图片包含 徽标

描述已自动生成

**设计报告**

**题目 单周期CPU控制器电路设计**

**姓名**

**学号**

**摘 要**

本文旨在设计一个基于Risc-V指令集的单周期CPU控制器电路，采用组合逻辑电路的方式，自己选择芯片并完成电路设计，用Altium Designer（AD）软件绘制电路。设计重点和主要创新点如下：

（1） opcode编码译码方式的讨论和相关芯片设计。

（2） 根据逻辑表达式对电路的输入端进行优化。

（3） 电路芯片的选择。

本文的最后对此次设计进行了总结、分析和评估。

**目 录**

[1 设计目标 4](#_Toc152146615)

[2 方案设计 4](#_Toc152146616)

[2.1 信号设计与编码 4](#_Toc152146617)

[2.1.1 信号设计 4](#_Toc152146618)

[2.1.2 信号编码 5](#_Toc152146619)

[2.2 电路设计 6](#_Toc152146620)

[2.2.1 输入处理 6](#_Toc152146621)

[2.2.2 逻辑表达式设计与输入优化 7](#_Toc152146622)

[2.2.3 opcode译码设计 8](#_Toc152146623)

[2.2.4 芯片选择 9](#_Toc152146624)

[3 原理图绘制 10](#_Toc152146625)

[3.1 opcode译码器原理图绘制 10](#_Toc152146626)

[3.2 控制器电路原理图绘制 11](#_Toc152146627)

[4 总结、分析与评估 12](#_Toc152146628)

# 1 设计目标

设计一个基于Risc-V指令集的单周期CPU控制器电路，采用组合逻辑电路的方式，自己选择芯片并完成电路设计，用Altium Designer（AD）软件绘制电路。

控制器真值表如下，其中（R-R op）指令为R-type指令，只需实现其中的add，sub，and和or指令，其中add和sub指令真值表已经给出，and和or指令真值表见图2。

