**MoonCore**

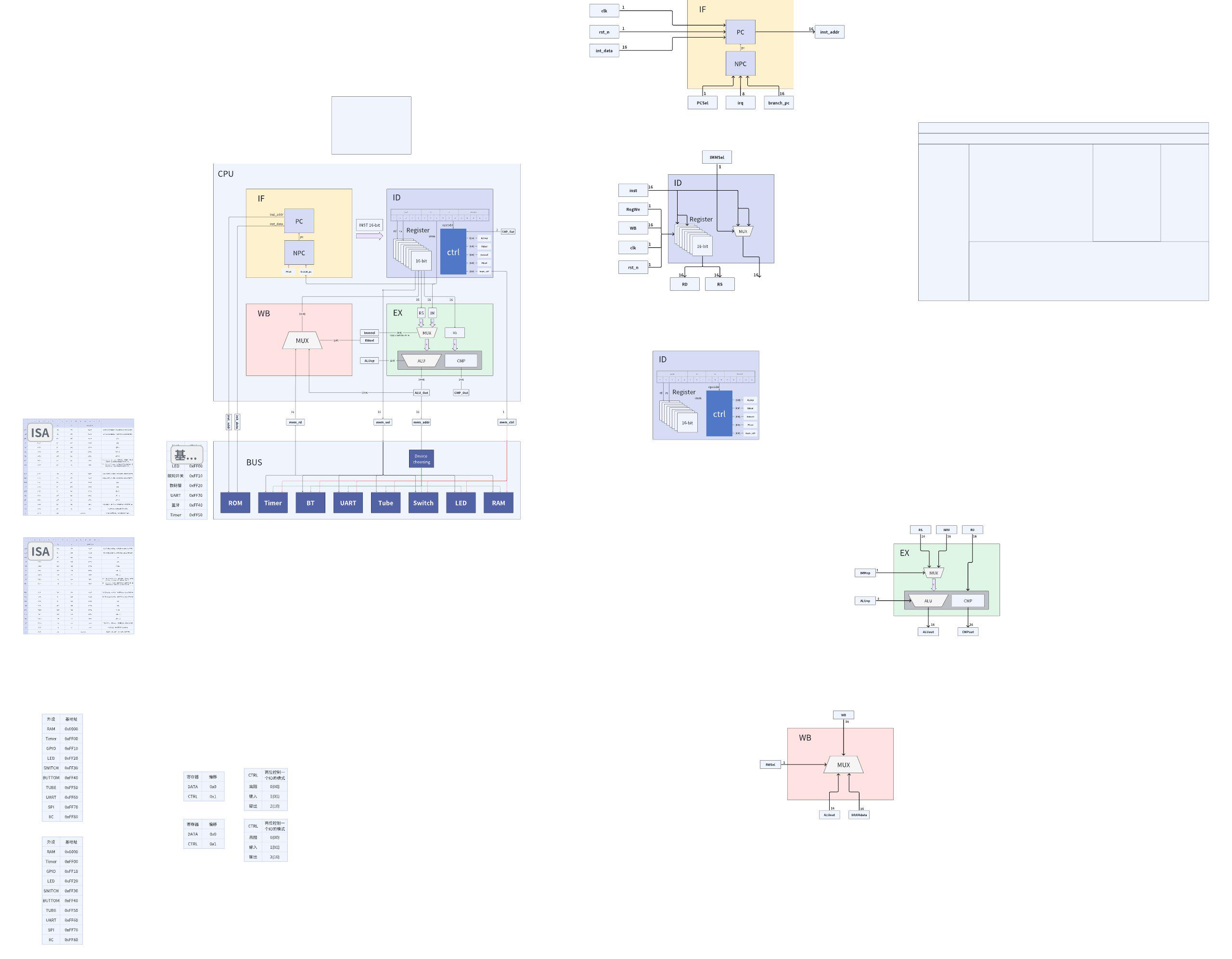
版本更新说明：

2023.9.5 更新了项目文档。完成初步CPU架构搭建；基本验证ALU功能。 Contributor: Tao, Liao;

