# 实验一 Vivado环境下组合电路的设计与仿真

## 一、 实验目的

1. 熟悉 的编译环境；
2. 了解在 Vivado 环境下运用 Verilog HDL 语言的编程开发流程，包括源程序的编写、编译、模拟仿真。

## 二、 实验内容

1. Vivado 环境下源程序的编写、编译
2. 模拟仿真

## 实验要求

1. 在Vivado 环境下完成对简单电路工作情况的仿真模拟；

2. 实现以下组合逻辑功能：编码／译码器，比较器，全加器。

## 四、 实验步骤

**（一）简单的led灯控制电路的设计**

1．介绍在 Vivado 环境下的编程开发流程

启动 Vivado。如图 1.1 所示：

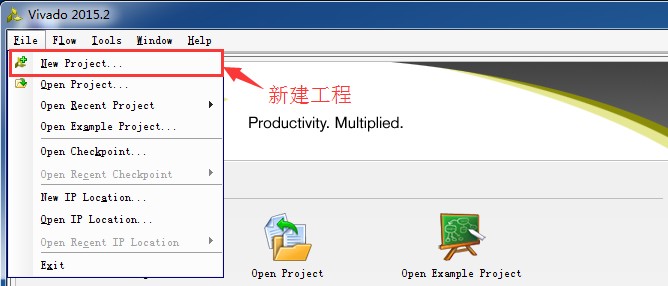


图 1.1 Vivado 窗口界面

1. 利用向导，建立一个新项目。
   * 在 File 菜单中选择 New Project 选项启动项目向导。
   * 填写所要新建的工程名。如这里的工程名： led， 工程所在位置： C:/Workspace/Vivado，然后点击 Next。如图 1.2 所示：

