可编程逻辑电路设计

Digital Design Using PLD

可编程逻辑电路设计教学组

二〇〇六年

PLD设计的VHDL语言实现

（概述及组合逻辑部分）

Hardware Description Language (HDL)

- 数字系统的描述方式

设计层次	行为描述	结构描述
系统级	算法流程图	由Cpu或控制器、存储器等组成的逻辑框图
寄存器传输级（RTL）	数据流图、真值表、状态机、状态图（表）	寄存器、ALU、MUX、ROM等组成的逻辑图
逻辑门级	布尔方程、真值表、卡诺图	逻辑门、触发器、锁存器等构成的逻辑图（网表）

- HDL语言既包含一些高级程序设计语言的结构形式，也兼顾描述硬件线路连接的构件，可以支持从系统级到门级的各个层次的行为描述和结构描述。
- HDL语言基本特征是并发的（硬件的基本特征），但也提供顺序功能的描述手段。
- HDL语言特点：
  - 文本输入
  - 适于设计大系统——表达功能，隐藏细节
  - 高表达效率
  - 易修改、开发周期短
  - 通用语言，模块可重用性好
- 常用HDL语言：VHDL、Verilog-HDL、AHDL

VHDL语言

- Very high speed integrated circuit HDL
- IEEE工业标准HDL语言
- 可支持仿真与综合
- 两个版本：
  - 1076－1987
  - 1076－1993

VHDL程序结构

- **1、USE定义区**
  - Library——定义所使用的元件库
  - Package——定义所使用的元件库中的包
- **2、Entity定义区：定义电路实体的I/O接口规格**
- **3、Architecture定义区：描述电路内部具体功能**
  - Component定义区
  - 信号定义
  - 行为描述/数据流描述/结构描述
- **4、Configuration定义区：决定使用哪一个architecture（非必须）**

Use定义区

```
Library IEEE;                          --库定义
Use IEEE.std_logic_1164.all;           --包引用
Use IEEE.std_logic_arith.all;          --包引用
```

引用语句的用法：

```
Library <library_name>,<library_name> ;
Use lib_name.pack_name.object;
```

Packages

Package <Package_name> is
Constant Decclarations
Type Declarations
Signal Declarations
Subprogram Declarations
Component Declarations
Other Declarations
End <package_name> ; **(1076-1987)**
End Package <package_name>; **(1076-1993)**

Package Body <Package_name> is
Constant Declarations
Type Declarations
Subprogram Body
End Package <package_name>; **(1076-1987)**
End Package Body <Package_name> **(1076-1993)**

Package Example

```
package  package_example is
    type life is (sleeping, working, eating, entertaiment, otheractions);
    subtype uint4 is integer range 0 to 15;
    subtype uint5 is integer range 0 to 31;
    function compare(a, b: integer) return boolean;
end package_example;

package body package_example is
    function compare(a,b: integer) return boolean is
        variable temp:boolean;
    begin
        if a<b then
            temp:=true;
        else
            temp:=false;
        end if;
        return temp;
    end compare;
end package_example;
```

Libraries

- 包括一系列的packages
- 隐含Libraries: 不用声明，自动引用
  - STD：
    - Standard:定义bit, Boolean, integer, real和time以及支持它们的运算符。
    - Textio:定义文件操作。
  - Work:
  - IEEE:
    - Std_logic_1164
    - Std_logic_arith
    - Std_logic_signed
    - Std_logic_unsigned
  - 其它库:
    - Altera的元件库
    - 用户自定义库

Entity（实体）

- Entity定义语法

```
Entity <Entity_name>  is
    generic declarations
    port declarations
End <Entity_name>;
```

- Entity Example

```
Entity adder is
    generic(data_width:integer:=4 );
    port(
        add_a, add_b: in std_logic_vector(data_width-1 downto 0);
        sum: out std_logic_vector(data_width downto 0) );
End adder;
```

