实验名称：_FPGA 实现全加器__姓名：____严旭铧_____学号：___3220101731____

专业：电气工程及其自动化
姓名：__严旭铧_____
学号：_3220101731_____
日期：_2024.4.2_____
地点：紫金港东三 406

浙江大学实验报告

课程名称：__电路与电子技术 2_实验___指导老师：___张伟___ _____成绩：_____

实验名称：_____FPGA 实现全加器_____实验类型：_数电实验_____同组学生姓名：褚玘铖

实验 13 FPGA 实现全加器

一、 实验目的

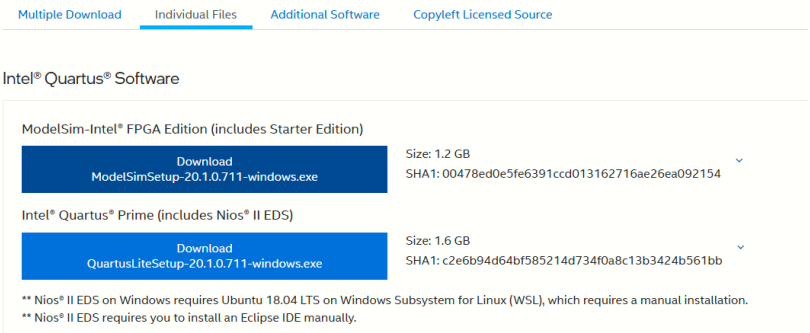
1. 了解数字电路基础知识
2. 掌握简单数字逻辑电路的设计
3. 学习 FPGA 开发工具 Quartus 及 VHDL 语言的使用

二、 实验要求

1. 设计并实现 1 位二进制全加器
2. 设计并实现 4 位二进制全加器

三、 实验准备与软件安装

1. 在如下网址下载 Quartus Prime Lite 20.1 版本。注意这里下载的时候不要选太高的版本，一个是可能不带 ModelSim 组件，另一个可能是不支持比较旧的板子。Intel® Quartus® Prime Lite Edition Design Software Version 20.1 for Windows。
2. 如果硬盘空间够就下载集成版本，否则建议下载分离版本。本课程中只用到 MAX 10 系列的板



子，因此可以分离下载软件和硬件库，硬件库下载这个就行。两个都下载完之后再进行安装，勾选这个库就能装上了。

四、 实验内容

1. 1 位二进制全加器
   (1) 全加器真值表

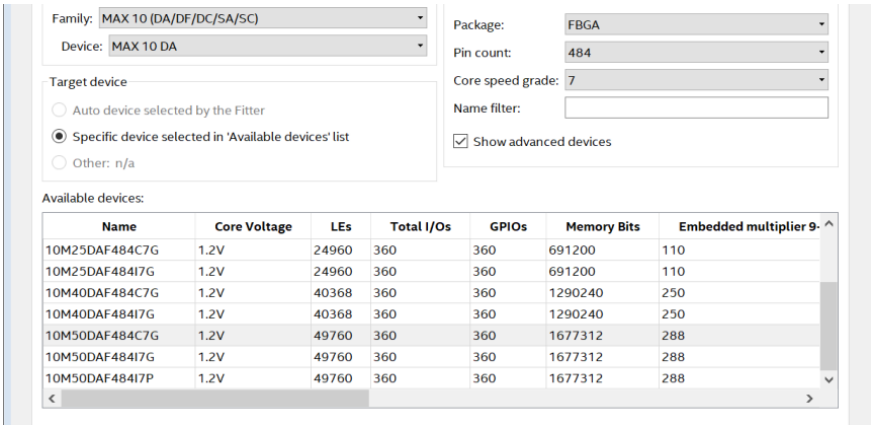
输入			输出	
a	**b**	**c0**	**s**	**c1**

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

(2) 实验步骤：

i. 创建文件夹 add1bit 于路径 G:\Code\Quartus\add1bit，打开 Quartus，选择 File > New Project Wizard…菜单命令， 在 Introduction 界面中点击 Next 按键,在弹出的对话框中输入项目文件夹（上面的路径），项目名称和顶层实体名这里均命名为 add1bit。

ii. 点击 Next 按钮进入 Project Type 界面， 选择 Empty project 选项，点击 Next 按键进入 Add Files 界面，不选择任何文件，再点击 Next 按键进入器件选择(Family,Device & Board Settings)界面，选取 MAX10 系列的 10M50DAF484C7G。



iii. 点击 Next 按键进入 EDA Tool Settings 界面和 Summary 界面， 由于在这两个界面均不需要进行设置，可点击 Next 按键然后点击 Finish 按键完成新项目的设置向导。选择 File>New…菜单命令，设计文件类型选择 VHDL File。进入到项目设计界面， 将缺省文件名 Vhdl.vhd 另存为(File>Save As…)文件 add1bit.vhd， 输入 1 位全加器的 VHDL 源程序如下：

