单周期 CPU 的 I/O 接口模块设计仿真

概述

本次实验借助提供的文字材料，实现了单周期 CPU 的 I/O 接口模块的设计，并进行了波形仿真，确认了其与外部设备交互的可行性。

实验目的

实验目的包括以下两点：

1. 在理解计算机 5 大组成部分的协调工作原理，理解存储程序自动执行的原理和掌握运算器、存储器、控制器的设计和实现原理基础上，掌握 I/O 端口的设计方法，理解 I/O 地址空间的设计方法。
2. 通过设计 I/O 端口与外部设备进行信息交互。

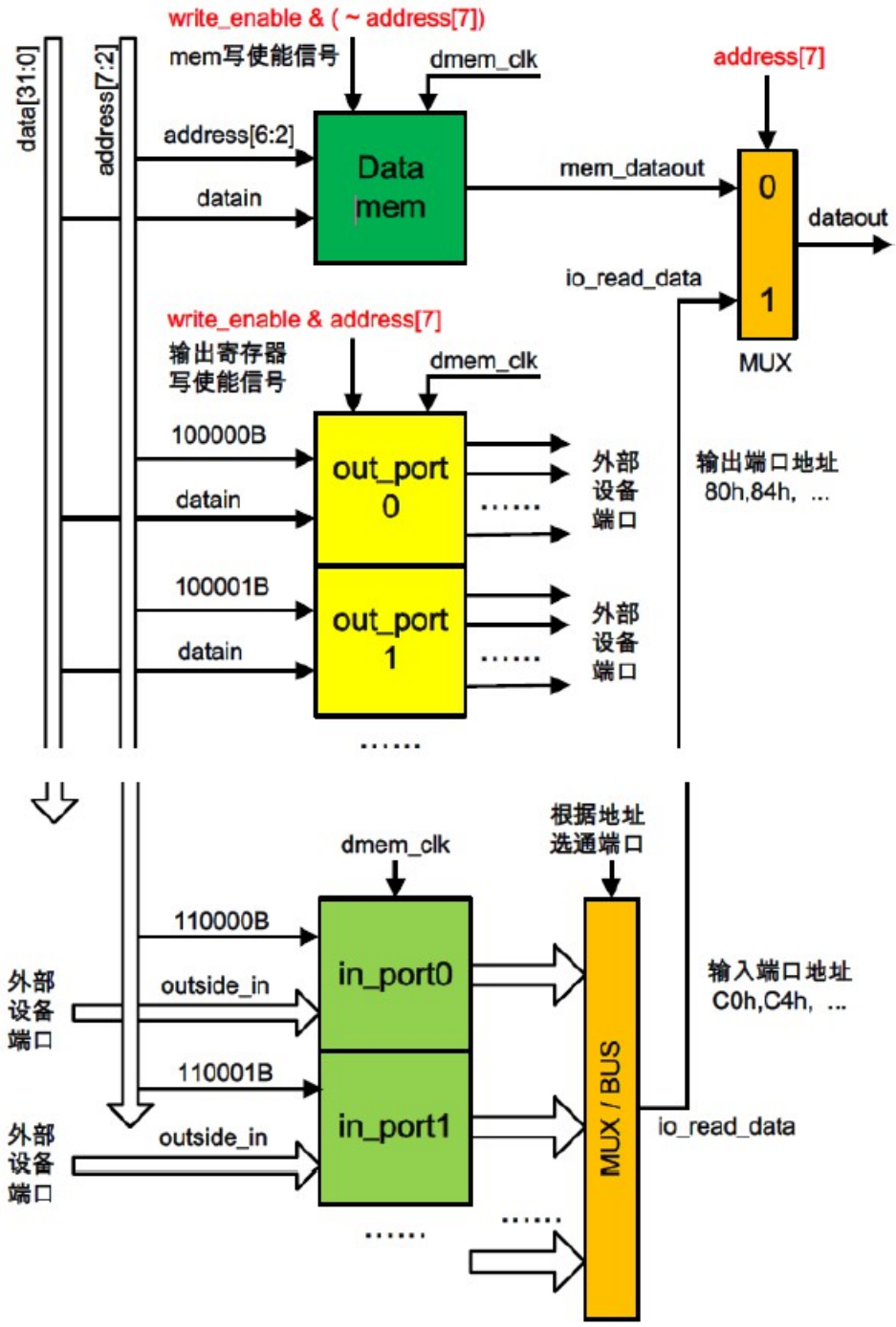
实验步骤

本次实验可以分为下面几个步骤：增加 I/O 地址空间、设计 I/O 端口、设计外部设备、添加顶层模块、测试并得到实验结果。

增加 I/O 地址空间

本次实验中采用的是 I/O 统一编址方式，即将输入输出的 I/O 地址空间作为数据存取空间的一部分。通过地址总线的最高位（即 `addr[7]`）判定要读写的地址属于数据存储器还是 I/O。如果是 1，则读写对象为 I/O；如果是 0，读写对象为数据存储器。

因此，需要对数据存储器的模块文件（即 `sc_datamem.v`）进行改写。改写依据的实验原理图如下所示（来自实验报告书）。



改写后的代码和实验指导书中给出的实例代码基本相同，在此不给出。

设计 I/O 端口

随后需要设计 I/O 的输入输出端口。这两个端口各有一些寄存器用于保存 I/O 需要的数据。

