**北京科技大学数字逻辑实验报告**

学院： 计算机与通信工程 专业：物联网工程 班级： 物联 201

姓名： 赵方程 学号：42024137 实验日期：2021 年 11月 11 日

**实验名称：**实验一、基础练习

**实验目的：**熟悉实验环境，学习如何安装Vivado、如何使用 Vivado 2018创建工程、代码编辑、RTL分析、仿真等设计流程。

**实验内容：**

（1）学习所有视频，了解 Vivado 设计流程和功能

（2）按照“01-流水灯设计.pdf”完成流水灯实验；

（3）Vivado 代码编辑和RTL分析；

**实验结果与分析：**

（1）观看提供的所有视频资料；学习：Vivado设计流程中的基本概念.pdf、约束文件.ppt。回答以下问题：

* 描述 Vivado 的设计流程

使用Vivado 创建工程，选择相应的part / board 后创建工程

编写Verilog 代码

进行仿真，观察输出情况

综合，添加约束，bind port with corresponding Package pin

实现

Generate BitStream

下载，连接板卡与电脑，使用已生成的BitStream 文件进行编程

* 什么是网表

网表是包括单元、引脚、端口和网络等，是Verilog语言经过综合（或编译）后的结果。

* Vivado 设计流程中，Synthesis 的作用是什么？

Synthesis，即综合，会将高级抽象层次的电路描述转化成较低层次的描述，将Verilog语言转化成与门、或门、非门、触发器等基本逻辑单元的互连关系，生成网表。

