

兰州大学信息科学与工程学院
计算机组成原理课程设计实验报告

一、实验目的

1. 熟悉 LS-CPU-EXB-002 实验箱和软件平台
2. 掌握利用该实验箱各项功能开发组成原理和体系结构实验的方法
3. 理解并掌握加法器的原理和设计方法
4. 熟悉并利用 Verilog 语言使用 vivado 进行电路设计
5. 为后续实验打下良好的基础
6. 使用+号完成 32 位加法器并尝试自己搭建 32 位加法器

二、实验器材与设备

1. 装有 Xilinx Vivado 的计算机一台
2. LS-CPU-EXB-002 教学系统实验箱一套

三、实验分析与设计

1、实验原理

a. 使用+号完成 32 位加法器

在 Verilog 中直接使用+来完成加法器的实现，使用 assign 语句进行两个变量的相加，并使用{ }来得到输出和进位

```
assign {cout, result} = operand1+operand2+cin;
```

b. 使用 8 个四位全加器完成 32 位加法器

采用四个一位全加器级联成为串行进位加法器，后一个加法器的输入需要等待前一个加法器的输出。

然后将八个四位全加器按照同样的方式进行串行拼接得到 32 位串行加法器。

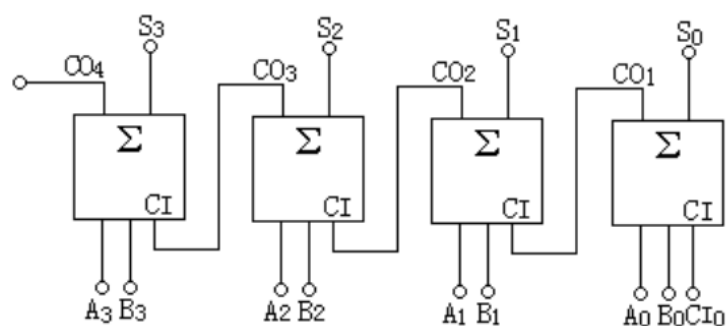


图 2 四位全加器的原理实现

