计算机组成原理课程实验

实 验 报 告

张洋铭（09013322）
邱传飞（09013323）

2015 年 6 月

实验一 运算器组成实验

一、 实验目的

（1）熟悉加/减法器的功能及使用方法。
（2）掌握算术逻辑部件(ALU)的功能及其逻辑组成。
（3）加深对运算器工作原理的理解。

二、实验内容

（1）掌握 Quartus II 的使用方法，能够进行数字电路的设计及仿真。
（2）验证 Quartus II 所提供加/减法器的功能及使用方法。
（3）设计具有加法、减法、逻辑与、逻辑非 4 种功能的 ALU，并进行功能仿真/验证。

三、实验原理及方案设计

算术逻辑单元（ALU）是 CPU 的核心器件之一，它可以完成基本的算术运算和逻辑运算。本次实验设计并在 Quartus II 上实现了一个具有加法、减法、逻辑与、逻辑非 4 种功能的 8 位字长 ALU。

该 ALU 可接收两个 8 位整数 X 和 Y，进行四种运算。由于 ALU 是组合逻辑器件，因此运算结果可以立即输出到结果端 Z。

该 ALU 除可以完成 4 种运算操作之外，还可以保存并输出 4 个运算状态标志，分别是

——ZF：零标志
——CF：进位标志
——OF：溢出标志
——SF：符号标志

这四个标志暂存到标志寄存器中，通过四条信号线输出。

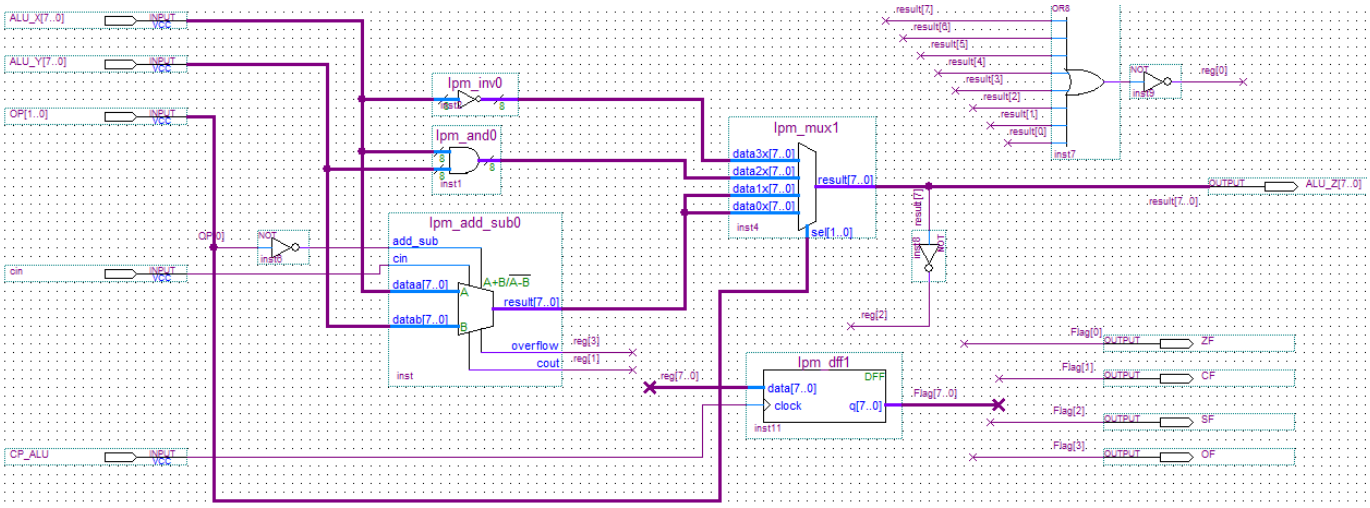
Quartus II 提供的 LPM 模块提供了 OF 和 CF 两个信号，直接输出即可；ZF 通过对 Z 各位进行“与”操作，若结果为 0，说明 Z 的各位均为 0，也即 Z 为 0。当运算结果 Z 以补码表示时，Z 的最高位 Z[7]即代表结果的符号，取反直接输出，即为符号标志 SF。

运算功能选择由两位宽度的功能选择线 OP[2]决定，功能表如下：

OP[1]	OP[0]	功能
0	0	X+Y（算术加法）
0	1	X-Y-1（算术减法）
1	0	X&Y（按位与）
1	1	~X（各位取反）

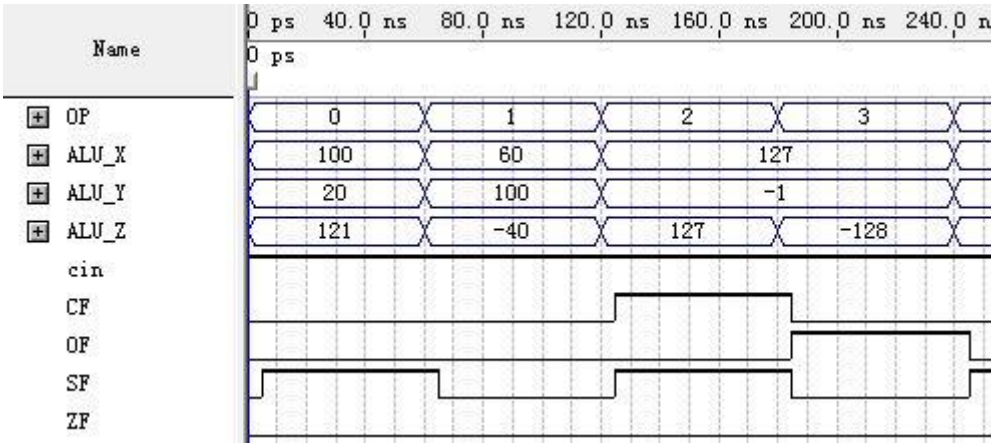
根据这个功能表，我们通过 OP[0]控制加减法器模块的加减选择端，并且采用四路数据选择器对结果进行选择输出，实现了如上表所示的功能选择。

