湖南大學

HUNAN UNIVERSITY

计算机系统设计实验

学生姓名 孟祥炜

学生学号 201607020301

专业班级 智能 1602

实验名称 相对简单的 CPU 设计

完成 日期 2019.1.4

一、实验题目

设计并实现相对简单的 CPU

二、实验要求

相对简单 CPU 的规格说明

1. **64K 字节的存储器，每个存储单元 8 位宽。**

   地址引脚 A[15..0]

   数据引脚 D[7..0]

2. **CPU 的内部寄存器**

   ◆ 8 位累加器 AC：

   接受任何算术或者逻辑运算的结果，并为使用两个操作数的算术或者逻辑操作指令提供一个操作数。

   ◆ 寄存器 R：

   一个 8 位通用寄存器。它为所有的双操作数算术和逻辑运算指令提供第二个操作数。它也可以用来暂时存放累加器马上要用到的数据。（减少存储器访问次数提高 CPU 的性能）

   ◆ 零标志位 Z：

   每次执行算术运算或者逻辑运算的时候，它都将被置位。

   ◆ 16 位的地址寄存器 AR：

   通过引脚 A[15..0]向存储器提供地址。

   ◆ 16 位的程序计数器 PC：

   存放的是将要执行的下一条指令的地址，或者指令需要的下一个操作数的地址。

   ◆ 8 位的数据寄存器 DR：

   通过 D[7..0]从存储器中接收指令和数据并且向存储器传送数据。

   ◆ 8 位的指令寄存器 IR：

   存放的是从存储器中取出来的操作码。

   ◆ 8 位的临时寄存器 TR：

   在指令执行过程中，临时存储数据。（程序员不能访问）

3. **指令集架构**

指令	指令码	操作
NOP	0000 0000	无
LDAC	0000 0001 Γ	AC←M[Γ]
STAC	0000 0010 Γ	M[Γ]←AC
MVAC	0000 0011	R←AC
MOVR	0000 0100	AC←R
JUMP	0000 0101 Γ	GOTO Γ
JMPZ	0000 0110 Γ	IF（Z＝1） THEN GOTO Γ
JPNZ	0000 0111 Γ	IF（Z＝0） THEN GOTO Γ

ADD	0000 1000	AC←AC＋R，IF（AC＋R＝0） THEN Z←1 ELSE Z←0
SUB	0000 1001	AC←AC－R，IF（AC－R＝0） THEN Z←1 ELSE Z←0
INAC	0000 1010	AC←AC＋1，IF（AC＋1＝0） THEN Z←1 ELSE Z←0
CLAC	0000 1011	AC←0，Z←1
AND	0000 1100	AC←AC∧R，IF（AC∧R＝0） THEN Z←1 ELSE Z←0
OR	0000 1101	AC←AC∨R，IF（AC∨R＝0） THEN Z←1 ELSE Z←0
XOR	0000 1110	AC←AC⊕R，IF（AC⊕R＝0） THEN Z←1 ELSE Z←0
NOT	0000 1111	AC←AC'，IF（AC'＝0） THEN Z←1 ELSE Z←0

