10月21日 晚上9:55

设计文档

	op	fun
add	000000	100000
sub	000000	100010
ori	001101	/
lw	100011	/
sw	101011	/
beq	000100	/
lui	001111	/
nop	000000	000000

①或门弄错了

② memtoreg写错了

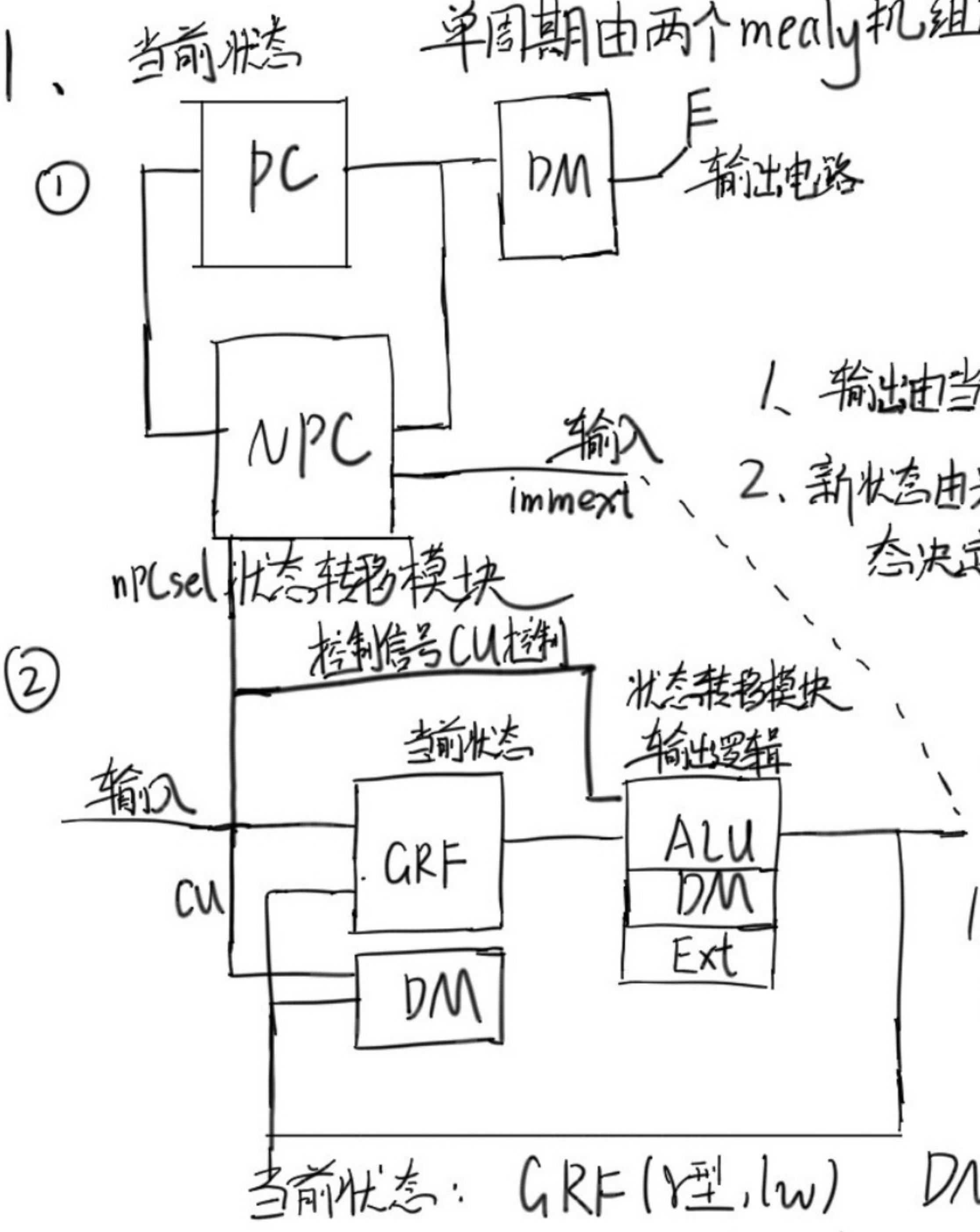
00	加
01	减
10	或

ExtOp

00	无符号
01	有符号
10	lui

设计文档：

1、 单周期由两个mealy机组成



1、输出由当前状态决定

2、新状态由另一个FSM的输出作为输入与当前状态决定

部分输出作为第一机的输入

1、在第二台FSM机中状态转移模块与输出逻辑重合有部分

当前状态： GRF(R型，lw)　DM(sw)

