目录

[一、实验目的 1](#_Toc137778365)

[二、 设计要求 1](#_Toc137778366)

[(1)模型机的设计要求 1](#_Toc137778367)

[(2)模型机的逻辑框图 1](#_Toc137778368)

[三、实验原理 2](#_Toc137778369)

[8位CPU结构： 3](#_Toc137778370)

[8位CPU组成： 4](#_Toc137778371)

[8位CPU的顶层设计电路原理图 5](#_Toc137778372)

[四、实验准备 5](#_Toc137778373)

[1.设计内容 5](#_Toc137778374)

[2.元件接口 6](#_Toc137778375)

[accumulator累加器 6](#_Toc137778376)

[ALU运算器 6](#_Toc137778377)

[controller硬布线控制器 7](#_Toc137778378)

[instruction\_register寄存器 7](#_Toc137778379)

[program\_counter程序计数器 7](#_Toc137778380)

[RAM静态储存器 8](#_Toc137778381)

[top\_level由六大模块组成的模型机 8](#_Toc137778382)

[五、实验过程 9](#_Toc137778383)

[1、8位CPU电路图 9](#_Toc137778384)

[2、ram中测试程序说明 9](#_Toc137778385)

[3、测试程序，波形图及指令的对应关系分析，数据运算过程分析 10](#_Toc137778386)

# 一、实验目的

1、融会贯通本课程各章节的内容，通过知识的综合运用，加深对计算机系统各功能部件的工作原理及相互联系的认识，加深计算机工作中“时间一空间”概念的理解，从而清晰地建立计算机的整机概念

2、学习设计和调试计算机的基本步骤和方法，提高使用Quartus等软件仿真工具和集成电路的基本技能

3、培养科学研究的独立工作能力，取得工程设计与组装调试的实践和经验

4、设计一台硬布线控制器的8位模型计算机，用VHDL语言完成设计并调试成功

# 设计要求

设计一台硬布线控制器的8位模型计算机，用VHDL语言完成设计并调试成功

## (1)模型机的设计要求

|  |  |
| --- | --- |
| 总线 | 单总线结构 |
| 数据线、地址线 | 8位 |
| 指令系统 | 寻址方式 2种（立即数寻址、直接寻址）  类型 5种（算术运算、逻辑运算、数据传送、程序控制）  指令 12条(add、sub、mul、div、neg、and 、not、 or、load、store、branch、halt) |
| 运算器 | 单累加器结构, 8种运算 |
| 控制器 | 硬布线，微控制信号根据需要确定 |

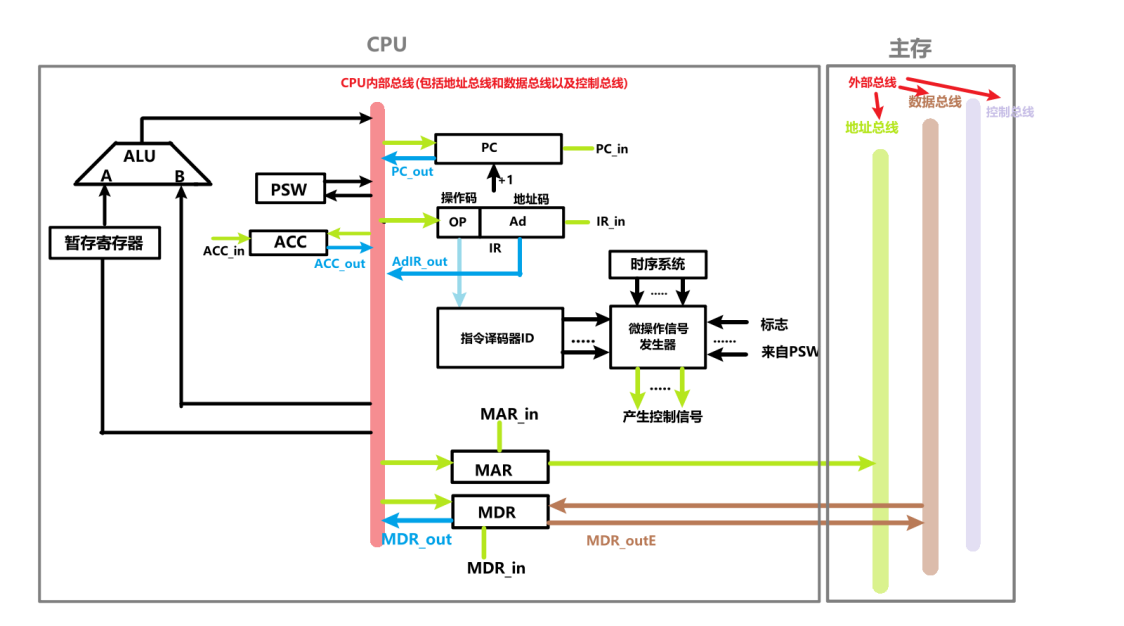
(2)模型机的逻辑框图



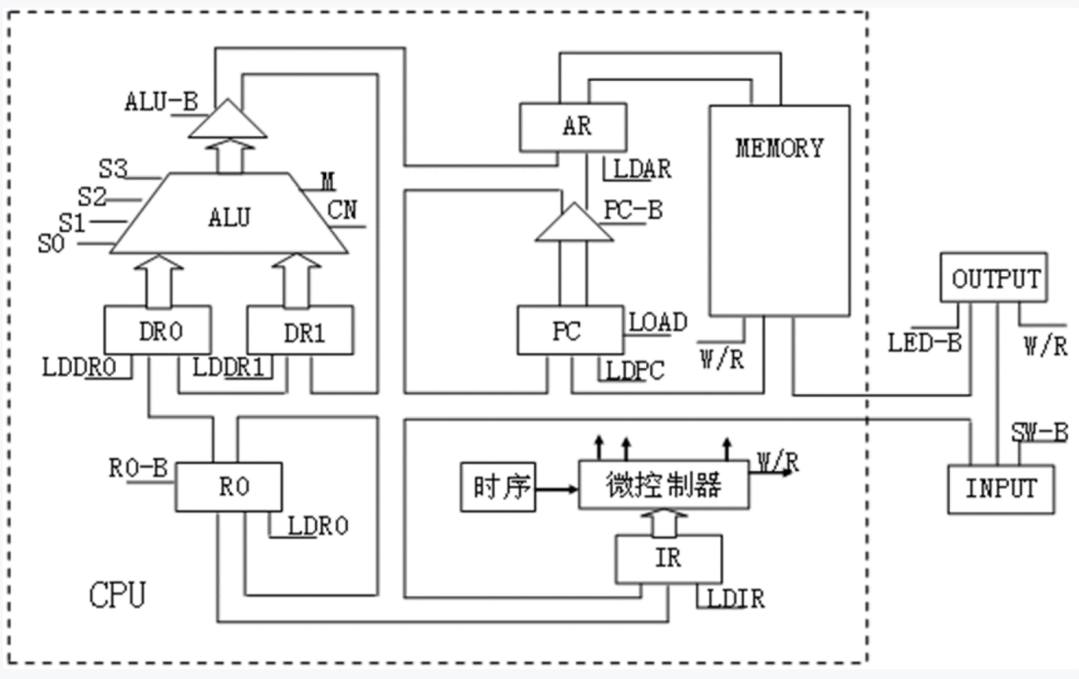
# 三、实验原理

首先，将PC寄存器里的指令传送到IR指令寄存器中，同时PC自增1；其次，以控制器为中心，控制器从指令寄存器取得指令，编译指令，再输出微控制信号，控制ALU的运算，并且使用MAR寄存器中的值借助地址总线访问主存中相应地址位下的数据，将数据取出通过数据总线放置到MDR寄存器中，再由MDR寄存器将数据通过CPU内部总线送至B，和暂存寄存器中的数据进行运算，运算后再把结果通过CPU内部总线 存到MDR中，结合MAR在内存中指向的地址，将

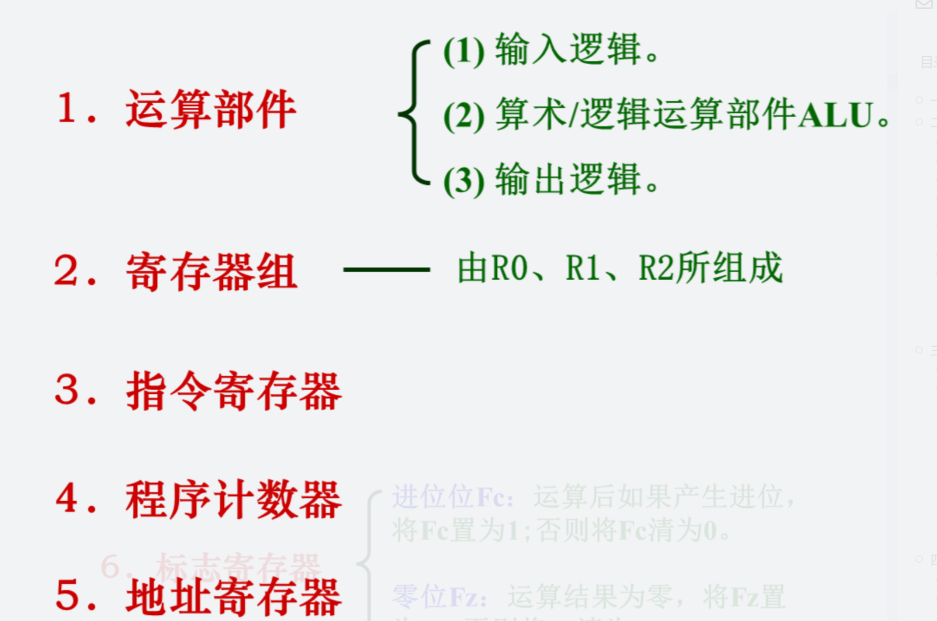
MDR中的数据通过数据总线送至相应物理位置下的存储单元中。指令寄存器继续执行下一条指令，如此循环。