根据以上设计思路，最终确定模块连接图如下。



四、实验结果分析

仿真波形图如下：



五、实验小结

首先，在设计实践中我们发现 Quartus II 是一款使用方便但难以掌握的数字电路设计工具，许多细节问题有待于进一步熟练掌握。工具自带大量封装好的功能模块，极大的方便了电路的设计工作。

第二，通过此次实验，我们深感之前学过的知识的重要性。借此次实验的机会，我们系统地复习了一遍关于数字电路和编码方式的知识，巩固了之前所学的知识，受益匪浅。

实验二 存储器组成实验

一、实验目的

（1）熟悉半导体存储器的存取方法。
（2）掌握存储器的扩展方法。
（3）掌握存储器与总线的连接方法。

二、实验内容

（1）验证 Quartus II 所提供半导体存储器的功能及使用方法。
（2）设计一个读/写端口分离的 256×8bit 的存储器，地址空间中前一半只读、后一半可读可写，并进行存取操作仿真/验证。
（3）将 256×8bit 的存储器连接到地址/数据复用的总线上，并进行存、取操作仿真/验证。

三、实验原理及方案设计

此次实验，按照实验内容要求，我们设计了一个由 DRAM 和 ROM 组成的 256×8bit 的存储器。该存储器读写端口分离，但我们将其连接到地址/数据复用的总线上，将地址输入和数据输入合二为一，通过分时复用的方式进行读写操作，减少了端口数量。

3.1 存储器的地址分配和连接方式

按照要求，地址空间中前一半只读，后一半可读可写，即前一半分配给 ROM，后一半分配给 DRAM。

存储器容量为 256B，则地址线需要 8 位宽度。ROM 和 DRAM 各有 128B 的存储空间，则二者各使用 7 位宽度的内部地址线。剩余一根地址线就作为每个存储器的片选线，通过这条线控制 ROM 和 DRAM 的片选使能，实现二者的统一编址。

为方便起见，我们选择地址线的最高位作为存储器片选，地址分配如下：

00H ~ 7FH ROM
80H ~ FFH DRAM

3.2 控制信号设置

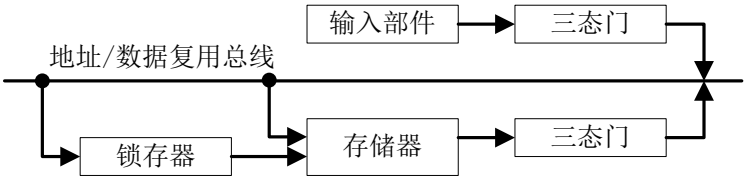
Quartus II 内置的 DRAM 模块提供了 wren 写使能信号,我们直接使用它作为 DRAM 的写使能信号;ROM 没有“写”的概念，因此 wren 只对 80H ~ FFH 的 DRAM 有效。

3.3 地址数据端口复用的实现方案

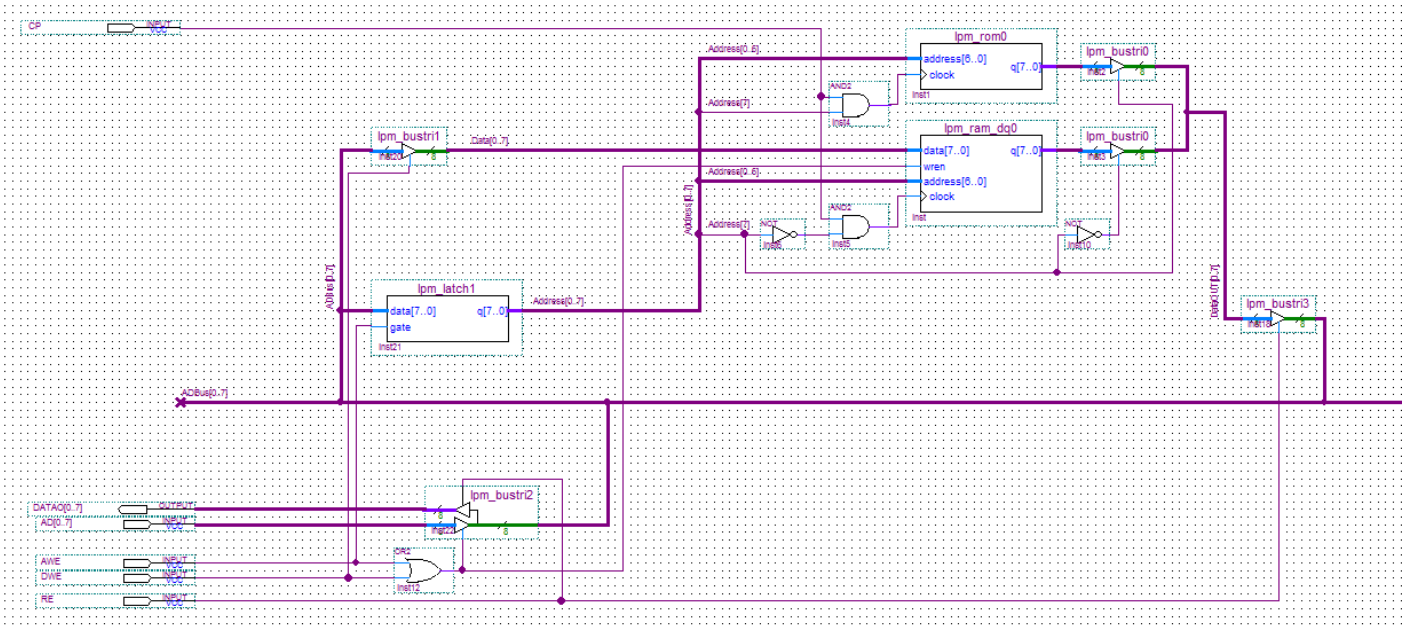
在之前的设计中，地址端口和数据输入端口是分离的，这样的设计极大的增加了信号线的数量，势必增加成品芯片的封装成本。因此，我们使用锁存器+三态门，实现了地址数据端口的分时复用。

