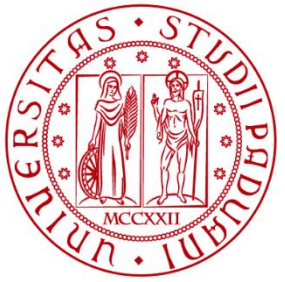
DEI
DIPARTIMENTO DI
INGEGNERIA DELL'INFORMAZIONE



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sistemi Digitali

Descrizione VHDL di registri e contatori

Marta Bagatin, marta.bagatin@unipd.it

Corso di Laurea in Ingegneria dell'Informazione

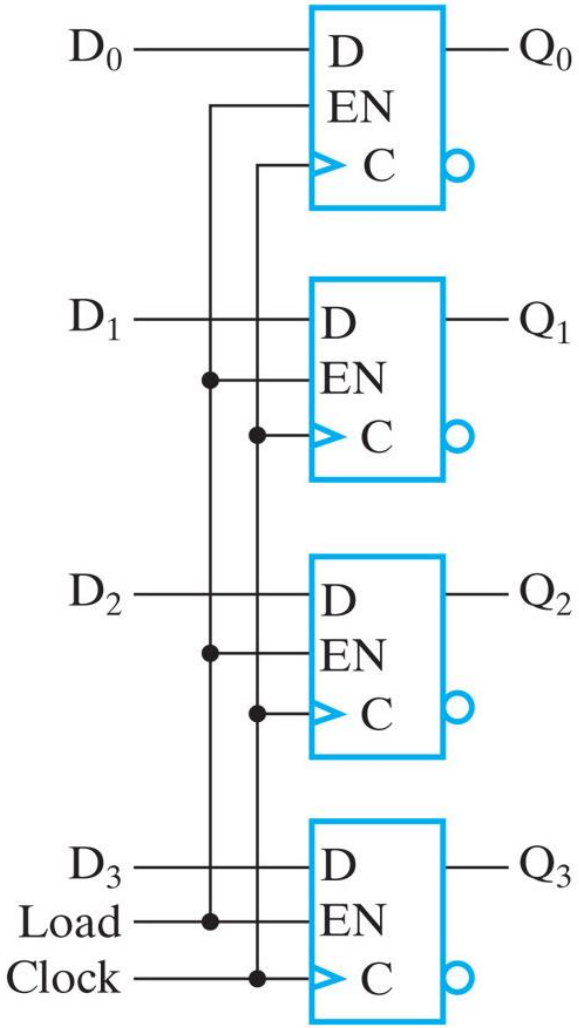
Anno accademico 2022-2023

Scopo della lezione

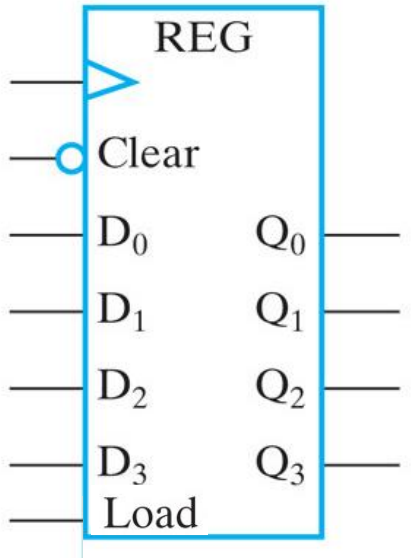
- Vedere una descrizione (behavioral) VHDL di registri e contatori
  - Registro con load e reset a n bit
  - Shift register a 4 bit
  - Contatore a 4 bit

Descrizione VHDL di un registro con load e reset

Registro a n bit con load (sincrono) e reset (asincrono)



