Esempi di reti combinatorie in VHDL

Esercitazione 04 di Sistemi dedicati

Docente: Giuseppe Scollo

Università di Catania
Dipartimento di Matematica e Informatica
Corso di Laurea Magistrale in Informatica, AA 2019-20

Indice

argomenti dell'esercitazione

in questa esercitazione si trattano descrizioni in VHDL di componenti hardware di frequente impiego:

- porte tri-state
- decodificatori:
  - decodificatori a *n* ingressi di selezione
  - decodificatori per display a 7 segmenti
- multiplatori
- funzioni di ALU

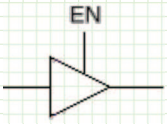
vari aspetti e costrutti del linguaggio VHDL vengono introdotti nel contesto degli esempi proposti

inoltre si propone un'esperienza di laboratorio in cui confluiscono diversi aspetti di VHDL illustrati dagli esempi qui presentati e alcuni componenti possono essere riusati per la realizzazione dell'esperimento proposto

porte tri-state

il tipo standard BIT, a due valori booleani, non è sufficiente a rappresentare tutte le situazioni che possono presentarsi nella progettazione di circuiti

e.g. si può disconnettere dinamicamente una linea dal circuito, permettendo l'accesso di più driver a un bus, uno per volta



porta tri-state

le porte tri-state hanno questo impiego tipico

lo standard IEEE 1164 definisce il tipo std_ulogic a nove valori:

'U' Uninitialized
'X' Forcing unknown
'0' Forcing 0
'1' Forcing 1
'Z' High impedance
'W' Weak unknown
'L' Weak 0
'H' Weak 1
'–' Don't care

ma std_ulogic non ammette l'accesso di più driver su una stessa linea

il suo sottotipo std_logic è dotato di una funzione di risoluzione della contesa che dunque lo permette

	U	X	0	1	Z	W	L	H	–
U	U	U	U	U	U	U	U	U	U
X	U	X	X	X	X	X	X	X	X
0	U	X	0	X	0	0	0	0	X
1	U	X	X	1	1	1	1	1	X
Z	U	X	0	1	Z	W	L	H	X
W	U	X	0	1	W	W	W	W	X
L	U	X	0	1	L	W	L	W	X
H	U	X	0	1	H	W	W	H	X
–	U	X	X	X	X	X	X	X	X

risoluzione della contesa fra valori del tipo std_logic

```
library ieee;
use ieee.std_logic_1164.all;
entity tri_state is
  port (x, en : in std_logic;
        y : out std_logic);
end entity tri_state;
architecture when_else of tri_state is
begin
  y <= x when en = '1' else 'Z';
end architecture when_else;
```

decodificatori

unità funzionale utile per diversi impieghi, per esempio:

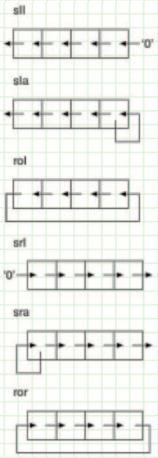
- selezione di un chip o di una riga di celle memoria
- decodifica del codice operativo di un'istruzione macchina

specifica VHDL del decodificatore 2→4

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

entity decoder2 is
  port (
    a : in std_logic_vector(1 downto 0);
    z : out std_logic_vector(3 downto 0)
  );
end entity decoder2;

architecture when_else of decoder2 is
begin
  z <= "0001" when a = "00" else
       "0010" when a = "01" else
       "0100" when a = "10" else
       "1000" when a = "11" else
       "XXXX";
end architecture when_else;
```

specifica VHDL generica del decodificatore (Zwoliński, Sect. 4.2.3):

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

entity decoder is
  generic (n : POSITIVE);
  port (
    a : in std_logic_vector(n-1 downto 0);
    z : out std_logic_vector(2**n-1 downto 0)
  );
end entity decoder;

architecture rotate of decoder is
  constant z_out : BIT_VECTOR(2**n-1 downto
0) :=
    (0 => '1', others => '0');
begin
  z <= to_StdLogicVector (z_out sll
                          to_integer(unsigned(a)));
end architecture rotate;
```