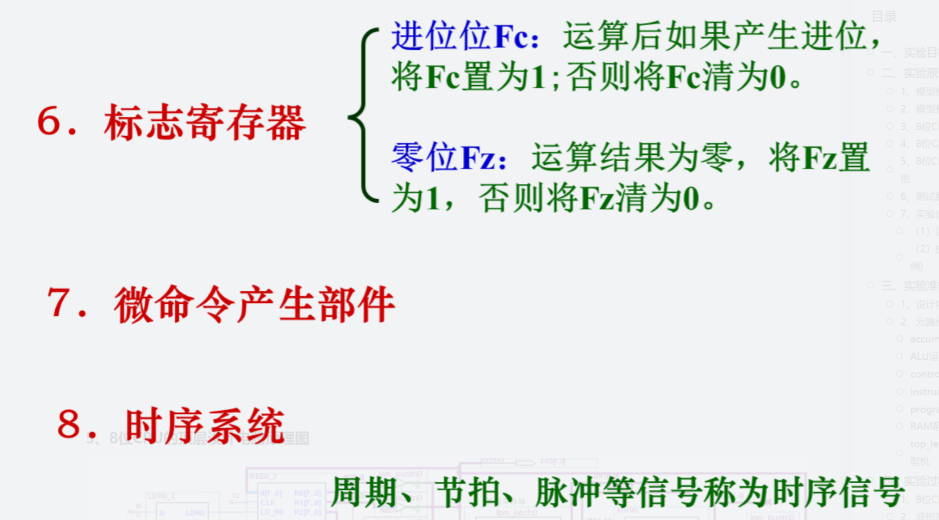


## 8位CPU结构：

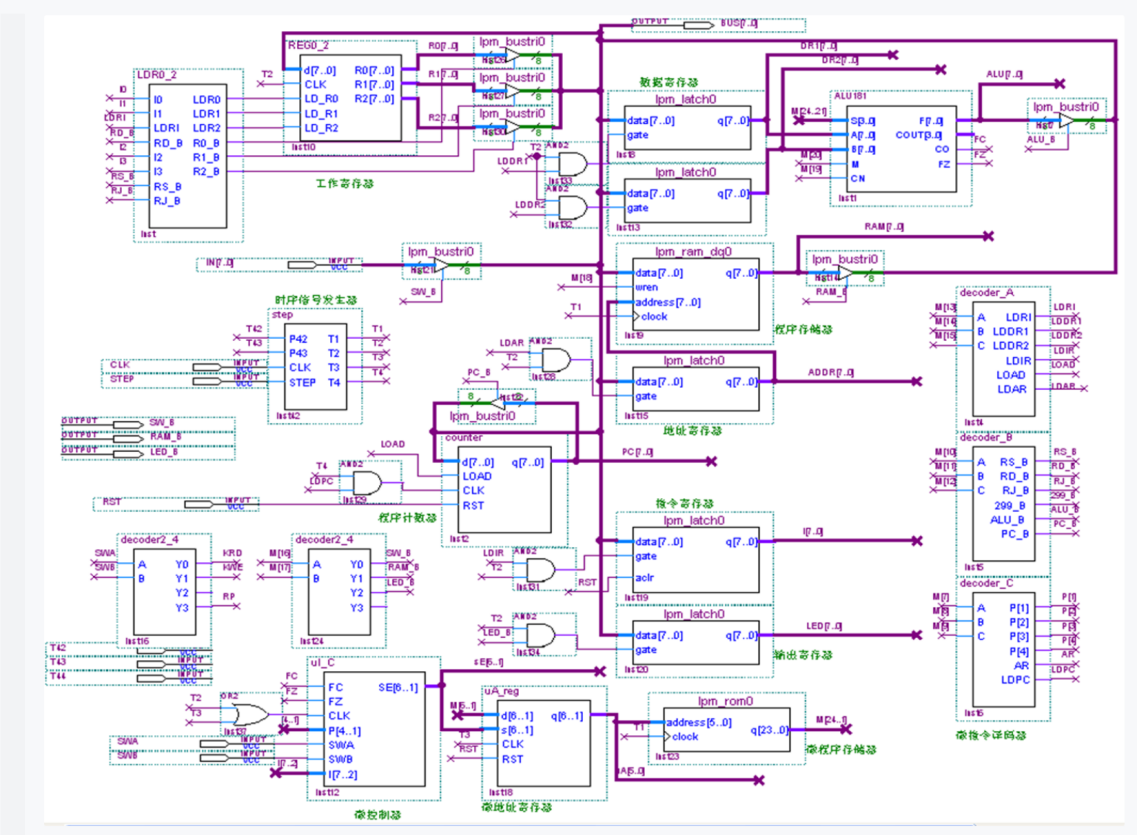


## 8位CPU组成：



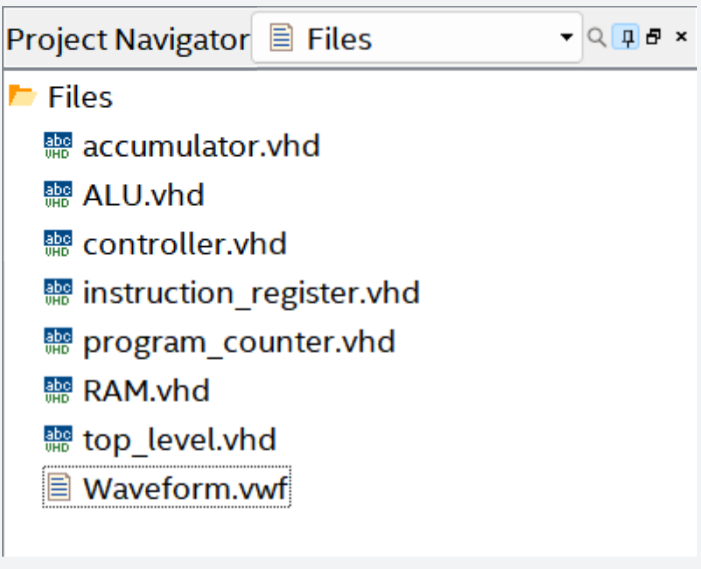


## 8位CPU的顶层设计电路原理图



# 四、实验准备

## 1.设计内容



## 2.元件接口

### accumulator累加器

|  |
| --- |
| entity accumulator is ----声明外部实体接口  port (  clk, en\_D, ld, selAlu, reset: in STD\_LOGIC; --时钟信号  aluD: in STD\_LOGIC\_VECTOR(7 downto 0);  dBus: inout STD\_LOGIC\_VECTOR(7 downto 0);  q: out STD\_LOGIC\_VECTOR(7 downto 0)  );  end accumulator; |

### ALU运算器

|  |
| --- |
| entity alu is -----实体声明外部接口  port (  op: in STD\_LOGIC\_VECTOR(3 downto 0); --选择控制运算类型  accD: in STD\_LOGIC\_VECTOR(7 downto 0); -- 累加器的8位数据  dBus: in STD\_LOGIC\_VECTOR(7 downto 0); -- 数据总线用于运算  result: out STD\_LOGIC\_VECTOR(7 downto 0); --结果的输出  accZ: out STD\_LOGIC  );  end alu; |

### controller硬布线控制器

|  |
| --- |
| entity controller is ----声明实体外部接口  port (  clk, reset:  in STD\_LOGIC;  mem\_enD, mem\_rw:  out STD\_LOGIC;  pc\_enA, pc\_ld, pc\_inc: out STD\_LOGIC;  ir\_enA, ir\_enD, ir\_ld: out STD\_LOGIC;  ir\_load, ir\_store, ir\_add: in STD\_LOGIC;  ir\_sub,ir\_mul,ir\_div: in STD\_LOGIC;  ir\_and,ir\_or, ir\_not: in STD\_LOGIC;  ir\_neg, ir\_halt, ir\_branch: in STD\_LOGIC;  acc\_enD, acc\_ld, acc\_selAlu: out STD\_LOGIC;  alu\_op: out STD\_LOGIC\_VECTOR(3 downto 0)  );  end controller; |

### instruction\_register寄存器

|  |
| --- |
| entity instruction\_register is ----声明实体外部接口  port (  clk, en\_A, en\_D, ld, reset: in STD\_LOGIC;  aBus: out STD\_LOGIC\_VECTOR(7 downto 0); ----数据总线输出  dBus: inout STD\_LOGIC\_VECTOR(7 downto 0);  load, store, add,sub,mul,div,andd,orr,nott,neg, halt, branch: out STD\_LOGIC  );  end instruction\_register; |

### program\_counter程序计数器

|  |
| --- |
| entity program\_counter is  port (  clk, en\_A, ld, inc, reset: in STD\_LOGIC;  aBus: out STD\_LOGIC\_VECTOR(7 downto 0); --数据总线输出  dBus: in STD\_LOGIC\_VECTOR(7 downto 0) --数据总线输入  );  end program\_counter; |

### RAM静态储存器

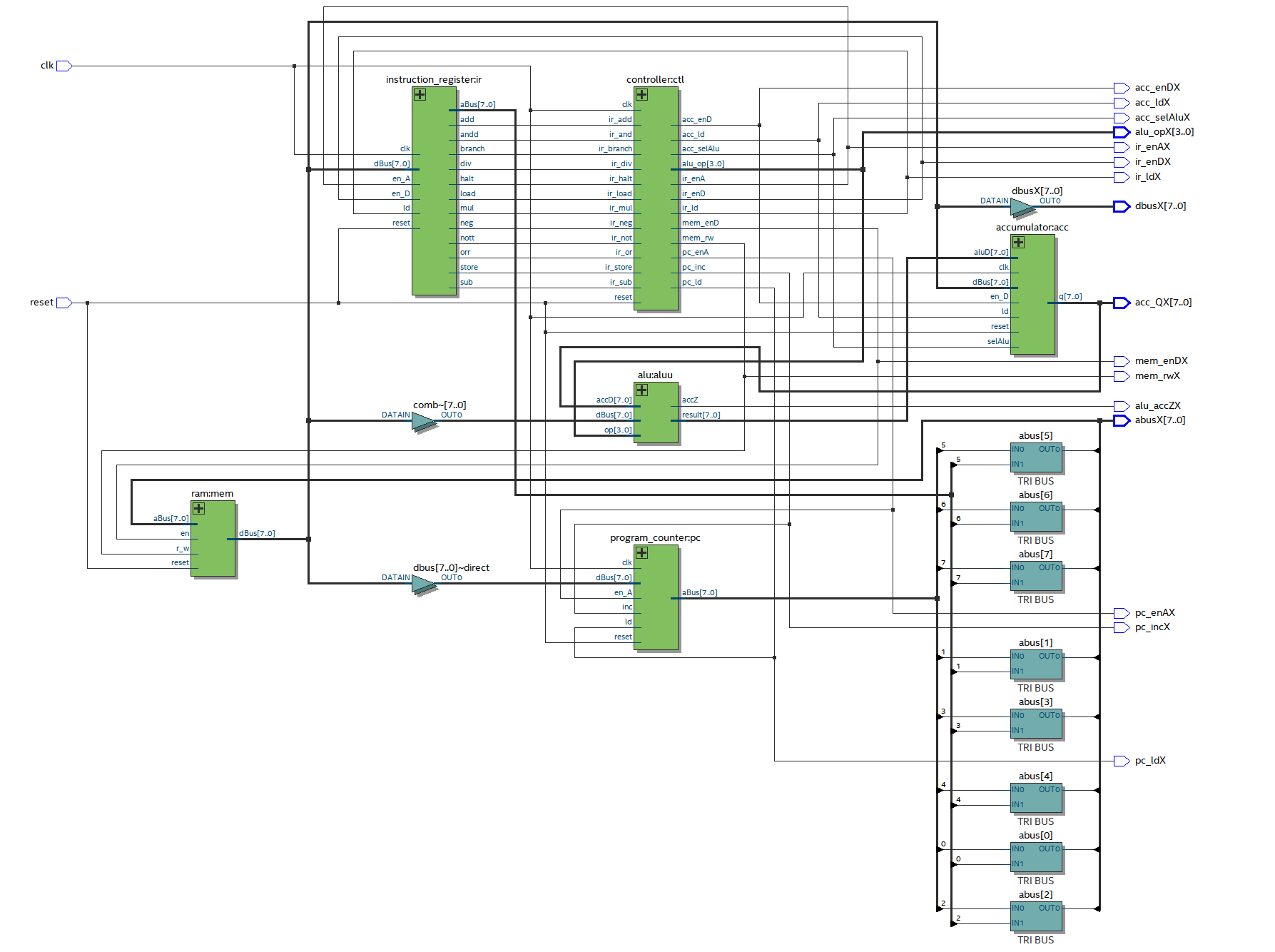
|  |
| --- |
| entity ram is  port (  r\_w, en, reset: in STD\_LOGIC;  aBus: in STD\_LOGIC\_VECTOR(7 downto 0); ----数据总线输入  dBus: inout STD\_LOGIC\_VECTOR(7 downto 0)  );  end ram; |