三、CPU 设计的过程

1. 指令执行过程分析

按照所给的要求，分析指令的执行过程

对于一条指令，分为取指令、指令译码、指令执行三个过程

对于取指令和指令译码阶段，对于所有指令都是相同的，使用 RTL 语言进行描述：

FETCH1： AR←PC

FETCH2： DR←M，PC←PC＋1

FETCH3： IR←DR，AR←PC

接下来是指令执行阶段，对于不同的指令，指令执行阶段不同。

a) NOP 指令

NOP1:（无操作）

b) LDAC 指令

LDAC 是一条多字指令。

它包含三个字：操作码 地址的低半部分 地址的高半部分

功能：从存储器中获得地址，然后从存储器中获得数据，并把数据装载到累加器中。

第一个状态：

LDAC1: DR←M，PC←PC＋1，AR←AR＋1

第二个状态：

LDAC2: TR←DR，DR←M，PC←PC＋1

LDAC3: AR←DR，TR

LDAC4: DR←M

LDAC5: AC←DR

c) STAC 指令

STAC 指令执行的是与 LDAC 完全相反的操作。

STAC1: DR←M，PC←PC＋1，AR←AR＋1

STAC2: TR←DR，DR←M，PC←PC＋1

STAC3: AR←DR，TR
STAC4: DR←AC
STAC5: M←DR

d) MVAC 和 MOVR 指令

MVAC1: R←AC
MOVR1: AC←R

e) JUMP 指令

JUMP1: DR←M，AR←AR+1
JUMP2: TR←DR，DR←M
JUMP3: PC←DR，TR

f) JMPZ 和 JPNZ 指令

JMPZ 指令的状态：

JMPZY1: DR←M，AR←AR＋1
JMPZY2: TR←DR，DR←M
JMPZY3: PC←DR，TR
JMPZN1: PC←PC＋1
JMPZN2: PC←PC＋1

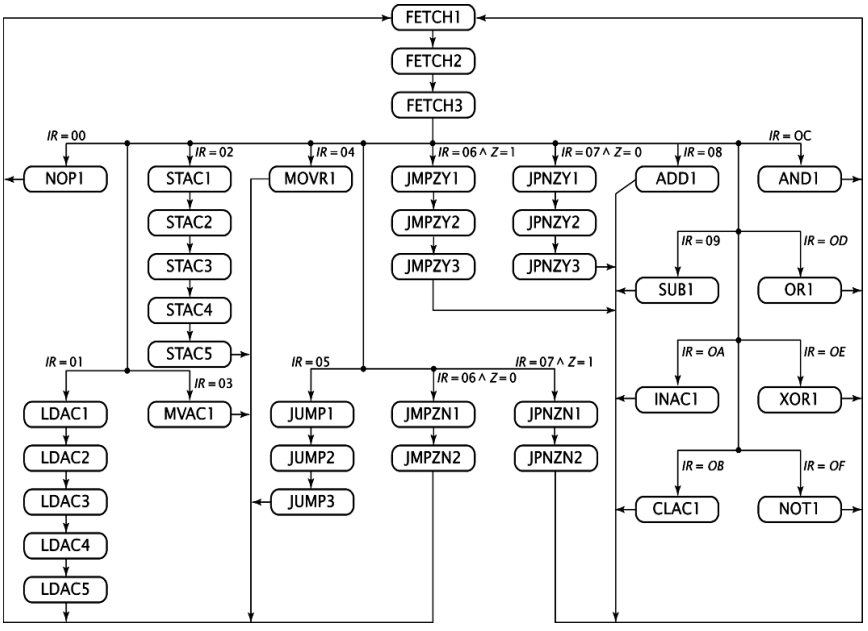
JPNZ 指令的状态：

JPNZY1: DR←M，AR←AR＋1
JPNZY2: TR←DR，DR←M
JPNZY3: PC←DR，TR
JPNZN1: PC←PC＋1
JPNZN2: PC←PC＋1

g) 其他指令

ADD1: AC←AC＋R，IF（AC＋R＝0）THEN Z←1 ELSE Z←0
SUB1: AC←AC－R，IF（AC－R＝0）THEN Z←1 ELSE Z←0
INAC1: AC←AC＋1，IF（AC＋1＝0）THEN Z←1 ELSE Z←0
CLAC1: AC←0, Z←1
AND1: AC←AC∧R，IF（AC∧R＝0）THEN Z←1 ELSE Z←0
OR1: AC←AC∨R，IF（AC∨R＝0）THEN Z←1 ELSE Z←0
XOR1: AC←AC⊕R，IF（AC⊕R＝0）THEN Z←1 ELSE Z←0
NOT1: AC←AC'，IF（AC'＝0）THEN Z←1 ELSE Z←0