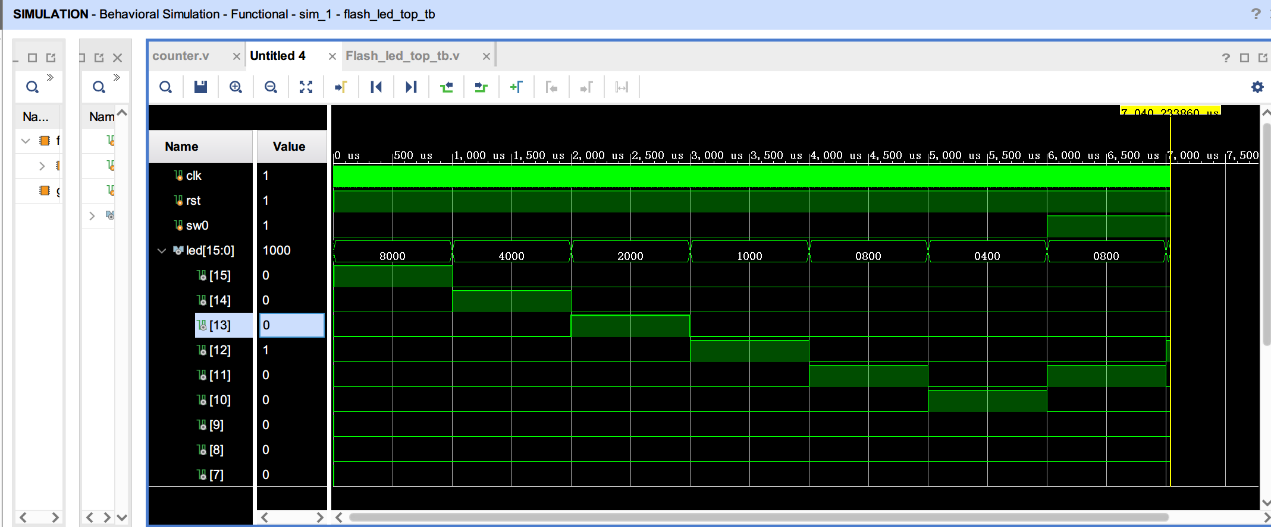
并且能像编译器一样高度优化语言描述的电路设计，使之更高效。

* Vivado 设计流程中，Implementation 的作用是什么？

Implementation，即实现，完成了门级网表的布局布线，将网表中的各个门电路映射至CLB中，利用布线资源将CLB根据逻辑关系连接在一起。

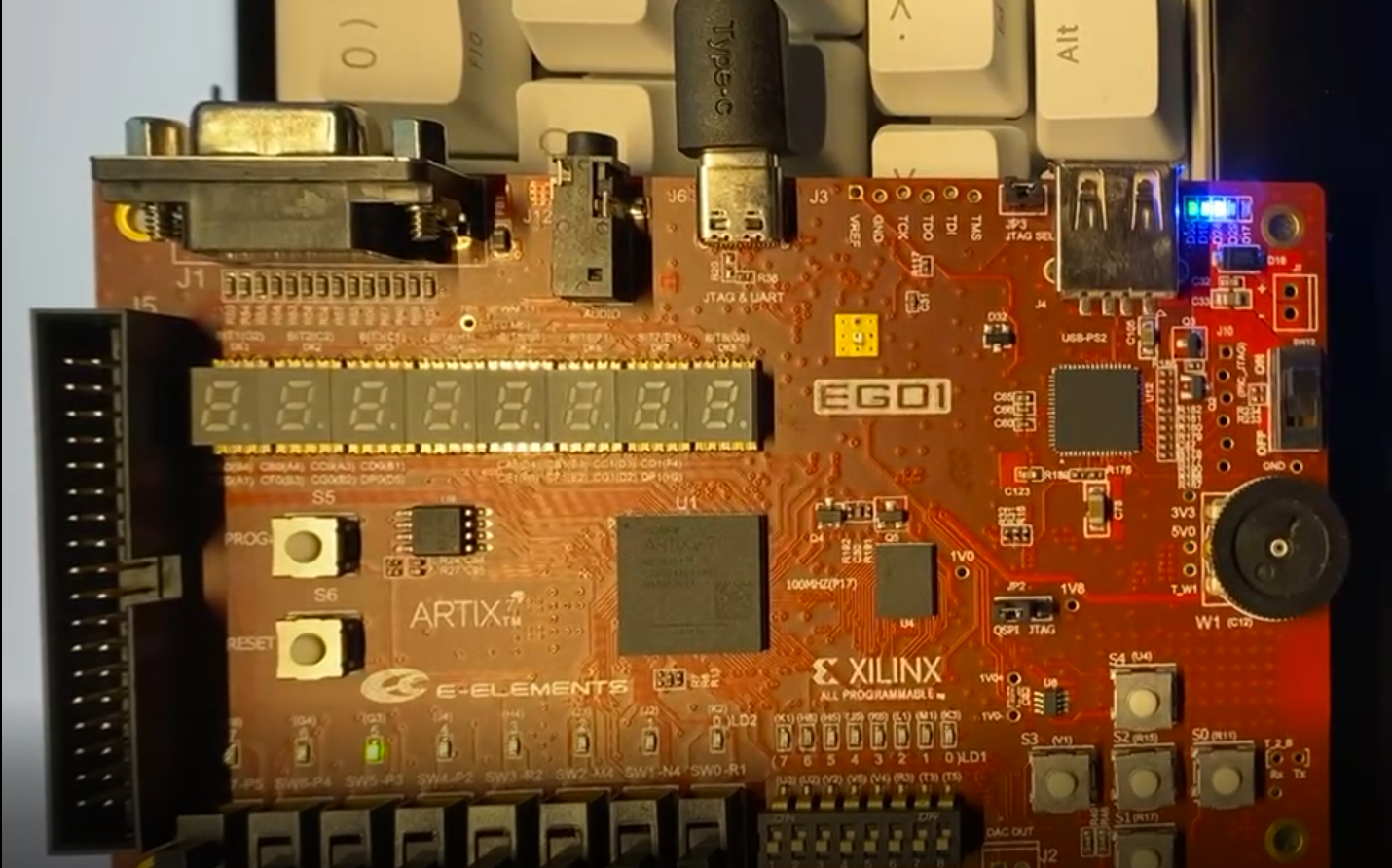
（2）按照“01-流水灯设计.pdf”完成流水灯实验；

