**实验一 数字集成电路上课讲解内容**

[实验目的： 2](#_Toc15812)

[实验重点： 2](#_Toc2875)

[实验难点： 2](#_Toc19631)

[一、实现数字系统的三种方法 2](#_Toc16691)

[二、 FPGA设计步骤 4](#_Toc21946)

[三、Verilog语言使用举例 4](#_Toc11231)

[1. 组合逻辑电路举例 4](#_Toc28784)

[2、面向设计的verilog语法 5](#_Toc1840)

[2.1 模块定义 5](#_Toc10571)

[2.2信号方向 6](#_Toc5333)

[2.3常用的两个基本的信号数据类型 reg型、wire型 6](#_Toc3543)

[2.4常量 6](#_Toc7265)

[2.5运算符 7](#_Toc16068)

[2.6条件语言 8](#_Toc32130)

[2.7连续赋值语句 assign 8](#_Toc24517)

[2.8过程 8](#_Toc27892)

[3、时序逻辑电路设计 9](#_Toc15151)

[3.1最简单的D触发器 9](#_Toc31260)

[3.2带低电平有效异步复位端的触发器 9](#_Toc24490)

[3.3带同步复位端的D触发器 9](#_Toc10864)

[4、面向测试的Verilog语法 10](#_Toc28801)

[4.1 测试示意图 10](#_Toc8463)

[4.2完整的测试模块 10](#_Toc25936)

[4.3测试举例 11](#_Toc15421)

[4.4元件例化 12](#_Toc7189)

[4.5产生激励向量 12](#_Toc8096)

[4.6 显示输出结果 12](#_Toc30971)

[4.7特殊符号# 13](#_Toc6503)

[4.8 “ ` ”表示编译引导语句 13](#_Toc26969)

[参考资料 13](#_Toc464)

# 实验目的：

1.了解数字集成电路设计方法。

2.熟悉并运用Verilog语言进行电路设计。

# 实验重点：

1.学会组合逻辑电路的设计方法：

第一步:确定输入信号和输出信号；

第二步:确定输入和输出的逻辑状态关系；

• 真值表

• 布尔表达式

• 流程图、数据流图…

第三步：优化逻辑状态表达式—减少器件；用Verilog代码正确描述。

2.回顾不同种类的D触发器

# 实验难点：

新接触verilog语言，可能在心理上有压力。认识到verilog只是一个工具，用来实现逻辑电路。

# 一、实现数字系统的三种方法

1.用ASSP搭建系统，通过已有芯片，在面包板上连线，实现草稿纸上规划好的电路。

Application-Specific-Standard\_Product.

