Project1 Logisim完成单周期处理器开发

# 设计说明

1. 处理器应支持的指令集MIPS-Lite：addu，subu，ori，lw，sw，beq，lui，j。
   1. addu，subu可以不支持实现溢出。
2. 处理器为单周期设计。

# 设计要求

1. 顶层设计视图包括如Figure1所示的部件，即Controller(控制器)、IFU(取指令单元)、GPR(通用寄存器组，也称为寄存器文件、寄存器堆)、ALU(算术逻辑单元)、DM(数据存储器)、EXT(扩展单元)、多路选择器及splitter。
   1. 顶层设计视图的顶层有效驱动信号包括且仅包括：clk、reset。
   2. 提示：图中的其他字符均不是端口信号。

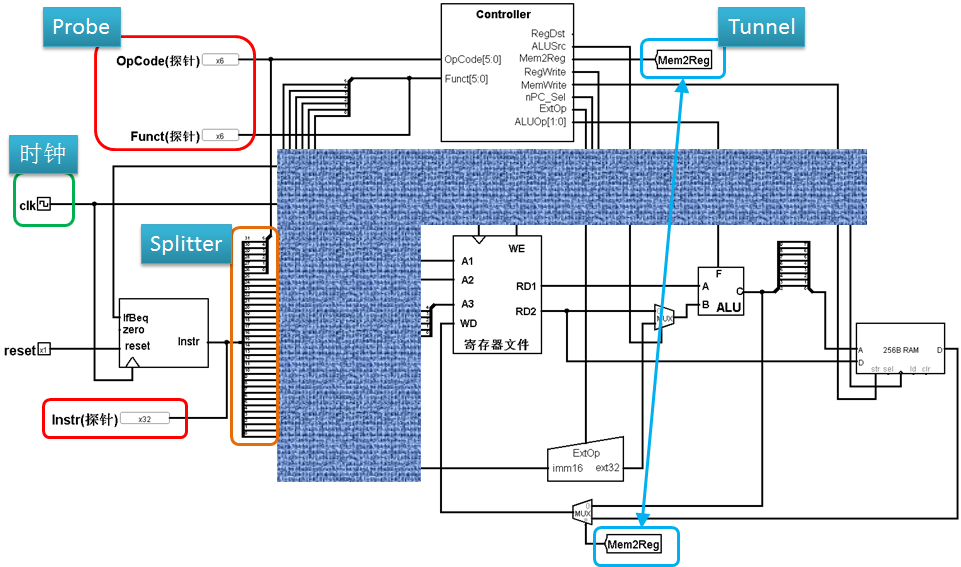


Figure1 顶层设计

* 1. 必须采用模块化和层次化设计。整个设计文件目录结构应类似于Figure2。

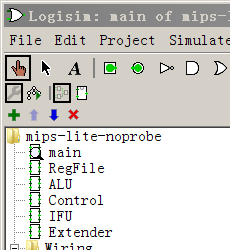


Figure2 设计层次(仅供参考)

1. IFU：内部包括PC、IM(指令存储器)及相关逻辑。
   1. PC：用寄存器实现，宽度为30位。PC应具有复位功能。
   2. IM：容量为32bit×32字，用ROM实现。
   3. 说明：由于IM地址仅为5位，因此请用2个对接的Splitter实现将PC低位地址与IM地址连接。
2. 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。
3. ALU：实现加法及减法时，不允许使用logisim内置的Adder及Subtractor！加法、减法实现必须以门电路为基础。
4. EXT：可以使用logisim内置的Bit Extender。
5. DM：容量为32bit×32字，用RAM实现。
   1. DM应采用双端口模式，即设置RAM的“Data Interface”属性为“Separate load and store ports”。
6. 必须有时钟源，即如Figure1中绿圈所示。
   1. 只有设置了时钟源，系统才能自动运行，从而让程序连续运行。

# 模块定义【WORD】

1. 仿照下面给出的IFU模块定义，给出所有功能部件的模块定义。
   1. IFU、GPR、ALU、EXT、DM、Controller。
2. IFU模块定义(参考样例)
3. 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。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 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 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 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🡨PC+1  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。 |

