**Project5 VerilogHDL开发多周期 MIPS-C2**

**一： 模块定义**

1. PC模块定义
2. 基本描述

PC主要功能是完成输出当前指令地址并保存下一条指令地址。复位后，PC指向 0x0000\_3000，此处为第一条指令的地址。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| NPC[31:2] | I | 下条指令的地址。 |
| PCWr | I | PC写使能。1：允许NPC写入PC内部寄存器；0：禁止写入PC内部寄存器 |
| clk | I | 时钟信号。 |
| Reset | I | 复位信号。 1：复位 0：无效 |
| PC[31:2] | O | 30位指令存储器地址(最低 2位省略) |

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 0x0000\_3000。 |
| 2 | 保存 NPC并输出 | 在每个 clock的上升沿保存 NPC，并输出。 |

1. NPC模块定义
2. 基本描述

NPC的主要功能是计算下一个PC值。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| NPCOp[1：0] | I | NPC的控制信号。00：pc+4；01：pc+imm；10：j，jal指令执行时pc结果；11：jr指令执行时pc |
| Imm\_O[31:0] | I | 32位指令。 |
| JR[31：2] | I | 31号寄存器中储存的pc地址 |
| PC[31：2] | I | 初始PC值。 |
| NPC[31:2] | O | 计算得到的下一个PC值。 |
| Jal\_o[31:2] | O | 执行Jal指令时要写入31号寄存器的pc值。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1. | 计算PC值 | 根据不同控制信号计算下一个PC值。 |

1. ext模块定义
2. 基本描述

Ext的主要功能是对16位立即数按要求进行扩展。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Imm[15:0] | I | 需要扩展的16位立即数。 |
| Sign | I | 扩展要求。0：“0”扩展 1：符号扩展 |
| Ext | O | 扩展后的32位数据。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1. | “0”扩展 | 对立即数进行“0”扩展。 |
| 2. | 符号扩展 | 对立即数进行符号扩展。 |

1. RegFile模块定义
2. 基本描述

RegFile的主要功能是完成读寄存器的值，并且将值输出；以及将所给的值写入寄存器。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| RS1[4:0] | I | 被读取寄存器的寄存器号 |
| RS2[4:0] | I | 被读取寄存器的寄存器号 |
| RD[4:0] | I | 被写入寄存器的寄存器号 |
| WData[31:0] | I | 被写入的数据。 |
| RegWr | I | 是否可以写入数据。1：可以写入，0：不可以写入。 |
| Clk | I | 时钟信号。 |
| RD1[31:0] | O | RS1[4:0]寄存器中的数据 |
| RD2[31:0] | O | RS2[4:0]寄存器中的数据 |

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 | 取寄存器中的数据 | 从寄存器堆中取指定寄存器中的数据。 |
| 3 | 改写寄存器中的数据 | 当RegWr为1时，将数据写入指定的寄存器中。 |

1. ALU模块定义
2. 基本描述

ALU的主要功能是用来计算算数运算结果。它根据不同的指令执行不同的功能。可以执行加法、减法、或运算、取立即数的高位四种不同的运算。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| ALUctr[3：0] | I | ALU的控制端。000：ADD;001:SUB;010:OR;011:取立即数高位。 |
| A[31:0] | I | ALU输入的第一位操作数。 |
| B[31:0] | I | ALU输入的第二位操作数。 |
| C[31:0] | O | ALU的计算结果。 |
| Zero[2:0] | O | ALU 计算结果为 0 标志。 Zero[0]:1：计算结果为 0 0：计算结果非 0。Zero[1]:1：计算结果大于0 0：计算结果不大于0。Zero[2]:1：计算结果小于0 0：计算结果不小于0。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | ADD | 对两个操作数执行加运算。 |
| 2 | ADDU | 对两个操作数执行无符号加运算。 |
| 3 | SUB | 对两个操作数执行减运算。 |
| 4 | SUBU | 对两个操作数执行无符号减运算。 |
| 5 | AND | 对两个操作数执行与运算。 |
| 6 | OR | 对两个操作数执行或运算。 |
| 7 | XOR | 对两个操作数执行异或运算。 |
| 8 | NOR | 对两个操作数执行或非运算。 |
| 9 | LUI | 对第二个操作数取低16位作为高16位组成32位数。 |
| 10 | SLL  SLLV | 逻辑左移运算。 |
| 11 | SRL  SRLV | 逻辑右移运算。 |
| 12 | SRA  SRAV | 算术右移运算。 |
| 13 | BLEZ\_BGTZ\_BLTZ\_BGEZ | 判断数字与0的大小。 |