两个端口分别被设计成 `io_input` 模块和 `io_output` 模块。同时还需要实现一个 I/O 输入端口用的多选器，以根据地址总线上的地址选出要从 `dataout` 口送出的数据。这个多选器被设计为 `io_input_mux` 模块。

三个模块各自放在以模块名命名的 `.v` 文件中。这一部分的代码和实验指导书中给出的实例代码基本相同，在此不给出。

设计外部设备

为了测试设备的交互，需要先设计较为方便的、可供观测的外部设备。

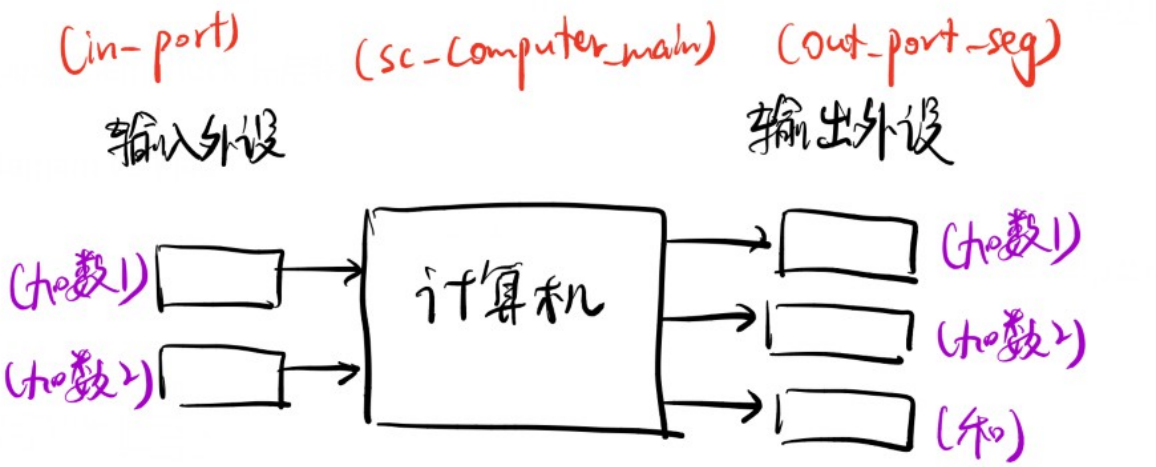
这里使用的外部设备的设计和实验指导书中相同。即输入设备可以输入为五位二进制数，输出设备包含两个七位译码器，用于显示两位的十进制数。

测试方案也和实验指导书中相同。使用两个输入设备作为两个加数的输入，和三个输出设备分别输出两个加数与和。

输入设备设计为模块 `in_port`，输出设备被设计为 `out_port_seg`，七位译码器被设置为 `sevenseg`。这三个部分的实现较为容易，故此处不展示其代码。

添加顶层模块

构建的设备交互模型如下图所示。



构建了这个模型后，由于现在的计算机成为了整个大的模块的一部分，因此需要修改顶层模块，以将输入设备和输出设备同计算机进行数据通路的连接。

我的做法是：将原有的 `sc_computer.v` 更改名称为 `sc_computer_main.v`，然后新建了另一个 `sc_computer.v` 作为顶层模块。这个新的顶层模块的核心代码如下。

