编号

**南京航空航天大学**

**机组实验**

|  |  |
| --- | --- |
| 题 目 | 龙芯FPGA板开发试验 |

|  |  |
| --- | --- |
| 学生姓名 | 沈王雄 |
| 学 号 | 161510326 |
| 学 院 | 计算机科学与技术学院 |
| 专 业 | 计算计科学与技术专业 |
| 班 级 | 1615104 |
| 指导教师 | 施惠斌 |

二〇一七年六月

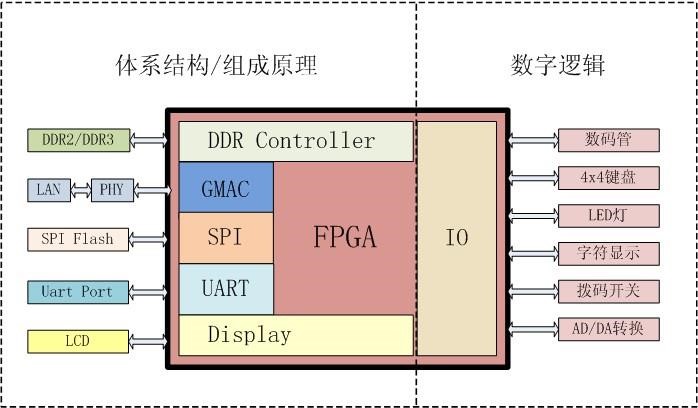
**目录**

1. LS-CPU-EXB-001 CPU 设计与体系结构教学实验系统介绍 .................................................... 5
2. 实验一 数据运算：定点加法 ............................................................................................................ 7
3. 实验二 寄存器堆实现 ...................................................................................................................... 42
4. 实验三 ALU 模块实现 ..................................................................................................................... 45
5. 实验四 存储器实现 ......................................................................................................................... 48
6. 心得体会

**一、LS-CPU-EXB-001 CPU 设计与体系结构教学实验系统介绍**

LS-CPU-EXB-001 CPU 设计与体系结构教学实验系统是根据高等院校计算机专业本科生及研究生相关专业开设的《数字逻辑》、《计算机组成原理》、《计算机体系结构》等计算机基础软硬件课程的实验教学需要，自主研发，具有自主知识产权的集成实验设计、开发与教学平台。该实验系统围绕计算机系统的基础软硬件专业能力培养，可以满足不同层次院校开设上述课程实验的验证型、综合型、创新型实验教学要求。

实验平台的基本结构和软硬件结构图如下所示。

图1.1 FPGA实验平台框架图实验箱里配套资源有

——FPGA 实验板；

——Xilinx 的下载线；

——串口线（含 USB 串口转化器）；

——电源线（含适配器）。

为了使在实验板上演示实验成为可能，LS-CPU-EXB-001 试验箱实现了 LCD 触摸屏

的硬件驱动，使得不需要处理器核就能使用触摸屏的显示和输入功能，且设计了简单清晰的调用接口方便学生使用。

为降低学生在 FPGA 硬件平台上的实际操作难度，方便教师指导，特针对组成原理和

体系结构课程编写了本实验指导手册。指导手册主要分为两部分，cpu 设计部分和 cpu 优化部分：前者主要为组成原理需要，包含实验一到实验七，以及课程设计；后者主要为体系结构需要，包含课程设计和课程设计拓展。

具体实验内容如下：

实验一： 数据运算：定点加法(软硬件平台入门)

实验二：寄存器堆实现

实验三： ALU 模块实现

实验四： 存储器实现

**二、实验一 数据运算：定点加法**

**1** 实验目的

1. 熟悉LS-CPU-EXB-001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。
4. 熟悉并运用verilog语言进行电路设计。
5. 为后续设计cpu的实验打下基础。

**2** 实验设备

1. 装有xilinx ISE的计算机一台。
2. LS-CPU-EXB-001教学系统实验箱一套。

**3** 实验任务

1. 阅读LS-CPU-EXB-001实验箱相关文档，熟悉硬件平台,特别需要掌握利用显示屏观察特定信号的方法。学习软件平台和设计流程。
2. 熟悉计算机中加法器的原理。
3. 自行设计本次实验的方案，画出结构框图，详细标出输入输出端口，本次实验的加法器可以使用全加器自己搭建加法模块，也可以在verilog中直接使用“+”（系统是自动调用库里加法IP，且面积时序更优），依据教师要求选择一种方法实现。
4. 根据设计的实验方案，使用verilog编写相应代码。
5. 对编写的代码进行仿真，得到正确的波形图。
6. 将以上设计作为一个单独的模块，设计一个外围模块去调用该模块，见图

2.1。外围模块中需调用封装好的触摸屏模块，显示两个加数和加法结果，且需要利用触摸功能输入两个加数。

定点加法

模块

外围

模块

来自

FPGA

板子

上

的输入

输出到

FPGA

板

上进行展示

图 2.1 定点加法设计实验的顶层模块大致框图

1. 将编写的代码进行综合布局布线，并下载到实验箱中的FPGA板上进行演示。

**4** 实验要求

1. 做好预习：
   1. 了解软硬件平台；
   2. 掌握定点加法的工作原理；
   3. 确定定点加法的输入输出端口设计；
   4. 在课前画好设计框图或实验原理图；
   5. 如果对FPGA板了解的话，可确定设计中与FPGA板上交互的接口，画出包含外围模块的整体设计框图，即补充完善图2.1。
2. 实验实施：
   1. 确认定点加法的设计框图的正确性；
   2. 编写verilog代码；
   3. 对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材料；
   4. 完成调用定点加法模块的外围模块的设计，并编写代码；
   5. 对代码进行综合布局布线下载到实验箱里FPGA板上，进行上板验证。
3. 实验检查：
   1. 完成上板验证后，让指导老师或助教进行检查，进行现场演示，可对演示结果进行拍照作为实验报告结果一项的材料。
4. 实验报告的撰写：
   1. 实验结束后，需按照规定的格式完成实验报告的撰写。

**5** 参考实现

本部分给出了定点加法实验的参考设计方案，详细阐述了软硬件平台的使用方法，特别是FPGA实验板上的LCD触摸屏的调用模块的使用。

5.1实验步骤

1. 新建工程

启动 ISE 软件，在菜单栏点击“File”->”New Project”，出现新建工程向导，输入工程名称，选择工程的文件位置，然后选择“Next”：

完成工程设置，芯片型号选择 Spartan6，XC6SLX150，FGG484，语言选

Verilog，如下图：

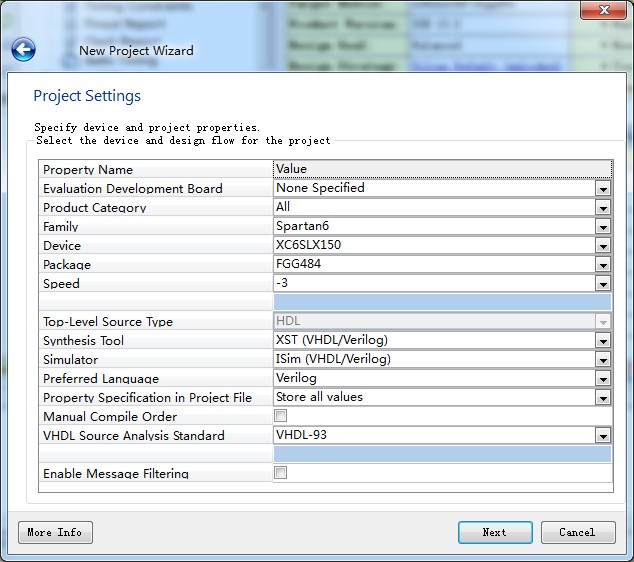


图 2.3 工程设置

图 2.4 工程信息

1. 添加 **verilog** 文件

verilog 代码都是以“.v”为后缀名的文件，可以在其他文件编辑器里写好，再添加到新建的工程中，也可以在工程中新建一个再编辑。

添加已有 verilog 文件的方法如下：在工程管理区任意位置右键，选择“Add Copy of Source”，在弹出窗口中选择加法器实验需要添加的.v 文件：

由于 ISE 工具里自带的编辑器不是很好用，建议使用其他编辑器(比如 Notepad++) 编辑好 verilog 文件再添加到工程中。

代码比较简单，有 2 个 32 位数的输入和 1 个进位输入，产生 1 个 32 位的加法和结果和 1 个向高位的进位。本实验提供的参考设计是直接写“+”号实现加法功能的，这样的写法综合工具会调用内部的模块库的加法器来实现，往往会比自行设计的加法模块更高效和省资源。

本例中，adder.v 为定点加法实验的主体代码，由于实验较简单，故只有这一个.v 文件，对于以后的 CPU 实验，则会有多个.v 文件，形成一定的调用层次。

**3)**添加展示外围模块

在本例中，按照实验要求还需要一个外围模块，该外围模块调用 adder.v，且要调用触摸屏模块的以便在板上实验结果，本例中将该模块命名为 adder\_display.v，添加该模块到工程里，结果如下图：

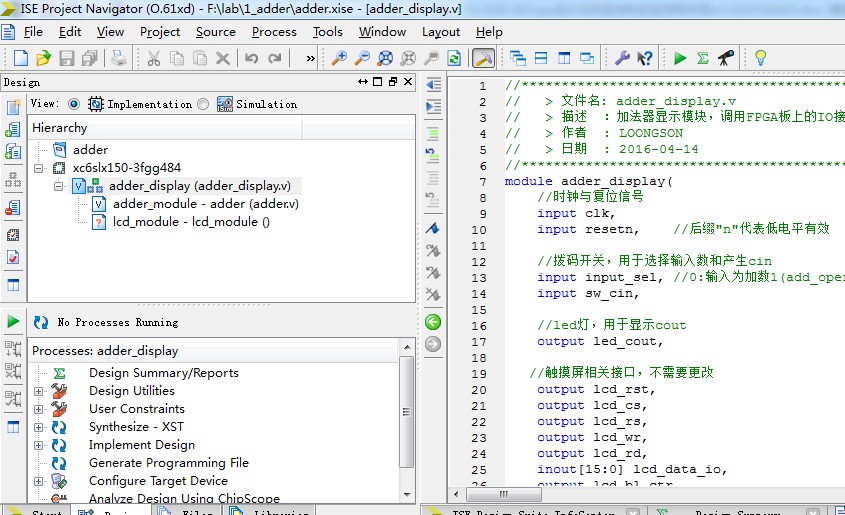


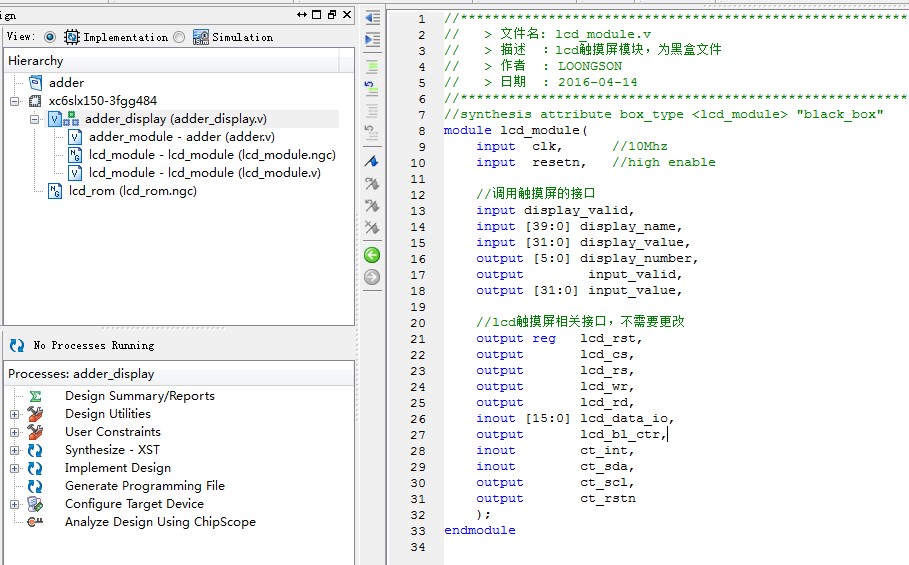
图 2.12 添加外围展示模块

在工程管理区可以看到各模块间的层次关系，本例中顶层模块即为

adder\_display，里面调用了两个子模块：一个为 adder，即定点加法主体代码；一个为 lcd\_module，即 LCD 触摸屏的模块。外围展示模块 adder\_display.v 具体代码解析见 5.2 小节，以后所有实验的外围展示模块都可以仿照 adder\_display.v 编辑。

在图 2.12 中，lcd\_module 前面打了问号，表示该模块还未添加。LS-CPU-EXB-001 配套资源设计时，将 lcd\_module 模块封装为一个黑盒的网表文件，使得学生不会过分关注而导致迷失在 LCD 触摸屏实现中，只需要调用即可。此处，继续在工程管理区任意位置右键，选择“Add Copy of Source”，添加 lcd\_module 的文件，如下图：

