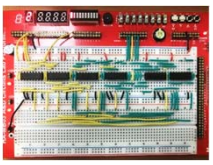
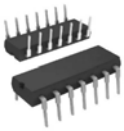
EDA讲座1

- 软件平台——QuartusII
- 硬件平台——实验板

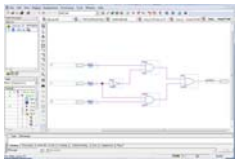
学习一种设计和实现电路的方法

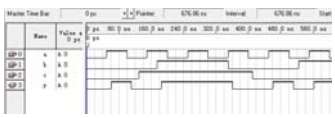
- 中小规模电路

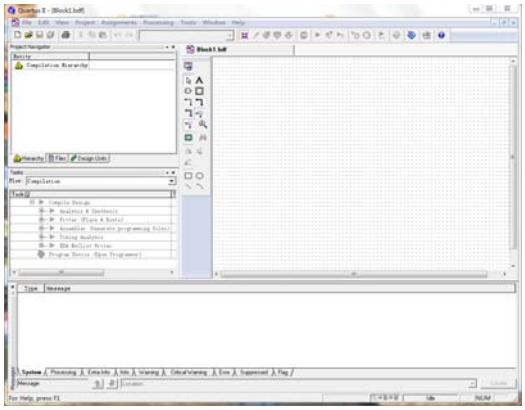
- 大规模电路

EDA工具

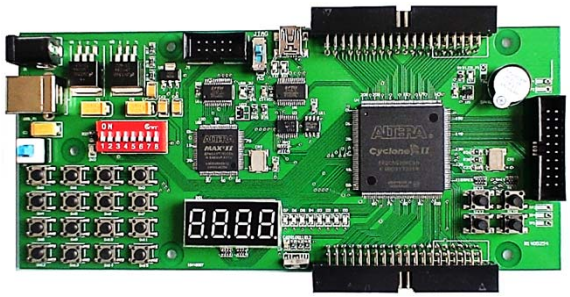




EDA工具

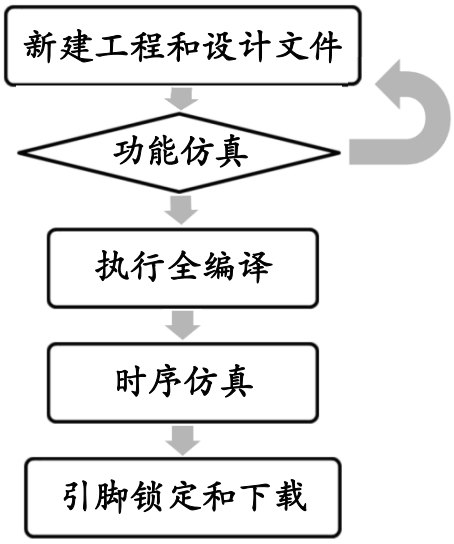


以QUARTUS II 13.0为例



Quartus II的设计流程

设计流程



原理图、硬件描述语言

建立波形文件
检查逻辑功能是否正确

综合、布局布线

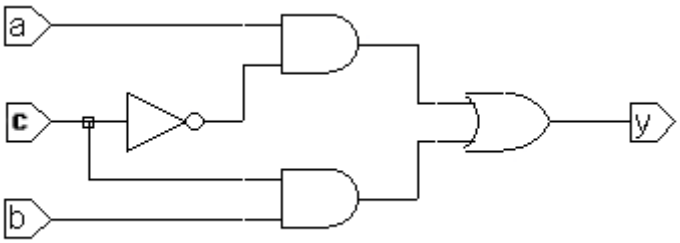
验证电路的时序

下载验证

今天的任务

- 了解QuartusII的使用流程
  - 设计输入——原理图、硬件描述语言
  - 功能仿真——加载波形
  - 时序仿真
  - 全编译——与器件对应
- 熟悉实验板，掌握板上外设的工作原理
  - 引脚锁定
  - 下载

$$y = c' \cdot a + c \cdot b$$



a	b	c	y
X	X	0	a
X	X	1	b

新建工程和设计文件

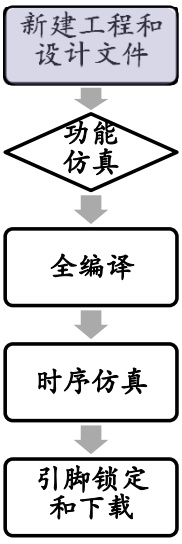
- 建立工作文件夹和设计工程的文件夹
- 新建工程向导File→New Project Wizard…
- 新建原理图文件