# 控制器设计

1. 请仿照下图给出MIPS-Lite指令集的单周期控制器真值表。【WORD】

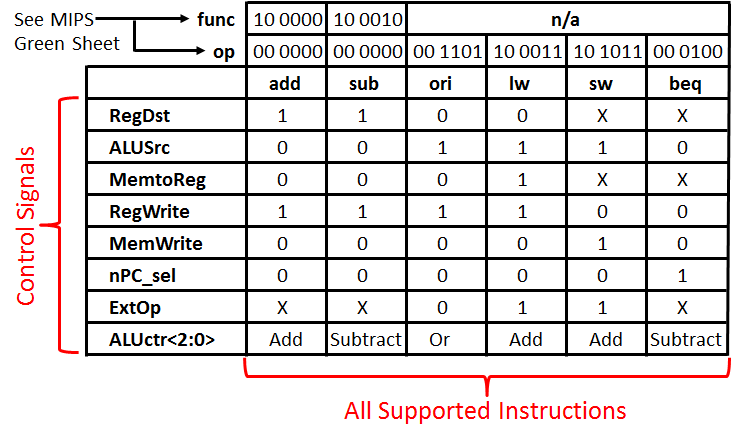


Figure3 控制信号真值表

* 1. 结合真值表，请给出数据通路每个功能部件的每个控制信号的布尔表达式。
  2. 表达式中只能使用“与、或、非”3种基本逻辑运算。
  3. 每个控制信号的表达式应该是指令opcode域与funct域的函数。
  4. 对于多位的控制信号(如ALUCtr)，应诸位给出其逻辑表达式。

