Architetture e reti logiche

Esercitazioni VHDL

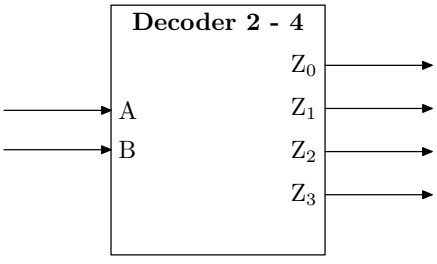
a.a. 2008/09

Componenti combinatori in VHDL

Stefano Ferrari

UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI TECNOLOGIE DELL'INFORMAZIONE

Decoder a due ingressi



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;
```

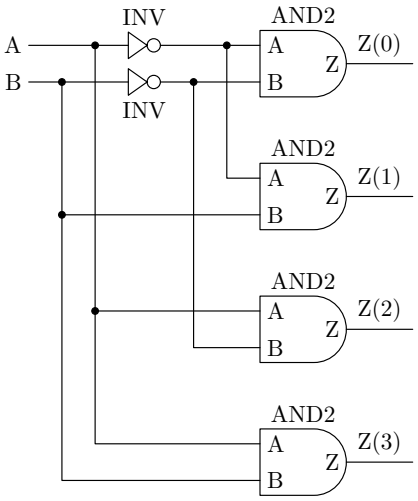
Decoder: descrizione dataflow

```
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;
```


Decoder: descrizione dataflow (2)

```
architecture DATA_FLOW2 of DEC24 is
  signal t: bit_vector(0 to 1);
begin
  t <= A & B;
  with t select
    Z <= "1000" when "00",  -- 0
         "0100" when "01",  -- 1
         "0010" when "10",  -- 2
         "0001" when "11";  -- 3
end DATA_FLOW2;
```

Decoder: rappresentazione circuitale




```
architecture DATA_FLOW3
  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_FLOW3;
```


Decoder: descrizione 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
```

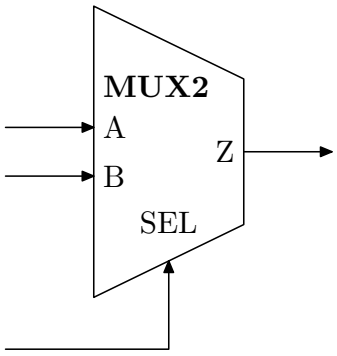
(continua)

Decoder: descrizione strutturale (2)

(continua)

```
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;
```


Multiplexer a due vie



SEL	A	B	Z
0	0	—	0
0	1	—	1
1	—	0	0
1	—	1	1

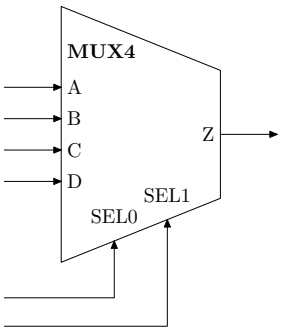
```
entity MUX2 is
  port (A, B, SEL: in bit;
                Z: out bit);
end MUX2;
```

Multiplexer: descrizione dataflow

```
architecture DATA_FLOW1 of MUX2 is
begin
  Z <= A when (SEL = '0') else
       B ;
end DATA_FLOW1;

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


Multiplexer a 4 vie



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

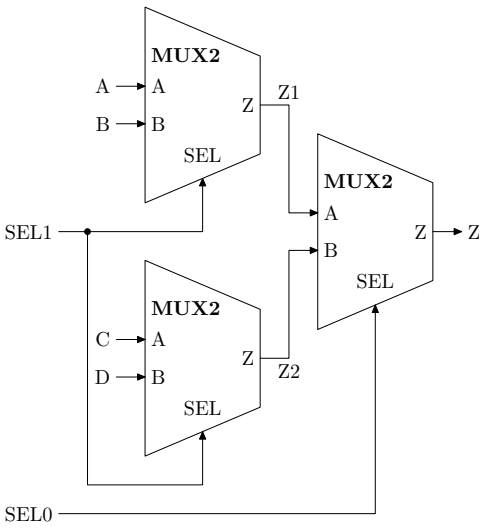
```
entity MUX4 is
  port (A, B, C, D, SEL0, SEL1: in bit;
                          Z: out bit);
end MUX4;
```

Multiplexer a 4 vie: dataflow

```
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;

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;
```

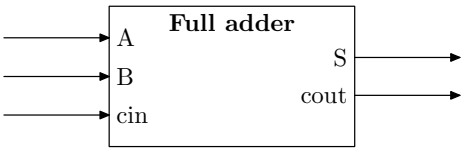

Multiplexer 4: descrizione strutturale



Multiplexer 4: descrizione strutturale (2)

```
architecture STRUCT2 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 STRUCT2;
```


Full adder



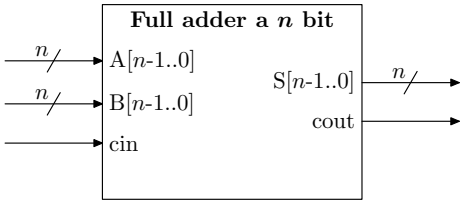
A	B	cin	S	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

```
entity fulladd is
  port (A, B : in bit;    -- addendi
        cin  : in bit;    -- riporto in input
        S    : out bit;   -- somma (output)
        cout : out bit);  -- riporto in output
end fulladd;
```

Full adder: descrizione dataflow

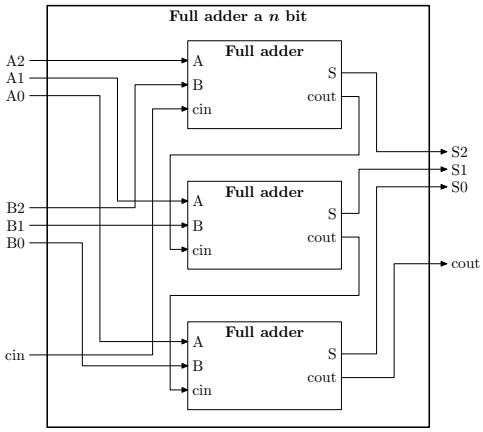
```
architecture dataflow of fulladd is
begin
  S <= A xor B xor cin;
  cout <= (A and B) or (A and cin) or (B and cin);
end dataflow;
```


Full adder a n bit




```
entity fulladd is
  port (A, B : in bit_vector(2 downto 0);  -- addendi
        cin  : in bit;                -- riporto in input
        S    : out bit_vector(2 downto 0);   -- somma
        cout : out bit);              -- riporto in output
end fulladd;
```

Full adder a 3 bit: modello strutturale



Full adder a 3 bit: modello strutturale (2)

```
architecture struct of fulladd3 is
  component fulladd
    port (A, B  : in std_logic;
          cin   : in std_logic;
          S     : out std_logic;
          cout  : out std_logic);
  end component;

  signal t   : std_logic_vector(0 to 3);

begin
```

(continua)

Full adder a 3 bit: modello strutturale (3)

(continua)

```
begin
  t(0) <= cin;
  u0: fulladd port map (A(0), B(0), t(0), S(0), t(1));
  u1: fulladd port map (A(1), B(1), t(1), S(1), t(2));
  u2: fulladd port map (A(2), B(2), t(2), S(2), t(3));
  cout <= t(3);
end struct;
```