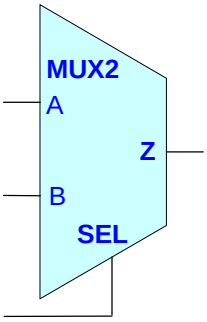
Università degli Studi di Milano – Laurea e Diploma Universitario in Informatica a Crema
Esercitazioni del Corso ARCHITETTURA DEGLI ELABORATORI I
Ing. Cristina Silvano – Anno Accademico 2000-2001
Modelli VHDL

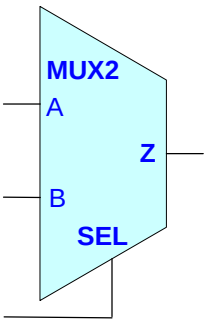
1. Scrivere il modello VHDL *(comportamentale e dataflow)* del componente **MUX2** descritto dalla sequente specifica:

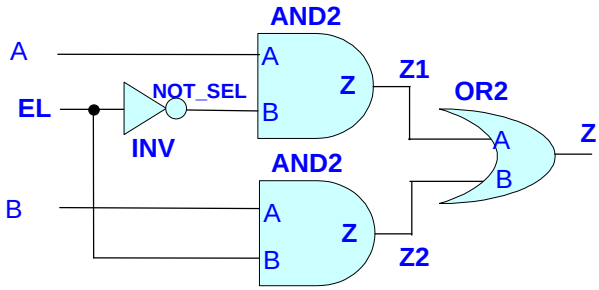


Ingressi			Uscita
SEL	A	B	Z
0	0	-	0
0	1	-	1
1	-	0	0
1	-	1	1

```
entity MUX2 is
port ( A, B, SEL: in bit;
       Z: out bit);
end MUX2;

-- modello comportamentale --
architecture ARC1 of MUX2 is
begin
 P1: process (A, B, SEL)
 begin
       if (SEL = '0') then
             Z <= A;
       else
             Z <= B;
       end if;
 end process P1;
end ARC1;

-- modello data-flow --
architecture DATA_FLOW1 of MUX2 is
begin
 Z <=  A     when (SEL = '0') else
       B ;
end DATA_FLOW;

-- modello data-flow --
architecture DATA_FLOW2 of MUX2 is
begin
 Z <=  (A and not SEL)or (B and SEL);
end DATA_FLOW2;
```

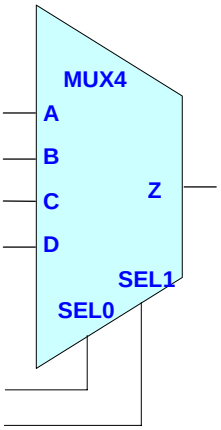
2. Scrivere il modello VHDL *(strutturale)* del componente **MUX2** descritto dalla sequente specifica:






```
entity MUX2 is
port ( A, B, SEL: in bit;
       Z: out bit);
end MUX2;

-- modello strutturale --
architecture STRUCT of MUX2 is

component inv
 port  (A: in bit;
        Z: out bit);
end component;

component and2
 port  (A, B: in bit;
        Z: out bit);
end component;

component or2
 port  (A, B: in bit;
        Z: out bit);
end component;

signal NOT_SEL, Z1, Z2: bit;

begin
 u1: inv port map (SEL, NOT_SEL);
 u2: and2 port map (A, NOT_SEL, Z1);
 u3: and2 port map (B, SEL, Z2);
 u4: or2 port map (Z1, Z2, Z);
end STRUCT;
```