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity add1bit is
port( a : in std_logic;
      b : in std_logic;
      c0 : in std_logic;
      s : out std_logic;
      c1 : out std_logic );
end add1bit;

architecture behavioral of add1bit is
begin
c1<=(a and b) or (a and c0) or (c0 and b);
s<=a xor b xor c0;
end behavioral;
```

注释

这里是导入 IEEE 库。

定义实体 add1bit

Port 后接端口定义，

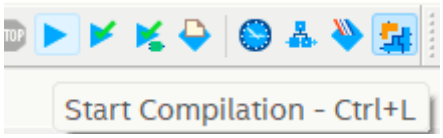
a,b,c0 定义为逻辑输入类型，a,b 分别代表两个加数，c0 是低位的进位，s 表示加法结果的和，c1 代表进位输出。

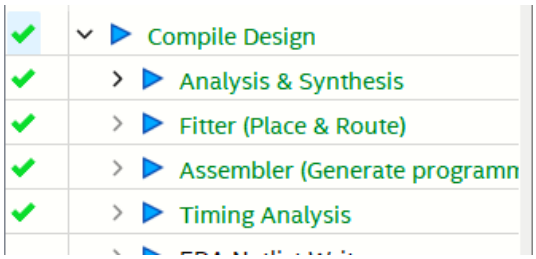
定义 add1bit 实体的行为架构

计算进位输出 c1，任意两个输入端口的逻辑与的逻辑或

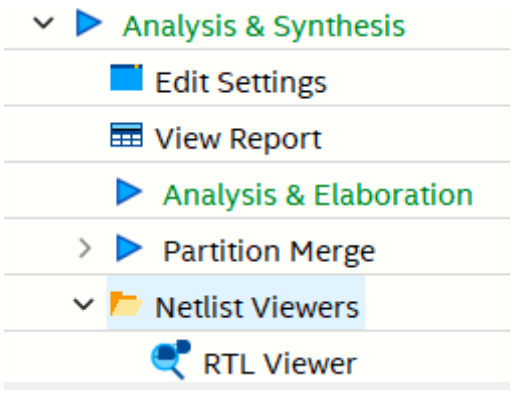
计算和输出 s，三个输入端口的逻辑异或，实现全加器的和运算