### top\_level由六大模块组成的模型机

|  |
| --- |
| entity top\_level is ----声明实体外部接口  port (  clk, reset:  in STD\_LOGIC;  abusX: out STD\_LOGIC\_VECTOR(7 downto 0); --数据总线输出  dbusX: out STD\_LOGIC\_VECTOR(7 downto 0);  mem\_enDX, mem\_rwX:  out STD\_LOGIC;  pc\_enAX, pc\_ldX, pc\_incX:  out STD\_LOGIC;  ir\_enAX, ir\_enDX, ir\_ldX:  out STD\_LOGIC;  acc\_enDX, acc\_ldX, acc\_selAluX:  out STD\_LOGIC;  acc\_QX: out STD\_LOGIC\_VECTOR(7 downto 0);  alu\_accZX: out STD\_LOGIC;  alu\_opX: out STD\_LOGIC\_VECTOR(3 downto 0)  );  end top\_level; |

# 五、实验过程

## 1、8位CPU电路图



## 2、ram中测试程序说明

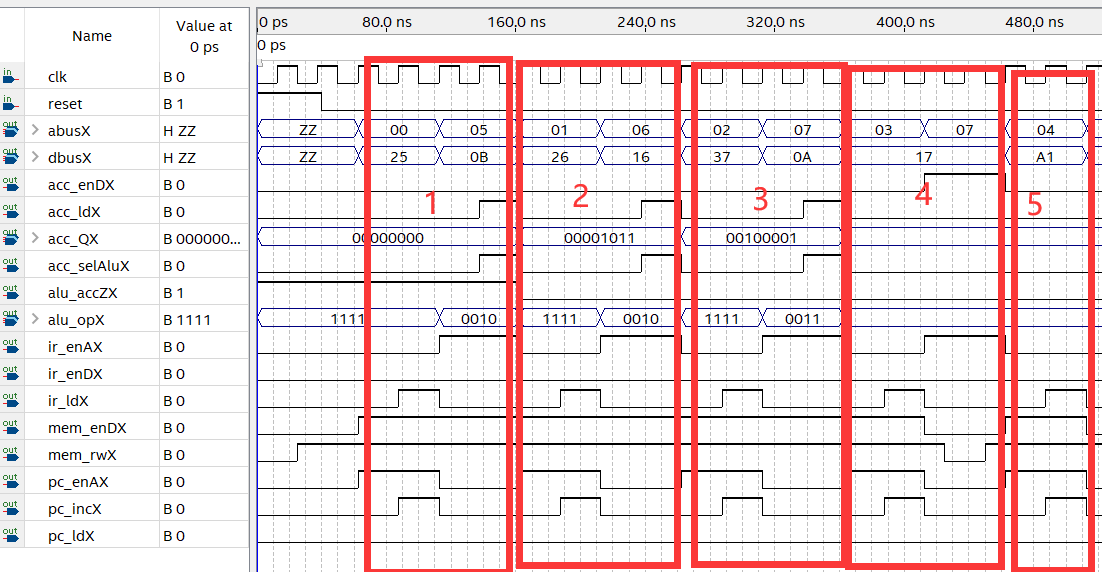
|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 指令字 | 汇编指令 | 说明 |
| 0 | ram(0) <= x"14"; | STORE 4 | 将MDR 中数据存储到4 号存储单元内 |
| 1 | ram(1) <= x"30"; | SUB 0 | ACC中数据减去0号单元数据结果放到ACC中 |
| 2 | ram(2) <= x"25"; | ADD 5 | ACC中数据加上5号单元数据结果放到ACC中 |
| 3 | ram(3) <= x"15"; | STORE 5 | 将MDR 中数据存储到5号存储单元内 |
| 4 | ram(4) <= x"46"; | MUL 6 | ACC中数据乘上5号单元数据结果放到ACC中 |
| 5 | ram(5) <= x"31"; | SUB 1 | ACC中数据加上1号单元数据结果放到ACC中 |
| 6 | ram(6) <= x"55"; | DIV 5 | ACC中数据除5号单元数据结果放到ACC 中 |
| 7 | ram(7) <= x"06"; | LOAD 6 | 将6号存储单元存储到 MDR 中 |
| 8 | ram(8) <= x"01"; | halte | 终止程序 |

## 3、测试程序，波形图及指令的对应关系分析，数据运算过程分析

1. 测试程序: 11+22-10

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 指令字 | 汇编指令 | 说明 |
| 0 | ram(0) <= x"25"; | ADD 5 | ACC中数据加上5号单元,结果放到ACC中 |
| 1 | ram(1) <= x"26"; | ADD 6 | ACC中数据加上6号单元,结果放到ACC中 |
| 2 | ram(2) <= x"37"; | **SUB 7** | ACC中数据减去7号单元,结果放到ACC中 |
| 3 | ram(3) <= x"17"; | **STORE 7** | 将MDR 中数据存储到7号存储单元内 |
| 4 | ram(4) <= x"A1"; | **halte** | 终止程序 |
| 5 | ram(5) <= x"0B"; | LOAD 11 | 将11号存储单元存储到 MDR中 |
| 6 | ram(6) <= x"16"; | **STORE 6** | 将MDR 中数据存储到6号存储单元内 |
| 7 | ram(7) <= x"0A"; | LOAD 10 | 将10号存储单元存储到 MDR中 |

（2）程序数据运算过程分析，波形图及逐条指令、逐个部件逐个数据变化的对应关系分析。



周期1分析：

取址：

pc\_enA置为高电平，使能pc寄存器的地址输出端，将pc寄存器的值00赋值给abus（地址寄存器MAR），pc\_inc置为高电平，pc寄存器的值自增1，mem\_rw置为高电平，对内存进行读操作，将MAR里的地址送入内存进行寻址，mem\_enD置为高电平，使能内存的数据输出端，将找到的数据25通过数据总线送入dbus（数据寄存器MDR），ir\_ld置为高电平，使能IR寄存器的数据输入端，将dbus的数据25输入到IR寄存器里，操作码为0010，地址码为0101。

执行：

ir\_enA置为高电平，使能IR寄存器地址输出端，将地址码送入abus（地址寄存器MAR），mem\_rw置为高电平，对内存进行读操作，根据abus的内容对内存进行寻址取数，mem\_enD置为高电平，将数据0B送入dbus（数据寄存器MDR），对操作码进行译码，译码结果为加法指令，alu\_op输出为0010，acc\_selAlu置为高电平，将dbus里的数据和acc里的数据进行加法运算，acc\_ld置为高电平，将运算的结果赋值给acc，此时acc的值为0B

周期2分析：

取址：

pc\_enA置为高电平，使能pc寄存器的地址输出端，将pc寄存器的值01赋值给abus（地址寄存器MAR），pc\_inc置为高电平，pc寄存器的值自增1，mem\_rw置为高电平，对内存进行读操作，将MAR里的地址送入内存进行寻址，mem\_enD置为高电平，使能内存的数据输出端，将找到的数据26通过数据总线送入dbus（数据寄存器MDR），ir\_ld置为高电平，使能IR寄存器的数据输入端，将dbus的数据26输入到IR寄存器里，操作码为0010，地址码为0110。

执行：

ir\_enA置为高电平，使能IR寄存器地址输出端，将地址码送入abus（地址寄存器MAR），mem\_rw置为高电平，对内存进行读操作，根据abus的内容对内存进行寻址取数，mem\_enD置为高电平，将数据16送入dbus（数据寄存器MDR），对操作码进行译码，译码结果为加法指令，alu\_op输出为0010，acc\_selAlu置为高电平，将dbus里的数据和acc里的数据进行加法运算，acc\_ld置为高电平，将运算的结果赋值给acc，此时acc的值为21

周期3分析：

取址：

pc\_enA置为高电平，使能pc寄存器的地址输出端，将pc寄存器的值02赋值给abus（地址寄存器MAR），pc\_inc置为高电平，pc寄存器的值自增1，mem\_rw置为高电平，对内存进行读操作，将MAR里的地址送入内存进行寻址，mem\_enD置为高电平，使能内存的数据输出端，将找到的数据37通过数据总线送入dbus（数据寄存器MDR），ir\_ld置为高电平，使能IR寄存器的数据输入端，将dbus的数据37输入到IR寄存器里，操作码为0011，地址码为0111。

执行：

ir\_enA置为高电平，使能IR寄存器地址输出端，将地址码送入abus（地址寄存器MAR），mem\_rw置为高电平，对内存进行读操作，根据abus的内容对内存进行寻址取数，mem\_enD置为高电平，将数据0A送入dbus（数据寄存器MDR），对操作码进行译码，译码结果为减法指令，alu\_op输出为0011，acc\_selAlu置为高电平，将dbus里的数据和acc里的数据进行减法运算，acc\_ld置为高电平，将运算的结果赋值给acc，此时acc的值为17

周期4分析：

取址：

pc\_enA置为高电平，使能pc寄存器的地址输出端，将pc寄存器的值03赋值给abus（地址寄存器MAR），pc\_inc置为高电平，pc寄存器的值自增1，mem\_rw置为高电平，对内存进行读操作，将MAR里的地址送入内存进行寻址，mem\_enD置为高电平，使能内存的数据输出端，将找到的数据17通过数据总线送入dbus（数据寄存器MDR），ir\_ld置为高电平，使能IR寄存器的数据输入端，将dbus的数据17输入到IR寄存器里，操作码为0001，地址码为0111。

执行：

ir\_enA置为高电平，使能IR寄存器地址输出端，将地址码送入abus（地址寄存器MAR），对操作码进行译码，译码结果为存储指令，mem\_enD置为低电平，mem\_rw置为低电平，对内存进行写操作，根据abus的内容对内存进行寻址，acc\_enD置为高电平，将acc里的数据17写入到dbus（数据寄存器MDR），并送入到内存中进行存储。

分析：

取址：

pc\_enA置为高电平，使能pc寄存器的地址输出端，将pc寄存器的值04赋值给abus（地址寄存器MAR），pc\_inc置为高电平，pc寄存器的值自增1，mem\_rw置为高电平，对内存进行读操作，将MAR里的地址送入内存进行寻址，mem\_enD置为高电平，使能内存的数据输出端，将找到的数据A1通过数据总线送入dbus（数据寄存器MDR），ir\_ld置为高电平，使能IR寄存器的数据输入端，将dbus的数据A1输入到IR寄存器里，操作码为1010，地址码为0001。