1. DM模块定义
2. 基本描述

DM的主要功能是取数据存储器中存储的数据，还可以将自己的数据写入数据存储器中。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Address[9:0] | I | 传入DM的地址，用来写入或读取数据存储器的数据。 |
| Data[31:0] | I | 传入的数据，用来写入数据存储器。 |
| Clk | I | 时钟信号。 |
| MemWrite | I | 是否可以写入数据。1：可以写入，0：不可以写入。 |
| be[3:0] | I | be[3:0]是字节使能，分别与 是字节使能，分别与 是字节使能，分别与 是字节使能，分别与 是字节使能，分别与 din[31:24]、din[23:16]、din[15:8]及 din[7:0]对应。 当 we有效时，对于 有效时，对于 有效时，对于 addr寻址的那个 word来说， be[3]为 1则 din[31:24]被写 入 byte3，类似的 be[2]对应 din[23:16]和 byte2，依次类推。 |
| DataO[31:0] | O | 读取的数据存储器的数据。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读取数据 | 读取数据存储器中的指定地址中的数据。 |
| 2 | 写入数据 | 当MenWrite为1时，将传入的数据写入数据存储器。 |

1. IM模块定义
2. 基本描述

IM模块功能是根据所给的地址读出指定的mips指令。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| pc[9:0] | I | pc地址。 |
| Instr[31:0] | O | pc地址的mips指令 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1. | 读取MIPS指令 | 读取pc地址上的mips指令。 |

1. control模块定义
2. 基本描述

control的主要功能是对每个指令产生所对应的控制信号的值。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| op[6:0] | I | 指令的31-26位 |
| funct[6:0] | I | 指令的0-5位 |
| r[4:0] | I | 指令的16-20位 |
| reset | I | 复位信号，产生初态。 |
| clk | I | 时钟信号，来实现状态转换。 |
| Zero[2:0] | I | ALU 计算结果为 0 标志。 Zero[0]:1：计算结果为 0 0：计算结果非 0。Zero[1]:1：计算结果大于0 0：计算结果不大于0。Zero[2]:1：计算结果小于0 0：计算结果不小于0。 |
| whb[7:0] | O | 用来区分sw,sh,sb,lw,lh,lhu,lb,lbu指令。 |
| NPCOp[1：0] | O | NPC模块的控制信号。 |
| WDSel[1：0] | O | 选择写入寄存器的数据。00：ALUOUT；01：R（DM中的数据）；10：{Jal\_O，2’b00} |
| RegDst[1：0] | O | 写入寄存器的选择。00：RS2；01：Imm[15:11]；10：1F |
| PCWr | O | 控制PC更改。 |
| IRWr | O | IR寄存器的控制信号。 |
| ALUSelA | O | ALU第二位输入信号的选择。0：选择寄存器堆中读取的第一个数据；1：选择寄存器堆中读取的第二个数据。 |
| ALUSelB | O | ALU第二位输入信号的选择。0：选择寄存器堆中读取的第二个数据；1：选择符号扩展后的32位立即数。 |
| RegWr | O | 是否将数据写入寄存器堆中。0：不写入；1：写入。 |
| MenWr | O | 是否将数据写入数据存储器中。0：不写入；1：写入。 |
| ExtOp | O | 判断扩展类型。 |
| ALUOp[3:0] | O | ALU的控制端。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 产生各个控制信号 | 见模块接口中各控制信号的作用。 |
| 2 | 状态机 | 状态机的转移。 |

1. mux模块定义
2. 基本描述

mux的功能是作为多选器选择有用的数据。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| a | I | 待选择的数据。 |
| b | I | 待选择的数据。 |
| c | I | 待选择的数据。 |
| d | I | 待选择的数据。 |
| S0 | I | 低位选择信号。 |
| S1 | I | 高位选择信号。 |
| y | O | 选择结果。 |
| size\_data |  | 用来确定待选择数据的长度。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1. | 二选一选择器 | 通过低位选择信号s0来从a,b中选择所需数据。s0:0 选a；s0:1 选b。 |
| 2. | 四选一选择器 | 通过低位高位选择信号来从a,b,c,d中选择所需数据。s1:0 s0:0 选a;s1:0 s0:1 选b;s1:1 s0:0 选c;s1:1 s0:1 选d。 |

1. BE模块定义
2. 基本描述

产生字节使能。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Aluout[1:0] | I | ALU计算结果的后两位。 |
| whb[7:0] | I | 用来区分sw,sh,sb,lw,lh,lhu,lb,lbu指令。 |
| be[4:0] | O | 产生字节使能。 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 产生字节使能be[4:0]。 | be[3:0]是字节使能，分别与 是字节使能，分别与 是字节使能，分别与 是字节使能，分别与 是字节使能，分别与 din[31:24]、din[23:16]、din[15:8]及 din[7:0]对应。 当 we有效时，对于 有效时，对于 有效时，对于 addr寻址的那个 word来说， be[3]为 1则 din[31:24]被写 入 byte3，类似的 be[2]对应 din[23:16]和 byte2，依次类推。 |

1. Data\_ext模块定义
2. 基本描述

对从存储器独处的数据根据指令的不同进行数据处理。

1. 模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| data[31:0] | I | DM模块输出的32位数据。 |
| be[3:0] | I | 字节使能。 |
| whb[7:0] | I | 用来区分sw,sh,sb,lw,lh,lhu,lb,lbu指令。 |
| ext[31:0] | O | 数据处理后得到的32位数据。 |