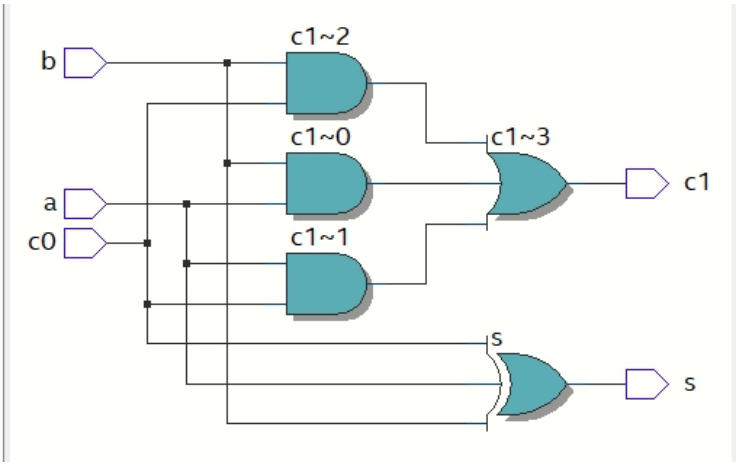
iv. 编译，如有报错再检查一下代码，直到不再报错（如右下）。





此时可以在 Netlist Viewers 文件夹中查看 RTL 图（逻辑图），也可以查看状态机等。





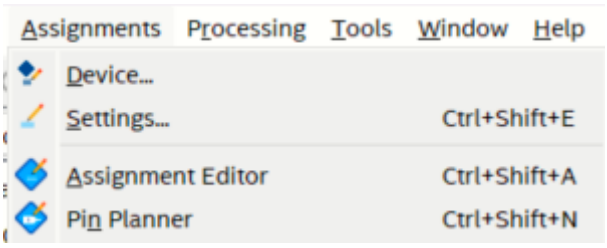
可以看到这个很漂亮。

v. 管脚分配。

执行 Assignments>Pin planner 菜单命令，进行引脚分配后的引脚分配编辑器窗口。

管脚按下面这样分配。

在这里将 3 个拨码开关 SW2、SW1、SW0 作为全加器的输入，将 2 个 LED 灯 LEDR1、LEDR0 作为全加器的输出



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standar
a	Input	PIN_D12	7	B7_N0	PIN_D12	2.5 V
b	Input	PIN_C11	7	B7_N0	PIN_C11	2.5 V
c0	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V
c1	Output	PIN_A9	7	B7_N0	PIN_A9	2.5 V
s	Output	PIN_A8	7	B7_N0	PIN_A8	2.5 V

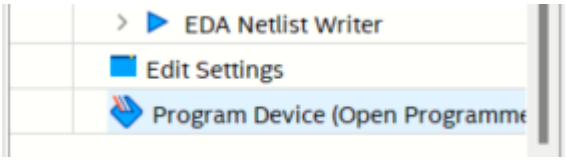