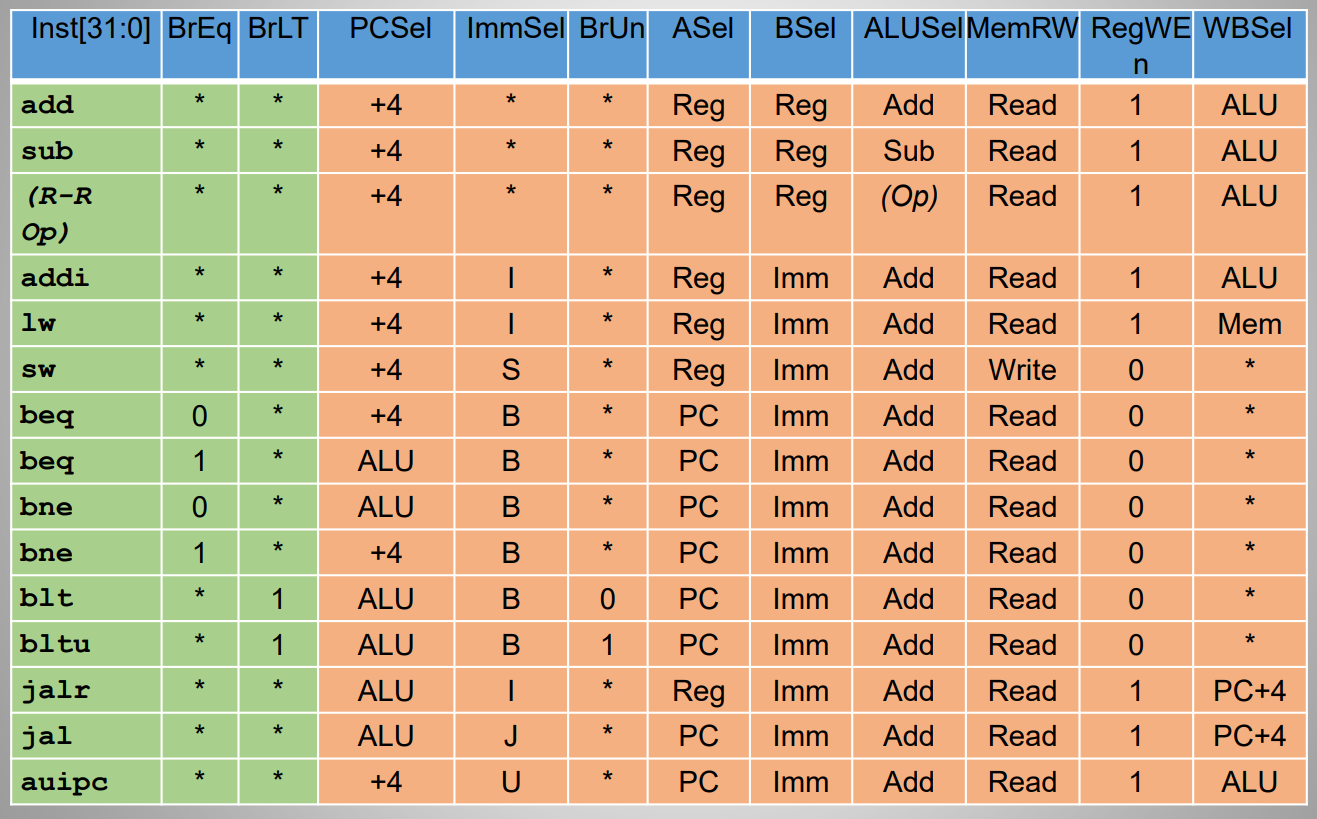


图1 控制器真值表

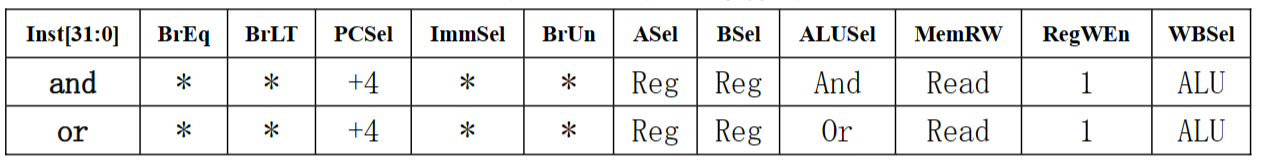


图2 and，or真值表

# 2 方案设计

## 2.1 信号设计与编码

### 2.1.1 信号设计

输入输出信号设计如表一所示，宽度多于一位的信号的编码将在后表给出。

|  |  |  |
| --- | --- | --- |
| 输入信号 | | |
| 信号 | 宽度 | 含义 |
| Inst[31:0] | 32 | Instruction. |
| BrEq | 1 | Set to 1 if the two values are equal. |
| BrLT | 1 | Set to 1 if the value in rs1 is less than the value in rs2. |
| 输出信号 | | |
| 信号 | 宽度 | 含义 |
| PCSel | 1 | Selects the ALU input for all B-type instructions where the branch is taken (according to the branch comparator output) and all jumps. Selects the PC+4 input for all other instructions. |
| ImmSel | 3 | Selects the instruction format so the immediate generator can extract the immediate correctly. The default encoding is [0b000 = I], [0b001 = S], [0b010 = B], [0b011 = U], [0b100 = J], though you're welcome to pick your own. |
| BrUn | 1 | 1 if the branch instruction is unsigned, and 0 if the branch instruction is signed. Don't care for all other instructions. |
| ASel | 1 | Selects whether to send the data in RegReadData1 or the PC to the ALU. |
| BSel | 1 | Selects whether to send the data in RegReadData2 or the immediate to the ALU. |
| ALUSel | 4 | Selects the correct operation for the ALU. |
| MemRW | 1 | 1 if the instruction writes to memory, and 0 otherwise |
| RegWEn | 1 | 1 if the instruction writes to a register, and 0 otherwise. |
| WBSel | 2 | Selects whether to write the memory read from DMEM, the ALU output, or PC+4 to rd. |

