MUX

```
--implementazione di un Multiplexer a due bit

entity Mux is
--a e b ingressi, s selettore,r risultato
    port(a,b,s: in bit;
        r: out bit);
end Mux;

architecture MX of Mux is
    begin

    r <= ( a and (not s) ) or (b and s);
end MX;
```

CARRY LOOK AHEAD

```
--Implementazione del Carry Look Ahead a 4 bit

Entity carrylookahead is

port( a,b: in bit_Vector(3 downto 0);
      cin: in bit;
      s: out bit_Vector(3 downto 0);
      cout: out bit);

end carrylookahead;

Architecture CarryLookAhead4bit of carrylookahead
is

signal p,g: bit_vector(3 downto 0);
signal c : bit_vector(3 downto 0);

    begin

--generate g = a and b
    g(0) <= a(0) and b(0);
    g(1) <= a(1) and b(1);
    g(2) <= a(2) and b(2);
    g(3) <= a(3) and b(3);
--propagate p = a xor b;
    p(0) <= a(0) xor b(0);
    p(1) <= a(1) xor b(1);
    p(2) <= a(2) xor b(2);
    p(3) <= a(3) xor b(3);

--carry

    c(0) <= cin;
    c(1) <= (g(0) or (c(0) and p(0)));
    c(2) <= (g(1) or((g(0) or (c(0) and p(0))) and
             p(1)));
```

```
    c(3) <= (g(2) or ( (g(1) or((g(0) or (c(0) and
            p(0))) and p(1))) and p(2)));

    cout <= (g(3) or ( (g(2) or ( (g(1) or((g(0)
            or (c(0) and p(0))) and p(1))) and
            p(2))) and p(3))) ;

--somma s(i-esima) = p(i-esimo) xor c(i-esimo-1)
    s(0) <= p(0) xor c(0);
    s(1) <= p(1) xor c(1);
    s(2) <= p(2) xor c(2);
    s(3) <= p(3) xor c(3);

end CarryLookAhead4bit;
```

CARRY SELECT

```
--Implementazione Carry Select ad otto bit
Entity CarrySelect8 is

--ingressi A(0:7), B(0:7), cin
--uscite: cout, S(0:7)
    port(
        a,b : in bit_Vector(7 downto 0 );
        cin : in bit;
        s : out bit_Vector(7 downto 0);
        cout : out bit
        );
end CarrySelect8;

Architecture CS8b of CarrySelect8 is


signal s0,s1 : bit_Vector(3 downto 0);
signal c0,c1,ci : bit;

--inclusione dei componenti CarryLookAhead a 4
bit
component carrylookahead is

port( a,b: in bit_Vector(3 downto 0 );
      cin: in bit;
      s: out bit_Vector(3 downto 0 );
      cout: out bit
      );
end component;

--inclusione dei componenti del Mux a 2 bit
component Mux is

port(a,b,s: in bit;
        r: out bit
    );
end component;
begin
--operazione dei Carry Look Ahead

CLAI : carrylookahead  port map(a(3 downto
0 ),b(3 downto 0 ),cin,s(3 downto 0 ),ci);
```

```
CLA0 : carrylookahead  port map(a(7 downto 4),b(7
downto 4),'0',s0,c0);

CLA1 : carrylookahead port map(a(7 downto 4),b(7
downto 4),'1',s1,c1);

--operazioni dei Mux
M4 : Mux port map(s0(0),s1(0),ci,s(4));
M5 : Mux port map(s0(1),s1(1),ci,s(5));
M6 : Mux port map(s0(2),s1(2),ci,s(6));
M7 : Mux port map(s0(3),s1(3),ci,s(7));
-- mux che calcola il riporto finale
MC :Mux port map(c0,c1,ci,cout);

end CS8b;
```

SIMULAZIONE

```
--file di testing per il CarrySelect ad 8 bit con almeno
dieci coppie di operanti

--definizione dell'entita' astratta
Entity TestCarrySelect8 is
--entity astratta e percio' vuota
end TestCarrySelect8;

--definizione dell'architettura
Architecture SimCS8b of TestCarrySelect8 is
--si richiama il componente da utilizzare
component CarrySelect8 is

     port(
          a,b : in bit_Vector(7 downto 0);
          cin : in bit;
          s : out bit_Vector(7 downto 0);
          cout : out bit
          );
end component;

--si definiscono i segnali che si vogliono usare per
l'utilizzo del componente

--ingressi
signal ia,ib: bit_Vector(7 downto 0);
signal icin : bit;
--uscite
signal os: bit_Vector(7 downto 0);
signal ocout: bit;

begin

--implementazione del circuito denominato tale(nella
dichiarazione del componente)

circuito : CarrySelect8

     port map(ia,ib,icin,os,ocout);

     process
     --si instanzia il componente
     begin


     --si devono assegnare dei valori che i segnali di
     ingresso devono assumere negli istanti di tempo
     considerati
     --gli ingressi sono vettori!
```

```
--PRIMA COPPIA:inizializzazione a zero
ia<="00000000";ib<="00000000";
icin<='1';    --risultato= 0 0000 0001
wait for 10 ns;

--SECONDA COPPIA:inizializzazione ad uno
ia<="11111111";ib<="11111111";
icin<='0';    --risultato 1 1111 1110
wait for 15 ns;

--TERZA COPPIA
ia<="00001111";
icin<='1';    --risultato 1 0000 1111
wait for 10 ns;

--QUARTA COPPIA: bit alternati
ia<="01010101";ib<="10101010";
icin<='0';    --risultato 0 1111 1111
wait for 15 ns;

--QUINTA COPPIA:
icin<='1';    --risultato 1 0000 0000
wait for 10 ns;

--SESTA COPPIA
ia<="10000001";ib<="00001000";
icin<='0';    --risultato 0 1000 1001
wait for 15 ns;

--SETTIMA COPPIA
ia<="00011000";ib<="00000111";
icin<='1';    --risultato 0 0010 0000
wait for 10 ns;

--OTTAVA COPPIA
ia<="01111111";ib<="00000000";
icin<='1';    --risultato 0 1000 0000
wait for 15 ns;

--NONA COPPIA
ia<="11101010";ib<="01110101";
icin<='1';    --risultato 1 0110 0000
wait for 10 ns;


--DECIMA COPPIA
ia<="00000001";ib<="00000001";
icin<='1';    --risultato 0 0000 0011
wait for 15 ns;

--UNDICESIMA COPPIA
ia<="10000000";ib<="10000000";
icin<='0';    --risultato 1 0000 0000
wait for 10 ns;
```

```
    --si conclude il processo
    end process;

--si conclude la simulazione
end simCS8b;
```