2023.10.25 更新项目文档

2024.2.4

**Content**



|  |
| --- |
| [**Content**](#heading_0)  [**1. CPU specification**](#heading_1)  [**2. CPU ISA**](#heading_2)  [**（1）16-bit-inst 构成**](#heading_3)  [**（2）指令效果解释**](#heading_4)  [**3. CPU架构**](#heading_5)  [**（1）整体框架图**](#heading_6)  [**（2）ctrl**](#heading_7)  [**（3）IF**](#heading_8)  [**（4）ID**](#heading_9)  [**（5）EX**](#heading_10)  [**（6）WB**](#heading_11)  [**4. BUS结构**](#heading_12)  [**（1）整体结构**](#heading_13)  [**（2）ROM**](#heading_14)  [**（3）RAM**](#heading_15)  [**（4）外设**](#heading_16)  [**5. Vivado 使用问题**](#heading_17)  [**（1）文件类**](#heading_18) |

**1. CPU specification**

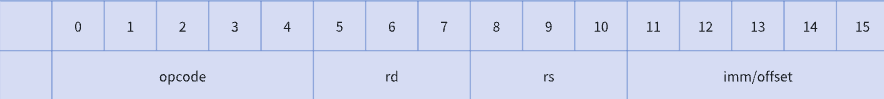
16位cpu 哈佛结构50MHz主频单周期

**2. CPU ISA**

**（1）16-bit-inst 构成**

第一种指令：适用于add、sub、and、or、xor、sll、srl、addi、subi、andi、ori、xori、slli、srli、beq、blt、lw、sw

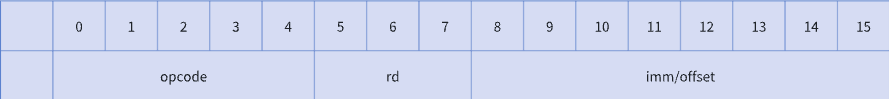
命令使用格式为：opcode rd rs xxx（将rd、rs两个寄存器指向的值，进行opcode对应的计算操作，再将结果存在rd中。这种命令空余两个指令位未使用）