Generic & Port Declarations

- **Generic Declarations**
  - 用于将参数传入Entity。例如：数据线宽度，器件的延时参数、负载电容电阻、驱动能力、功耗等等。

- **Port Declarations： port_name: <mode> : <type>**
  - <mode>：管脚的模式
    - In (输入)
    - Out (输出)
    - Buffer (输出带内部反馈)
    - Inout (双向)
  - <Type>: 数据类型
    - Boolean: False, True
    - Bit: '0', '1'
    - Std_logic: 'X', '0', '1', 'Z', '-', 'W', 'L', 'H'，支持多信号判决
    - Std_ulogic: 与Std_logic类型一样，不支持多信号判决
    - Bit_vector & std_logic_vector:
    - Integer:
      - Natural
      - Positive
    - Real: time
    - 自定义数据类型：如数组等

Architecture（结构体）

- **描述设计的功能或者结构**
- **必须与某个entity相联系**
- **一个entity可以对应多个architecture(编译时通过configuration来指定)**
- **Architecture中的各个语句是并发执行的，对应于电路硬件中的不同部件**
- **Architecture的描述风格**
  - 行为描述——描述实体的功能，RTL级
    - 算法描述
    - 数据流描述
  - 结构描述——描述实体的结构，门级
  - 混合描述

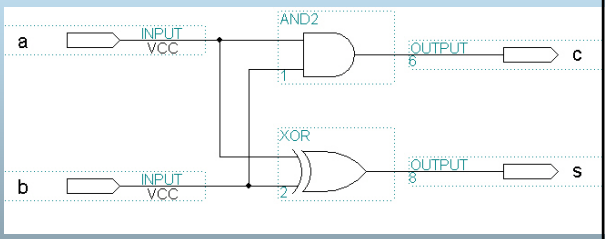
Architecture组成

❑ Architecture定义

```
ARCHITECTURE arch_name OF entity_name IS
    signal declarations;
    constant declarations;
    type  & subtype declarations;
    component declarations;
    subprogram declarations;
BEGIN
    Process Statements;
    Concurrent Procedure Calls;
    Concurrent Signal Assignment;
    Conditional Signal Assignment;
    Selected Signal Assignment;
    Component Instantiation Statements;
    Generate Statements;
END arch_name;
```

结构描述例子——半加器

```
Architecture struct_ha of  half_adder is
    component and_gate
        port( a1,a2: in std_logic;
              a3: out std_logic);
    end component;
    component xor_gate
        port( a1,a2: in std_logic;
              a3: out std_logic);
    end component;
Begin
    g1: and_gate port map(a,b,c);
    g2: xor_gate port map(a,b,s);
End struct_ha;
```



行为描述例子——半加器

- **算法描述**

```
Architecture behave_ha of  half_adder is
Begin
 g1: process(a,b)
     begin
        if a='1' and b='1' then
              c<='1';
        else
              c<='0';
        end if;
     end process;
 g2: process(a,b)
     begin
        if a='1' and b='0' then
              c<='1';
        elsif a='0' and b='1'
              c<='1';
        else
              c<='0';
        end if;
     end process;
End struct_ha;
```

- **数据流描述**

```
Architecture behave_ha of  half_adder is
Begin
      c<= a and b;
      s<=a xor b;
End struct_ha;
```

Configuration

- 用于将**entity**和**architecture**联系起来
- 广泛应用于仿真环境
- 被综合器有限支持或者不支持
- 语法：

```
Configuration <config_name> of <entity_name> is
    for <architecture_name>
    end for;
End;
```

数据对象

- **Constant(常数)**
  - 定义语法： Constant <const_name>: <type> := <设定值>
  - 用符号代表常数，增加程序的可读性和可维护性
  - 一经定义，不得更改
- **Signal(信号)**
  - 定义语法： Signal <signal_name>: <type> := <设定值>
  - 对应于电路中的某一节点
  - 通常定义于entity、architecture或者package中，全局使用
  - 可再赋值 ： “ <=”
  - 进程中对信号的赋值在该进程结尾才生效
- **Variable(变量)**
  - 定义语法： Variable <variable_name>: <type> :=<设定值>
  - 不对应于具体电路中的节点，一般用于计算用途
  - 只局限于进程和子程序中定义并使用
  - 可再赋值： “:=”
  - 对变量的赋值立即生效