紧接着进行未用引脚的设置，执行 Assignments>Device 菜单命令，继续点击 Device and Pin Options…按键，出现 Device and Pin Options 设置窗口，选中 Unused Pins 条目，在 Reserve all unused pins 下拉列表中选择 As input tri-stated 设置项，点击 OK 按键返回主界面。重新对项目进行完整编译后生成 add1bit.sof 文件，经程序下载后就可以在 DE10-Lite 开发板上验证程序的正确与否了。

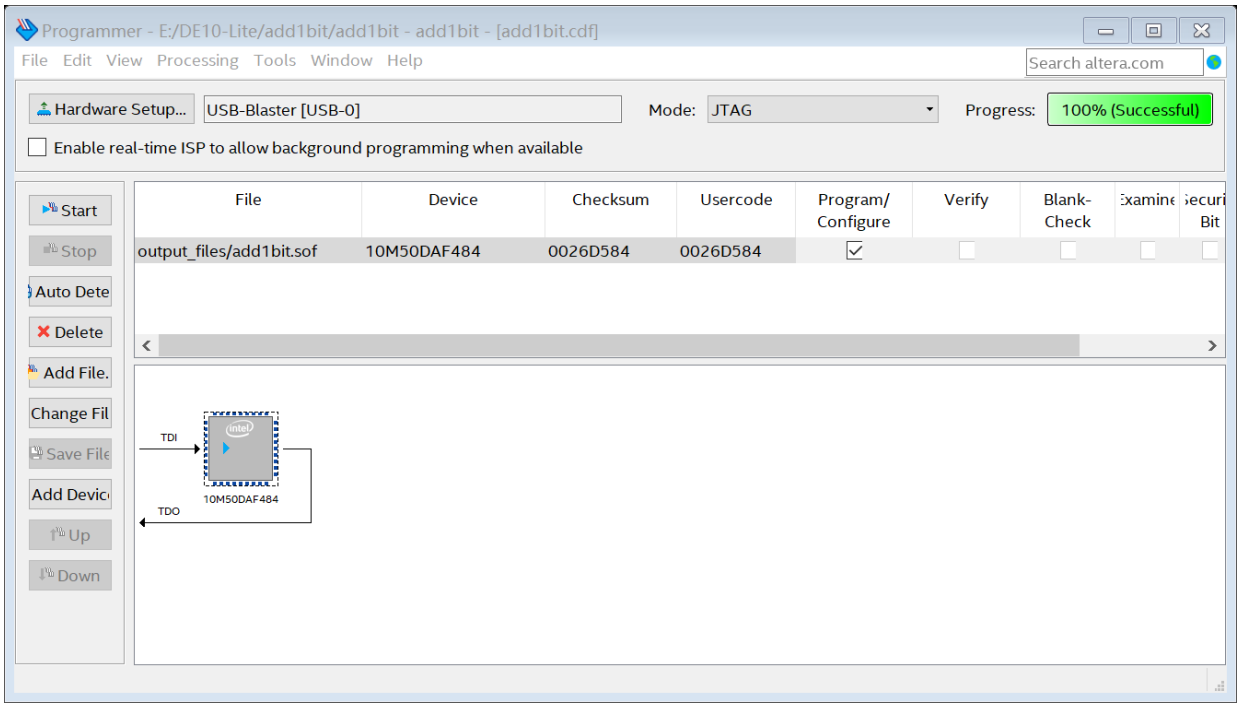
vi. 程序烧录与验证

一定要确保已经正确安装好驱动，否则可能导致连接不上 FPGA 开发板。

单击 Program Device，进入程序烧录界面。将硬件连接选为 USB-Blaster[USB-0]即可。

之后点击 Start 就行。
如果不能选中那么就是没连接好，或者驱动有问题。Progress 到 100%就是读完了，可以在板子上测试。

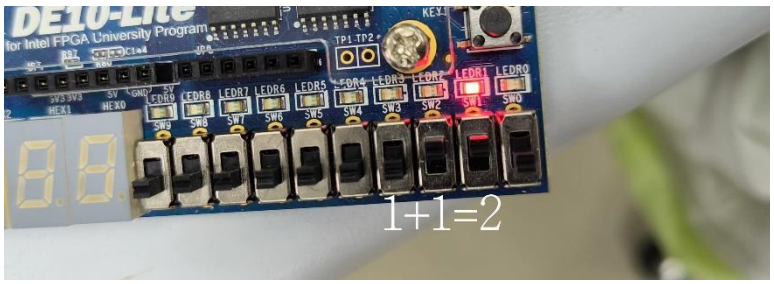




(3) 测试结果

1+1+1=（11）B

1+1=（10）B



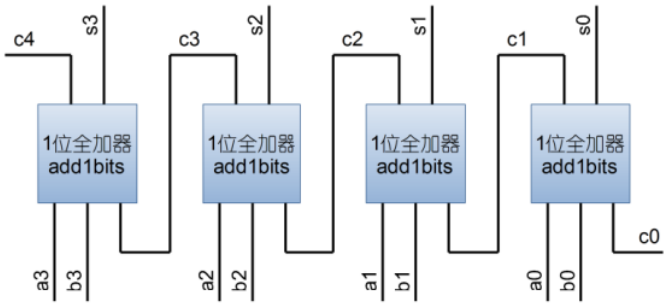
其他情况也符合真值表，这里不一一列举了。

2. 4 位二进制全加器

(1) 设计原理

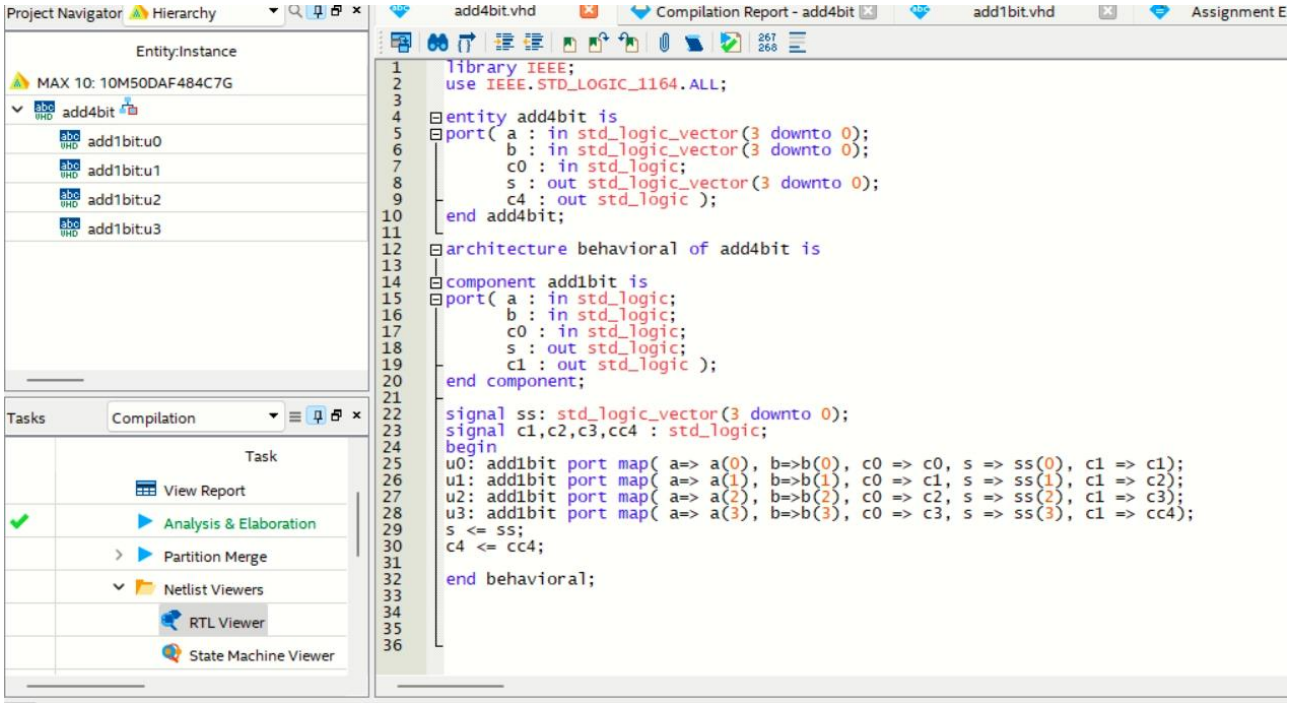
