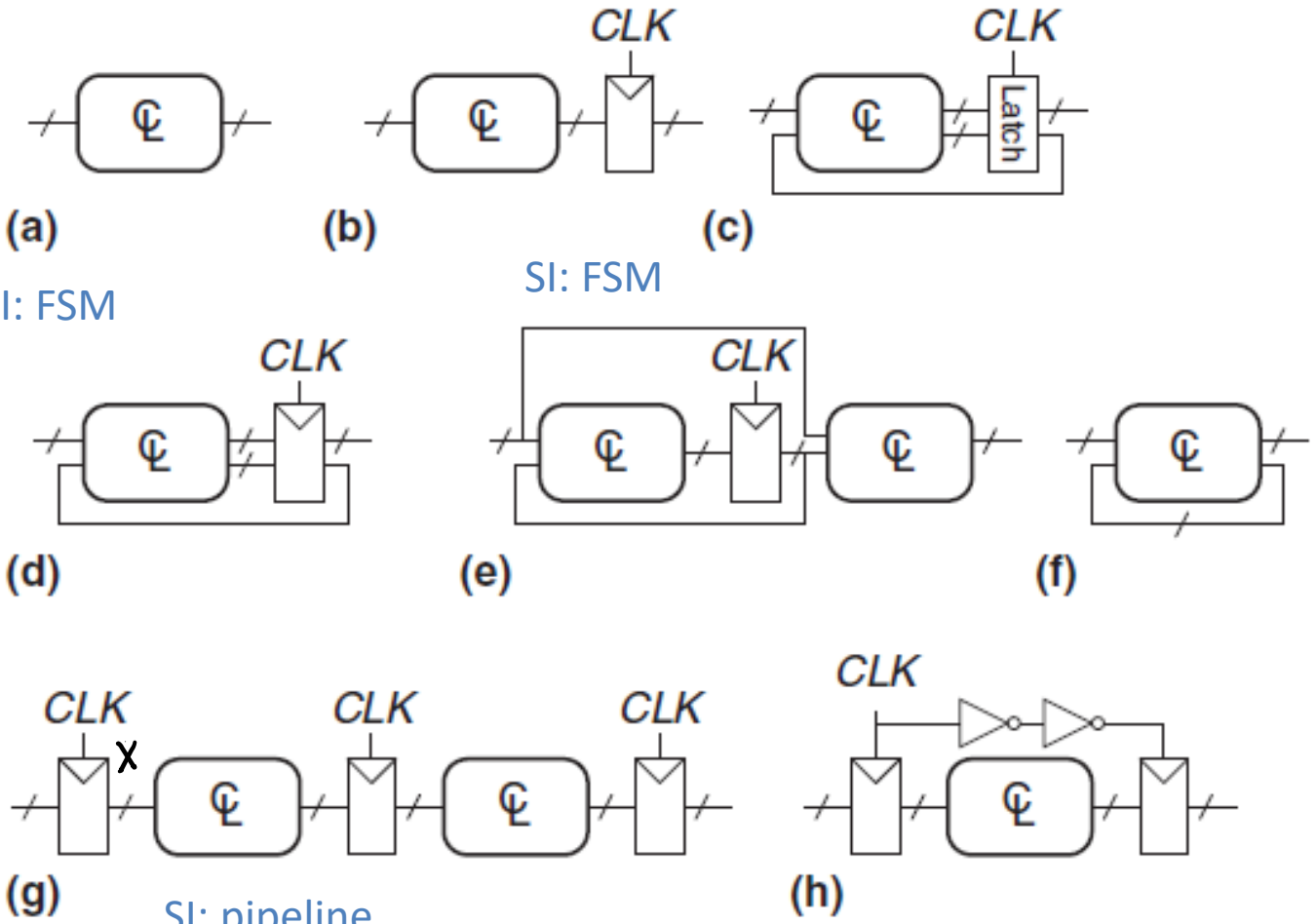




NO: i registri non hanno lo stesso clock

Esempi

SI: ma senza feedback



SI: pipeline