对于半导体存储器，在数据读写期间，地址信号必须保持稳定不变。因此，我们在原先存储器的地址输入端设置了锁存器，保证操作时地址信号的稳定。同时，在输出端增设三态门，控制总线在每一时刻都只被一个设备所占用。

总体框图如下：

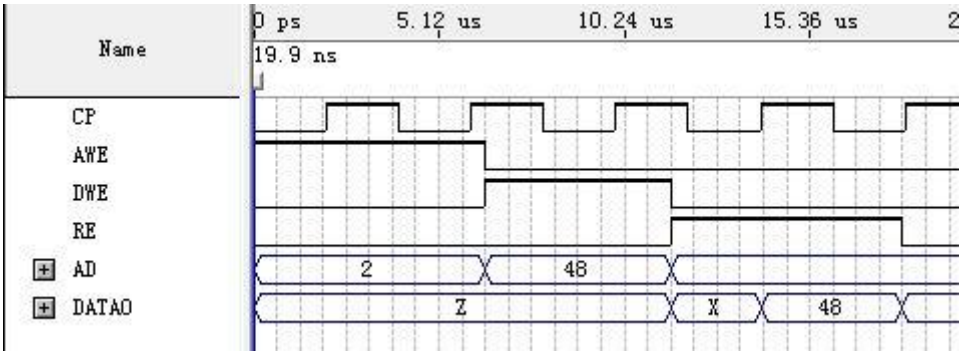


电路实现如下：



四、实验结果

仿真波形如下：



五、实验小结

此次实验，我们实现了数据/地址分时复用的总线，这种分时复用的设计思路对我们来说是一个全新的思路，值得认真思考和理解。通过这次实验，我们复习了存储器扩展的相关知识，并在本次试验中付诸实践，加深了理解和记忆，对于存储器的工作原理又多了一层认识。

实验三 寄存器组成实验

一、实验目的

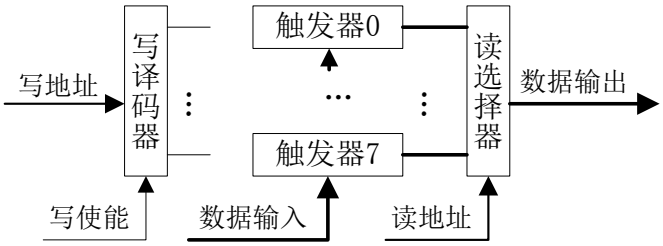
（1）熟悉 D 触发器的功能及使用方法。
（2）掌握寄存器文件的逻辑组成及使用方法。

二、实验内容

（1）验证 Quartus II 所提供 D 触发器的功能及使用方法。
（2）设计具有 1 个读端口、1 个写端口的寄存器文件，并进行存取操作仿真/验证。

三、实验原理与实现方案

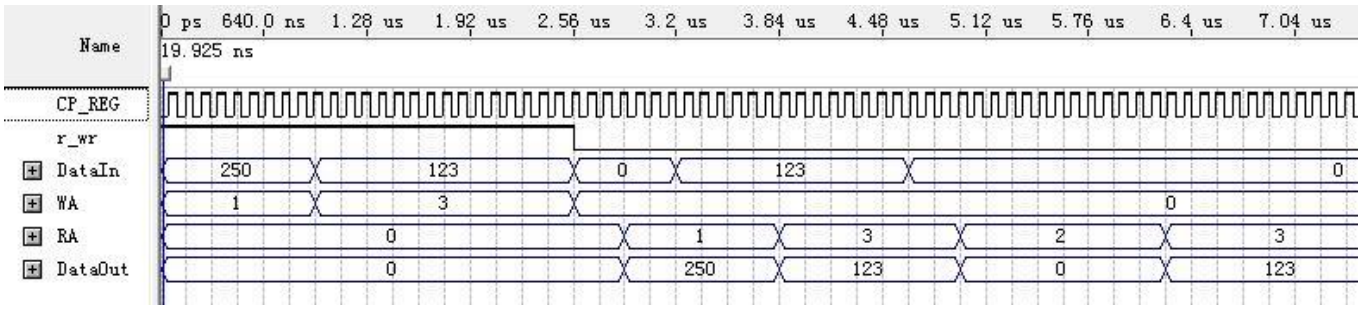
本次实验，我们利用 Quartus II 提供的八位 D 触发器模块，实现了一个 4×8bit 的寄存器组。该寄存器具备写使能信号 r_wr、读地址信号 RA、写地址信号 WA、数据输入 DataIn 和数据输出 DataOut 共 5 个对外端口。



Quartus II 提供的八位 D 触发器模块可存储 8 位二进制数。我们使用 4 个相同的 D 触发器模块，通过写地址译码器使能对应的寄存器，就可以实现写地址选择。

读取数据时，通过选择器，将对应的寄存器选通，即实现读地址选择。

四、实验结果



五、实验小结

通过此次实验，我们再次感受到 Quartus II 内置模块的强大。此次实验，我们将这个 4 字节寄存器封装成模块，方便后续实验的使用。

实验四 CPU 数据通路实验

一、实验目的

（1）掌握 CPU 数据通路的逻辑组成。
（2）了解指令功能的实现过程及其控制方法。

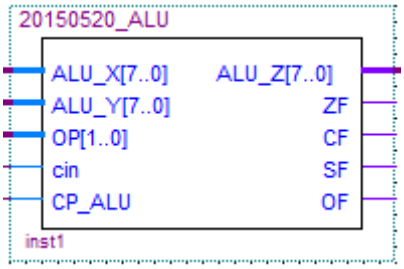
二、实验内容

（1）设计一个单总线结构的 CPU 数据通路，部件包括 4 种功能的 8 位 ALU、4×8 位的寄存器文件、256×8 位的 RAM、8 位计数器各一个。
（2）给出相关部件控制信号，分别实现取数、加法、条件转移指令的功能。