第二种指令：适用于li、

命令使用格式为：li rd xxxxxxxx （将后面八位的立即数，放入rd指向的寄存器中）

（多这么一条指令主要是为了方便向寄存器填入值。有了这个指令能一次性向某个寄存器填入8位数据）



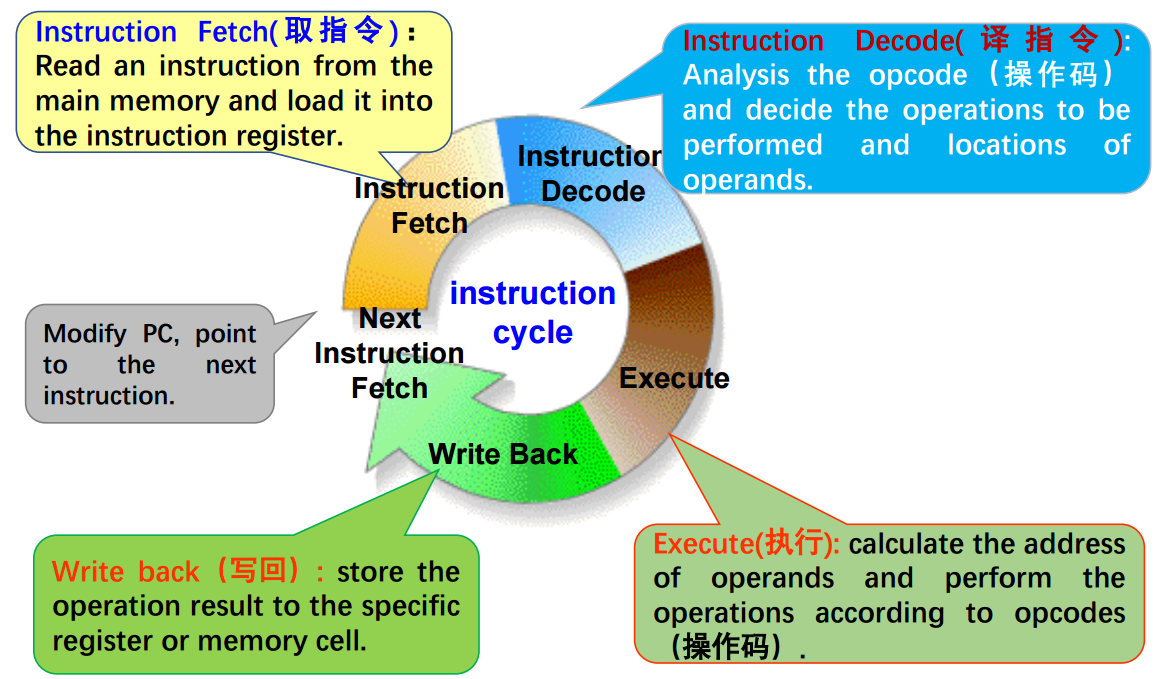
**（2）指令效果解释**

参考前面画板中框图内容