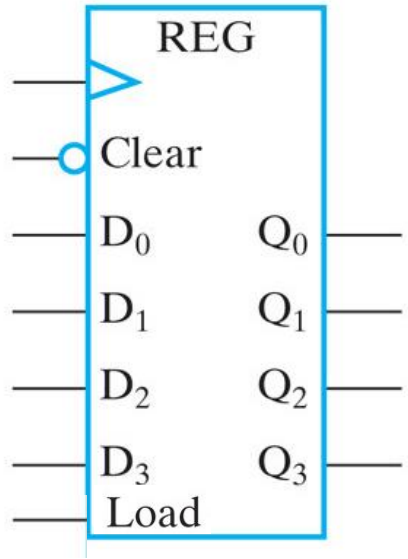
- Il **clock è in comune** tra i tutti i flip-flop: al fronte di salita, i dati vengono trasferiti dall'ingresso all'uscita dei FF
- L'operazione di **caricamento di nuovi dati** (in parallelo) nel registro è abilitata dal segnale **load**
- Un segnale opzionale di **RESET comune attivo basso** ($\overline{clear}$) consente di porre a 0 tutte le uscite Q_i in modo **asincrono**



Registro a n bit con load e reset: VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
entity register_wLoad is
        generic ( n : integer := 8);
        port( clk, clear_n, load : in std_logic;
                                 D : in std_logic_vector (n-1 downto 0);
                                 Q : out std_logic_vector (n-1 downto 0));
end register_wLoad;

architecture beh of register_wLoad is
begin
    process ( clk, clear_n )
    begin
        if ( clear_n = '0' ) then
                Q <= ( others => '0' );
         elsif ( clk'event and clk = '1' ) then
                if ( load = '1' ) then
                        Q <= D;
                end if;
         end if;
    end process;
end beh;
```

generic: usato per definire quantità parametrizzabili

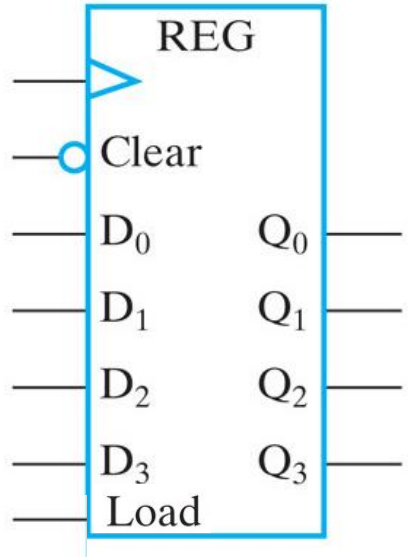


Q <= (others => '0') pone a '0' tutti i bit del vettore Q

(7 downto 4 => "1111", 3 => '0', others => '1') costrutto «aggregate» per aggregare bit o porzioni di vettore. others indica tutti gli elementi non ancora menzionati (si può usare anche da solo per indicare l'intera lunghezza del vettore)

Registro a n bit con load e reset: VHDL testbench (1/2)

```
library IEEE;
use IEEE.std_logic_1164.all;

entity testbench is
end testbench;

architecture test of testbench is

constant m : integer := 8;
signal clock, reset_n, load : std_logic;
signal d, q : std_logic_vector (m-1 downto 0);

begin

DUT : entity work.register_wLoad generic map (m) port map( clock, reset_n, load, d, q );

   generate_clock: process
   begin
        clock <= '1';
        wait for 5 ns;
        clock <= '0';
        wait for 5 ns;
   end process;
```

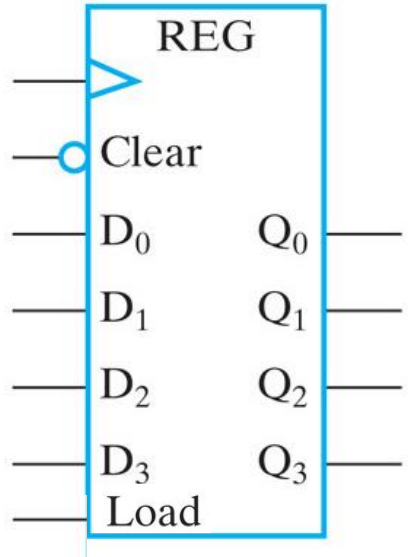


1° processo: genera il clock con un periodo 10 ns

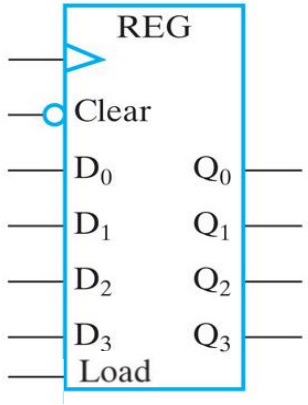
Registro a n bit con load e reset: VHDL testbench (2/2)

```
apply_inputs: process
  begin
        reset_n <= '0';
        load <= '0';
        d <= x"01";
        wait for 15 ns;
        reset_n <= '1';
        load <= '1';
        d <= x"db";
        wait for 10 ns;
        d <= x"fa";
        wait for 10 ns;
        d <= x"bc";
        wait for 20 ns;
        load <= '0';
        d <= x"83";
        wait for 10 ns;
        load <= '1';
        wait for 20 ns;
        std.env.stop;
   end process;
end test;
```

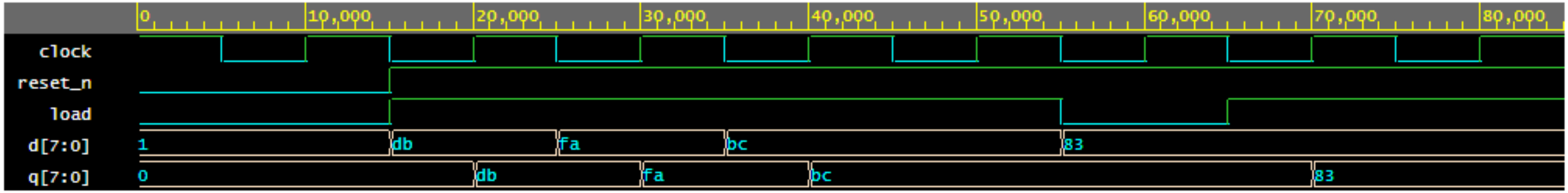
2° processo: applica gli input al device under test con opportuna temporizzazione



Registro a 8 bit con load e reset : forme d'onda simulate



Se al fronte di salita del clock load = 0 (e reset_n = 1), l'uscita mantiene il valore precedente (stato di hold-no change)

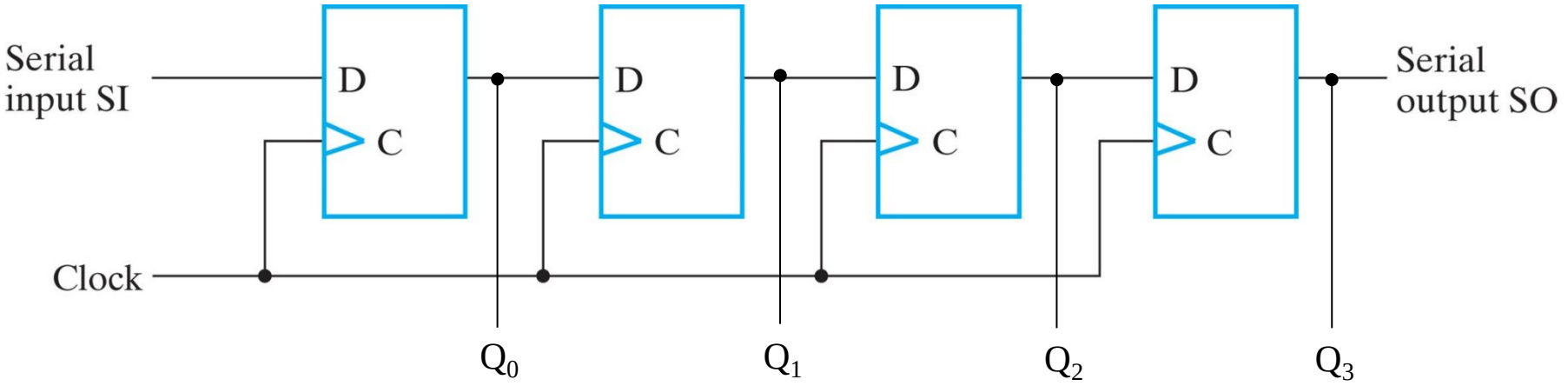


Reset iniziale (attivo basso!) pone a zero tutti i bit del registro

Ad ogni fronte di salita del clock l'uscita viene posta pari all'ingresso se entrambi reset_n = 1 e load = 1

Descrizione VHDL di uno shift register a 4 bit

Shift register a 4 bit con reset



- **Input seriale (SI)**: ingresso del flip-flop più a sinistra
- **Output seriale (SO)**: uscita del flip-flop più a destra
- Tutti i FF sono pilotati dal **clock** di sistema: lo scorrimento avviene ad ogni fronte di salita del clock dal FF di sinistra al FF di destra ⇨ dal LSB al MSB (**shift left**)
- **Reset asincrono**: imposta a 0 il valore del registro

Shift register a 4 bit con reset: VHDL (1/2)

```
library IEEE;
use IEEE.std_logic_1164.all;

entity ShiftRegister_4_r is
      port (CLK, RST, SI: in std_logic;
            SO: out std_logic;
            Q: out std_logic_vector(3 downto 0));
end ShiftRegister_4_r;
```

Shift register a 4 bit con reset: VHDL (2/2)

```
architecture beh of ShiftRegister_4_r is
signal shift: std_logic_vector(3 downto 0);
begin
    process (RST, CLK)
    begin
        if (RST = '1') then
            shift <= "0000";
        elsif (CLK'event and CLK = '1') then
            shift <= shift(2 downto 0) & SI;
        end if;
    end process;

    Q <= shift;
    SO <= shift(3);

end beh;
```

&: operatore di concatenazione tra vettori

Tempo t

FF0	FF1	FF2	FF3
A	B	C	D

SI → SO = D

Tempo t + 1

FF0	FF1	FF2	FF3
SI	A	B	C

→ SO = C

Shift register a 4 bit con reset: testbench (1/2)

```
library IEEE;
use IEEE.std_logic_1164.all;

entity test_sr4 is
end test_sr4;

architecture test of test_sr4 is

signal clk, rst, si, so: std_logic;
signal q: std_logic_vector(3 downto 0);
signal test_sequence: std_logic_vector(0 to 9):="1011000110";
constant PERIOD: time := 100 ns;

begin

DUT: entity work.ShiftRegister_4_r port map (clk, rst, si, so, q);
```

Shift register a 4 bit con reset: testbench (2/2)

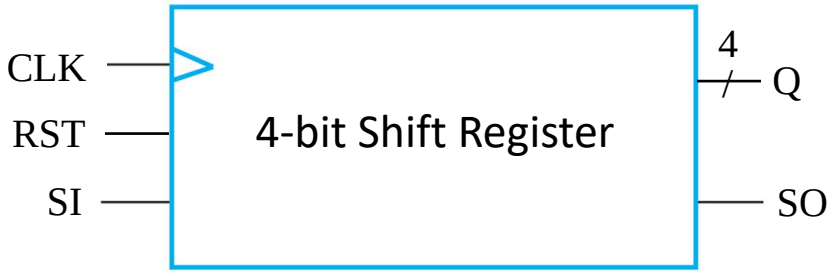
```
generate_clock: process begin
    clk <= '1';
    wait for PERIOD/2;
    clk <= '0';
    wait for PERIOD/2;
end process;