可以看到此处有三个文件需要添加，两个为 ngc 文件，即网表文件；一个为.v 文件，是一个黑盒 verilog 文件。添加成功后，结果如下图：

图 2.14 添加 lcd\_module 文件成功

至此，代码实现都已经完成，下面需要对代码功能进行仿真验证功能的正确性。

**4)**功能仿真

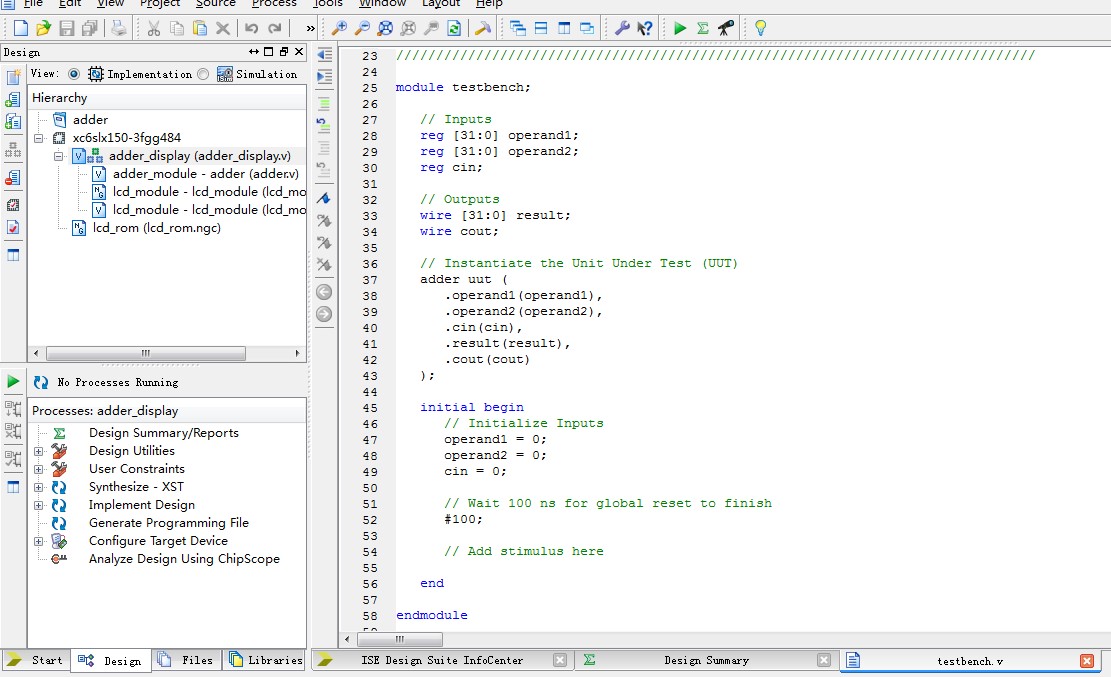
在进行功能仿真时，需要先建立一个 testbench(测试平台)。一个比较完备的 testbench 就是产生输入激励，送入到要测试的功能模块里，然后读出功能模块的执行

结果，与预期的结果进行比较，以此验证功能模块的正确性。但在目前的实验设计上，只需要一个最简单的 testbench——只产生输入激励即可。

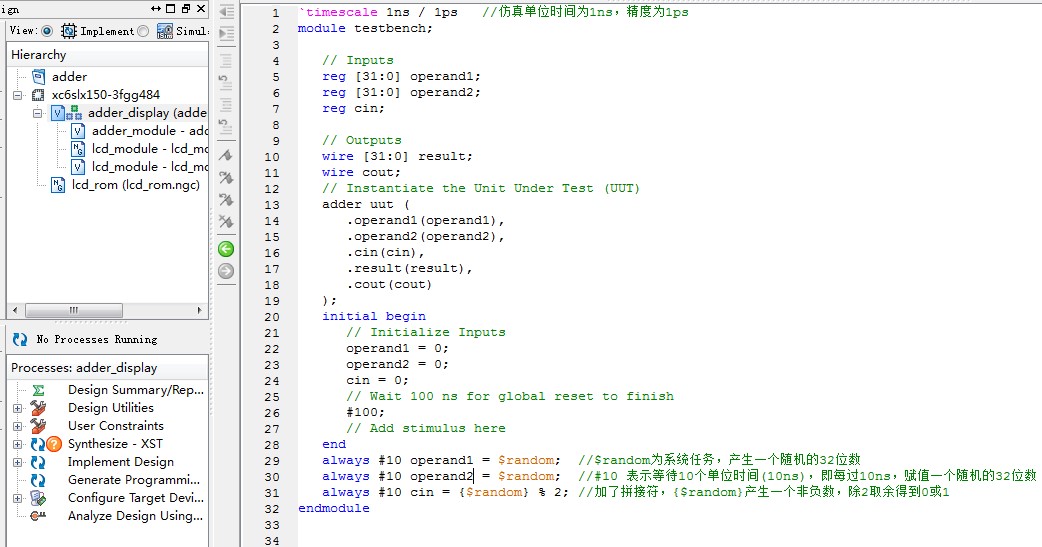
在本例中，需要产生的输入激励就是 2 个加数和 1 个低位进位信号，在该激励输入到加法功能模块中后，会输出加法结果和向高位的进位信号。仿真的过程中会产生波形文件，可以通过观察波形文件确定功能的正确性，在出错的情况下可以定位错误位置。

在工程管理区任意位置右键，选择“New Source”，出现下图界面，源文件类型选择“Verilog Test Fixture”，文件名可随意，此处命名为“testbench.v”：

此处要选择的是测试文件 testbench.v 关联的功能代码文件，即选择需要进行仿真测试的文件。此处需要选择加法功能模块(adder)，而不要选择顶层外围模块(adder\_display)，因为该外围模块含有网表文件 lcd\_module，无法进行仿真，以后的实验在进行仿真时也都要如此选择。

图 2.18 新建测试文件成功

在编辑区(图 2.18 中的右侧区域)可以看到新建的 testbench.v 的内容，ISE 依据管理的功能文件 adder.v 自动生成了大部分测试文件的代码，下面我们只需要添加输入激励产生的机制就可，利用$random 系统任务可以产生随机输入，完成后的 testbench.v 如下图：

图 2.19 编辑测试文件

在工程管理区域(图 2.18 中的左上区域)可以看到并没有 testbench.v 的相关信息，这是因为当前的视图(view)是实现(Implemention)视图，需要选择仿真(Simulation)视图才能看到仿真相关文件，

图 2.20 仿真视图

在语法检查通过后(只要 Check Syntax 不出现红叉即是通过)，右键“Simulate Behavioral Model”选择“Process Properties”，可以更改仿真时间等设置，然后双击

“Simulate Behavior Model”启动 ISE Simulator 进行仿真，可以得到如下界面：

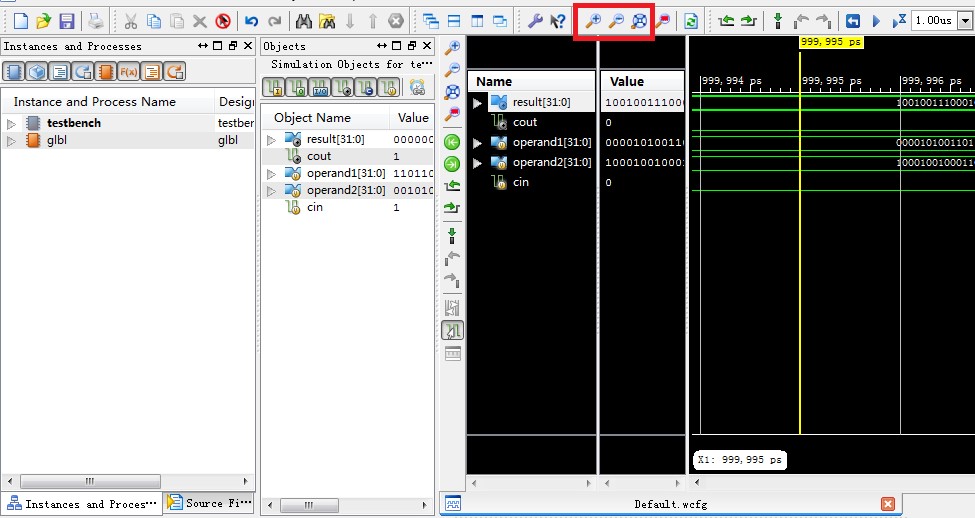
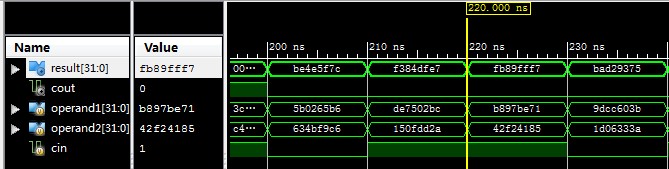


图 2.22 仿真界面

可以通过图 2.22 中红色圈出的放大、缩小、缩放到全屏三个按钮以及鼠标左键点击波形界面(出现图中黄线，缩放会以黄线为中心)观察特定波形区域的信息。

在 ISim 界面左侧的“Instances and Proceses”(图 2.22 的最左列)选择要观察信号所在源文件对象，然后在“Objects”窗口在添加新的信号到波形窗口后，需要点击 Re-launch 才能看到该信号的数据。由于定点加法模块内部所有信号均为输入输出信号，故此处没有新的信号可以添加。但在以后的实验中，会经常需要添加新的信号到波形窗口。另外，当前波形窗口显示的数据均为 2 进制数，对于 32 位的数不便于观察，可以在波形窗口的 Name 列左键选择信号，然后右键会出现多很多可选项，选择“Radix”可以选择要数据显示的进制。

此处选择 16 进制，结果如下图：

图 2.25 数据以 16 进制形式显示

可以检查几组数据，比如：b897be71+42f24186+1=fb89fff7，正确。类似的，通过观察波形可以看到加法功能模块随机测试并没有出错，我们认为功能趋于稳定，可以认为是正确的。

至此，代码编辑和功能仿真都已完成，认为功能基本正确，后续流程就是上板验证了。

**5)**上板验证

所谓上板验证是指将功能代码进行综合和布局布线后下载到 FPGA 板上运行，在板上检查运行的正确性。

因此，需要设定一套在板上检查结果的机制。比如，对于本例中的定点加法，可以设定使用拨码开关作为加数的输入，使用 led 灯作为加法结果的输出，这样就能在 FPGA 实验板上观察加法模块的运行结果了。但遗憾的是，对于 32 位加法，需要 64 个拨码开关和 32 个 led 灯，但显然实验板上提供不了如此之多的 IO 接口，当然也可以通过精巧的设计使用实验板上有限的资源来完成加数输入和结果显示的，但使用起来不够方便。

对于 LS-CPU-EXB-001 试验箱并没有上述的问题，因为可以通过 LCD 触摸屏输入

32 位加数，并能显示加法结果。之前提到的外围展示模块 adder\_display.v 就是调用

LCD 触摸屏来完成上板验证的机制的设计的，具体验证机制的解析见本章的 5.2 小节。

在有了板上验证机制后，需要添加引脚绑定的约束文件。所谓约束文件就是将顶层模块(本例中为 adder\_display)的输入输出端口与 FPGA 板上的 IO 接口引脚绑定，以完成在板上的输入输出。

约束文件后缀名为.ucf，可以通过“Add Copy of Source”添加已有的 ucf 文件，如下图：

也可以通过“New Source”新建一个空白 ucf 文件，如下图，文件类型选择

“Implementation Constraints File”：

adder.ucf 内容如下：

可以看到主要有时钟与复位信号的引脚连接，led 灯和拨码开关的引脚连接，以及 LCD 触摸屏引脚的连接。此处时钟与复位信号和 LCD 触摸屏引脚都是 lcd\_module 需要用到的，可以不用管。

对于 led 灯的连接（led\_cout 绑定到 A8 引脚上），可以看到是用来显示加法向高

位的进位的。

对于拨码开关的连接，“sw\_cin”是用来输入低位的进位的，而“input\_sel”是用来选

择通过触摸屏输入的 32 位数据为加数 1 还是加数 2。

由于以后的实验都需要用到 LCD 触摸屏，故 LCD 触摸屏相关引脚的绑定是固定

不变的，故建议以后实验都通过添加已有 ucf 文件，然后再根据需求修改 led 和拨码开关等引脚的绑定。

在完成约束文件的编辑后，整个工程的层次图如下图：

后续的流程就是综合、布局布线和产生可烧写文件，图 2.28 中红色框出的正是这三步，可以依次双击运行，也可以只双击“Generate Programming File”会自动运行这三步，运行结果如下：

######################################################################

# VCC AUX VOLTAGE

######################################################################

CONFIG VCCAUX=2.5; # Valid values are 2.5 and 3.3

#时钟信号连接

NET "clk" LOC = J1 | IOSTANDARD = "LVTTL";

NET "clk" TNM\_NET = clk;

TIMESPEC TS\_clk = PERIOD "clk" 100 ns HIGH 50%;

NET "clk" CLOCK\_DEDICATED\_ROUTE = FALSE;

PIN "clk\_BUFGP/BUFG.O" CLOCK\_DEDICATED\_ROUTE = FALSE;

#脉冲开关，用于输入作为复位信号,低电平有效

NET "resetn" LOC = U3 | IOSTANDARD = "LVTTL";

#led灯连接，用于输出

NET "led\_cout" LOC = A8 | IOSTANDARD = "LVTTL"; #led3

#拨码开关连接，用于输入

NET "input\_sel" LOC = T2 | IOSTANDARD = "LVTTL"; #sw0

NET "sw\_cin" LOC = M2 | IOSTANDARD = "LVTTL"; #sw6

#触摸屏引脚连接，不需要更改

NET "lcd\_rst" LOC = D14 | IOSTANDARD = "LVTTL";

NET "lcd\_cs" LOC = D12 | IOSTANDARD = "LVTTL";

NET "lcd\_rs" LOC = E12 | IOSTANDARD = "LVTTL";

NET "lcd\_wr" LOC = D13 | IOSTANDARD = "LVTTL";

NET "lcd\_rd" LOC = F13 | IOSTANDARD = "LVTTL";

NET "lcd\_bl\_ctr" LOC = A15 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[0]" LOC = C13 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[1]" LOC = E14 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[2]" LOC = C14 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[3]" LOC = D15 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[4]" LOC = C15 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[5]" LOC = F14 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[6]" LOC = C16 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[7]" LOC = D17 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[8]" LOC = E16 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[9]" LOC = B12 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[10]" LOC = F15 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[11]" LOC = C12 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[12]" LOC = A12 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[13]" LOC = A14 | IOSTANDARD = "LVTTL";

NET "lcd\_data\_io[14]" LOC = A13 | IOSTANDARD = "LVTTL";

