

# Logisim单周期CPU设计文档

## (一)CPU设计要求

### 32位单周期CPU

支持指令{add, sub, lw, sw, lui, beq, nop, jr, jal, j}

## (二)关键模块设计

### ALU

信号名	方向	描述
SrcA[31:0]	I	32位输入信号，第一个操作数A
SrcB[31:0]	I	32位输入信号，第二个操作数B
ALUControl[2:0]	I	3位输入信号，选择ALU的功能 000 : SrcA + SrcB 001 : SrcA - SrcB 010 : A   B 011 : A & B 100 : A>>B 101 : \$signed(A)>>>B 110 : A<B 置1
Equal	O	1位输出信号，标志A,B是否相等 A=B置Equal=1
Less	O	1位输出信号，标志A是否小于B A<B置Less=1
ALUResult[31:0]	O	32位输出信号，输出运算结果

### GRF

#### 模块定义

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号，将32个寄存器中的值全部清零 1: 复位 0: 无效
WE	I	写使能信号 1: 可向GRF中写入数据 0: 不能向GRF中写入数据

信号名	方向	描述
A1[4:0]	I	5位地址输入信号，指定32个寄存器中的一个，将其中存储的值读出到RD1
A2[4:0]	I	5位地址输入信号，指定32个寄存器中的一个，将其中存储的值读出到RD2
A3[4:0]	I	5位地址输入信号，指定32个寄存器中的一个作为写入的目标寄存器
WD[31:0]	I	32位数据输入信号
RD1[31:0]	O	输出A1指定的寄存器的32位数据
RD2[31:0]	O	输出A2指定寄存器中的32位数据

功能定义

序号	功能名称	描述
1	复位	reset信号有效是，所有寄存器存储的数值清零
2	读数据	读出A1，A2地址对应寄存器中所存储的数据到RD1，RD2
3	写数据	当WE有效且时钟上升沿来临时，将WD写入A3对应的寄存器中

DM

模块定义

信号名	方向	描述
clk	I	时钟信号
Reset	I	复位信号，将RAM中的值全部清零 1：复位 0：无效
WE	I	写使能信号 1：可向GRF中写入数据 0：不能向GRF中写入数据
WD[31:0]	I	32位数据输入信号，要写入的数据
A[31:0]	I	32位输入信号，指定RAM中的的一个地址
DMop[1:0]	I	2位输入信号，用于lb，lh等特殊指令 00：正常读写，lw，sw 01：用于lh和sh，根据WE选择进行哪一条指令 10：用于lb和sb，根据WE选择进行哪一条指令
RD[31:0]	O	32位输出信号，读出A指定的地址中的数据

功能定义

序号	功能名称	描述
----	------	----

序号	功能名称	描述
1	复位	reset信号有效是，RAM所有地址存储的数值清零
2	读数据	读出A指定的地址的所存储的数据到RD
3	写数据	当WE有效且时钟上升沿来临时，将WD写入A对应的地址中

## EXT

信号名	方向	描述
Imm[15:0]	I	15位输入立即数
EXTop	I	功能选择信号 0: Imm无符号拓展到32位 1: Imm符号位拓展到32位
EXTImm[31:0]	O	32位输出信号，输出Imm拓展之后的数

## NPC

### 模块定义

信号名	方向	描述
PC[31:0]	I	32位输入信号，当前指令的地址
NPCop[1:0]	I	两位控制信号，控制NPC的值 00: $PC \leftarrow PC + 4$ 01: $PC \leftarrow PC_{31 \dots 28} \parallel instr\_index \parallel 0_2$ 10: $PC \leftarrow PC + 4 + sign\_extend(offset \parallel 0_2)$ 11: $PC \leftarrow GPR[rs]$
instr_index[26:0]	I	26位输入信号，用于PC的计算
offset[16:0]	I	16位输入信号，PC的偏移量
Reg[32:0]	I	32位输入信号，用于寄存器地址的跳转
Branchop[2:0]	I	用于指示进行何种跳转判断 000: 判断equal 001: 判断n_equal 010: 判断less 011: 判断big 100: 判断less or equal 101: 判断big or equal
equal	I	用于指示ALU两端数据是否 $SrcA = SrcB$
n_equal	I	用于指示ALU两端数据是否 $SrcA \neq SrcB$
less	I	用于指示ALU两端数据是否 $SrcA < SrcB$