apply_inputs: process begin
    rst <= '1';
    si <= '0';
    wait for 3*PERIOD/2;
    rst <= '0';
    for i in 0 to 9 loop
        si <= test_sequence(i);
        wait for PERIOD;
    end loop;
    wait for 3*PERIOD;
    std.env.stop;
end process;
end test;
```

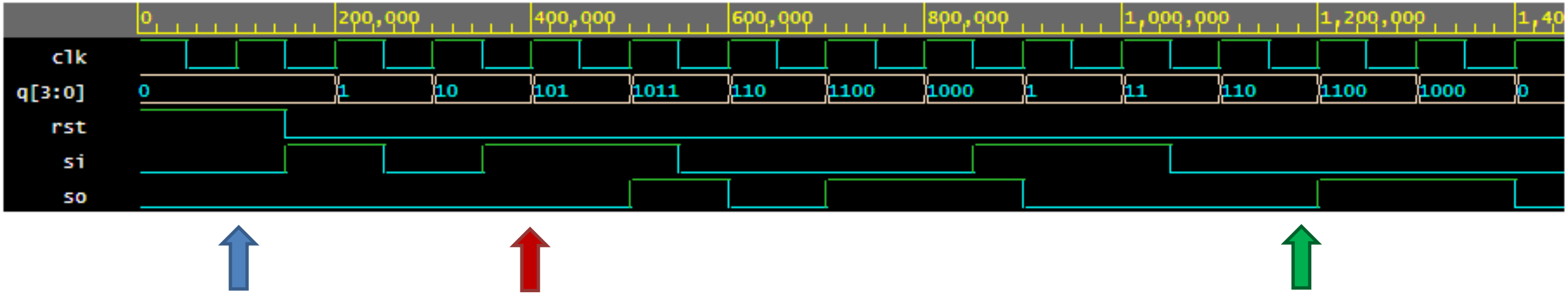
1° processo: **genera il clock** con un periodo `period`

2° processo: **applica reset** e poi la **sequenza di ingressi** contenuti nel vettore di bit `test_sequence`

Shift register a 4 bit con reset: forme d'onda simulate

Sequenza di test: 1011000110





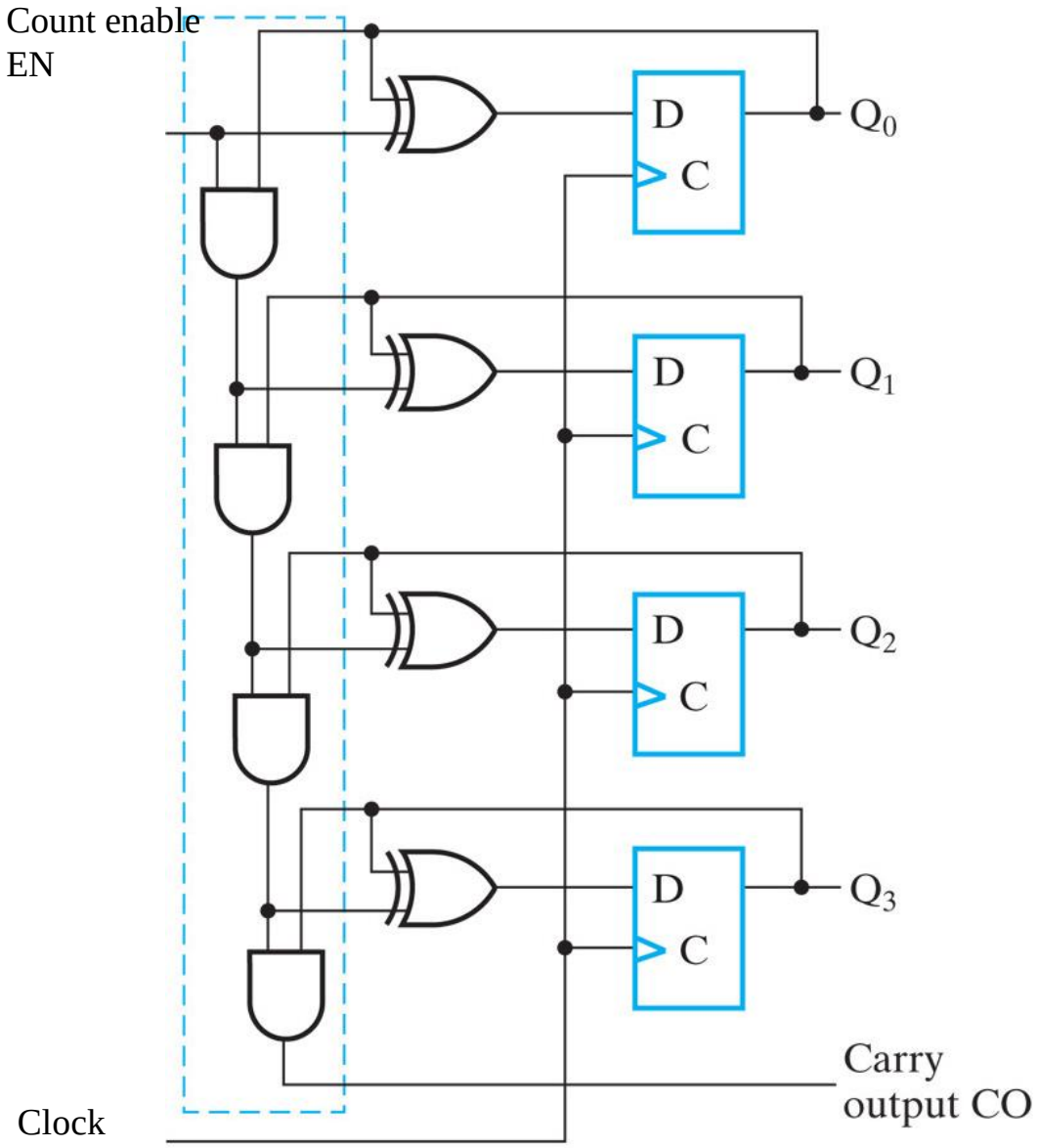
Reset iniziale:
pone a zero tutti i bit del registro

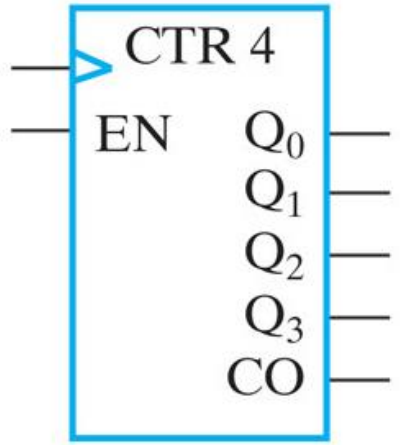
Ad ogni fronte di salita del clock i bit scorrono di una posizione dal LSB verso il MSB

