# 单周期 CPU 设计报告

## 一、模块规格

### 1. IFU

（1）端口说明

| 端口名 | 方向 | 描述 |
|---|---|---|
| CLK | I | 时钟信号 |
| RESET | I | 异步复位信号 |
| PC_in | I | 原 PC 信号 |
| PC_out | O | PC+4 后的信号 |
| INSTRUCTION[31:0] | O | 输出 32 位指令 |

（2）功能定义

| 序号 | 功能名称 | 功能描述 |
|---|---|---|
| 1 | 复位 | 当 RESET 信号为 1 时，对保存 PC 地址的寄存器进行异步清零操作。 |
| 2 | 取指 | 以 PC 的值为地址，从 IM 中取出对应指令 |

### 2. GRF

（1）端口说明

| 端口名 | 方向 | 描述 |
|---|---|---|
| CLK | I | 时钟信号 |
| RESET | I | 异步复位信号 |
| RegWrite | I | 写入使能 |
| Read_Reg1 | I | 读取 Rs |
| Read_Reg2 | I | 读取 Rt |
| Write_Reg | I | 读取存储地址 |
| Write_Data | I | 读取存储数据 |
| Read_Data | O | Rs 输出 |

| | | |
|---|---|---|
| Read_Data2 | O | Rt 输出 |

（2）功能定义

| 序号 | 功能名称 | 功能描述 |
|---|---|---|
| 1 | 复位 | 当 RESET 信号为 1 时，对 GRF 中所有寄存器进行异步清零操作。 |
| 2 | 写入 | Regwrite=1 时，将数据写入 |
| 3 | 读取 | 将 Rs，Rt 输出 |

3、ALU

（1）端口说明

| 端口名 | 方向 | 描述 |
|---|---|---|
| Input1[31:0] | I | 输入 1 |
| Input2[31:0] | I | 输入 2 |
| ALUOp[31:0] | I | 选择计算功能 |
| Equ | O | 输入是否相等 |
| Ans[31:0] | O | 输出 |

（2）功能定义

| 序号 | 功能名称 | 功能描述 |
|---|---|---|
| 1 | 逻辑运算 | ALUOp=00，C=A&B<br>ALUOp=01，C=A\|B<br>ALUOp=10，C=A+B<br>ALUOp=11，C=A-B |
| 2 | 相等判断 | 若 A=B，则 Equ=1<br>若 A≠B，则 Equ=0 |

### 4、EXT

#### （1）端口说明

| 端口名 | 方向 | 描述 |
| --- | --- | --- |
| Imm16[15:0] | I | 16 位立即数输入 |
| ExtOp | I | Ext 功能选择 |
| LuiOp | I | Lui 功能选择 |
| Ext32 | O | 32 位数输出 |

#### （2）功能定义

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 符号拓展 | ExOp=0，无符号拓展<br>ExOp=1，有符号拓展 |
| 2 | Lui 指令功能 | LuiOp=1，将立即数添加至 32 位数高位，低位补零 |

### 5、DM

#### （1）端口说明

| 端口名 | 方向 | 描述 |
| --- | --- | --- |
| CLK | I | 时钟信号 |
| RESET | I | 异步复位信号 |
| Addr[5:0] | I | 地址读入 |
| Input[31:0] | I | 数据读入 |
| MemWrite | I | 写入使能 |
| MemRead | I | 读取使能 |
| Data[31:0] | O | 32 位数据输出 |

#### （2）功能定义

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 写入 | 当写入使能=1 时，将 Input 写入 Addr 对应的寄存器 |
| 2 | 读取 | 当读取使能=1 时，将 Addr 对应的寄存器的值输出 |

6、Controller

（1）端口说明

| 端口名 | 方向 | 描述 |
| --- | --- | --- |
| OP | I | Operation |
| FUN | I | Function |
| Regdst | O | GRF 写入地址 MUX 控制<br>0：写入至 Rt<br>1：写入至 Rd |
| ALUsrc | O | ALU 输入 MUX 控制<br>0：input2 = Rt<br>1：input2 = ext32 |
| MemtoReg | O | GRF 写入数据 MUX 控制<br>0：写入 DM 的输出<br>1：直接输出 ALU 结果 |
| RegWrite | O | GRF 写入使能 |
| MemWrite | O | DM 写入使能 |
| MemRead | O | DM 输出使能 |
| ExtOp | O | Ext 使能信号 |
| LuiOp | O | Lui 使能信号 |
| BeqOp | O | Beq 使能信号 |
| ALUOp[1:0] | O | ALU 功能选择信号 |