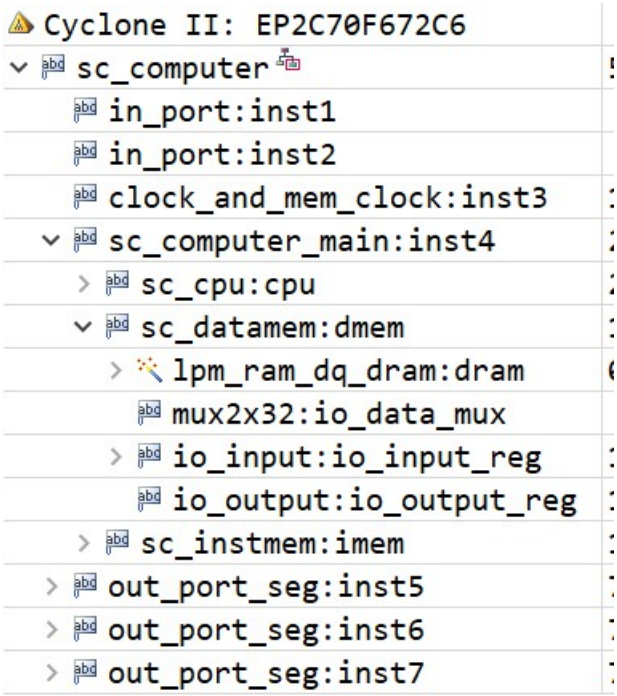
```
module sc_computer (resetn, clk,
    sw0, sw1, sw2, sw3, sw4, sw5, sw6, sw7, sw8, sw9,
    HEX0, HEX1, HEX2, HEX3, HEX4, HEX5);
    // 第一行对应一些接线，第二行对应 I/O 接口

    input               resetn, clk;
    input               sw0, sw1, sw2, sw3, sw4, sw5, sw6, sw7, sw8, sw9;

    output  [6: 0]      HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;

    // 省略一部分代码

    wire    [31: 0]     in_port0, in_port1;
    wire    [31: 0]     out_port0, out_port1, out_port2;

    in_port             inst1(sw0, sw1, sw2, sw3, sw4, in_port1);
    in_port             inst2(sw5, sw6, sw7, sw8, sw9, in_port0);

    clock_and_mem_clock inst3(clk, clock_out, mem_clk);
    // clock_and_mem_clock(main_clk, clock_out, mem_clk);

    sc_computer_main    inst4(resetn, clock_out, mem_clk, pc, inst, aluout,
memout, imem_clk, dmem_clk,
```

```
                                in_port0, in_port1, out_port0, out_port1,
out_port2);
    // sc_computer_main (resetn, clock, mem_clk, pc, inst, aluout, memout,
imem_clk, dmem_clk,
    // in_port0, in_port1, out_port0, out_port1, out_port2,
    // mem_dataout, io_read_data);

    out_port_seg        inst5(out_port0, HEX1, HEX0);
    out_port_seg        inst6(out_port1, HEX3, HEX2);
    out_port_seg        inst7(out_port2, HEX5, HEX4);
    // out_port_seg(data_in, led_out_ten, led_out_mod);

endmodule
```

其中的 `clock_and_mem_clock` 是按照实验指导书中的要求设计的。代码如下。

```
module clock_and_mem_clock(main_clk, clock_out, mem_clk);
    // 采用 main_clk 作为主时钟输入，内部二分频成 clock_out 和 mem_clk
    // 分别作为 CPU 的工作时钟和对 CPU 模块内含的存储器进行读写控制

    input       main_clk;
    output      clock_out;
    // 和仿真中的名称对上
    output      mem_clk;

    reg         clock_out;

    assign      mem_clk = main_clk;
    // 直接连通到 mem_clk

    initial
    begin
        clock_out <= 1'b0;
    end

    always @ (posedge main_clk)
    begin
        clock_out <= !clock_out;
        // 二分频
    end

endmodule
```