File→New→Design File→

Block Diagram/Schematic File

- 绘制电路图
- 对设计文件执行分析与综合

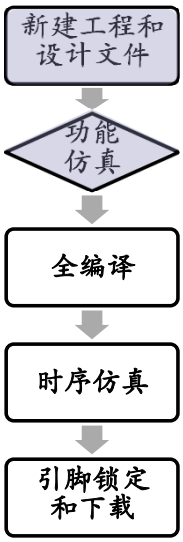
Processing→Start→Start Analysis and Sythesis



功能仿真

进行逻辑功能的测试。
通过观察输入输出波形的关系来检查是否满足设计要求。

- 建立波形文件
  File→New→Verification/Debugging File
  →University Program VWF
- 添加待观测信号节点，输入激励
- 在Simulation→Options中指定仿真工具：
  选择 QuartusII Simulator
- 执行功能仿真
  Simulation→Run Functional Simulation
- 观察仿真结果



电路设计的输入方式

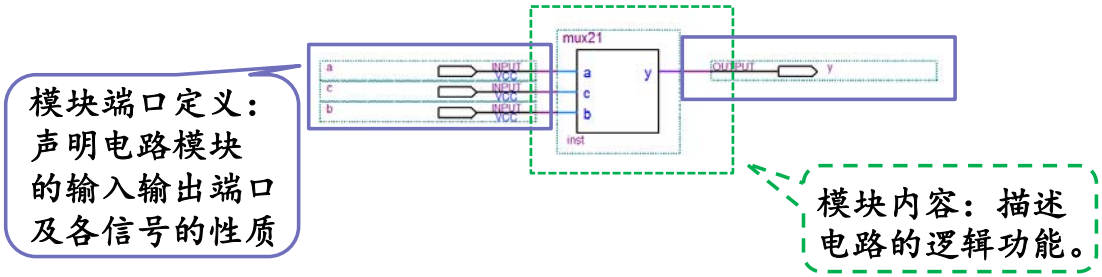
	原理图 VS.	硬件描述语言
输入方式	逻辑门 绘制好的电路图生成图形符号	源代码
对设计者的要求	熟悉电路的结构和连接关系	电路的功能 输入输出之间的转换
特点	形象直观	可移植性好 能形式化地抽象表示电路的行为和结构
两种方式发挥各自优势，可以混合使用		

Verilog HDL vs. VHDL

- Verilog HDL结构比较灵活，是一种非常容易掌握的硬件描述语言（类C语言）。
- VHDL语言的高层抽象能力要稍优一些。语言规范十分严谨，甚至于繁琐，但是可读性却十分好。
- 大学、研究机构更多使用VHDL，而工业界更多使用Verilog HDL。

模块的基本结构

把一个电路以**模块**的形式加以定义

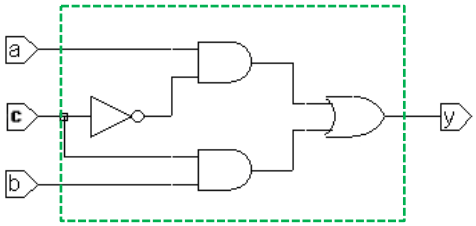


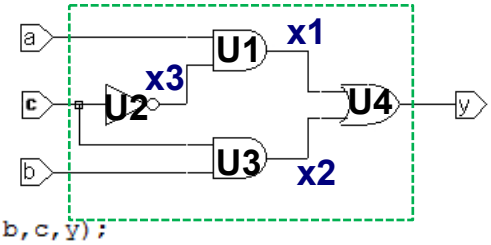
- 模块对应硬件电路的逻辑实体
- 一个电路模块可以是一个逻辑门、一片中规模芯片……

