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

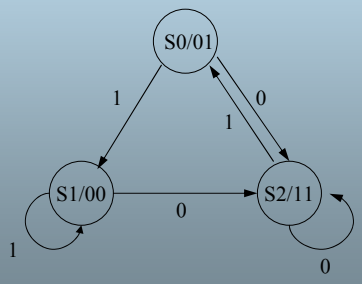
二〇〇六年

PLD设计的VHDL语言实现

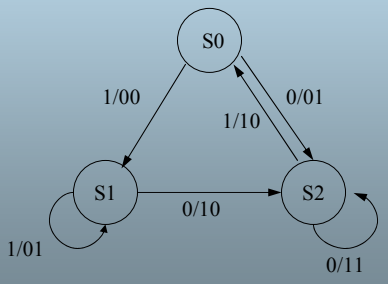
——状态机电路设计

状态机(State Machine)电路设计

- **状态机的两种类型：**
  - Moor状态机：输出仅与当前状态有关而与输入无关
  - Mealy状态机：输出与当前状态和输入都有关系



Moor状态机一例



Mealy状态机一例

状态机的定义

- **定义状态机数据类型**
  - 状态编码采用二进制编码

    语法：　TYPE State is (s0,s1,s2,s3,s4,s5);

    特点：　s0="000",s1="001",s2="010",.....
  - 状态编码采用其他码型

    语法：　TYPE State is (s0,s1,s2,s3,s4,s5);

    ATTRIBUTE ENUM_ENCODING : string;

    ATTRIBUTE ENUM_ENCODING OF

    State: TYPE is " 000 001 011 010 110 111 101 100";

    特点：　s0="000",s1="001",s2="011",.....
  - 其他常用码型
    - Gray码：相邻状态只变化1比特
    - One-Hot码型：每个状态只有1位有效，其他各位都无效，例如：

      s0="000001",s1="000010",s2="000100",.....
- **定义状态机信号**
  - 语法：

    Signal PresentState,NextState: State;
  - 隐性的定义状态机：

    Signal PresentState: std_logic_vector(2 downto 0);

状态机的实现

- 实现状态转移

```
process(clk)
begin
    if clk'event and clk='1' then
      PresentState<=NextState;
    end if;
end process;
```

- 判断移出状态及输出

```
process(input,PresentState)
begin
    case PresentState is
      when s0=>
            ...
            ...
      when s1=>
            ...
            ...
            ...
      when others=>
            ...
            ...
    end case;
end process;
```

状态机实例1

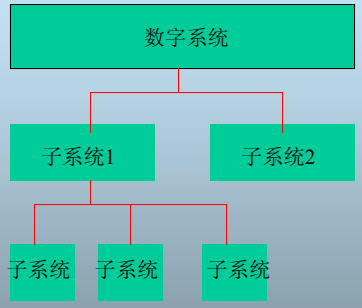
```
entity statemachine1 is
port(clk,din: in std_logic;
      dout: out std_logic_vector(1 downto 0));
end statemachine1;
architecture rtl of statemachine1 is
   type state is (s0,s1,s2);
   signal PresentState: state;
   signal NextState: state;
begin
   process(clk)
   begin
      if clk'event and clk='1' then
         PresentState<=NextState;
      end if;
   end process;

   process(din,PresentState)
   begin
      case PresentState is
         when s0=>
               if din='1' then
                  NextState<=s1;
                  dout<="00";
               else  NextState<=s2;
                     dout<="01";
               end if;
         when s1=>
               if din='1' then
                     NextState<=s1;
                     dout<="01";
               else  NextState<=s2;
                     dout<="10";
               end if;
         when s2=>
               if din='1' then
                     NextState<=s0;
                     dout<="10";
               else  NextState<=s2;
                     dout<="11";
               end if;
         when others=>
                     NextState<=s0;
                     dout<="11";
      end case;
end process;
```