从图中可以看出， 4 位全加器由 4 个 1 位全加器级联而成，把 4 位全加器作为一个顶层设计， 1 位全加器就是一个底层设计，相当与一个子程序。在 VHDL 文件中可以用元

件说明(component)来表述。



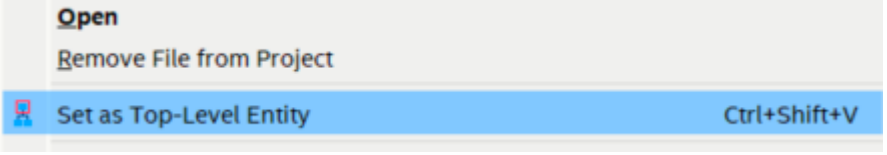
(2) 实验步骤

i. 在启动 Quartus Prime Lite Edition 集成开发软件后，按建立项目向导的 5 个步骤建立 add4bit 项目，项目文件夹为 G:\Code\Quartus\add4bit。选择 File>New…菜单命令，设计文件类型选择 VHDL File。进入到项目设计界面，输入 4 位全加器的 VHDL 源程序。最下面是代码的注释。



在这个 4 位全加器的源程序中，定义了一个 1 位全加器(add1bit)的元件，然后通过“port map”指令将 1 位全加器的 5 个输入输出信号与 4 位全加器的信号一一对应地联系起来了。编写完 4 位全加器的顶层设计文件后，还得在这个项目文件夹中编写 1 位全加器 add1bit.vhd 的底层设计文件。需要注意的是，为了使项目编译能正常进行，底层设计文件必须保存在与顶层设计文件相同的目录下。

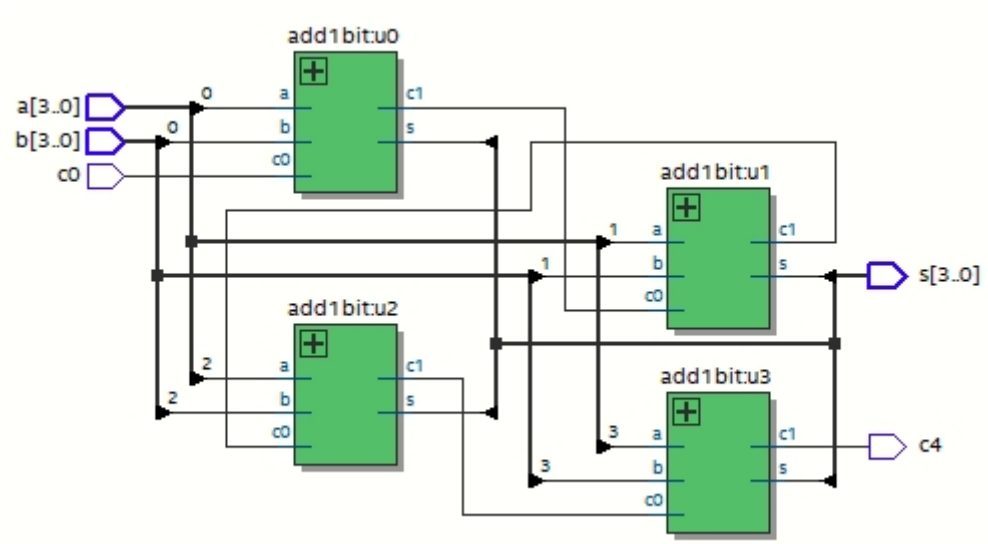
ii. 编译，编译成功后如上图所示。这里要记得把 add4bit 设置为顶层文件。



实验名称：_FPGA 实现全加器__姓名：____严旭铧_____学号：___3220101731_____