3. Scrivere il modello VHDL *(comportamentale e dataflow)* del componente **MUX4** descritto dalla sequente specifica:

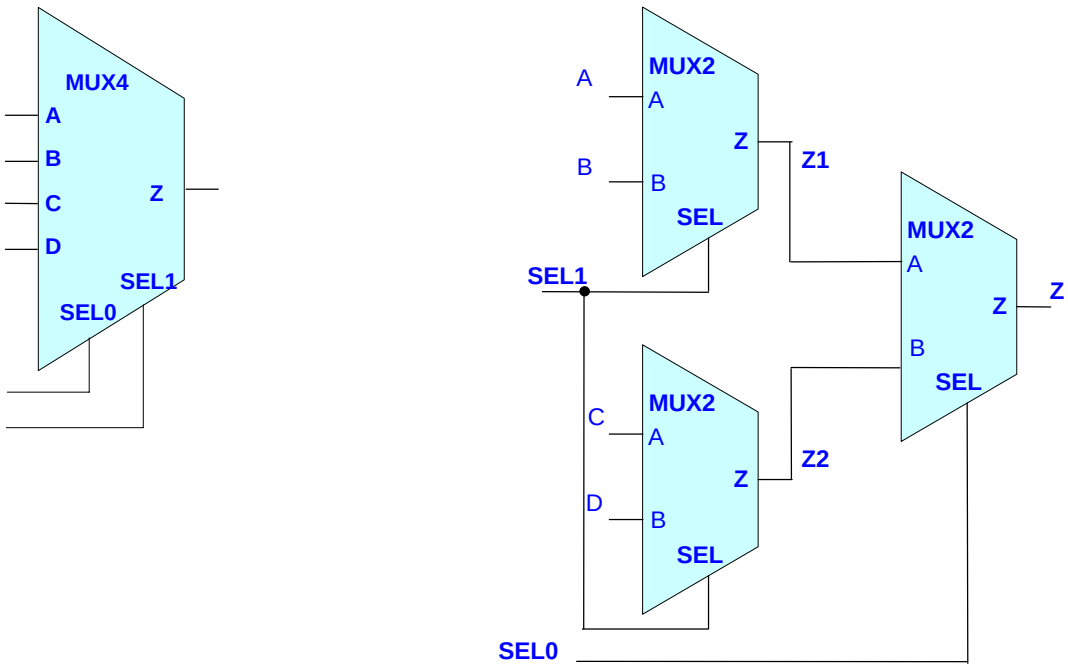


Ingressi						Uscita
SEL0	SEL1	A	B	C	D	Z
0	0	0	-	-	-	0
0	0	1	-	-	-	1
0	1	-	0	-	-	0
0	1	-	1	-	-	1
1	0	-	-	0	-	0
1	0	-	-	1	-	1
1	1	-	-	-	0	0
1	1	-	-	-	1	1

```
entity MUX4 is
port (A, B, C, D, SEL0, SEL1: in bit;
      Z: out bit);
end MUX4;

-- modello comportamentale --
architecture ARC1 of MUX4 is
begin
 P1: process (A, B, C, D, SEL0, SEL1)
 begin
      if (SEL0 = '0' and SEL1 = '0') then
            Z <= A;
      elsif (SEL0 = '0' and SEL1 = '1') then
            Z <= B;
      elsif (SEL0 = '1' and SEL1 = '0') then
            Z <= C;
      elsif (SEL0 = '1' and SEL1 = '1') then
            Z <= D;
      end if;
 end process P1;
end ARC1;

-- modello comportamentale --
architecture ARC2 of MUX4 is
begin
 P2: process (A, B, C, D, SEL0, SEL1)
 begin
      case (SEL0 & SEL1) is
      when "00" =>
            Z <= A;
      when "01" =>
            Z <= B;
      when "10" =>
            Z <= C;
      when "11" =>
            Z <= D;
      end case;
```