2、端口设计

- a. ADD_1: 第一个加数
- b. ADD_2: 第二个加数
- c. RESULT: 加法的结果

注：通过 FPGA 面板上下方的前两个 01 开关进行加数输入的切换和 Cin 的设置

最左边的开关 01 控制两个加数输入的切换

左边第二个开关 01 控制 Cin 的 01

3、设计框图

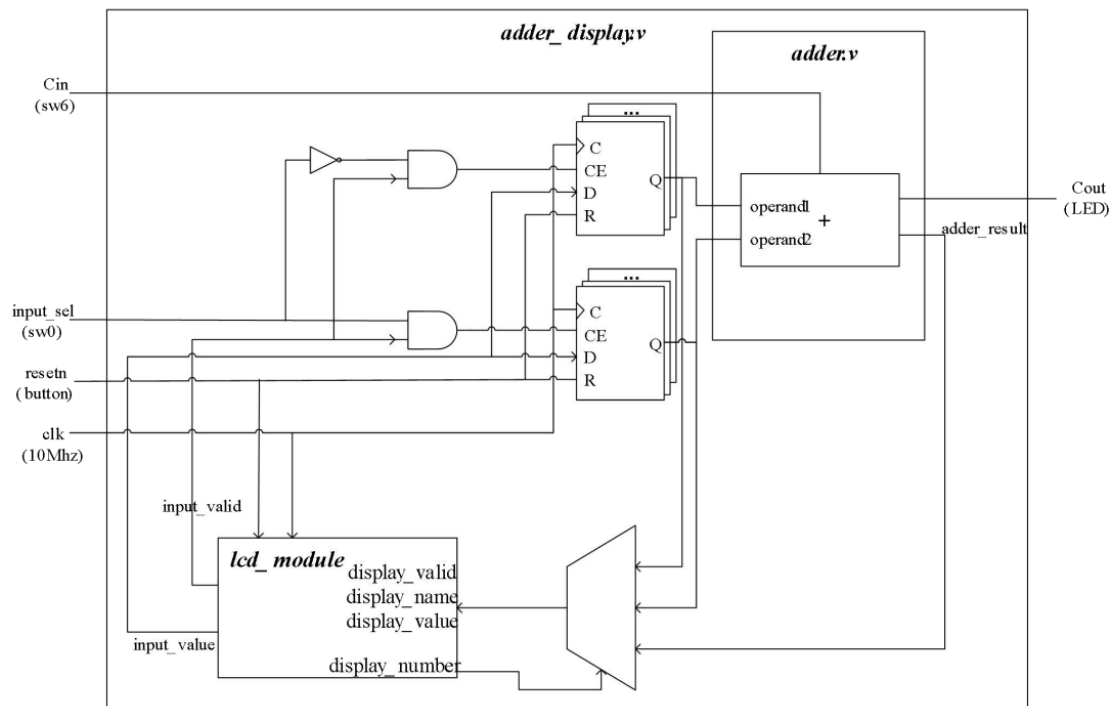


图 1：顶层设计模块

四、实验步骤

注：由于两个加法器的实验步骤一致，这里采用全加器实现的 32 位加法器作为实验对象上板进行验证

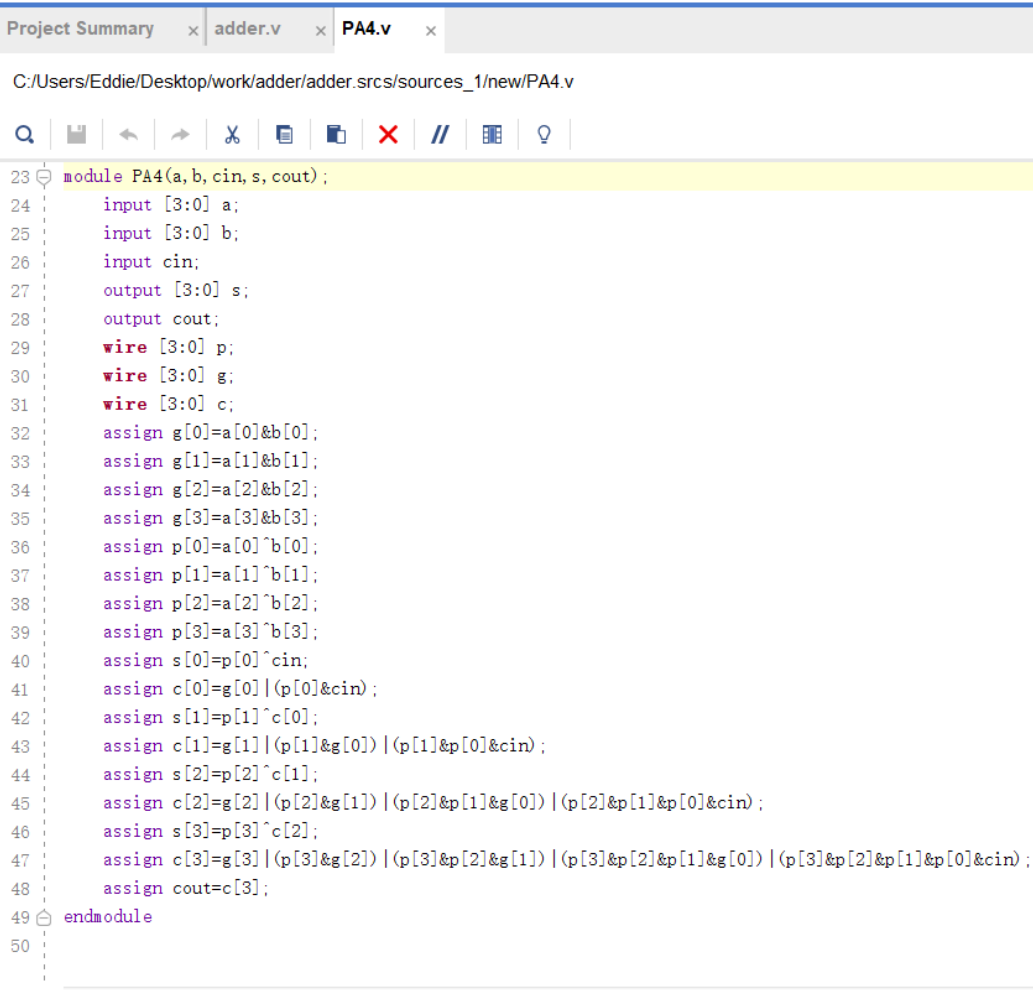
1、实现代码

a. 调用加法 IP——也就是+号得到的加法器代码：

```
//module adder(  
//    input [31:0] operand1,  
//    input [31:0] operand2,  
//    input cin,  
//    output [31:0] result,  
//    output cout  
  
//    );  
//    assign {cout, result} = operand1+operand2+cin;  
//endmodule
```