表1 信号位数设计和说明

### 2.1.2 信号编码

Inst[31:0]是Risv-V指令集的指令信号，参考Risv-V的划分（见图3），本设计需要[31:25]、[14:12]和[6:0]位，分别是func7、func3和opcode；进一步地根据设计要求的指令设计后（[2.2.2](#_2.2.2_逻辑表达式设计与输入优化)），最后用到了 [14:12]和[6:2]位。

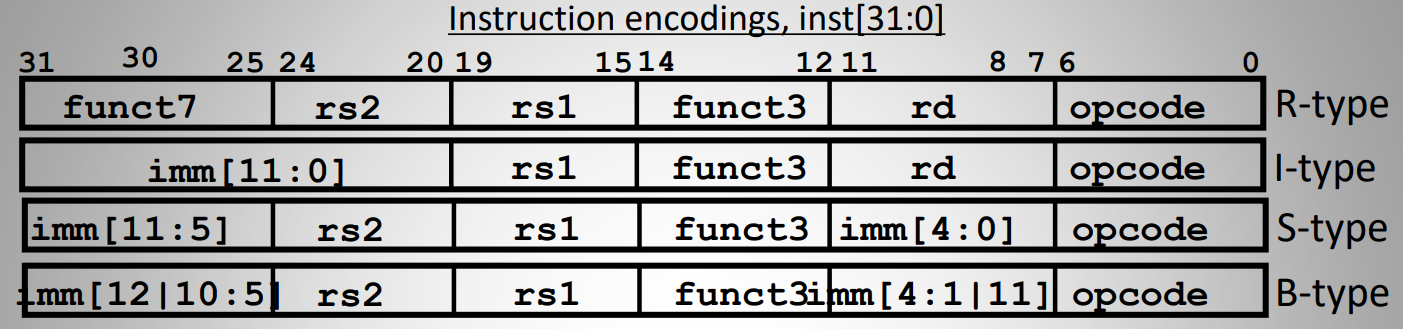


图3 指令编码表

ImmSel[2:0]需要根据指令的format让立即数产生器正确地提取立即数。表1显示默认编码如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Type | ImmSel | Imm[31:20] | Imm[19:12] | Imm[11] | Imm[10:5] | Imm[4:1] | Imm[0] |
| I | 3’b000 | Inst[31] | | | Inst[31:20] | | |
| S | 3’b001 | Inst[31] | | | Inst[30:25] | Inst[11:7] | |
| B | 3’b010 | Inst[31] | | Inst[7] | Inst[30:25] | Inst[11:8] | 0 |
| U | 3’b011 | Inst[31:12] | | 0 | | | |
| J | 3’b100 | Inst[31] | Inst[19:12] | Inst[20] | Inst[30:21] | | 0 |

表2 ImmSel编码表

ALUSel[3:0]为ALU选择正确的运算指令，本设计仅涉及add、sub、and和or运算指令。编码设计如表3。

|  |  |
| --- | --- |
| ALUSel Value | Instruction |
| 2’b00 | add |
| 2’b01 | sub |
| 2’b10 | and |
| 2’b11 | or |

表3 ALUSel编码表

WBSel[1:0]选择是从DMEM写数据进memory，还是输出ALU或者对rd写入PC+4。编码设计如表4。

|  |  |
| --- | --- |
| WBSel Value | Control component |
| 2’b00 | ALU |
| 2’b01 | Mem |
| 2’b10 | PC+4 |

表4 ALUSel编码表

另外，图1真值表中并未一些一位信号的编码。本设计在此给出，如表5。

|  |  |  |  |
| --- | --- | --- | --- |
| Value | ASel | BSel | MemRW |
| 1’b0 | PC | Imm | Read |
| 1’b1 | Reg | Reg | Write |

表5 一位信号编码表

## 2.2 电路设计

### 2.2.1 输入处理

在本设计的输入信号中，控制器电路仅关心BrEq和BrLT，还有Inst中的opcode、func7和func3位。opcode、func7和func3在Inst中都有固定的位置，这使得我们可以仅关心Inst中固定的几位即可：opcode对应[6:0]，func7对应[31:25]，func3对应[14:12]。

