Logisim 单周期 CPU 设计文档

一、 CPU 设计方案综述

本 CPU 为 logisim 实现的单周期 CPU（32 位），支持的指令集包含{addu, subu, ori, lw, sw, beq, lui, nop}，并进行了适当扩展。

该 CPU 采用层次化，模块化的设计，主要包含 IFU，GRF，EXT，ALU，DM，Controller 等模块。

二、 关键模块定义

1、 IFU

① 基本描述

IFU 内部主要包括 PC，NPC，IM（容量 32bit*32，起始地址为 0x00000000）以及相关逻辑。NPC 中产生下一条指令的地址，当时钟上升沿到来时，PC 更新指令地址并将其输出，IM 根据地址输出对应指令

② 端口说明

信号名	方向	描述
CLK	I	时钟信号
Reset	I	复位信号
NPCOp[1:0]	I	控制 NPC 进行相应的操作： 00：当前为顺序执行指令，NPC 输出 PC+4

		01：当前指令为 beq，作为决定是否跳转的条件之一 10：当前指令为 jal，NPC 输出 PC31..28||instr_index||02 11：当前指令为 jr，NPC 输出 GRF[rs]
RA[31:0]	I	将 GRF[rs]的值输入 IFU
Zero	I	相等标志信号，判断 ALU 两操作数是否相等
Instru[31:0]	O	根据地址取到的当前指令
PC4[31:0]	O	输出 PC+4 作为地址
imm[15:0]	O	输出 Instru[15:0]

③ 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	根据当前 PC 的值从 IM 中取出相应的指令并通过 Instru 端口输出
3	输出 PC+4	在 PC4 输出端口输出 PC
4	计算 NPC	NPC 根据 NPCOp 取值确定： 00：PC+4 01：PC+4+sign_extend(offset||00)

		（Zero 为真时） 10：PC31..28||instr_index||00 11：GRF[rs]
5	更新 PC	当时钟上升沿到来时，更新 PC 为 NPC

2、 GRF

① 基本描述

GRF 模块内部具有 32 个具有写使能和复位功能的寄存器，0 号寄存器内的值始终为 0。GRF 支持同时读取两个寄存器的值以及写入一个寄存器的操作。

② 端口说明

信号名	方向	描述
CLK	I	时钟信号
Reset	I	复位信号
A1[4:0]	I	地址输入信号，将对应地址寄存器的值输出至 RD1
A2[4:0]	I	地址输入信号，将对应地址寄存器的值输出至 RD2
A3[4:0]	I	地址输入信号，指定要进行写入的寄存器
RFWr	I	写使能信号

WD[31:0]	I	要写入寄存器的值
RD1	O	数据输出信号，输出 A1 地址对应的寄存器的值
RD2	O	数据输出信号，输出 A2 地址对应的寄存器的值

③ 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，GRF 中的寄存器全部复位（初值设置为 0x00000000）
2	读取数据	读取 A1 和 A2 地址所对应寄存器的数据至 RD1 和 RD2 输出端口
3	写入数据	当时钟上升沿到来时，如果 RFWr 信号有效，则将 WD 输入端口的数据写入 A3 地址所对应的寄存器中

3、 EXT

① 基本描述

EXT 用于将 16 位立即数进行符号（无符号）扩展成 32 位并输出。

② 端口说明

信号名	方向	描述
imm16[15:0]	I	数据输入信号，输入要进行扩展的 16 位立即数

EXTOp	I	控制信号： 0：无符号扩展 1：符号扩展
Ext[31:0]	O	数据输出信号，输出扩展完毕的数据

③ 功能定义

序号	功能名称	功能描述
1	无符号扩展	将 16 位立即数无符号扩展为 32 位
2	符号扩展	将 16 位立即数符号扩展为 32 位

4、 ALU

① 基本描述

ALU 对输入的两个操作数（32bit）进行加、减、或、大小比较功能，输出运算的结果以及比较结果。

② 端口说明

信号名	方向	描述
A[31:0]	I	数据输入信号，输入 ALU 的第一个操作数

B[31:0]	I	数据输入信号，输入 ALU 的第二个操作数
ALUOp[1:0]	I	控制信号： 00：A+B 01：A-B 10：A|B
C[31:0]	O	数据输出信号，输出 ALU 的计算结果
Zero	O	数据输出信号，输出两操作数进行相等比较的结果