执行：

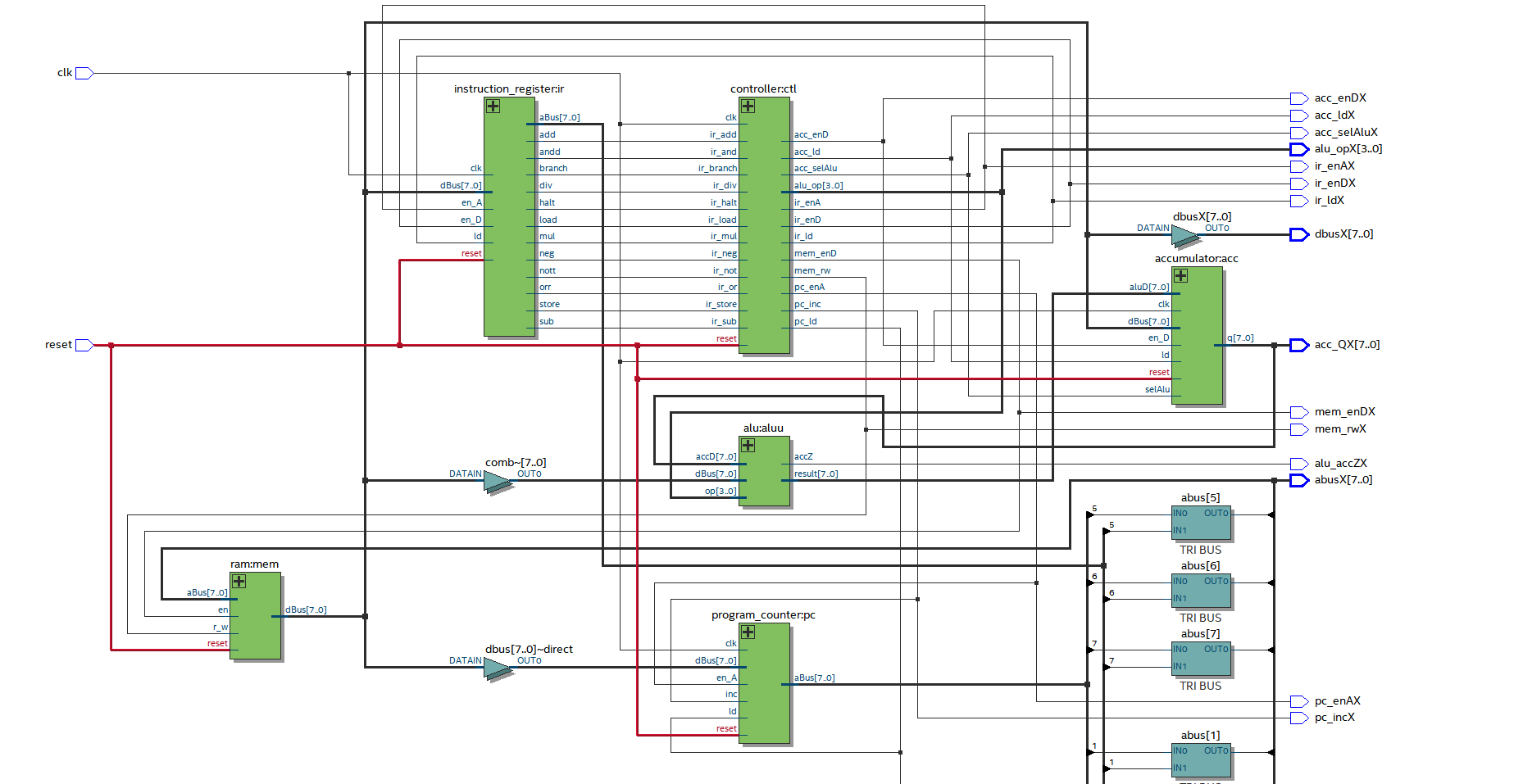
对操作码进行译码，译码结果为停机指令，停止执行。

（3）任选一条指令为例，必须结合测试波形、电路图、代码、指令系统设计具体说明，标注控制信号，地址，数据等信息的每一次流动、变化。

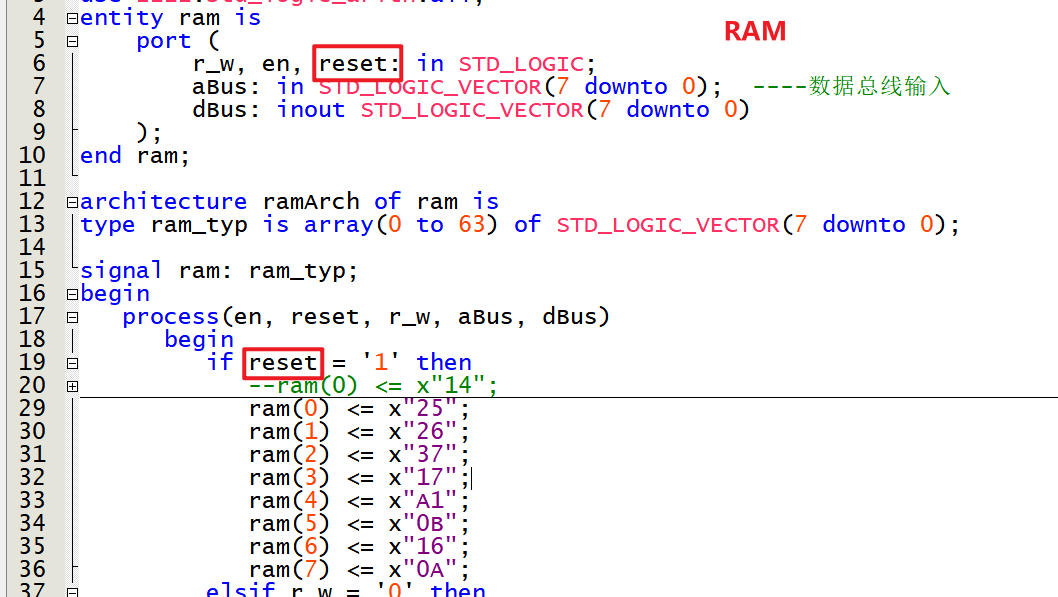
这里选择指令”25”进行讲解

**初始化：**

reset置为高电平，



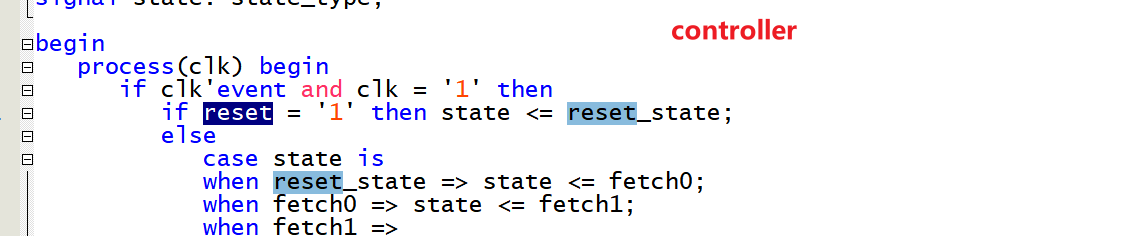
从电路图中可看出，ram、IR、control、PC、ACC这几个模块会进行初始化，以下是这些模块的代码部分，



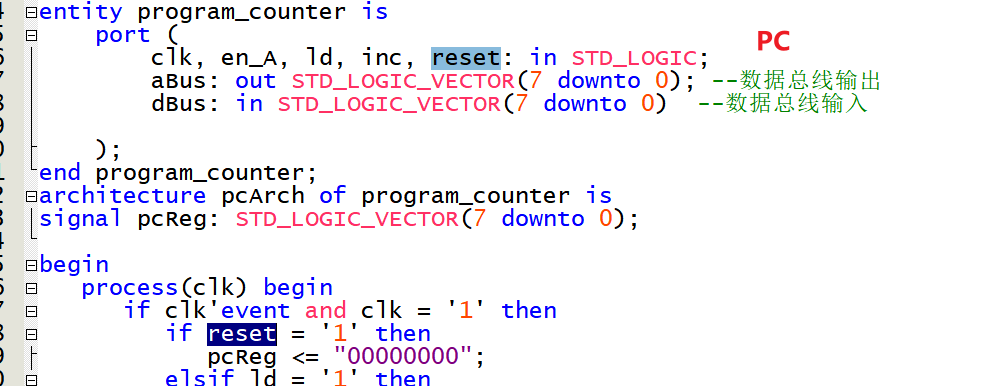
RAM：当检测到reset为高电平时，初始化内存，使得地址0到7所对应存储单元存入相应的数据；



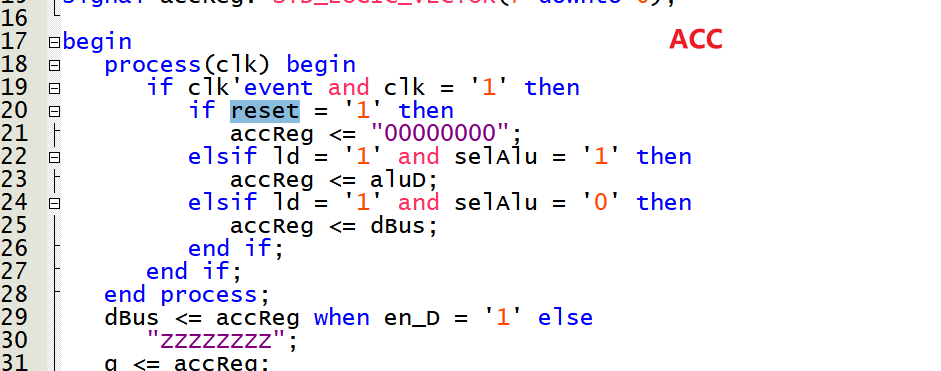
IR：当检测到reset为高电平时，irReg初始化为“00000000”，在这段代码中irReg是用来保存指令的，用于后续分析；