1. Pld可编程逻辑器件programmable logic device

在安装软件的时候选择了

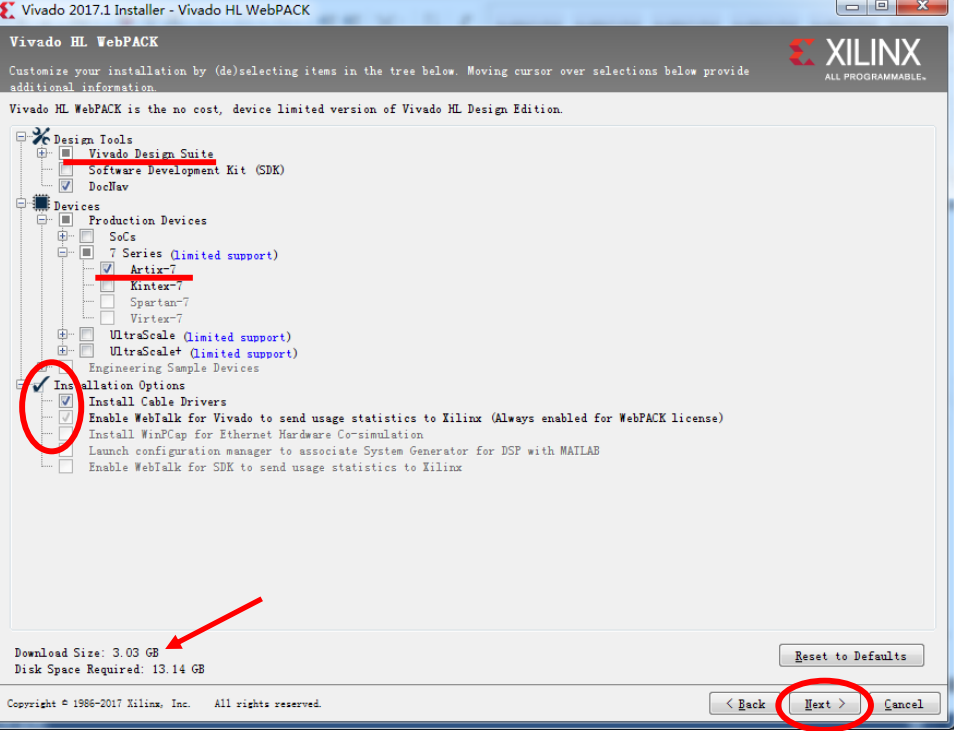


图1.1 vivado软件安装过程选择内容

移动通信的标准每隔两三年换一遍，在基站里面采用FPGA,用FPGA把内部电路更新一下。需要升级的时候去改一下。

编程选择

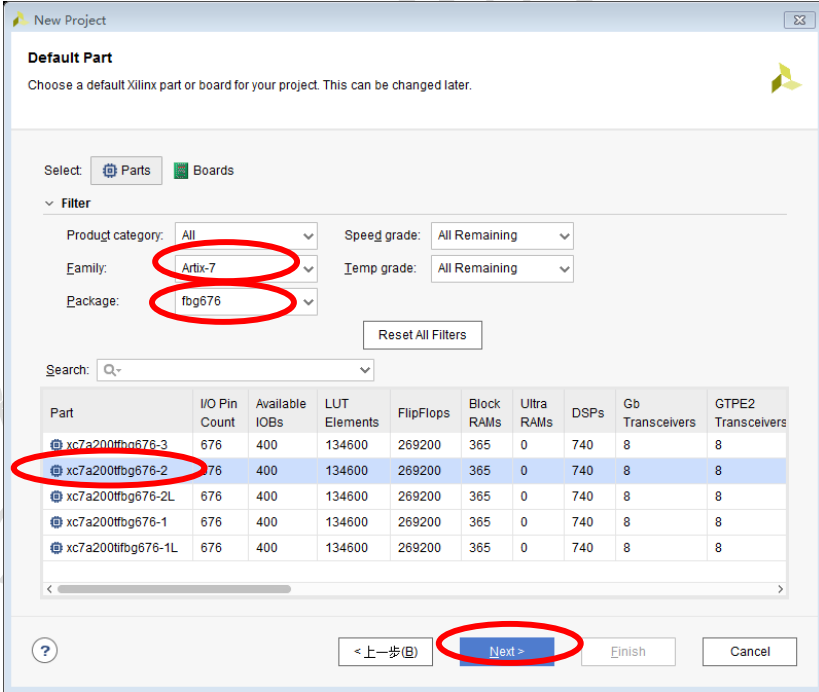


图1.2 编程选择芯片

1. Asic专用集成电路   
   Application Specific Integrated Circuit十万片以上可以考虑使用ASIC