1. 功能定义

根据不同的加载指令产生不同的数据。

**二： 测设要求**

1. **测试期望：**

**第一段代码：**

0号寄存器存储数字为0，4~14号寄存器分别为00000014，00000000，0000000a,0000001e,000a0000,00140000,001e0000,000a0000,00140000,000a0000,00000014;16~23:00000014,00000000,0000000a,0000001e,000a0000,00140000,001e0000,000a0000。

31号寄存器：00003018

DM:0~3地址上存储信息:00000014,00000000,0000000a,0000001e;5~8地址:000a0000,00140000,001e0000,000a0000。

**第二段代码：**

0号寄存器存储数字为0，1号寄存器98765432,5-10号寄存器98765432,98765432，ffff9876，00009876，ffffff98,00000098。

DM：0:98765432 1：54325432 2:00003232。

**第三段代码：**

0号寄存器存储数字为0，1号寄存器91234566,3-10号寄存器91234566,00000002,12345660,02448d15,fe448d15,448d1598,2448d159,e448d159。

**第四段代码：**

0:0;4-18:00000000,00bc614e,03e80000,0000010e,00bc614e,00bc616e,00bc614e,00bc6176,ff439eb1,ffffff9c,00000062,00000000,00000000,00000001,00000001;20:ffffffff

**剩余指令均为一条一条测试。**

**第四段代码条件跳转语句分别替换为其余条件跳转语句即可测试。**

1. **代码注释：**

**第一段代码：**

ori $a0 , $0 , 20寄存器$a0中存储寄存器$0中数字和20或的结果

ori $a2 , $0 , 10寄存器$a2中存储寄存器$0中数字和10或的结果

ori $a3 , $0 , 30寄存器$a3中存储寄存器$0中数字和30或的结果

jal LABEL

jal LABEL

jal LABEL

addu $a1 , $a0 , $0执行无符号加法运算,$a1中存储$a0,$0的和

addu $a1 , $a1 , $a0执行无符号加法运算,$a1中存储$a1,$a0的和

addu $a1 , $a1 , $a0

addu $a1 , $a1 , $a0

subu $a1 , $a1 , $a0

subu $a1 , $a1 , $a0执行无符号减法运算，$a1中的存储的为$a1,$a0的差

subu $a1 , $a1 , $a0

subu $a1 , $a1 , $a0

lui $t0 , 10取立即数10放到高16位给寄存器$t0

lui $t1 , 20取立即数20放到高16位给寄存器$t1

lui $t2 , 30取立即数30放到高16位给寄存器$t2

ori $t6 , $0 , 20寄存器$t6中存储寄存器$0中数字和20或的结果

addu $t3 , $t0 , $t1执行无符号加法运算,$t3中存储$t0,$t1的和

addu $t4 , $t0 , $t2执行无符号加法运算,$t4中存储$t0,$t2的和

addu $t5 , $t1 , $t2执行无符号加法运算,$t5中存储$t1,$t2的和

subu $t3 , $t1 , $t0执行无符号减法运算，$t3中的存储的为$t1,$t0的差

subu $t4 , $t2 , $t0执行无符号减法运算，$t4中的存储的为$t2,$t0的差

subu $t5 , $t2 , $t1执行无符号减法运算，$t5中的存储的为$t2,$t1的差

sw $a0 , 0($0) 将寄存器$a0中的数字存储到$0中存储地址加0的位置

sw $a1 , 4($0) 将寄存器$a1中的数字存储到$0中存储地址加4的位置

sw $a2 , 8($0) 将寄存器$a2中的数字存储到$0中存储地址加8的位置

sw $a3 , 12($0) 将寄存器$a3中的数字存储到$0中存储地址加12的位置

sw $t0 , 0($t6) 将寄存器$t0中的数字存储到$t6中存储地址加0的位置

sw $t1 , 4($t6) 将寄存器$t1中的数字存储到$t6中存储地址加4的位置

sw $t2 , 8($t6) 将寄存器$t2中的数字存储到$t6中存储地址加8的位置

sw $t3 , 12($t6) 将寄存器$t3中的数字存储到$t6中存储地址加12的位置

lw $s0 , 0($0) 寄存器$s0存储$s0中存储地址加0中的字

lw $s1 , 4($0) 寄存器$s1存储$s0中存储地址加4中的字

lw $s2 , 8($0) 寄存器$s2存储$s0中存储地址加8中的字

lw $s3 , 12($0) 寄存器$s3存储$s0中存储地址12中的字

lw $s4 , 0($t6) 寄存器$s4存储$s0中存储地址加0中的字

lw $s5 , 4($t6) 寄存器$s5存储$s0中存储地址加4中的字

lw $s6 , 8($t6) 寄存器$s6存储$s0中存储地址加8中的字

lw $s7 , 12($t6) 寄存器$s7存储$s0中存储地址加12中的字

beq $0 , $0 , LABEE 执行有条件跳转指令

addu $t8 , $0 , $t6

addu $t9 , $0 , $t6

LABEL: jr $ra

LABEE: