Logisim开发单周期CPU实验报告

13061076赵乐

## 一、模块定义

1.IFU

（1）基本描述

IFU 主要功能是完成取指令功能。 IFU 内部包括 PC、 IM(指令存储器)以及其他相关逻辑。 IFU 除了能执行顺序取值令外，还能根据 BEQ 指令的执行情况决定顺序取值令还是转移取值令。

（2）模块接口

表1

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| IfBeq | I | 当前指令是否为 beq 指令标志。 1：当前指令为 beq 0：当前指令非 beq |
| Zero | I | ALU 计算结果为 0 标志。 1：计算结果为 0 0：计算结果非 0 |
| Clk | I | 时钟信号 |
| Reset | I | 复位信号。 1：复位 0：无效 |
| Instr[31:0] | O | 32 位 MIPS 指令 |

（3）功能定义

表2

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时， PC 被设置为 0x00000000。 |
| 2 | 取指令 | 根据 PC 从 IM 中取出指令。 |
| 3 | 计算下一条指令地址 | 如果当前指令不是 beq 指令，则 PC←如果当前指令是 beq 指令，并且 zero 为 0，则 PC🡨PC+1 如 果 当 前 指 令 是 beq 指 令 ， 并 且 zero 为 1 ， 则 PC🡨PC+sign\_ext(当前指令 15..0) [注]PC 取地址为 4 字节，固低 2 位地址可以去除。 |

2.GPR

（1）基本描述

GPR 以 32 个 32 位具有写使能的寄存器为基础，辅以多路选择器。其主要 功能是对寄存器堆进行存值取值的操作。为了便于测试， GPR 除了以上操作外， 可有清零的操作，即将寄存器的值变为 0x00000000。

（2）模块接口

表3

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Clk | I | 时钟信号。 |
| Reset | I | 复位信号。 1：复位 0：无效 |
| WE | I | 写使能信号。 1：可向 GPR 写入数据 0：只可从 GPR 读出数据 |
| A1 | I | 5 位的地址输入，用于指定 32 个寄存器中的一个。 在单周期 CPU 中为指令的[21:25]即 rs 字段 |
| A2 | I | 5 位的地址输入，用于指定 32 个寄存器中的一个。 在单周期 CPU 中为指令的[16:20]即 rt 字段 |
| A3 | I | 5 位的地址输入，用于指定 32 个寄存器中的一个。 在单周期 CPU 中为指令的[16:20]rt 字段或[11:15]rd字 段之一 |
| WD | I | 32 位的数据输入，当写使能时，可以向寄存器中写入。 |
| RD1 | O | 输出 32 位的 A1 指定的寄存器的数据。 |
| RD2 | O | 输出 32 位的 A2 指定的寄存器的数据。 |

（3）功能定义

表4

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号有效时， 32 个寄存器中的值被置为’h0。 |
| 2 | 读数据 | 根据 5 位地址输入确定寄存器，并将寄存器中的数据 输出。 |
| 3 | 写数据 | 当写使能时，根据 5 位地址输入确定寄存器，并将数 据写入寄存器。 |

3.ALU

（1）基本描述

ALU 的主要功能是对输入到 ALU 的两个数进行加法、减法、按位或等操作。 ALU 内部包括 32 位加法器、减法器等部件。 ALU 除了以上操作外，还需要判 断输入的两个值是否相等。

（2）模块接口

表5

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| InputA | I | 参与 ALU 计算的第一个值 |
| InputB | I | 参与 ALU 计算的第二个值 |
| Aluop[1：0] | I | ALU 功能的选择信号。 00： ALU 进行加法运算 01： ALU 进行减法运算 10： ALU 进行或运算 |
| Result | O | ALU 计算的结果 |
| Zero | O | 1：result 为 0  0：result 不为 0 |

（3）功能定义

表6

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加法运算 | 将 inputA 与 inputB 进 行 加 法 运 算 。 result🡨inputA+inputB |
| 2 | 减法运算 | 将 inputA 与 inputB 进 行 减 法 运 算 。 result🡨inputA-inputB |
| 3 | 或运算 | 将 inputA 与 inputB 进 行 按 位 或 运 算 。 result🡨inputA|inputB |