b. 使用八个 4 位全加器拼接得到的 32 位加法器

```
module adder (a,b,cin,s,cout);
    input [31:0] a;//被加数
    input [31:0] b;//加数
    input cin;//进位输入
    output [31:0] s;//和
    output cout;//进位输出
    wire [6:0] carry;//级联进位
    PA4 u0(a[3:0],b[3:0],cin,s[3:0],carry[0]);
    PA4 u1(a[7:4],b[7:4],carry[0],s[7:4],carry[1]);
    PA4 u2(a[11:8],b[11:8],carry[1],s[11:8],carry[2]);
    PA4 u3(a[15:12],b[15:12],carry[2],s[15:12],carry[3]);
    PA4 u4(a[19:16],b[19:16],carry[3],s[19:16],carry[4]);
    PA4 u5(a[23:20],b[23:20],carry[4],s[23:20],carry[5]);
    PA4 u6(a[27:24],b[27:24],carry[5],s[27:24],carry[6]);
    PA4 u7(a[31:28],b[31:28],carry[6],s[31:28],cout);
endmodule
```

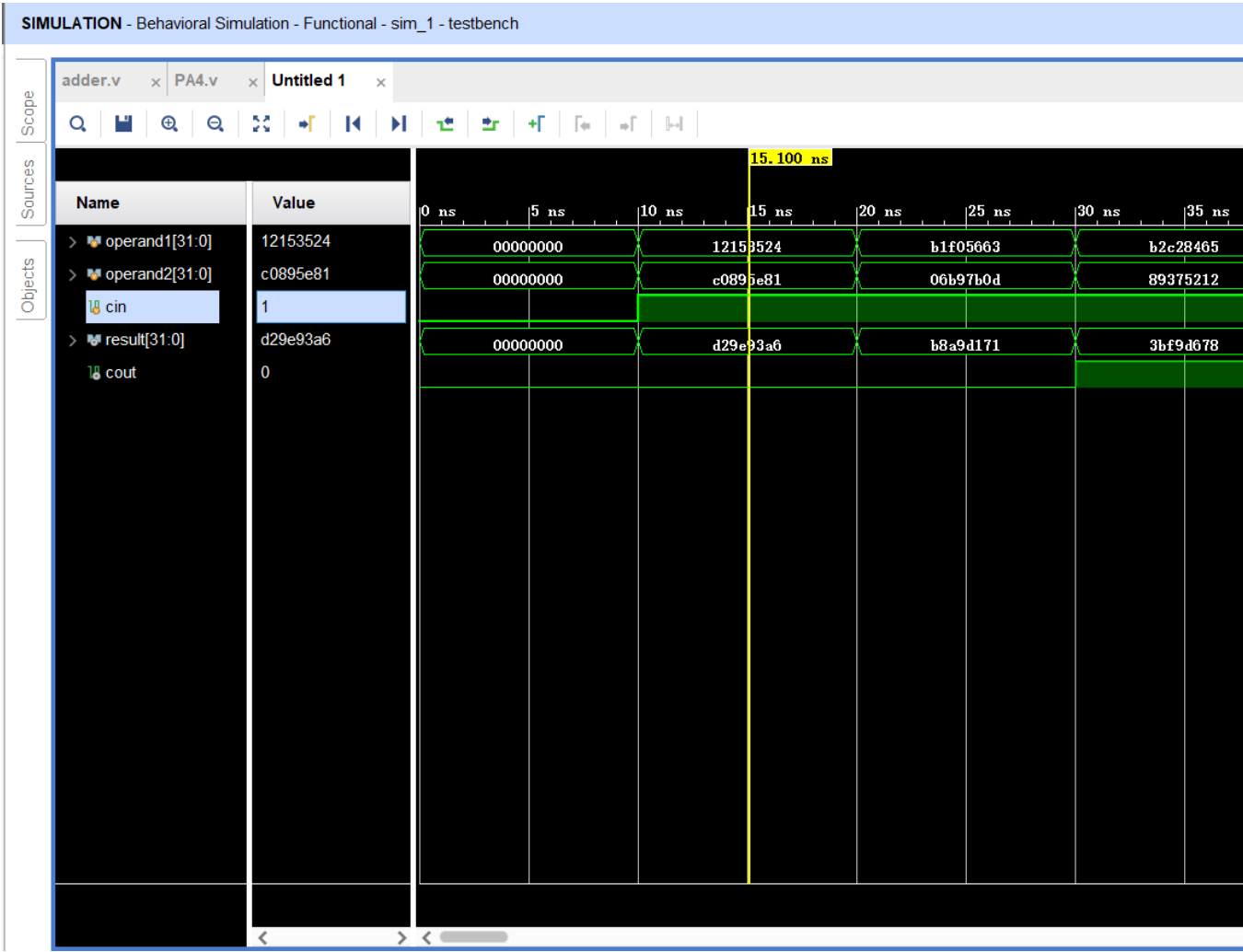


```
23 module PA4(a,b,cin,s,cout);
24     input [3:0] a;
25     input [3:0] b;