operatori di scorrimento VHDL

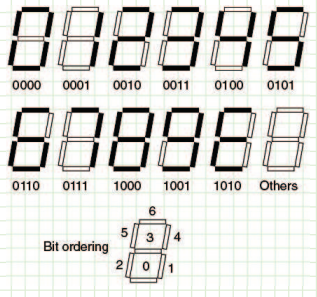
decodificatori per display a 7 segmenti

dispositivo familiare di uso quotidiano ...

un decodificatore per display a 7 segmenti converte un input a 4 bit nella configurazione binaria dei 7 bit, associati ai segmenti del display, che visualizza il carattere corrispondente alla cifra (esadecimale o decimale) che ha il valore codificato dall'input

l'esempio che segue (tratto da Zwoliński, Sect. 4.2.3)

- visualizza cifre decimali
- visualizza E se il valore dell'input è ≥ 10
- in tutti gli altri casi il display è oscurato



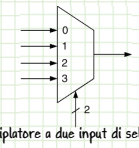
Zwoliński, Figure 4.3 - Seven-segment display

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

entity seven_seg is
  port (a : in std_logic_vector(3 downto 0);
        z : out std_logic_vector(6 downto 0));
end entity seven_seg;

architecture with_select of seven_seg is
begin
  with a select
    z <= "1110111" when "0000",
         "0010010" when "0001",
         "1011101" when "0010",
         "1011011" when "0011",
         "0111010" when "0100",
         "1101011" when "0101",
         "1101111" when "0110",
         "1010010" when "0111",
         "1111111" when "1000",
         "1111011" when "1001",
         "1101101" when "1010"|"1011"|"1100"|"1101"|"1110"|"1111",
         "0000000" when others;
end architecture with_select;
```

multiplatori

specifica VHDL di un multiplatore a singolo input di selezione:

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

entity mux1 is
 port (
   s : in std_logic;
   a : in std_logic;
   b : in std_logic;
   y : out std_logic
 );
end entity mux1;

architecture basic of mux1 is
begin
 y <= a when s = '0' else
      b when s = '1' else
      'X';
end architecture basic;
```



multiplatore a due input di selezione

come generalizzare a *n* ingressi di selezione?

specifica di una porta OR a *n* ingressi:

```
library ieee;
use ieee.std_logic_1164.all;
entity wide_or is
 generic (n : POSITIVE);
 port (
     x : in std_logic_vector(n-1 downto 0);
     y : out std_logic);
end entity wide_or;
```

multiplatore a *n* ingressi di selezione

```
library ieee;
use ieee.std_logic_1164.all;
entity mux is
 generic (n : POSITIVE := 1);
 port (
   s : in std_logic_vector(n-1 downto 0);
   a : in std_logic_vector(2**n-1 downto 0);
   z : out std_logic);
end entity mux;
```

istanziazione del multiplatore generico

```
library ieee;
use ieee.std_logic_1164.all;
entity mux2 is
 port (
   s : in std_logic_vector(1 downto 0);
   a : in std_logic_vector(3 downto 0);
   z : out std_logic);
end entity mux2;
```

```
architecture sequential of wide_or is
begin
 process (x) is
  variable z : std_logic;
 begin
  z := x(0);
  if n > 1 then
   for i in 1 to n-1 loop
    z := x(i) or z;
   end loop;
  end if;
  y <= z;
 end process;
end architecture sequential;
```

```
architecture structural of mux is
 signal decout : std_logic_vector(2**n-1 downto 0);
 signal andout : std_logic_vector(2**n-1 downto 0);
begin
 di: entity WORK.decoder generic map (n)
     port map (a => s, z => decout);
 oi: entity WORK.wide_or generic map (2**n)
     port map (x => andout, y => z);
 andout <= a and decout;
end architecture structural;
```

```
architecture instance of mux2 is
begin
 di: entity WORK.mux generic map (2)
     port map (s, a, z);
end architecture instance;
```


funzioni di ALU

una ALU è una unità multifunzione: un input di controllo seleziona l'operazione da eseguire