状态机实例2：电子表

```
architecture rtl of watch is
   type state is (work,adj_sec,adj_min,adj_hour);
   signal PresentState: state;
   signal NextState: state;
begin
   process(clk)
   begin
      if clk'event and clk='1' then
         PresentState<=NextState;
      end if;
   end process;

   process(input,PresentState)
   begin
      case PresentState is
         when work=>
            if <chg_state> then
               NextState<=adj_sec;
            else
               NextState<=work;
            end if;
         when adj_sec=>
            if <chg_state> then
               NextState<=adj_min;
            else
               NextState<=adj_sec;
            end if;
         when adj_min=>
            if <chg_state> then
               NextState<=adj_hour;
            else
               NextState<=adj_min;
            end if;
         when adj_hour=>
            if <chg_state> then
               NextState<=work;
            else
               NextState<=adj_hour;
            end if;
         when others=>
               NextState<=work;
      end case;
   end process;
end rtl;
```

层次化（Hierarchy）设计方法

- 对于大型系统来说，如果将其所有功能放在一个设计文件中实现，则带来许多问题：
  - 开发代码困难、程序可读性降低
  - 程序调试困难
  - 不适合团队分工合作
- 层次化设计方法是大系统开发时所采用的通用方法
  - 将功能复杂的大系统分成若干个子系统，子系统又可以分成更小的子系统等等，每个子系统功能相对独立。
  - 各个子系统的设计和修改可以相对独立地进行
  - 适合于多个人同时并行开发



VHDL语言层次化设计

- **Component是VHDL语言层次化设计中的基本单元**
  - Component是在别的高层次entity中作为模块被引用的低层次entity
  - VHDL语言通过Component说明和例化实现层次化设计
  - Component说明
    - Component在被使用前必须说明
    - Component说明用语定义其对外接口<==>Entity
    - Component的说明可以在package里或者是architecture的说明中
  - Component例化
    - 使用Component
      一个Component可以被多次例化
    - 例化在Architecture中进行
    - Component的例化是并行语句
      - 出现于architecture中
      - 不能由输入信号决定有选择的例化

Component说明和例化语句

- **Component说明：与entity类似**

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

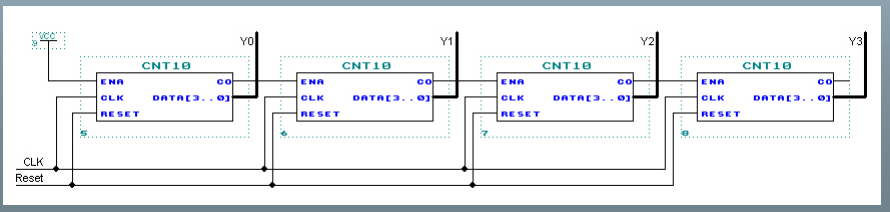
- **Component例化语句：**

```
<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>, …);

层次化设计范例：4位十进制加计数器

- **设计要求：**
  - 4位十进制加计数器
  - 具有同步复位功能
- **设计分析：**
  - 4位十进制加计数器可以分解为4个十进制加计数器
  - 顶层模块：4位十进制加计数器；子模块：1位十进制加计数器
  - 子模块接口：
    - 输入：时钟、复位信号、使能信号
    - 输出：进位信号、4比特数据（表示1个10进制数）



层次化设计范例：4位十进制加计数器

- **顶层设计文件：cnt10x4.vhd**

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

entity cnt10x4 is
port(
    clk: in std_logic;
    reset: in std_logic;
    Y0: out std_logic_vector(3 downto 0);
    Y1: out std_logic_vector(3 downto 0);
    Y2: out std_logic_vector(3 downto 0);
    Y3: out std_logic_vector(3 downto 0)
    );
end cnt10x4;
```

```
architecture rtl of cnt10x4 is
    Component cnt10 is
    port(
            clk: in std_logic;
            ena: in std_logic;
            reset: in std_logic;
            co: out std_logic;
            data: out std_logic_vector(3
                    downto 0));
    end component;
    signal t: std_logic_vector(4 downto 0);
begin
    u0: cnt10 port map(clk,t(0),reset,t(1),Y0);
    u1: cnt10 port map(clk,t(1),reset,t(2),Y1);
    u2: cnt10 port map(clk,t(2),reset,t(3),Y2);
    u3: cnt10 port map(clk,t(3),reset,t(4),Y3);
    t(0)<='1';
end rtl;
```