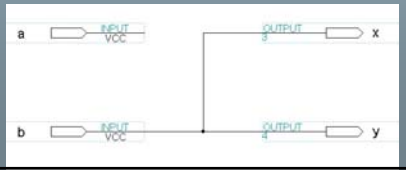
信号和变量的区别——例1

- 一段程序:

```
a, b: in std_logic;
x, y: out std_logic;

signal  temp: std_logic;
process (a, b, temp)
begin
        temp<=a;
        x<=temp;
        temp<=b;
        y<=temp;
end process;
```

- 等价电路图：

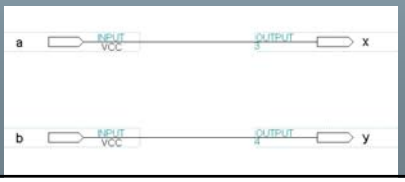


- 另一段程序:

```
a, b: in std_logic;
x, y: out std_logic;

process (a, b, temp)
  variable temp: std_logic;
begin
        temp:=a;
        x<=temp;
        temp:=b;
        y<=temp;
end process;
```

- 等价电路图：



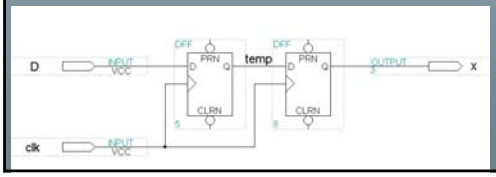
信号和变量的区别——例2

- 一段程序:

```
D, clk: in std_logic;
x: out std_logic;

signal  temp: std_logic;
process (clk)
begin
    if (clk'event and clk='1') then
        temp<=D;
        x<=temp;
    end if;
end process;
```

- 等价电路图：

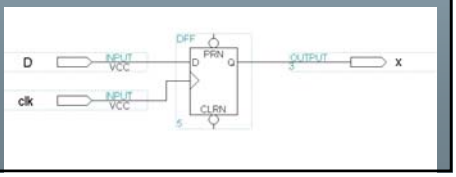


- 另一段程序:

```
D, clk: in std_logic;
x: out std_logic;

process (clk)
    variable temp: std_logic;
begin
    if (clk'event and clk='1') then
        temp:=D;
        x<=temp;
    end if;
end process;
```

- 等价电路图：



数据类型

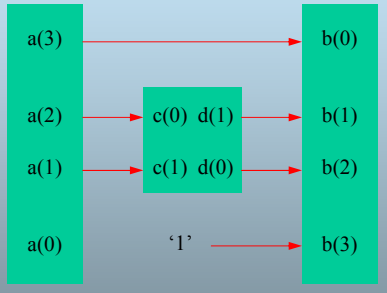
- **基本数据类型**
  - 逻辑类型: bit, boolean, std_logic, std_ulogic, std_logic_vector, bit_vector
  - 整数类型: Integer(及其子类如Natural, Positive), unsigned, signed
  - 实数类型: Real
- **高级数据类型**
  - 枚举类型：
    - 定义语法：TYPE <enumerated_type_name> IS (<name>, <name>, <name>)
    - 使程序易读，例如在定义状态机时
  - 数组类型：
    - 定义语法:TYPE <array_type_name> IS ARRAY (integer1 DOWNTO integer2) OF <type_name>
    - 多用于定义ROM、RAM等

运算符

- 关系运算符：
  - > < >= <= = /=
- 逻辑运算符：
  - and, or, not, nand, nor, xor
- 算术运算符：
  - ＋ － * / MOD REM ** ABS
- 其他运算符：
  - ＋(正号) －(负号) &(连接)
- 运算符重载
  - 对于不同的数据类型定义同样一个运算，其实际操作可能不同。例如+(加)运算，可以对整数使用，也可以对std_loigc_vector等类型用(在std_logic_arith包中定义)。
  - 通常定义与package中。
  - IEEE的package中定义了大量的重载运算符