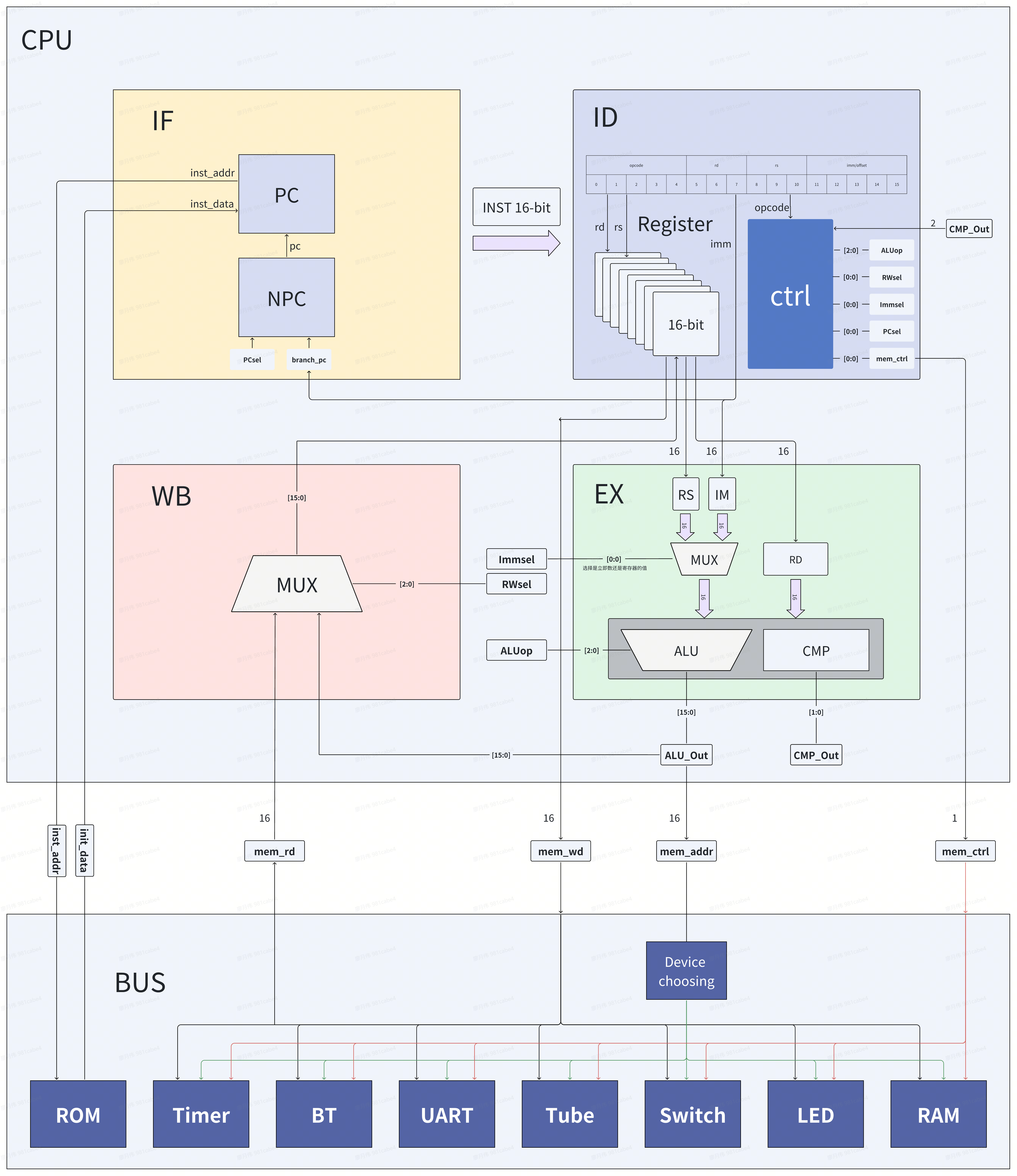
**3. CPU架构**

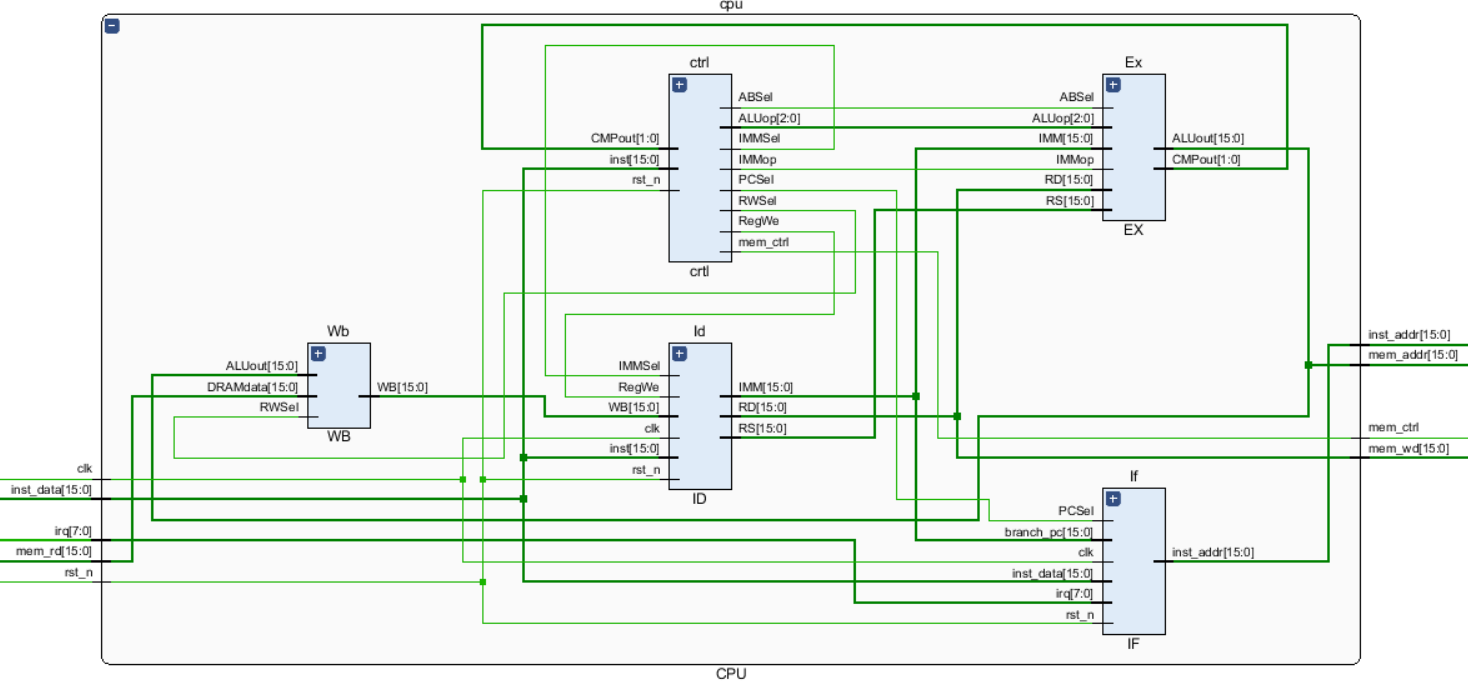
cpu架构主要参考：https://www.zhihu.com/column/c\_1530950608688910336