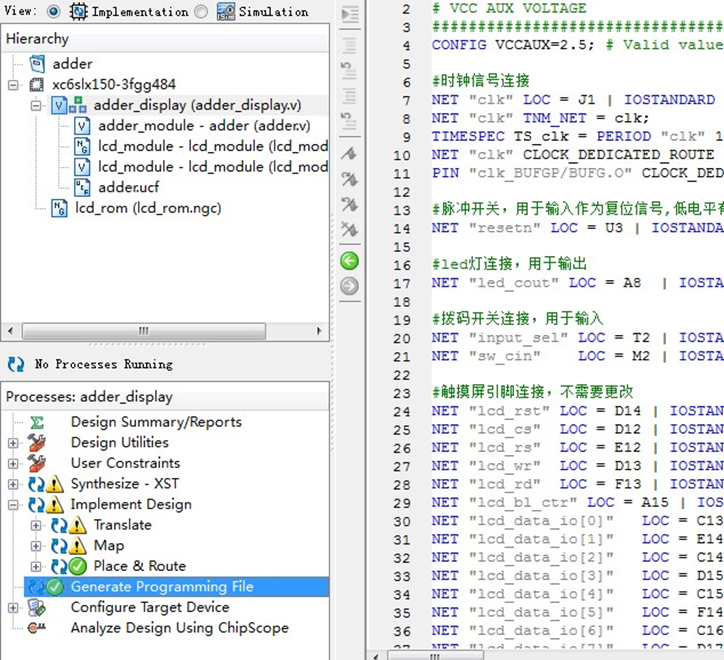
NET "lcd\_data\_io[15]" LOC = B14 | IOSTANDARD = "LVTTL";

NET "ct\_int" LOC = A18 | IOSTANDARD = "LVTTL";

NET "ct\_sda" LOC = A16 | IOSTANDARD = "LVTTL";

NET "ct\_scl" LOC = B18 | IOSTANDARD = "LVTTL";

NET "ct\_rstn"LOC = C17 | IOSTANDARD = "LVTTL";



2.29 综合、布局布线和产生可烧写文件运行结果

出现黄色感叹号说明有警告，但不是错误，一般可以不用管。只要没有出现红叉，就说明可烧写的文件产生成功了，后缀名为.bit，

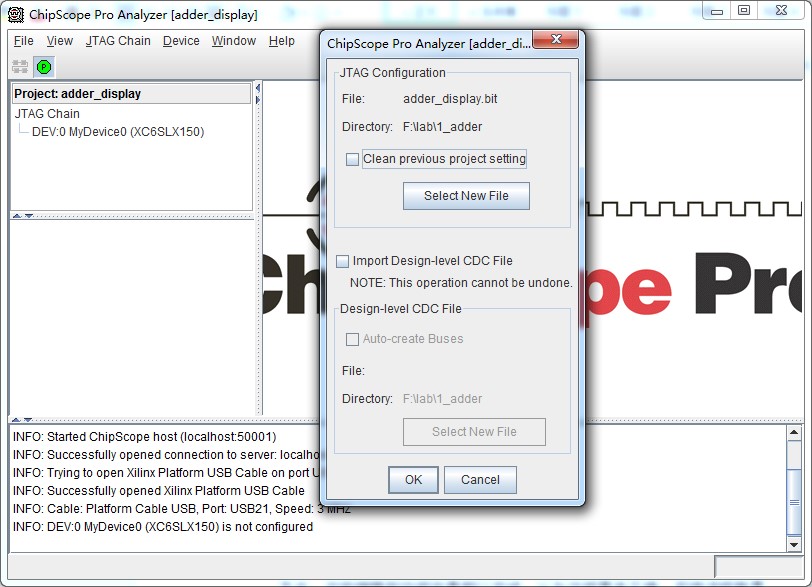
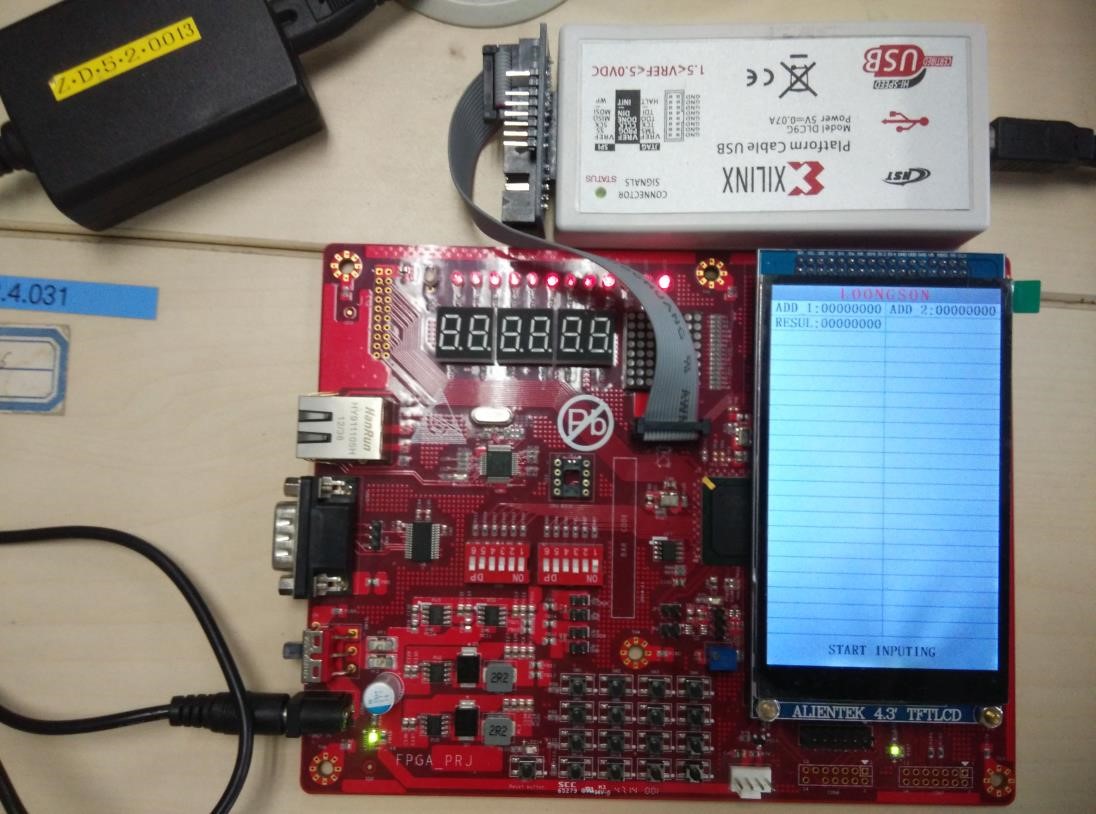


图 2.35 选择要下载烧写的 bit 文件

从图 2.35 中可以看到，软件自动选择了需要下载烧写的 bit 文件，也可以点击

图 2.36 右下角会显示烧写进度，烧写完成后的 FPGA 板显示如下：

图 2.37 加法模块下载成功后的 FPGA 板

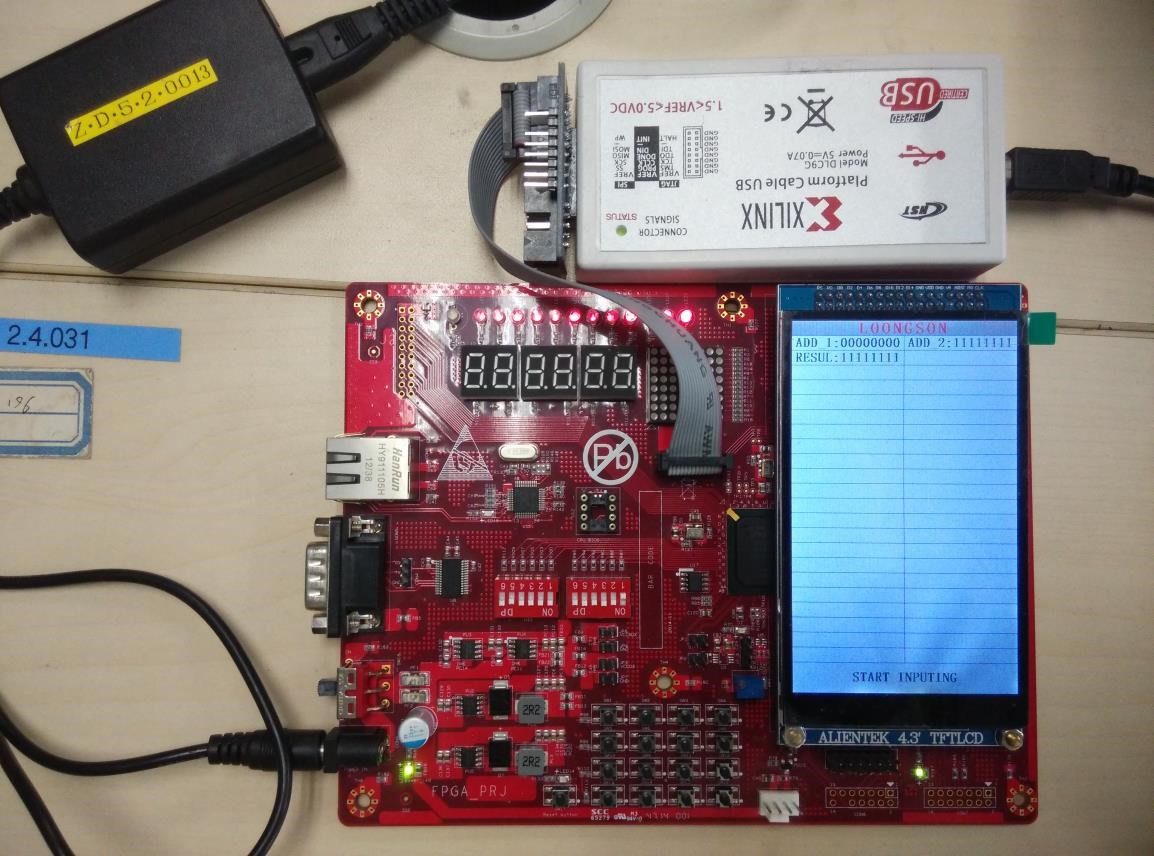
从图 2.37 中可以看到 LCD 触摸屏上分别显示了 2 个加数和加法结果，最右侧的 led 灯为向高位的进位，该 led 灯为共阴极的，即输入为 0 是亮，为 1 是不亮，由于当前进位为 0，故 led 灯为亮。拨码开关最右侧的开关用来选择触摸屏输入的数据为加数

1 还是加数 2，当前该开关置为 1，说明触摸屏输入的 32 位数位加数 2。

在需要使用触摸屏的输入功能时，手指点击屏上最后一列的“START INPUTING” 即可进入输入模式，

从图 2.38 中可看到，触摸屏的输入模式为小键盘，包含 0~F，回退键(BACK)和确认键(OK)。输入的数据为 32 位的 16 进制数，当输出错误的时候，可以按 BACK 键回退一格，当输入完成时按 OK 键完成输入，同时会退出输入模式。当按 OK 时，输入未满 32 位，则会高位补 0，比如：只输入了 123，则最终屏调用模块输入时，输入数据为 0x00000123。

在本例中，给加数 2 输入 0x11111111 后，可看到 FPGA 板上结果如下图：

图 2.39 加数 2 输入 0x11111111

可看到加法结果页变为 0x11111111，正确。可以将拨码开关最右侧的置为 0，此时表示输入数据为加数 1。因此，通过触摸屏可以给加数 1 和加数 2 输入任意 32 位数，并能实时看到加法结果，完成上板验证。

5.2LCD触摸屏调用方法

正如前面所述，adder\_display.v 作为顶层模块，里面实例化了 adder.v，同时也调用了 lcd\_module 用来上板演示。此小节着重讲解 adder\_display.v 里调用 LCD 触摸屏的方法。

**1)LCD** 触摸屏调用接口

先来看 lcd\_module.v 的代码：

module lcd\_module(

input clk, //连接10Mhz的时钟

input resetn, //低使能

input display\_valid, //调用触摸屏的接口

input [39:0] display\_name,input [31:0] display\_value,

output [5:0] display\_number, output input\_valid,

output [31:0] input\_value,

output reg lcd\_rst, //lcd触摸屏相关接口，不需要更改

output lcd\_cs, output lcd\_rs,

output lcd\_wr, output lcd\_rd,

inout [15:0] lcd\_data\_io, output lcd\_bl\_ctr,

inout ct\_int,inout ct\_sda,

output ct\_scl,output ct\_rstn);

endmodule

可以看到该模块为黑盒文件，其中时钟和复位信号以及 LCD 触摸屏引脚接口不需

要关注，下面介绍调用触摸屏的 6 个接口。

2) 调用 LCD 触摸屏显示：

input display\_valid,

input [39:0] display\_name,

input [31:0] display\_value,

output [ 5:0] display\_number,

display 的 4 个接口用于在屏上显示数据。从图 2.37 FPGA 板的实物图中可以看

到，LCD 屏用于显示的区域块共有 2 列，22 行，故共可显示 44 组数据。

display\_number 就是输出到外部说明当前需要显示的区域块为第几块，有效编号

从 1~44，指示 44 块显示区域块。每块显示区域块可显示 14 个字符，其中头五个字符为块名，指示当前块显示的数据的意义，由 display\_name 输入指定。display\_name 输入的为要显示字符的 ASCII码，5 个 ASCII 码共 40 位。块名可显示字符为大写的26 个字母，下划线”\_”，0~9 数字以及空格。每块显示区域块的第 6 个字符为冒号，用于区分块名和块数据段。后 8 个字符显示该块的数值，显示的为 32 位 16 进制数，故占用 8 个字符。该段由 display\_value 输入指定，display\_value 输入的为 32 位 2 进制数，lcd\_module 内部会自动转换为 8 个字符显示。最后还有 display\_valid 输入，用于指示是否需要在当前显示区域块(由 display\_number)显示数据，为 1 有效。

在本例中，顶层模块 adder\_display.v 调用 LCD 触摸屏显示功能的代码如下：

/-----{从触摸屏获取输入}begin,根据实际需要输入的数修改此小节，//建议对每一个数的输入，编写单独一个always块

always @(posedge clk)begin

if (!resetn begin

adder\_operand1 <= 32'd0;end

else if (input\_valid && !input\_sel0 && !input\_sel1 begin

adder\_operand1 <= input\_value; end

end

always @(posedge clk) //当input\_sel为1时，表示输入数为加数2，即operand2

begin

if (!resetn) beginadder\_operand2 <= 32'd0; end

else if (input\_valid && !input\_sel0 && input\_sel1)begin

adder\_operand2 <= input\_value;end

end

always @(posedge clk)begin

if (!resetn)begin

adder\_operand3 <= 32'd0;

end

else if (input\_valid && input\_sel0 && !input\_sel1)

begin

adder\_operand3 <= input\_value;

end

end

1. 调用 **LCD** 触摸屏输入

//-----{输出到触摸屏显示}begin

//根据需要显示的数修改此小节，

//触摸屏上共有44块显示区域，可显示44组32位数据

//44块显示区域从1开始编号，编号为1~44，

always @(posedge clk)

begin

case(display\_number)

6'd1 :

begin

display\_valid <= 1'b1;

display\_name <= "ADD\_1";

display\_value <= adder\_operand1;

end

6'd2 :

begin

display\_valid <= 1'b1;

display\_name <= "ADD\_2";

display\_value <= adder\_operand2;

end

6'd3 :

begin

display\_valid <= 1'b1;

display\_name <= "RESUL";

display\_value <= adder\_result;

end

default :

begin

display\_valid <= 1'b0;

display\_name <= 40'd0;

display\_value <= 32'd0;

end

endcase

end

input 的两个接口用于使用触摸屏的输入功能。从图 2.37 可以看到，当需要使用输

入功能时，触摸屏底部的“START INPUTING”栏即可进入输入模式，见图 2.38。点击屏小键盘上的 OK 键完成输入，会退出输入模式，同时 lcd\_module 会拉高 input\_valid 信号一拍，表示有数据要输出，而输出数据 input\_value 会依据之前的输入确定，当输入不足 32 位时，会自动高位补 0，比如：只输入了 123，就按 OK 键，则最终 input\_valid 的值为 0x00000123。当输入有误时，可以按 BACK 键回退一格。在本例中，顶层模块 adder\_display.v 调用 LCD 触摸屏输入功能的代码.可以看到加法模块需要使用触摸屏输入 2 个加数，故需要一个拨码开关指示输入

的数据为加数 1 还是加数 2。故理论上，触摸屏加上外部的选择信号，可以给任意信号输入 32 位的数，比如可以输入一条 32 位的指令，可以输入内存的 32 位地址等等，这些在后续的 CPU 实验中可能会经常用到。当需要使用触摸屏输入多个数据时.