③ 功能定义

序号	功能名称	功能定义
1	加法	将两操作数相加
2	减法	将两操作数相减
3	或运算	将两操作数按位或
4	相等比较	判断两操作数是否相等，相等则 Zero 为真，反之为假

5、 DM

① 基本描述

DM 用于数据存储（容量为 32bit*32，起始地址为 0x00000000）。DM 支持复位功能，采用单向双端口设计。每当时钟上升沿到来时，如果写使能有效则能将数据写入对应地址，其余时间根据地址信号读出相应数据。

② 端口说明

信号名	方向	描述
CLK	I	时钟信号
Reset	I	复位信号
DMWr	I	写使能信号
A[31:0]	I	地址信号，指定要进行操作的存储单元的地址
WD[31:0]	I	数据输入信号，输入要写入存储单元的数据
RD[31:0]	O	数据输出信号，输出地址对应的存储单元的数据

③ 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，每一个存储单元都被复位为 0x00000000
2	读取	根据 A 地址信号输出对应存储单元的数据至 RD 输出端口
3	写入	当时钟上升沿到来时，如果 DMWr 有效，则根据 A 地址信号将 WD 输入端口中的数据写入对应的存储单元

三、 数据通路设计

部件	IFU	GRF				EXT	ALU		DM	
输入信号	RA	A1	A2	A3	WD	imm	A	B	A	WD
addu		IFU.Instr[25:21]	IFU.Instr[20:16]	IFU.Instr[15:11]	ALU.C		GRF.RD1	GRF.RD2		
subu		IFU.Instr[25:21]	IFU.Instr[20:16]	IFU.Instr[15:11]	ALU.C		GRF.RD1	GRF.RD2		
ori		IFU.Instr[25:21]		IFU.Instr[20:16]	ALU.C	IFU.Instr[15:0]	GRF.RD1	GRF.RD2		
lw		IFU.Instr[25:21]		IFU.Instr[20:16]	DM.RD	IFU.Instr[15:0]	GRF.RD1	EXT.Ext	ALU.C	
sw		IFU.Instr[25:21]	IFU.Instr[20:16]			IFU.Instr[15:0]	GRF.RD1	EXT.Ext	ALU.C	GRF.RD2
beq		IFU.Instr[25:21]	IFU.Instr[20:16]				GRF.RD1	GRF.RD2		

lui				IFU.Instr[20:16]	IFU.Instr[15:0]||0^{16}					
jal				0x1f	IFU.PC4					
jr	GRF.RD1	IFU.Instr[25:21]								
nop										
综合	RA	IFU.Instr[25:21]	IFU.Instr[20:16]	IFU.Instr[15:11] IFU.Instr[20:16] 0x1f	ALU.C,DM.RD,IFU.PC4, IFU.Instr[15:0]||0^{16}	IFU.Instr[15:0]	GRF.RD1	GRF.RD2 EXT.Ext	ALU.C	GRF.RD2

输出	0 端口	1 端口	2 端口	3 端口
GRF.A3	IFU.Instr[15:11]	IFU.Instr[20:16]	0x1f	
GRF.WD	ALU.C	DM.D	IFU.PC4	IFU.Imm32

ALUO.B	GRF.RD2	EXT.Ext		

四、 控制器设计

① 基本思路

通过指令的 opcode 和 funct 产生数据通路所需要的控制信号，具体操作为先通过与阵列得到指令变量，再通过或阵列得到各控制信号的取值。

② 真值表

指令	NPCOp[1:0]	GRFWr	EXTOp	ALUOp[2:0]	DMWr	A3Sel[1:0]	WDSel[2:0]	BSel[1:0]	SSel	DMOp[1:0]	DMSOp
addu (000000/100001)	00	1	x	000	0	00	000	00	x	xx	x
subu (000000/100011)	00	1	x	001	0	00	000	00	x	xx	x
or (000000/100101)	00	1	x	010	0	00	000	00	x	xx	x
and	00	1	x	011	0	00	000	00	x	xx	x