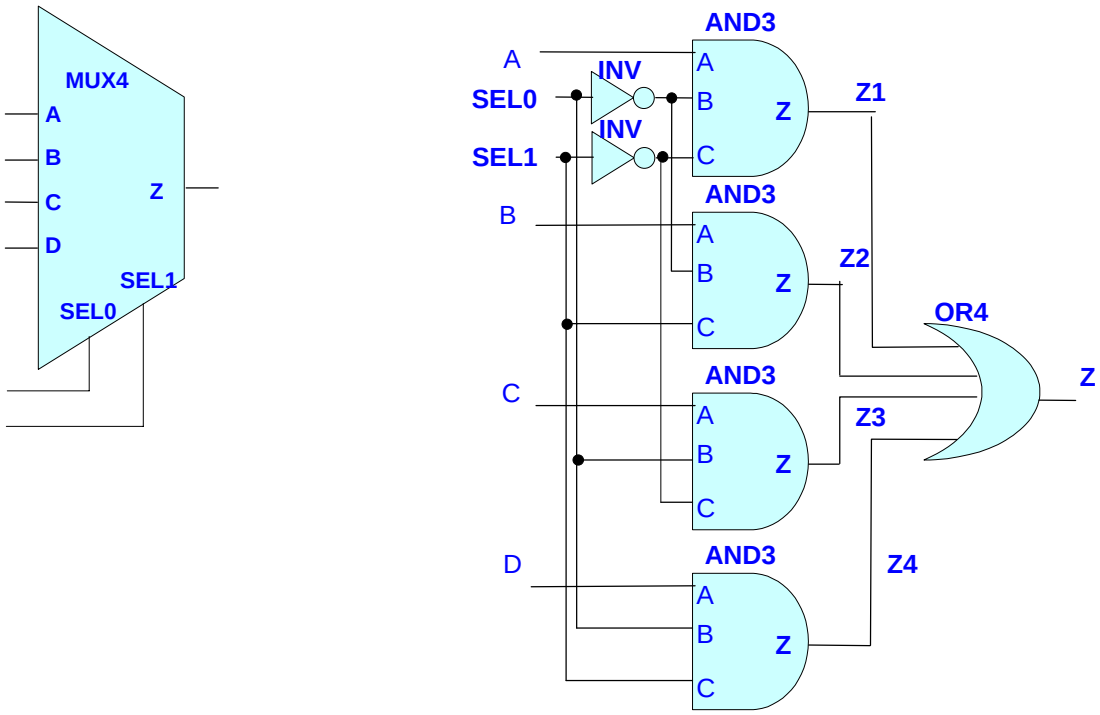
```
 end process P2;
end ARC2;


-- modello data-flow --
architecture DATA_FLOW1 of MUX4 is
begin
 Z <= A          when (SEL0 = '0' and SEL1 = '0') else
      B          when (SEL0 = '0' and SEL1 = '1') else
      C          when (SEL0 = '1' and SEL1 = '0') else
      D;
end DATA_FLOW1;


-- modello data-flow --
architecture DATA_FLOW2 of MUX4 is
begin
 Z <=  (A and not SEL0 and not SEL1)or (B and not SEL0 and SEL1)
      or (C and SEL0 and not SEL1) or (D and SEL0 and SEL1) ;
end DATA_FLOW2;
```

4. Scrivere il modello VHDL *(strutturale)* del componente **MUX4** descritto dalla sequente specifica:




```
entity MUX4 is
port (A, B, C, D, SEL0, SEL1: in bit;
      Z: out bit);
end MUX4;

-- modello strutturale --
architecture STRUCT1 of MUX4 is

component MUX2
 port (A, B, SEL: in bit;
       Z: out bit);
end component;

signal Z1, Z2: bit ;

begin
 u1: MUX2 port map (A, B, SEL1, Z1);
 u2: MUX2 port map (C, D, SEL1, Z2);
 u3: MUX2 port map (Z1, Z2, SEL0, Z);
end STRUCT1;
```