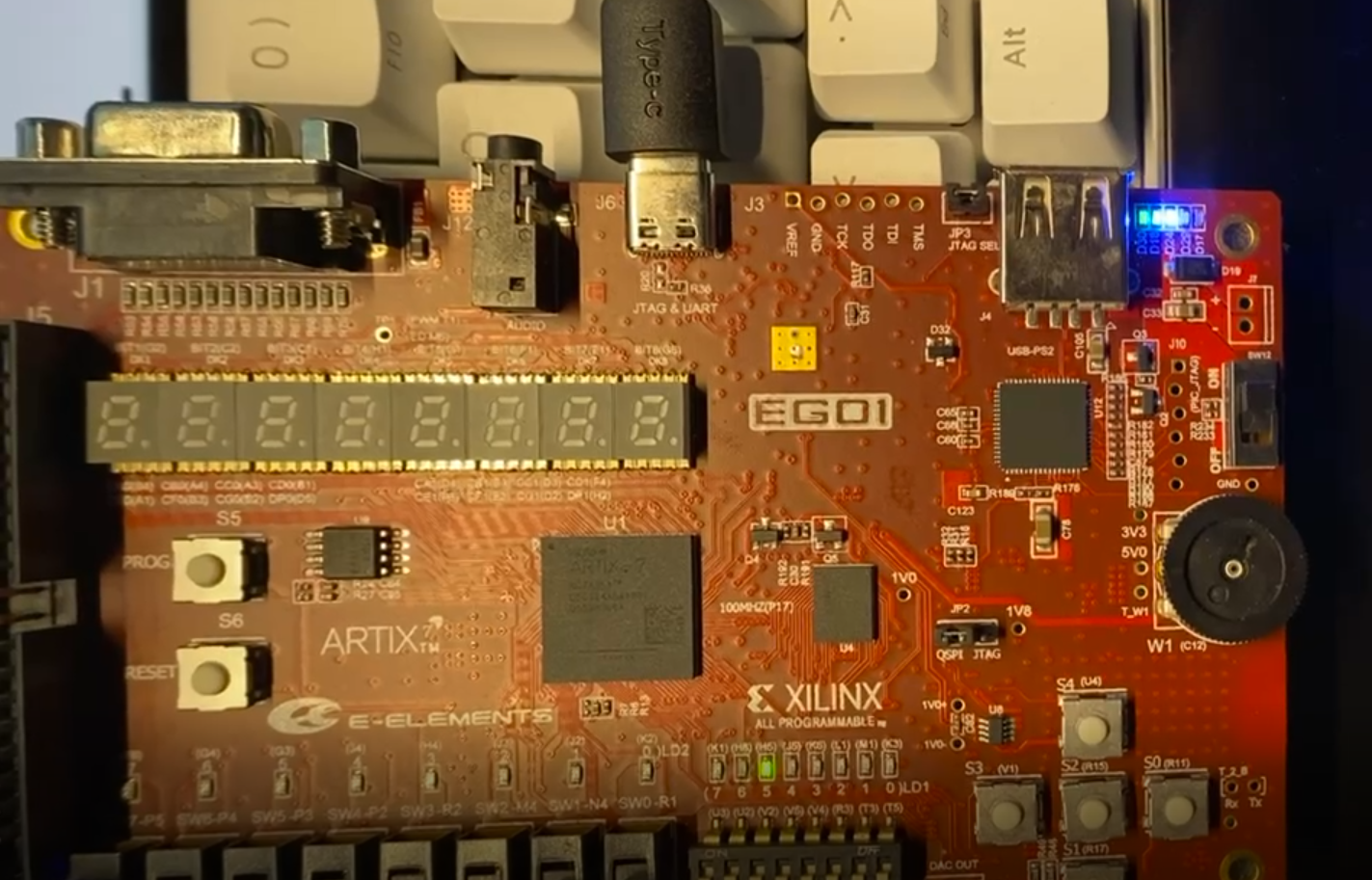
* 给出流水灯仿真结果截图，对波形进行简要解释：



仿真结果表明，每1ms 向右移动一个LED，经过6ms后移动至 led10，之后改变方向向左移动，仿真波形与预期结果一致。

* 给出板子运行结果照片，以及你的操作过程：





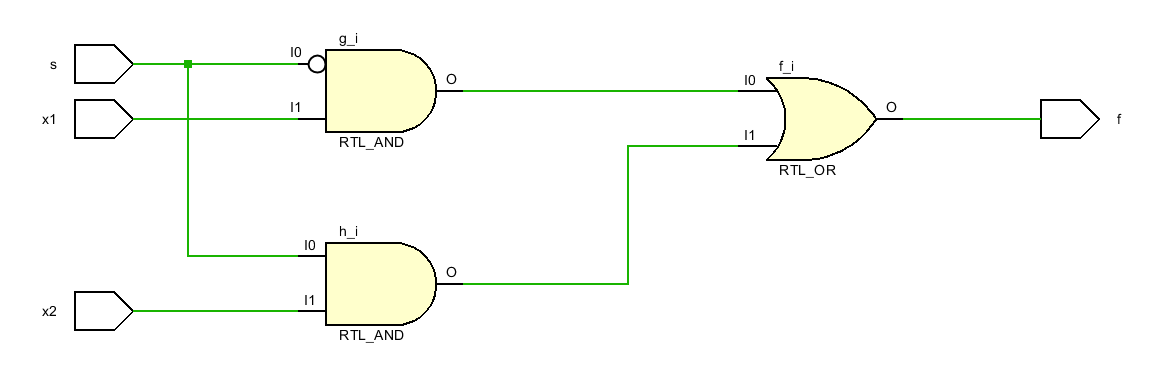
操作过程:编写代码后进行仿真，仿真结果与预期一致后进行Implementation后生成BitStream文件,再使用线缆连接电路板,打开目标后使用 