层次化设计范例：4位十进制加计数器

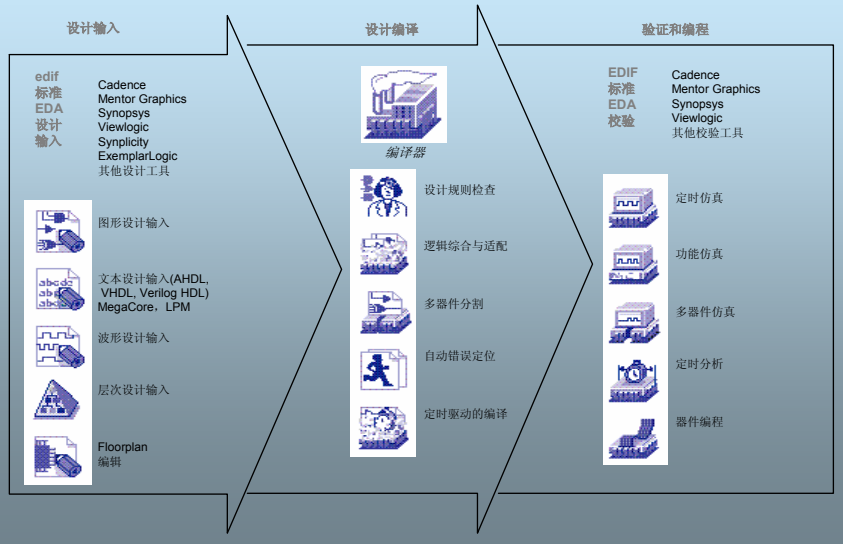
- **子模块文件：cnt10.vhd**

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

entity cnt10 is
port(
      clk: in std_logic;
      ena: in std_logic;
      reset: in std_logic;
      co: out std_logic;
      data: out std_logic_vector(3 downto 0));
end cnt10;