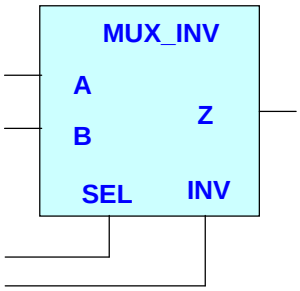
5. Scrivere il modello VHDL *(strutturale)* del componente **MUX4** descritto dalla sequente specifica:




```
entity MUX4 is
port (A, B, C, D, SEL0, SEL1: in bit;
      Z: out bit);
end MUX4;

-- modello strutturale --
architecture STRUCT2 of MUX4 is

component inv
 port (A: in bit;
       Z: out bit);
end component;
component and3
 port (A, B, C: in bit;
       Z: out bit);
end component;
component or4
 port (A, B, C, D: in bit;
       Z: out bit);
end component;

signal NOT_SEL0, NOT_SEL1, Z1, Z2, Z3, Z4: bit;

begin
 u1: inv port map (SEL0, NOT_SEL0);
 u2: inv port map (SEL1, NOT_SEL1);
```

```
      u3: and3 port map (A, NOT_SEL0, NOT_SEL1, Z1);
      u4: and3 port map (B, NOT_SEL0, SEL1, Z2);
      u5: and3 port map (C, SEL0, NOT_SEL1, Z3);
      u6: and3 port map (D, SEL0, SEL1, Z4);
      u7: or4 port map (Z1, Z2, Z3, Z4, Z);
    end STRUCT2;
```

6. Scrivere il modello VHDL *(comportamentale e dataflow)* del componente **MUX_INV** descritto dalla seguente specifica:

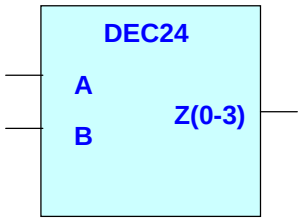


Ingressi				Uscita
SEL	INV	A	B	Z
0	0	0	-	0
0	0	1	-	1
0	1	0	-	1
0	1	1	-	0
1	0	-	0	0
1	0	-	1	1
1	1	-	0	1
1	1	-	1	0

```
entity MUX_INV is
port (A, B, SEL, INV: in bit;
      Z: out bit);
end MUX_INV;

-- modello comportamentale --
architecture ARC1 of MUX_INV is
begin
 P1: process (A, B, SEL, INV)
 begin
      if (SEL = '0' and INV = '0') then
            Z <= A;
      elsif (SEL = '0' and INV = '1') then
            Z <= not A;
      elsif (SEL = '1' and INV = '0') then
            Z <= B;
      elsif (SEL = '1' and INV = '1') then
            Z <= not B;
      end if;
 end process P1;
end ARC1;

-- modello comportamentale --
architecture ARC2 of MUX_INV is
begin
 P2: process (A, B, SEL, INV)
 begin
      case (SEL & INV)  is
      when "00" =>
            Z <= A;
      when "01" =>
            Z <= not A;
      when "10" =>
            Z <= B;
      when "11" =>
            Z <= not B;
      end case;
```

```
 end process P2;
end ARC2;


-- modello data-flow --
architecture DATA_FLOW1 of MUX_INV is
begin
 Z <= A          when (SEL='0' and INV='0') else
      not A      when (SEL='0' and INV='1') else
      B          when (SEL='1' and INV='0') else
      not B;
end DATA_FLOW1;


-- modello data-flow --
architecture DATA_FLOW2 of MUX_INV is
begin
 Z <=  (A and not SEL and not INV)or (not A and not SEL and INV)
      or (B and SEL and not INV) or (not B and SEL and INV);
end DATA_FLOW2;
```