Controller：当检测到reset为高电平时，state置为reset\_state，state用于后续分析产生相应控制信号；



PC：当检测到reset为高电平时，pcReg计数值置为“00000000”；



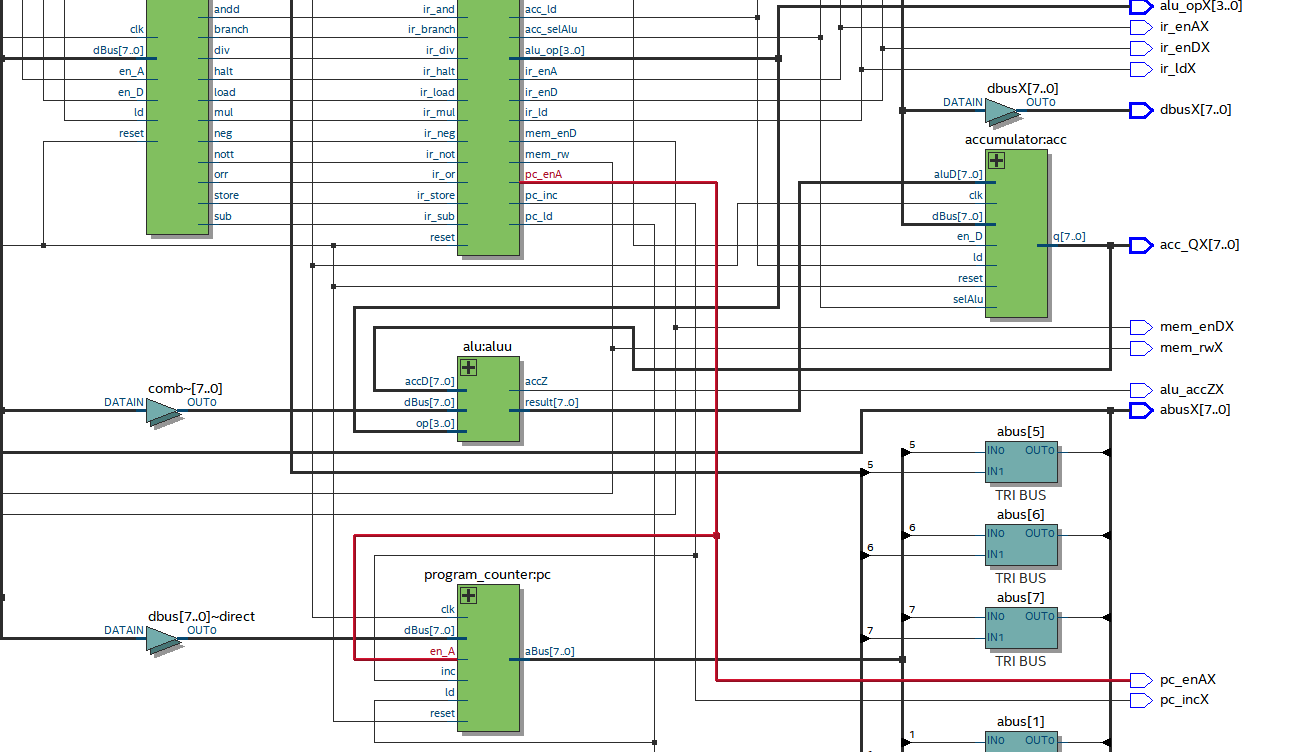
ACC：当检测到reset为高电平时，accReg置为“00000000”；

**整条指令的执行分为取址和执行两部分**

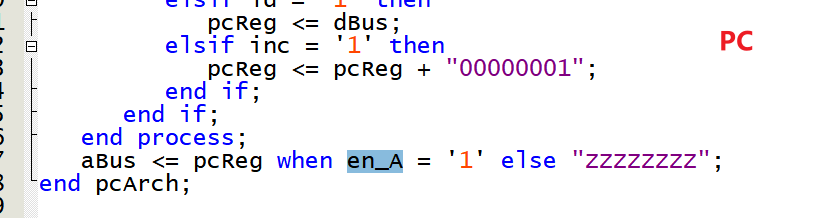
**取址部分：**

由波形图可看出，

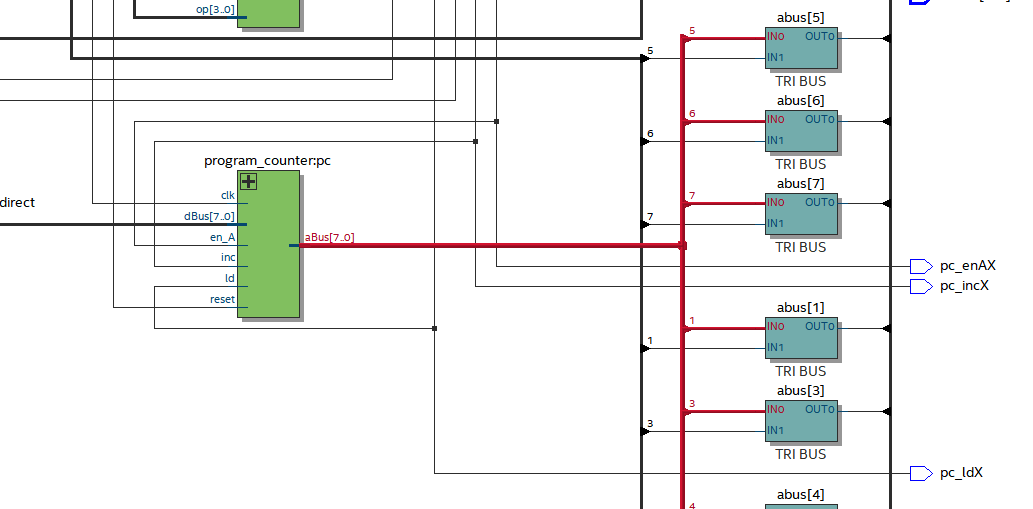
pc\_enA为高电平，reset为低电平，



由电路图可看出，pc\_enA传入pc中，pc的代码如图

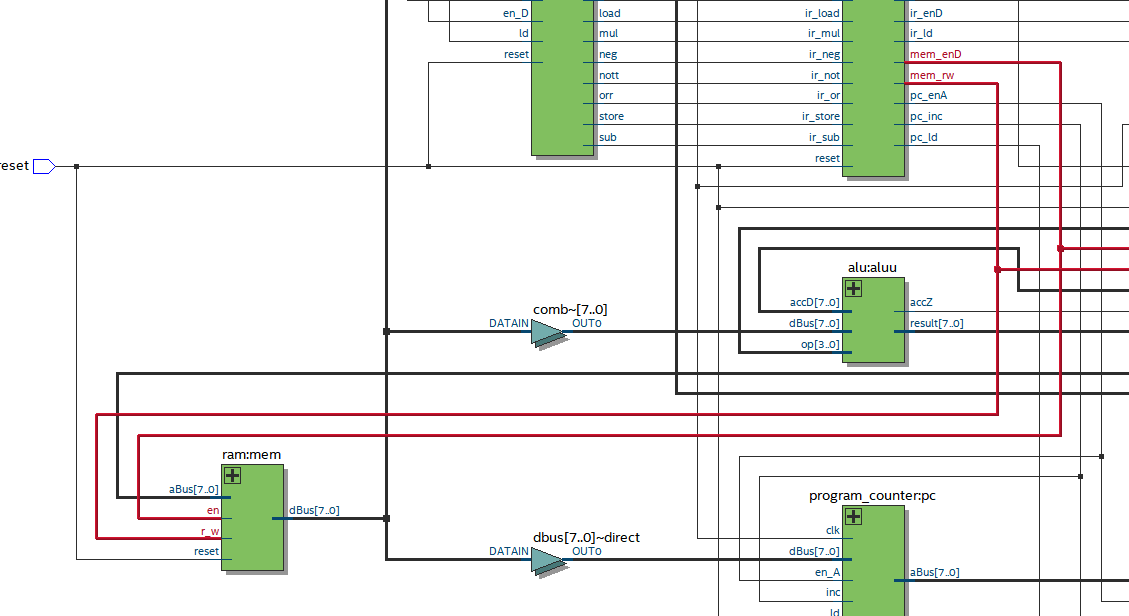


PC检测到en\_A为高电平，将pcReg赋值给aBus，而之前pcReg被赋值为“00000000”，所以aBus赋值为“00000000”，也就是MAR（地址寄存器）赋值为“00000000”，波形图上的abus输出为“00”（十六进制）。流经的电路如下图所示，

