

[微处理器设计 4](#_Toc13903624)

[一、实验要求 4](#_Toc13903625)

[二、实验工具 4](#_Toc13903626)

[三、数据通路图 4](#_Toc13903627)

[四、分工描述 4](#_Toc13903628)

[五、开发指令集介绍 4](#_Toc13903629)

[(1)指令：无符号加addu 5](#_Toc13903630)

[(2)指令：无符号减subu 5](#_Toc13903631)

[(3)指令：逻辑与and 5](#_Toc13903632)

[(4)指令：逻辑或or 5](#_Toc13903633)

[(5)指令：逻辑异或xor 6](#_Toc13903634)

[(6)指令：逻辑或非nor 6](#_Toc13903635)

[(7)指令：加无符号立即数addiu 6](#_Toc13903636)

[(8)指令：与立即数andi 6](#_Toc13903637)

[(9)指令：或立即数ori 7](#_Toc13903638)

[(10)指令：异或立即数xori 7](#_Toc13903639)

[(11)指令：加载字lw 7](#_Toc13903640)

[(12)指令：存储字sw 7](#_Toc13903641)

[(13)指令：等于转移beq 8](#_Toc13903642)

[(14)指令：不等于转移bne 8](#_Toc13903643)

[(15)指令：跳转并链接j 8](#_Toc13903644)

[(16)指令：跳转并链接jal 8](#_Toc13903645)

[六、负责部分模块定义 9](#_Toc13903646)

[1.IR模块 10](#_Toc13903647)

[(1)基本描述 10](#_Toc13903648)

[(2)模块接口 10](#_Toc13903649)

[(3)功能定义 10](#_Toc13903650)

[2.mux模块 10](#_Toc13903651)

[(1)基本描述 10](#_Toc13903652)

[(2)模块接口 11](#_Toc13903653)

[(3)功能定义 11](#_Toc13903654)

[3.flopr模块 11](#_Toc13903655)

[(1)基本描述 11](#_Toc13903656)

[(2)模块接口 11](#_Toc13903657)

[(3)功能定义 12](#_Toc13903658)

[4.RF模块 12](#_Toc13903659)

[(1)基本描述 12](#_Toc13903660)

[(2)模块接口 12](#_Toc13903661)

[(3)功能定义 13](#_Toc13903662)

[5.宏定义文件 13](#_Toc13903663)

[1)global\_def.v 13](#_Toc13903664)

[2)instruction\_def.v 13](#_Toc13903665)

[3)ctrl\_encode\_def.v 13](#_Toc13903666)

[6.mips模块 13](#_Toc13903667)

[1)基本描述 13](#_Toc13903668)

[2)模块接口 14](#_Toc13903669)

[3)功能定义 14](#_Toc13903670)

[4)数据通路设计 15](#_Toc13903671)

[5)数据通路合成 33](#_Toc13903672)

[7.mips\_tb模块 34](#_Toc13903673)

[1)基本描述 34](#_Toc13903674)

[七、负责测试样例与波形详解部分 34](#_Toc13903675)

[*1）SW指令示例:* 35](#_Toc13903676)

[*2)LW指令示例:* 38](#_Toc13903677)

[*3)B型指令示例:* 41](#_Toc13903678)

[八、FPGA程序烧录测试 43](#_Toc13903679)

[九、心得体会 43](#_Toc13903680)

[十、程序附件 43](#_Toc13903681)

[十一、参考文献 44](#_Toc13903682)

