Architetture e reti logiche

Esercitazioni VHDL

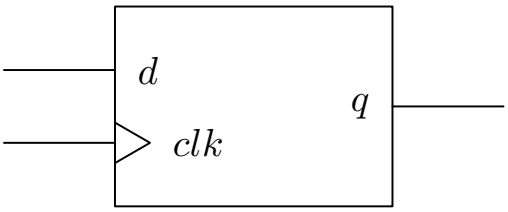
a.a. 2005/06

Componenti sequenziali in VHDL

Stefano Ferrari

Università degli Studi di Milano
Dipartimento di Tecnologie
dell'Informazione

Flip-flop di tipo D



d	clk	q
0	↑	0
1	↑	1
—	—	q

```
entity FF_D is
  port (d, clk: in bit;
              q: out bit);
end FF_D;
```

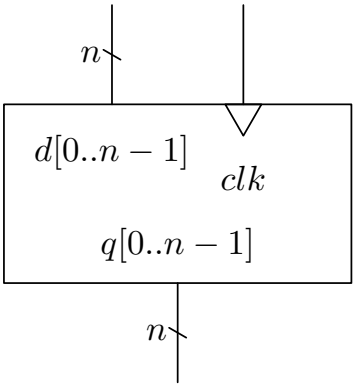
Flip-flop D: modello comportamentale

```
architecture behav1 of FF_D is
begin
  process (clk)
  begin
    if (clk'event and clk='1') then
      q <= d;
    end if;
  end process;
end behav1;
```


Flip-flop D: modello comportamentale (2)

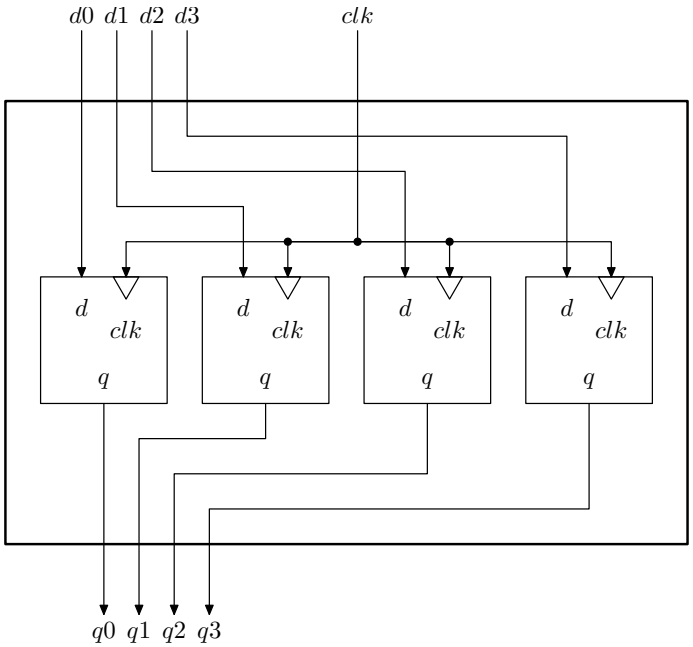
```
architecture behav2 of FF_D is
begin
  process -- no sensitivity list
  begin
    wait until clk'event and clk='1';
    q <= d;
  end process;
```

Registro a n bit




```
entity REG4 is
  port (  d: in bit_vector(0 to 3);
          clk: in bit;
          q: out bit_vector(0 to 3))
end REG4;
```

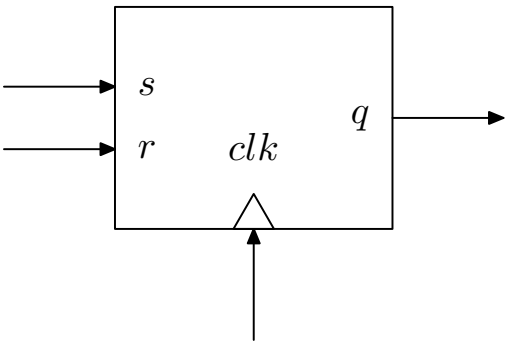

Registro a 4 bit: modello strutturale



Registro a 4 bit: modello strutturale (2)

```
architecture STRUCT of REG4 is
  component FF_D
    port (d, clk: in bit;
          q: out bit);
  end component;
begin
  u1: FF_D port map (d(0),clk,q(0));
  u2: FF_D port map (d(1),clk,q(1));
  u3: FF_D port map (d(2),clk,q(2));
  u4: FF_D port map (d(3),clk,q(3));
end STRUCT;
```


Flip-flop di tipo SR



s	r	clk	q
0	0	↑	q
0	1	↑	0
1	0	↑	1
1	1	↑	**X**
-	-	0	q
-	-	1	q

```
entity FF_set_reset is
  port (s, r, clk: in std_logic;
                q: out std_logic);
end FF_set_reset;
```