26     input cin;
27     output [3:0] s;
28     output cout;
29     wire [3:0] p;
30     wire [3:0] g;
31     wire [3:0] c;
32     assign g[0]=a[0]&b[0];
33     assign g[1]=a[1]&b[1];
34     assign g[2]=a[2]&b[2];
35     assign g[3]=a[3]&b[3];
36     assign p[0]=a[0]^b[0];
37     assign p[1]=a[1]^b[1];
38     assign p[2]=a[2]^b[2];
39     assign p[3]=a[3]^b[3];
40     assign s[0]=p[0]^cin;
41     assign c[0]=g[0]|(p[0]&cin);
42     assign s[1]=p[1]^c[0];
43     assign c[1]=g[1]|(p[1]&g[0])|(p[1]&p[0]&cin);
44     assign s[2]=p[2]^c[1];
45     assign c[2]=g[2]|(p[2]&g[1])|(p[2]&p[1]&g[0])|(p[2]&p[1]&p[0]&cin);
46     assign s[3]=p[3]^c[2];
47     assign c[3]=g[3]|(p[3]&g[2])|(p[3]&p[2]&g[1])|(p[3]&p[2]&p[1]&g[0])|(p[3]&p[2]&p[1]&p[0]&cin);
48     assign cout=c[3];
49 endmodule
50
```