**（1）整体框架图**



整体rtl框图





**整体cpu模块：**

clk & rst\_n 时钟和复位信号

irq 中断信号（）

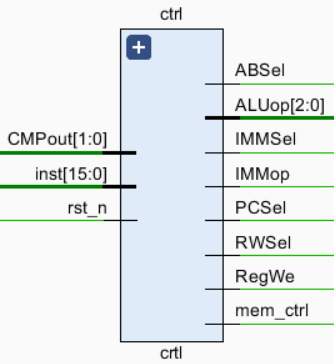
inst\_data & inst\_addr 指令地址以及指令内容

mem\_rd & mem\_wd cpu的输入输出数据

mem\_addr cpu的输入输出地址

mem\_crtl cpu的输入输出控制

**（2）ctrl**



**控制模块**：（CU）

将输入的指令截取出相应部分，通过译码opcode，生成各个多路复用器的控制信号

**输入**：

rst\_n 复位信号

inst 指令

CMPout EX模块中CMP的比较结果

**输出**：

ALUop alu模块执行的操作(加、减、and、or、xor、左移、右移）

IMMSel 控制ID输出是指令中的5位立即数，还是8位立即数

IMMop 控制输入给alu和cmp的数据是立即数，还是寄存器中的数

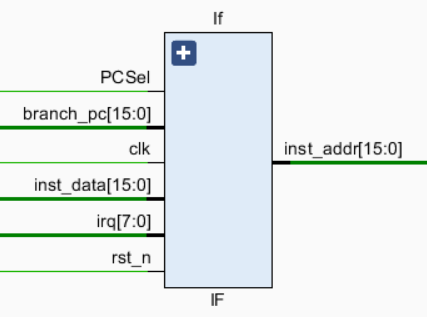
PCSel 控制pc是下一条指令，还是branch\_pc

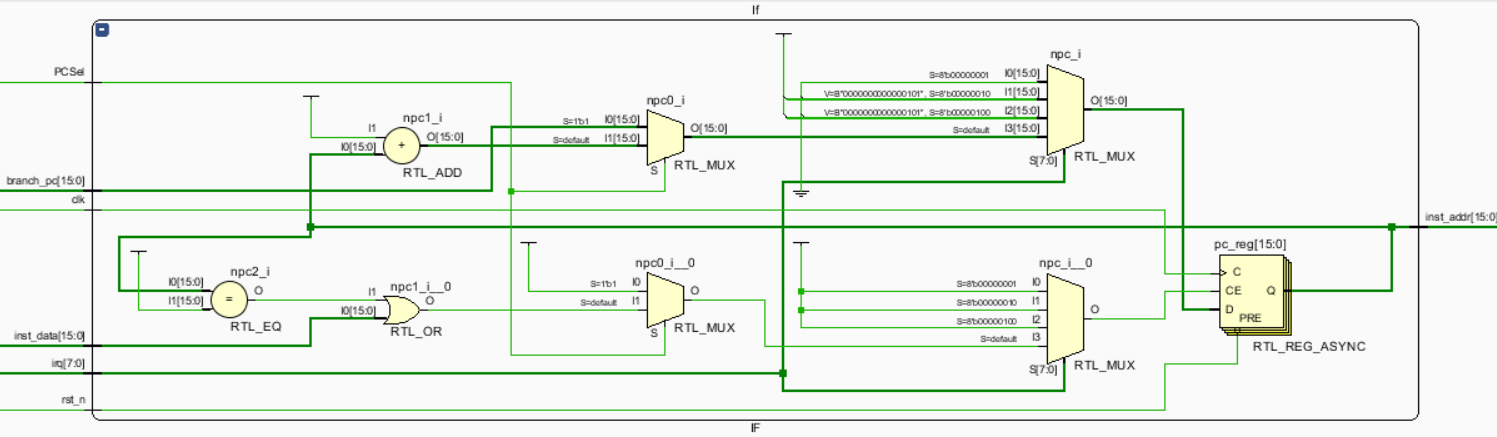
RWSel 控制将alu输出结果写回到寄存器组，还是写回外部读取的地址

RegWe 寄存器堆的写写使能信号

mem\_ctrl 控制对外部设备是读操作，还是写操作

**（3）IF**





**取指令模块**：

输出PC（inst\_addr）向外部rom读取指令。由输入的PCSel控制，下一条指令是顺位下一条，还是输入的branch\_pc

**输入**：

clk & rst\_n 时钟和复位信号

inst\_data 读取外部rom得到的16位指令

branch\_pc 跳转命令地址

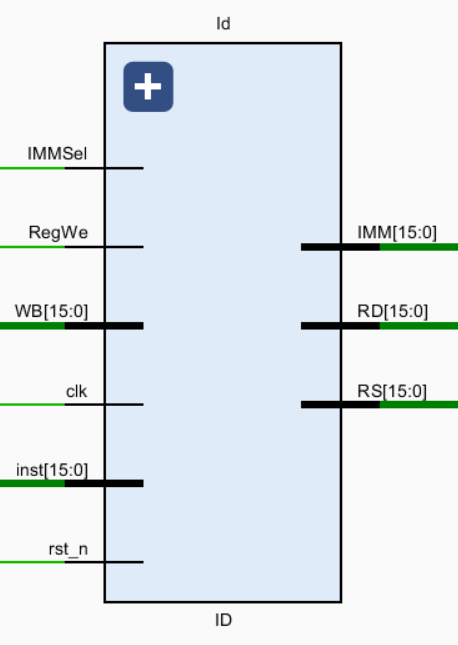
PCSel 命令选择信号，控制下一条信号是输入的跳转命令，还是顺序读取。该信号由ctrl块生成