### **6** 定点加法实验拓展

根据之前所述，可以画出定点加法实验的顶层模块框图，如下图：

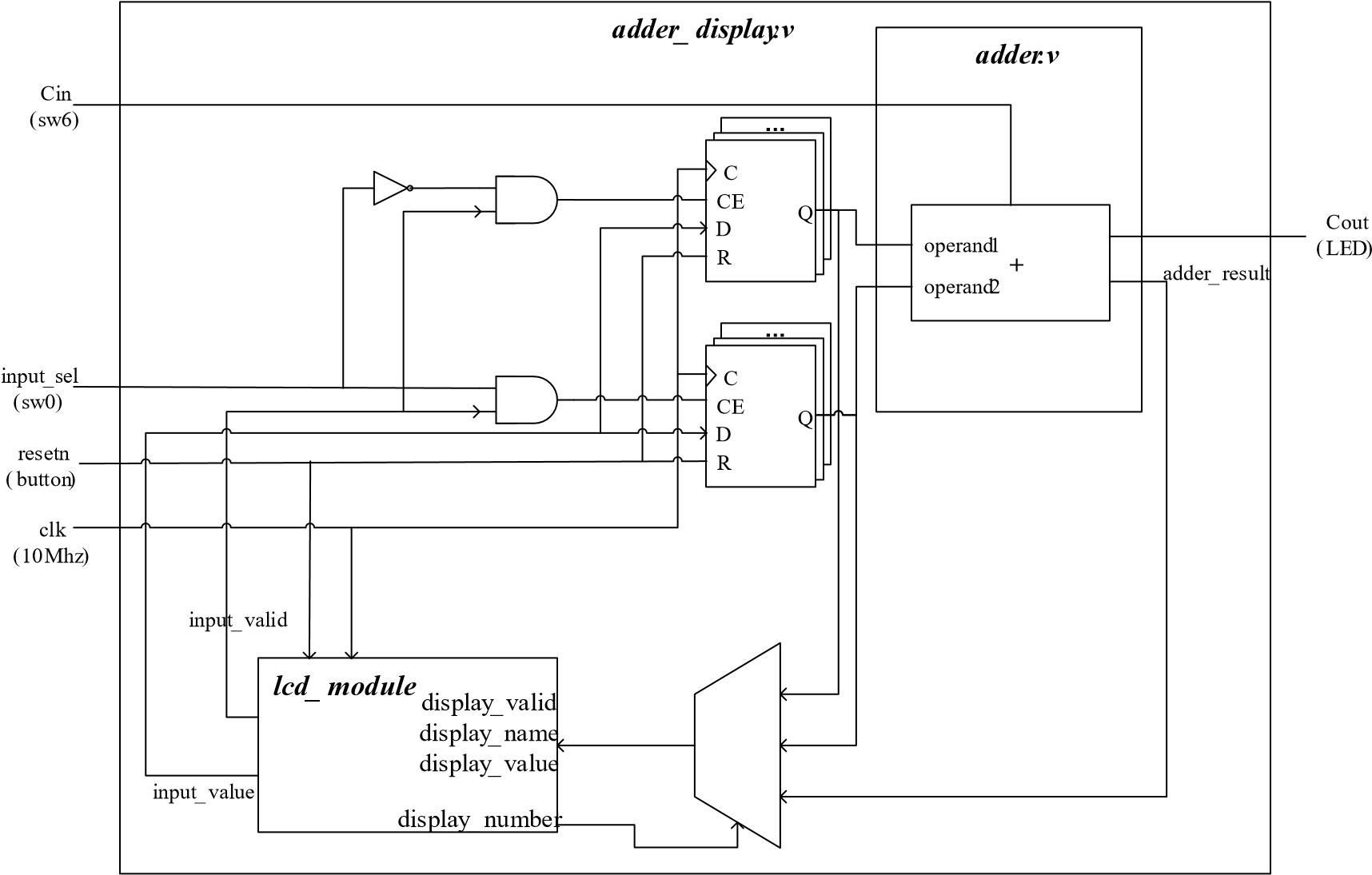


图 2.40 定点加法参考设计的顶层模块框图

从图中可以看到，外围模块 adder\_display 内部调用了 adder 和 lcd\_module 模块。在外部除了时钟和复位信号外，还有 cin 和 input\_sel 通过拨码开关输入，以及 cout 输出到 led 灯上。

从实验二开始，针对每个实验需要画出类似图 2.40 的顶层模块。并且对内部功能模块(此处为 adder.v)需要画出原理图或实现框图，此处由于 adder.v 就用“+”实现，比较简单，故未给出加法模块的原理图。

**三、实验二 寄存器堆实现**

**1** 实验目的

1. 熟悉并掌握MIPS计算机中寄存器堆的原理和设计方法。
2. 初步了解MIPS指令结构和源操作数/目的操作数的概念。
3. 熟悉并运用verilog语言进行电路设计。
4. 为后续设计cpu的实验打下基础。

**2** 实验设备

1. 装有xilinx ISE的计算机一台。
2. LS-CPU-EXB-001教学系统实验箱一套。

**3** 实验任务

1. 学习MIPS计算机中寄存器堆的设计及原理，如：有多少个寄存器，有无特殊设置的寄存器，mips指令如何去索引寄存器的等。
2. 自行设计本次实验的方案，画出结构框图，详细标出输入输出端口，本次实验建议设计为异步读同步写的寄存器堆，即读寄存器不需要时钟控制，但写寄存器需时钟控制。
3. 本次实验建议寄存器堆设计为1个写端口和2个读端口，后续CPU实验用到的寄存器堆需要1个写端口和2个读端口。
4. 根据设计的实验方案，使用verilog编写相应代码。
5. 对编写的代码进行仿真，得到正确的波形图。
6. 将以上设计作为一个单独的模块，设计一个外围模块去调用该模块，见图

4.1。外围模块中需调用封装好的LCD触摸屏模块，显示寄存器堆的读写端口地址和数据，最好能扫描出所有寄存器的值显示在LCD触摸屏上，并且需要利用触摸功能输入寄存器堆的读写地址和写数据。

寄存器

堆

模块

外围

模块

来自

FPGA

板子

上

的输入

输出到

FPGA

板

上进行展示

图 4. 1 寄存器堆设计实验的顶层模块大致框图

1. 将编写的代码进行综合布局布线，并下载到实验箱中的FPGA板子上进行演示。

**4** 实验要求

1. 做好预习：
   1. 掌握寄存器堆的工作原理；
   2. 确定寄存器堆的输入输出端口设计；
   3. 在课前画好寄存器堆的设计框图或实验原理图；
   4. 如果对FPGA板了解的话，可确定设计中与FPGA板上交互的接口，画出包含外围模块的整体设计框图，即补充完善图4.1。
2. 实验实施：
   1. 确认寄存器堆的设计框图的正确性；
   2. 编写verilog代码；
   3. 对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材料；
   4. 完成调用寄存器堆模块的外围模块的设计，并编写代码；
   5. 对代码进行综合布局布线下载到实验箱里FPGA板上，进行上板验证。
3. 实验检查：
   1. 完成上板验证后，让指导老师或助教进行检查，进行现场演示，按照检查人员的要求，对特定寄存器读写，可对演示结果进行拍照作为实验报告结果一项的材料。
4. 实验报告的撰写：
   1. 实验结束后，需按照规定的格式完成实验报告的撰写。

**5** 调用 **LCD** 触摸屏

在寄存器堆实验中，调用LCD触摸屏的显示和输入功能时，有几点需要注意的地方。

5.1显示功能

在本实验中需要扫描出寄存器堆里的所有寄存器的值显示在LCD触摸屏上，比如我们在LCD屏上1~32号显示块上对应显示0~31号寄存器的显示地址。由于LCD屏在显示时，是根据显示块号“display\_number”来选择要显示的数据的，故可以给寄存

器堆再增加一个调试读端口(此时寄存器堆有3个读端口)，该读端口的地址为

display\_number-1，对应0~31,读出的数据即为要显示的模块，代码可以如下编写：

test\_addr为寄存器堆新增的调试读端口，而test\_data为通过该端口读出的数

据可以看到在显示32个寄存器时，其“display\_name”域有特殊处理，其头3个字

符显示“REG”，后两个字符则显示16进制的寄存器号“00~1F”，由于0~9字符的

ASCII码为48~57(8’b0011\_0000~8’b0011\_1001)，故“display\_name”第4位字符(0 或1)赋值{7’b0011\_000,test\_addr[4]}即可。

但对于第5位字符(0~F)的赋值则有稍许麻烦，因为A~F字符的ASCII编码不是随

后的58~63，即与0~9的编码不连续。但我们在设计LCD屏显示字符时，将内部的

ASCII编码58~63也作为A~F字符的编码，即ASCII编码48~63(8’b0011\_0000 ~

8’b0011\_1111)对应字符0~F，，故我们可以直接给第5位字符赋值

{4'b0011,test\_addr[3:0]}。

5.2输入功能

在调用LCD触摸屏的输入功能时，其输入的数据为32位的数据，但在寄存器堆的

实验中，读写地址为5位的，但依然可以使用屏输入读写地址，比如，可以取输入的 32位数据的低5位作为读写地址。

1 assign test\_addr = display\_number-5'd1;

2 //-----{输出到触摸屏显示}begin

3 //根据需要显示的数修改此小节，

4 //触摸屏上共有44块显示区域，可显示44组32位数据

//44块显示区域从1开始编号，编号为1~44，

always @(posedge clk)

begin

if (display\_number >6'd0 && display\_number <6'd33 )

begin //块号1~32显示32个通用寄存器的值

display\_valid <= 1'b1;

display\_name[39:16] <= "REG";

display\_name[15: 8] <= {4'b0011,3'b000,test\_addr[4]}; display\_name[7 : 0] <= {4'b0011,test\_addr[3:0]};

display\_value <= test\_data;

end

else

begin

case(display\_number) 6'd33: //显示读端口1的地址 begin

display\_valid <= 1'b1;

display\_name <= "RADD1";

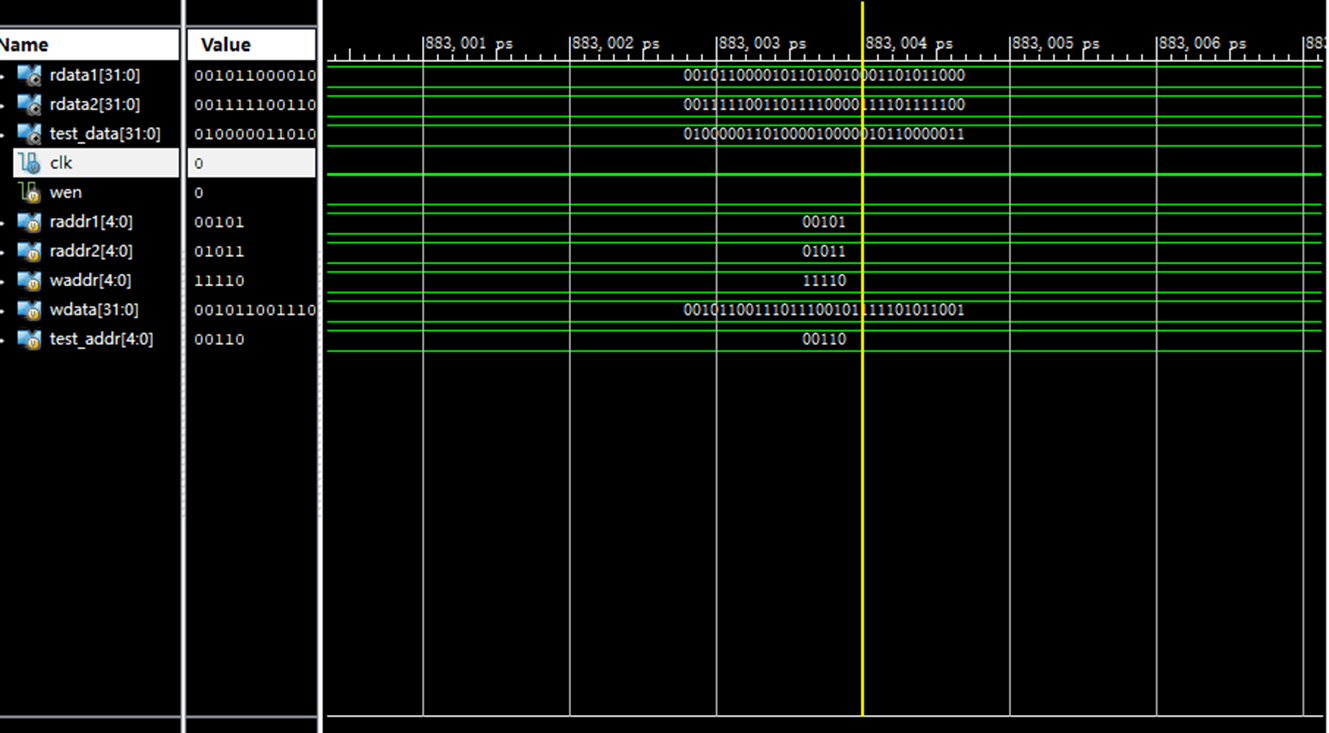
display\_value <= raddr1;

end

……

……

波形仿真：

Fpga板验证：



**四、实验三 ALU 模块实现**

**1** 实验目的

1. 熟悉MIPS指令集中的运算指令，学会对这些指令进行归纳分类。
2. 了解MIPS指令结构。
3. 熟悉并掌握ALU的原理、功能和设计。
4. 进一步加强运用verilog语言进行电路设计的能力。
5. 为后续设计cpu的实验打下基础。

