版本

更新记录	文档名	实验指导书_lab4
	版本号	0.2
	创建人	计算机组成原理教学组
	创建日期	2022/1/1

更新历史

序号	更新日期	更新人	版本号	更新内容
1	2022/1/1	陈颖琪	0.1	初版，单周期 CPU IO扩展实验
2	2023/5/7	陈颖琪	0.2	增加对产生bit流时可能出现的错误的解决方式的说明

文档错误反馈:

Yingqichen@sjtu.edu.cn

1 实验四 简易单周期 CPU I/O扩展和新指令扩展实验

本次实验在实验三完成的单周期 CPU核基础上进行外设拓展，增加CPU对I/O口的读写支持，以及对新指令的功能支持。并要求在EGO1实验板上完成板级测试验证。

在进行本次实验前，你需要具备以下基础能力：

1. 理解CPU 的外设地址空间和数据空间的关系，理解I/O数据通路。
2. 理解指令译码和控制器的设计原理。

1.1 实验目的

1. 掌握CPU 的外设I/O 模块的设计方法。理解 I/O 地址空间的译码设计方法。
2. 掌握Vivado 仿真、实现、板级验证方式。
3. 通过扩展新指令的实现，深入理解CPU对指令的译码、执行原理和实现方式。

1.2 实验设备

1. 计算机 1 台(尽可能达到 8G 及以上内存)；
2. Xilinx 的Vivado 开发套件(2020.2 版本)。
3. Xilinx 的EGO1 FPGA开发板

1.3 实验任务

1.3.1 实验要求

阅读实验原理完成以下任务：

1. 添加I/O接口模块，实现CPU对I/O空间的访问。
2. 依据提供的标准测试程序代码，对单周期CPU的I/O模块进行板级验证。要求CPU能够采用查询方式接收输入按键或开关的状态，并产生相应的输出状态，驱动数码管显示结果，从而验证CPU可正确执行I/O读写指令。
3. 自定义一条R-Type新指令，求两个32bit操作数的汉明距离。修改CPU控制部件和执行部件模块代码，支持新指令的操作。
4. 修改提供的测试程序代码，对新增加指令进行板级验证。

实验给出顶层文件，可参考兼容顶层文件端口设定。实验最终以板级输出结果判断是否成功实现要求的功能。

1.3.2 实验步骤

1. 从实验三导入CPU核，sc_computer及其所有子模块，修改其子模块sc_datamem，添加io_input.v、io_output.v、在其中添加对I/O的支持；
2. 修改sc_cpu，sc_computer的端口，添加I/O端口定义，改sc_computer为sc_computer_main模块。
3. 从实验二中导入display、seg7模块；
4. 从实验包中导入in_port.v，clock_and_mem_clock.v、out_port_hex2dec.v；
5. 使用Block Memory，例化指令ROM和数据RAM， 例化 lpm_rom_irom时导入本实验提供的I/O测试汇编指令机器码coe 文件lpm_rom_irom_io.coe;
6. 从实验包中导入本实验顶层文件sc_cpu_iotest.v，参考实验原理，补充完整，连接各子模块；
7. 导入仿真文件TB_sc_computer_iotest.v，设置其为仅用于仿真，运行仿真，确认功能正确；
8. 添加管脚约束文件sc_computer_iotest.xdc，指定设计的顶层sc_cpu_iotest.v的管脚与实验板上的时钟、复位、按键、数码管、LED等的管脚绑定；
9. 完成sc_cpu_iotest.v的硬件综合、FPGA实现，产生FPGA下载编程bit文件，注意工程实现时设置的芯片型号与EGO1实验板上芯片型号一致；
10. 连接EGO1 开发板，编程下载，完成I/O功能的板级测试验证。
11. 修改控制器和ALU，增加对新指令的支持；
12. 改写coe 文件lpm_rom_irom_io.coe，执行对新指令的操作，并更新指令ROM的IP。
13. 完成sc_cpu_iotest.v的硬件综合、FPGA实现、编程下载和新指令功能的板级测试验证。