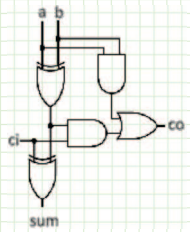
S	funzione
00	Q <= A or B
01	Q <= A and B
10	Q <= not B
11	Q <= A xor B

un esempio di unità logica multifunzione ha la selezione specificata in tabella e può descriversi in VHDL come segue:

```
library ieee;
use ieee.std_logic_1164.all;
entity alu_logic is
 generic ( n : POSITIVE := 16 );
 port (
   a : in std_logic_vector((n−1) downto 0);
   b : in std_logic_vector((n−1) downto 0);
   s : in std_logic_vector(1 downto 0);
   q : out std_logic_vector((n−1) downto 0)
 );
end entity alu_logic;
```

```
architecture with_select of alu_logic is
 constant undefined : std_logic_vector(n-1 downto 0) := (others => 'X');
begin
 with s select
  q <= a or b    when “00”,
       a and b when “01”,
       not b;    when “10”,
       a xor b  when “11”,
       undefined when others;
end architecture with_select;
```

per operandi di più bit, la funzione aritmetica dell'ALU può specificarsi in stile strutturale o funzionale; quest'ultimo come segue



full-adder da 1 bit

```
library ieee;
use IEEE.std_logic_1164.all;
entity adder is
 generic(n : POSITIVE := 16);
 port (
  a : in std_logic_vector (n-1 downto 0);
  b : in std_logic_vector (n-1 downto 0);
  ci : in std_logic;
  sum : out std_logic_vector (n-1 downto 0);
  co : out std_logic
 );
end entity adder;
```

```
architecture sequential of adder is
begin
 process(a,b,ci)
  variable carry : std_logic;
  variable psum : std_logic_vector(n-1 downto 0);
 begin
  carry := ci;
  for i in 0 to n-1 loop
   psum(i) := a(i) xor b(i) xor carry;
   carry := (a(i) and b(i)) or (((a(i) xor b(i)) and carry);
  end loop;
  sum <= psum;
  co <= carry;
 end process;
end architecture sequential;
```

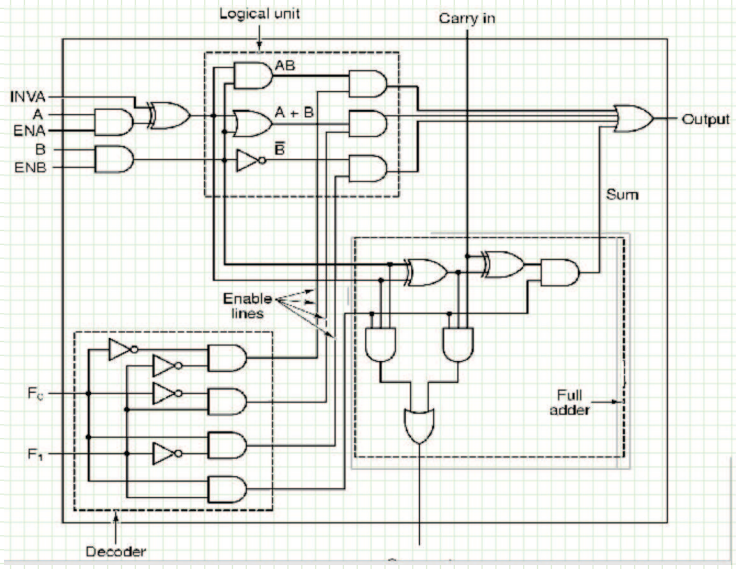
il simulatore di ALU a 8 bit realizzato da S. Lentini e G. Nicotra illustra come si possa costruire iterativamente l'ALU a 8 bit componendo opportunamente stadi di ALU a 1 bit