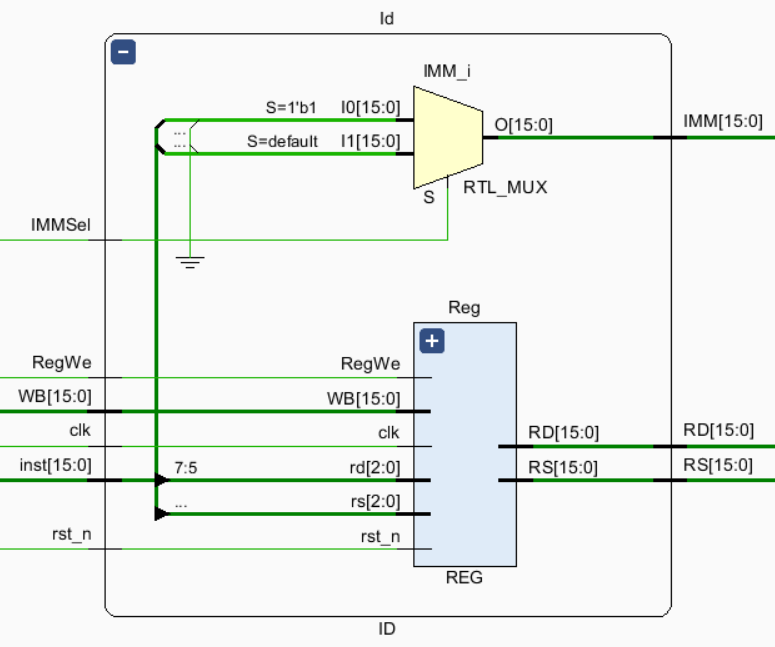
~~irq 中断信号（）~~

**输出**：

inst\_addr 程序指针，读取外部rom的地址（实际上就是pc指针）

**（4）ID**





**译码模块：**

主要分为两块，上方多路复用通过IMMsel选择输出指令中的5位立即数还是8位立即数。下方是寄存器堆（通用寄存器），解码指令中的rs rd，输出rs rd指向寄存器的值RD RS；并将WB数据写入到rd指向的寄存器。