模块的三种描述方式

1. 结构描述：通过**定义端口**和使用**元件例化语句**描述元件或底层模块之间的互连关系。

——用语言实现层次化、模块化，与原理图输入方式类似






```
module mux21 (a,b,c,y);
  input a,b,c;
  output y;

  wire x1,x2,x3;

  and U1 (x1,a,x3);
  not U2 (x3,c);
  and U3 (x2,c,b);
  or  U4 (y,x1,x2);

endmodule
```

☞ 寄存器型reg和网线型wire
分别对应不同的电路结构和电路连线

1.变量数据类型说明，用wire表示内部连线
2.描述电路的结构

☞ 使用元件例化语句将元件相连

模块的三种描述方式

1. 结构描述：通过**定义端口**和使用**元件例化语句**描述元件或底层模块之间的互连关系。

——用语言实现层次化、模块化，与原理图输入方式类似

2. 数据流描述：使用**连续赋值语句**来描述输入输出的逻辑关系。既含有逻辑单元的结构信息，又隐含某种行为。

$$y = ac' + bc$$

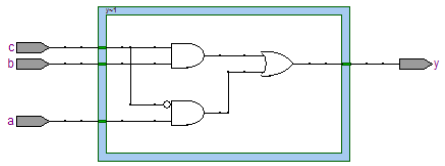
——描述方式与布尔代数类似

$$y = ac' + bc$$

```
module mux21 (a,b,c,y);
  input a,b,c;
  output y;

  assign y=(a&~c)|(b&c);

  endmodule
```



门级

assign引导连续赋值语句。

☞ assign使用变量类型规定为wire

按位逻辑运算符	
~	非
|	或
&	与
^	异或
~^ , ^~	同或

逻辑运算符	
!	非
||	或
&&	与

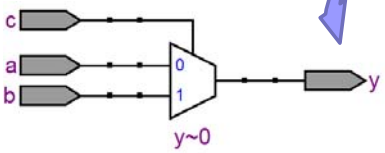
$$y = ac' + bc$$

```
module mux21 (a,b,c,y);
  input a,b,c;
  output y;

  assign y=(c?b:a);

endmodule
```

Tools →Netlist Viewers →RTL Viewer



RTL级

☞ 条件运算符： c=1时y=b

模块的三种描述方式

1. 结构描述：通过定义端口和使用元件例化语句描述元件或底层模块之间的互连关系。

——用语言实现层次化、模块化，与原理图输入方式类似

2. 数据流描述：使用连续赋值语句来描述输入输出的逻辑关系。既含有逻辑单元的结构信息，又隐含某种行为。

——描述方式与布尔代数类似

3. 行为描述：描述的是电路的功能或行为，表达输入与输出之间转换的行为。而并非真实的硬件结构、连接方式或逻辑关系。

——抽象程度高、由EDA工具将行为描述语言转换为“真实电路”

```
module mux21 (a,b,c,y);
  input a,b,c;
  output y;

  reg y;

  always@(a or b or c)
    begin
      case (c)
      1'b0:y<=a;
      1'b1:y<=b;
      //default:y<=a;
      endcase
    end
endmodule
```

1.always@引导过程语句结构。
括号中为敏感表，敏感信号之间用or或"，"连接。

☞always引导的顺序语句中，变量必须是reg型。

2.多路分支语句Case-endcase，类似真值表表达方式的描述。

c	y
0	a
1	b

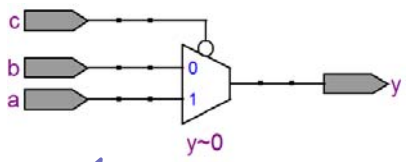
☞ 若不能全部覆盖表达式的取值，必须加上default语句。

if-else语句及其综合得到的电路结构

```
module mux21 (a,b,c,y);
  input a,b,c;
  output y;

  reg y;

  always @ (a,b,c)
    begin
      if (c==0)   y<=a;
      else        y<=b;
    end
endmodule
```