**2** 实验设备

1. 装有xilinx ISE的计算机一台。
2. LS-CPU-EXB-001教学系统实验箱一套。

**3** 实验任务

1. 学习MIPS指令集，熟知指令类型，了解指令功能和编码，归纳基础的ALU运算指令。
2. 归纳确定自己本次实验中准备实现的ALU运算，要求不实现定点乘除指令和浮点运算指令，要求至少实现5种ALU运算，其中要包含加减运算，其中减法在内部要转换为加法，与加法运算共同调用实验一里自己完成的加法模块去做。
3. 自行设计本次实验的方案，画出结构框图，大致结构框图如图5.1。图5.1中的操作码位数和类型请自行设计，可以设计为独热码（一位有效编码）或二进制编码。比如，设计方案中预定实现7种ALU运算，则操作码采用独热码，则需7bit数据，每位单独指示一种运算；若采用二进制编码，则只用3bit数据位即可，但在需

ALU内部先进行解码，才能确定ALU作何种运算。

32位源操作数

运算结果

操作码

32位源操作数

图 5.1 ALU 模块的大致框图

1. 根据设计的实验方案，使用verilog编写相应代码。
2. 对编写的代码进行仿真，得到正确的波形图。

8.将以上设计作为一个单独的模块，设计一个外围模块去调用该模块，见图

5.2。外围模块中需调用封装好的LCD触摸屏模块，显示ALU的两个源操作数、操作码和运算结果，并且需要利用触摸功能输入源操作数。操作码可以考虑用LCD触摸屏输入，也可以用拨码开关输入。

外围

模块

来自

FPGA

板子

上

的输入

输出到

FPGA

板

上进行展示

图 5.2 ALU 设计实验的顶层模块大致框图

6.将编写的代码进行综合布局布线，并下载到试验箱中的FPGA板子上进行演示。

**4** 参考设计

1. 做好预习：
   1. 熟知指令类型，了解指令功能和编码；
   2. 归纳基础的ALU运算指令，确定自己准备实现的ALU运算；
   3. 设计本次实验的方案，列出准备实现的ALU运算和操作码的编码；
   4. 在课前画好实验方案的设计框图，即补充完善图5.1；
   5. 如果对FPGA板了解的话，可确定设计中与FPGA板上交互的接口，画出包含外围模块的整体设计框图，即补充完善图5.2。
2. 实验实施：
   1. 确认ALU模块的设计框图的正确性；
   2. 编写verilog代码；
   3. 对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材料；
   4. 完成调用ALU模块的外围模块的设计，并编写代码；
   5. 对代码进行综合布局布线下载到试验箱里FPGA板上，进行上板验证。
3. 实验检查：

1）完成上板验证后，让指导老师或助教进行检查，进行现场演示。先说明自己实现的ALU运算类型，按照检查人员的要求，对特定源操作数进行特定运算操作，检查运算结果的正确性，可对演示结果进行拍照作为实验报告结果一项的材料。

4.实验报告的撰写：

1）实验结束后，需按照规定的格式完成实验报告的撰写。

**5** 可选 **ALU** 操作

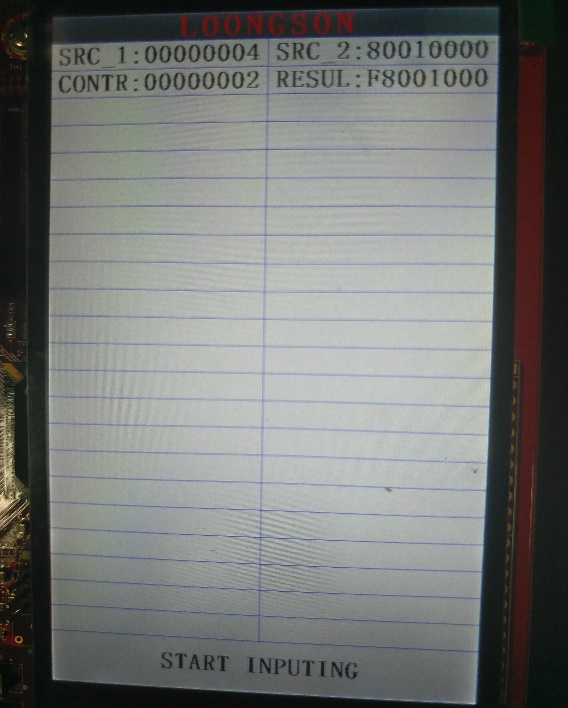
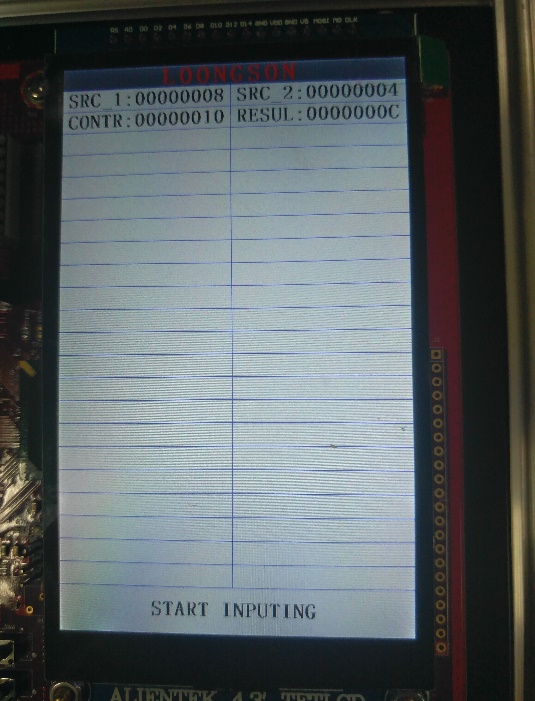
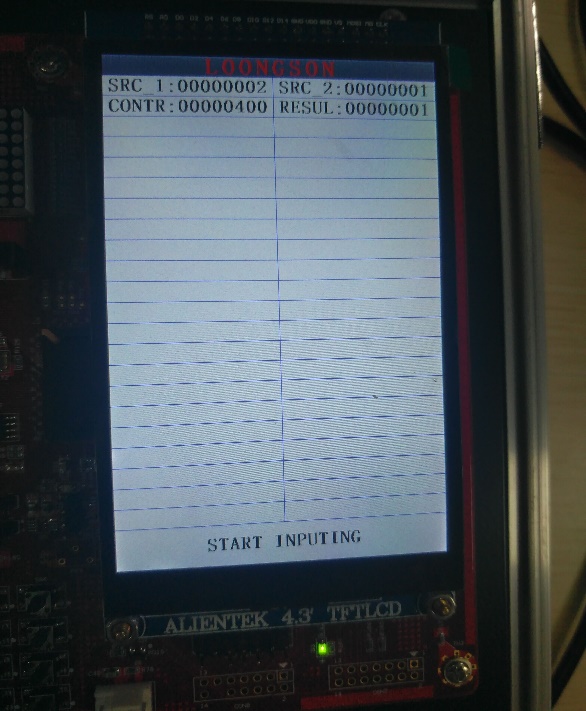
本部分共给出12种ALU操作，可以选择全部或部分实现。

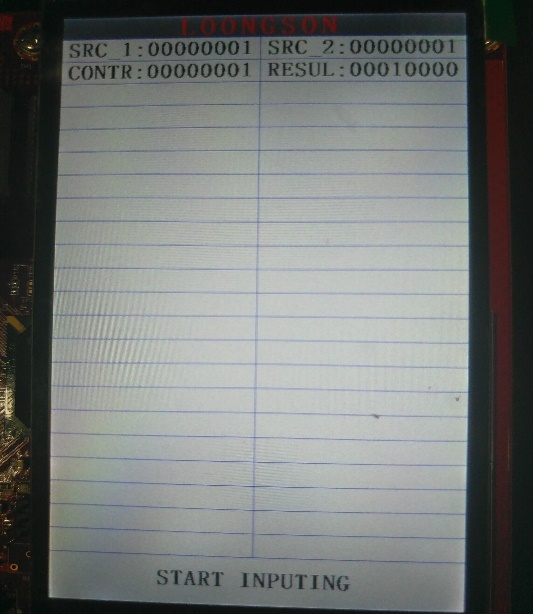
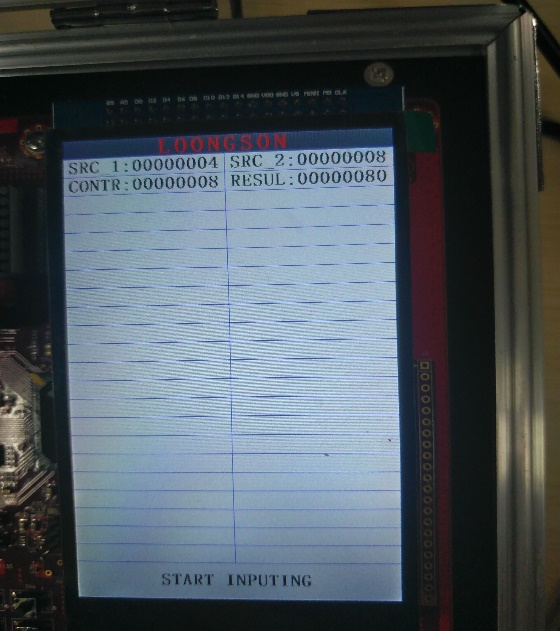
|  |  |  |
| --- | --- | --- |
| 16进制 | 二进制 | Alu控制信号(独热码) |
| 000 | 0000 00000000 | 无 |
| 800 | 100000000000 | 加法 |
| 400 | 0100 00000000 | 减法 |
| 200 | 0010 0000 0000 | 有符号比较（小于置位） |
| 100 | 0001 0000 0000 | 无符号比较（小于置位） |
| 080 | 0000 1000 0000 | 按位与 |
| 040 | 0000 0100 0000 | 按位或非 |
| 020 | 0000 00100000 | 按位或 |
| 010 | 0000 00010000 | 按位异或 |
| 008 | 0000 00001000 | 逻辑左移 |
| 004 | 0000 00000100 | 逻辑右移 |
| 002 | 0000 00000010 | 算术右移 |
| 001 | 0000 0000 0001 | 高位加载 |

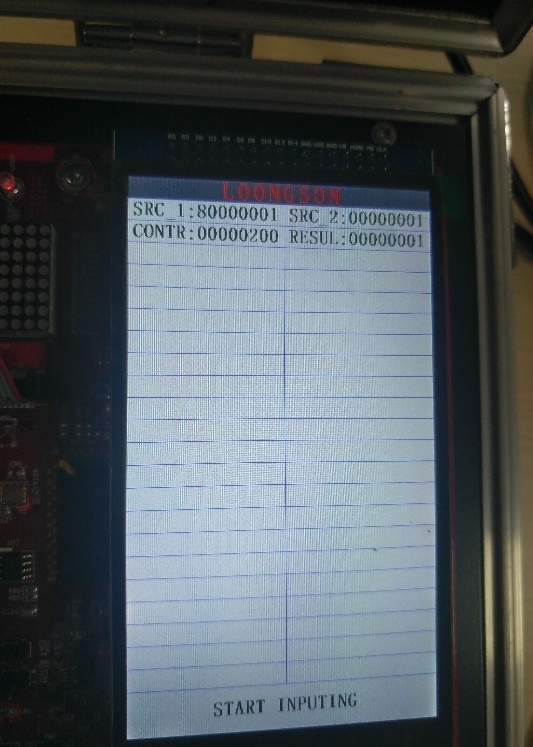
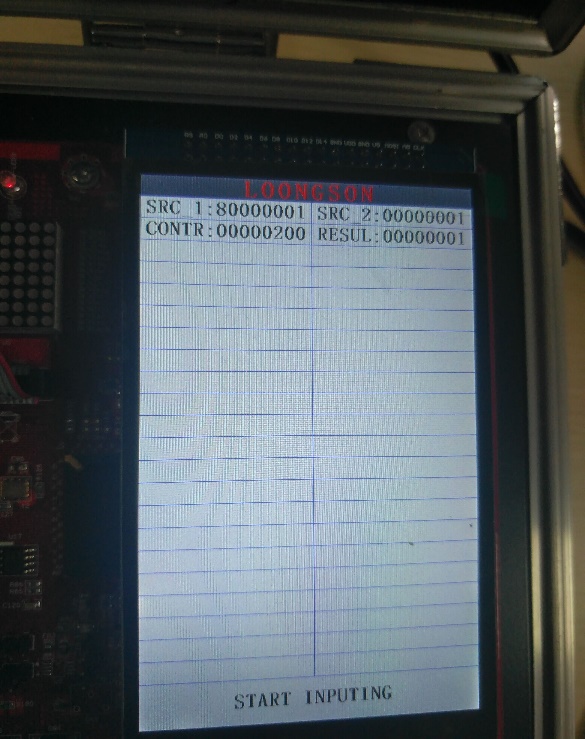
****

alu结果验证：

**4.上板验证**







**三、实验结果分析**

给出特定的操作数SRC\_1,和SRC\_2,控制信号CONTR，输出相应的结果RESUL

**验证1**

|  |  |
| --- | --- |
| SRC\_1：00000002 | SRC\_2：00000001 |
| CONTR:00000400 | RESUL:00000001 |

结果分析：此时两个操作数时用十六进制表示，用32位二进制表示为：SRC\_1：0000 0000 0000 0000 0000 0000 0000 0010

SRC\_2：0000 0000 0000 0000 0000 0000 0000 0001

CONTR（12位表示）:0100 0000 0000

参考ALU控制信号表：此时ALU是在做减法操作

验证：32’d2-32’d1=32’d1

根据FPGA显示可得，答案正确。

**验证2**

|  |  |
| --- | --- |
| SRC\_1：80000001 | SRC\_2：00000001 |
| CONTR:00000200 | RESUL:00000001 |

结果分析：此时两个操作数时用十六进制表示，用32位二进制表示为：SRC\_1：1000 0000 0000 0000 0000 0000 0000 0001

SRC\_2：0000 0000 0000 0000 0000 0000 0000 0001

CONTR（12位表示）:0010 0000 0000

参考ALU控制信号表：此时ALU是在做slt操作(有符号比较，小于置位)