```
library IEEE; -- 引入 IEEE 库
use IEEE.STD_LOGIC_1164.ALL; -- 使用 IEEE 标准逻辑类型库

entity add4bit is -- 定义一个名为 add4bit 的实体，用于四位全加器
port( a : in std_logic_vector(3 downto 0); -- 输入端口 a，表示一个四位加数
      b : in std_logic_vector(3 downto 0); -- 输入端口 b，表示另一个四位加
数
      c0 : in std_logic; -- 输入端口 c0，表示最低位的进位输入
      s : out std_logic_vector(3 downto 0); -- 输出端口 s，表示四位加法结果
的和
      c4 : out std_logic ); -- 输出端口 c4，表示最高位的进位输出
end add4bit; -- 结束实体的定义

architecture behavioral of add4bit is -- 定义 add4bit 实体的行为架构

component add1bit is -- 内部声明一个一位全加器组件
port( a : in std_logic; -- 输入端口 a，表示加数
      b : in std_logic; -- 输入端口 b，表示加数
      c0 : in std_logic; -- 输入端口 c0，表示低位进位输入
      s : out std_logic; -- 输出端口 s，表示加法结果的和
      c1 : out std_logic ); -- 输出端口 c1，表示进位输出
end component; -- 结束组件声明

signal ss: std_logic_vector(3 downto 0); -- 中间信号 ss，用于存储加法结果的
和
signal c1, c2, c3, cc4 : std_logic; -- 中间信号，用于存储进位信息

begin
u0: add1bit port map( a=> a(0), b=>b(0), c0 => c0, s => ss(0), c1 =>
c1); -- 实例化第一个一位全加器，用于最低位加法
u1: add1bit port map( a=> a(1), b=>b(1), c0 => c1, s => ss(1), c1 =>
c2); -- 实例化第二个一位全加器，用于次低位加法
u2: add1bit port map( a=> a(2), b=>b(2), c0 => c2, s => ss(2), c1 =>
c3); -- 实例化第三个一位全加器，用于次高位加法
u3: add1bit port map( a=> a(3), b=>b(3), c0 => c3, s => ss(3), c1 =>
cc4); -- 实例化第四个一位全加器，用于最高位加法
s <= ss; -- 将中间信号 ss 的值赋给输出端口 s
c4 <= cc4; -- 将中间信号 cc4 的值赋给输出端口 c4