其余模块的代码在书中均有详细的讲解，这里不再赘述

2、仿真与综合

a. Simulation



加数是通过 testbench.v 文件中的 random 进行随机生成

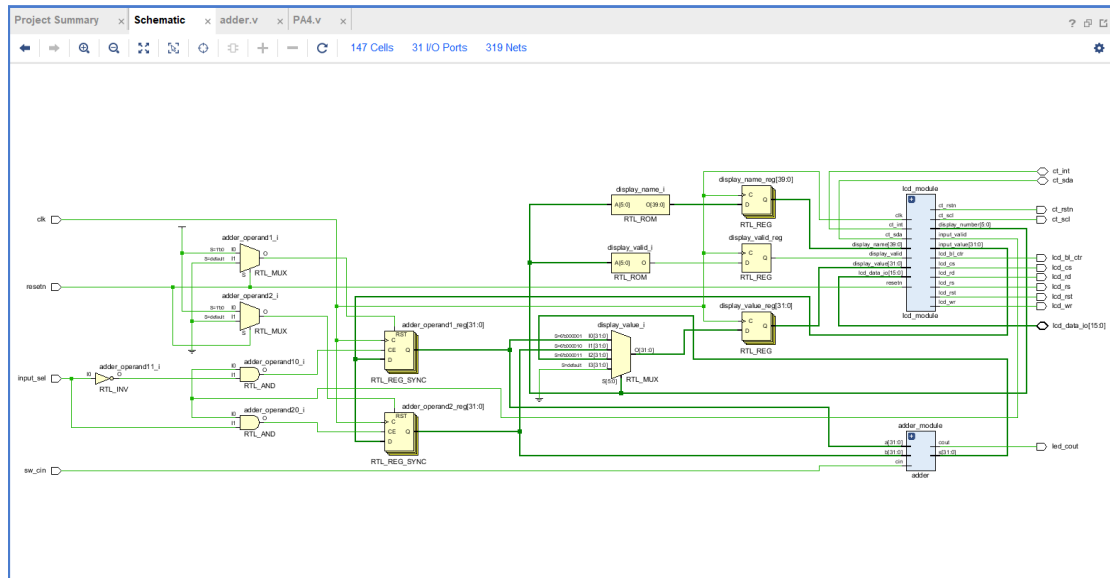
我们拿第二组加法进行手工验证：

$$1215324 + c0895e81 + 1 = d29e3a6$$

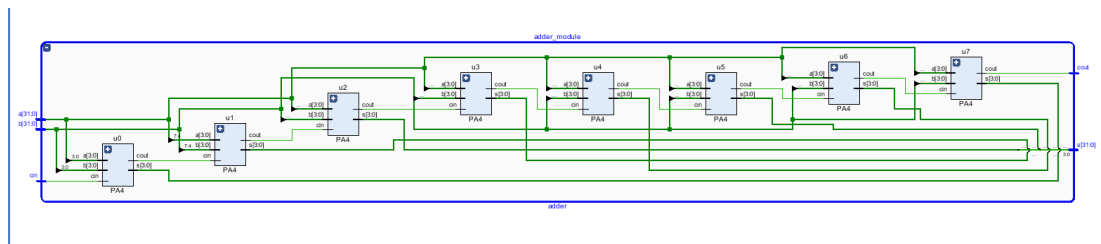
加法的结果无误

b. Elaborated Design

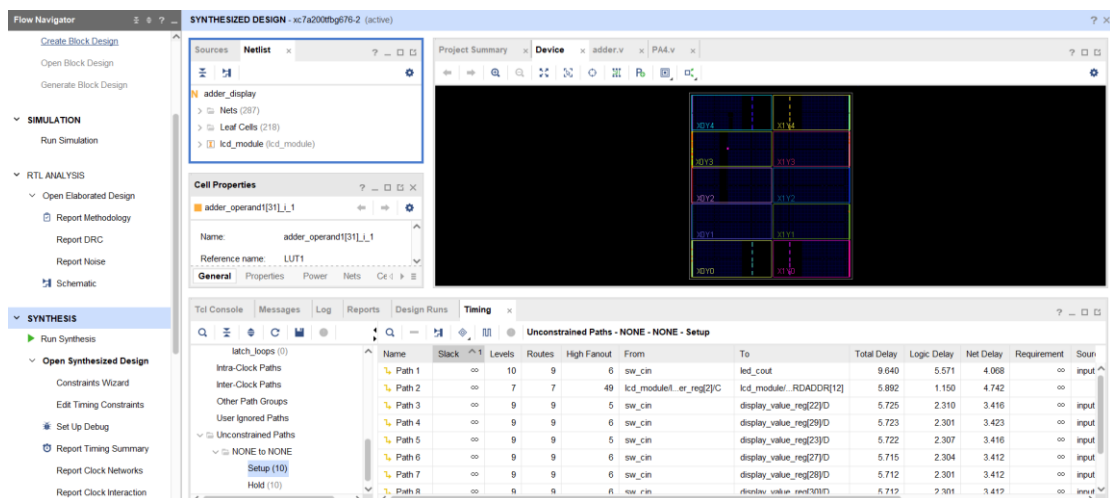
顶层模块：



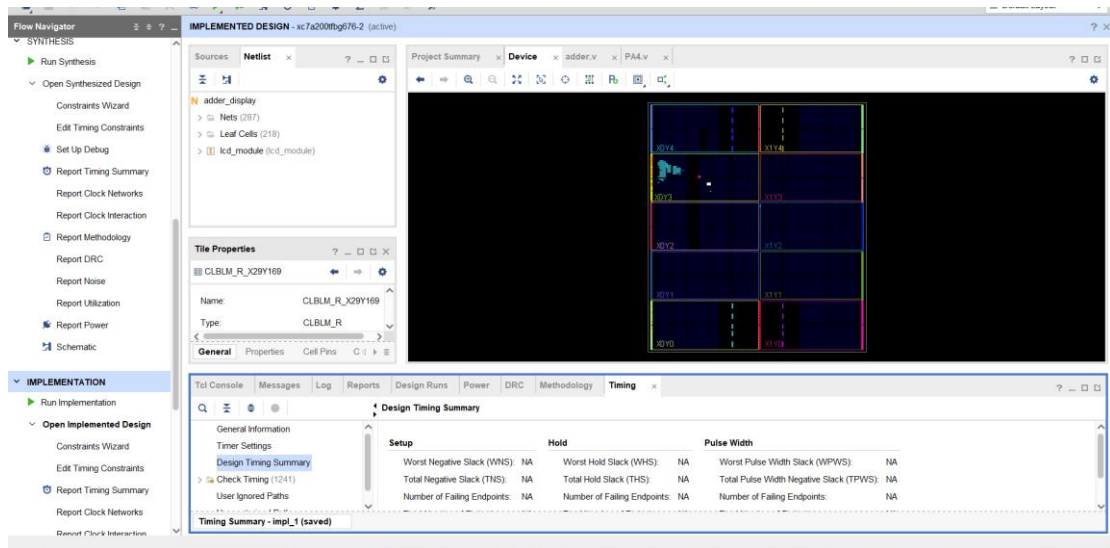
加法器模块：



c. Synthesis 综合



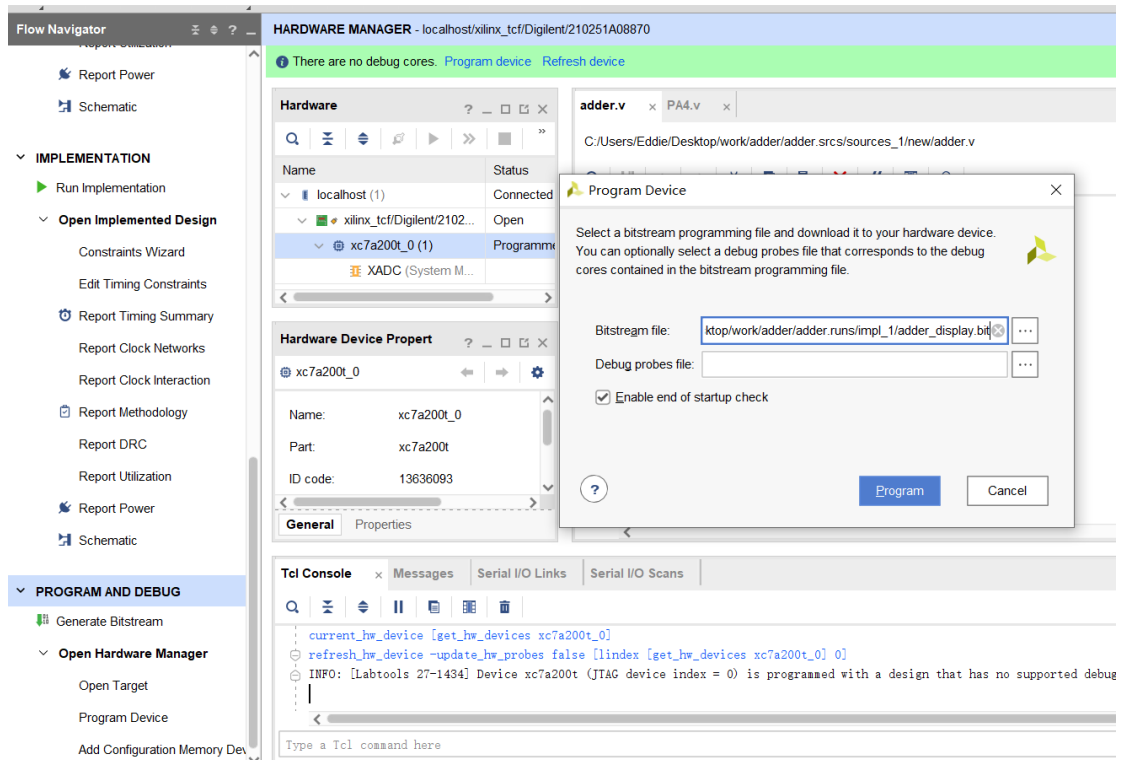
d. Impelementation



3、上板验证及演示结果拍照

点击 generate BitStream 生成二进制文件并将电脑与 FGPA 板进行连接

点击 open Target 与板子连接点击 Program Device 将刚刚生成的二进制文件烧入板子中



验证一：第二个开关是否位进位输入的开关