# FPGA设计步骤

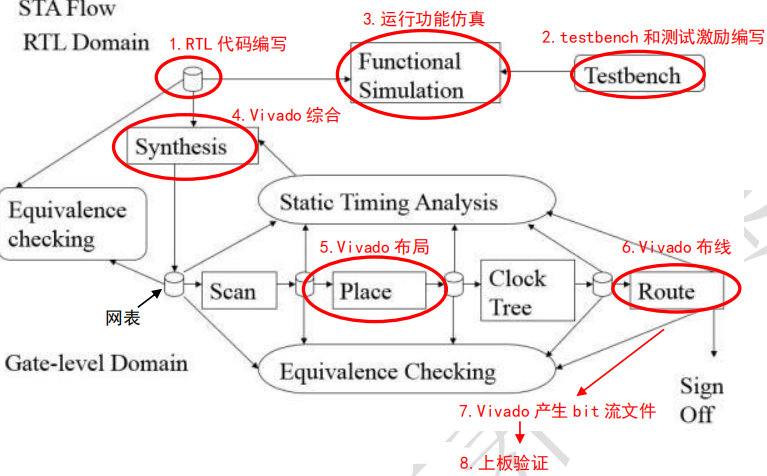
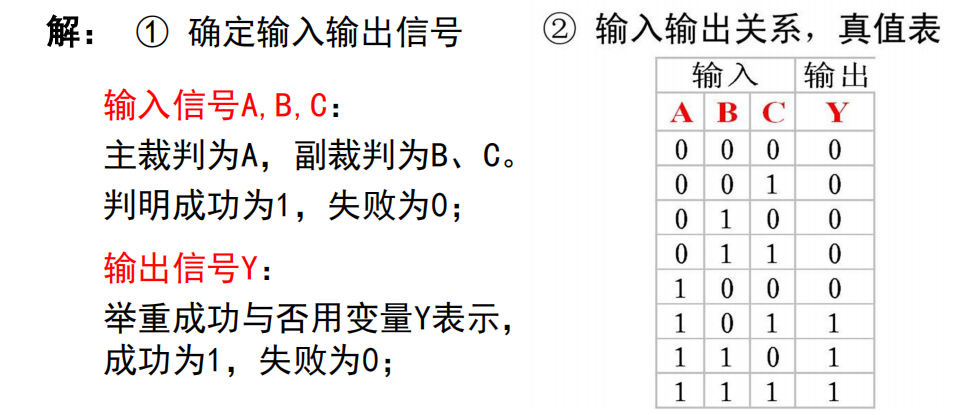


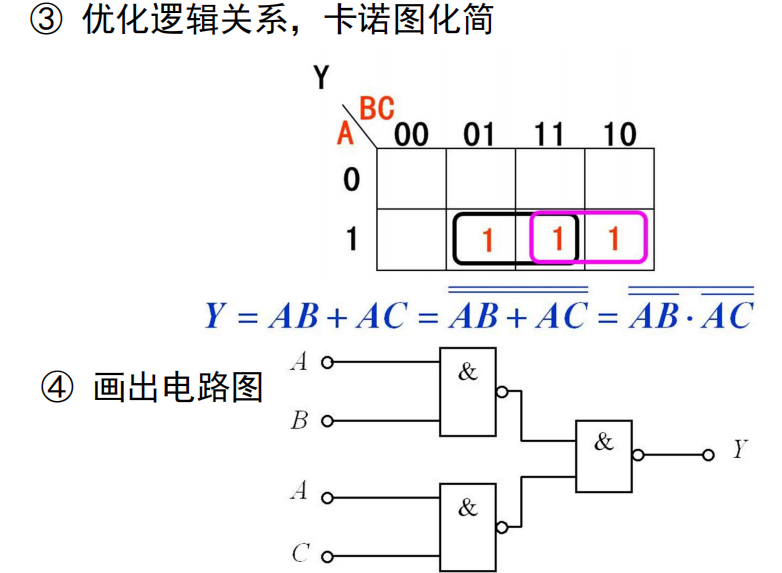
图2.1 FPGA设计步骤

# 三、Verilog语言使用举例

## 组合逻辑电路举例

完成组合逻辑电路裁判表决器设计。设计一个比赛裁判表决电路，设比赛有三个裁判，一个主裁判和两个副裁判。只有当两个或两个以上裁判判明成功，并且有一个为主裁判时，表明成功。使用verilog语言进行描述并且仿真，得到正确的波形图。