end behavioral; -- 结束架构的定义
```

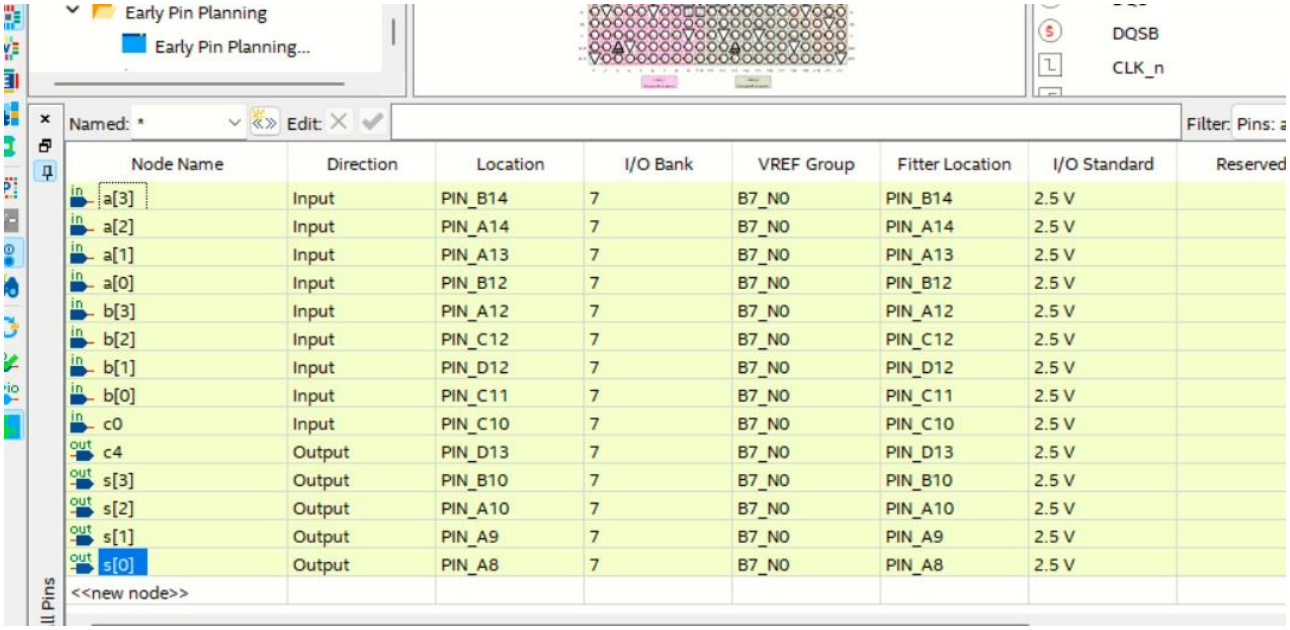
编译后可以查看 RTL 图



很清晰，4 位全加器由 4 个一位全加器组成。

iii. 管脚分配：

执行 Assignments>Pin planner 菜单命令，进行引脚分配后的引脚分配编辑器窗口如图 2.2.4 所示。在这里将拨码开关 SW8、 SW7、 SW6、 SW5 作为 a3、 a2、 a1、 a0 信号输入，将拨码开关 SW4、 SW3、 SW2、 SW1 作为 b3、 b2、b1、 b0 信号输入，将拨码开关 SW0 作为 c0 信号输入，将 LEDR3、LEDR2、LEDR1、LEDR0 作为 s3、s2、 s1、 s0 信号的输出，将 LEDR4 作为 c4 信号的输出。



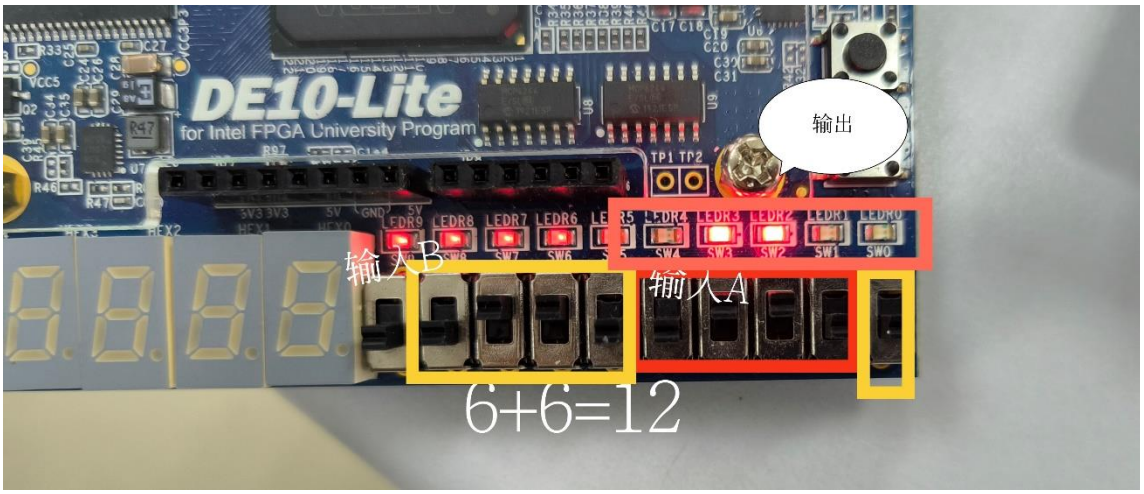
Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved
a[3]	Input	PIN_B14	7	B7_N0	PIN_B14	2.5 V	
a[2]	Input	PIN_A14	7	B7_N0	PIN_A14	2.5 V	
a[1]	Input	PIN_A13	7	B7_N0	PIN_A13	2.5 V	
a[0]	Input	PIN_B12	7	B7_N0	PIN_B12	2.5 V	
b[3]	Input	PIN_A12	7	B7_N0	PIN_A12	2.5 V	
b[2]	Input	PIN_C12	7	B7_N0	PIN_C12	2.5 V	
b[1]	Input	PIN_D12	7	B7_N0	PIN_D12	2.5 V	
b[0]	Input	PIN_C11	7	B7_N0	PIN_C11	2.5 V	
c0	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V	
c4	Output	PIN_D13	7	B7_N0	PIN_D13	2.5 V	
s[3]	Output	PIN_B10	7	B7_N0	PIN_B10	2.5 V	