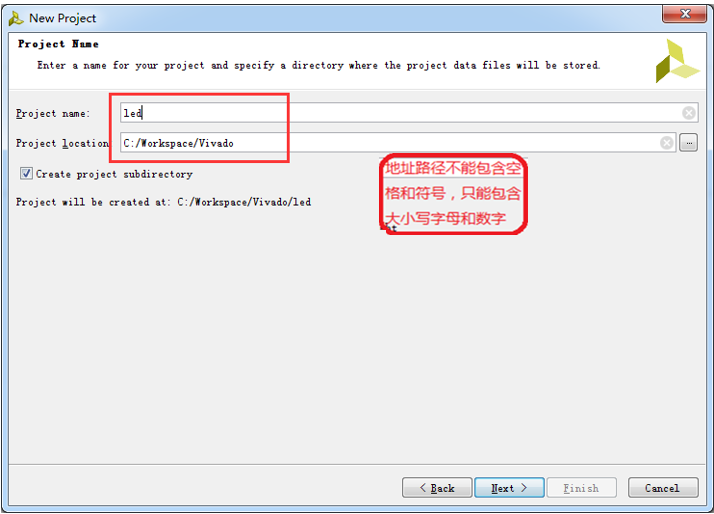


图 1.2 Vivado 项目名称、路径设定窗口

* + 在 File 选择项目类型，如图 1.3 所示：

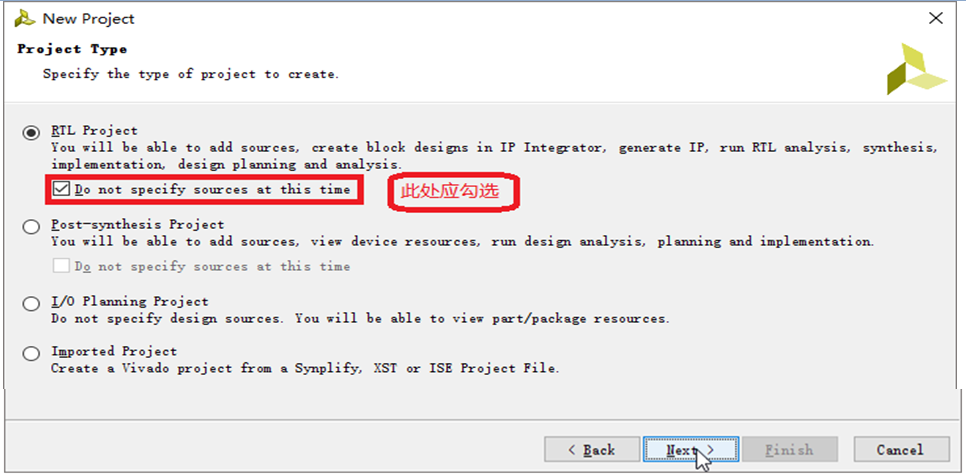


图 1.3 项目类型

* + 由于我们是新建工程，所以此处默认没有可以添加的源文件，并且设置编程语言和仿真语 言均设置为 Verilog。点击 Next，如图 1.4 所示：

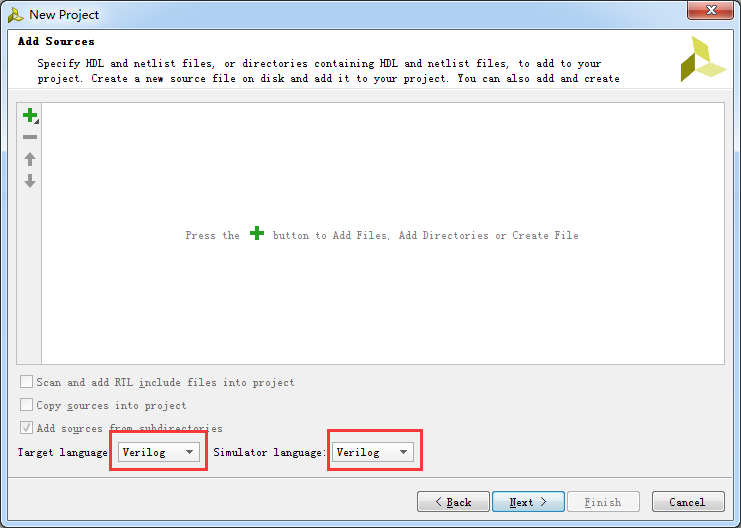


图 1.4、语言选择

* + 没有可以添加的 IP，所以不添加 IP,直接点击 Next
  + 也没有可以添加的约束文件，所以不添加，直接点击 Next
  + 器件的选择是和实验平台的硬件相关的，根据我们的 EGO1 实验开发板，它使用的是 xc7a35tcsg324-1的器件，找到相应的器件，如图 1.5 所示：

板卡选项分别为Family: Artix-7

Package: csg324

Speed grade: -1

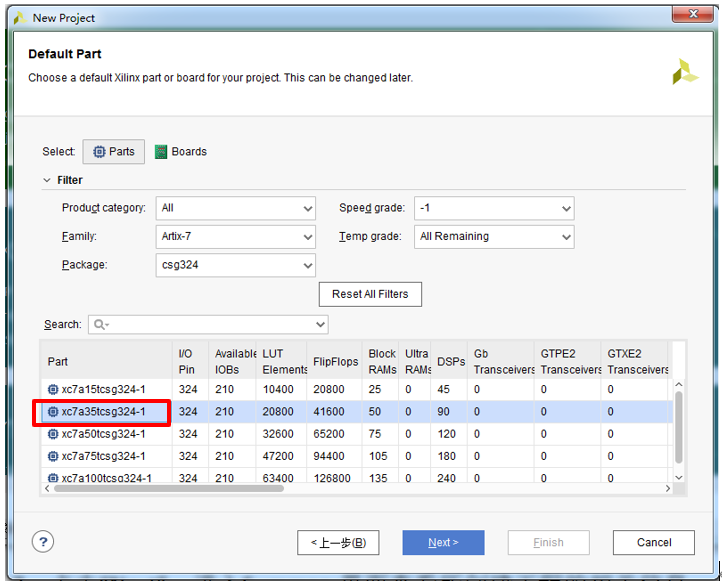


图 1.5 Vivado 中器件选择窗口

* + VIVADO 中包含完整的文本编辑程序 （Text Editor），在此用 Verilog HDL 来编写源程序。新建一个 Verilog HDL 文件，可以通过右击 Design sourse 选择 Add Sourse，如图 1.6 所示：

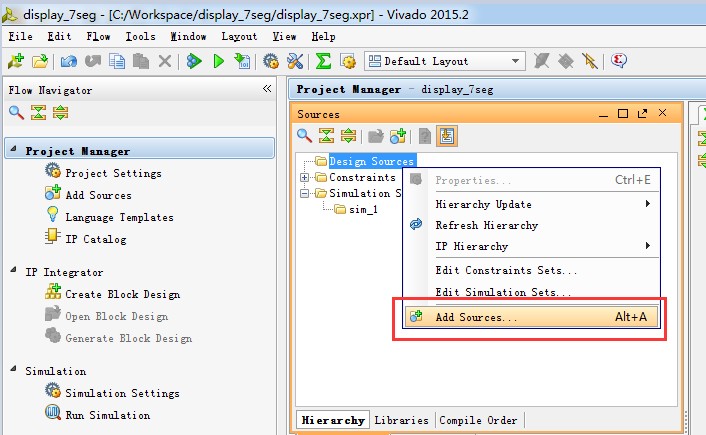
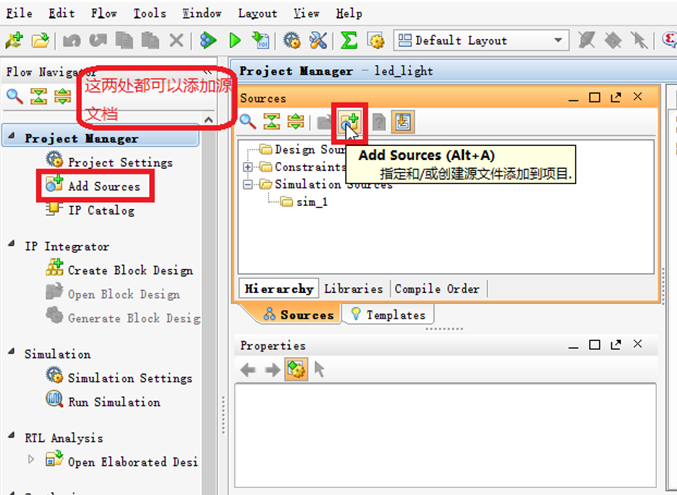
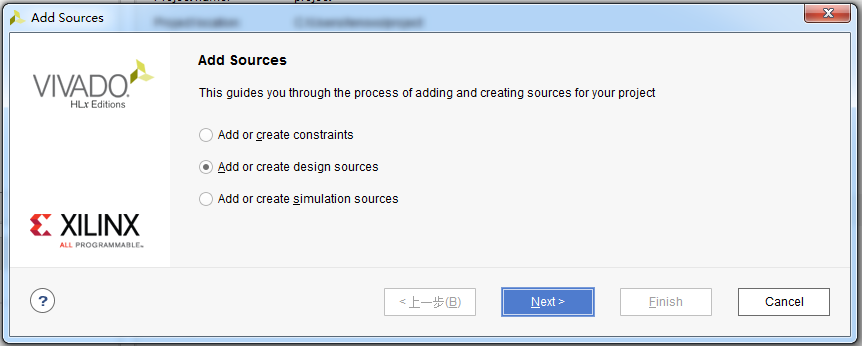
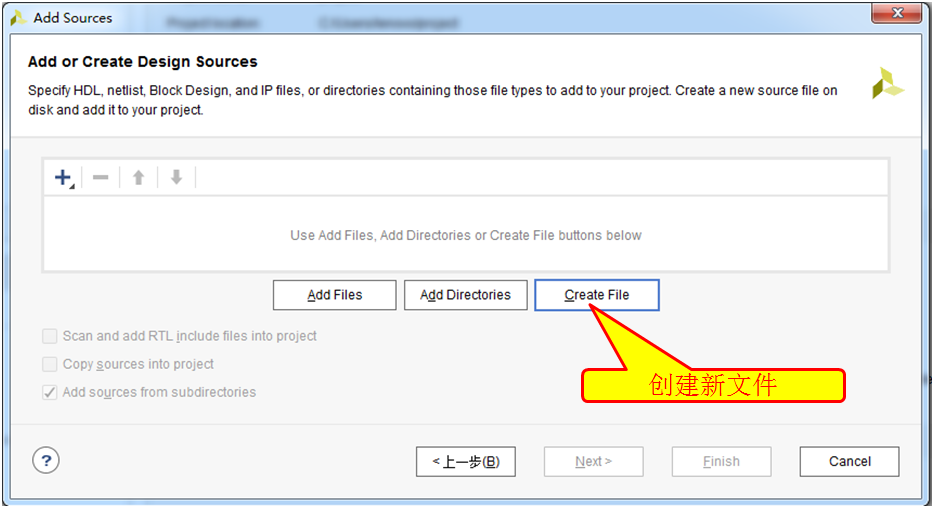