(000000/100100)											
sll (000000/000000)	00	1	x	100	0	00	000	00	0	xx	x
sllv （000000/000100 ）	00	1	x	100	0	00	000	00	1	xx	x
slt (000000/101010)	00	1	x	001	0	00	100	00	x	xx	x
jr (000000/001000)	11	0	x	xxx	0	xx	xxx	xx	x	xx	x
jalr (000000/001001)	11	1	x	xxx	0	00	010	xx	x	xx	x
ori (001101)	00	1	0	010	0	01	000	01	x	xx	x
andi (001100)	00	1	0	011	0	01	000	01	x	xx	x
slti (001010)	00	1	1	001	0	01	100	01	x	xx	x

lw (100011)	00	1	1	000	0	01	001	01	x	00	x
lh (100001)	00	1	1	000	0	01	001	01	x	01	1
lhu (100101)	00	1	1	000	0	01	001	01	x	01	0
lb (100000)	00	1	1	000	0	01	001	01	x	10	1
lbu (100100)	00	1	1	000	0	01	001	01	x	10	0
sw (101011)	00	0	1	000	1	xx	xxx	01	x	00	x
sh (101001)	00	0	1	000	1	xx	xxx	01	x	01	x
sb (101000)	00	0	1	000	1	xx	xxx	01	x	10	x
lui	00	1	x	xxx	0	01	011	xx	x	xx	x

(001111)											
beq (000100)	01	0	x	001	0	xx	xxx	00	x	xx	x
blez (000110)	01	0	x	001	0	xx	xxx	10	x	xx	x
jal (000011)	10	1	x	xxx	0	10	010	xx	x	xx	x
j (000010)	10	0	x	xxx	0	xx	xxx	xx	x	xx	x

五、 测试方案

① 测试代码

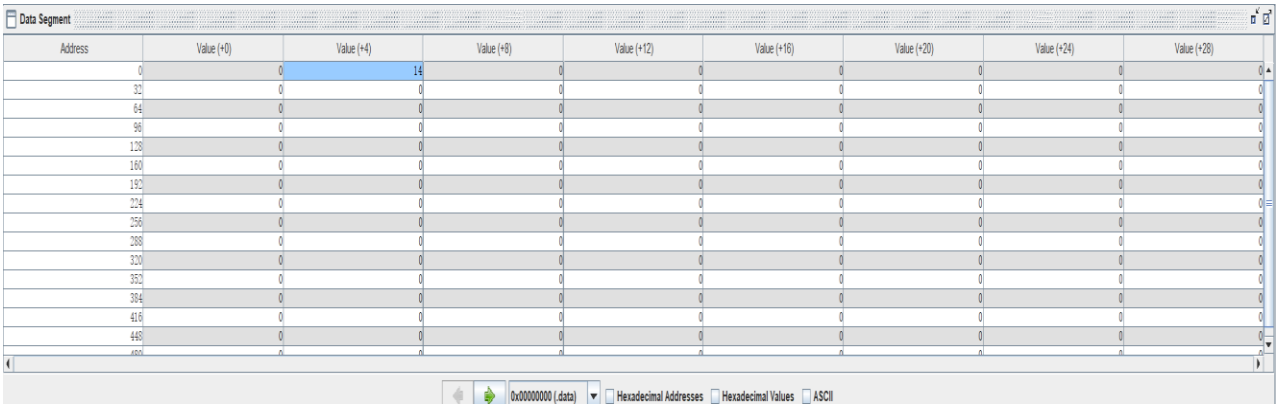
```
.data

a: .word 1:32

.text

ori $t0,$t0,7 #$t0=7

ori $t1,$t1,15 #t1=15
```

```
ori $t3,$t3,4 #t3=2

ori $t5,$t5,1 #t5=1

nop

ori $t6,$t6,1

beq $t5,$t6,next

ori $t0,$t0,6

ori $t1,$t1,6

ori $t3,$t3,6

next:

addu $t1,$t0,$t0 #t1=14

subu $t2,$t1,$t0 #t2=7

lw $t4,a($t3)

sw $t1,a($t3)

jal test

addu $s0,$s0,$t5

subu $s0,$s0,$t5

test:

lui $s0,1
```

```
jr $ra
```