7. Scrivere il modello VHDL *(comportamentale e dataflow)* del componente **DEC24** descritto dalla seguente specifica:

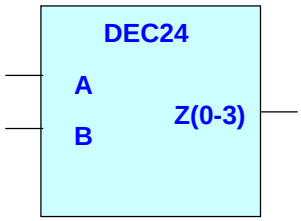


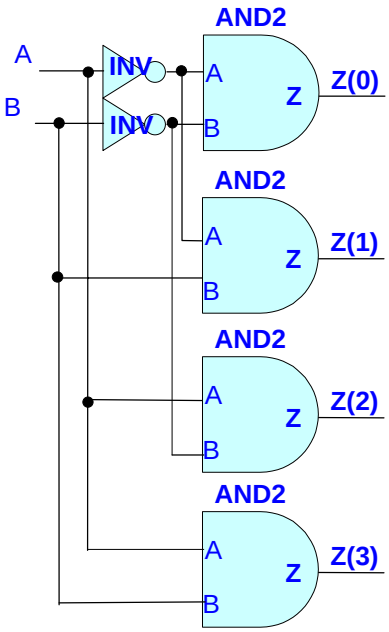
Ingressi		Uscita			
A	B	Z(0)	Z(1)	Z(2)	Z(3)
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

```
entity DEC24 is
port (A, B: in bit;
      Z: out bit_vector (0 to 3));
end DEC24;

-- modello comportamentale --
architecture ARC1 of DEC24 is
begin
 P1: process (A, B)
 begin
      if (A='0' and B='0') then
            Z <= "1000";
      elsif (A='0' and B='1') then
            Z <= "0100";
      elsif (A='1' and B='0') then
            Z <= "0010";
      elsif (A='1' and B='1') then
            Z <= "0001";
      end if;
 end process P1;
end ARC1;

-- modello comportamentale --
architecture ARC2 of DEC24 is
begin
 P2: process (A, B)
 begin
      case (A & B)  is
      when "00" =>
            Z <= "1000";
      when "01" =>
            Z <= "0100";
      when "10" =>
            Z <= "0010";
      when "11" =>
            Z <= "0001";
      end case;
 end process P2;
end ARC2;
```

```
-- modello data-flow --
architecture DATA_FLOW1 of DEC24 is
begin
 Z <= "1000"          when (A='0' and B='0') else
      "0100"          when (A='0' and B='1') else
      "0010"          when (A='1' and B='0') else
      "0001";
end DATA_FLOW1;


-- modello data-flow --
architecture DATA_FLOW2 of DEC24 is
begin
 Z(0) <=  (not A and not B);
 Z(1) <=  (not A and  B);
 Z(2) <=  (A and not B);
 Z(3) <=  (A and  B);
end DATA_FLOW2;
```

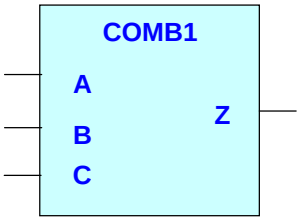
8. Scrivere il modello VHDL *(strutturale)* del componente **DEC24** descritto dalla seguente specifica:






```
entity DEC24 is
port (A, B: in bit;
      Z: out bit_vector (0 to 3));
end DEC24;

-- modello strutturale --
architecture STRUCT of DEC24 is
component inv
 port (A: in bit;
       Z: out bit);
end component;
component and2
 port (A, B: in bit;
       Z: out bit);
end component;
signal NOT_A, NOT_B: bit;

begin
 u1: inv port map (A, NOT_A);
 u2: inv port map (B, NOT_B);
 u3: and2 port map (NOT_A, NOT_B, Z(0));
 u4: and2 port map (NOT_A, B, Z(1));
 u5: and2 port map (A, NOT_B, Z(2));
 u6: and2 port map (A, B, Z(3));
end STRUCT;
```

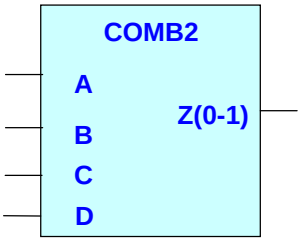
9. Scrivere il modello VHDL *(comportamentale)* del componente **COMB1** descritto dalla sequente specifica:



Ingressi			Uscita
A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

```
entity COMB1 is
port (A, B, C: in bit;
      Z: out bit);
end COMB1;

-- modello comportamentale
architecture ARC1 of COMB1 is
begin
 P1: process (A, B, C)
 begin
      case (A & B & C)  is
      when "000" =>
           Z <= '0';
      when "001" =>
           Z <= '1';
      when "010" =>
           Z <= '1';
      when "011" =>
           Z <= '0';
      when "100" =>
           Z <= '1';
      when "101" =>
           Z <= '0';
      when "110" =>
           Z <= '0';
      when "111" =>
           Z <= '1';
      end case;
 end process P1;
end ARC1;
```