1. 请在logisim中完成控制器设计。
   1. 控制器整体结构需要仿照Figure 4实现。

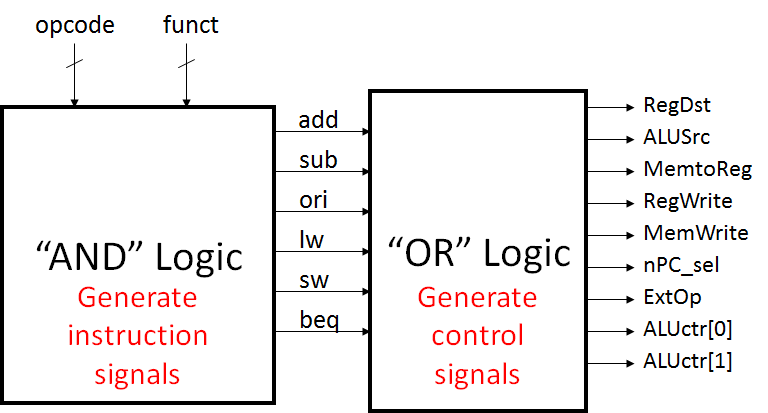


Figure 4 控制器基本结构

* 1. 控制信号必须仿照下图方式实现。

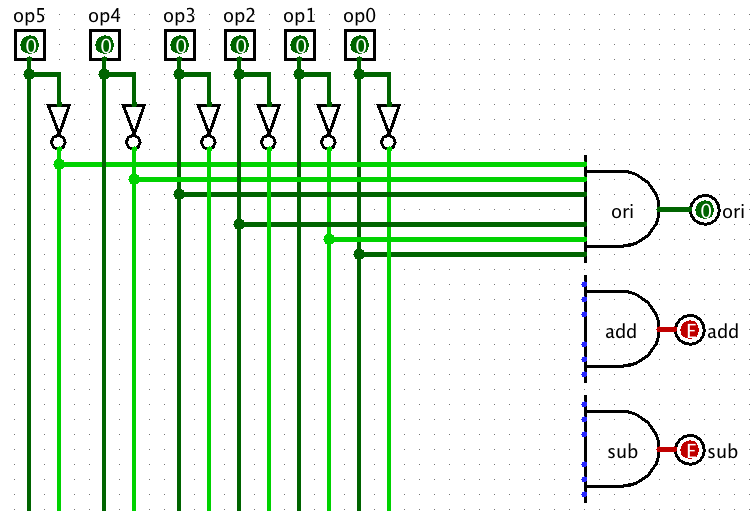


Figure5 与阵列：译码产生指令标识

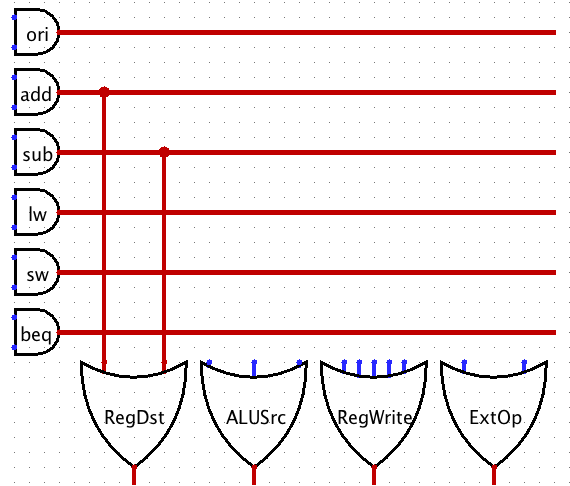


Figure6 或阵列：产生最终的控制信号

# 测设要求

1. 所有指令都应被测试充分。
2. 构造1个至少20条以上指令的测试程序，并加载至IFU中运行通过。
   1. MIPS-Lite定义的每条指令至少出现1次以上。
   2. 演示时，测试程序必须已经通过IFU中的IM的“Load Image”加载完毕。
3. 详细说明你的测试程序原理。【WORD】
   1. 应明确说明测试程序的测试期望，即应该得到怎样的运行结果。
   2. 每条汇编指令都应该有注释。

# 问答【WORD】

1. 请充分利用Figure3中的X可以将控制信号化简为最简单的表达式。
2. 对于Figure5、Figure6中的与或阵列来说，1个3输入与门最终转化为2个2输入与门，1个4输入与门最终转化为3个2输入与门，依次类推。或阵列也类似计算。那么
   1. 请给出采用Figure5、Figure6中的方法设计的每个控制信号所对应的2输入与门、2输入或门、非门的数量。
   2. 请与第17项对比，你更喜欢哪种设计方法。为什么

# 其他要求

1. 打包文件：Logisim工程文件、测试程序二进制文件、项目报告。
2. 时间要求：实验指导教师指定。
3. 本实验要求文档中凡是出现了【WORD】字样，就意味着该条目需要在实验报告中清晰表达。

# 实验测试要求

1. 实验成绩由下列部分组成：回答问题、MIPS-Lite处理器正确性、增加新指令后的处理器正确性等。
2. 实验测试时，你需要展示你的设计并证明其正确性。
3. 实验指导教师会临时增加1～2条指令，你需要在规定的时间内完成对原有设计的修改，并通过实验指导教师提供的测试程序。

# 开发与调试技巧

1. 对于每条指令，请认真阅读《MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set》！
   1. 如果测试时，你无法清楚的解释所要求的指令，测试成绩将减一档！
2. Figure1中Tunnel的用途是将具有相同name的tunnel连接在一起。Tunnel可以避免将图画的很乱。
3. Figure1中Probe的用途是显示被probed信号的值，便于调试。
4. Figure1中Splitter的用途是从某组信号中提取其中部分信号。例如，IFU输出32位指令，需要提取高6位(OpCode)和低6位(Funct)分别输入controller。
   1. splitter是有位序的！但字号太小，需要放大设计图(界面左下有比例设置)。
   2. 建议高位永远在上，低位永远在下
5. 如果你对于logisim内置的某个部件的端口不明白，请：
   1. 仔细阅读Help🡪Library Refrence关于该部件的描述。
   2. 放大logisim显示比例直至能清晰看到代表部件的各个端口的圆点，然后将鼠标停留相应的圆点上，就可以读取端口具体信息。
6. 建议先在MARS中编写测试程序并调试通过。
   1. 注意MARS中的“Settings🡪Memory Configuration”只能配置指令存储器起始地址为0地址，而不能将指令存储器和数据存储器的起始地址均配置为0地址！
   2. 由于logisim设计中的DM起始地址为0，因此请仔细观察所用到的指令，在把MARS中调试通过的二进制码导出后，你可能需要手工修改指令码中的数据偏移。
   3. 提示：事实上，在现代主流计算机中，数据存储器和指令存储器的起始地址不应该重叠。但在本设计中，由于采用分离存储器设计方案，因此可以暂时忽略这一点。
7. 当然，如果你能再自学一点点存储器译码的知识，那么只需再增加一个DM片选信号，一切都搞定了(就不需要再考虑第30.b)了)。
   1. 片选信号就是对指令发出的数据存储器地址的高位分析。
   2. 假设DM有256MB容量，并且映射在0x3000\_0000～0x3FFF\_FFFF区间。那么只需要把高4位地址与0x3进行比较，比较结果就是DM的片选信号。
   3. Logisim内置的RAM有片选信号！
8. 提示：你可以考虑增加7段数码管等输入输出来让你的测试结果更加直观。
   1. 本条非必做要求。
   2. 7段数码管也需要类似片选等信号，其工作原理与第31项类似。