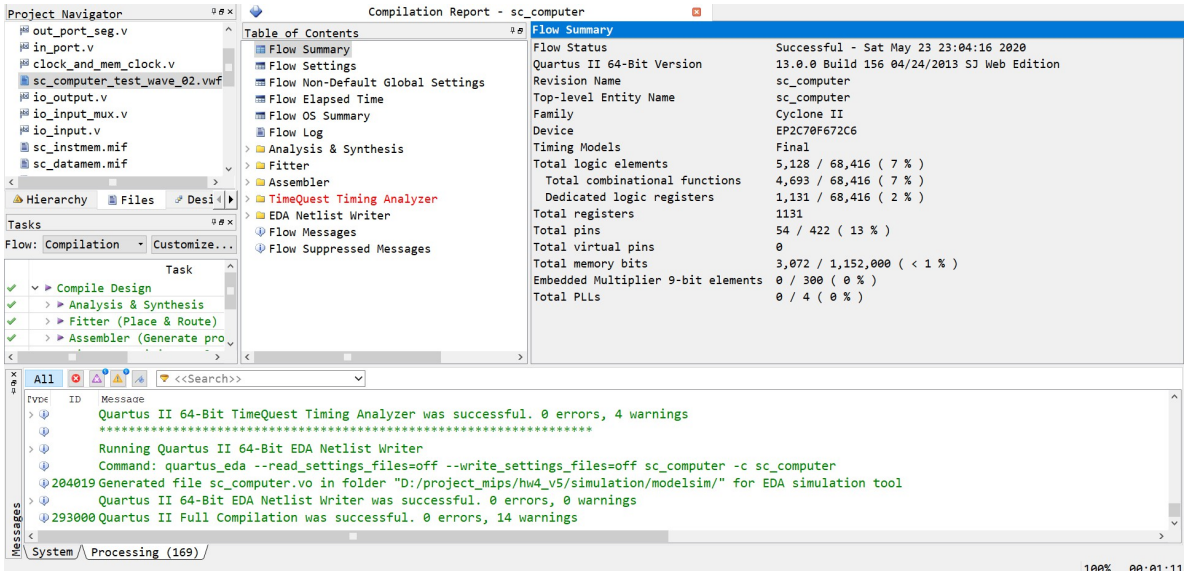
添加顶层模块后，整个项目的架构如图所示。



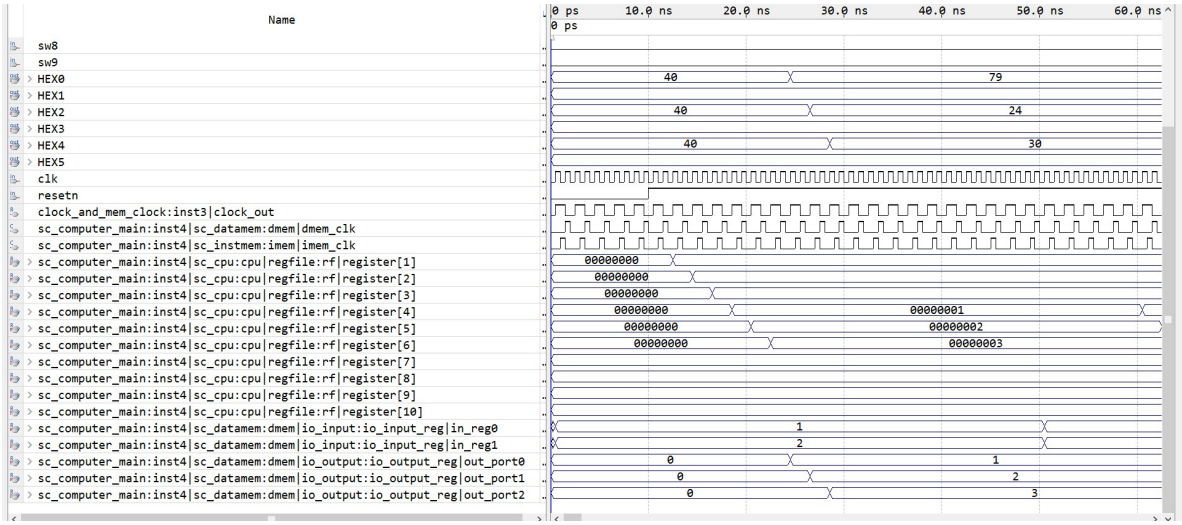
实验结果

本次实验的结果如下。

编译结果



波形仿真结果



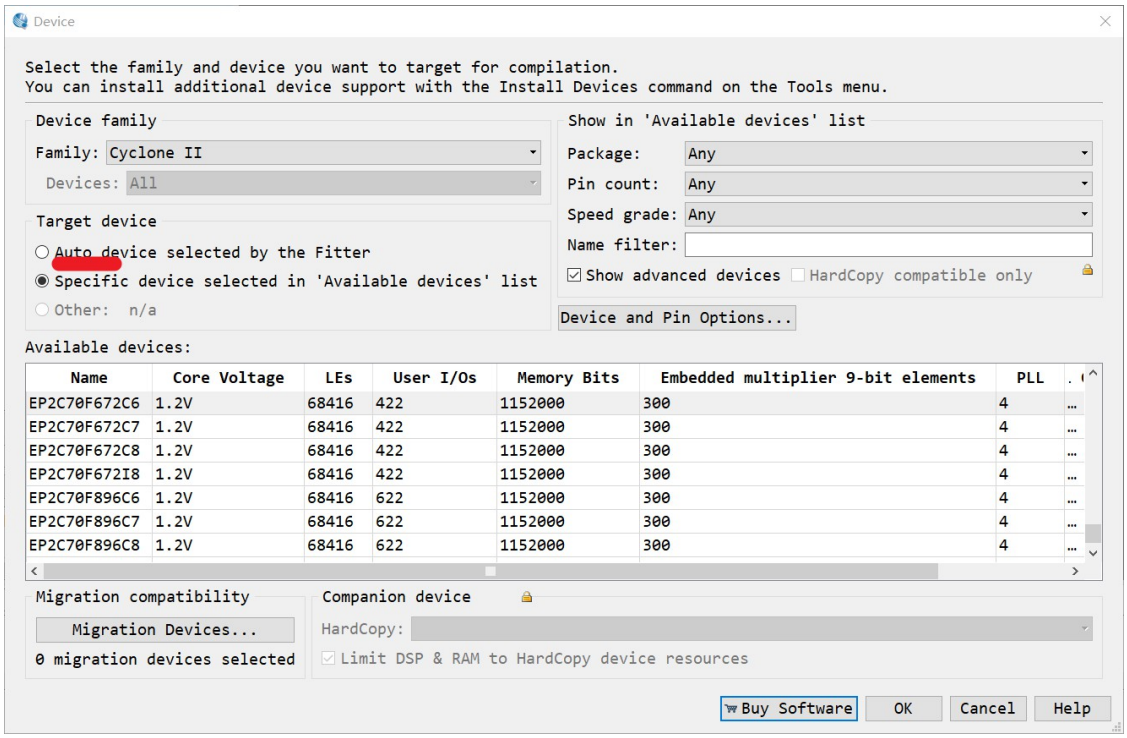
可以看出，波形仿真的结果和实验指导书中给出的相同。

实验问题汇总

1. 出现如下针脚不够的问题。

   ```
   Error (176205): Can't place 291 pins with 3.3-V LVTTL I/O standard because Fitter has only 153 such free pins available for general purpose I/O placement
   Error (176204): Can't place pins due to device constraints
   Info (171121): Fitter preparation operations ending: elapsed time is 00:00:02
   Error (171000): Can't fit design in device
   ```

   解决方法：换一个板子，或者直接在设备选项中选择 auto 选项，如图。

2. 波形仿真过程中，`in_reg` 的高 27 位始终是高阻态，但低 5 位有效。

   解决方法：可能是综合器做了一些优化，将这些线优化掉了。对此似乎没有什么好的解决方法。可以不显示这些高阻态的位，只显示真正有用的位。

小结

本次实验难度较大，是一个挑战。但同时也锻炼了我的硬件设计能力。

本次实验由于中途遇到了多个问题导致没有时间按照选做的要求实现一条新的指令，但我会在其他时间自行尝试。

参考资料和致谢

本次实验主要参考了实验参考书。

本次实验过程中受到了 IEEE 班级群中大量匿名好心人士的帮助，在此表示感谢。