4.EXT

（1）基本描述

将输入的数据改变位宽后输出，其中变化的为可以用0,1扩充，也可以把符号扩展，还可以额外输入一个数填充。

（2）模块接口

表7

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Input6 | I | 输入16位立即数 |
| Extop | I | 输入控制信号 |
| Output32 | O | 输出改变位宽后的32位数 |

（3）功能定义

表8

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 位扩展 | 改变数的位宽 |
| 2 | 选择扩展方式 | 可以根据信号选择从高位还是低位扩展 |

5.DM

（1）基本描述

数据存储器,为随机存储器，用RAM实现，可读可写，用以存放指令执行过程中的数据变量。

（2）模块接口

表9

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Address | I | 输入需要访问的地址 |
| Input | I | 数据将储存的地址 |
| Chip select | I | 当为0时禁用芯片 |
| Store | I | 当为1时，从输入端存储数据 |
| Clock | I | 当从0到1时更新内存内的值 |
| Clear | I | 当为1时，异步重置内存为0 |
| Load | I | 当为1时将内存加载到输出 |
| Data | O | 输出从地址加载的数据 |

（3）功能定义

表10

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读 | 当读控制信号有效时，地址线选择的存储字被放在读数据输出总线上（组合元件操作） |
| 2 | 写 | 当写控制信号有效且时钟信号clk下跳沿时，写数据总线上的数据 被写入地址选择的存储单元中（状态元件操作）。 |

6.Controller

（1）基本描述

对指令进行译码并生成指令执行所需的控制信号，以实现对数据 通路中各件的功能控制，以及相应路径的开关控制等，是指令的控制部件。

（2）模块接口

表11

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Op[0：5] | I | 输入32位指令的31:26位op段 |
| FUNC[0：5] | I | 输入32位指令的5:0位func段 |
| Aluop[0:1] | O | 指明ALU的运算类型   00： 访存指令所需加法   01 ： beq指令所需减法   10： R型指令功能码决定 |
| RegDst | O | 失效时作用 ：RegDst 寄存器堆写入端地址来选择Rt字段。  有效时作用：寄存器堆写入端地址选择 Rd字段 |
| ALUSrc | O | 失效时作用 ：ALU输入端B选择寄存器堆输出R[rt] 。  有效时作用： ALU输入端B选择Signext输出 |
| MemtoReg | O | 失效时作用 ：寄存器堆写入端数据来自 ALU输出。  有效时作用： 寄存器堆写入端数据来自 DM输出 |
| RegWrite | O | 把数据写入寄存器堆中对应寄存器 |
| MemWrite | O | 数据存储器DM写数据（输入） |
| Branch | O | Branch=1，此时若Zero=1， PC输入选择加法器Nadd  输出（分支指令目的地址），否则选择加法器Add输出（PC+4）  其他指令： Branch=0， PC输入选择加法器Add输出（PC+4） |

（3）功能定义

表12

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 译码 | 将输入的op和func码译成相应的控制信号 |

## 二、单周期控制器真值表

表13

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| func | 10 0000 | 10 0010 | n/a | | | | |
| Op | 00 0000 | 00 0000 | 00 1101 | 10 0011 | 10 1011 | 00 0100 | 001111 |
|  | addu | subu | ori | lw | sw | beq | lui |
| RegDst | 1 | 1 | 0 | 0 | x | x | 0 |
| ALUSrc | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| MemtoReg | 0 | 0 | 0 | 1 | x | x | 0 |
| RegWrite | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| Extop1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Extop0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| MemWrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Branch | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ALUOp1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| ALUOp0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