验证二：加法器的结果是否正确

五、问题回答

基础问题

1. FPGA 开发板上主要都有哪些功能部件，它们都有什么作用？他们分别在实验箱中的什么位置？

答：实验平台的基本结构和软硬件结构图如下：

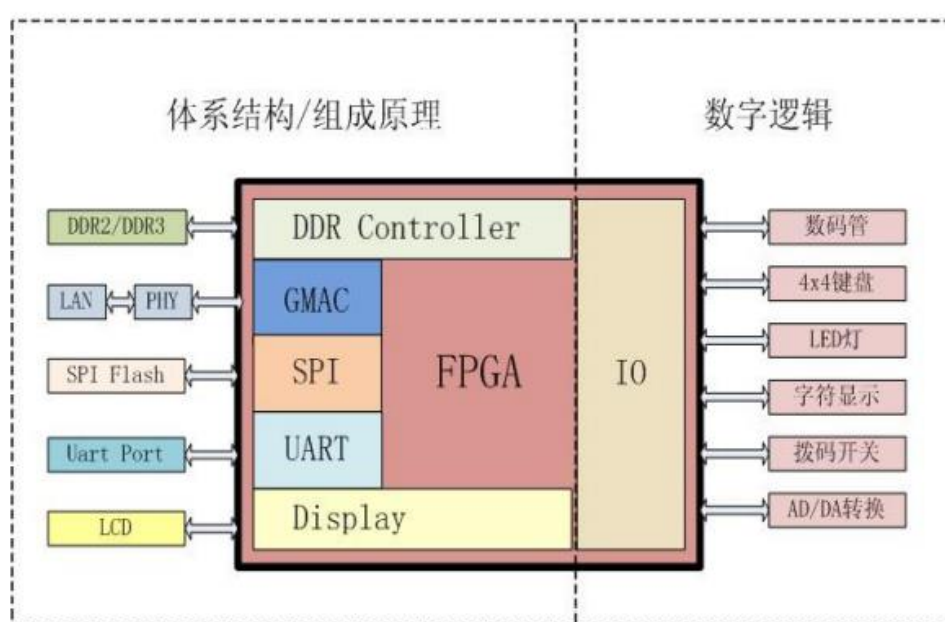


图 1.1 FPGA 实验平台框架图

- a. DDR Controller: DDR(Double Data Rate SDRAM 的缩写)双倍速率同步动态随机存储器控制器，对存储器进行管理包括存储分配等
- b. GMAC:千兆网媒体访问控制。GMAC 在七层 ISO 标准中属于数据链路层，它对逻辑链路和物理链路之间的通道进行控制和协调，可以连接各种不同物理媒介，不同物理媒介有不同的 GMAC 标准，GMAC 的标准由

IEEE802 工作组制定。GMAC 可以分为四个部分：帧发送（Frame Transmission）、帧接收（Frame Reception）、GMAC 控制（GMAC Control）、媒体独立接口管理（GMII）。

- c. SPI：串行外设接口。可以使单片机与各种外围设备以串行的方式进行通信以交换信息
- d. UART：通用异步收发传输器，并行输入转成串行输出
- e. Display：LCD 显示屏。将设置好的变量在显示屏上显示并且可以调用键盘进行数据输入与验证