RTL级

等式运算符

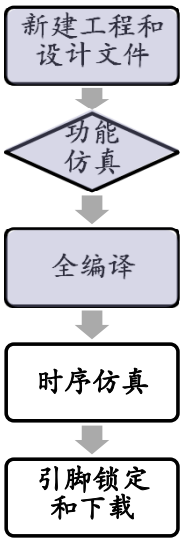
==	等于
!=	不等于
===	全等
!==	不全等

☞ 写全If-else所有可能的分支，否则必须加上default语句。

全编译

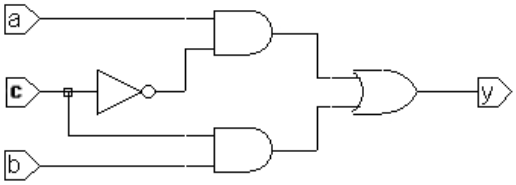
Processing → Start Compilation

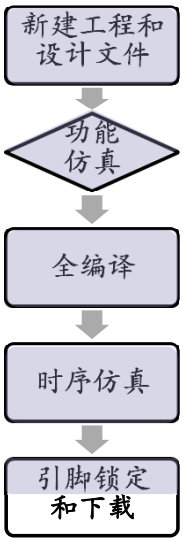
- 将设计项目适配到指定的目标器件中。
- 产生多种用途的输出文件。
  功能和时序信息文件、器件编程的目标文件等。
- 编译成功后，可以读取硬件耗用统计报告、
  布局布线报告及时序特性报告等信息 。