根据以上逻辑画出 CPU 的状态图



2. 建立数据通路

实现数据通路的两种方式：硬布线逻辑，微程序控制

硬布线逻辑

时序逻辑和组合逻辑产生控制信号

微程序控制

使用存储器查找表方式来输出控制信号

数据通路的实现逻辑：在这里使用硬布线逻辑实现相对的简单的 CPU

数据通路的两种连接方式：总线、直连

在所有需要传送数据的部件之间创建一条数据通路

直连方式

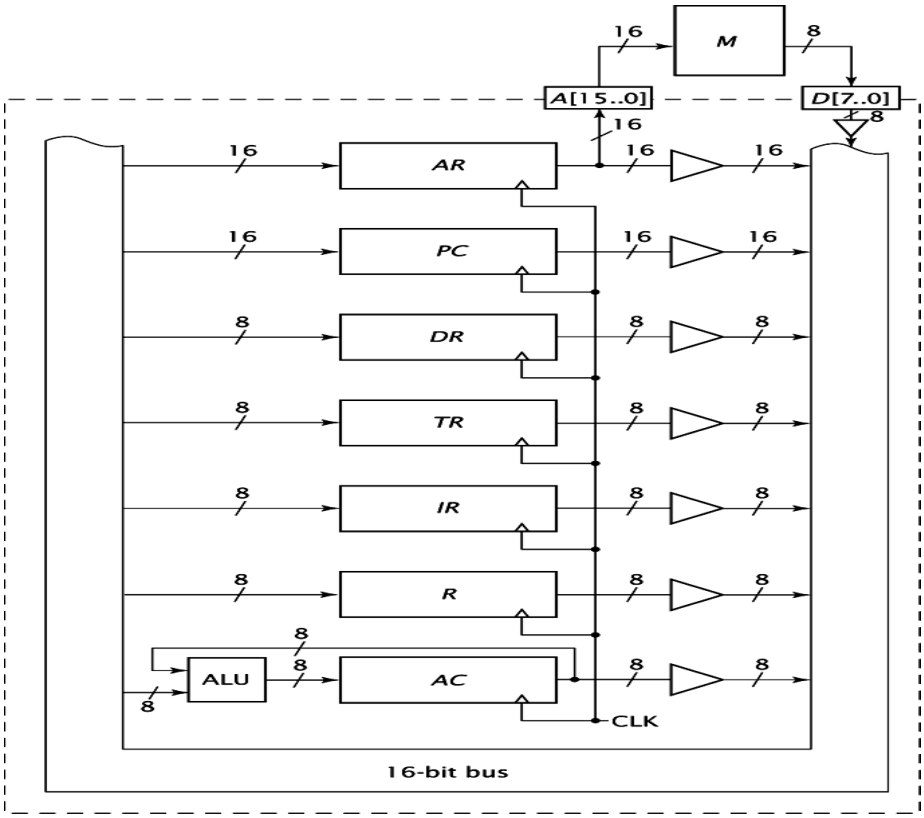
在所有需要传送数据的部件之间创建一条数据通路

总线方式

在 CPU 内部创建一条总线，各个部件之间使用总线传递数据

数据通路的连接方式：在这里使用总线的方式连接数据通路