L'uscita SO segue l'input SI con un ritardo di 4 cicli di clock

Descrizione VHDL di un contatore a 4 bit

Contatore a 4 bit con reset



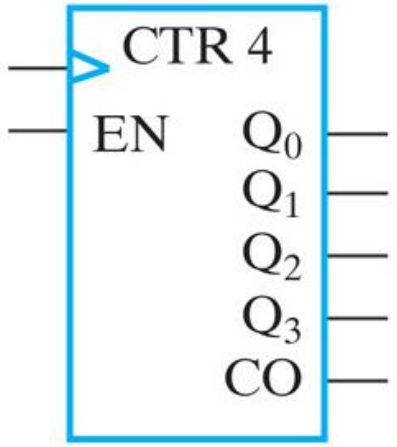


- **Input EN**: ingresso che abilita il conteggio
- **RST**: reset asincrono per azzerare il contenuto del contatore
- Tutti i FF sono pilotati dal **clock** di sistema: ad ogni fronte di salita del clock in cui EN = 1, il conteggio (Q) viene incrementato di 1
- CO: uscita di carry-out

Contatore a 4 bit con reset: VHDL (1/2)

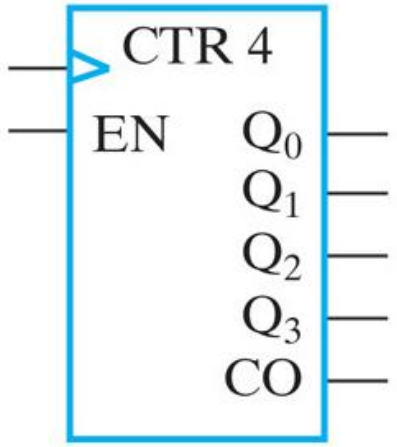
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity Counter_4_r is
      port (CLK, RST, EN: in std_logic;
                       Q: out std_logic_vector(3 downto 0));
                      CO: out std_logic;
end Counter_4_r;
```