验证：SRC\_1此时为负数，SRC\_2此时为正数，SRC\_1比SRC\_2肯定小

根据FPGA显示可得，答案正确。

**验证3**

|  |  |
| --- | --- |
| SRC\_1：00000008 | SRC\_2：00000001 |
| CONTR: 00000040 | RESUL: FFFFFFF3 |

结果分析：此时两个操作数时用十六进制表示，用32位二进制表示为：SRC\_1：0000 0000 0000 0000 0000 0000 0000 1000

SRC\_2：0000 0000 0000 0000 0000 0000 0000 0001

CONTR（12位表示）:0000 0100 0000

参考ALU控制信号表：此时ALU是在做按位或非操作

验证: 依次进行按位或非，只有第0位为0，第3位为0

根据FPGA显示可得，答案正确。

**验证4**

|  |  |
| --- | --- |
| SRC\_1：00000004 | SRC\_2：00000008 |
| CONTR:00000008 | RESUL:00000080 |

结果分析：此时两个操作数时用十六进制表示，用32位二进制表示为：SRC\_1：0000 0000 0000 0000 0000 0000 0000 0100

SRC\_2：0000 0000 0000 0000 0000 0000 0000 1000

RESUL: 0000 0000 0000 0000 0000 0000 1000 0000

CONTR（12位表示）:0000 0000 1000

参考ALU控制信号表：此时ALU是在做逻辑左移操作

验证：逻辑左移,可见得出结果，

根据FPGA显示可得，答案正确。

**验证5**

|  |  |
| --- | --- |
| SRC\_1：00000002 | SRC\_2：00000001 |
| CONTR:00000800 | RESUL:00000003 |

结果分析：此时两个操作数时用十六进制表示，用32位二进制表示为：SRC\_1：0000 0000 0000 0000 0000 0000 0000 0010

SRC\_2：0000 0000 0000 0000 0000 0000 0000 0001

RESUL: 0000 0000 0000 0000 0000 0000 0000 0011

CONTR（12位表示）:1000 0000 0000

参考ALU控制信号表：此时ALU是在做加法操作

验证：加法,可见得出结果，

根据FPGA显示可得，答案正确。

**五、实验四 存储器实现**

**1** 实验目的

1. 了解只读存储器ROM和随机存取存储器RAM的原理。
2. 理解ROM读取数据及RAM读取、写入数据的过程。
3. 理解计算机中存储器地址编址和数据索引方法。
4. 理解同步RAM和异步RAM的区别。
5. 掌握调用xilinx库IP实例化RAM的设计方法。
6. 熟悉并运用verilog语言进行电路设计。
7. 为后续设计cpu的实验打下基础。

**2** 实验设备

1. 装有xilinx ISE的计算机一台。
2. LS-CPU-EXB-001教学系统实验箱一套。

**3** 实验任务

1. 学习存储器的设计及原理，如：ROM读地址索引读取数据过程及时序，RAM读写时序，同步和异步的区别等。
2. 学习计算机中内存地址编址和数据索引方法。
3. 自行设计本次实验的方案，画出结构框图，详细标出输入输出端口，确定存储器宽度、深度和写使能位数。
4. 学习ISE工具中调用库IP的方法。
5. 本次实验要求调用xilinx库IP实例化一块RAM。实例化的RAM选择为同步

RAM。本次实验的RAM建议设置为两个端口，一个端口用来正常的读写，另一个端口作为调试端口只使用读功能用于观察存储器内部数据。

1. 调用xilinx库IP实例化一块RAM，并进行仿真，得到正确的波形图。
2. 将以上设计作为一个单独的模块，设计一个外围模块去调用该模块，见图

6.1。外围模块中需调用封装好的LCD触摸屏模块，显示RAM的正常端口的地址、待写入的数据和读出的数据，显示调试端口的地址和读出的数据。并且需要利用触摸功能输入正常端口的地址和写数据，以及调试端口的地址。

存储器模块

外围

模块

来自

FPGA

板子

上

的输入

输出到

FPGA

板

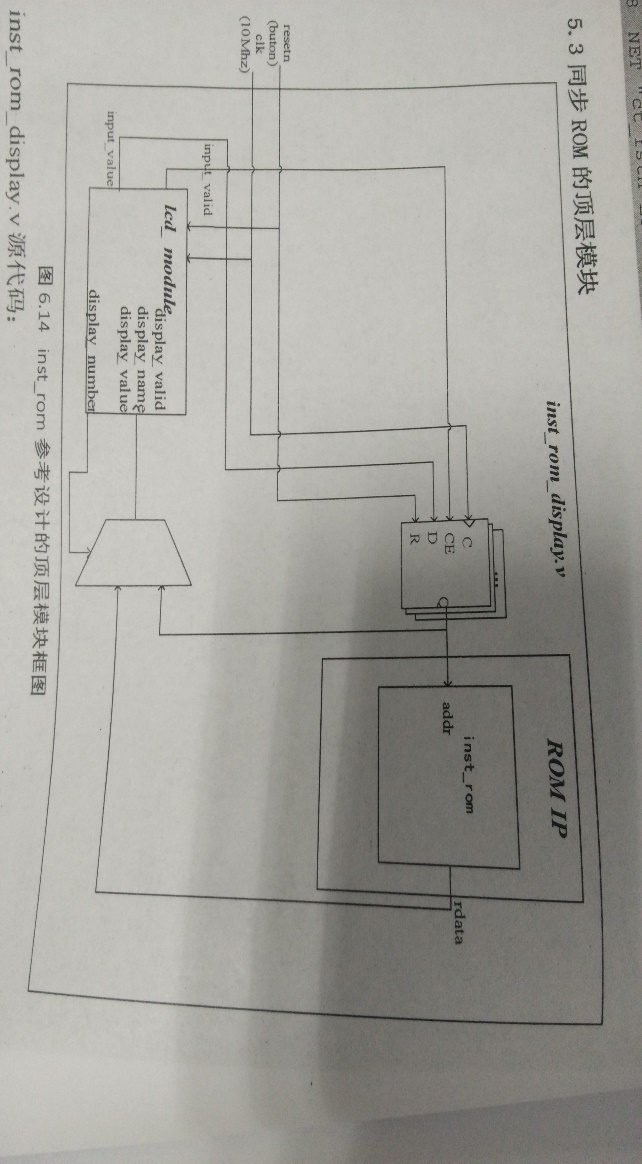
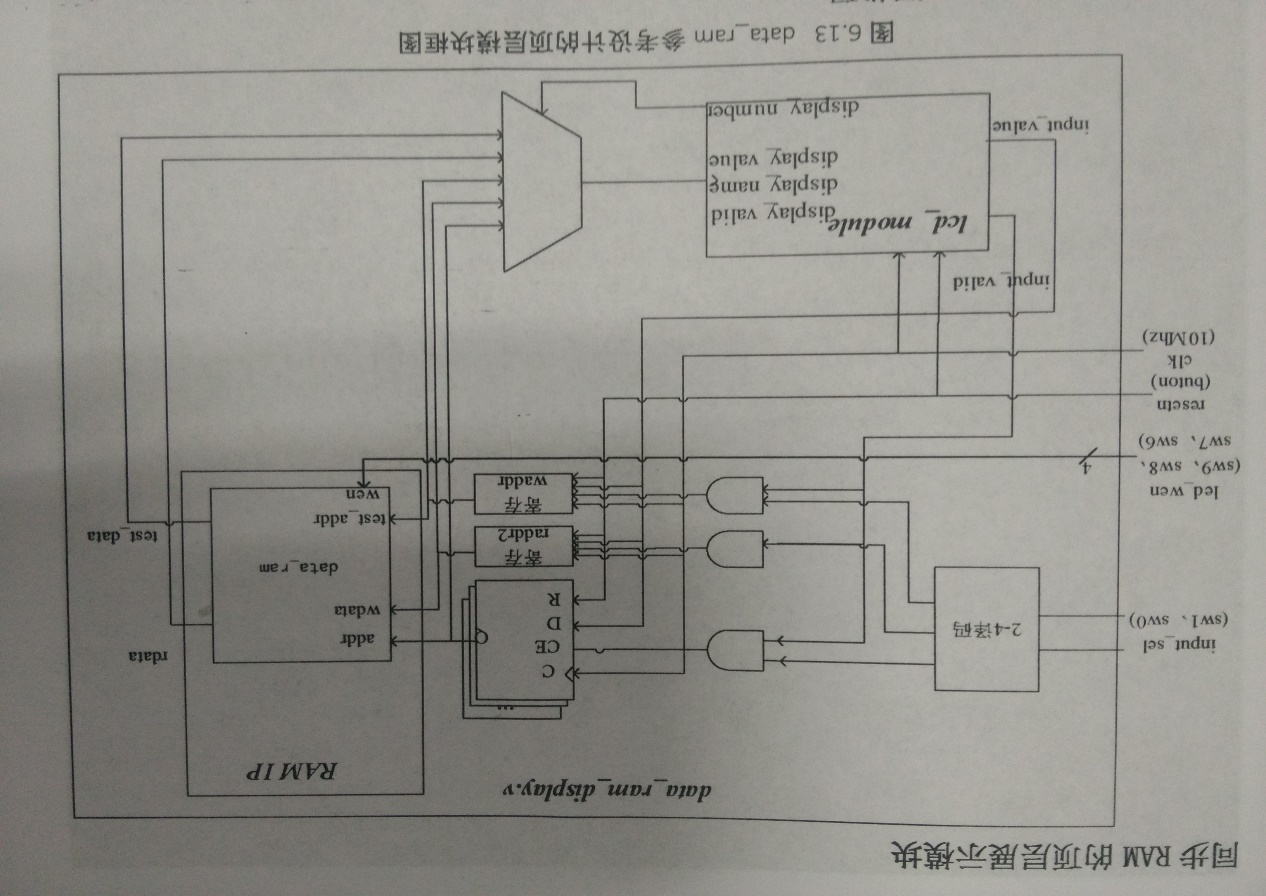
上进行展示

图 6.1 存储器设计实验的顶层模块大致框图

1. 将编写的代码进行综合布局布线，并下载到实验箱中的FPGA板子上进行演示。

注意：存储器深度不要过大，避免耗费过多的FPGA上的资源。本次实验要求实现同步的存储器。而异步存储器的搭建方法同寄存器堆的搭建，但不同的是，寄存器堆中读写端口是分开的，但对于异步RAM要求读写共用一个端口，只是会增加一个写使能信号。可以自行尝试搭建异步的ROM和RAM，在单周期CPU实验中会用到异步的ROM 作为指令存储器，而异步RAM作为数据存储器。

**4** 实验要求

1. 做好预习：
   1. 掌握存储器的工作原理，明白ROM和RAM，同步和异步的区别；
   2. 学习并掌握调用xilinx库IP进行设计的方法；
   3. 确定存储器的输入输出端口及宽度、深度和写使能设计；
   4. 如果对FPGA板了解的话，可确定设计中与FPGA板上交互的接口，画出包含外围模块的整体设计框图，即补充完善图6.1。
2. 实验实施：
   1. 确认存储器的设计方案的正确性；
   2. 编写verilog代码；
   3. 对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材料；
   4. 完成调用存储器模块的外围模块的设计，并编写代码；
   5. 对代码进行综合布局布线下载到实验箱里FPGA板上，进行上板验证。 、
3. 实验检查：
   1. 1）完成上板验证后，让指导老师或助教进行检查，进行现场演示，按照检查人员的要求，对特定存储器单元读/写，可对演示结果进行拍照作为实验报告结果一项的材料
4. 实验报告的撰写及上层存储器设计：
5. 实验结束后，需按照规定的格式完成实验报告的撰写。
6. 存储器的上层设计：