输出逻辑： Ext (beq/j)，输入通过CU分析的nPCSel

状态转移模块： ALU，Ext ⟶ GRF(r)，DM(sw)

ALU，Ext，DM ⟶ GRF(lw)

可见该状态机各模块功能很多变，且每个模块不单一，但仍有规律可循，两个FSM复合后(输入—输出)即为CPU基本框架

2、 IM　ROM (Read Only Memory) 只读不写　外存

DM　RAM (Random Access Memory) 可读可写　内存

GRF　Reg

我认为是合理的，RAM快于ROM，可快捷对RAM进行修改；ROM通常用作存储指令，需要人为加载进数据，不许在运行中更改；而直接使用寄存器是效率最高的方法，故最频繁调用的GRF应使用Reg.

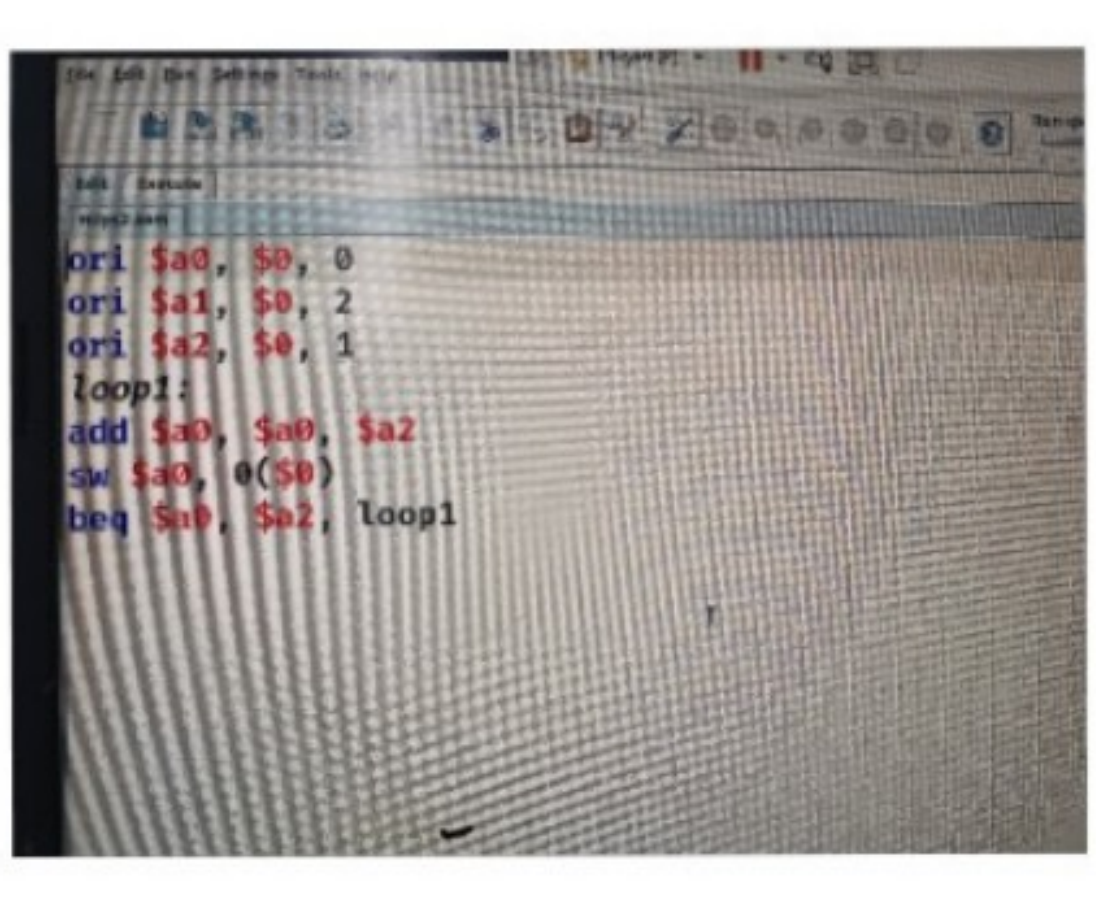
3. 无 (可以将CU拆成 MainController && ALUController)