1.2 实验环境

|sc_cpu_iotest.v	设计顶层文件，已提供接口定义。需要自行补充完整子模块连接关系
|–clock_and_mem_clock.v	时钟设置模块，已提供
|–in_port.v	5bit I/O输入管脚扩展为32bit模块，已提供
|–out_port_hex2dec.v	32bit 二进制数据转十进制数据模块（输出仅取低2个位数），已提供
|–Display.v	七段数码管显示驱动模块，已提供
|—seg7.v	七段数码管显示译码模块，已提供
|–sc_computer_main.v	RISC-V软核顶层，连接CPU核与指令、数据存储器, 在实验三sc_computer.v基础上修改实现
|—sc_cpu.v	RISC-V CPU核数据通路和控制模块，在实验三代码上修改端口定义
|——sc_cu.v	控制器模块，使用实验三代码，在其基础上修改以支持新指令
|——dff32.v	PC模块，使用实验三代码
|——alu.v	ALU 模块，使用实验三代码，在其基础上修改以支持新指令
|——immext.v	有符号立即数扩展模块，使用实验三代码
|——mux2x32.v	二选一选择器，使用实验三代码
|——mux4x32.v	四选一选择器，使用实验三代码
|——regfile.v	寄存器堆，使用实验三代码
|——cla32.v	加法器，使用实验三代码
|—sc_instmem.v	指令ROM，例化lpm_rom_irom.ip模块，已提供
|—— *lpm_rom_irom.ip*	*单口ROM IP，自行通过Block memory generator 进行实例化*
|————lpm_rom_irom_io.coe	I/O端口测试程序指令ROM初始值文件，已提供
|—sc_datamem.v	数据RAM，例化lpm_ram_dq_dram.ip模块，已提供，需补充I/O，完善设计
|—— *lpm_ram_dq_dram.ip*	*单口RAM IP，自行通过Block memory generator 进行实例化*
|——io_input.v	I/O 输入接口，已提供，依据需要可修改设计
|——io_output.v	I/O 输出接口，已提供，依据需要可修改设计
|sc_computer_iotest.xdc	综合实现时的管脚约束文件，已提供。
|TB_sc_computer_iotest.v	仿真激励文件，用于对顶层文件sc_cpu_iotest.v的仿真，已提供