**5** 调用 **xilinx** 库 **IP** 的方法

本部分以生成同步RAM和ROM为例说明调用xilinx库IP的方法。

5.1生成IP核RAM 1) 新建工程：

2）新建ip

通过“New Source”新建一个源文件，类型选择“IP”：

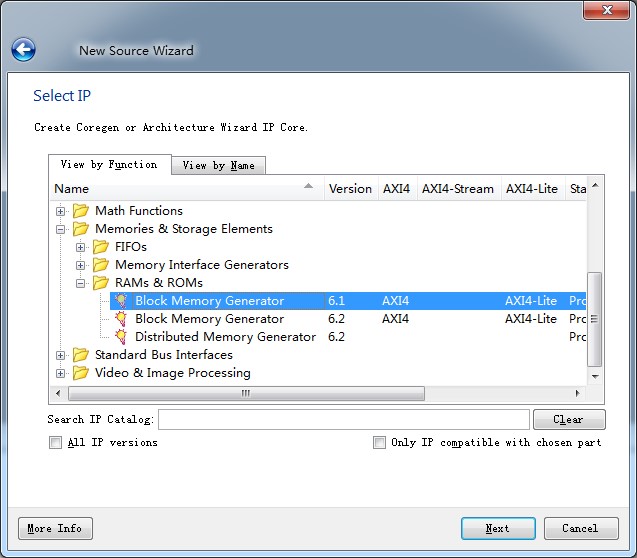
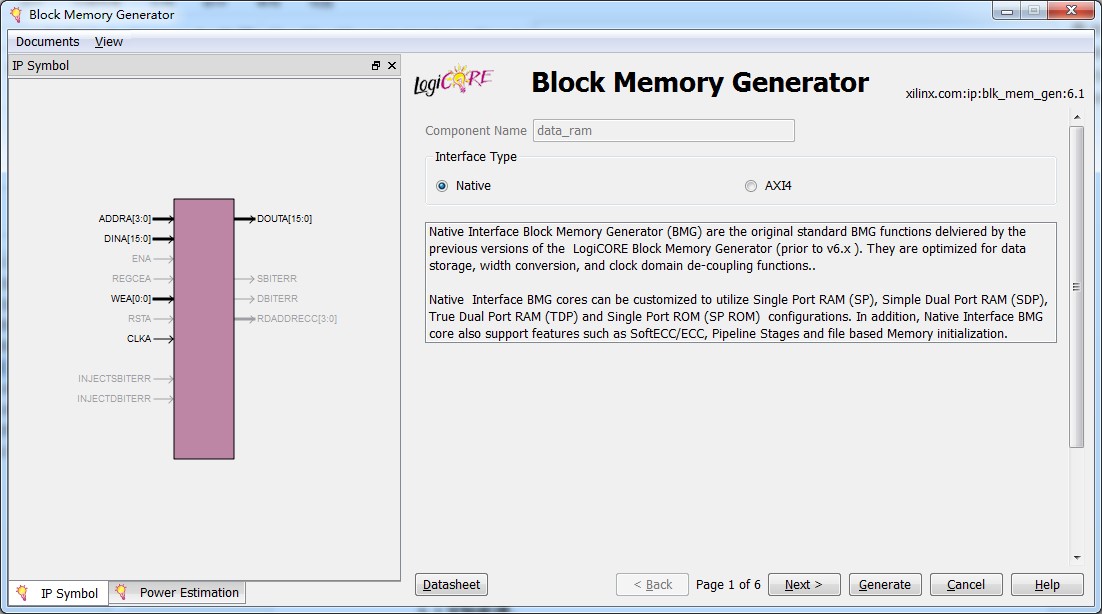


图 6.4 选择 IP 类型

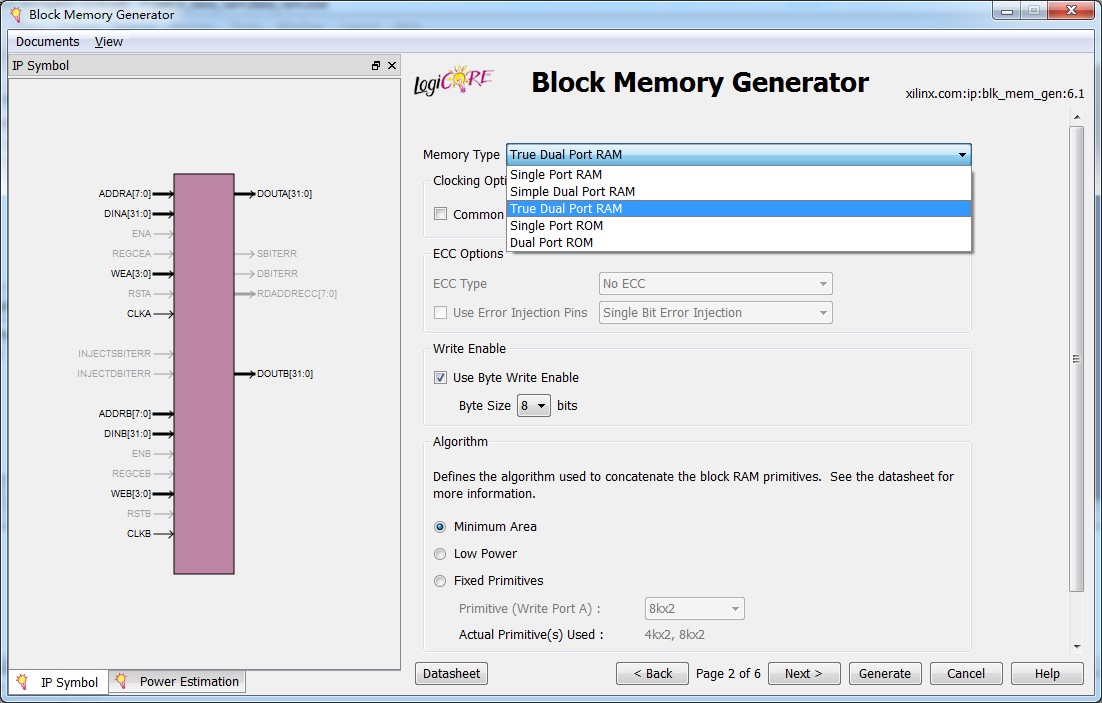
点击“Next”，再点“Finish”完成创建。

1. 设置RAM参数

完成第二步后，会出现如下界面，需要依次选择Memory的参数：

图 6.5 Memory 设定参数界面

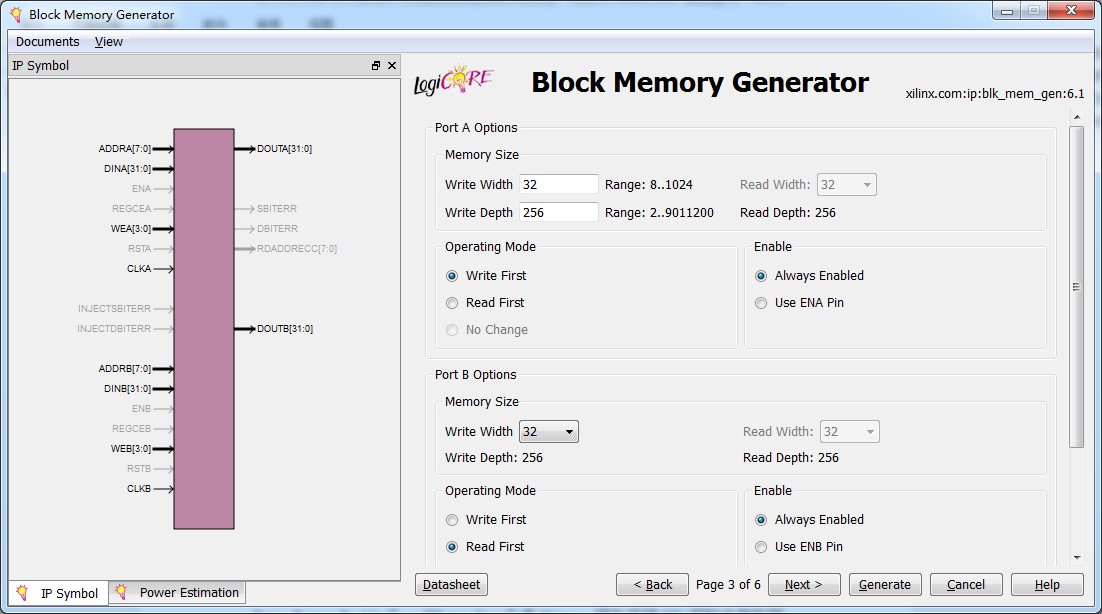
在图6.5中，直接点击“Next”：

图 6.6 选择 Memory 类型

在图6.6中，选择Memory类型为“True Dual Port RAM”，一个端口作为正常的

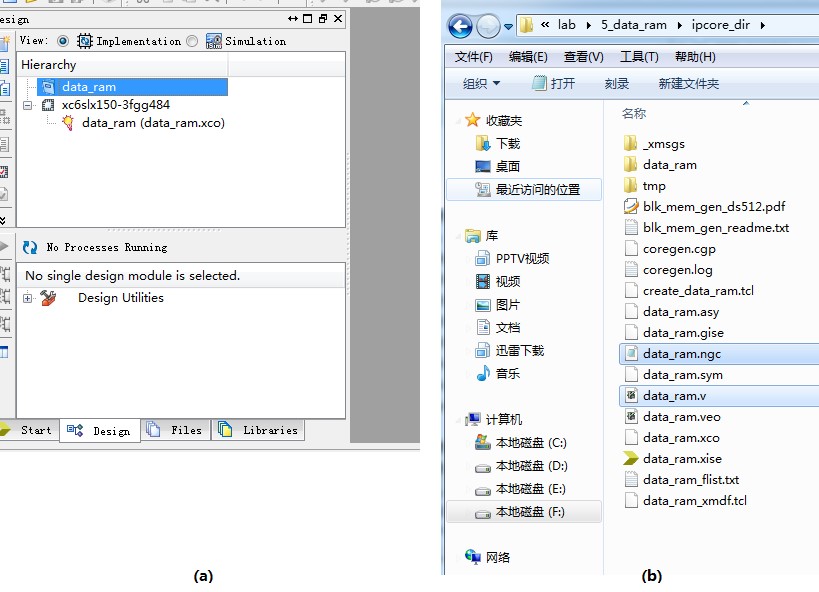
读写端口，一个端口作为调试端口。勾选“Write Enable”下的“Use Byte Write Enable”，“Byte Size”选8bits，因为后续CPU实验中存在写一个字节的store指令，故需要数据RAM为字节写使能。

点击“Next”：

图 6.7 设置 RAM 宽度和深度

此处RAM宽度设置为32位，深度为256，因为后续CPU实验是基于32位数据运算的。

对于一般的RAM生成，后续的步骤都不需要了，故此处直接点击“Generate”生成IP核即可。

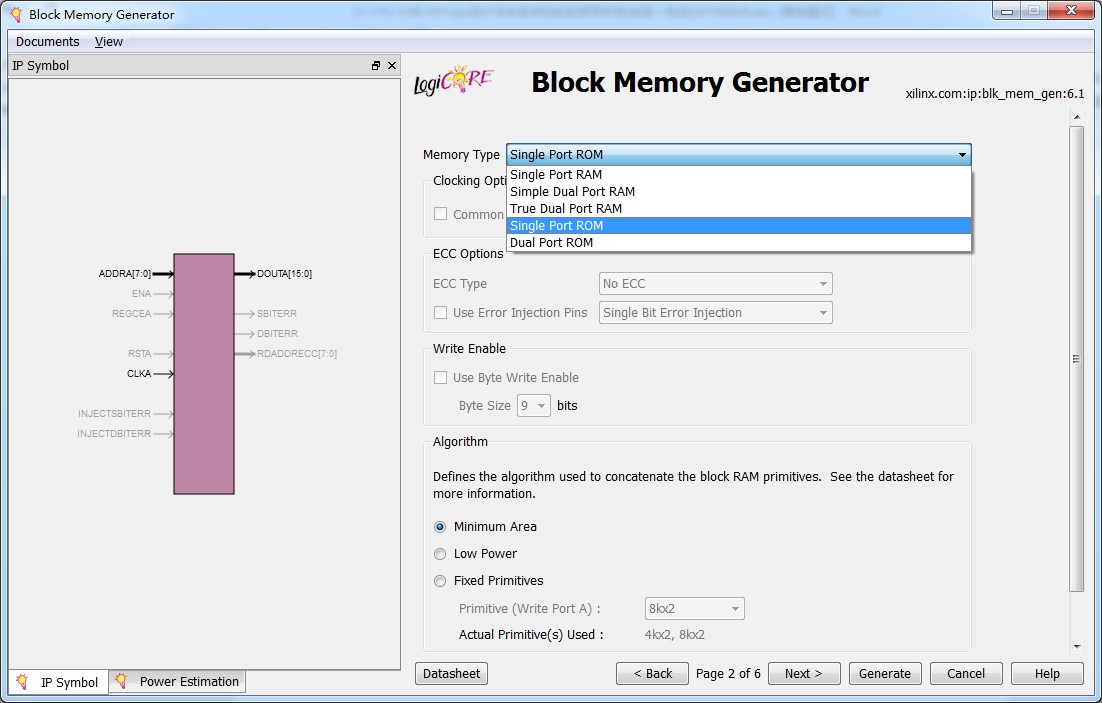
图 6.8 生成 RAM 成功

生成成功的工程管理界面如图 6.8-(a)，.xco 文件为存放 IP 核配置信息的文件，以后需要更改直接双击该文件即可重新配置 IP 核。在图 6.8-(b)所示的工程目录下，可以看到“data\_ram.ngc”和“data\_ram.v”文件，前者为生成的IP核对应的网表文件，后者为只封装了接口的黑盒verilog文件，在需要调用该IP核时，通过“Add Copy of Source”添加这两个文件即可

5.2生成IP核ROM

ROM为只读存储器，需要初始化内部数据，可作为指令存储器。生成ROM的方法同RAM类似，首先新建一个IP核，取名为inst\_rom,后续步骤同5.1所述，但在图