> lo schematico e un'illustrazione animata delle funzioni dell'ALU sono presentate nella parte "Unità Aritmetica Logica" dell'animazione Flash del progetto

usando i costrutti VHDL visti negli esempi di questa esercitazione:

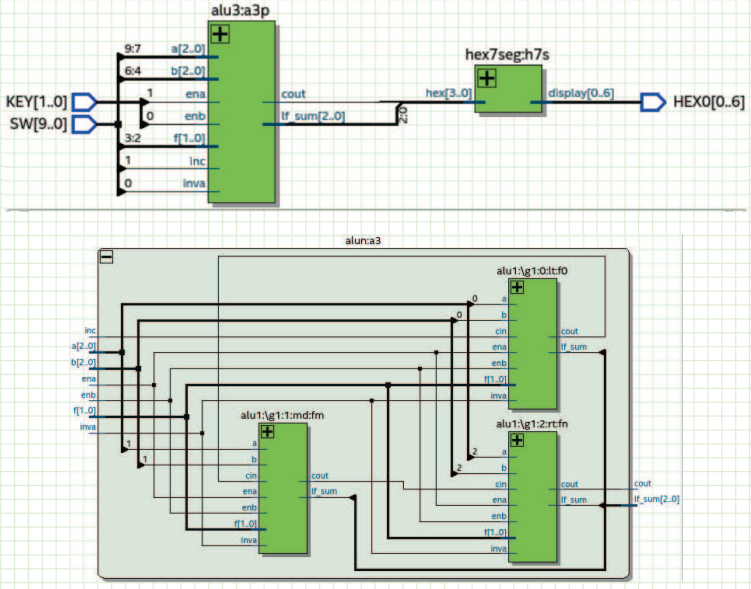
1. costruire un modello VHDL dell'ALU a 1 bit; una soluzione equivalente a quella rappresentata dallo schematico riprodotto appresso può ottenersi impiegando opportunamente un multiplatore in luogo del decodificatore
2. sulla base di tale risultato, costruire un modello generico dell'ALU a *n* bit
3. comporre un'istanza a 3 bit del modello generico di ALU con un modello VHDL di decodificatore di cifre esadecimali per la visualizzazione su display a 7 segmenti (disponibile nell'area riservata di laboratorio), mappando le porte di I/O del modello ottenuto sui pin della FPGA DE1-Soc, come indicato appresso e illustrato nel successivo schematico RTL
4. programmare la FPGA usando i suoi 10 switch e 2 tasti per l'input alla ALU (i 2 tasti per gli input ENA, ENB, 6 switch per gli operandi A, B, gli switch restanti per gli altri 4 input di controllo) e un display a 7 segmenti per la visualizzazione del risultato (riporto incluso, per l'addizione)
5. collaudare il funzionamento dell'ALU a 3 bit sulla FPGA agendo sugli switch e tasti assegnati

schematico RTL di ALU a 1-bit



schematico di ALU a 1 bit (Figura 3-19 in (Tanenbaum, 2006))

schematico RTL di ALU a 3-bit su FPGA



schematico RTL di ALU a 3 bit su FPGA generato da Quartus Prime Lite 16.1

riferimenti

letture raccomandate: Zwoliński, Ch. 4, Sect. 4.4-4.6

materiali utili per l'esperienza di laboratorio proposta
(fonti: Biblioteca DMI, Intel Corp. - FPGA University Program, 2016)

- Tanenbaum: *Architettura dei calcolatori, Quinta Edizione* (Pearson, 2006) Sez. 3.2.3
- Making Qsys Components - For Quartus Prime 16.1, Appendix A
- Quartus Prime Introduction Using VHDL Designs - For Quartus Prime 16.1, Sect. 9
- DE1-SoC Quartus Setting File with Pin Assignments
- Problem when programming Cyclone V (DE1-SoC)

sorgenti VHDL (nello spazio riservato di laboratorio):

- codice degli esempi in questa presentazione
- esempi nel testo di Zwoliński