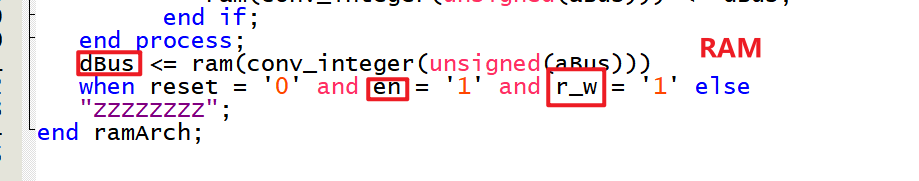


紧接着，通过波形图可看出，

mem\_enD和mem\_rw均为高电平，流经电路如下，

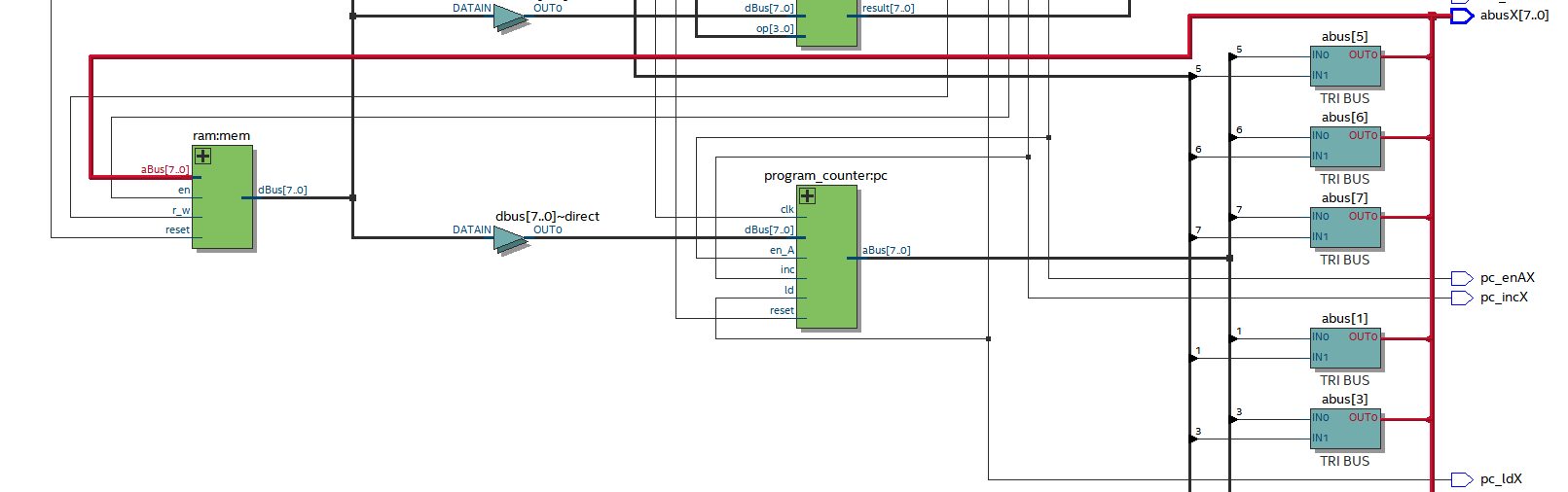


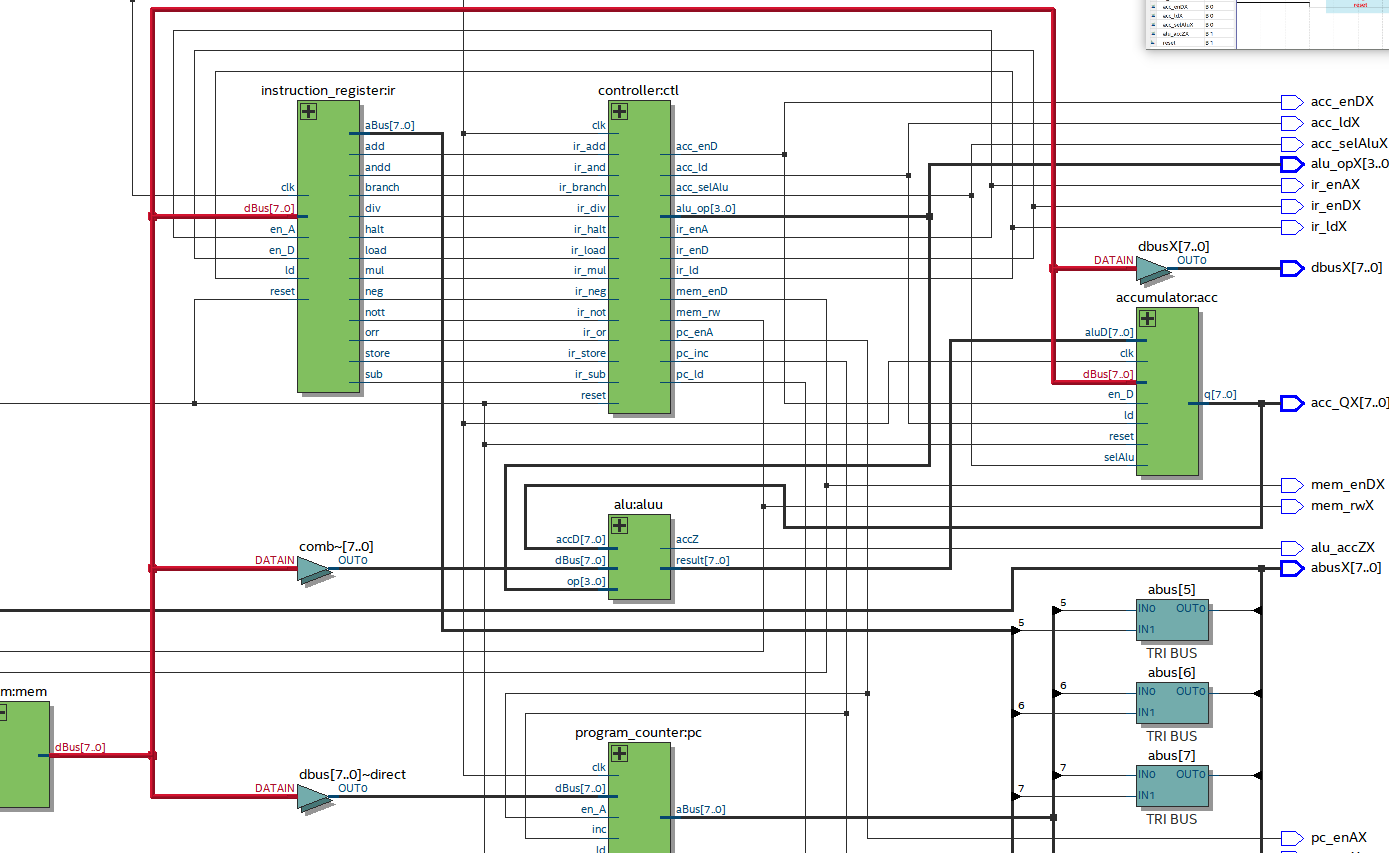
以下是RAM的部分代码，



不难看出，根据MAR，将内存中该地址下的数据存放到dBus上，也就是数据寄存器，波形图也能够看出，

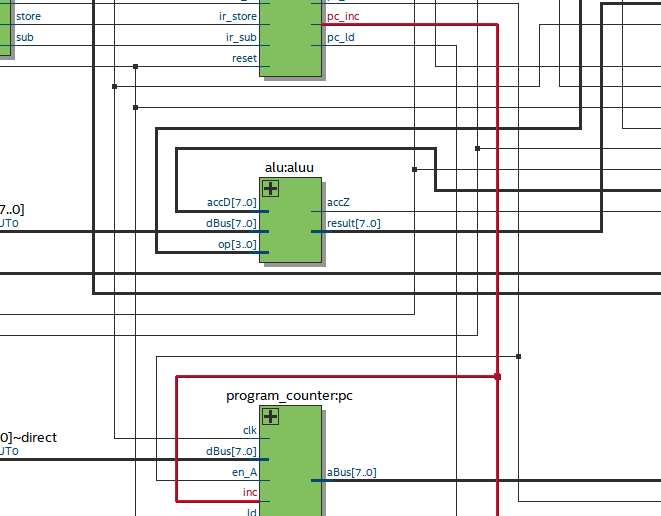
dbus的值为“25”（十六进制），流经的电路如下图所示，



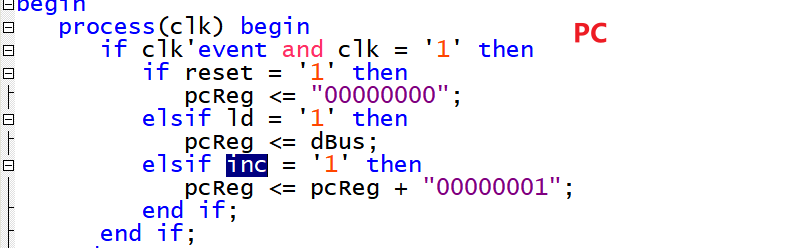


紧接着，通过波形图可看出，

pc\_inc为高电平，流经电路如下，

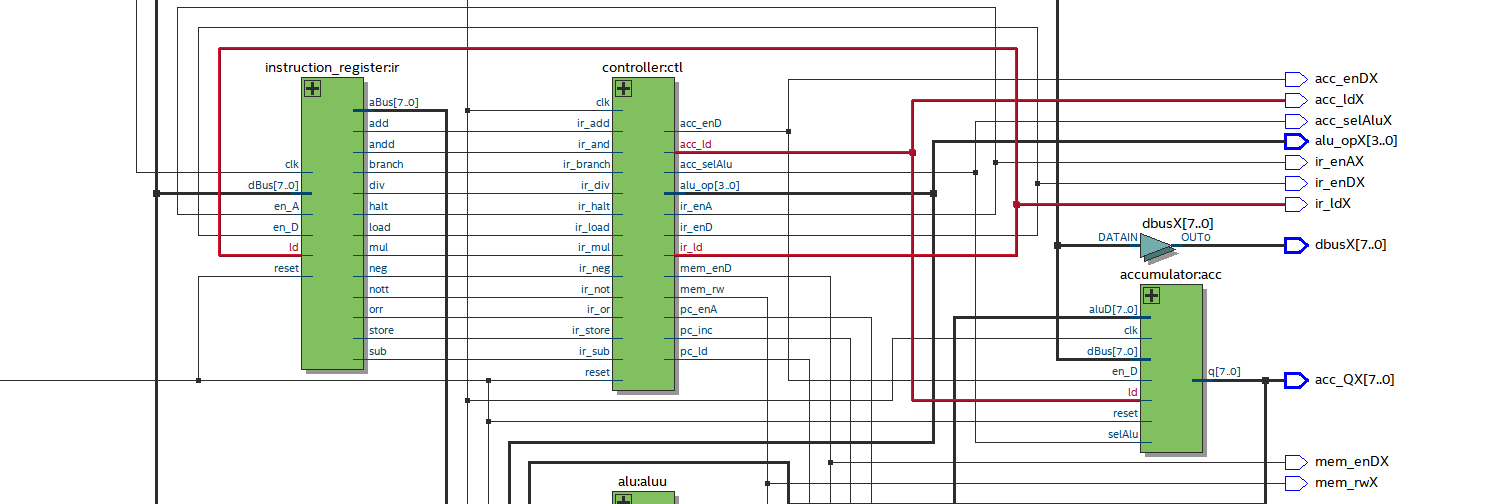


以下是PC的部分代码，

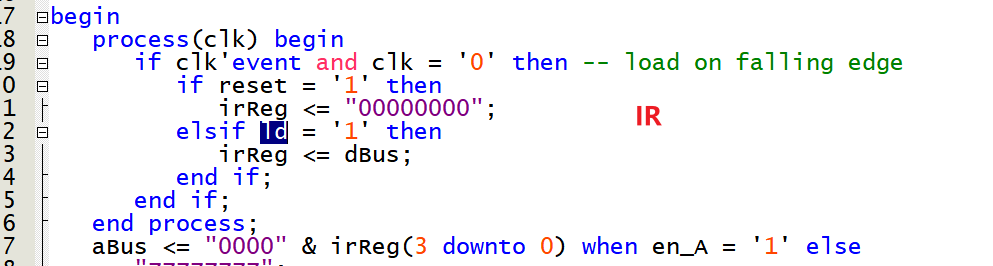


不难看出，pcReg自增一，紧接着，通过波形图可看出，

acc\_ld为低电平，ir\_ld为高电平，流经电路如下，

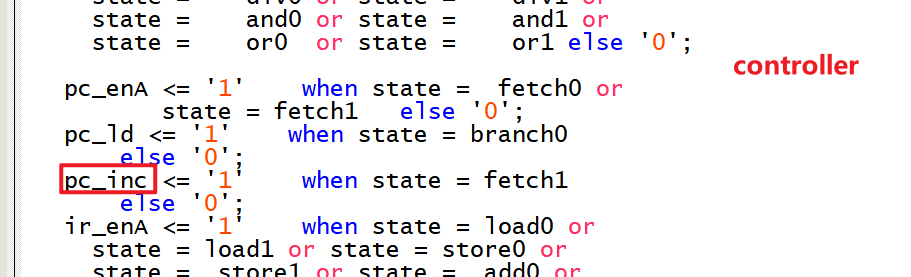


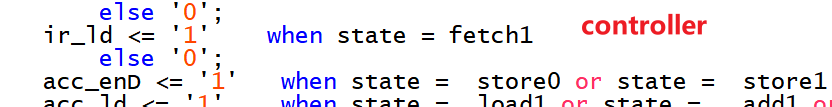
以下是IR的部分代码



不难看出，dbus的值，也就是MDR的值会赋给irReg，由于acc\_ld为低电平，故dbus的数据不会流入acc中，波形图上的acc\_Q保持不变，

为“00”（十六进制），以下是controller的部分代码，





不难看出，

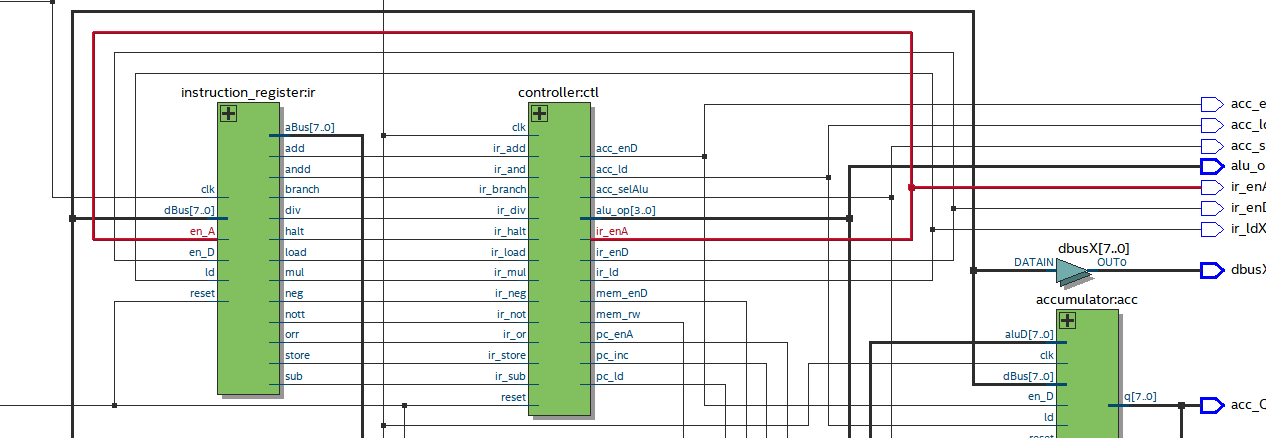
pc\_inc置为0，pc\_enA置为0，ir\_ld置为0，到此取址阶段就算完成了。