Auto Connect进行连接，之后使用Program Device载入BitStream文件.完成后流水灯即按照设置的逻辑周期性闪烁,使用R1按钮控制流水灯闪烁方向。

* counter模块的作用是什么？

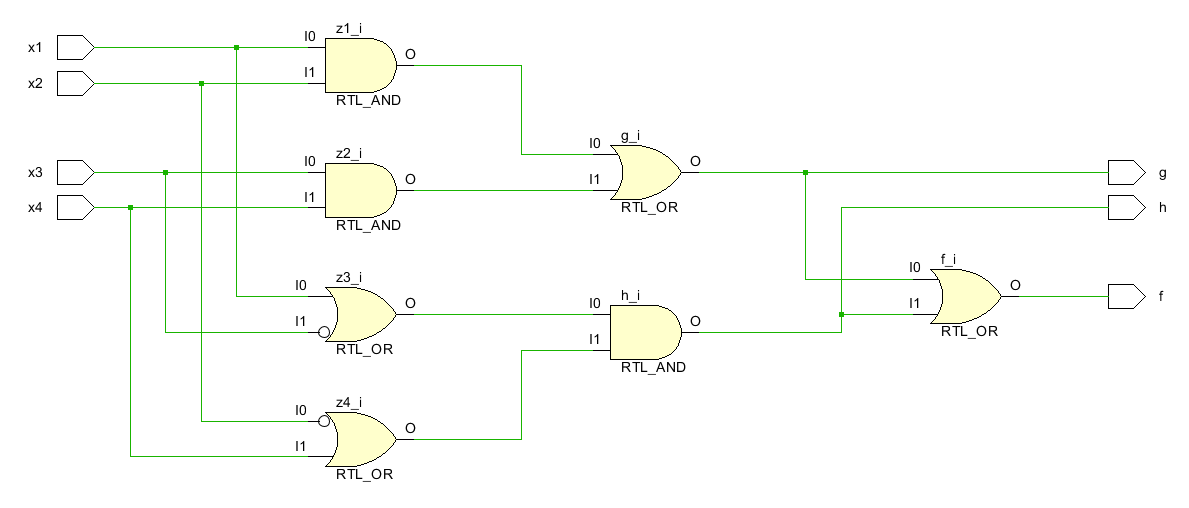
Counter，计数器，根据时间变化更改first , 待first到达临界值后更改second ，通过first与second 信号控制Flash led 闪烁