各个器件的位置和个数如下图所示

接口器件	描述	数量
FPGA	XILINX XC7A200T	1
JTAG	FPGA调试接口	1
EJTAG	FPGA调试接口	1
内存	片内集成硬件内存控制器，DDR3，128MB	1
网口	100M-RJ45	1
SPI	外挂flash 用于作为操作系统启动rom	1
UART	串口	1
LCD	数字RGB接口，外挂LCD显示屏	1
数码管	7段LED数码管	8
按键	4x4矩阵键盘	16
LED		18
拨码开关		8

2. `adder_display.v` 的结构是怎怎样的？怎么调用显示屏进行输入和输出？详细讲解有关程序流程

答：总体上 `adder_display.v` 作为顶层模块，实例化了 `adder.v`，同时调用了 `lcd_module` 进行上板演示。

其结构大致分为五个模块：

- A. 接口相关模块：进行输入输出设置的初始化
- B. 调用的加法模块：将写好的加法器进行实例化并将所得结果与输出对接
- C. 调用触摸屏模块：对触摸屏进行实例化
- D. 触摸屏获取输入模块：从触摸屏获取输入存入预先设置好的变量中
- E. 触摸屏显示模块：共有 44 块显示区域，可显示 44 组 32 位数据，编号为 1-44 每个区域三个属性：