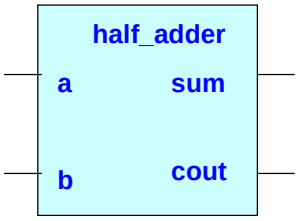
10. Scrivere il modello VHDL *(comportamentale)* del componente **COMB2** descritto dalla sequente specifica:



Ingressi				Uscite	
A	B	C	D	Z(0)	Z(1)
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	1	0
1	-	-	-	0	1

```
entity COMB2 is
port (A, B, C, D: in bit;
      Z: out bit_vector (0 to 1));
end COMB2;

-- modello comportamentale --
architecture ARC1 of COMB2 is
begin
 P1: process (A, B, C, D)
 begin
      case (A & B & C & D)  is
      when "0000" =>
            Z <= "00";
      when "0001" =>
            Z <= "00";
      when "0010" =>
            Z <= "00";
      when "0011" =>
            Z <= "00";
      when "0100" =>
            Z <= "00";
      when "0101" =>
            Z <= "10";
      when "0110" =>
            Z <= "00";
      when "0111" =>
            Z <= "10";
      when others =>
            Z <= "01";
      end case;
 end process P1;
end ARC1;
```

11. Scrivere il modello VHDL *(comportamentale e dataflow)* del componente **HALF_ADDER** descritto dalla sequente specifica:

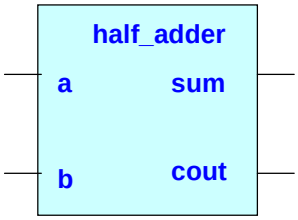


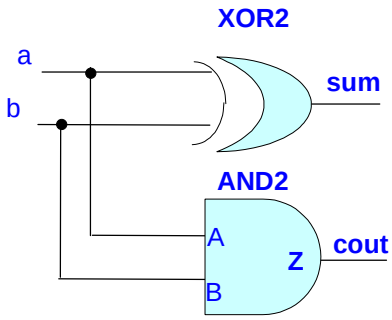
Ingressi		Uscite	
a	b	sum	cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

```
entity half_adder is
    port (a    : in bit;
          b    : in bit;
          sum  : out bit;
          cout : out bit);
end half_adder;

-- modello comportamentale --
architecture ARC1 of half_adder is
begin
 P1: process (a, b)
 begin
      if (a='0' and b='0') then
           sum <= '0'; cout <= '0';
      elsif (a='0' and b='1') then
           sum <= '1'; cout <= '0';
      elsif (a='1' and b='0') then
           sum <= '1'; cout <= '0';
      elsif (a='1' and b='1') then
           sum <= '0'; cout <= '1';
      end if;
 end process P1;
end ARC1;

-- modello comportamentale --
architecture ARC2 of half_adder is
begin
 P2: process (a, b)
 begin
      case (a & b)  is
      when "00" =>
           sum <= '0'; cout <= '0';
      when "01" =>
           sum <= '1'; cout <= '0';
      when "10" =>
           sum <= '1'; cout <= '0';
      when "11" =>
           sum <= '0'; cout <= '1';
      end case;
 end process P2;
end ARC2;
```

```
-- modello data-flow --
architecture DATA_FLOW1 of half_adder is
begin
 sum <= ‘0’      when ((a='0' and b='0') or (a='1' and b='1')) else
       ‘1’;
cout <= ‘1’      when (a='1' and b='1') else
       ‘0’;
end DATA_FLOW1;


-- modello data-flow --
architecture DATA_FLOW2 of half_adder is
begin
 sum <=a xor b ;
 cout <= a and b ;
end DATA_FLOW2;
```