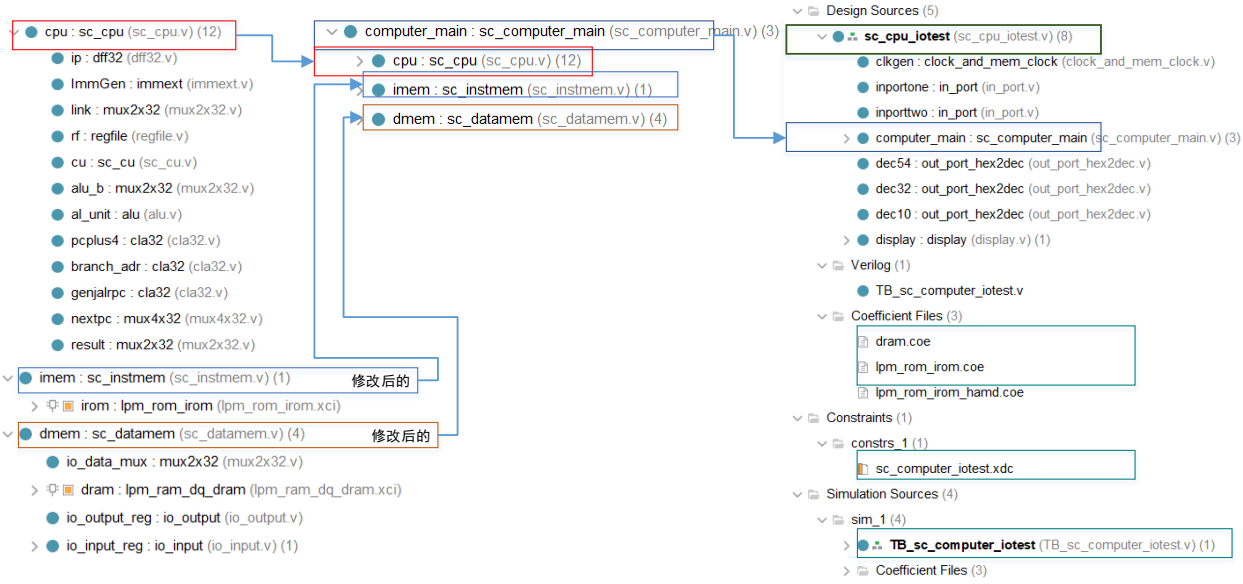

表 1: 实验文件树

2 实验原理

2.1 总体框架及通路图

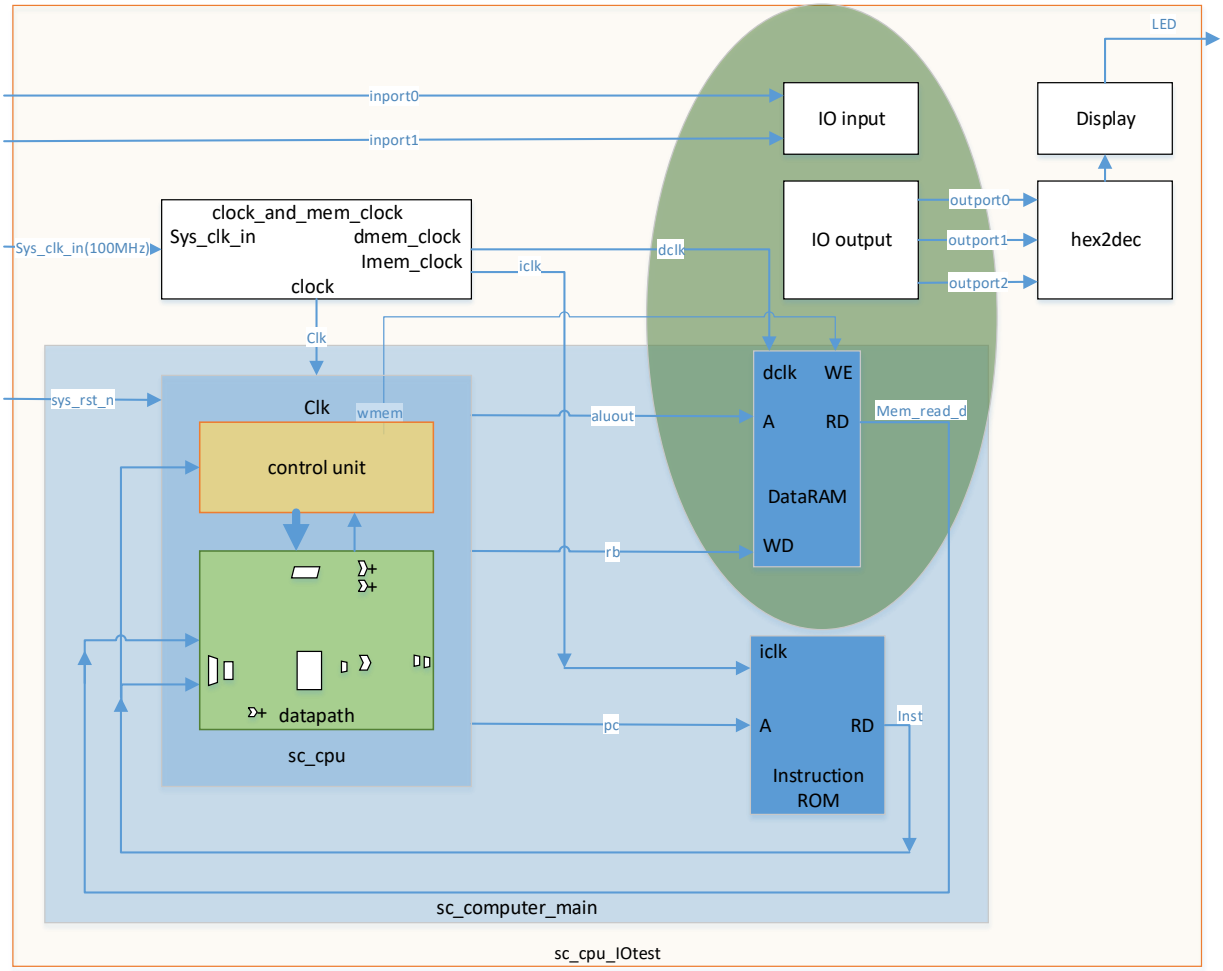

图 1: 单周期 CPU I/O扩展框架图

如图 1，完整的单周期 CPU I/O扩展实现框架图。实验要求给CPU核增加两个32b输入口，inport0，inport1。增加三个32b输出口，outport0-outport2。

在实验三完成的CPU核基础上进行改进。和实验三的单周期 CPU 实现框架图对比，图中的绿色椭圆框内的I/O输入输出模块部分，就是本实验要修改的增加I/O支持的内容。

另外, 可通过对datapath的修改支持实现新指令功能。

图中的clock_and_mem_clock模块是一个独立的模块，实现对存储器读写时钟的处理。其功能是采用板上sys_clk_in作为主时钟输入，在内部给出二分频的clock作为CPU模块的工作时钟，并产生imem_clk、dmem_clk，用于对CPU模块内含的指令和数据存储器读写控制。

可根据实验三3.3节说明的时钟原理，注意与相关模块（sc_cpu、数据RAM和指令ROM模块）的时钟连接，修改对应的信号连接。

hex2dec模块将输出的32b数据，转换为十进制数据。以便显示后更直观。

Display模块为七段数码管译码驱动模块。在实验一中已经使用过。这里，将三个输出口的数据经hex2dec模块转换为十进制数后，以每个数据两位的形式显示在EGO1实验板的七段数码管低6位上。