Vector信号的分解与合并

```
Architecture rtl of test is
    signal a: std_logic_vector(3 downto 0);
    signal b: std_logic_vector(0 to 3);
    signal c: std_logic_vector(0 to 1);
    signal d: std_logic_vector(1 downto 0);
Begin
    c<=a(2 downto 1);
    b<=A(3) & D & '1'
End rtl;
```



并发语句

- **VHDL程序的结构体中的各个语句是并发的，代表电路的不同节点，它们是同时运作的**
- **并发语句的顺序不影响其执行结果**
- **并发语句列表：**
  - 直接信号赋值： <=
  - 条件信号赋值：when--else
  - 选择信号赋值：select—when
  - 进程：process
  - 断言：assert——面向仿真的语句，不能综合
  - 块语句：Block
  - Component语句
  - For-Generate

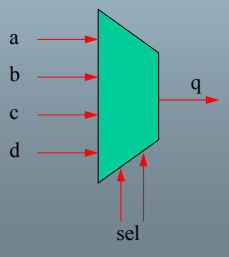
When--else

- **语法：**

```
<signal_name> <= <value1> when <condition1> else
                 <value2> when <condition2> else
                 ……..
                 <value_n> when <condition_n> else
                 <value_n1>;
```

- **特点：**
  - 各个条件可以不互斥
  - 主要用于译码器、多路复接器和解复接器
- **例子：**

```
q <=   a when sel="00" else
       b when sel="01" else
       c when sel="10" else
       d;
```



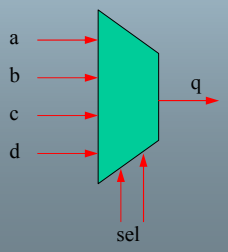
Select--when

- 语法：

with <expression> select
<signal_name> <= <value1> when <condition1> ,
<value2> when <condition2> ,
……..
<value_*n*> when others;

- 特点：
  - 各个条件必须互斥
  - 主要用于译码器、多路复接器和解复接器
- 范例：

with sel select
q <= a when “00”,
b when “01”,
c when “10”,
d when others;



Block

- 用于将电路划分成几个模块，增加程序的可读性
- 语法：

<block_name> block
<数据对象定义区>
Begin
<并发命令区块>
End block <block_name>;

- 范例：

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity half_addsub is
    port(  a,b: in std_logic;
           sum, carry: out std_logic;
           diff, borrow: out std_logic);
End half_addsub;

Architecture rtl of half_addsub is
Begin

Half_adder: block
     sum<=a xor b;
     carry<=a and b;
End block half_adder;

Half_sub: block
     diff<=a xor b;
     borrow<=(not a) and b;
End block half_sub;

End rtl;
```

Component

- **Component定义语句：与entity类似**

```
Component <component_name>
        generic ( generic_declarations );
        port ( port_declarations);
End component;
```

- **Component例化语句：**

```
<example_name>:  <component_name>
        generic map ( generic_mapping)
        port map (port_mapping);
```

- **Port map语句两种格式：**
  - Port map (<port1>=><signal1>, <port2>=><signal2>, …);
  - Port map (<signal1>, <signal2>, …);

Component范例——4位全加器

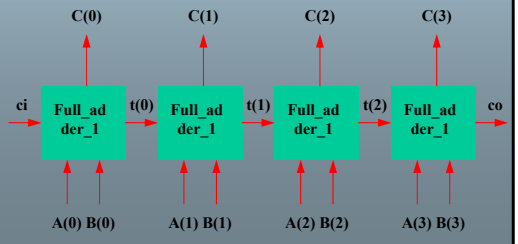
```
Library ieee;
Use ieee.std_logic_1164.all;