信号名	方向	描述
big	I	用于指示ALU两端数据是否SrcA>rcB
less or equal	I	用于指示ALU两端数据是否SrcA<=SrcB
big or equal	I	用于指示ALU两端数据是否SrcA>=SrcB
NPC[31:0]	O	32位输出，输出下一条指令的地址

### 功能定义

序号	功能名称	描述
1	计算PC的下一个值	<p>两位控制信号，控制NPC的的值</p> <p>00: <math>PC \leftarrow PC + 4</math></p> <p>01: <math>PC \leftarrow PC_{31 \dots 28} \parallel instr\_index \parallel 0_2</math></p> <p>10: 结合equal,n_equal,less,big,big or equal,less or equal判断是否需要跳转</p> <p><math>PC \leftarrow PC + 4 + sign\_extend(offset \parallel 0_2)</math></p> <p>11: <math>PC \leftarrow GPR[rs]</math></p>

## IFU

### 模块定义

信号名	方向	描述
clk	I	时钟信号
Reset	I	异步复位信号
NPCop[1:0]	I	<p>两位控制信号，控制NPC的的值</p> <p>00: <math>PC \leftarrow PC + 4</math></p> <p>01: <math>PC \leftarrow PC_{31 \dots 28} \parallel instr\_index \parallel 0_2</math></p> <p>10: <math>PC \leftarrow PC + 4 + sign\_extend(offset \parallel 0_2)</math></p> <p>11: <math>PC \leftarrow GPR[rs]</math></p>
instr_index[26:0]	I	26位输入信号，用于PC的计算
offset[16:0]	I	16位输入信号，PC的偏移量
Reg[32:0]	I	32位输入信号，用于寄存器地址的跳转
Branchop[2:0]	I	<p>用于指示进行何种跳转判断</p> <p>000: 判断equal</p> <p>001: 判断n_equal</p> <p>010: 判断less</p> <p>011: 判断big</p> <p>100: 判断less or equal</p> <p>101: 判断big or equal</p>

信号名	方向	描述
equal	I	用于指示ALU两端数据是否SrcA=SrcB
less	I	用于指示ALU两端数据是否SrcA<SrcB

信号名	方向	描述
Instr[31:0]	O	32为输出信号，输出当前要执行的指令
PC[31:0]	O	32位输出信号，当前PC的地址

## CTRL

### 模块定义

信号名	方向	描述
OP[5:0]	I	6位输入信号，指令操作码
Func[6:0]	I	6位输入信号，指令的func段
RegDst[1:0]	O	GRFA3输入端控制信号 00 : $A3 \leftarrow \text{Instr}_{20...16}$ 01 : $A3 \leftarrow \text{Instr}_{15...11}$ 11 : $A3 \leftarrow 0x1f$
RegWrite	O	寄存器写入控制信号 0 : 不能向GRF写入 1 : 可以向GRF写入
EXTop	O	功能选择信号 0 : Imm无符号拓展到32位 1 : Imm符号位拓展到32位
ALUsrc[1:0]	O	ALUSrcB输入控制信号 00 : $\text{SrcB} \leftarrow \text{RD2}$ 01 : $\text{SrcB} \leftarrow \text{EXTImm}$ 10 : $\text{SrcB} \leftarrow \text{sll指令的s}$ $\text{SrcA} \leftarrow \text{RD2}$
ALUctrl[2:0]	O	3位输出信号，选择ALU的功能 000 : $\text{SrcA} + \text{SrcB}$ 001 : $\text{SrcA} - \text{SrcB}$ 010 : $A \mid B$ 011 : $A \& B$ 100 : $A \gg B$ 101 : $\$signed(A) \ggg B$ 110 : $A < B$ 置1 111 : $A << B$