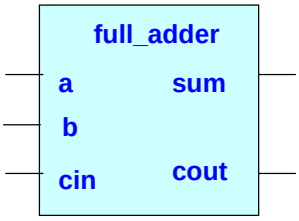
12. Scrivere il modello VHDL *(strutturale)* del componente **HALF_ADDER** descritto dalla sequente specifica:






```
entity half_adder is
    port (a    : in bit;
          b    : in bit;
          sum  : out bit;
          cout : out bit);
end half_adder;

-- modello strutturale --
architecture STRUCT of half_adder is
component and2
 port (A, B: in bit;
       Z: out bit);
end component;
component xor2
 port (A, B: in bit;
       Z: out bit);
end component;
begin
 u1: xor2 port map (a, b, sum );
 u2: and2 port map (a, b, cout);
end STRUCT;
```

13. Scrivere il modello VHDL *(comportamentale e dataflow)* del componente **FULL_ADDER** descritto dalla sequente specifica:

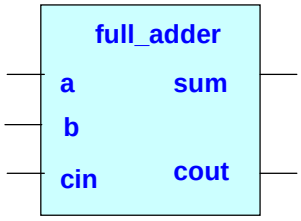


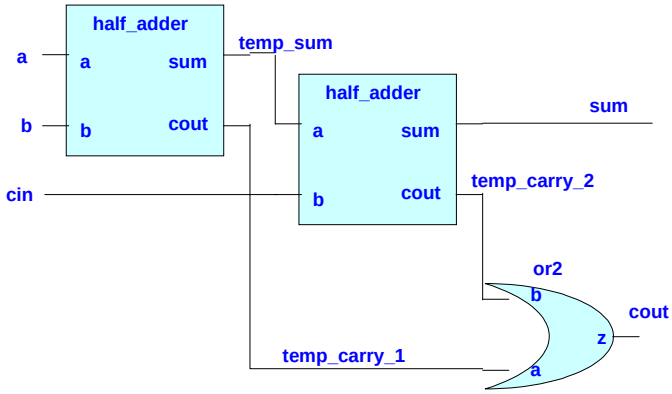
Ingressi			Uscite	
a	b	cin	sum	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

```
entity full_adder is
    port (a      : in bit;
          b      : in bit;
          cin    : in bit;
          sum    : out bit;
          cout   : out bit);
end full_adder;

-- modello comportamentale
architecture ARC1 of full_adder is
begin
 P1: process (a, b,cin)
 begin
       case (a & b & cin)  is
       when "000" =>
             sum <= '0'; cout <= '0';
       when "001" =>
             sum <= '1'; cout <= '0';
       when "010" =>
             sum <= '1'; cout <= '0';
       when "011" =>
             sum <= '0'; cout <= '1';
       when "100" =>
             sum <= '1'; cout <= '0';
       when "101" =>
             sum <= '0'; cout <= '1';
       when "110" =>
             sum <= '0'; cout <= '1';
       when "111" =>
              sum <= '1'; cout <= '1';
       end case;
 end process P1;
end ARC1;
```

```
-- modello data-flow --
architecture DATA_FLOW of full_adder is
 signal temp_sum: bit;
begin
       temp_sum <=a xor b ;
       sum <= temp_sum xor cin ;
       cout <= (a and b) or (temp_sum and cin) ;
end DATA_FLOW;
```

14. Scrivere il modello VHDL *(strutturale)* del componente **FULL_ADDER** descritto dalla sequente specifica:






```
entity full_adder is
    port (a      : in bit;
          b      : in bit;
          cin    : in bit;
          sum    : out bit;
          cout   : out bit);
end full_adder;

-- modello strutturale --
architecture STRUCT of full_adder is

    component half_adder
         port (a, b : in bit; sum, cout : out bit);
    end component;

    component or2
         port (a, b: in bit; z: out bit);
    end component;

    signal temp_sum, temp_carry_1, temp_carry_2: bit;

begin
    u1: half_adder port map(a, b, temp_sum, temp_carry_1);
    u2: half_adder port map(temp_sum, cin, sum , temp_carry_2);
    u3: or2 port map(temp_carry_1, temp_carry_2, cout);
end STRUCT;
```