## 三、测试程序原理

程序代码：



图1

程序分析：

Line1：立即数1与0或，由于初始都为0，结果1赋给t0，此时GPR中有一个寄存器值变为1

Line2：同line1，下一个寄存器变为3

Line3：将寄存器t0的值存入第一个内存地址，地址为0x00002000

Line4：从第一个内存地址中取出数并赋给寄存器t2，相当于下一个寄存器t2值变为1

Line5：寄存器t3的值=t2+t0=2

Line6：寄存器t4的值=t1-t2=2

Line7：将16进制立即数3加载到寄存器t5的高位，t5值为00030000（16进制）

Line8：判断寄存器t3t4内的值是否相等，相等则转移到go1语句再开始执行

Line9：t0=t1-t0=2，如果beq指令执行正确，则不执行词句，即t0没有变为2，仍然是1，所以由此可判断beq是否执行

Line10：t1=5，如果beq指令执行正确，则不执行词句，即t1仍然是3，所以由此可判断beq是否执行

Line11：寄存器s0内的值变为6

Line12：把s0的值（6）存入到第二个内存地址（0x00002020）

Line13：从第一个内存地址中取出数并赋给s1，s1=1

Line14：s2=s0-t1=6-3=3

Line15：如果t1=s0=3，转到go2后开始执行

Line16：把3加载到s0的高位，s0变为00030006（16进制），如果beq指令执行正确，则不执行词句，即s0仍然是6，所以由此可判断beq是否执行

Line17：s3=t2+t3=3

Line18：从第二个内存地址中取出数并赋给寄存器s4，s4=6

Line19：寄存器s5的值=6

Line20：s5=s3+s4=9

程序执行期望：

t0=00000001,t1=00000003,t2=00000001,t3=00000002,t4=00000002,t5=00030000

s0=00000006,s1=00000001,s2=00000003,s3=00000003,s4=00000006,s5=00000009

内存0x00002000：1

内存0x00002020：6

程序执行结果：

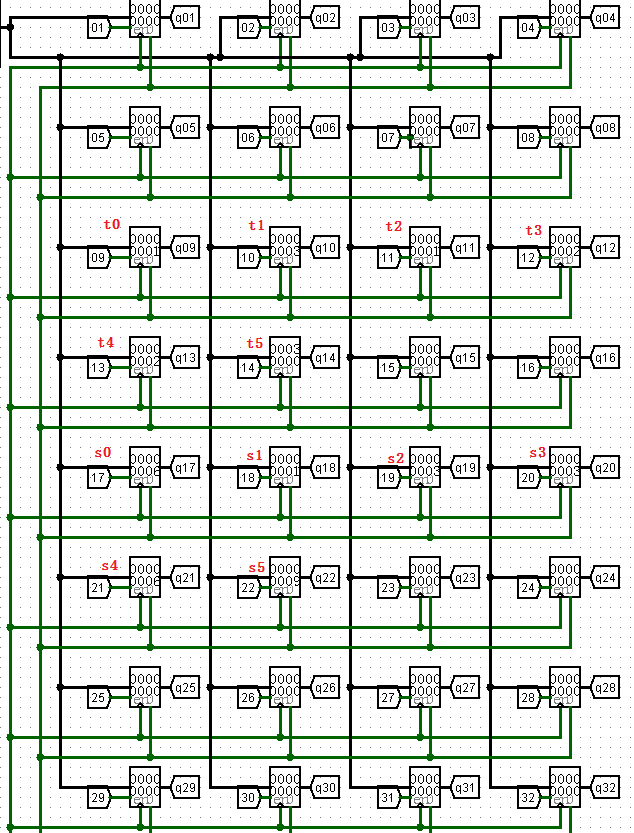


图2

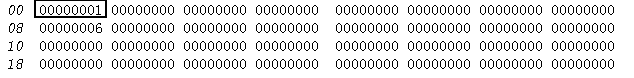


图3

## 四、问答

最简表达式

设op从高位到低位为，func从高位到低位为



1. 所需逻辑门数量表

表14

|  |  |  |  |
| --- | --- | --- | --- |
|  | 2输入与门 | 2输入或门 | 非门 |
| Addu | 11 | 0 | 10 |
| Subu | 11 | 0 | 9 |
| Ori | 5 | 0 | 3 |
| Lw | 5 | 0 | 3 |
| Sw | 5 | 0 | 2 |
| Beq | 5 | 0 | 5 |
| Lui | 5 | 0 | 2 |
| Regdst | 0 | 1 | 0 |
| Alusrc | 0 | 3 | 0 |
| Memtoreg | 0 | 0 | 0 |
| Regwrite | 0 | 4 | 0 |
| Extop | 0 | 1 | 0 |
| Memwrite | 0 | 0 | 0 |
| Branch | 0 | 0 | 0 |
| Aluop | 0 | 2 | 0 |
| 总计 | 47 | 11 | 34 |

b）

答：更喜欢Figure5，Figure6中的方法，因为由上表可知，这样可以大大减少所需逻辑门数量，简化电路，使电路更清晰，而不用将大量精力浪费在分析逻辑门的组合上。