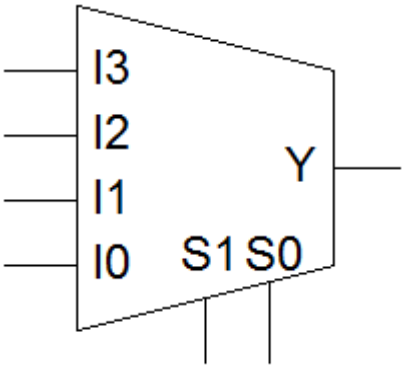
Blocchi RTL combinatori

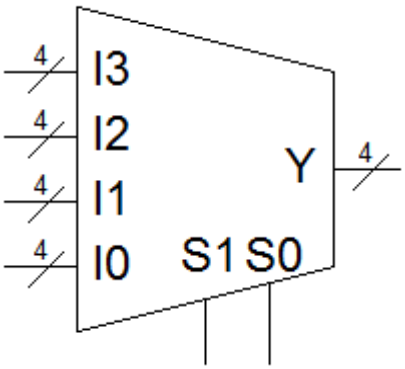
Multiplexer

Mux 4:1 ad 1 bit



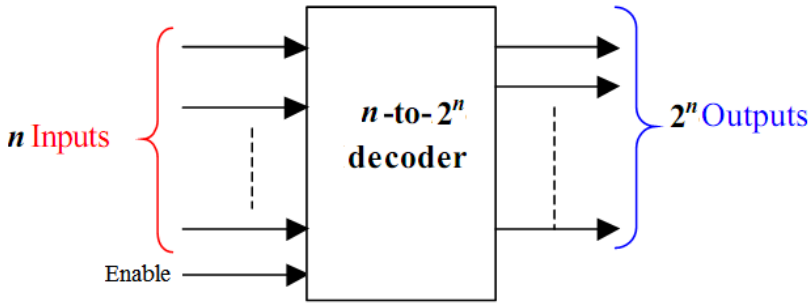

```
library ieee;
use ieee.std_logic_1164.all;

entity mux_4_to_1_w1 is
 Port(  I3 : in  STD_LOGIC;
        I2 : in  STD_LOGIC;
        I1 : in  STD_LOGIC;
        I0 : in  STD_LOGIC;
        S : in  STD_LOGIC_VECTOR (1 downto 0);
        Y : out  STD_LOGIC);
end mux_4_to_1_w1;

architecture Behavioral of mux_4_to_1_w1 is
begin
 with S select
        Y <=    I0 when "00",
                I1 when "01",
                I2 when "10",
                I3 when  others;
end Behavioral;
```

Mux 4:1 ad 4 bit




```
library ieee;
use ieee.std_logic_1164.all;

entity mux_4_to_1_w4 is
 Port (  I3 : in  std_logic_vector(3 downto 0);
        I2 : in  std_logic_vector(3 downto 0);
        I1  : in  std_logic_vector(3 downto 0);
        I0 : in  std_logic_vector(3 downto 0);
        S   : in  std_logic_vector(1 downto 0);
        Y   : out  std_logic_vector(3 downto 0));
end mux_4_to_1_w4;

architecture Behavioral of mux_4_to_1_w4 is
begin
 with S select
        Y <=    I0 when "00",
                I1 when "01",
                I2 when "10",
                I3 when  others;
end Behavioral;
```

Decoder



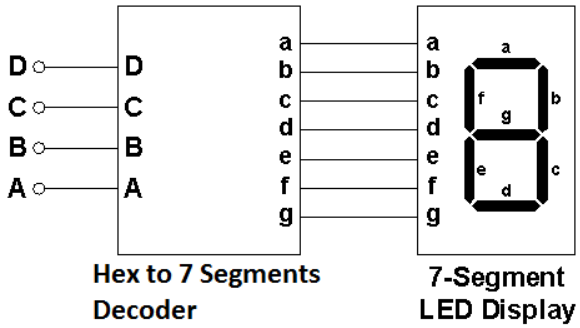
Esempi di decoder binatio da 3 to 8 con uscite attive basse ed enable attivo basso.