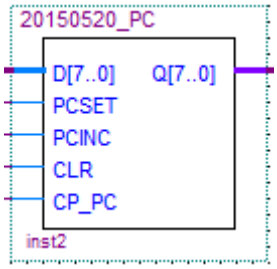
三、实验原理与实现方案

3.1 数据通路设计

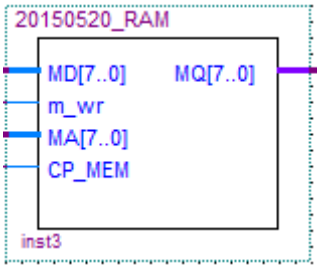
通过之前的三次实验，我们已经实现了算术逻辑单元、寄存器、存储器三大部件，其接口形式如下图：
八位算术逻辑单元：



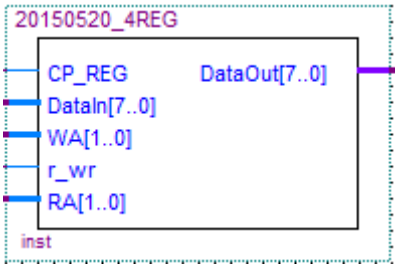
八位程序计数器：



256 字节 DRAM：

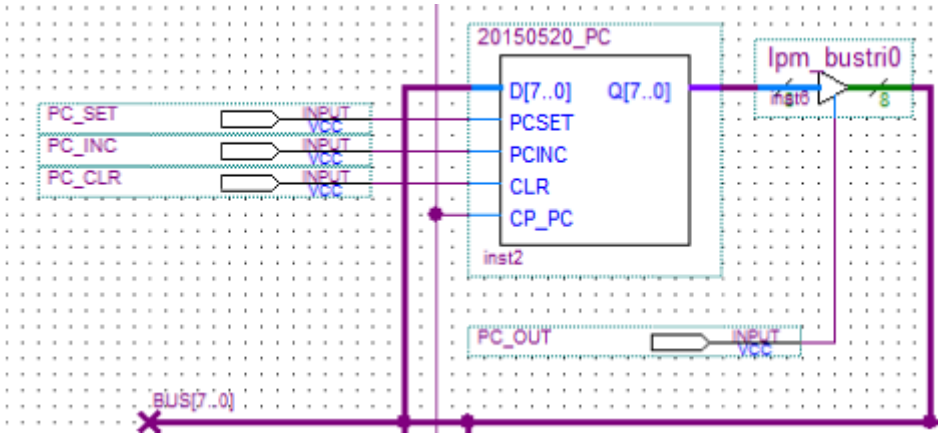


4 字节 8 位通用寄存器组：

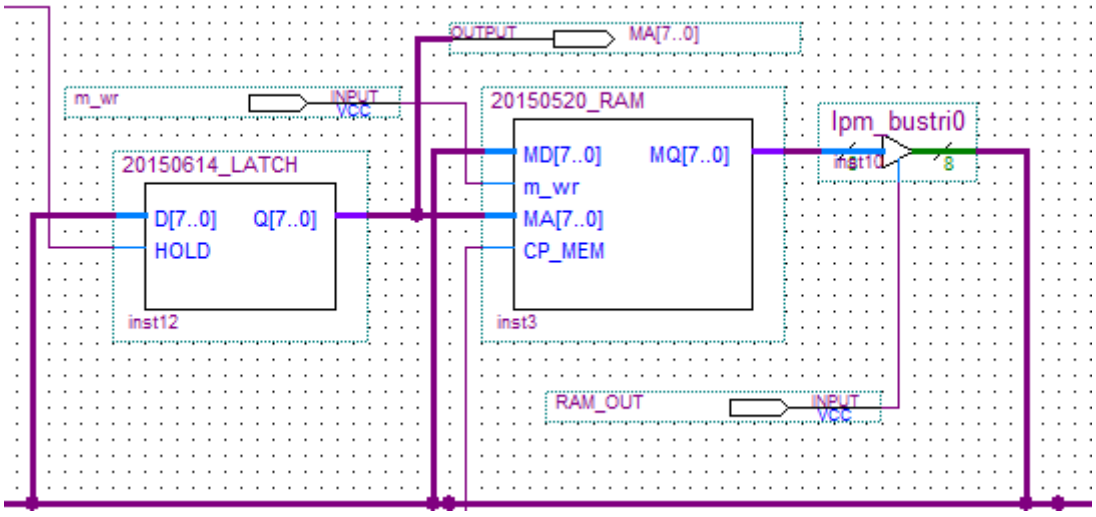


为实现单总线结构的 CPU 数据通路，我们将以上部件挂载到一个 8 位宽的总线上，每个部件的具体挂载方式如下。

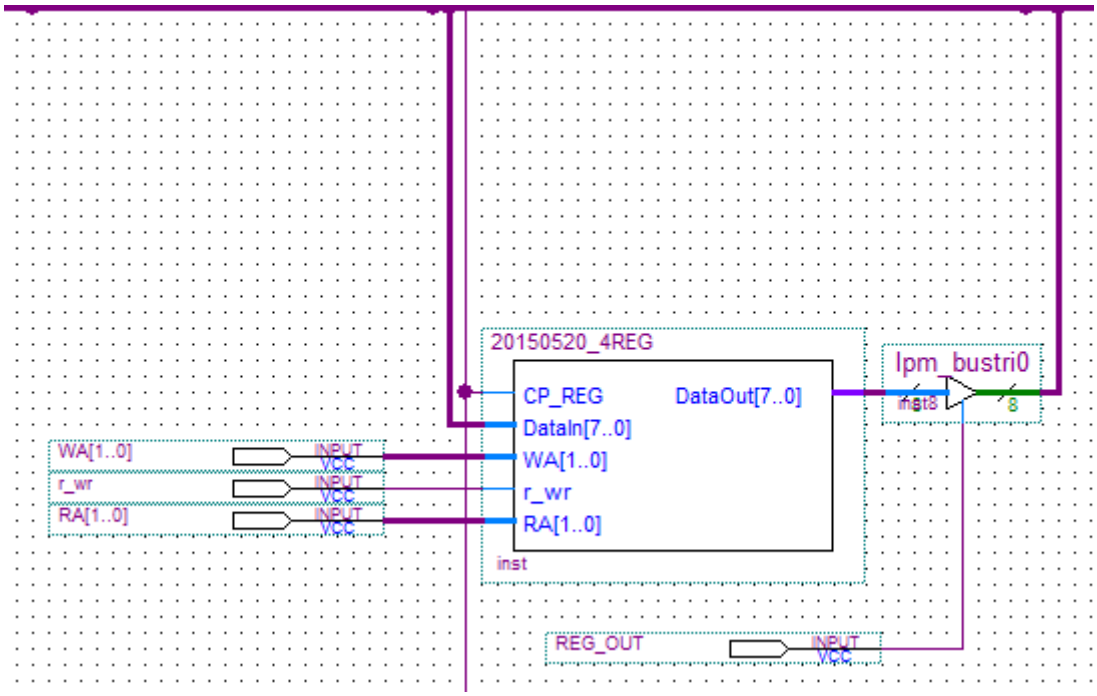
程序计数器 PC：在输出端增加三态门，实现 PC 与总线间的分时段隔离，避免信号之间的冲突。