Entity full_adder_4 is
    port(
        A, B: in std_logic_vector(3 downto 0);
        ci: in std_logic;
        C: out std_logic_vector(3 downto 0);
        co: out std_logic);
End full_adder_4;
Architecture rtl of full_adder_4 is
    component full_adder_1
    port(
            a,b: in std_logic;
            ci: in std_logic;
            c: out std_logic;
            co: out std_logic);
    end component;
    signal t: std_logic_vector(2 downto 0);
Begin
    U0: full_adder_1
    port map(A(0),B(0),ci,C(0),t(0));

    U1: full_adder_1
    port map(A(1),B(1),t(0),C(1),t(1));

    U2: full_adder_1
    port map(A(2),B(2),t(1),C(2),t(2));

    U3: full_adder_1
    port map(A(3),B(3),t(2),C(3),co);
End rtl;
```



For-Generate

- 例化多个相同的Component
- 语法：

```
<gen_name>: For i in <start_value> to <stop_value> generate
    component例化语句；
End generate <gen_name>;
```

- 范例: 4位全加器

```
Architecture rtl of full_adder_4 is
    <component full_add_1 declaration>;
    signal t: std_logic_vector(4 downto 0);
Begin
    gen: for I in 0 to 3 generate
      add1: full_adder_1 port map(A(i),B(i),t(i),C(i),t(i+1));
    end generate gen;
    t(0)<=ci;
    co<=t(4);
End rtl;
```

- If-Generate语句可以条件生成。
- 生成语句可以嵌套。

顺序语句

- 可用于进程、函数和子程序中
- 依次执行，语句顺序影响其执行结果
- 顺序语句的执行需要被激活
- 顺序语句列表：
  - 变量和信号赋值：:=　<=
  - If语句：if--then--elsif--else
  - Case语句：case--when
  - Loop语句
  - Wait语句

Process

- 特点：
  - Process语句本身是一个并发式语句，但其内部包含的是顺序语句
  - 仅当进程的某个敏感信号发生变化时，进程内部的顺序语句块才被激活依次执行；否则进程处于被挂起的状态。
  - 进程内的变量在进程被挂起和再次激活时，保持原值。
  - 可以通过两种方式定义敏感信息：用敏感信息表或者用wait语句。但这两者不能同时出现在一个进程中(wait语句可以有多个)。
- 定义语法：

```
Process(<sensitivity_list>)
    constant declaration;
    type declarations;
    variable declarations;
Begin
    sequential statement #1;
    sequential statement #2;
            …….
    sequential statement #n;
End process;
```

Process范例

```
Process(clk)
Begin
    if (clk‘event and clk=‘1’) then
        q<=d;
    end if;
End process;


Process
Begin
    wait on clk;
    if clk’event and clk=‘1’ then
        nq<=not d;
    end if;
End process;
```

Process——敏感列表

（1）正常的代码

```
Process (ck, d)
Begin
    if ck='1' then
        q<=d;
    end if;
End process;
```

Max+plus II综合结果：Latch
FPGA Express综合结果: Latch

（2）危险的代码

```
Process (ck)
Begin
    if ck='1' then
        q<=d;
    end if;
End process;
```

Max+plus II综合结果： D触发器
FPGA Express综合结果: Latch

If语句

- 语法：

```
If <condition 1> then
    sequential statements;
Elsif  <condition 2>  then
    sequential statements;
        ……..
Else
    sequential statements;
End if;
```

- 对应于when-else语句

- 范例：

```
if sel="00" then
        q<=a;
elsif sel="01" then
        q<=b;
elsif sel="10" then
        q<=c;
else
        q<=d;
end if;
```

Case语句

- 语法：

```
case <expression> is
  when <value 1>=>sequential statements;
  when <value 2>=>sequential statements;
    …….
  when others=>sequential statements;
end case;
```

- 对应于select-when语句

- 范例：

```
case sel is
   when “00” =>q<=a;
   when “01” => q<=b;
   when “10” => q<=c;
   when others => q<=d;
end case;
```

Loop语句

- 无限循环：使用Exit退出循环
  - 语法：