（2）功能定义

| 序号 | 功能名称 | 功能描述 |
| --- | --- | --- |
| 1 | 控制 | 根据 Op 和 Fun 输出相应指令与 MUX 的控制信号 |

## 二、控制器设计

### 1. 端口说明

| 端口名 | 方向 | 描述 |
|---|---|---|
| OP | I | Operation |
| FUN | I | Function |
| Regdst | O | GRF 写入地址 MUX 控制<br>0：写入至 Rt<br>1：写入至 Rd |
| ALUsrc | O | ALU 输入 MUX 控制<br>0：input2 = Rt<br>1：input2 = ext32 |
| MemtoReg | O | GRF 写入数据 MUX 控制<br>0：写入 DM 的输出<br>1：直接输出 ALU 结果 |
| RegWrite | O | GRF 写入使能 |
| MemWrite | O | DM 写入使能 |
| MemRead | O | DM 输出使能 |
| ExtOp | O | Ext 使能信号 |
| LuiOp | O | Lui 使能信号 |
| BeqOp | O | Beq 使能信号 |
| ALUOp[1:0] | O | ALU 功能选择信号 |

### 2. 指令真值表

| Instr | addu | subu | ori | lw | sw | beq | lui |
|---|---|---|---|---|---|---|---|
| Op | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 |

| Func | 100001 | 100011 | NA | NA | NA | NA | NA |
|---|---|---|---|---|---|---|---|
| Regdst | 1 | 1 | 0 | 0 | X | X | 0 |
| ALUsrc | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| MemtoReg | 1 | 1 | 1 | 0 | X | X | 1 |
| RegWrite | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| MemWrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| MemRead | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| ExtOp | X | X | 0 | 1 | 1 | 0 | X |
| LuiOp | X | X | 0 | 0 | 0 | 0 | 1 |
| BeqOp | X | X | X | X | X | 1 | X |
| ALUOp | 10 | 11 | 01 | 10 | 10 | XX | 10 |

## 三、测试数据

A：测试 beq,ori,subu

```
ori $t0,1
ori $t1,1
beq $t0,$t1,jump
subu $t0,$t0,$t1
jump:
subu $s0,$s1,$t1
```

```
    nop
```

期望输出：

```
    $t0=0x0000_0001

    $t1=0x0000_0001

    $s0=0xffff_ffff
```

B:测试 sw,lw,ori,addu

```
    ori $t0,2

    ori $t1,3

    ori $t2,4

    addu $s0,$zero

    sw $t0,0($s0)

    sw $t1,4($s0)

    addu $s0,$t2,$s0

    lw $t1,0($s0)

    lw $t0,-4($s0)
```

期望输出：

```
    $t0=0x0000_0002

    $t1=0x0000_0003

    $s0=0x0000_0004
```

C：测试极端数据

```
lui $t0,65535
ori $t0,$t0,65535
```

期望输出：

```
$t0=0xffff_ffff
```

## 四、思考题

1、若 PC（程序计数器）位数为 30 位，试分析其与 32 位 PC 的优劣：

答：

优点：省去 32 位 PC 每次都需要自加 4，左移 2 位的过程，每次只需 PC 自加 1，并且可以直接将低 5 位接至 ROM，简化了电路。

缺点：机器字长为 4 个字节，及 32 位，且在多周期中，将 IM 与 DM 合并时，若字长不统一，则会出现问题。

2、现在我们的模块中 IM 使用 ROM， DM 使用 RAM， GRF 使用寄存器，这种做法合理吗？ 请给出分析，若有改进意见也请一并给出：

答：

显然是合理的。

（1）将 CPU 主存分成 IM 与 DM 两部分，对于单周期 CPU，简化了电路复杂度。

（2）IM 使用 ROM，保证了指令存储的只读性与安全性

（3）DM 使用 RAM，并使用双端口模式，即设置 RAM 的 Data Interface 属性为 Separate load and store ports，有效控制了写入和读出的信号。