**执行部分：**

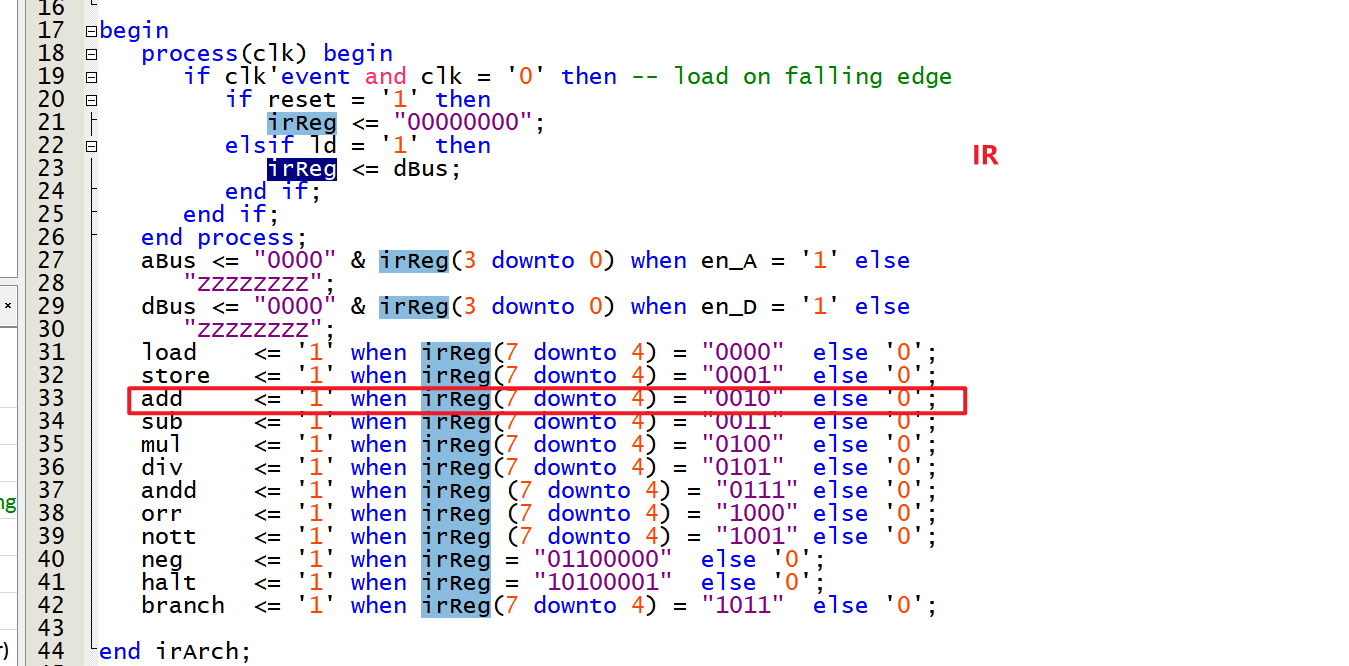
前面取址完成后，指令已经存入IR寄存器里了，也就是irReg存储的值为“00100101”现从IR寄存器开始下面的分析，

从波形图中，不难看出，

ir\_enA置为高电平，以下是流经电路，

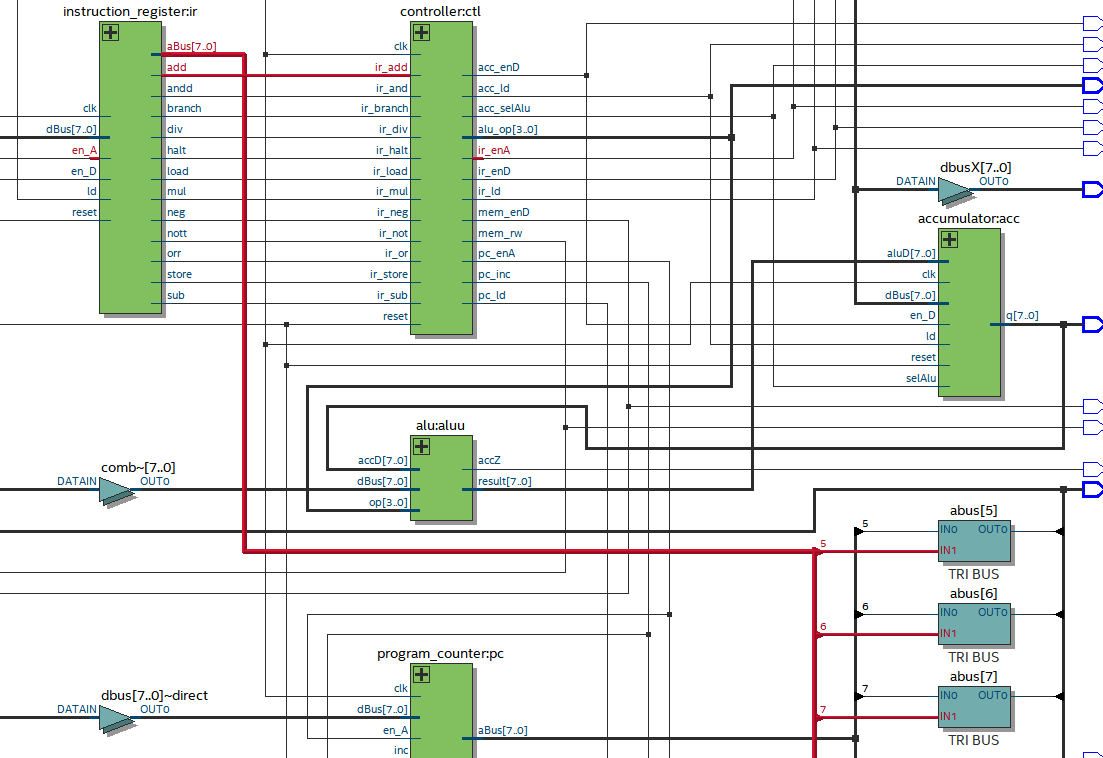


以下是IR部分代码



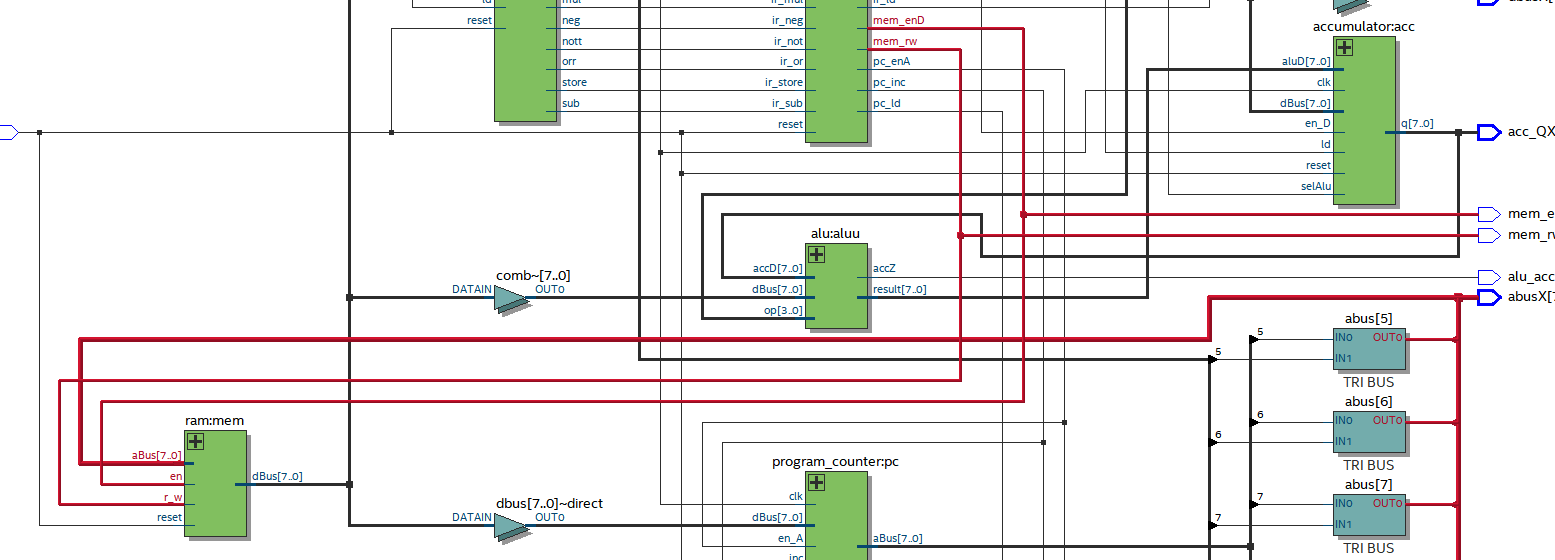
不难看出，aBus的被赋值为“00000101”，并送到地址寄存器（MAR）中，此时abus为“05”（十六进制），