图 1.6 添加源文件

* + 还有两种添加源文档的选项，如图 1.7 所示：

图 1.7 选择文件类型

* + 选择 Create File…在弹出下面窗口填写新建源文件名称为 led



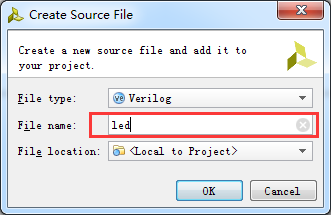


图 1.8 创建文件

* + 创建完成点击 Finish

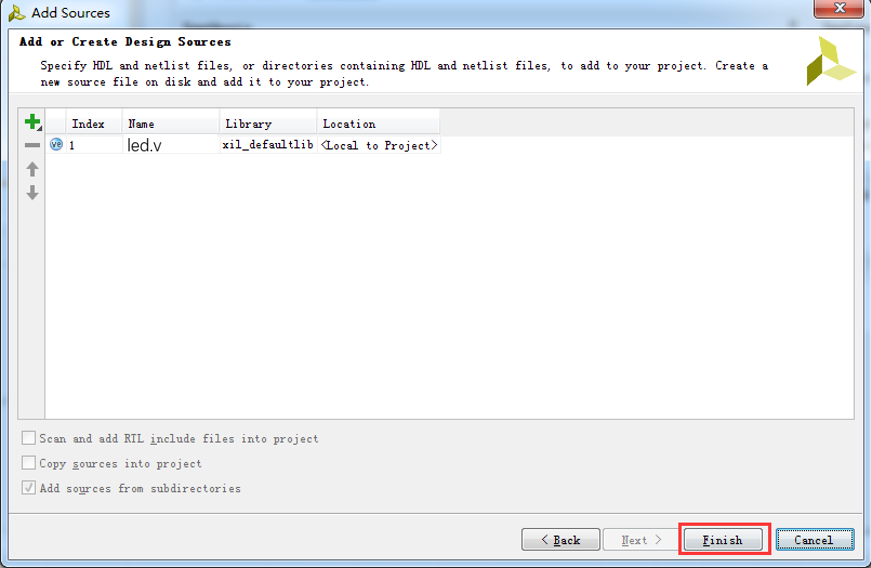
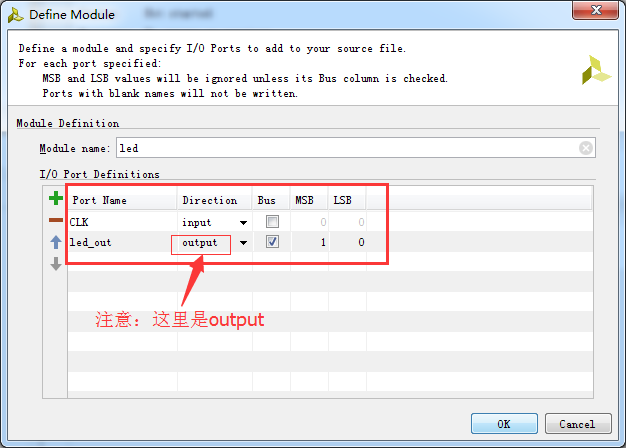