2.2 设计原理

I/O空间和数据存储器空间是统一编址的。可以用高端地址空间作为IO空间。地址译码可以用高位地址线。统一编址方式I/O 端口扩展原理图解如图2。

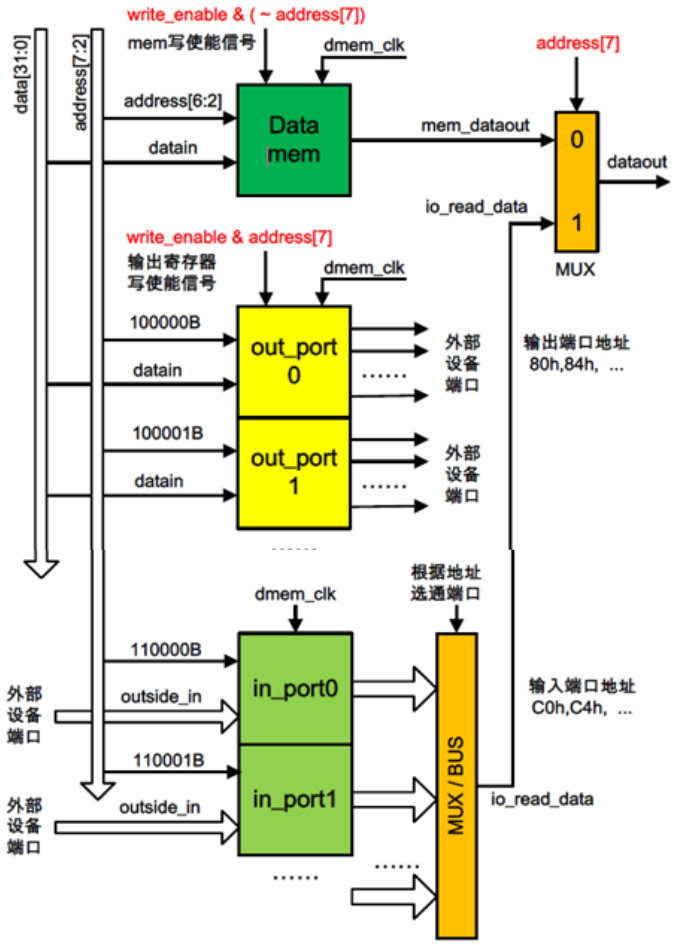

图2: 统一编址方式I/O 端口扩展原理图解

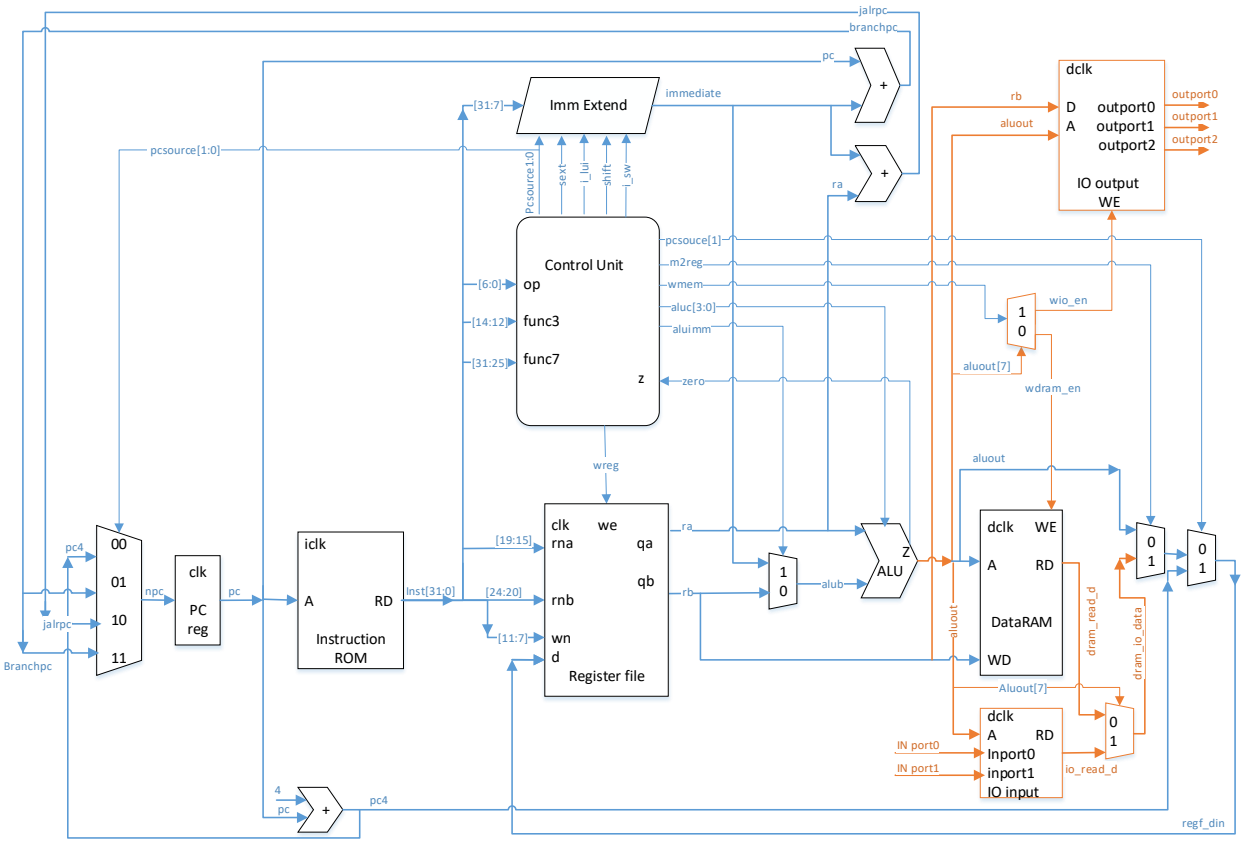

图 3：支持22条指令的单周期 CPU 扩展I/O 框架图

图3为单周期CPU的I/O扩展完整通路图

例如，对可读的I/O口，CPU对inport0或inport1口发出lw指令，读取数据。在图2中的io_input模块，一是增加对两个I/O端口数据的锁存，二是要增加对不同端口对应地址的译码。根据不同地址，输出数据io_read_d，提供给CPU的寄存器堆。