## 2、面向设计的verilog语法

### 2.1 模块定义

module模块名(端口1，端口2，端口3，端口4, ………);

endmodule

模块的内容包括I/O说明、内部信号声明、功能定义。

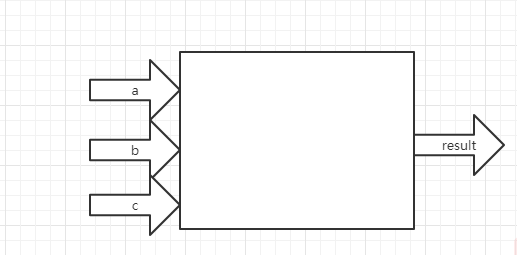


图3.1 表决器电路框图

module judge(

input wire a,

input wire b,

input wire c,

output reg result

);

always@(a or b or c)begin

result=(a&b)|(a&c);

end

endmodule

### 2.2信号方向

input,output,inout双向信号

2.3常用的两个基本的信号数据类型 reg型、wire型。

（1）wire

表示元件之间的物理连接，缺省值是Z

wire [n-1:0] 数据名1,数据名2,…数据名i;

wire a; //定义了一个一位的wire型数据

wire [7:0] b; //定义了一个八位的wire型数据

1. reg(具有存储功能的一根线，可以看成代码书写规范要求，而不是把它看成D触发器)

always/initial过程的输出中用reg型。reg 型数据的缺省初始值是不定值。

reg rega; //定义了一个一位的名为rega的reg型数据

reg [3:0] regb; //定义了一个四位的名为regb的reg型数据

1. 除了应该定义为reg的都定义为wire

### 2.4常量

verilog的四值逻辑

0, 1, X, Z

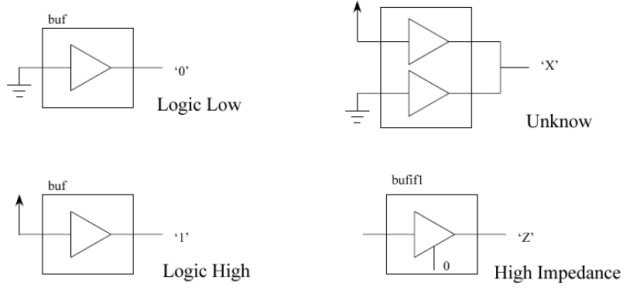


图3.2 verilog四值逻辑

<size>’<base><value>

• Size: 以bit为单位

• Base: b(二进制),o(八进制),d(十进制),h(16进制)

• Value:和进制相应的数值，x, z, ? (x,z不区分大小写)

例

• 16 //只给出一个数字，默认位宽为32位

• 8’d16 //位宽为8的十进制数16

• 8’h10 //位宽为8的十六进制数10h

16’b1010\_1011\_1111\_1010 //合法格式

8'b\_0011\_1010 //非法格式

• 32’bx //32位x

4'b10x0 //位宽为4的二进制数从低位数起第二位为不定值

4'b101z //位宽为4的二进制数从低位数起第一位为高阻值

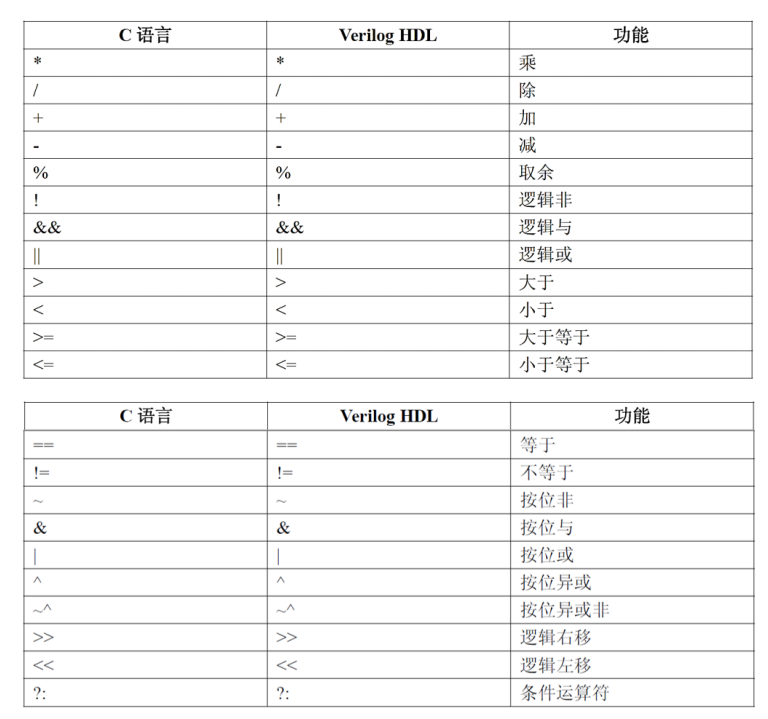
12'dz //位宽为12的十进制数其值为高阻值(第一种表达方式)