② 期望结果

DM:



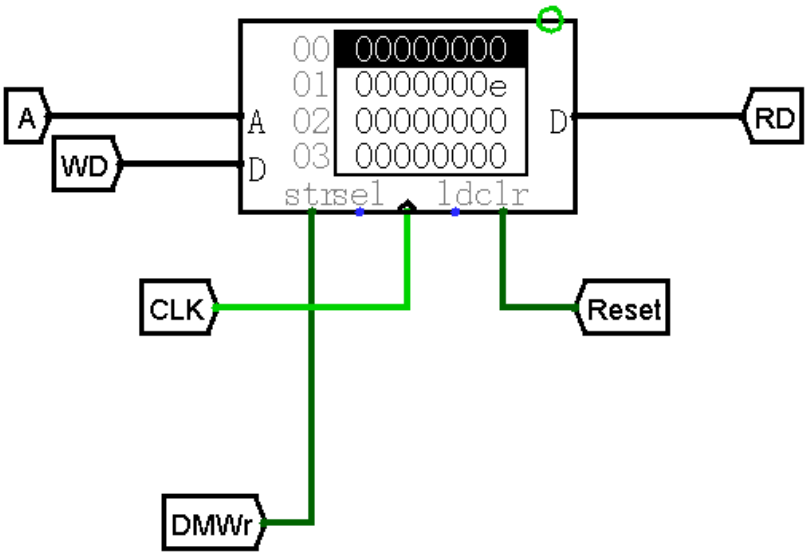
GRF:

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	7
$t1	9	14
$t2	10	7
$t3	11	4
$t4	12	0
$t5	13	1
$t6	14	1
$t7	15	0
$s0	16	65536
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	12284
$fp	30	0
$ra	31	12348
pc		12360
hi		0
lo		0