下表给出了本次设计相关的所有指令和对应的opcode、func7和func3编码。可见opcode是最直观的区分方式，因此本设计将重点处理Inst[31:0]中的opcode位，将输入分成branch、jal、jalr、R-type、I-type、lw、sw、U-type即可处理大部分的输出信号。其他信号再根据func7和func3得到instruction后进一步运算。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode[6:0] | Func7 | Func3 | Format |
| beq | 7’b1100011 | \* | 3’b000 | branch |
| bne | 7’b1100011 | \* | 3’b001 |
| blt | 7’b1100011 | \* | 3’b100 |
| bltu | 7’b1100011 | \* | 3’b110 |
| jalr | 7’b1100111 | \* | 3’b000 | jalr |
| jal | 7’b1101111 | \* | \* | jal |
| add | 7’b0110011 | 7’b0000000 | 3’b000 | R-type |
| sub | 7’b0110011 | 7’b0100000 | 3’b000 |
| and | 7’b0110011 | 7’b0000000 | 3’b111 |
| or | 7’b0110011 | 7’b0000000 | 3’b110 |
| addi | 7’b0010011 | \* | 3’b000 | I-type |
| lw | 7’b0000011 | \* | 3’b010 | lw |
| sw | 7’b0100011 | \* | 3’b010 | sw |
| auipc | 7’b0010111 | \* | \* | U-type |

表6 指令和对应的编码及格式表

### 2.2.2 逻辑表达式设计与输入优化

逻辑表达式以Verilog代码呈现：

    //intermediate signal

    assign branch=(opcode==7'b1100011)?1'b1:1'b0;//Branch

    assign R\_type=(opcode==7'b0110011)?1'b1:1'b0;

    assign I\_type=(opcode==7'b0010011)?1'b1:1'b0; //I-type

    assign auipc=(opcode==7'b0010111)? 1'b1:1'b0;//U-type

    assign lw=(opcode==7'b0000011)?1'b1:1'b0;//load

    assign sw=(opcode==7'b0100011)?1'b1:1'b0;//store

    assign jal=(opcode==7'b1101111)?1'b1:1'b0;

    assign jalr=(opcode==7'b1100111)?1'b1:1'b0;

    assign beq= branch & (func3==3'b000);

    assign bne= branch & (func3==3'b001);

    assign blt= branch & (func3==3'b100);

    assign bltu= branch & (func3==3'b110);

    assign add=R\_type & (func3==3'b000) & (func7==7'b0000000);

    assign sub=R\_type & (func3==3'b000) & (func7==7'b0100000);

    assign and\_op=R\_type & (func3==3'b111) & (func7==7'b0000000);

    assign or\_op=R\_type & (func3==3'b110) & (func7==7'b0000000);

    //output signal

    assign PCSel= ((beq|!bne) & BrEq) | ((blt|bltu) & BrLt) | jalr | jal;

    assign ImmSel[0]= sw | auipc;

    assign ImmSel[1]= branch | auipc;

    assign ImmSel[2]= jal | jalr;

    assign BrUn= (!blt&bltu)?1'b1:1'b0;

    assign ASel= R\_type | I\_type | lw | sw |jalr; //ASel=1, Reg; else PC

    assign Bsel= R\_type; //Bsel= 1, Reg; else Imm

    assign ALUSel[0]= sub | or\_op;

    assign ALUSel[1]= and\_op | or\_op;

    assign MemRW= sw; //MemRW= 1, Write

    assign RegWEn= R\_type | I\_type | lw | jal | jalr |auipc;

    assign WBSel[0]= R\_type | I\_type | jal | jalr | auipc;

    assign WBSel[1]= lw;

值得注意的是，只要确保四个运算指令一直有值而不处在高阻态，ALUSel信号的表达式就可与add无关。另外，还注意到opcode、func7具有多余的位数。这些将致使输入的进一步优化。

对于opcode，可以看到所有的opcode的[1:0]位都是’11’，因此不需要考虑最低两位。

而func7仅用作四个运算指令的区分，更详细地来说，是add和sub的区分；而ALUSel经过设计后已经与add无关，则func7不再需要——sub、and、or的区分只需要参考func3。