图 1.9 创建完成

* + 填写模块名称和端口，如图 1.10 所示：

图 1.10 Define Module 窗口

1. Verilog HDL 程序输入。

在用户区 Verilog HDL 文件窗口中输入源程序，保存时文件名与实体名保持一致。如图 1.11 所示：

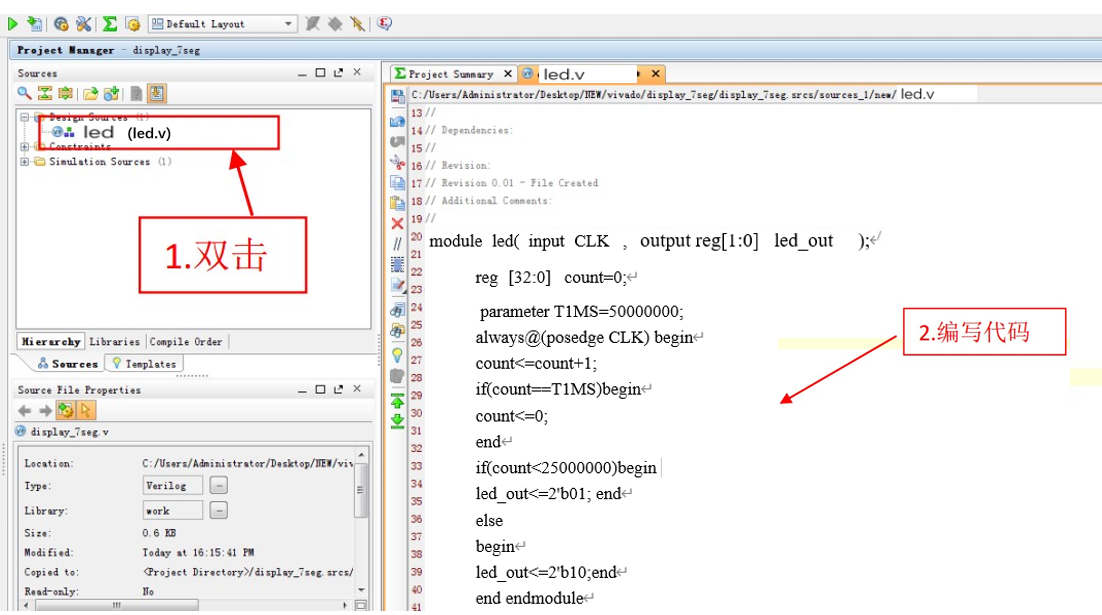


图 1.11 Verilog 代码编辑窗口

编辑代码如下：

module led( input CLK , output reg[1:0] led\_out );

reg [32:0] count=0;