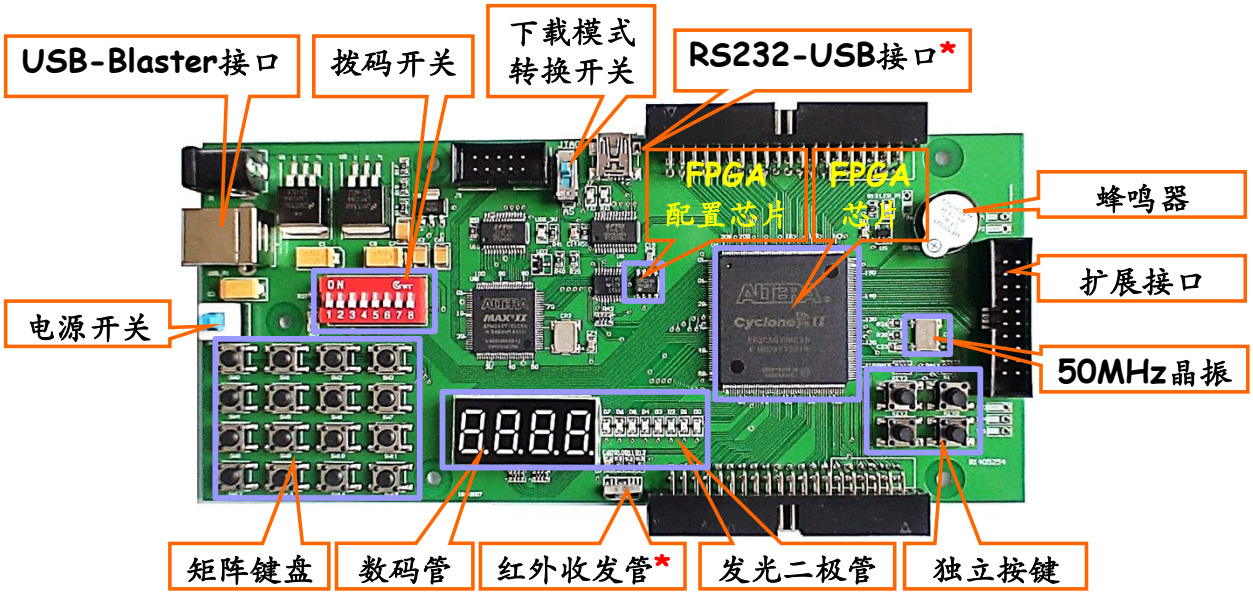
引脚锁定

- FPGA芯片引脚与板上外设已固定连接
  （拨码开关、发光二极管等）
- 将原理图中的输入输出引脚与之相对应起来

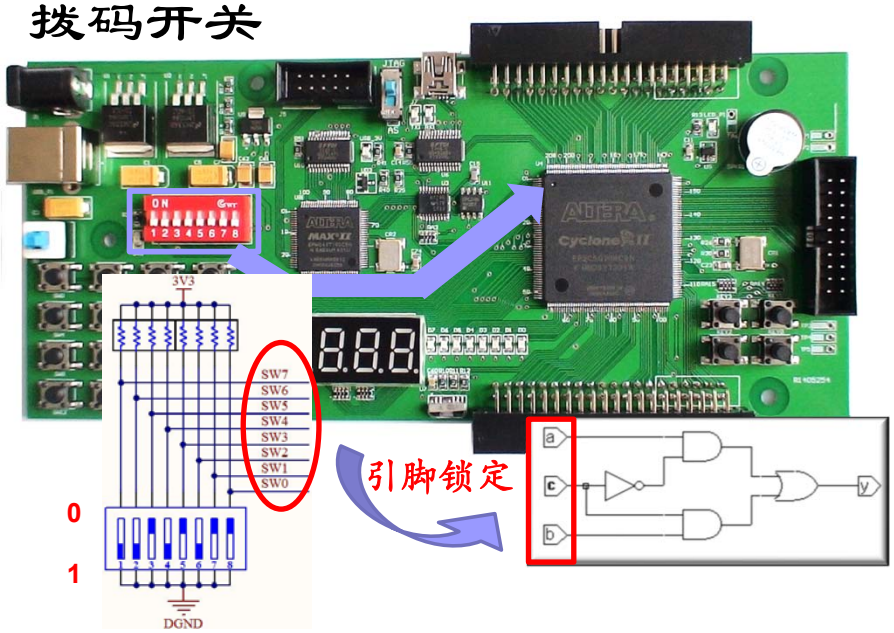




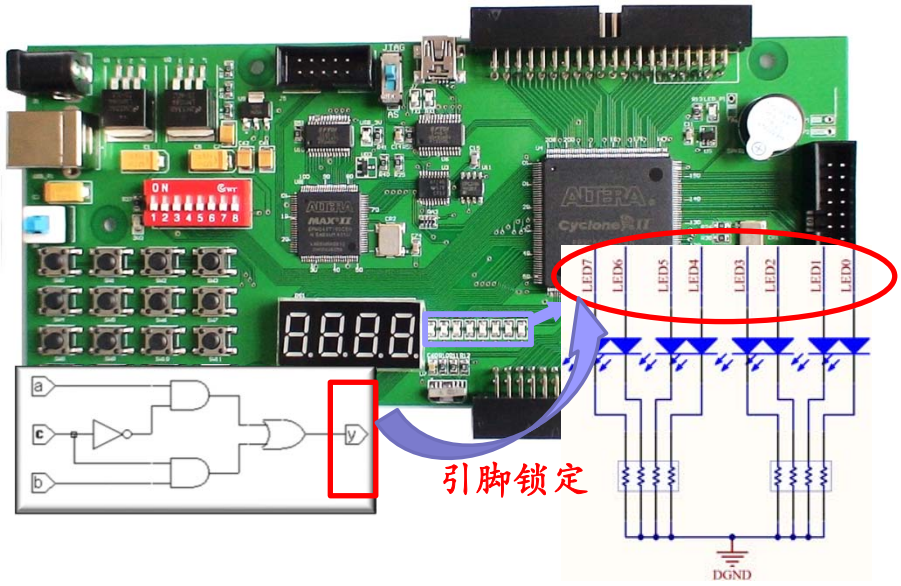
FPGA实验板


USB-Blaster接口
拨码开关
下载模式
转换开关
RS232-USB接口*
FPGA
配置芯片
FPGA
芯片
蜂鸣器
扩展接口
电源开关
50MHz晶振
矩阵键盘
数码管
红外收发管*
发光二极管
独立按键