```
<Loop_lable>: Loop
   sequential statements;
   exit Loop_lable [when <condition>];
End loop;
```

- While循环：根据条件结束循环

```
While <contition> Loop
   sequential statements;
End loop;
```

- For循环：

```
FOR <循环变量> IN <范围> Loop
   sequential statements;
End loop;
```

Loop语句范例

- 例1

```
Signal a: std_logic_vector(7 downto 0)

Process (a)
    variable temp: std_logic;
Begin
    temp:='1';
    for I in 0 to 7 loop
      temp:=temp and a(i);
    end loop;
    b<=temp;
End process;
```

- 例2

```
Signal a: std_logic_vector(7 downto 0)

Process (a)
    variable cnt: integer;
    variable temp: std_logic;
Begin
    temp:='1';
    cnt=0;
    while cnt<8 loop
      temp:=temp and a(cnt);
      cnt:=cnt+1;
    end loop;
    b<=temp;
End process;
```

Subprogram

- **用于描述一定的算法**
- **内部是串行执行—类似process**
- **可以在程序中任何地方被调用**
- **调用方法类似于Component例化**
- **综合结果是组合逻辑电路**
- **包括function和procedure**
  - Function: 有返回值
  - Procedure: 无返回值

- **子程序定义范例：**

```
Package example is
  procedure p(A: in integer, B:inout integer);
  function inc(A: in integer) return integer;
Package example;

Package body of example is
  procedure p(A: in integer, B:inout integer) is
  begin
       B:=A+B;
  end;

 function inc(A: in integer) return integer is
 begin
    return (A+1)
 end;
End example;
```

- **子程序调用范例：**

```
p(a,b);
x<=inc(a);
```

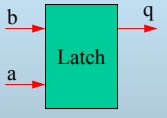
设计中应注意的问题

- Process的敏感列表应完全
- If, case-when语句应注意条件的完备性
- 尽量避免采用可能有歧义的语法
- 使用IEEE提供的数据包中的函数

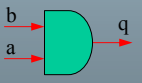
IF语句对比

```
a,b: in std_logic;
q: out std_logic;

Process(a,b)
begin
    If a='1' then
        q<=b;
    End if;
End process;
```




```
Process(a,b)
begin
    If a='1' then
        q<=b;
    Else
        q<='0';
    End if;
End process;
```



Library IEEE

- 定义了许多运算符的重载函数，方便实现各种组合逻辑和算术功能
- 定义了各种数据类型的相互转换函数，如conv_signed()将其他类型如integer, unsigned
- 在\maxplus2\vhdl93\ieee\目录下可以看这些库文件的源代码，例如arith.vhd是std_logic_arith包的声明文件，而arithb.vhd是该包的body。

例：减法器

- 输入：
  - A: 二进制无符号整数，8位
  - B: 二进制无符号整数，8位
- 输出
  - C: 二进制补码表是整数，9位
- 功能：
  - C=A-B
- 要求：输入输出都是std_logic_vector类型。

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_arith.all;

Entity sub_8 is
    port( A,B: in std_logic_vector(7 downto 0);
          C: out std_logic_vector(8 downto 0)
          );
End sub_8;

Architecture rtl of sub_8 is
    signal t1,t2: integer range 0 to 255;
    signal t3: integer range –256 to 255;
Begin
    t1<=conv_integer(unsigned(A));
    t2<=conv_integer(unsigned(B));
    t3<=t1-t2;
    C<=conv_std_logic_vector(t3,9);
End rtl;
```

本次实验内容

- **必做：**
  - 1、三——八译码器
  - 2、二——十进制转换器
  - 3、十六进制数显示
  - 4、二——十进制转换

- **选做：**
  - 1、四位无符号数加法器
  - 2、四位带符号数加/减法器
  - 4bits×16的ROM

- **实验有关资料下载：**

  Eelab.pku.edu.cn