parameter T1MS=50000000; always@(posedge CLK) begin

count<=count+1; if(count==T1MS) begin

count<=0; end

if(count<25000000) begin led\_out<=2'b01; end

else begin

led\_out<=2'b10; end

end endmodule

(4 ) Vivado 程序编译。

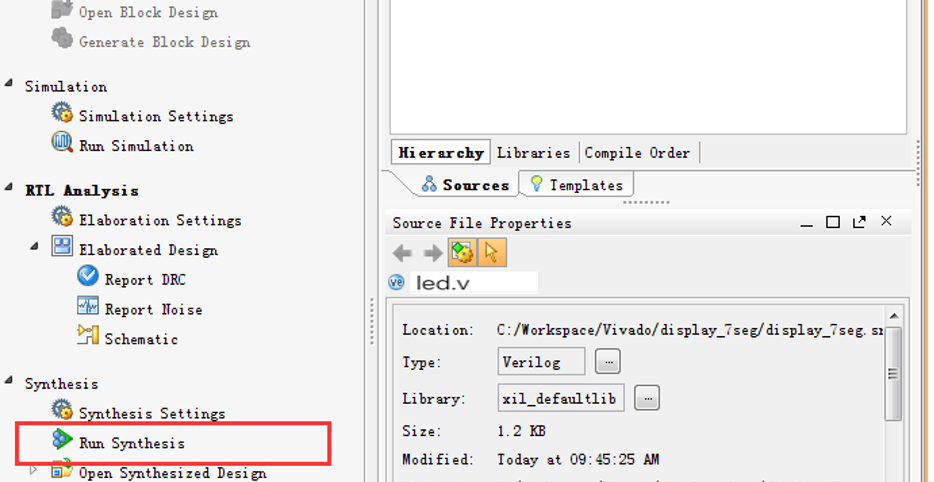
* 完成 Synthesize 的综合编译。如图 1.12 所示：

图 1.12 Verilog 代码 Synthesize 综合编译

* 编译成功后双击 Schematic 可以查看 RTL 级电路图。如图 1.13 所示：

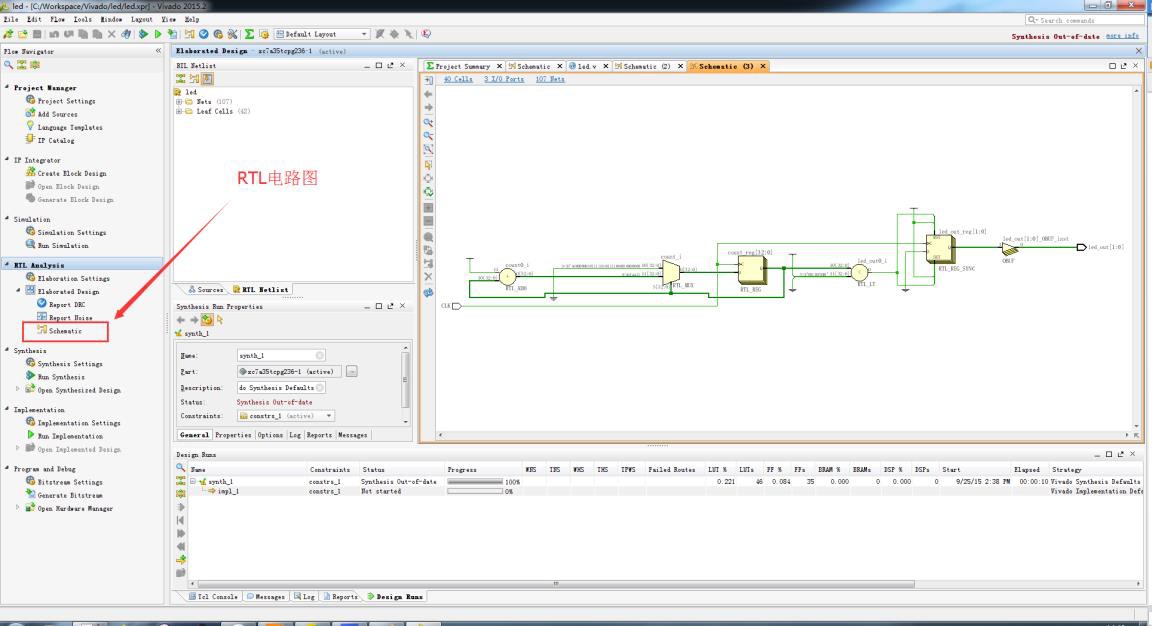


图 1.13 RTL 级电路图

**（二）一个简单编码器电路的设计**

**在** Vivado **环境下重新创建一个新项目，具体的步骤可以参考前面内容。**

1、编码器的实现

编码器通常分为两大类： 普通编码器和优先编码器。 其中普通编码器就是对某一个给定时刻只能对一个输入信号进行编码的编码器， 它的输入端口不允许同一时刻出现两个以上的有效输入信号； 优先编码器就是对某一个给定时刻只对优先级最高的输入信号进行编码的编码器，它的输入端口允许多个输入信号同时有效。

现以编码器为例，介绍普通编码器的 Verilog HDL 语言程序设计。通常，四至二线编码器的逻辑电路符号如图 1.14 所示，真值表如表 1.1 所示。不难看出该编码器的工作原理为：编码器将对四个输入信号进行编码操作，然后以两位二进制码的形式输出，这里输入信号为低电平有效。

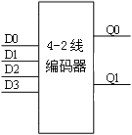


图 1.14 四至二线编码器的电路符号表 1.1、四至二线编码器的真值表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| D3 | D2 | D1 | D0 | Q1 | Q0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

## 具体操作过程如下：

* 1. 利用项目向导，建立一个新项目，建议工程名为 exper2。
  2. 新建一个 Verilog HDL 文件，并输入源程序：

module encoder4\_2(q,d); input[3:0] d;

output[1:0] q;

reg[1:0] q;

always@(d) begin case(d)

4'b0111: q<=2'b11;

4'b1011: q<=2'b10;

4'b1101: q<=2'b01;

4'b1110: q<=2'b00;

default: q<=2'bzz; endcase

end endmodule

1. 对源程序进行语法检查并编译。
2. 对项目进行时序逻辑功能仿真。

**（三）一个简单比较器电路的设计**

**在** Vivado **环境下重新创建一个新项目，具体的步骤可以参考前面内容。**

1.比较器的实现

数字比较器的设计，通常依据两组二进制数码的数值大小来进行比较，即

a>b、a=b 或 a<b，这三种情况有一种值为真。比较器的电路符号如图 1.15。

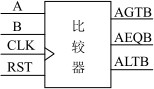


图 1.15 比较器电路符号

各引脚说明： A、B：皆为二位信号； CLK：时钟脉冲输入； RST：清除控制。

AGTB ：当 A>B 时，其值为 1，否则为 0； AEQB ：当 A=B 时，其值为 1，否则为 0； ALTB ：当 A<B 时，其值为 1，否则为 0。

其操作过程同译码器的实现，这里不再赘述。顶层文件名可以设为comp（）。

## 源程序如下：

module comp(CLK,RST,A,B,AGTB,ALTB,AEQB);