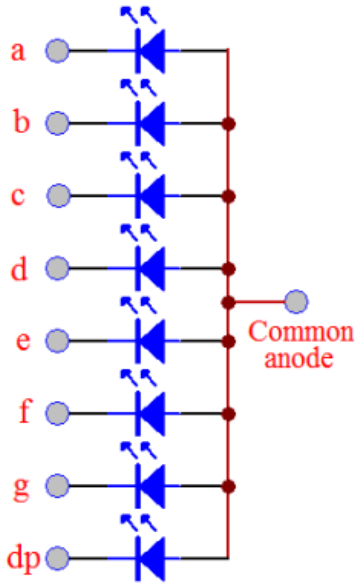
```
-- Versione descritta mediante processo
--
library ieee;
use ieee.std_logic_1164.all;

entity dec_3_to_8_le is
  Port (n_g : in  std_logic;
        s   : in  std_logic_vector(2 downto 0);
        n_y : out  std_logic_vector(7 downto 0));
end dec_3_to_8_le;

architecture Behavioral of dec_3_to_8_le is

begin
process(n_g, s)
begin
    if ( n_g = '1') then n_y <= "11111111";
    else
      case s is
        when "000"     => n_y <= "11111110";
        when "001"     => n_y <= "11111101";
        when "010"     => n_y <= "11111011";
        when "011"     => n_y <= "11110111";
        when "100"     => n_y <= "11101111";
        when "101"     => n_y <= "11011111";
        when "110"     => n_y <= "10111111";
        when others    => n_y <= "01111111";
      end case;
    end if;
end process;
end Behavioral;
```

```
-- Versione descritta mediante with-select-when

library ieee;
use ieee.std_logic_1164.all;

entity dec_3_to_8_le is
  Port ( n_g : in  std_logic;
         s   : in  std_logic_vector(2 downto 0);
         n_y : out  std_logic_vector(7 downto 0));
end dec_3_to_8_le;

architecture Behavioral of dec_3_to_8_le is

signal ens: std_logic_vector(3 downto 0);

begin
         ens <= n_g & s;
         with ens select
                n_y<=   "11111110" when "0000",
                        "11111101" when "0001",
                        "11111011" when "0010",
                        "11110111" when "0011",
                        "11101111" when "0100",
                        "11011111" when "0101",
                        "10111111" when "0110",
                        "01111111" when "0111",
                        "11111111" when other;
end Behavioral;
```

Decoder per display a 7 segmenti

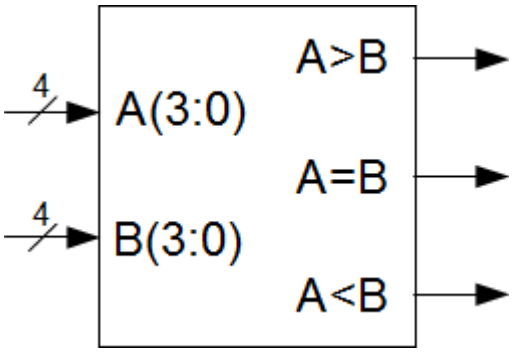
Esempi di decodifica per display a 7 segmenti – anodo comune -





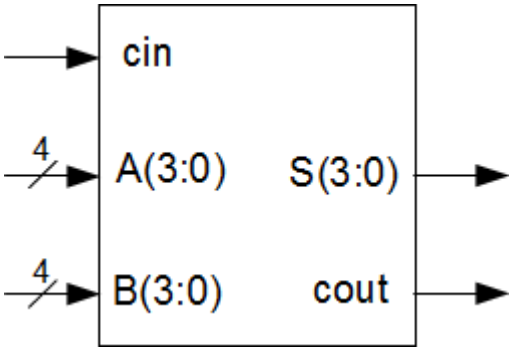

```
-- hex(3:0) = D, C, B, A
-- seg(6:0) = g, f, e, d, c, b, a

library ieee;
use ieee.std_logic_1164.all;

entity hex2disp7 is
port(   hex: in   std_logic_vector(3 downto 0);
        seg: out std_logic_vector(6 downto 0));
end hex2disp7;

architecture behavior of hex2disp7 is
begin
with hex select
  seg <= "1000000"    when   "0000",      -- 0
         "1111001"    when   "0001",      -- 1
         "0100100"    when   "0010",      -- 2
         "0110000"    when   "0011",      -- 3
         "0011001"    when   "0100",      -- 4
         "0010010"    when   "0101",      -- 5
         "0000010"    when   "0110",      -- 6
         "1111000"    when   "0111",      -- 7
         "0000000"    when   "1000",      -- 8
         "0010000"    when   "1001",      -- 9
         "0001000"    when   "1010",      -- a
         "0000011"    when   "1011",      -- b
         "1000110"    when   "1100",      -- c
         "0100001"    when   "1101",      -- d
         "0000110"    when   "1110",      -- e
         "0001110"    when   "1111",      -- f
         "1111111"    when   others;

end behavior;
```