architecture rtl of cnt10 is
   signal temp: std_logic_vector(3 downto 0);
begin
   process(clk)
   begin
      if clk'event and clk='1' then
         if reset='1' then
            temp<="0000";
         elsif ena='1' then
            if temp="1001" then
               temp<="0000";
            else
               temp<=temp+1;
            end if;
         end if;
      end if;
   end process;
   data<=temp;
   co<='1' when (temp="1001" and ena='1')
     else '0';
end rtl;
```

EDA软件与MAX+PLUS II接口

- **MAX+PLUS II软件可以与其他大多数符合工业标准的EDA软件接口**
- **工作流程：**
  - 用MAX+PLUS II编辑器或标准EDA工具产生设计
  - 用MAX+PLUS II 编译设计
  - 用MAX+PLUS II 进行定时分析
  - 用MAX+PLUS II仿真或者用标准仿真器仿真
  - 下载
- **为什么要用其他EDA工具？**
  - 标准、通用
  - 在某些方面如对VHDL语言的支持以及综合效率等方面，某些EDA软件可能做得比MAX+PLUS II更好

EDA软件与MAX+PLUS II接口

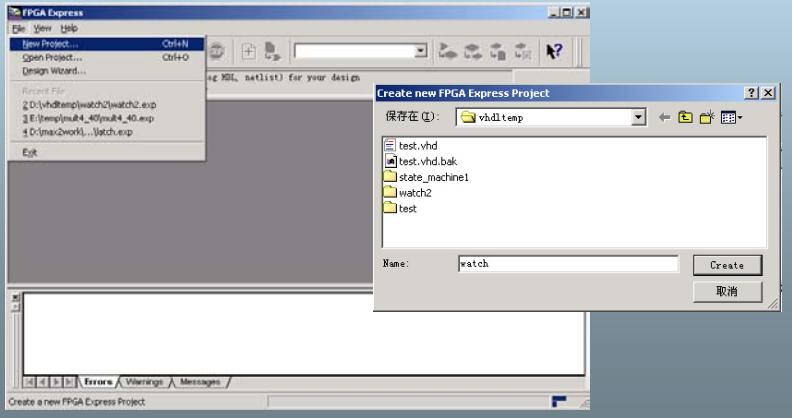


EDA软件与MAX+PLUS II接口举例

- 用Synopsys公司的FPGAExpress软件对VHDL语言设计进行综合、优化，输出.edf文件和.lmf文件
- 用MAX+PLUS II软件对FpgaExpress输出的.edf文件进行编译

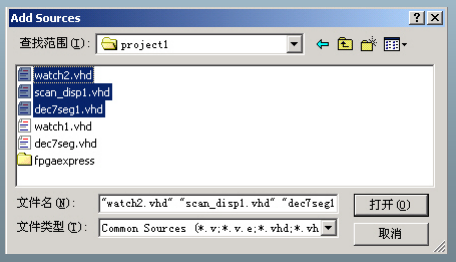
在FPGAExpress中建立新设计

- 建立新Project: File->New Project
- 选择Project存放目录及Project名：FPGAExpress将在该目录下建立一个与Project同名的目录以存放该project的有关文件
- 点击Creat按钮



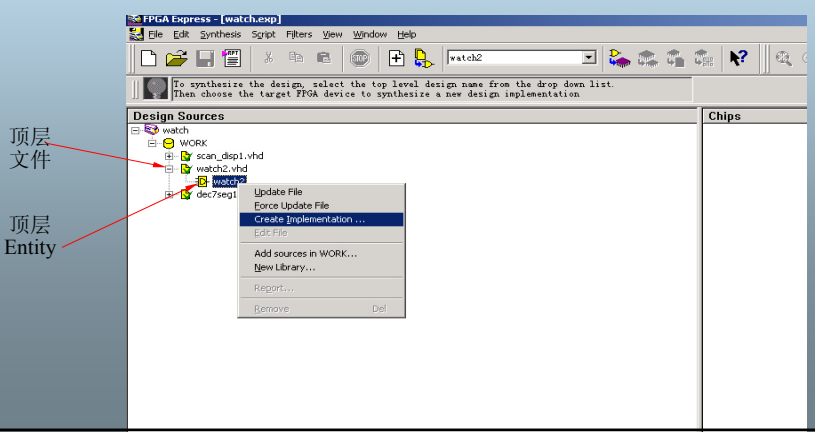
加入设计文件

- 选择该Project包含的设计文件：这些文件可以在别的目录下
- 点击“打开”按钮
- 可以反复通过选择Synthesis->Add Source Files...来加入别的设计文件
- 加入后FPGAExpress会自动对各个文件进行语法检查



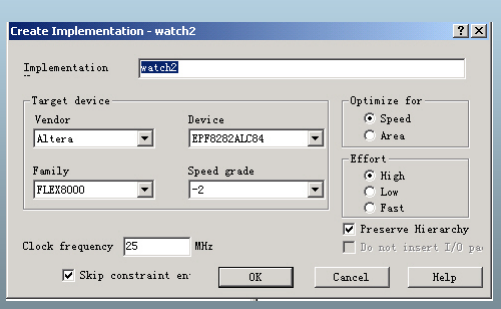