input CLK,RST;

input[1:0] A,B;

output AGTB,ALTB,AEQB; reg AGTB,ALTB,AEQB;

always@(posedge CLK or negedge RST) begin

if(!RST)

begin AGTB<=0; AEQB<=0; ALTB<=0;

end else

begin if(A>B)

begin AGTB<=1; AEQB<=0; ALTB<=0;

end

else if(A==B) begin AGTB<=0; AEQB<=1; ALTB<=0;

end

else begin

AGTB<=0; AEQB<=0; ALTB<=1;

end end

end endmodule

1. **一个简单全加器电路的设计**

**在** Vivado **环境下重新创建一个新项目，具体的步骤可以参考前面内容。**

1.全加器的实现

全加器其实就是考虑到进位的加法器。一位全加器的电路符号如图 1.16 所

示，真值表如表 1.2 所示。



图1.16 全加器电路符号表 表1.2 一位全加器真值表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 全加器输入 | | | 全加器输出 | |
| A | B | Cin | BCDout | Cout |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

具体操作步骤不再一一给出。这里仅给出一位全加器的源程序。**源程序如下：**

module ful\_adder(cout,sum,a,b,cin);

input a,b;

input cin; output sum; output cout; reg sum;

reg cout;

always@(a or b or cin) begin

{cout,sum}=a+b+cin;

end

endmodule

## 五、 实验结果

1. **Led灯的控制电路**

（1）.给关键代码添加注释，或者给出模块代码的流程图。

（2）.编程与自己的学号有关联的测试程序（也称激励文件），在测试程序中关键代码后给出注释。可以将延迟时间，对应于自己的学号，或者将学号\*2作为延迟时间，或者将学号的后6位作为延迟时间……

（3）.对电路进行仿真，根据仿真波形，验证电路功能是否正确。将仿真后获得的波形图截图保存后，放到实验报告中。

（4）.在Vivado中，通过项目管理窗口中的schematic菜单，生成与verilog代码对应的电路原理图，将该电路原理图放到实验报告中。

1. **编码器**

（1）.给关键代码添加注释，或者给出模块代码的流程图。

（2）.编程与自己的学号有关联的测试程序（也称激励文件），在测试程序中关键代码后给出注释。可以将延迟时间，对应于自己的学号，或者将学号\*2作为延迟时间，或者将学号的后6位作为延迟时间……

（3）.对电路进行仿真，根据仿真波形，验证电路功能是否正确。将仿真后获得的波形图截图保存后，放到实验报告中。

（4）.在Vivado中，通过项目管理窗口中的schematic菜单，生成与verilog代码对应的电路原理图，将该电路原理图放到实验报告中。

1. **比较器：**

（1）.给关键代码添加注释，或者给出模块代码的流程图。

（2）.编程与自己的学号有关联的测试程序（也称激励文件），在测试程序中关键代码后给出注释。可以将延迟时间，对应于自己的学号，或者将学号\*2作为延迟时间，或者将学号的后6位作为延迟时间……

（3）.对电路进行仿真，根据仿真波形，验证电路功能是否正确。将仿真后获得的波形图截图保存后，放到实验报告中。

（4）.在Vivado中，通过项目管理窗口中的schematic菜单，生成与verilog代码对应的电路原理图，将该电路原理图放到实验报告中。

1. **全加器**

（1）.给关键代码添加注释，或者给出模块代码的流程图。

（2）.编程与自己的学号有关联的测试程序（也称激励文件），在测试程序中关键代码后给出注释。可以将延迟时间，对应于自己的学号，或者将学号\*2作为延迟时间，或者将学号的后6位作为延迟时间……

（3）.对电路进行仿真，根据仿真波形，验证电路功能是否正确。将仿真后获得的波形图截图保存后，放到实验报告中。

（4）.在Vivado中，通过项目管理窗口中的schematic菜单，生成与verilog代码对应的电路原理图，将该电路原理图放到实验报告中。