（3）创建一个工程，自己指定工程位置和工程名称，新建空白源程序文件，依次完成下面代码编辑和RTL分析：

* 分别编写教材图2.37、2.38例子，观察vivado工具的RTL分析结果，截图如下；

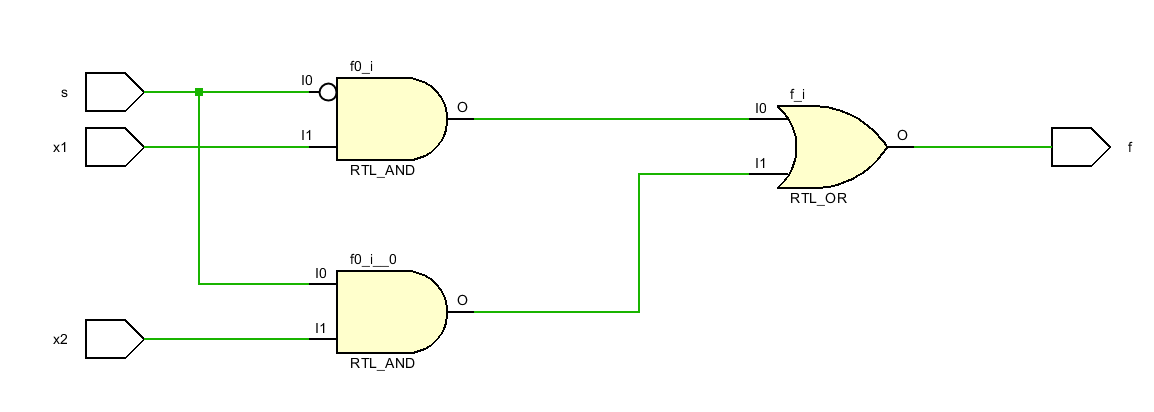


2.37

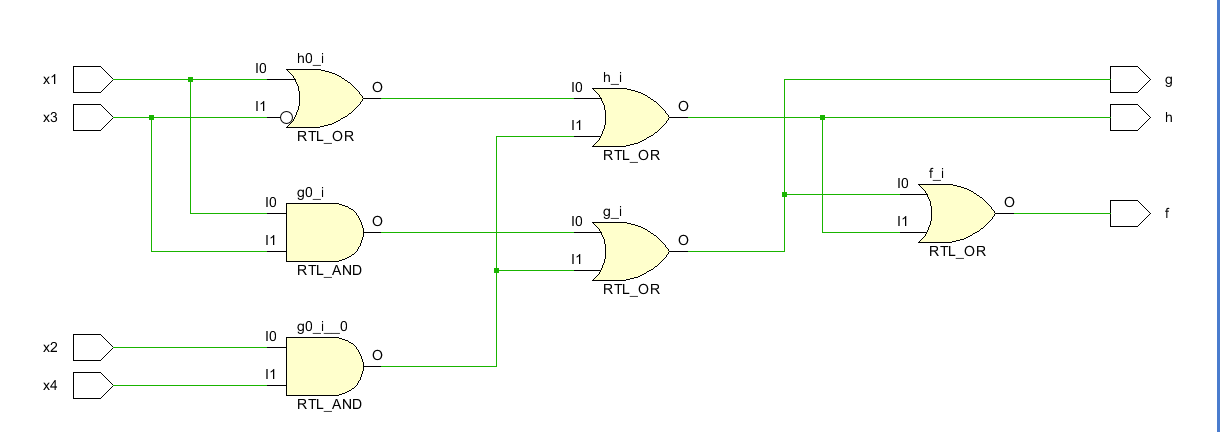


2.38

* 分别编写教材图2.40、2.41例子，观察vivado工具的RTL分析结果，截图如下；

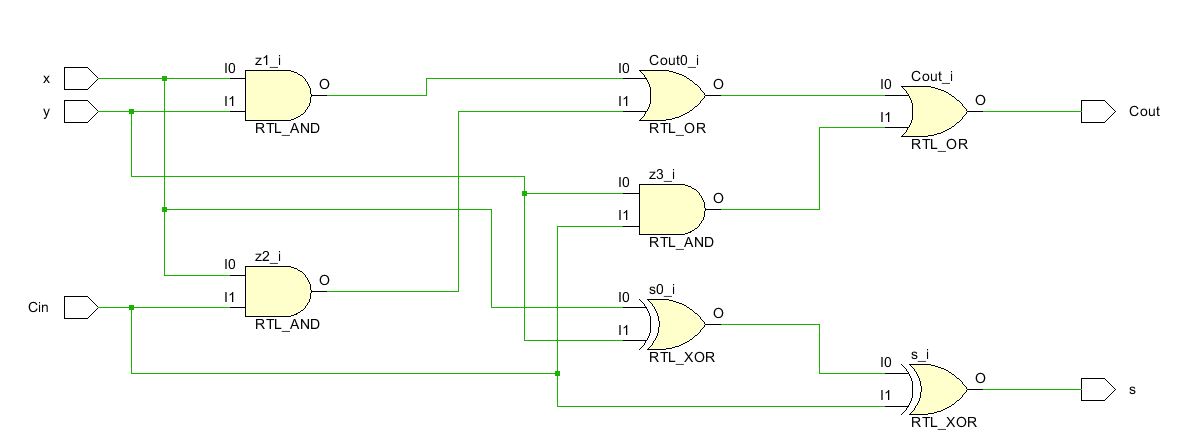


2.40

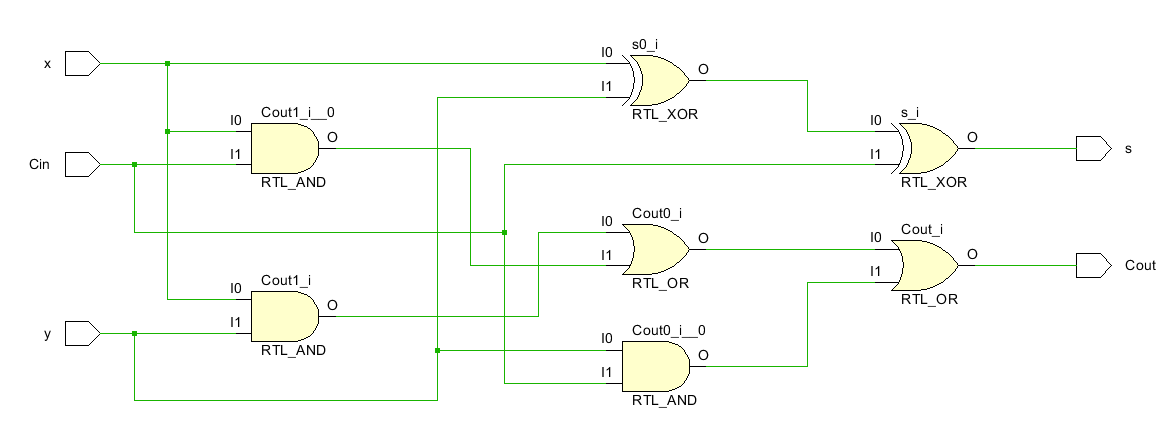


2.41

* 分别编写教材图3.18、3.20例子，对比 vivado工具的RTL分析结果，截图并给出



3.18

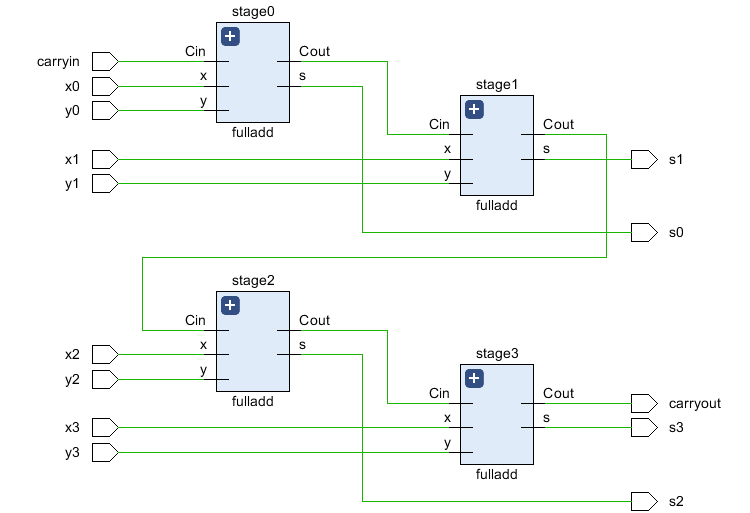


3.20

* 你对结果的理解；

观察发现两种代码综合得到的结果是一样的，逻辑语句与assign 语句的逻辑相同，于是综合的结果就是相同的。可见，逻辑相同的语句可以得到相同的综合结果，使得Verilog的编程风格更加灵活。

* 编写教材图3.22例子，观察vivado工具的RTL分析结果，截图并给出你对结果的理解；



在模块adder4里调用了全加器, 也就是说adder4是一个封装了fulladd模块的四位加法器, 使功能相似的模块实现了较高的内聚度，提高了封装性。

（4）实验中遇到哪些问题，是如何解决的。

1. Counter代码编写错误，三元运算符使用错误，后经过更正代码解决

2. 实现时报错，经过检查后发现I/O port 分配时没有全部分配，重新分配后解决

（5）本次实验的感受及建议

感受就是，实验过程中对Verilog的编程思维仍存在误解，难以将其带入到OOP和FP等模式中。