选择顶层Entity进行综合

- 选中顶层Entity，右键弹出菜单，选择Create Implementation...

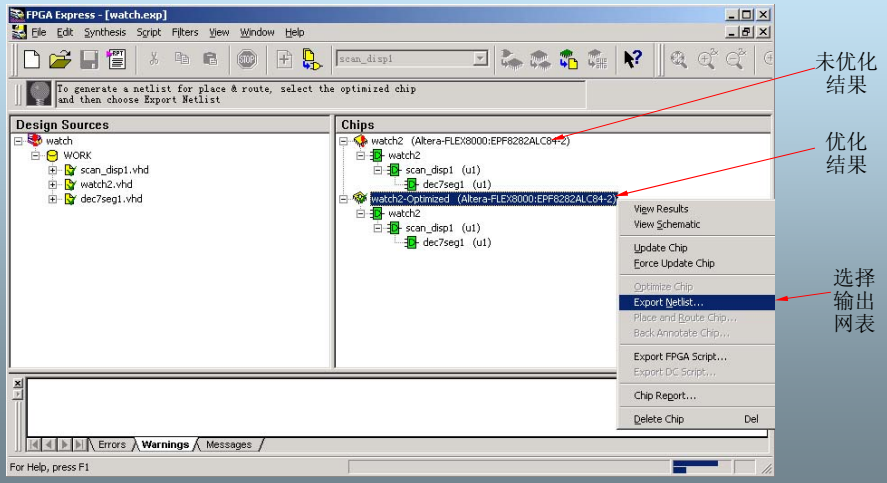


选择器件型号及综合选项

- 在弹出的菜单中选择器件型号以及优化选项
- 点击ok则FPGAExpress开始综合和优化设计

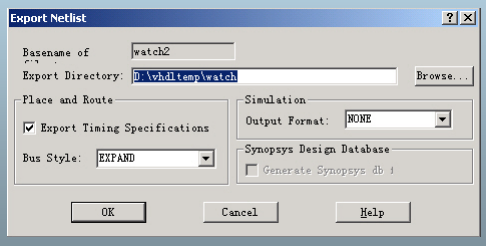


选择输出网表文件



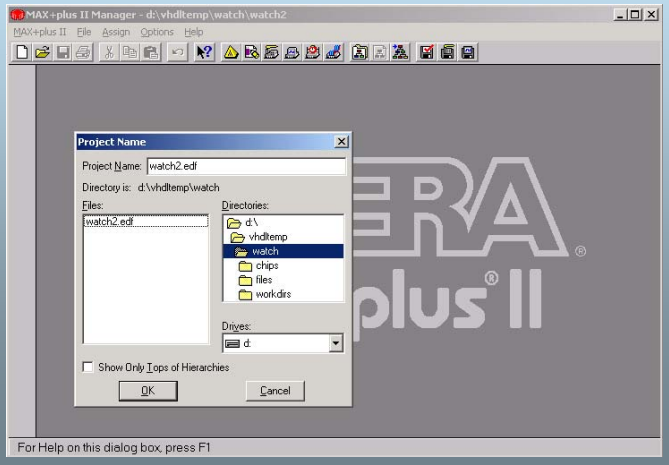
选择输出目录

- 选择输出目录：缺省为设计所在目录
- 点击ok按钮
- 将输出目录中产生的与顶层entity同名的后缀分别为.edf和.lmf的两个文件拷贝到MAX+PLUS II的Project目录中准备编译



用MAX+PLUS II编译设计

- 打开用MAX+PLUS II
- 建立一个新Project并将FPGAExpress产生的.edf文件作为设计文件
- 选择ok

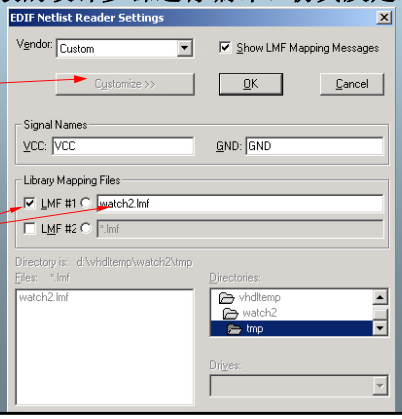


准备编译

- 打开Compiler
- 选择Interface->Edif Netlist Reader Settings...
- 选择customize...
- 选择library Mapping File
- 选择ok后，按一般的设计步骤进行编译、仿真及定时估计

点击
Customize

选择lmf文件为
FPGAExpress生
成的.lmf文件



使用FPGAExpress优化结果

- **电子表程序用年两种方式综合结果对比**
  - VHDL源程序直接用MAX＋PLUS II综合编译
  - VHDL源程序经FPGAExpress优化后再用MAX＋PLUS II编译
  - 采用EPF8282ALC64-2芯片编译、速度优化
- **直接用MAX+PLUS II编译结果：**
  - 占用LCs: 206 个 (99%)
  - 最高时钟频率: 29.76MHz
- **用FPGAExpress综合：**
  - 占用LCs: 176 个 (84%)
  - 最高时钟频率: 29.23MHz

本次实验内容

- **必做：**
  - 设计一个状态机
  - 脉宽测量电路 （采用状态机实现）
- **选做：**
  - 用层次化设计完成试验三选做中的倒计时装置
  - 16进制——10进制转换电路
  - 串行除法器

Project 1 内容介绍：电子表

- **功能要求：**
  - 具有时、分、秒计时功能
  - 具有设定时间功能
- **界面要求：**
  - 时、分、秒同时用6个7段显示器显示，并用小数点隔开
  - 系统有四个状态：正常计时、调时、调分、调秒
  - 状态切换按键每按下一次系统自动切换到下一状态，循环往复。
  - 在调时、调分、调秒状态时，相应的时、分、秒应闪烁显示
- **提示：**
  - 用两个脉冲发生器按键设定时间
  - 用一个脉冲发生器切换系统状态（正常计时状态/调秒状态/调分状态/调时状态）；用一个脉冲发生器设定时间