Package che definisce le operazioni tra numeri senza segno



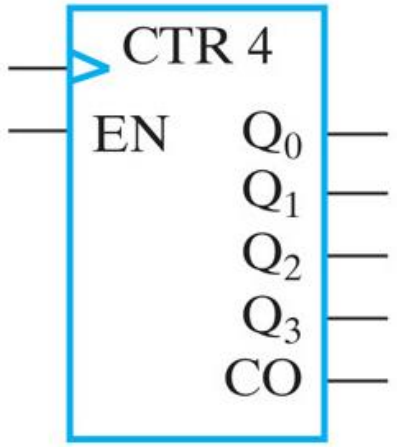
Contatore a 4 bit con reset: VHDL (2/2)

```
architecture beh of Counter_4_r is
signal count: std_logic_vector(3 downto 0);
begin
   process (RST, CLK)
   begin
     if (RST = '1') then
       count <= "0000";
     elsif (CLK'event and CLK = '1' and EN = '1') then
           count <= count + "0001";
     end if;
     end process;

   Q <= count;
   CO <= '1' when (EN = '1' and count = "1111")
             else '0';
end beh;
```



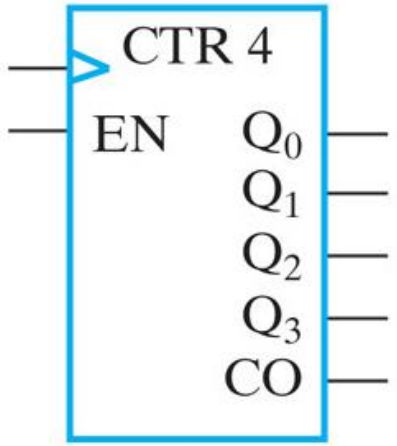
Contatore a 4 bit con reset: testbench (1/2)

```
library IEEE;
use IEEE.std_logic_1164.all;

entity test_sr4 is
end test_sr4;

architecture test of test_sr4 is

signal clk, rst, en, co: std_logic;
signal q: std_logic_vector(3 downto 0);
constant period: time := 100 ns;

begin

DUT: entity work.Counter_4_r(beh) port map (clk, rst, en, q, co);
```

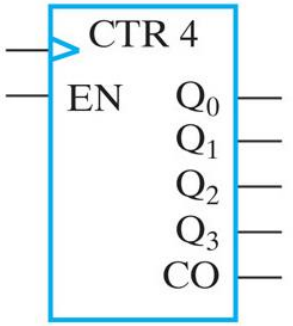


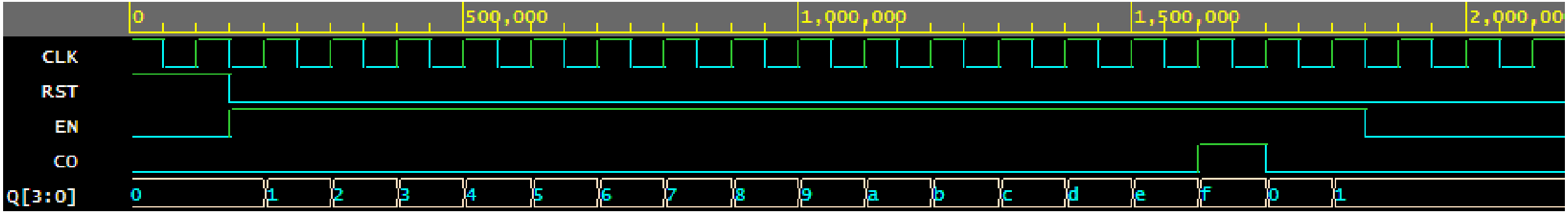
Contatore a 4 bit con reset: testbench (2/2)

```
generate_clock: process begin
    clk <= '1';
    wait for period/2;
    clk <= '0';
    wait for period/2;

apply_inputs: process begin
    rst <= '1';
    en <= '0';
    wait for 3*period/2;
    rst <= '0';
    en <= '1';
    for i in 0 to 16 loop
       wait for period;
    end loop;
    en <= '0';
    wait for 3*period;
    std.env.stop;
end process;
end test;
```



Contatore a 4 bit con reset: forme d'onda simulate





Reset iniziale: pone a zero il conteggio

Se EN = '1', ad ogni fronte di salita del clock il conteggio (Q) avanza di una unità

L'uscita CO diventa '1' quando il conteggio vale «1111»

Conteggio disabilitato se EN = '0'

Disclaimer

Figures from *Logic and Computer Design Fundamentals*, Fifth Edition, GE Mano | Kime | Martin