Comparatore binario a 4 bit



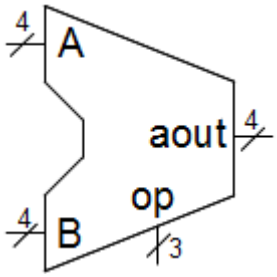

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity compare is
  Port ( A: in std_logic_vector(3 downto 0) ;
         B: in std_logic_vector(3 downto 0) ;
         Aeqb: out std_logic;
         Agtb: out std_logic;
         Alob : out std_logic) ;
end compare;

architecture behavior of compare is
begin
         Aeqb   <= '1'   when a = b else '0';
         Agtb   <= '1'   when a > b else '0';
         Altb   <= '1'   when a < b else '0';
end behavior ;
```

Sommatore a 4 bit




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

entity adder_4_bits is
 port ( cin : in          std_logic;
        A : in            std_logic_vector (3 downto 0);
        B : in            std_logic_vector (3 downto 0);
        S : out           std_logic_vector (3 downto 0);
        cout : out        std_logic);
end adder_4_bits;

architecture behavioral of adder_4_bits is
signal int: std_logic_vector(4 downto 0);
begin

  int(4 downto 0)      <= ('0'& a) + ('0'& b)+cin;
  s                    <= int_add(3 downto 0);
  cout                 <= int_add(4);

end behavioral;
```

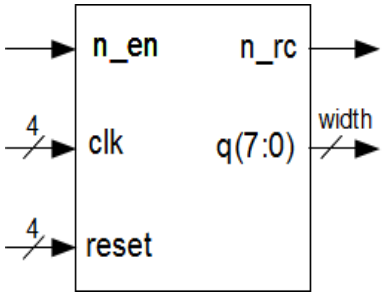
ALU a 4 bit




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

entity alu is
 port (  a, b :    in std_logic_vector(3 downto 0);
         aout :    out std_logic_vector(3 downto 0));
         op :      in std_logic_vector(2 downto 0);
end alu;

architecture behavior of alu is
begin
 process ( op, a, b )
         begin
         case op is
                 when "000"     => aout <= "0000";
                 when "001"     => aout <= b – a;
                 when "010"     => aout <= a – b;
                 when "011"     => aout <= a + b;
                 when "100"     => aout <= a xor b;
                 when "101"     => aout <= a or b;
                 when "110"     => aout <= a and b;
                 when others    => aout <= "1111";
         end case;
 end process;
end behavior ;
```

Contatore binario

clk : ingresso di clock ;
reset : ingresso di reset, attivo alto;
n_en : ingresso di abilitazione al conteggio, attivo basso;
q(7:0) : uscita contatore;
n_rc : segnale di ripple carry attivo basso. Si attiva per ½ ciclo di clock quando il contatore è nell'ultimo stato.




```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity cnt_bin is
        generic(width : integer:=8);
port (  reset   : in  std_logic;
        clk     : in  std_logic;
        n_en    : in  std_logic;
        q       : out std_logic_vector (width-1 downto 0);
        n_rc    : out std_logic);
end cnt_bin;

architecture behavioral of cnt_bin is
signal tmp : std_logic_vector (width-1 downto 0);
begin
 q <= tmp;
 cnt: process(clk, reset)        -- counter process
        begin
         if (clk'event and clk = '1')      then
                if (reset='1')   then
                                 tmp <= (others => '0');
                elsif (n_en ='1') then
                                 tmp <= tmp + 1;
                end if;
        end if;
      end process;

 rc: process(clk, n_en, tmp)              -- ripple carry
        begin
         n_rc <= '1';
         if(tmp = (width-1 downto 0 => '1')) then
                n_rc <= n_en or clk;
         end if;
     end process;
end behavioral;
```