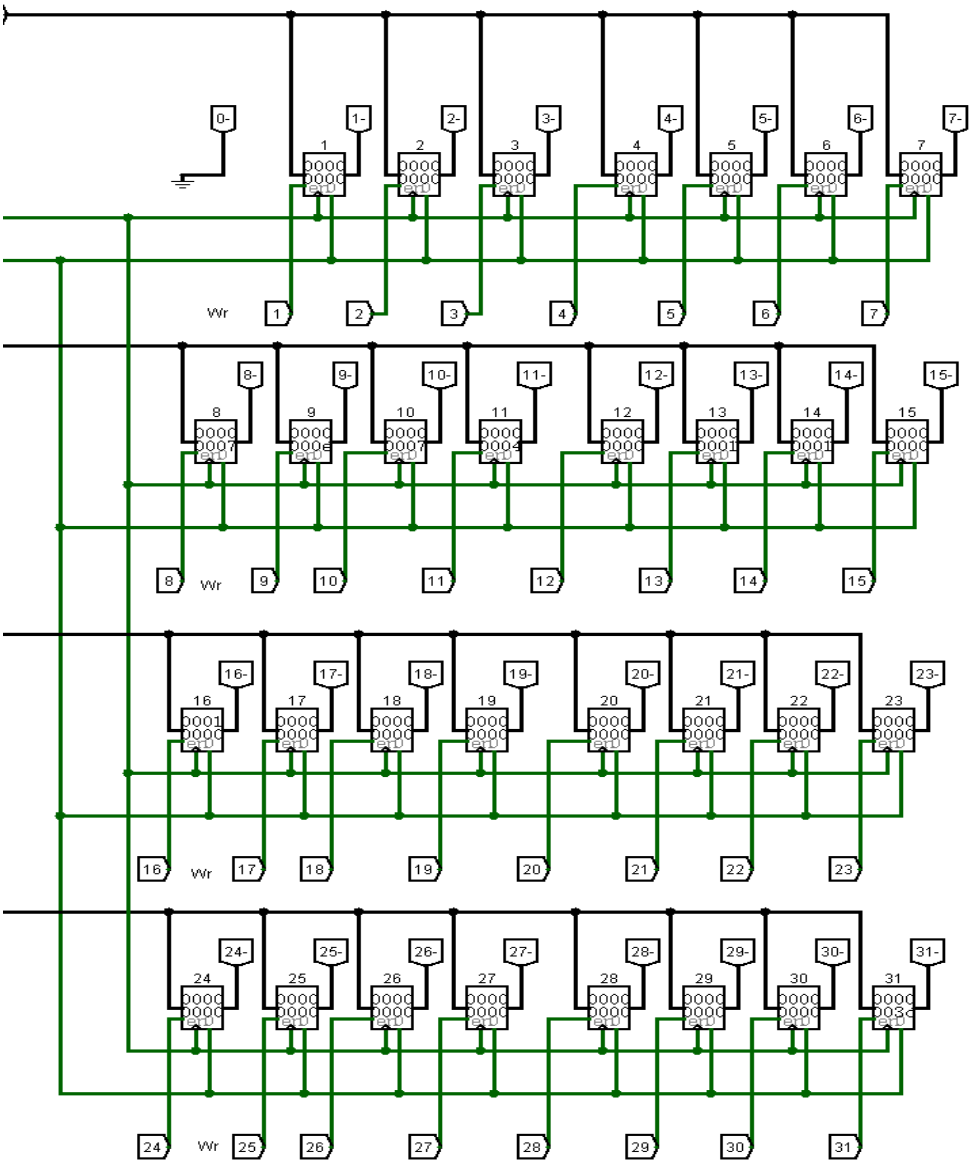
本 CPU 中$gp,$sp 中应为 0

③ 测试结果

DM：



GRF：



六、 思考题

① 现在我们的模块中 IM 使用 ROM， DM 使用 RAM， GRF 用 Register，这种做法合理吗？ 请给出分析，若有改进意见也请一并给出。

合理。

IM 需要导入指令并且无需修改，只会同时读取一个存储单元；DM 需要对存储单元进行读写的操作，同时也只会写或者读一个存储单元，因此 IM 用 ROM，DM 用

RAM（读写分离、单向双端口）是合理的；GRF 要搭建一个寄存器堆，用寄存器来完成也是合理的。

② 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。

nop 的机械码为 0x00000000，在与阵列中不对应任何一个指令变量，所有控制信号都无效，实际上只执行了 PC 的更新。

③ 上文提到，MARS 不能导出 PC 与 DM 起始地址均为 0 的机器码。实际上，可以通过为 DM 增添片选信号，来避免手工修改的麻烦，请查阅相关资料进行了解，并阐释为了解决这个问题，你最终采用的方法。

可以将高位地址信号与相应位数的起始地址比较大小，将大于等于的真值作为DM的片选信号，并将低位地址接入DM的地址输入信号。

由于本 CPU 中的 DM 实际上仅使用 7 位地址，故地址从 0 开始和从 0x00003000 开始实际上没有区别，故无需进行改造。

④ 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，简要阐述相比于测试，形式验证的优劣之处。

优点：速度快，能够覆盖所有情况，无需开发测试模块；

缺点：无法模拟出元件的物理特性和延迟。

指令	NPCOp[1:0]	GRFWr	EXTOp	ALUOp[1:0]	DMWr	A3Sel[1:0]	WDSel[1:0]	BSel
addu (000000/100001)	00	1	x	00	0	00	00	0
subu (000000/100011)	00	1	x	01	0	00	00	0
ori (001101)	00	1	0	10	0	01	00	1
lw (100011)	00	1	1	00	0	01	01	1
sw (101011)	00	0	1	00	1	xx	xx	1
beq (000100)	01	0	x	01	0	xx	xx	0
jal (000011)	10	1	x	xx	0	10	10	x

jr (000000/001000)	11	0	x	xx	0	xx	xx	x
lui (001111)	00	1	x	xx	0	01	11	x