6.6那步时，Memory类型需要选择为“Single Port ROM”，如下图：

图 6.9 生成 ROM 时选择 Memory 类型

由于ROM为只读的，故我们不需要增加调试端口去观察内部数据的变化，故此处选择单端口即可。

点击“Next”设置宽度和深度，如下图：

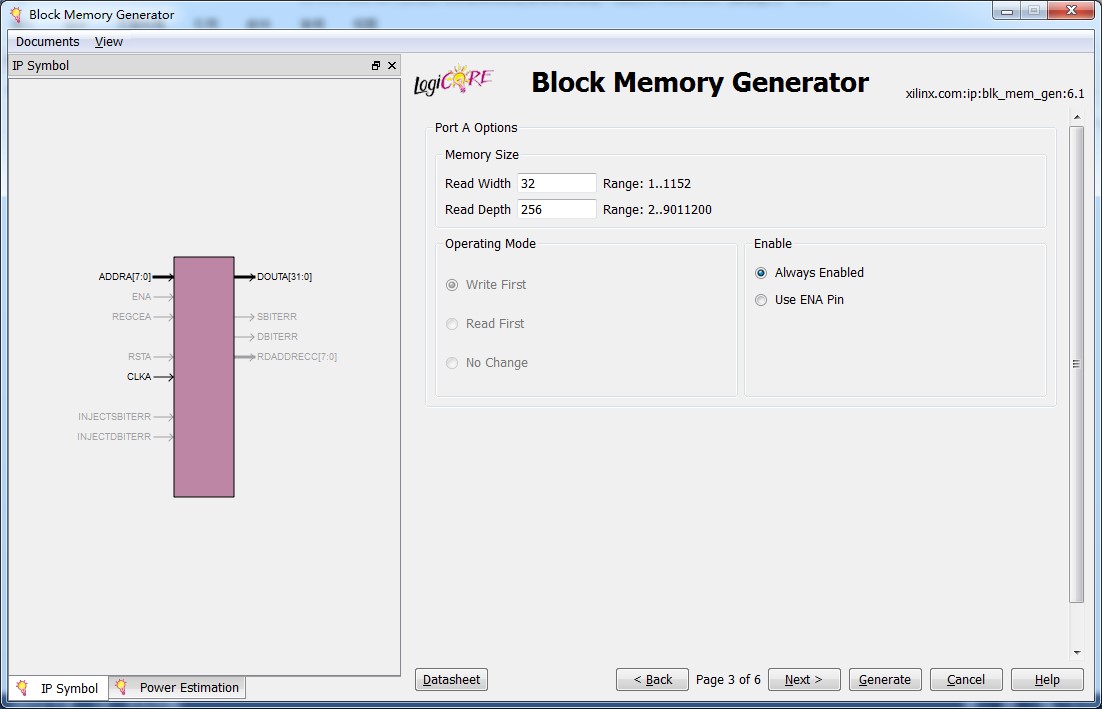


图 6.10 生成 ROM 时宽度和深度

宽度需要设置为32位，因为一条指令占用32位，深度可以依据自己要执行的指

令数设定，此处先设定为256。点击“Next”：

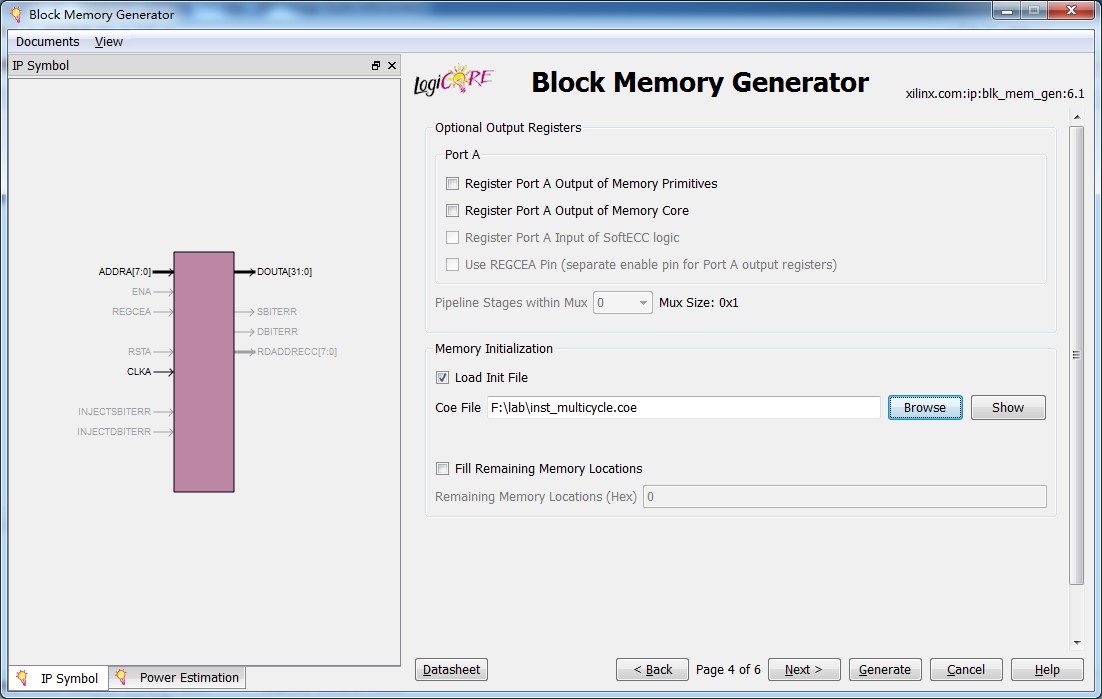


图 6.11 给 ROM 装载初始数据

在图6.11中，需要勾选“Load Init File”,并选中需要装载的初始化文件(.coe

文件)。.coe文件为ISE中ROM初始化文件，其格式如下：

1. memory\_initialization\_radix = 16;
2. memory\_initialization\_vector =
3. 24010001
4. 00011100
5. ……

第一行指定了初始化数据格式，此处为16进制，也可以设置为2进制。第二行说明从第三行开始为初始化的数据向量，由于ROM宽度为32位，故一个初始化向量为

32位数据。初始化向量之间必须用空格或换行符隔开，此处使用换行符，故一行为一个初始化向量。初始化数据会从ROM中的0地址处开始依次填充。当初始化数据格式设置为2进制时，后续的初始化向量需要用二进制编写。

至此，生成ROM的所有参数都已设置完成，直接点击“Generate”生成ROM即可。

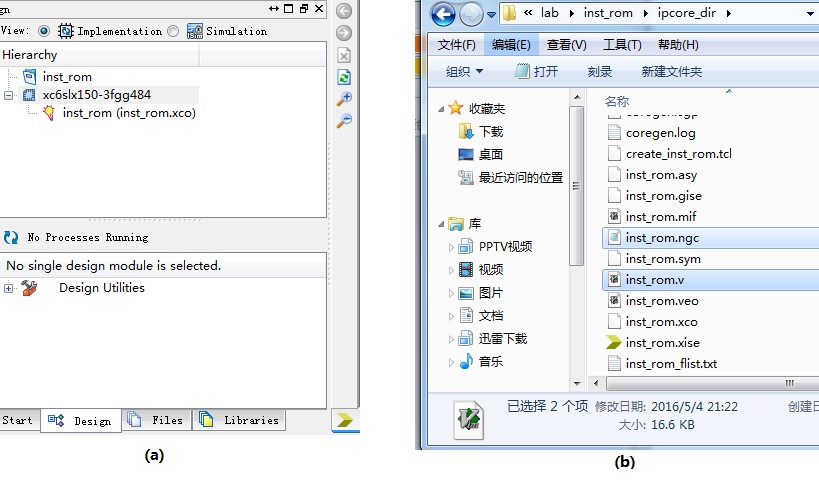
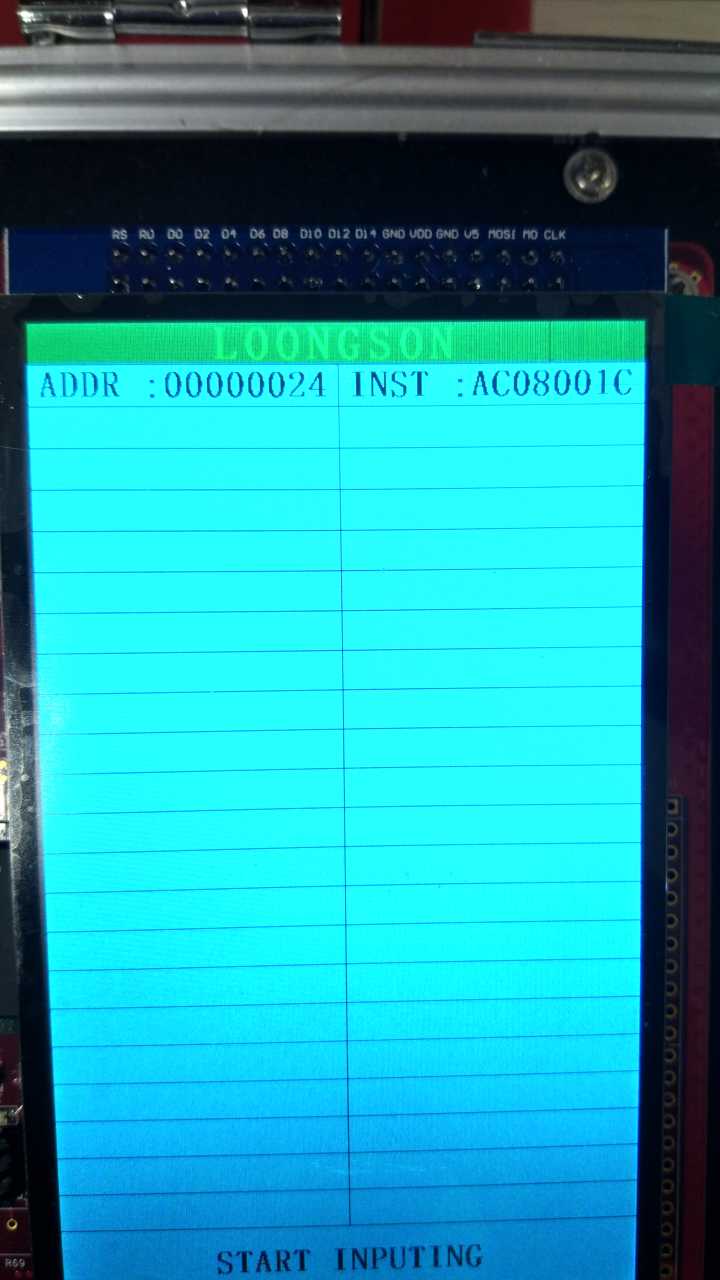


图 6.12 生成 ROM 成功

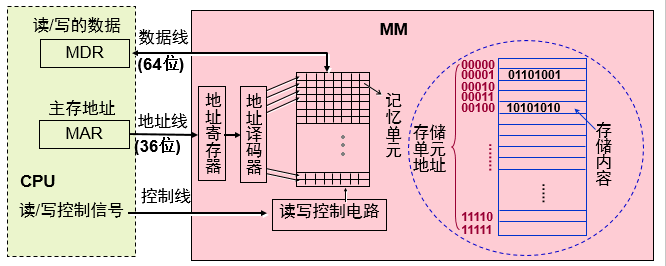
从图6.12中可以看到，生成结果和RAM的类似，以后需要更改参数设置或初始化文件时，双击图(a)中的xco文件即可；而需要在别的模块中调用inst\_rom时，添加图(b)中的“inst\_rom.ngc”和“inst\_rom.v”即可。需要注意的是图6.12(b)中还有一个文件为“inst\_rom.mif”，存放了IP核inst\_rom的初始化数据,在对调用了该 inst\_rom IP的模块进行仿真时，需要将该mif文件拷贝到工程目录下，仿真时会加载该文件对inst\_rom进行初始化。比如后续多周期和流水CPU实验需要用到同步rom 作为指令存储器，需要将对应的mif文件拷贝到工程目录下，才能仿真成功，否则指令ROM里为空，取不到指令。





## 6 、实验结果分析

存储器在计算机系统中用来存放程序和数据。通俗地讲：存储器使计算机系统有了记忆能力，从而能自动地从存储器中取出保存的指令按序进行操作。指令执行过程中需要访问主存时，CPU首先把被访问的单元的地址送到储存地址寄存器中，然后通过地址线将主存地址送到主存中的地址寄存器，以便地址译码器进行译码选中的相应单元，同时，CPU将读写信号通过控制线送到主存的读写控制电路。执行过程中：数据线的宽度与主存数据寄存器的宽度相同，地址线的宽度与MAR的宽度相同。主存结构图如下：



写操作：CPU同时将要写的信息送到主存数据寄存器，在读写控制电路的控制下，经数据线将信息写入选中的单元。

读操作：主存读出选中单元的内容送数据线，然后被送到主存数据寄存器。

* CPU和主存之间有同步和异步两种通信方式

（1）异步方式（读操作）过程（需握手信号）：CPU送地址到地址线，主存进行地址译码。CPU发读命令，然后等待存储器发回“完成”信号。主存收到读命令后开始读数，完成后发“完成”信号给CPU。CPU接收到“完成”信号，从数据线取数。写操作过程类似。

（2）同步方式：CPU和主存由统一时钟信号控制，无需应答信号。主存总是在确定的时间内准备好数据，CPU送出地址和读命令后，总是在确定的时间取数据，存储器芯片必须支持同步方式。

* xilinx库IP进行设计的方法

（1）调用xilinx库IP实例化一块RAM。实例化的RAM选择为同步 RAM。本次实验的RAM 建议设置为两个端口，一个端口用来正常的读写，另一个端口作为调试端口只使用读功能用于观察存储器内部数据。

（2）调用xilinx库IP实例化一块RAM，并进行仿真，得到正确的波形图。

（3）以上设计作为一个单独的模块，设计一个外围模块去调用该模块，见下图。外围模块中需调用封装好的LCD触摸屏模块，显示RAM的正常端口的地址、待写入的数据和读出的数据，显示调试端口的地址和读出的数据。并且需要利用触摸功能。

（4）将编写的代码进行综合布局布线，并下载到实验箱中的 FPGA板子上进行演示。

## 六、实验体会与收获：

通过这次试验的体验，让我们真正的体会到了硬件层面的计算机世界，拓宽了我们对计算机的认识，第一次真正意义上的从门电路去理解计算机世界，感受到了模块化的计算机组成。总体来说也是收益颇丰。首先设计模块的程序框图，然后搞清楚模块的输入和输出以及控制信号的关系。然后用Xilinx ISE软件将我们所写的存储模块和外围模块以及所写的testbench文件，经过软件进行仿真测试之后，就可以通过下载器将它下载到实验箱中，在LED屏幕上完成该模块的演示操作。这次开发的过程中，我们开始忽略了拨码开关与实验时变量之间的对应关系，导致了我们进行了错误的操作，在经过商量讨论了我之后，我们发现了问题的所在，然后将它解决了。通过这个过程，我明白了不论进行什么工程开发，我们都应该理清楚各个部分之间的组成和关系，然后实际进行实现，否则中间会出现很多问题。在这个过程中，我们遇到了一些问题，除了互相讨论之外，老师也对我们进行了耐心的指导和帮助，非常感谢老师的帮助。