因此这里还添加了一个二选一的MUX，根据地址空间译码范围，选择送数据存储器的输出dram_read_d还是读入的IO口数据io_read_d 给到寄存器堆的写数据端口d。

对可写的I/O口，CPU对outport0或outport1或outport2口发出sw指令，写出数据。在图2中的io_output模块，根据输出端口地址信号aluout的译码，选择将来自寄存器堆的输出数据rb，输出到对应的输出端口outport0或outport1或outport2。

原来的数据存储器地址空间wmem信号相应需要调整，添加译码逻辑，分为两个写允许信号，根据地址译码区分是写数据到存储器（wdram_en） 还是写到I/O口（wio_en）。

对于增加R-Type新指令的支持，要修改哪些部分？是否要增加新的模块？请自行思考实现。

3 仿真验证

3.1 测试程序

一个测试IO功能的简单测试程序如下所示。及其机器码coe文件lpm_rom_irom_io.coe也在实验包中提供，用以验证I/O读写过程。也可以自己编写或修改该段示例程序代码，以测试对新指令的支持。该段代码主要完成循环读取两个端口数据，做加法，再输出两个加数以及两个数之和的值到三个输出端口。功能描述及对应机器码参见表1。

```
main: addi x11, x0, 128
addi x12, x0, 132
addi x13, x0, 136
loop: lw   x14, 0(x11)
lw x15, 0(x12)
add x16, x14, x15
sw x14, 0(x11)
sw x15, 0(x12)
sw x16, 0(x13)
j loop
```

表1 测试程序的机器码和功能描述

Pc	标号	代码	机器码（二进制）	机器码（16 进制）	描述	指令类型
0	Main	addi x11, x0, 128	00001000000000000000010110010011	08000593	#X11<-0x80, IOPORT 0 byte address	I
4		Addi x12, x0, 132	00001000010000000000011000010011	08400613	#X12<-0x84, IOPORT 1 byte address	I
8		Addi x13, x0, 136	00001000100000000000011010010011	08800693	#X13<-0x88, IOPORT 2 byte address	I
c	Loop	lw x14, 0(x11)	00000000000001011010011100000011	0005a703	#X14 <- [80]	I
10		Lw x15, 0(x12)	00000000000001100010011110000011	00062783	#X15 <- [84]	I
14		Add x16, x14, x15 （Hamd x16，x14，x15）	00000000111101110000100000110011	00f70833（40f77833）	#X16<- [80] +[84], (HAMD, NEW INST: X16<- HAM DISTANCE OF [80] AND [84])	R
18		Sw x14, 0(x11)	00000000111001011010000000100011	00e5a023	#[80] <- X14	S
1c		Sw x15, 0(x12)	00000000111101100010000000100011	00f62023	#[84] <- X15	S
20		Sw x16, 0(x13)	00000001000001101010000000100011	0106a023	#[88] <- X16	S
24		J loop	11111111010011111111100000110111	fe9ff06f	#Loop, go to pc = 0x 0000000c	U J

假如将add换为新指令hamd，求两个32bit数的汉明距离，也就是求两数对应不相同的位的数量之和。自定义R-type指令hamd rd, rs1, rs2的func3=111,func7=0100000,op=0110011，则有hamd x16, x14, x15, 指令码为 0100 000|0 1111| 0111 0|111 |1000 0|011 0011 =40f77833。

3.2 仿真激励文件与仿真验证

测试用仿真激励文件TB_sc_computer_iotest.v随实验包提供。完成modelsim仿真，对照仿真波形，检查指令的执行结果与RISC-V CPU模型虚拟仿真结果是否一致。通过仿真波形，可查看到示例程序的运行过程符合程序设想，PC指针、指令数据、结果数据等均正确。

图4为仿真结果的部分截图，程序执行到400ns，完成第一次计算求和的输出。690ns完成第二次求和输出。Outport2输出inport0和inport1的和值。Outport0输出值与inport0一致。Outport1输出值与inport1一致, 结果正确。Display模块输出结果正确。程序之后不断循环。