综上所述，本设计仅关心opcode的[6:4]和func3，即Inst的[14:12]和[6:2]位。

### 2.2.3 opcode译码设计

如果要直观地根据输入的Inst[6:2]得到表6的Format，则需要一个5to8的译码器芯片。这种芯片市面上很难找到，故自行设计。

|  |  |  |
| --- | --- | --- |
| Input(ABCDE) | Instruction | Output |
| 5’b00000 | lw | 3’b000 |
| 5’b01000 | sw | 3’b001 |
| 5’b00100 | I-type | 3’b010 |
| 5’b01100 | R-type | 3’b011 |
| 5’b00101 | U-type |
| 5’b11001 | jal | 3’b100 |
| 5’b11000 | branch | 3’b101 |
| 5’b11011 | jal | 3’b110 |

表7 opcode译码器芯片编码设计表

逻辑表达式为：

另外还注意到5’b01100和5’b0010的结果都是3’b011，因此以这路输出为En、E为输入再接一个三态门。得到的芯片原理图见[3.1](#_3.1_opcode译码器原理图绘制)。

### 2.2.4 芯片选择

本设计选择了两个3-8译码器芯片SN74F138N、两个6-channel反相器SN74LS04、四个single-channel反相器SN74AHCT1G14、两个4-channel两路与门芯片CD4081BE、一个single-channel两路与门芯片SN74AUP1G08、一个三态门芯片SN74AUC1G126、三个4-channel两路或门芯片CD4071BE、两个3-channel三路或门芯片CD4075BE、一个两路或门芯片SN74AUP1G32、一个异或芯片SN74AUC1G86。

具体参数见表8：

|  |  |  |
| --- | --- | --- |
| 芯片/作用 | 示意图 | 真值表/端口说明 |
| SN74F138N  3-8译码器 |  |  |
| SN74AHCT1G14  反相器 |  |  |
| CD4081BE  4-channel两路与门 |  |  |
| SN74AUP1G08  single-channel两路与门 |  |  |
| SN74AUC1G126  三态门 |  |  |
| CD4071BE  4-channel两路或门 |  |  |
| CD4075BE  3-channel三路或门 |  |  |
| SN74AUP1G32  两路或门 |  |  |
| SN74AUC1G86  异或 |  |  |

表8 芯片及其说明表

# 3 原理图绘制

## 3.1 opcode译码器原理图绘制

根据上文得到原理图如图4，封装图如图5。

先将5位输入“压缩”至3位，再经过3to8译码器，对输出进行反相，可以直接得到lw、sw、I-type、jalr、branch和jal信号；译码器的Y3输出口经过反相后作为三态门的使能端，能够分别得到R-type和U-type信号。