`display_valid`:有效的 flag 0|1

`display_name`:显示的前缀名

`display_value`:显示的值

关于如何调用显示屏进行输入和输出：

- a. 首先设置触摸屏相关接口(不需要更改)
- b. 对触摸屏进行实例化

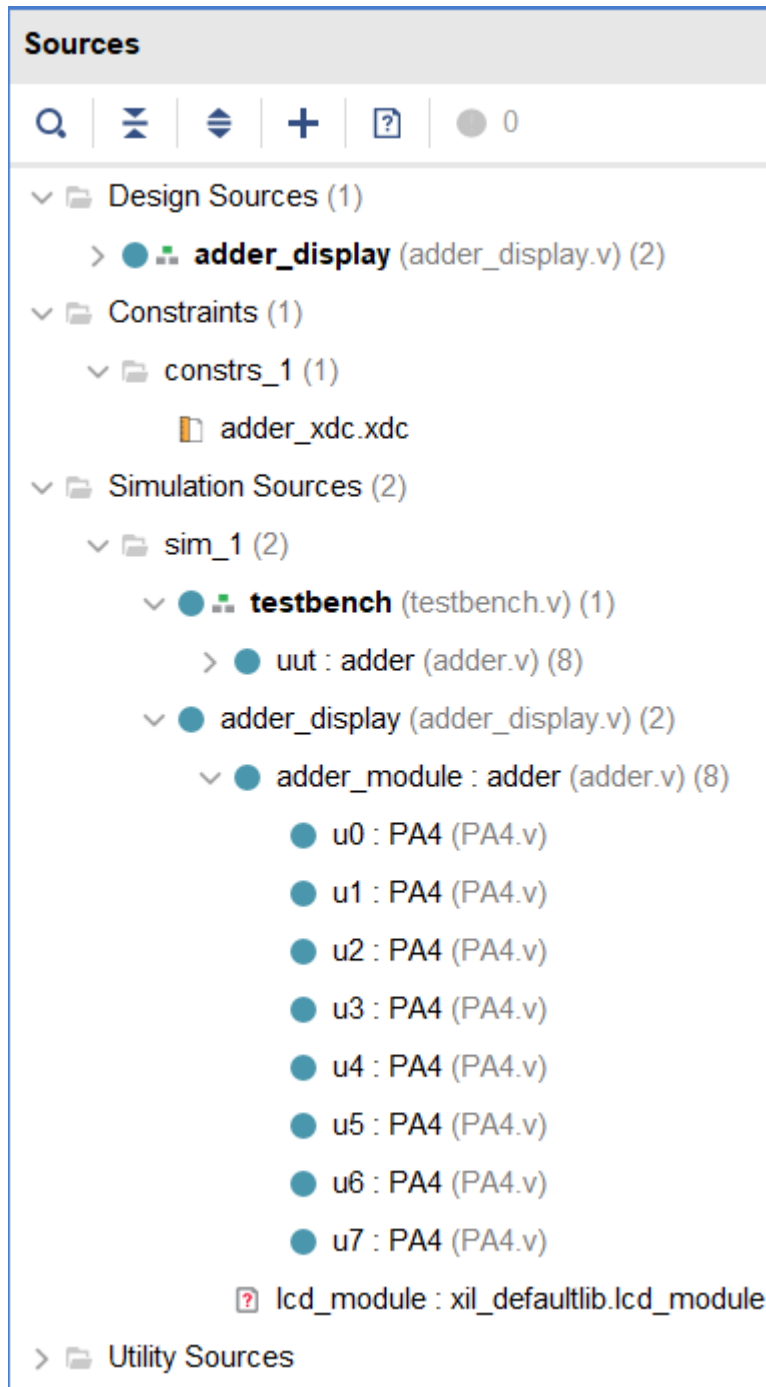
- c. 通过 `input_value` 实现对输入接收
- d. 通过 `display` 的三个属性实现数据的显示屏输出

整个 `adder_display` 工作的流程:

- a. 首先定义输入的变量，比如本次实验只需要两个加数的输入和 `Cin` 的输入，由于每次输入只能进行一次赋值，也就是对一个变量进行赋值，所以设置 `input_sel`: 0 表示给第一个数据赋值，1 表示给第二个数据进行赋值，`sw_cin` 用 01 来表示是否有进位输入
- b. 进行加法器的实例化，绑定显示屏和加法模块的变量用于输入和输出
- c. 对显示屏进行实例化，得到 `input_value`——也就是后面显示屏从键盘输入的值
- d. 将 `input_value` 与加法器的两个加数进行绑定——通过 `input_sel` 进行加数的选择
- e. 最后设置显示屏的显示，通过 `display` 的三个属性跟编号指定显示的内容和区域

3. 整个加法器项目的结构如何，如何理解项目中各文件的作用？

可以从 `vivado` 中看到整个加法器项目的结构：



从外到内模块层层封装调用：

testbench.v：最外层的测试模块，用于在上板之前的测试，测试模块是否能够正常工作

adder_display.v：加法器的外围文件，用于实例化加法器跟触摸屏，使得两者连结，能够从显示屏输入加法器的两个加数

并且在显示屏中得到结果的输出

adder.v: 8 个 4 位全加器串行形成的 32 位加法器文件

PA4.v: 4 位串行全加器的实现文件

lcd_module: lcd 触摸屏模块，为黑盒文件

adder_xdc.xdc: 管脚等约束文件

进阶问题

1. 实现 32 位超前进位加法器，对其原理和实现进行讲解

六、收获与体会

1. 熟悉并掌握了 LS-CPU-EXB-002 实验箱的使用
2. 回顾联系了加法器的原理和 Verilog 语言的使用
3. 熟悉了 vivado 软件从代码编写到上板测试的全过程