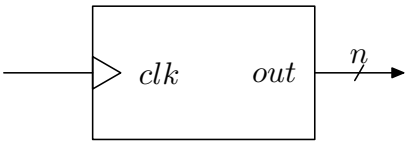
Flip-flop SR: modello comportamentale

```
architecture BEHAV of FF_set_reset is
begin
  process (clk)
  begin
    if (clk'event and clk='1') then
      if (s='1' and r='1') then
        q <= 'X';
      elsif (r='1') then
        q <= '0';
      elsif (s='1') then
        q <= '1';
      end if;
    end if;
  end process;
```


Flip-flop SR con reset asincrono

```
architecture BEHAV2 of FF_set_reset is
begin
  process (clk, r)
  begin
    if (r='1') then  -- reset asincrono e prioritario
      q <= '0';
    elsif (clk'event and clk='1') then
      if (s='1') then
        q <= '1';
      end if;
    end if;
  end process;
end BEHAV2;
```

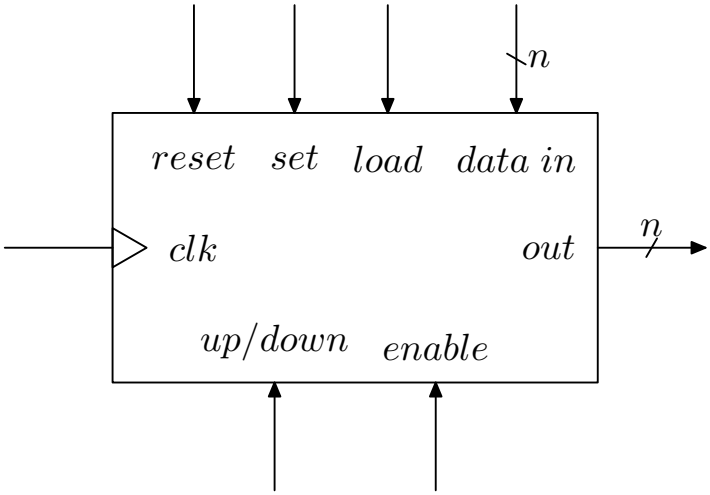
Contatore



# fronti	out
0	"0000"
1	"0001"
2	"0010"
3	"0011"

Contatore a n bit $\Leftrightarrow$ Contatore modulo 2^n

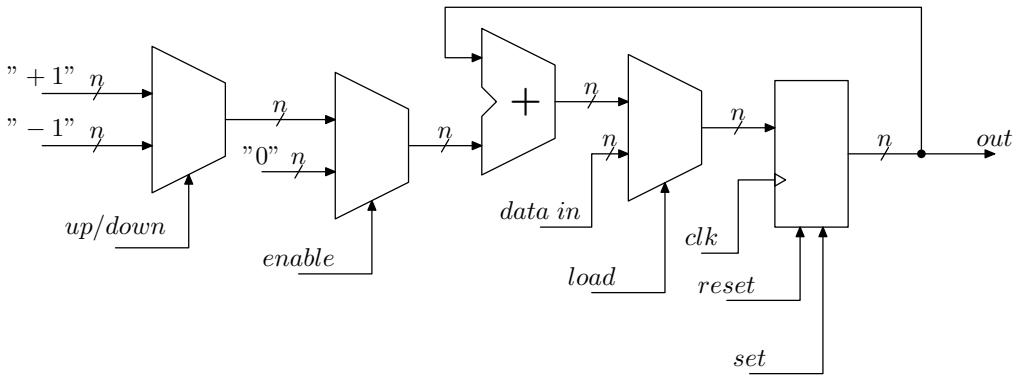
Contatore (2)



Contatore con reset

```
architecture behav of cont8 is
  signal t: std_logic_vector(0 to 7);
begin
  process(clk, reset)
  begin
    if (reset = '1' ) then
      t <= "00000000";
    elsif (clk'event and clk = '1' ) then
        t <= t + "00000001";
    end if;
  end process;
  outa <= t;    -- assegnamento concorrente
end behav;
```


Contatore completo



Shift register


```
entity ShiftRegister is
  port (
    d, clk : in std_logic;
    q : out std_logic);
end ShiftRegister;
```

Il numero di bit del registro non compare nell'interfaccia.

Shift register (2)

```
architecture behav of ShiftRegister is
  signal t : std_logic_vector(0 to 3);
begin
  process (clk)
  begin
    if clk'event and clk = '1' then
      t <= d & t(0 to 2);
      q <= t(3);
    end if;
  end process;
end behav;
```

Shift register: segnali addizionali

$reset$	pone tutti i bit a `"0"`
set	pone tutti i bit a `"1"`
$right/left$	direzione dello scorrimento
$enable$	abilita/disabilita lo scorrimento
$data\ in/load$	scrittura parallela
$data\ out$	lettura parallela