信号名	方向	描述
Menwrite	O	内存写入控制信号 0 : 不能向DM写入 1 : 可以向DM写入
MemtoReg[1:0]	O	控制向寄存器的写入数据 00 : $WD \leftarrow ALUResult$ 01 : $WD \leftarrow RD$ 10 : $WD \leftarrow NPC_{31...0}$ 11 : $WD \leftarrow [ALUResult_{15...0} \parallel 0_{16}]$

信号名	方向	描述
NPCop[1:0]	O	两位控制信号，控制NPC的值 00 : $PC \leftarrow PC + 4$ 01 : $PC \leftarrow PC_{31...28} \parallel instr\_index \parallel 0_2$ 10 : $PC \leftarrow PC + 4 + sign\_extend(offset \parallel 0_2)$ 11 : $PC \leftarrow GPR[rs]$
Branchop[2:0]	O	用于指示进行何种跳转判断 000 : 判断equal 001 : 判断n_equal 010 : 判断less 011 : 判断big 100 : 判断less or equal 101 : 判断big or equal
DMop[1:0]	O	用于lb,sb,lh,sh等操作的拓展 00 : 正常读取，以字为单位 01 : 用于lh或sh Menwrite = 1 ---> sh Menwrite = 0 ---> lh 11 : 用于lb或sb Menwrite = 1 ---> sb Menwrite = 0 ---> lb

## 相关数据通路信号说明

### RegDst

控制对于A3的输入信号，RegDst=00,  $A3 \leftarrow Instr_{16...20}$

, RegDst=01,  $A3 \leftarrow Instr_{11...15}$

, RegDst=10,  $A3 \leftarrow 0x1f$ , 用于jal指令

## ALUsrc

控制对于SrcA的输入信号，ALUsrc=10，SrcA←RD2，用于sll指令

，ALUsrc=other，SrcA←RD1

控制对于SrcB的输入信号，ALUsrc=00，SrcB←RD2

，ALUsrc=01，SrcB←EXTImm

，ALUsrc=10，SrcB←Instr<sub>6...10</sub>(shamt)

## MemtoReg

控制对于寄存器的写入信号，MemtoReg=00，WD←ALUResult

，MemtoReg=01，WD←RD

，MemtoReg=10，WD←PC+4

，MemtoReg=11，WD←ALUResult<sub>15...0</sub>||0<sub>16</sub>，用于lui指令

## 指令控制信号

Instr	RegDst[1:0]	Regwrite	EXTop	ALUsrc[1]	ALUsrc[0]	ALUctrl[2:0]	Memwrite	MemtoReg[1:0]	NPCop[1:0]	Branchop[2:0]/CMPop[2:0]	DMop[1:0]
add	0 1	1	x	0	0	0 0 0	0	xx	0 0	xxx	xx
sub	0 1	1	x	0	0	0 0 1	0	xx	0 0	xxx	xx
ori	0 0	1	0	0	1	0 1 0	0	xx	0 0	xxx	xx
lw	0 0	1	1	0	1	0 0 0	0	0 1	0 0	xxx	0
sw	xx	0	1	0	1	0 0 0	1	xx	0 0	xxx	0
beq	xx	0	x	0	0	x x x	x	xx	1 0	0 0 0	xx
lui	0 0	1	0	0	1	0 0 0	x	1 1	0 0	xxx	xx
jal	1 0	1	x	x	x	xxx	x	1 0	0 1	xxx	xx
jr	0 0	0	x	x	x	xxx	0	xx	1 1	xxx	xx
j	xx	0	x	x	x	xxx	0	xx	0 1	xxx	xx