12'd? //位宽为12的十进制数其值为高阻值(第二种表达方式)

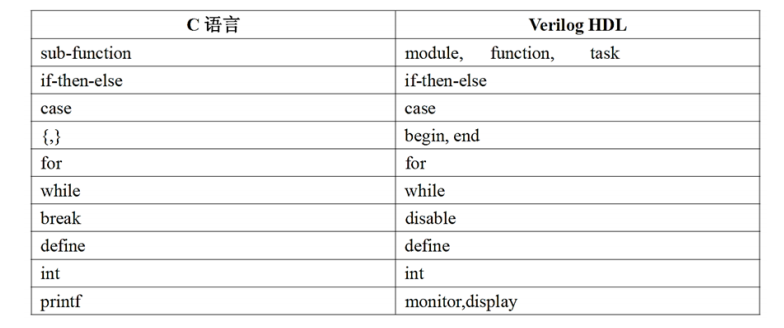
8'h4x //位宽为8的十六进制数其低四位值为不定值

### 2.5运算符

和C语言一样。（看看即可，用到哪个再学哪个）



### 2.6条件语言

看看即可，用到哪个再学哪个

2.7连续赋值语句 assign 用来实现简单的组合逻辑电路。

module block (

input wire a,

input wire b,

output wire c,

output wire d);

wire e=1’b1;//declare and assign

assign c= a | b;

assign d= e?a:b;//二选一

endmodule

要避免出现反馈电路assign a=b+a;

### 2.8过程

过程 (always)

• 阻塞赋值(=)

always @(a or b) //由输入信号中任意一个电平发生变化所引起，所有在赋值语句右边出现的信号我们都要放到敏感表中。if,case里面的条件信号也要放到敏感表中。

begin

语句块

(=, if, case)

end

• 非阻塞赋值(<=) //由单个跳变沿所引起

always @(posedge/negedge sig or…)

begin

语句块 (<=, if ,case)

end

(1)两个或更多always模块是同时执行的，模块内部阻塞赋值是顺序执行

(2)always模块描述组合逻辑电路时，用阻塞赋值语句，和c语言一样。

(3)always模块描述时序逻辑电路时，用非阻塞赋值语句，多条非阻塞语句并行执行。计算过程和赋值过程分开进行，先把右边计算过程的全部计算出来，再并行赋值。

(4)在同一个always/initial里面不要同时出现阻塞赋值和非阻塞赋值。

## 3、时序逻辑电路设计

时序电路设计都是D触发器基础上去设计的。

3.1最简单的D触发器。

always @(posedge clk) begin

q <= d ;

end

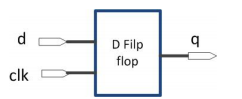


图3.3 简单D触发器示意图

### 3.2带低电平有效异步复位端的触发器

always @(posedge clk or negedge reset )

begin

if ( !reset )

begin

q <= 0;

end

else//对应了时钟边沿

begin

q <= d ;

end

end

### 3.3带同步复位端的D触发器

always @(posedge clk ) //reset没在敏感表中。在时钟上升沿和复位信号同时有效，才可以复位。

begin

if ( !reset )

dout <= 0;

else

dout <= din ;