4、 即 RegWrite，MemWrite均为0，不会写入DM/GRF 对过程无影响 执行nop时

5、 加一个模块，当地址大于0x3000时减去0x3000

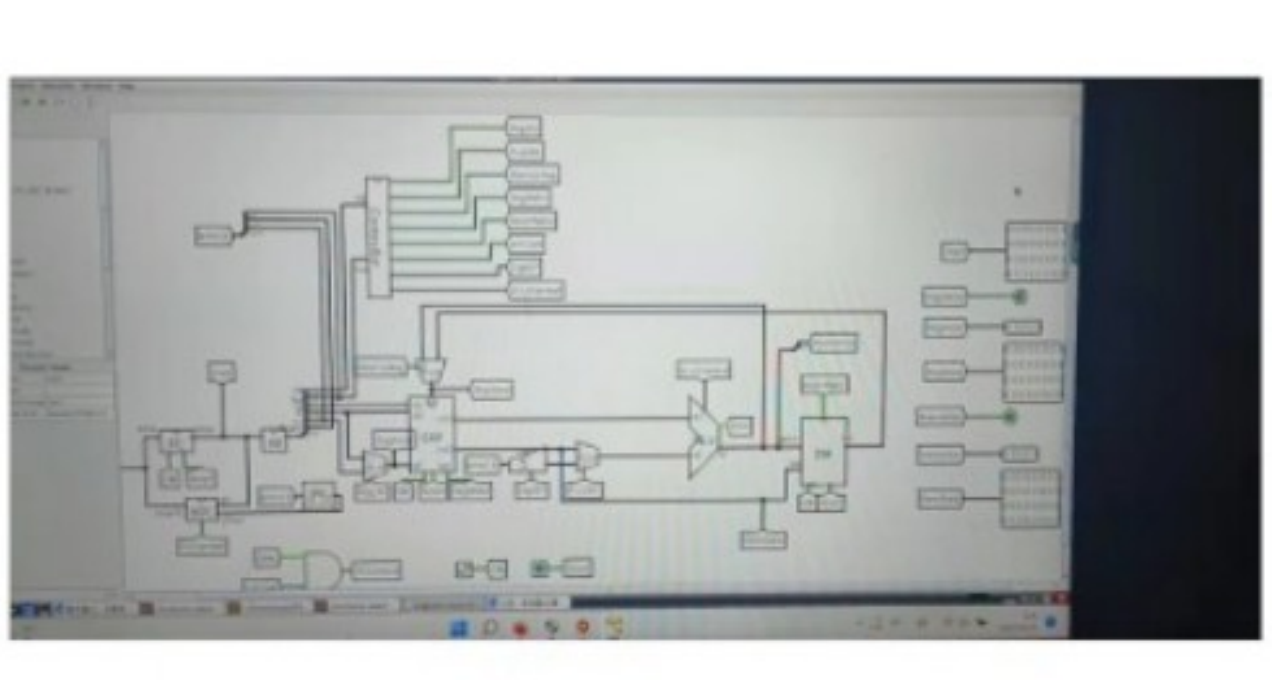
6、 总体较为合理，应再测试与$0相关的指令，beq跳转之前的指令，由于目前不用考虑溢出问题，因此不用对极大极小数特殊验证

测试方案：由于单周期CPU的指令独立性，只要保证每条指令正确性即可保证整体正确性，因此，我根据网站给出的单指令测试样例，并针对beq成功跳转与否及往前/后跳转额外进行测试，最后使用给出的综合测试集完成验证



```
ori $a0, $0, 0
ori $a1, $0, 2
ori $a2, $0, 1
loop1:
add $a0, $a0, $a2
sw $a0, 0($0)
beq $a0, $a2, loop1
```

3. 设计方案



按照模块化思想，将核心功能器件PC，NPC，IM，GRF，ALU，DM，CU，Ext独立封装，高内聚低耦合，用清晰直观的数据通路相连，并在难以连线处使用tunnel简化；在CU设计上选择将ALU控制与主控制合二为一，依据先与后或原则进行解码，格式化，规范化，具有极高的可扩展性，易于增加其它指令。