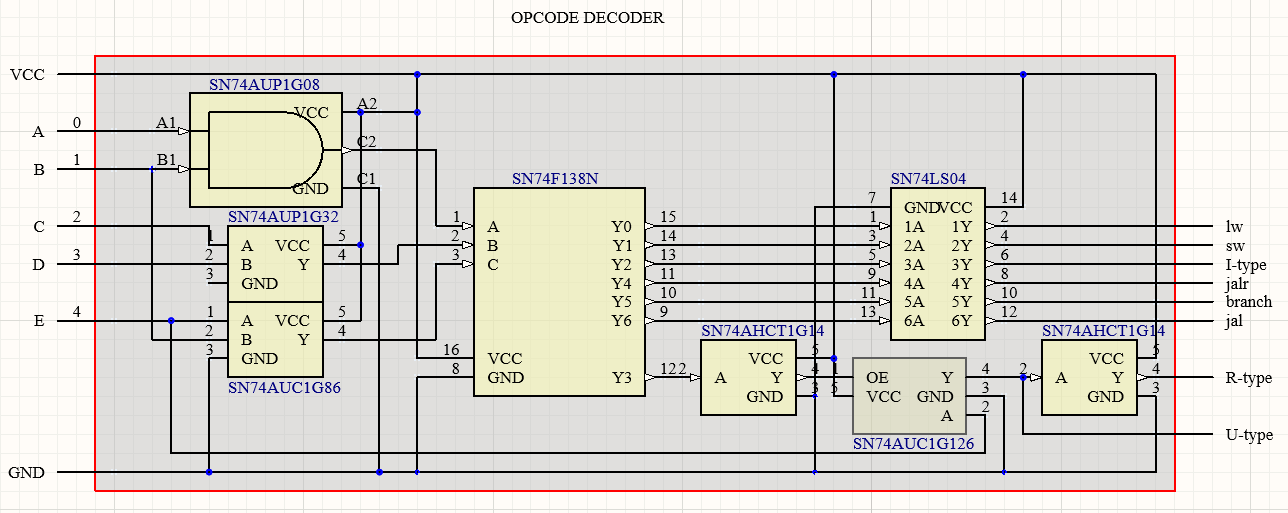


图4 opcode decoder原理图

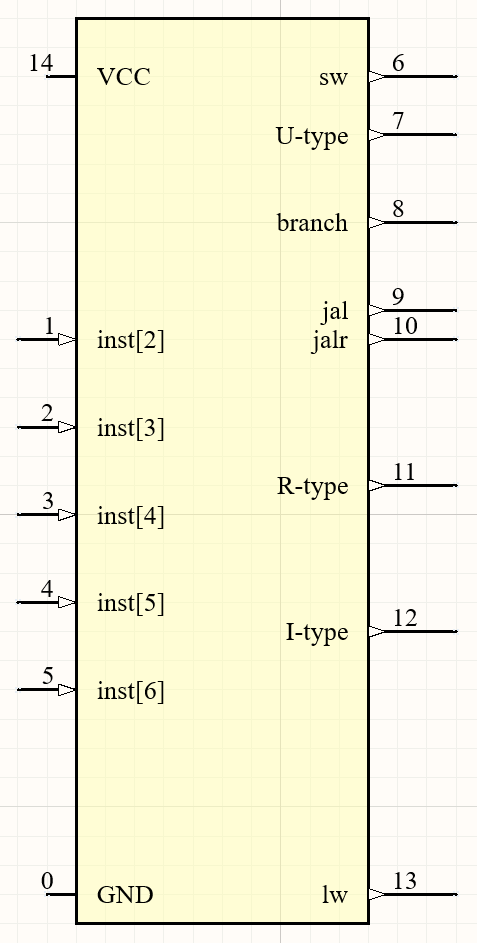


图5 opcode decoder芯片封装图

## 3.2 控制器电路原理图绘制

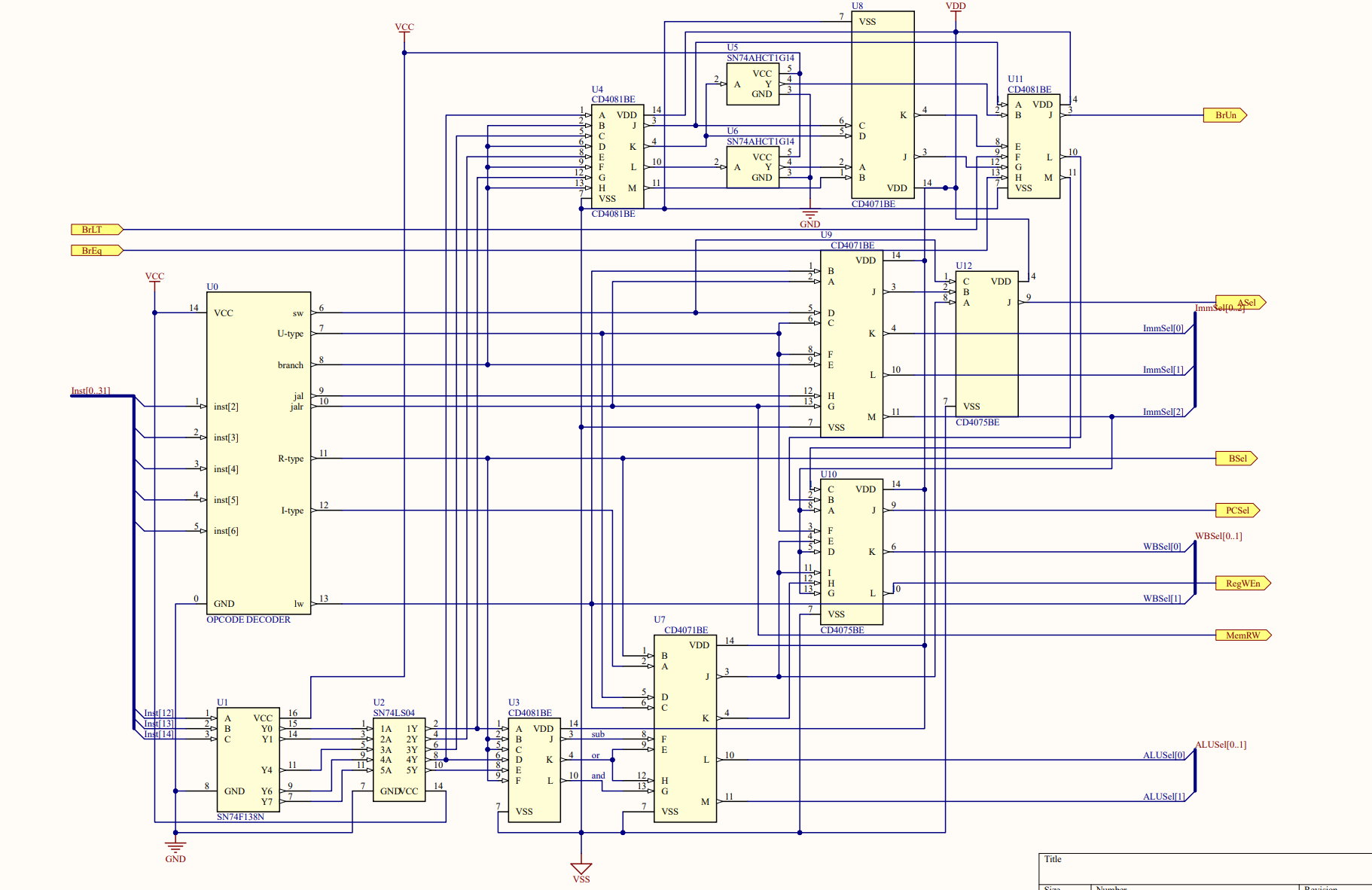


图6 控制器电路原理图

# 4 总结、分析与评估

本次设计采用组合逻辑电路的方式，自己选择芯片实现了一个基于Risc-V指令集的单周期CPU控制器电路，并用Altium Designer（AD）软件绘制出了电路原理图。

除此之外，本设计还在实现功能要求的基础上完成了一系列的优化。例如通过对逻辑表达式的分析有效缩减了电路的输入位数，致使电路能仅仅考虑Inst[31:0]中的8位数据；另外，还观察到R-type | I-type、jal | jalr等组合经常出现在表达式中，因此将它们通过或门后的结果统一引出，大大缩短了芯片的使用数量。

本设计最大的亮点在于opcode 译码芯片的设计，巧妙避开了5输入译码器难以获取的困难，并用最基本的逻辑关系将5输入“压缩”成3输入，实现SN74F138N芯片的反复利用；但是输出设计仍然存在冲突的问题，说明逻辑表达式的设计不够优雅，还有优化的空间以节省三态门、反相器的应用。

从芯片的选择上，本设计共选取了9种共19个芯片，选择种类偏多但总数偏少，其中多channel与或门芯片和多channel反相器芯片的选择大大缩减了芯片数量和电路体积；另外，本设计并没有对供电电压进行严谨的设计，不同型号的芯片的工作电压各异，这是实际应用中应当注意的，也是本设计最主要的改进点。

综上所述，虽然本设计存在一部分问题有待进一步改善，但总体思路和实现已经具有良好可观的完成度。