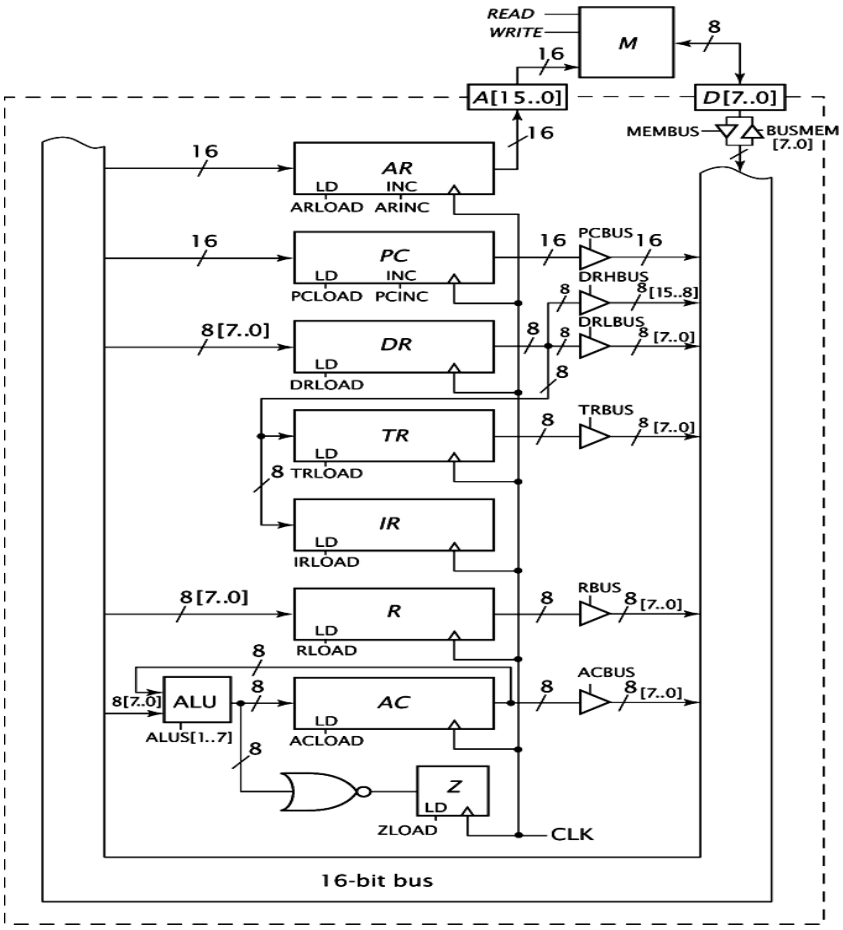
根据状态图首先创建基础的数据通路



通过将不同的操作分组整合、添加/删除 必要/不必要 的部件 以及逻辑优化

同时对运算器 ALU 进行设计与优化

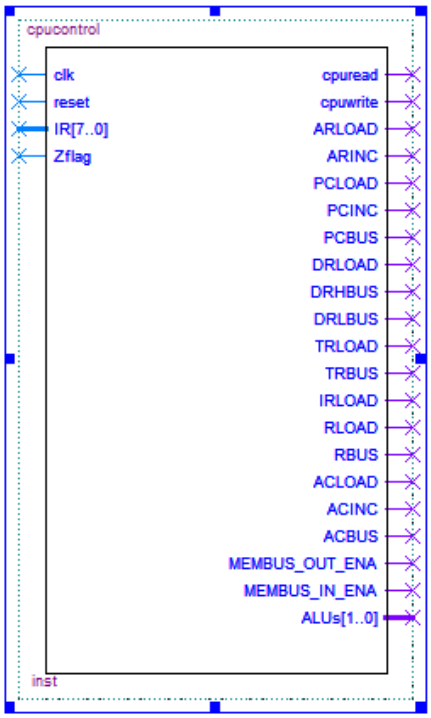
得到经过优化后的数据通路，如下



本实验即基于该数据通路完成

3. 根据数据通路和指令执行过程设计控制逻辑单元

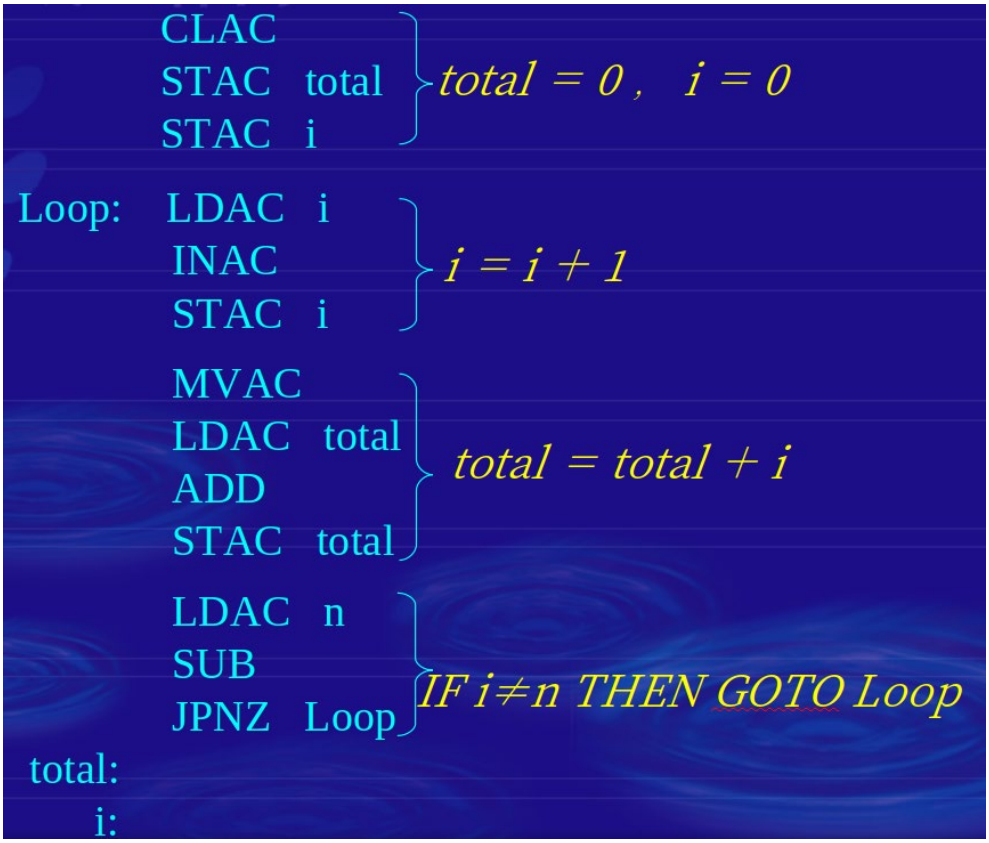
下一状态的控制逻辑有当前指令执行到的状态和当前数据通路中的数据决定，因此可以得到控制单元逻辑