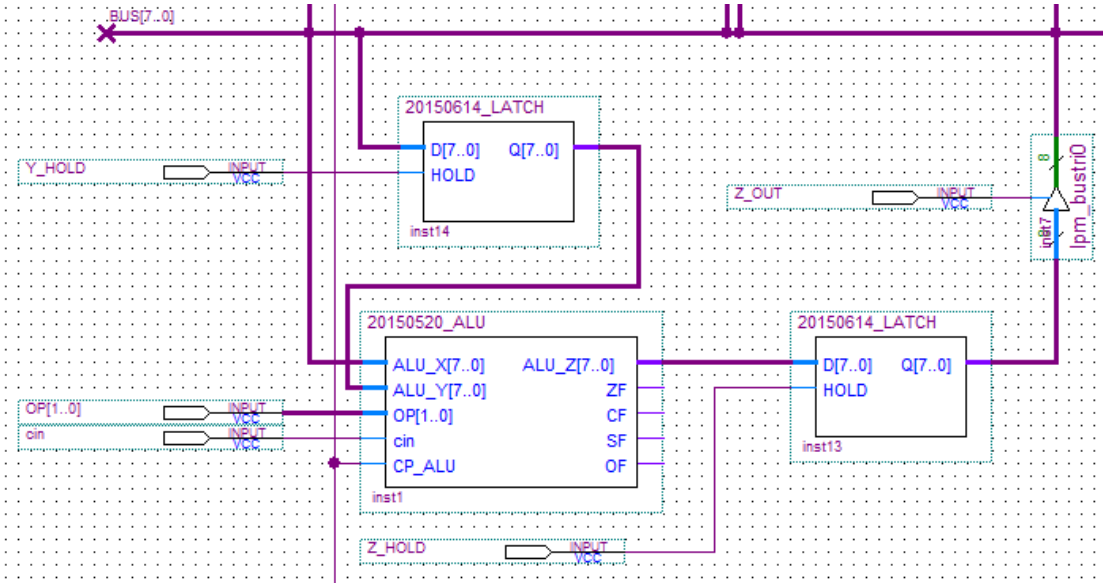
DRAM：除在输出端增加三态门以外，还在地址输入端增设地址锁存器，其目的是为保证数据存取时地址信号的稳定。



寄存器组：在输出端增加三态门。



ALU：由于 ALU 是组合逻辑器件，不能保存信息，因此要求输入信号保持稳定，为此，在数据输入端增加锁存器。ALU 的运算结果也需要暂存，因此输入端也设置了锁存器，通过三态门控制与总线的连接通断。



另外，外部数据输入接口也通过三态门挂载到总线上。

通过在各部件与总线之间加设三态门，就可以通过控制信号使得总线分时复用，实现 CPU 内部各部件之间的数据传输。

各部件的控制信号如下：

控制信号名称	功能描述
程序计数器控制信号	
PC_SET	PC 置数
PC_INC	PC 自增
算术逻辑单元控制信号	
Y_HOLD	操作数锁存器 锁存
Z_HOLD	运算结果锁存器 锁存
OP[2]	ALU 操作类型
寄存器组控制信号	
WA[2]	寄存器写地址
RA[2]	寄存器读地址
r_wr	寄存器写使能
DRAM 控制信号	
RAM_ADDR_HOLD	地址锁存器 锁存
三态门控制信号（以下信号同时至多一个有效）	
PC_OUT	程序计数器输出
Z_OUT	运算结果输出
REG_OUT	寄存器输出
RAM_OUT	RAM 输出
BUS_IN	外部输入允许

3.2 指令微操作序列设计

本次实验要求实现以下三条共五种指令：

取数 LD	RD←[(RS)]
加法 ADD	RD←(RD)+(RS)
加法 ADD	RD←(RD)+[(RS)]
条件转移 JZ（ZF=1）	PC←(PC)+disp
条件转移 JZ（ZF=0）	PC←(PC)+1

根据每种指令的执行过程，分别设计微操作序列如下：

（注：对于控制信号，不加标记代表置高有效，加下划线代表置低无效。）

取数 LD ： RD←[(RS)]

1）RS→RA，REG_OUT，RAM_ADDR_HOLD，REG_OUT
2）RD→WA，r_wr，RAM_OUT
3）RAM_OUT，RAM_ADDR_HOLD，结束

加法 ADD：RD←(RD)+[(RS)]

1）RD→RA，REG_OUT，Y_HOLD，REG_OUT
2）RS→RA，REG_OUT，RAM_ADDR_HOLD，REG_OUT
3）RAM_OUT，ADD→OP，Z_HOLD，RAM_OUT，RAM_ADDR_HOLD，Y_HOLD
4）RD→WA，r_wr，Z_OUT
5）Z_OUT，结束