**输入：**

clk & rst\_n 时钟和复位信号

WB WB模块写回寄存器的数据

IMMSel 控制输出是指令中的5位立即数，还是8位立即数（会自动将立即数扩展为16位）

inst 传入指令，并从指令中解码出寄存器指针

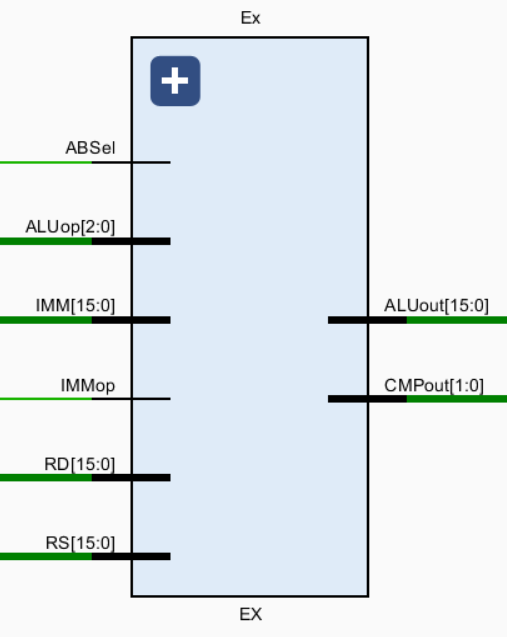
RegWe 寄存器写入使能信号

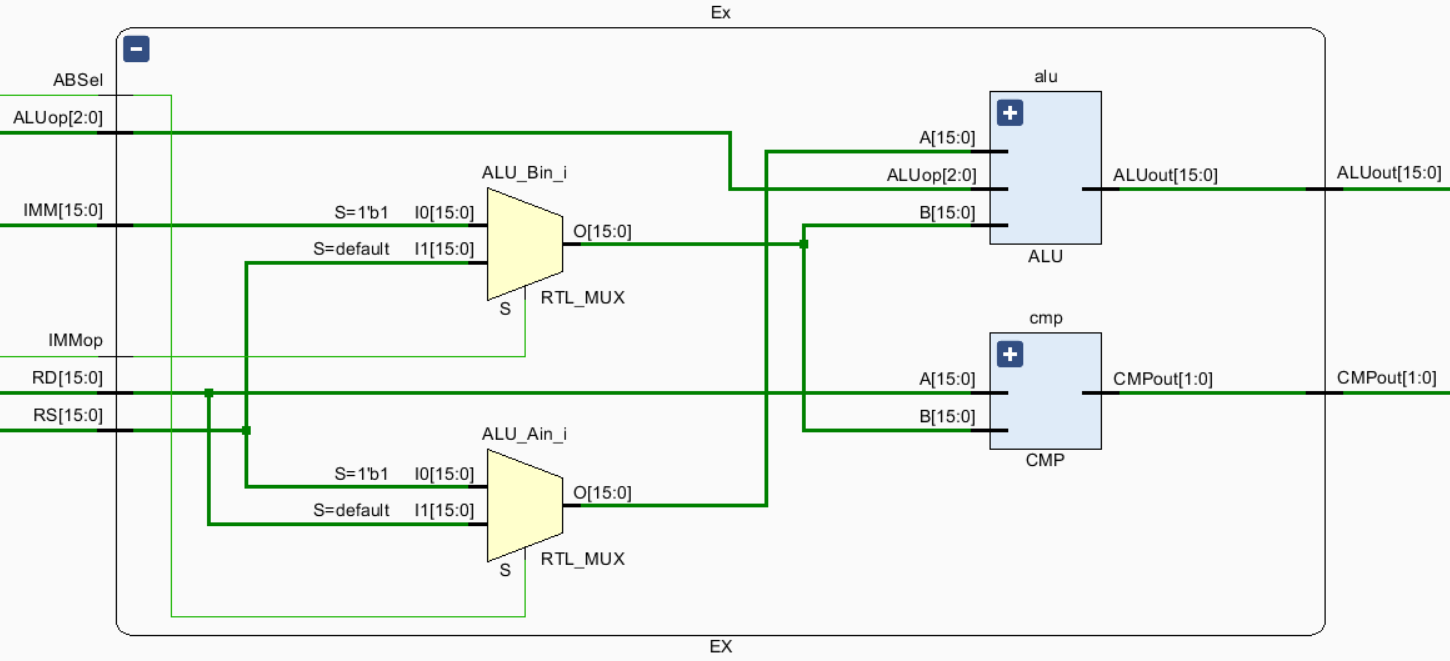
**输出：**

IMM 解码得到的立即数

RD & RS 解码rs，rd得到的寄存器中值

**（5）EX**





**执行模块：由ALU，CMP，MUX构成**

经过IMMop选择后，将输入的数据计算或比较出结果。

**输入：**

RD RS IMM 输入执行模块的数据：rs rd指向的数据以及立即数

ALUop ALU执行的操作(加、减、and、or、xor、左移、右移）

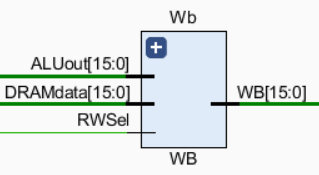
IMMop 控制输入给alu和cmp的数据是立即数，还是寄存器中的数

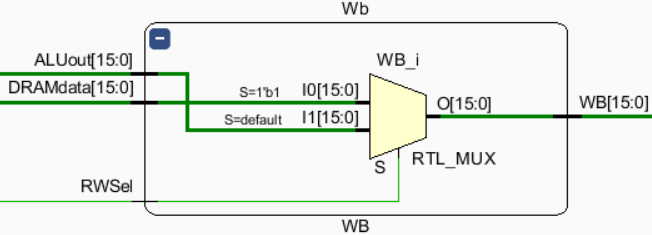
**输出：**

ALUout ALU计算结果

CMPout 比较结果

**（6）WB**





**写回模块：**

一个多路复用模块，用于选择将哪个数据写回到ID模块的寄存器堆中。

**输入：**

ALUout 计算结果

DRAMdata lw命令，从cpu外部读取的数据

RWSel 控制多路复用器，选择写回的数据类型

**输出：**

WB 写回寄存器的数据

**4. BUS结构**

**（1）整体结构**



**BUS总体结构**

cpu为哈佛结构，指令和地址分开读取。cpu与BUS有两组线，数据通路：地址（addr）、数据（data）、控制（ctrl）；指令通路：地址（inst\_addr）、数据（inst\_data）。指令通路直接与BUS中的rom相连。数据通路与ram和其他外设相连，通过addr的地址与设备基地址比较来使能相应功能。BUS的其他线路直接直接连接外设：led、switch、bt、uart、tube。

**输入：**

clk & rst\_n 时钟和复位信号

bt\_rx 蓝牙接收口

uart\_rx uart接收口