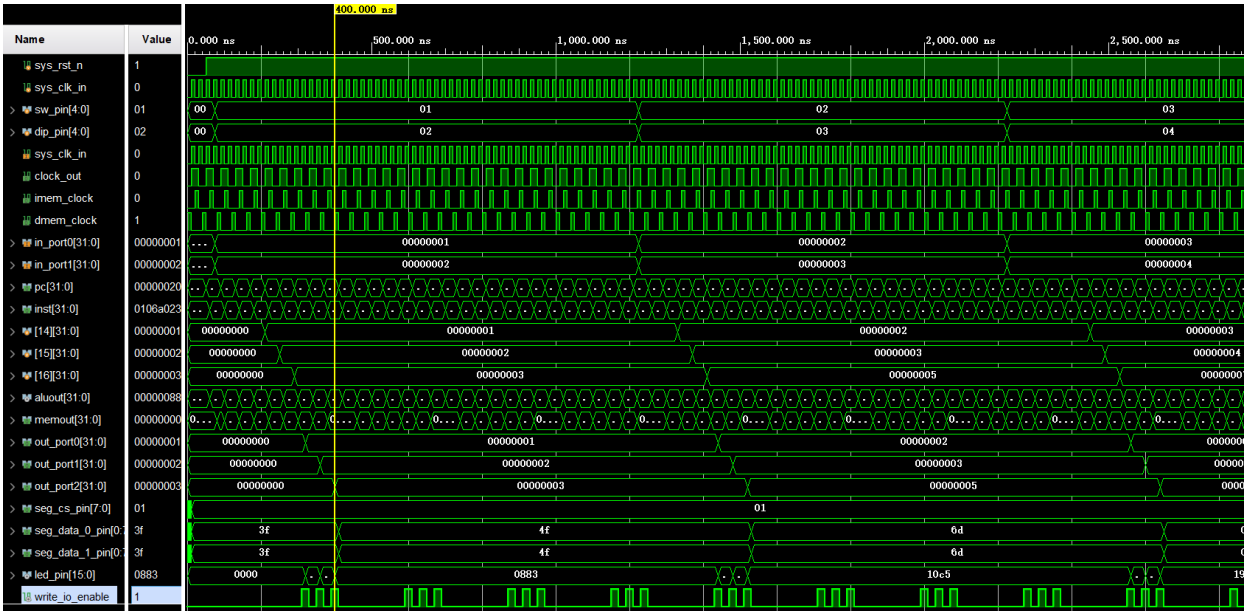

图4

4 板级验证

参照实验一中测试ALU功能的方式，完成电路管脚分配与连接。完成设计的综合、实现与FPGA下载。

将sw_pin、dip_pin各5bit 作为输入端口inport0 和inport1的低5位，进行数据位数扩展后，相加，以十进制结果输出其值于outport0到outport2，连接到七段数码管低6位显示。验证实际输出结果是否符合预期。图5 所示为sw_pin输入11000（24），dip_pin输入（11111）31时的求和输出结果55。

修改coe文件和数据通路代码，重新综合实现你的设计，验证新添加的自定义指令是否能够正确实现。

注意：在综合实现过程中，进行 bit 流文件的生成时，控制台可能出现如图6的错误警告。可以查阅资料了解一下其含义。每次进行综合实现步骤后，可能会提示不同的信号路径上出现这样的问题。根据给出的提示，选择其中任意一条路径，此处以computer_main/cpu/al_unit/register [1][31]_i_18信号为例，在约束文件中加入以下约束即可正常生成 bit 流文件：

```
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets computer_main/cpu/al_unit/register [1][31]_i_18]
```