（4）GRF 使用寄存器 Register，相较于基础 RS,JK 触发器与锁存器，功能更完整与一体化，方便设计。

3、结合上文给出的样例真值表，给出 RegDst， ALUSrc， MemtoReg，RegWrite, nPC_Sel, ExtOp 与 op 和 func 有关的布尔表达式（表达式中只能使用“与、或、非”3 种基本逻辑运算。）

答：

Add=func[5]~func[4]~func[3]~func[2]~func[1] ~func[0]~op[5] ~op[4] ~op[3] ~op[2] ~op[1] ~op[0]

Sub= func[5]~func[4]~func[3]~func[2]func[1] ~func[0]~op[5] ~op[4] ~op[3] ~op[2] ~op[1] ~op[0]

Ori=~op[5] ~op[4] op[3] op[2] ~op[1] op[0]

Lw=op[5] ~op[4] ~op[3] ~op[2] op[1]op[0]

Sw=op[5] ~op[4] op[3] ~op[2] op[1] op[0]

Beq=~op[5] ~op[4] ~op[3] op[2] ~op[1] ~op[0]

RegDst=add+sub

ALUsrc=ori+lw+sw

MemtoReg=lw

RegWritw=add+sub+ori+lw

MemWrite=sw

nPC_sel=beq

ExtOp=lw+sw

4、充分利用真值表中的 X 可以将以上控制信号化简为最简单的表达式，请给出化简后的形式。

答：Add=func[5] ~func[4]~func[3]~func[2]~func[1] ~func[0]~op[5] ~op[4] ~op[3] ~op[2] ~op[1] ~op[0]

Sub= func[5]~func[4]~func[3]~func[2]func[1] ~func[0]~op[5] ~op[4] ~op[3] ~op[2] ~op[1] ~op[0]

Ori=~op[5] ~op[4] op[3] op[2] ~op[1] op[0]

Lw=op[5] ~op[4] ~op[3] ~op[2] op[1]op[0]

Sw=op[5] ~op[4] op[3] ~op[2] op[1] op[0]

Beq=~op[5] ~op[4] ~op[3] op[2] ~op[1] ~op[0]

RegDst=add+sub

ALUsrc=ori+lw+sw

MemtoReg=lw

RegWritw=add+sub+ori+lw

MemWrite=sw

nPC_sel=beq

ExtOp=lw+sw

5、事实上，实现 nop 空指令，我们并不需要将它加入控制信号真值表，为什么？请给出你的理由

答：指令位 nop 时，若不将它加入控制信号真值表，则真值表处处为 0，相等于所有使能信号都置 0，满足条件。

6、前文提到，“可能需要手工修改指令码中的数据偏移”，但实际上只需再增加一个 DM 片选信号,就可以解决这个问题。请阅读相关资料并设计一个 DM 改造方案使得无需手工修改数据偏移：

答：由于 DM 低 2 位无用，且输入地址宽度位 5 位，则可以以 32-2-5=15，及高 15 位作为片选信号，当高 15 位为某一特定值时，片选信号有效，从而解决偏移的问题

7、除了编写程序进行测试外，还有一种验证 CPU 设计正确性的办法——形式验证。 形式验证的含义是根据某个或某些形式规范或属性，使用数学的方法证明其正确性或非正确性。请搜索“形式验证(Formal Verification)"了解相关内容后，简要阐述相比与测试，形式验证的优劣。

答：

优点：

1. 由于形式验证技术是借用数学上的方法将待验证电路和功能描述或参考设计直接进行比较，因此测试者不必考虑如何获得测试向量。

2. 形式验证是对指定描述的所有可能的情况进行验证，而不是仅仅对其中的

一个子集进行多次试验，因此有效地克服了模拟验证的不足。

3. 形式验证可以进行从系统级到门级的验证，而且验证时间短，有利于尽早、尽快地发现和改正电路设计中的错误，有可能缩短设计周期。

缺点:

1. 形式验证到目前为止仍然不能有效的验证电路的性能，如电路的时延和功耗等。

2.不能发现代码中的功能错误和时序错误，规模过大会消耗极长时间。