ctrl BUS读写控制，1为写，0为读

addr BUS读写地址

data BUS读写数据(这是一个双端口，既可以输出也可以输入)

inst\_addr 指令地址

switch 外部拨码开关（8个）

**输出：**

bt\_tx 蓝牙发送口

uart\_tx uart发送口

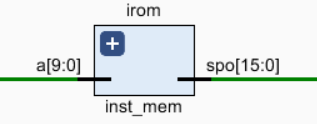
inst\_data inst\_addr读取外部rom得到的指令内容

~~irq\_bt irq\_timer irq\_uart 蓝牙、定时器、uart中断~~

led led接口（8个）

tube\_en seg\_led1 seg\_led2 数码管接口（4个数码管为一组，有两组）

**（2）ROM**



指令储存。例化一个rom ip核，写入coe初始化文件。

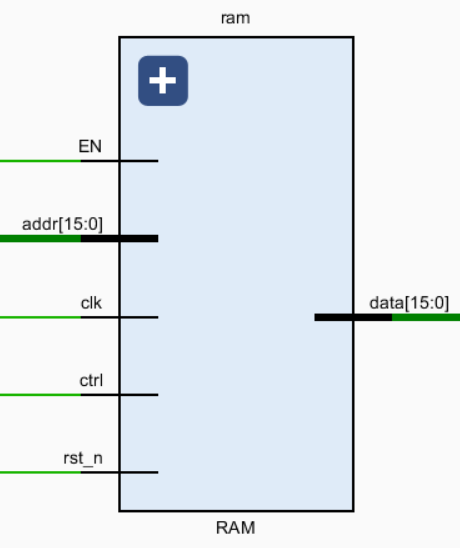
**输入：**

a 连接inst\_addr

**输出：**

spo 输出相应的指令内容。直接连接inst\_data

**（3）RAM**



指令储存。例化一个rom ip核，写入coe初始化文件。

**输入：**

clk & rst\_n 时钟和复位信号

EN 模块使能信号

addr 暂时没用，主要是用来读写ram内部寄存器，但目前内部无配置寄存器

ctrl 控制是读数据还是写数据

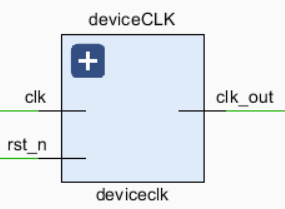
**输入&输出：**

data 输入或输出数据

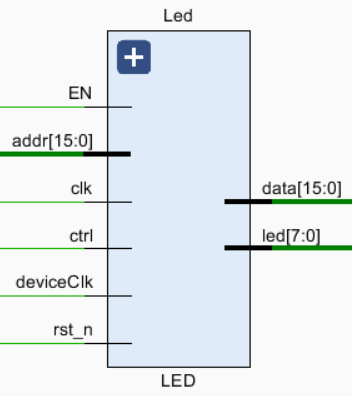
**（4）外设**

注意：外设led、tube等设备的控制时钟频率不能太高。需要将100MHz的主频分频后共给外设设备。

**分频模块：**将输入时钟分频



外设模块（led为例；与前面ram类似）：



**LED模块**

控制8个led灯的亮灭。

**输入：**

clk & rst\_n 时钟和复位信号

deviceClk 外设时钟

EN 模块使能信号

addr 暂时没用，主要是用来读写led设备内部寄存器，但目前内部无配置寄存器

ctrl 控制是读数据还是写数据（对于led，不能读数据，只能写）

**输出：**

led 直接连接上外部led引脚

**输入&输出：**

data 输入或输出数据