4.设计并验证 CPU

对于 CPU 设计要求中的指令，设计一段指令并运行，验证各指令是否正常、正确的执行

在这里验证一下指令段（这里设置 n=6）



四、实验内容

根据实验要求，在 EDA 软件中设计并验证相对简单的 CPU 的实现

采用 VHDL+原理图的方式，各个子模块（内部寄存器、控制单元）采用 VHDL 语言实现，顶层设计采用原理图的方式将各个模块按照数据通路连接在一起。

子模块的代码详见 Sub-Modules 文件夹

注：BUStri8 和 BUStri16 为 8 输入/输出和 16 输入/输出的三态门

顶层设计的原理/逻辑图详见 Top-level design 文件夹

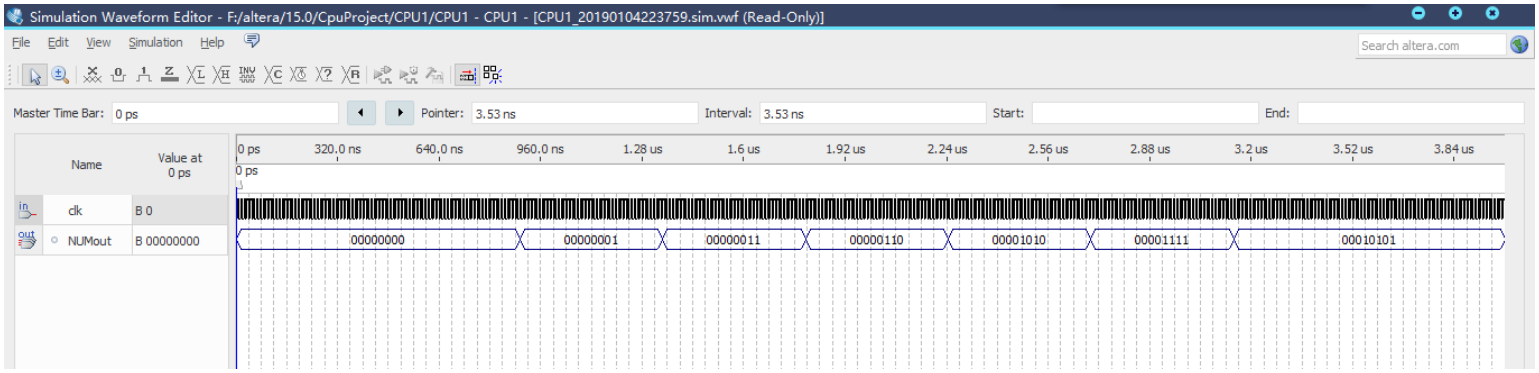
五、实验结果

观察 total 值随着 clk 的变化

设置一个寄存器/触发器，当输入 lpm_ram_dp 的地址为 total 所在的地址，且为读操作的时候，同时将 total 值更新到这个寄存器/触发器中，并设置输出引脚。

为了便于观察，设置 clk 和这个寄存器的输出引脚到波形图文件中

波形图



根据仿真波形图，可知所设计的 CPU 正确实现了待验证代码段的功能

六、实验中的问题

实验中碰到的主要问题就是存储指令的器件的调试，刚开始使用库器件中的 lpm_ram_io，但是经过对比器件帮助文件和多次调试都无法正确的实现逻辑，后来改用了器件库中的 lpm_ram_dp，经过调试验证可以正常使用，因此使用这个器件作为存储指令的 RAM(功能类似于实际计算机中的内存)