拨码开关



发光二极管（LED）



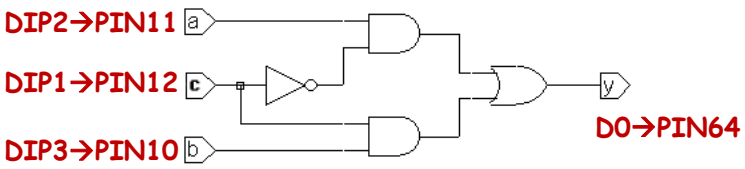
FPGA引脚表

器件名称	编号	引脚号	器件名称	编号	引脚号
拨码开关	DIP1	PIN_12	LED	D7	PIN_56
	DIP2	PIN_11		D6	PIN_57
	DIP3	PIN_10		D5	PIN_58
	DIP4	PIN_8		D4	PIN_59
	DIP5	PIN_6		D3	PIN_60
	DIP6	PIN_5		D2	PIN_61
	DIP7	PIN_4		D1	PIN_63
	DIP8	PIN_3		D0	PIN_64

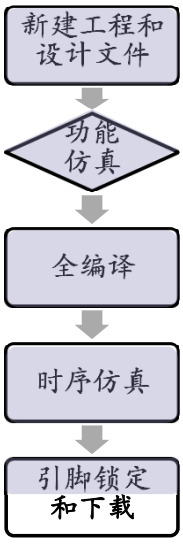
27

引脚锁定（续）

Assignments→pin planner

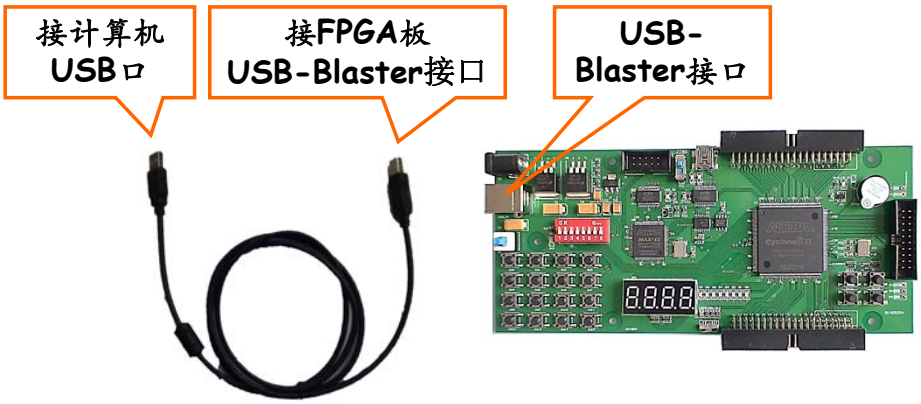


- 再做一次全编译以便将引脚锁定信息编译进下载文件中。



28

下载

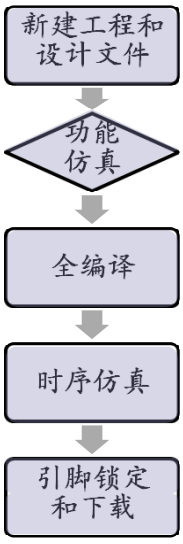


29

下载（续）

- 连接下载线
- 打开电源开关
- Tools →Programmer
  Hardware Setup：USB-Blaster
  Mode：JTAG
- Start下载SOF文件
- 板上验证设计

a	b	c	y
X	X	0	a
X	X	1	b



30

- **提供的资源：**
  - ☞提供教学视频（网络学堂--课程文件--EDA视频）
  - ☞提供文档资料（网络学堂--课程文件—实验文档）
- **今天的任务：**
  - ☞完成样例
  - ☞补发器材（实验盒、FPGA板）
- **下周预告：**

  EDA大作业一

31