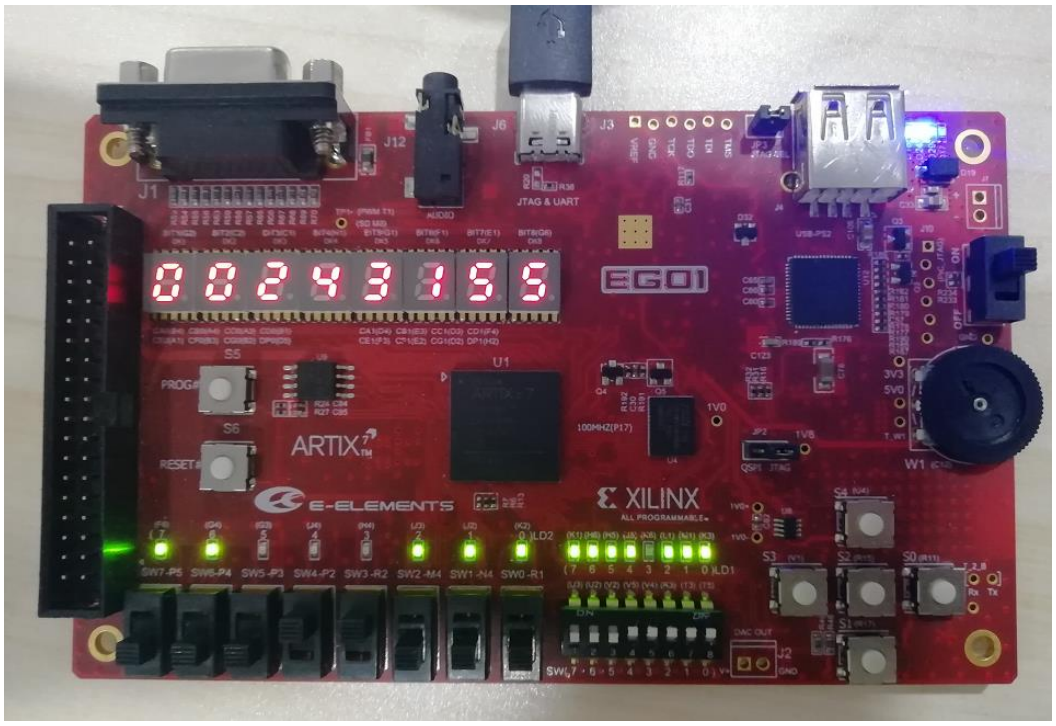

图5 板级测试结果

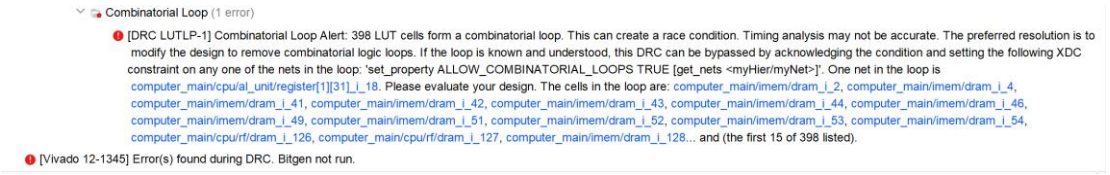

图6 lab4综合实现过程中产生bit流时的一个错误

5 实验报告要求

实验结束后完成实验报告，应简明扼要，重点突出。具体内容包括以下几部分：

1．实验目的。

2．本次实验所用的实验平台、仪器以及其它实验器材和部件等。

3．实验任务。这部分是实验报告最主要的内容。根据任务要求，描述设计原理与思路，设计的实现方式，结构框架等。重点介绍实验三基础之上增加的内容。（不要仅贴整段Verilog 程序代码，而是以表述介绍设计思路为主，如有必要，可贴关键代码段，并加注释）。结合实验原理，描述设计的全过程和实验步骤。

应该说明的几个要点：

I/O模块：阐述I/O设计思想方法，添加几个IO，地址定义方式，对应代码做了什么添加，在何处做了何种修改;给出测试程序流程图、对应汇编语言代码。

扩展指令：指出增加指令需要代码里何处添加修改，新指令格式各位段二进制码，新指令对应的机器码，支持扩充指令的sc_cu控制器模块各相应控制信号值等

4．板上验证结果照片或演示视频截图。演示总结。

5．整理、归纳做完的实验内容并解释相应的实验现象或问题。最后也可总结心得体会，给出建议意见。