加法 ADD：RD←(RD)+ (RS)

1）RD→RA，REG_OUT，Y_HOLD，REG_OUT
2）RS→RA，REG_OUT，ADD→OP，Z_HOLD，REG_OUT，Y_HOLD
3）RD→WA，r_wr，Z_OUT
4）Z_OUT，结束

条件转移 JZ（ZF=1）：PC←(PC)+disp

1）PC_OUT，Y_HOLD，PC_OUT
2）BUS_IN，ADD→OP，Z_HOLD，BUS_IN，Y_HOLD
3）Z_OUT，PC_SET，Z_OUT，结束

条件转移 JZ（ZF=0）：PC←(PC)+1

1）PC_INS

据此，即可设计波形并仿真。

四、实验结果

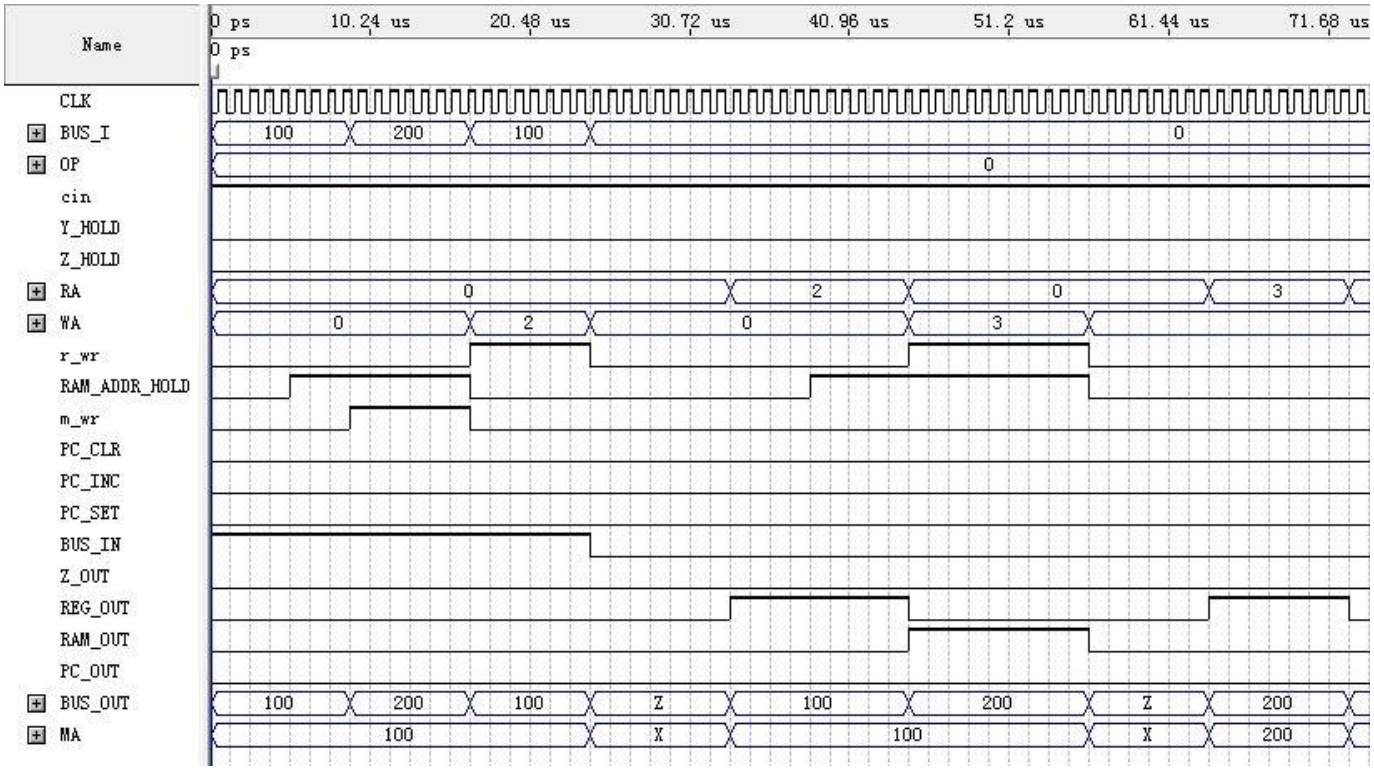
在五条指令的测试中，均采取如下假定：

寄存器地址	寄存器数值
2H	100
3H	30

DRAM 地址	DRAM 数值
100	200

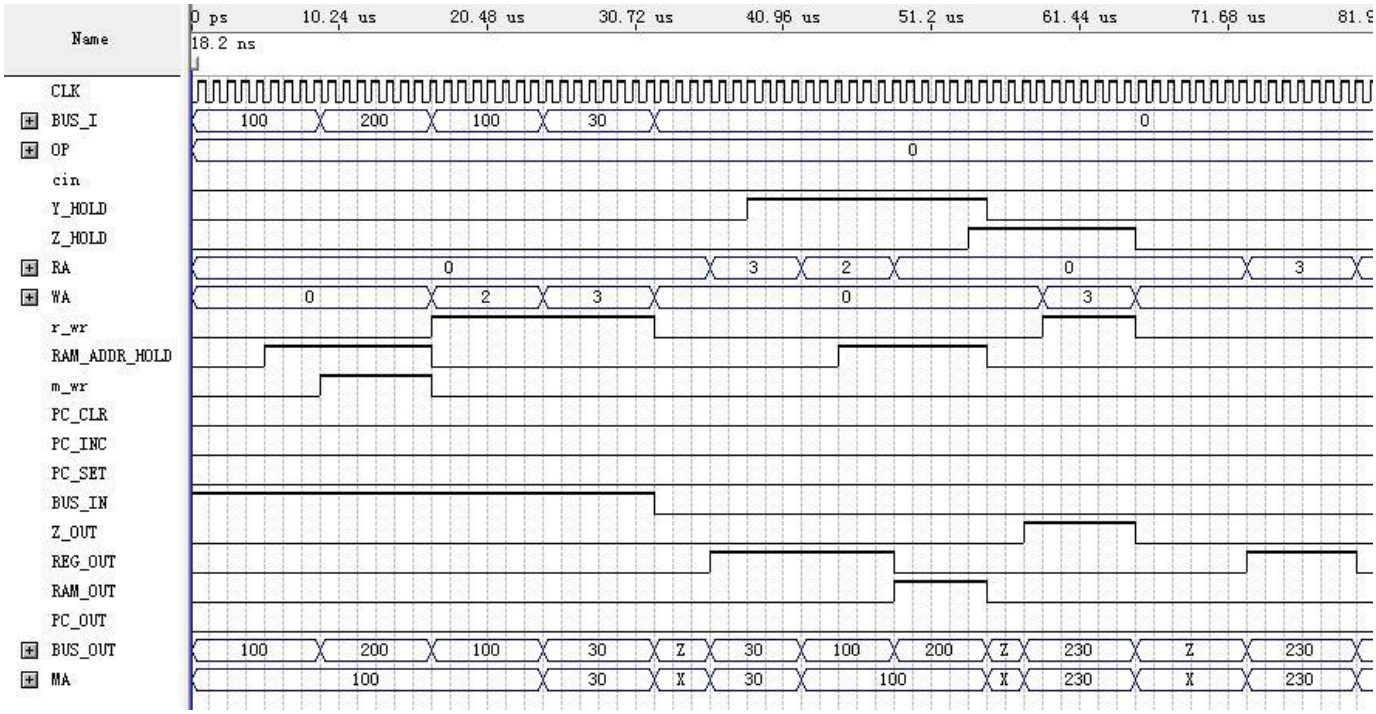
4.1 取数指令

设 RS =2H，RD=3H，执行结果应该是(RD)=200。



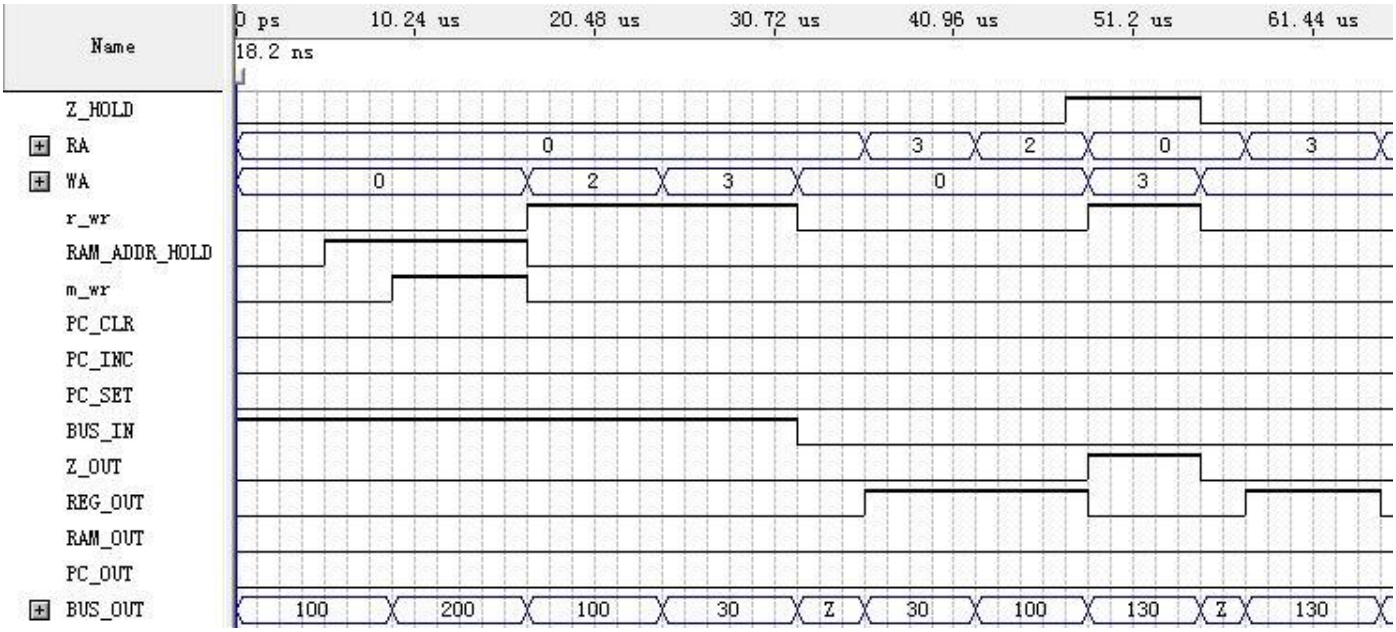
4.2 加法指令 RD←(RD)+[(RS)]