跟波形图的显示是一致的；接着irReg经解析后，add被赋值为1，并流向controler中，以下是流经电路，



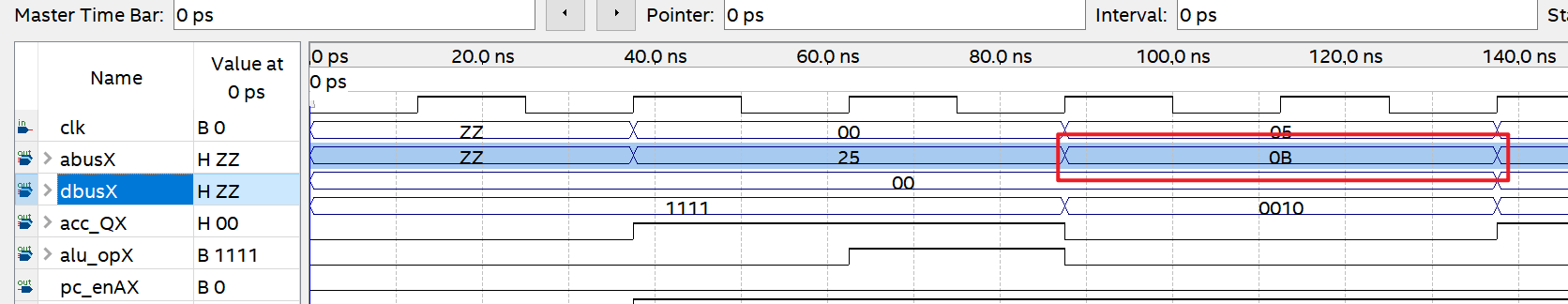
接着是根据地址到内存中相应位置取出数据，

mem\_enD和mem\_rw均为高电平，流经电路如下，



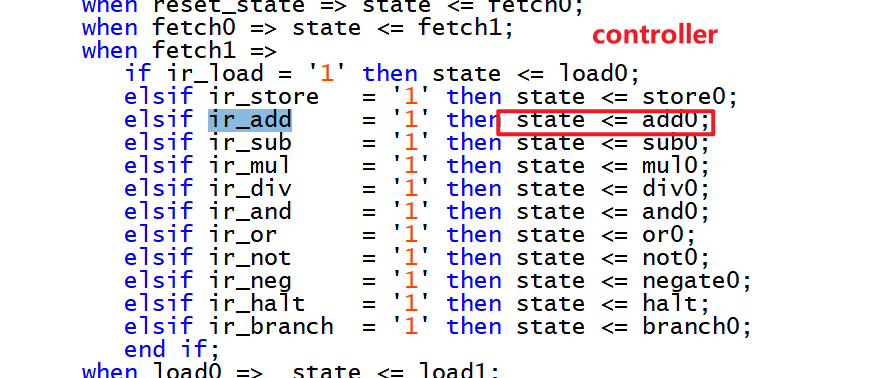
以下是RAM部分代码，

不难看出，dBus被赋值为0B，同时将数据送往dbus（数据寄存器MDR），此时dbus的值为“0B”（十六进制），

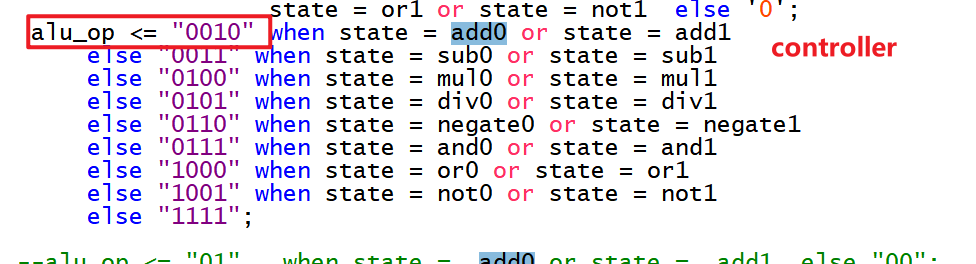


与波形图的一致，流经电路如下，

紧接着，刚才的add会传入controller里面，以下是controller部分代码，

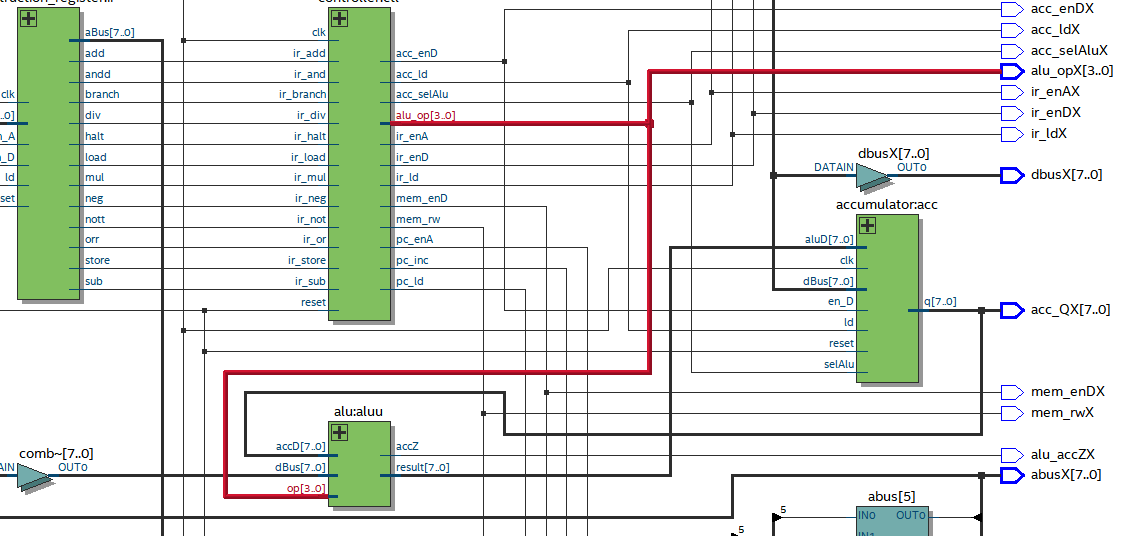


state会被赋值为add0，

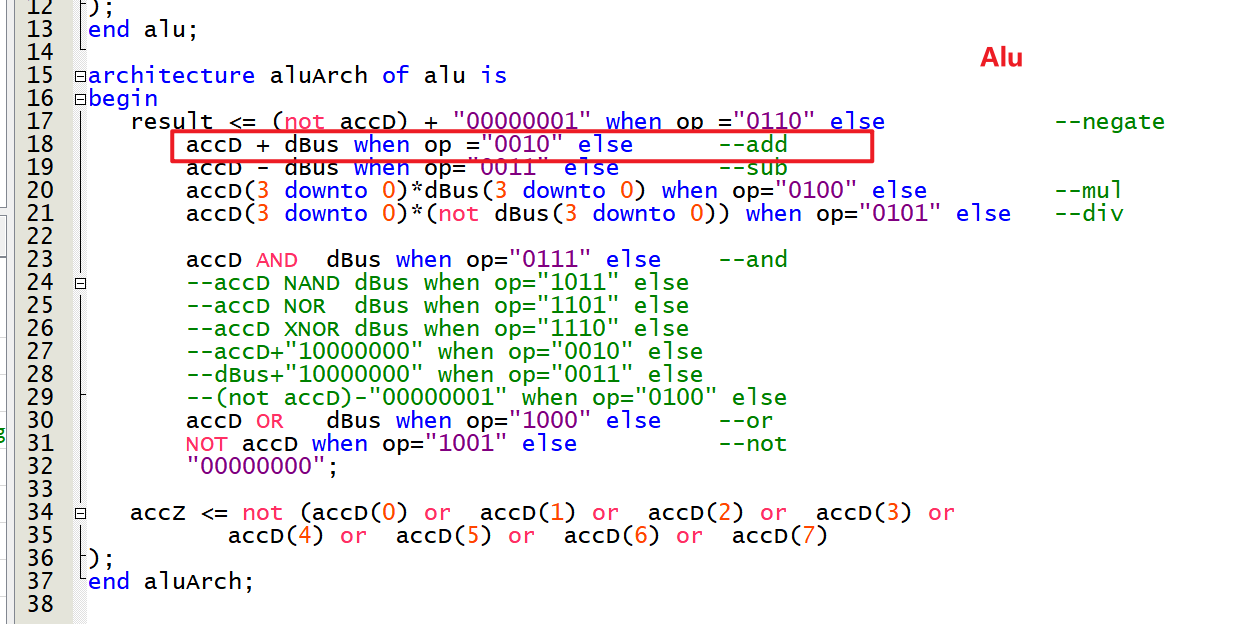


alu\_op被赋值为“0010”，

alu\_op会流入alu中，流经电路如下，



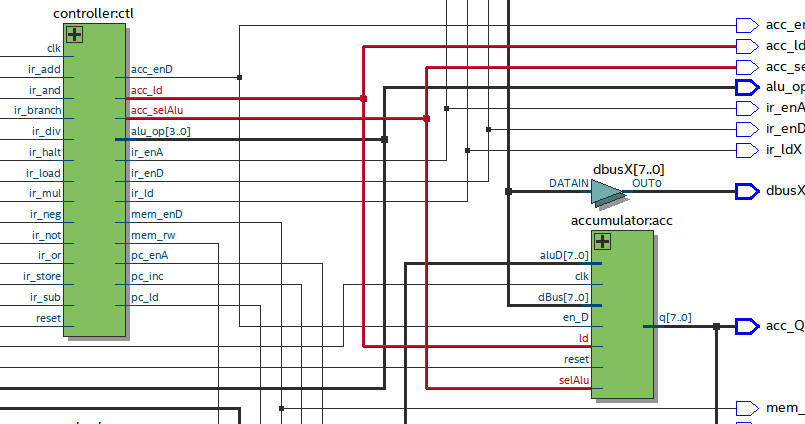
以下是alu的部分代码，



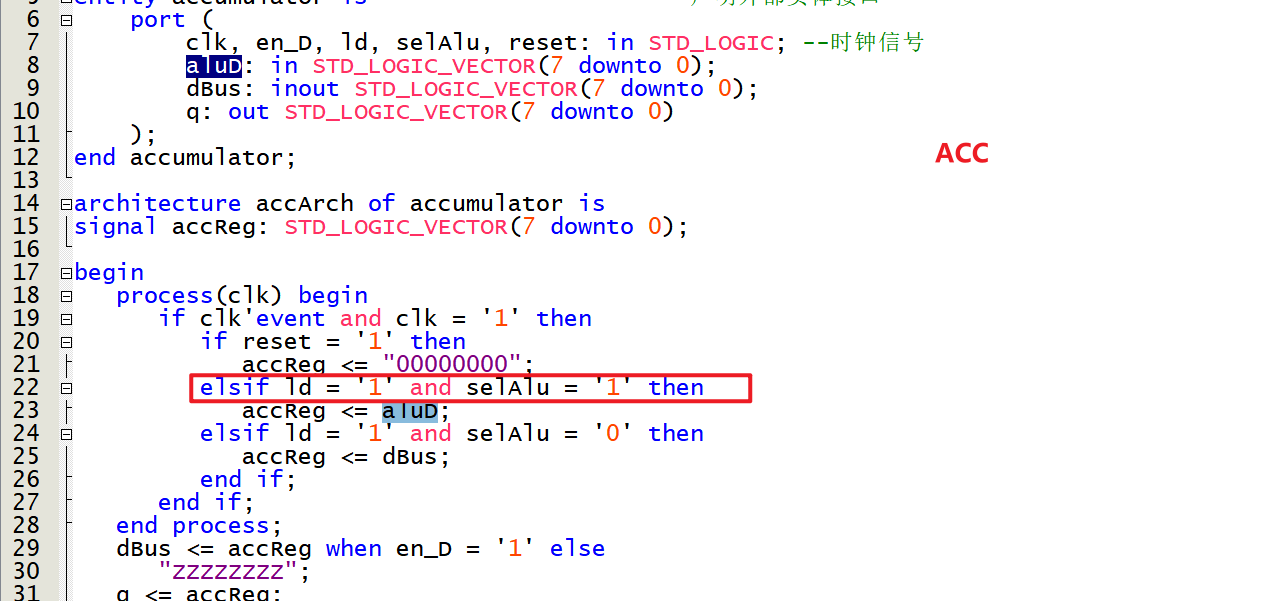
经分析，op为加法指令，accD与dBus相加，accD为累加器里的值，之前分析过，累加器的初始值为0，所以这里的accD的值为“00000000”，dBus的值为dbus所对应的值，也就是MDR的值，为“00001011”，相加后的结果为“00001011”，即result的值为“00001011”，

通过波形图可看出，

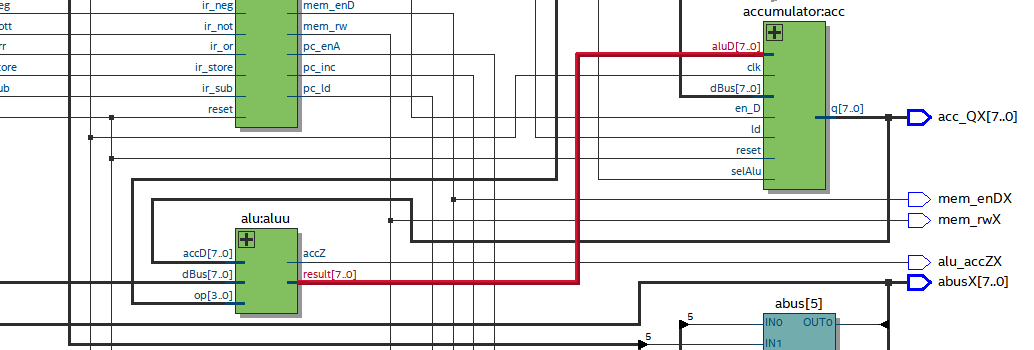
acc\_ld与acc\_selAlu都为高电平，流经电路如下，



以下是ACC部分代码，



不难看出，accReg被赋值为aluD，而aluD连接的是ALU的result输出端，如下



所以aluD的值为“00001011”，十六进制为“0B”，即ACC寄存器里的值为“0B”，与下图波形图中的结果相同

acc\_ld与acc\_selAlu均置为低电平，ir\_enA也置为低电平。

到此执行部分就算完成了。