end

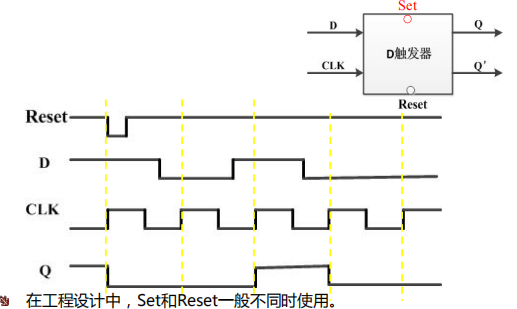


图3.4 波形图

## 4、面向测试的Verilog语法

### 4.1 测试示意图

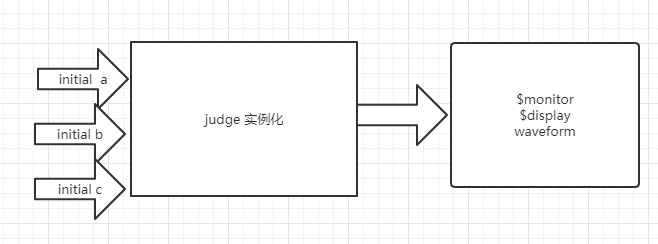


图3.5 测试示意图

### 4.2完整的测试模块

产生输入信号，观察输出信号

`timescale 1ns/1ps //显示的单位为1ns,精度为1ps,可以想象成时间步长是1ps,时间1ps,1ps这样走。

module tb; //没有输入信号，输出信号。

// 信号定义

// 元件例化

// 产生激励向量

// 显示输出结果

endmodule

### 4.3测试举例

module test();

reg a;

reg b;

reg c;

wire result;

judge judge1(.a(a),.b(b),.c(c),.result(result));

initial begin

a=0;

b=0;

c=0;

#10

a=0;

b=0;

c=1;

#10

a=0;

b=1;

c=0;

#10

a=0;

b=1;

c=1;

#10

a=1;

b=0;

c=0;

#10

a=1;

b=0;

c=1;

#10

a=1;

b=1;

c=0;

#10

a=1;

b=1;

c=1;

#10 $finish;

end

initial begin

$monitor($time,",a=%b,b=%b,c=%b,result=%b",a,b,c,result);

end

endmodule

### 4.4元件例化

相当于把一个电路例化。可以用在测试代码，也可以用在RTL级代码设计。

reg a;

reg b;

reg c;

wire result;

judge judge1(.a(a),.b(b),.c(c),.result(result));

### 4.5产生激励向量

initial:非周期信号

always:周期信号

initial begin

clk = 0;

forever

#10 clk = ~clk ;

end

initial begin

rst = 1;

#15 rst = 0;

#10 rst = 1;

#175 $finish;

end

initial begin

$monitor($time,,

“%b %b %b“, rst,clk,dout);

end

### 4.6 显示输出结果

(1)特殊符号$－系统任务和函数

$<标识符> 表示Verilog 的系统任务和函数

常用的系统任务和函数有下面几种：

– $time //当前的仿真时间

– $display, $monitor //显示和监视信号值的变化

– $stop //暂停仿真

– $finish //结束仿真

1. 观察波形

### 4.7特殊符号#

在实际的物理电路中，一般是不允许使用延时语句来描述电路的时间特性。但在仿真的时候我们需要描述电路的时间特性和时序特性。例如Clock信号周期，频率。

#特殊符号 “#” 表示延迟

（1）过程赋值语句里的延迟

initial begin

#10 rst=1; //10个时间单位之后，rst变为1

end

（2）门级实例引用的延迟

not #1 not1(nsel, sel);

1. 元件例化的参数传递

### 4.8 “ ` ”表示编译引导语句

用于指导仿真编译器在编译时采取一些特殊处理。

编译引导语句一直保持有效，直到被取消或重写。

`resetall 编译引导语句把所有设置的编译引导恢复到缺省状态。

常用的编译引导有：

• `define

• `include

• `timescale

• `resetall

`timescale 10ns/1ns

module MUX2\_1(out,a,b,sel);

… …

and #2.33 and1(a1, a, nsel);

延时 2.33×10＝23.3，精确到1ns,即23ns

# 参考资料

1. 《IC设计与方法》，学堂在线 ，张春。
2. 龙芯杯系统能力培养大赛资料
3. 《夏宇闻数字逻辑设计》