设 RS =2H，RD=3H，执行结果应该是(RD)=200+30=230。



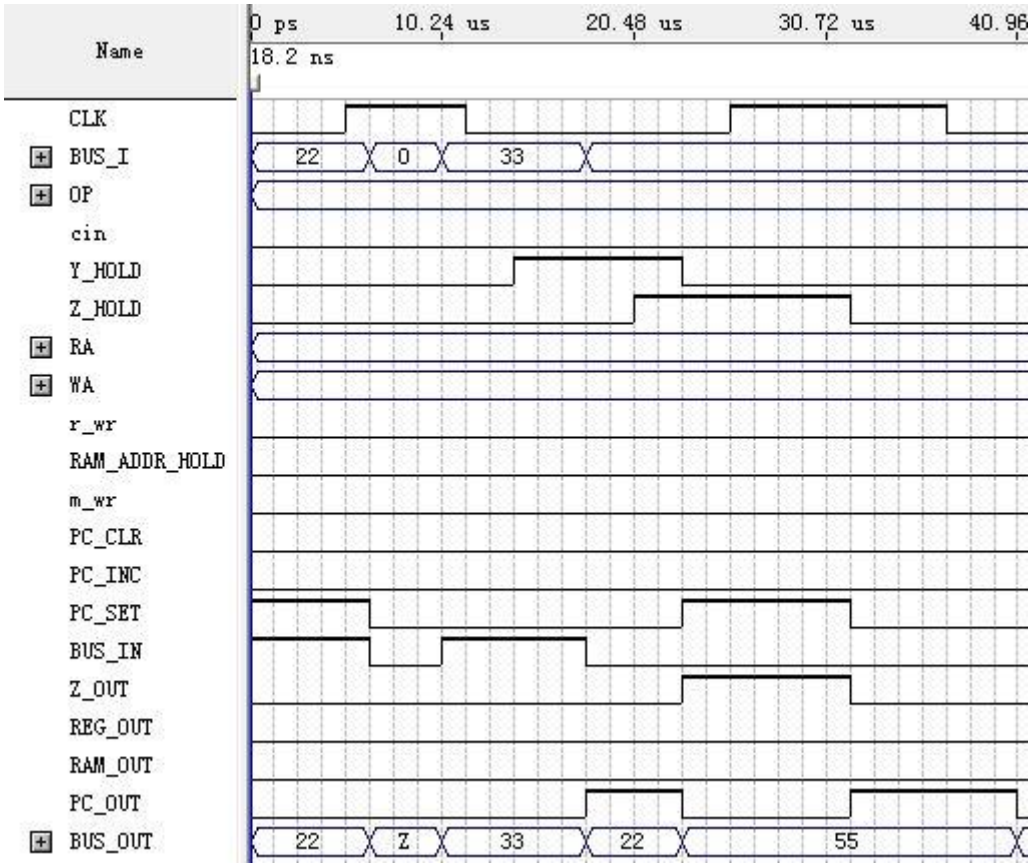
4.3 加法指令 RD←(RD)+ (RS)

设 RS =2H，RD=3H，执行结果应该是(RD)=100+30=130。



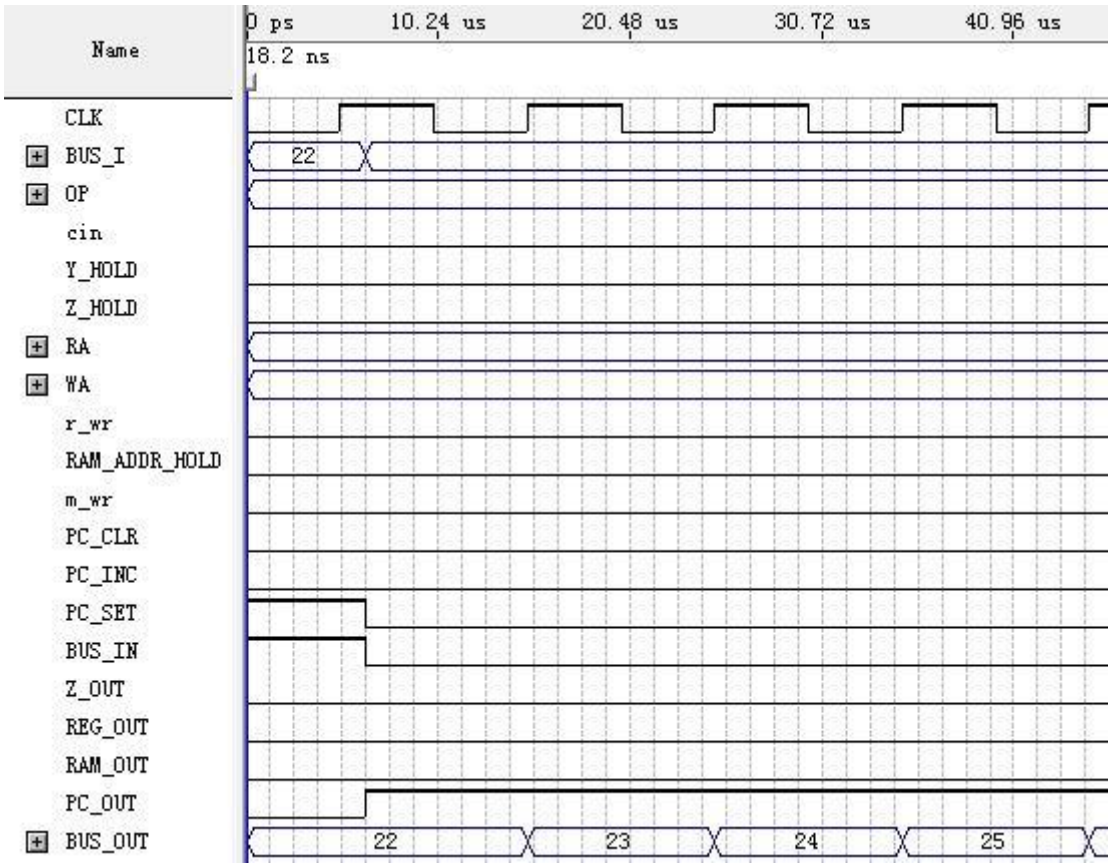
4.4 条件转移（ZF=1）：PC←(PC)+disp

设 PC =22，disp=33，执行结果应该是 PC=22+33=55。



4.5 条件转移（ZF=0）：PC←(PC)+1

PC 值随机器运行自动增加。



四、实验小结

作为计算机专业的学生，弄清楚计算机的工作原理十本必要。本次试验，我们实现了一个简单的单总线 CPU 数据通路，并且在人工给出控制信号时序的情况下，成功实现了五条指令的执行。原本打算在实验要求之外实现一个包含指令寄存器 IR 的硬布线控制器电路，并且实现更多指令功能，由于时间和精力所限，并没有在实验要求的时限内完成。这个设想有待于以后逐步实现。

总而言之，通过软件仿真的方式实现 CPU 内部数据通路，使我们对于计算机的工作原理有了极其深刻的认识和实践经验。对于提高我们在计算机技术方面的整体素养，我觉得应该有极大的帮助。