s[2]	Output	PIN_A10	7	B7_N0	PIN_A10	2.5 V	
s[1]	Output	PIN_A9	7	B7_N0	PIN_A9	2.5 V	
s[0]	Output	PIN_A8	7	B7_N0	PIN_A8	2.5 V	
<<new node>>							

紧接着进行未用引脚的设置，执行 Assignments>Device 菜单命令，继续点击 Device and Pin Options…按键，出现 Device and Pin Options 设置窗口，选中 Unused Pins 条目，在 Reserve all unused pins 下拉列表中选择 As input tri-stated 设置项，点击 OK 按键返回主界面。重新对项目进行完整编译后生成 add4bit.sof 文件，经程序下载后就可以在 DE10-Lite 开发板上验证程序的正确与否了。

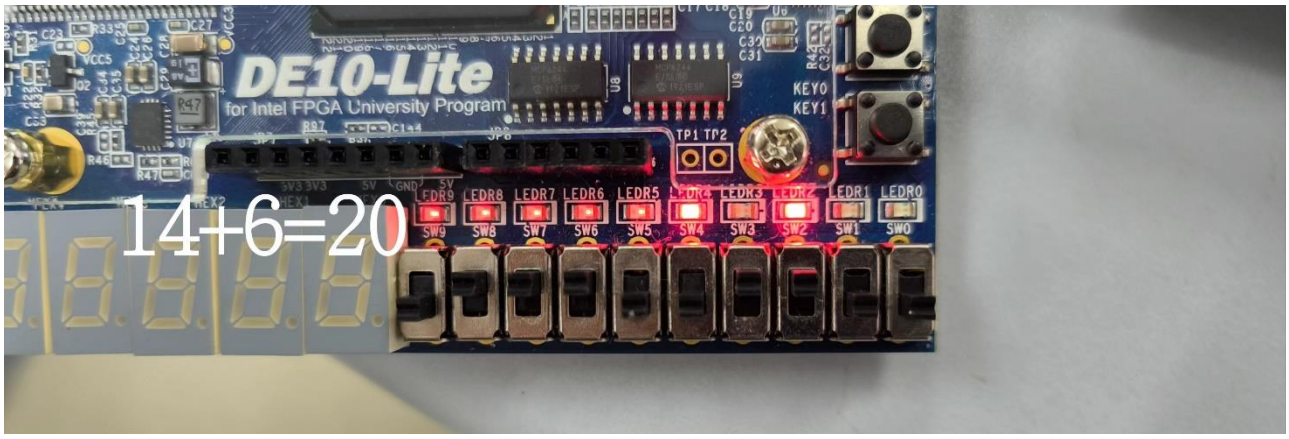
实验名称：_FPGA 实现全加器__姓名：____严旭铧_____学号：___3220101731_____

(3) 测试结果

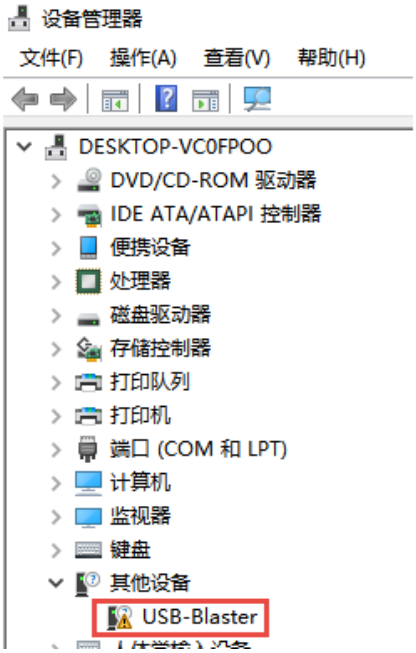
6+6=12=（01100）B



14+6=20（10100）B



五、 实验体会与思考



1. Quartus 的功能强大，但是环境配置和使用过程中难免出现一些小问题，特别是之后和 ModelSim 联合仿真的时候，容易出大问题。平时要养成良好的习惯，例如一个工程一个文件夹，命名要规范等等。

2. **关于驱动：**

在 Quartus 上进行到最后一步，即用 USB 线连接电脑和开发板时，第一次连接可能会出现无法识别 USB 端口的情况。此时是因为 USB Blaster 驱动没有安装或更新。在较新版本的 Quartus 安装包中是有集成这一驱动的，只要在设备管理器安装即可。右击如图红框标注的图表，选择“更新驱动程序”，然后选择手动查找并安装。在 Quartus 安装路径里“D:\Quartus\quartus\drivers\usb-blaster”（quartus 之前的是自己安装的路径），查找并安装驱动完毕后，就能正常识别出硬件了