# 微处理器设计

# 一、实验要求

1. 设计一个CPU

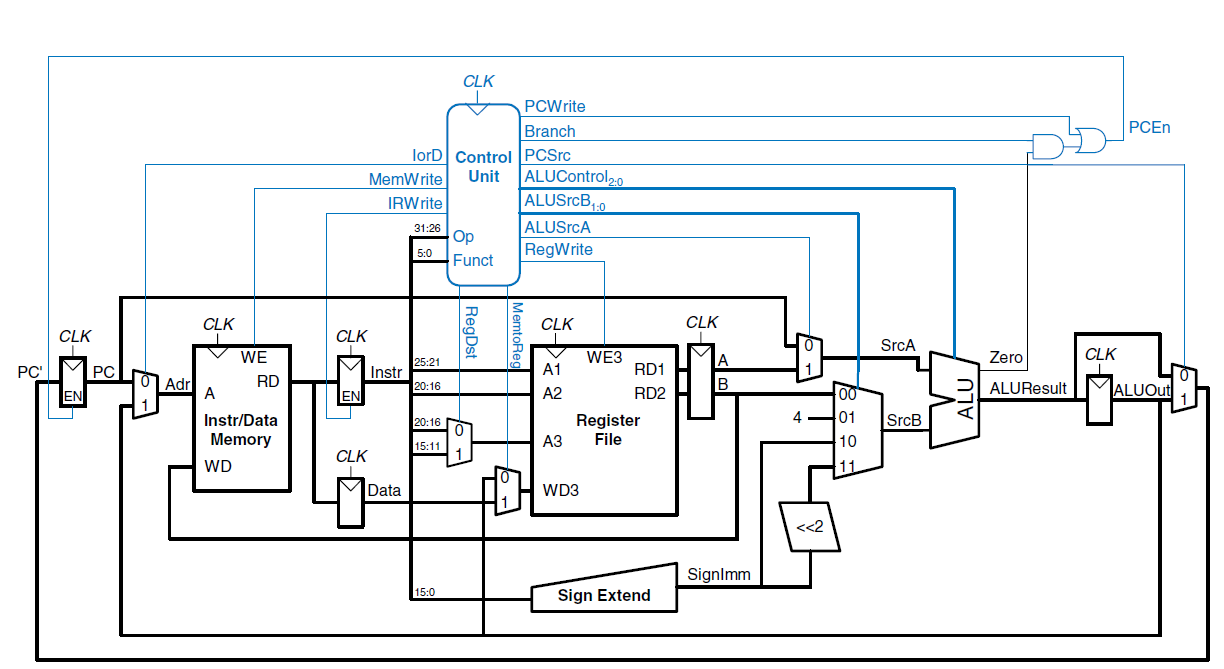
2.掌握CPU的设计方法

# 二、实验工具

1.Xilinx Vivado 2017.4软件。

2.Windows系统PC机。

# 三、数据通路图



# 四、分工描述

我负责IR,mux,flopr,RF模块实现，mips数据通路实现，测试程序，以及撰写实验报告中的SW,LW,B型指令测试

彭宇飞负责ALU、PC、NPC、dm、im、EXT、IR、Ctrl模块的设计以及实验报告中的R,I,J型指令的测试。

# 五、开发指令集介绍

1.设计一个MIPS微处理器，支持MIPS32-C16指令集={addu，subu，and, or, xor, nor, addiu, andi, ori，xori, lw, sw, beq, bne, j, jal}。

### (1)指令：无符号加addu

指令格式：addu rd,rs,rt

指令功能：rd ← rs + rt

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000000 | rs | rt | rd | 00000 | 100001 |

### (2)指令：无符号减subu

指令格式：subu rd,rs,rt

指令功能：rd ←rs - rt

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000000 | rs | rt | rd | 00000 | 100011 |

### (3)指令：逻辑与and

指令格式：and rd,rs,rt

指令功能：rd ←rs & rt

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000000 | rs | rt | rd | 00000 | 100100 |

### (4)指令：逻辑或or

指令格式：or rd,rs,rt

指令功能：rd ←rs | rt

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000000 | rs | rt | rd | 00000 | 100101 |

### (5)指令：逻辑异或xor

指令格式：xor rd,rs,rt

指令功能：rd ←rs ^ rt

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000000 | rs | rt | rd | 00000 | 100110 |

### (6)指令：逻辑或非nor

指令格式：nor rd,rs,rt

指令功能：rd ←~(rs | rt )

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000000 | rs | rt | rd | 00000 | 100111 |

### (7)指令：加无符号立即数addiu

指令格式：addiu rt,rs,imm16

指令功能：rt ← rs+ (zero-extend)imm16

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 001001 | rs | rt | immediate | | |

### (8)指令：与立即数andi

指令格式：andi rt,rs,imm16

指令功能：rt ← rs& (zero-extend)imm16

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 001100 | rs | rt | immediate | | |

### (9)指令：或立即数ori

指令格式：ori rt,rs,imm16

指令功能：rt ← rs | (zero-extend)imm16

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 001101 | rs | rt | immediate | | |

### (10)指令：异或立即数xori

指令格式：ori rt,rs,imm16

指令功能：rt ← rs | (zero-extend)imm16

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 001110 | rs | rt | immediate | | |

### (11)指令：加载字lw

指令格式：lw rt, imm16(rs)

指令功能：rt ← memory[rs + (sign-extend)imm16]

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 100011 | rs | rt | imm16 | | |

### (12)指令：存储字sw

指令格式：sw rt, imm16(rs)

指令功能：memory[rs + (sign-extend)imm16] ←rt

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 101011 | rs | rt | immediate | | |

### (13)指令：等于转移beq

指令格式：beq rs,rt, imm16

指令功能： if (rs == rt) PC ← PC+4 + (sign-extend)imm16<<2

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000100 | rs | rt | immediate | | |

### (14)指令：不等于转移bne

指令格式：bne rs,rt, imm16

指令功能： if (rs != rt) PC ← PC+4 + (sign-extend)imm16<<2

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000101 | rs | rt | immediate | | |

### (15)指令：跳转并链接j

指令格式：j addr26

指令功能： PC ←(PC+4)[31..28],addr26,0,0

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000010 | address | | | | |

### (16)指令：跳转并链接jal

指令格式：jal addr26

指令功能：$31<-PC+4；PC ←(PC+4)[31..28],addr26,0,0

指令编码：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 31..26 | 25..21 | 20..16 | 15..11 | 10..6 | 5..0 |
| op | rs | rt | rd | shamt | func |
| 000011 | address | | | | |

2.在设计好的MIPS微处理器上执行一段程序，给出程序执行结果。

# 六、负责部分模块定义

注：模块描述详见附录代码

表1 负责模块及模块接口定义

|  |  |  |
| --- | --- | --- |
| 序号 | 文件 | 模块接口定义 |
| 1 | IR.v | module IR (clk, rst, IRWr, im\_dout, instr);  input clk;  input rst;  input IRWr;  input [31:0] im\_dout;  output [31:0] instr; |
| 2 | mux.v |  |
| 3 | flopr.v | module flopr #(parameter WIDTH = 8) (clk, rst, d, q);  input clk;  input rst;  input [WIDTH-1:0] d;  output [WIDTH-1:0] q; |
| 4 | RF.v | module RF( A1, A2, A3, WD, clk, RFWr, RD1, RD2 );  input [4:0] A1, A2, A3;  input [31:0] WD;  input clk;  input RFWr;  output [31:0] RD1, RD2; |
| 5 | global\_def.v  ctrl\_encode\_def.v  instruction\_def.v |  |
| 6 | mips.v | module mips(clk, rst) ;  input clk ;  input rst ; |
| 7 | mips\_tb | module ctrl(clk, rst);  input clk, rst; |

## 1.IR模块

### (1)基本描述

IR主要功能是完成对来自IM的指令的缓冲。



### (2)模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [31:0] im\_dout; | I | 指令输入 |
| IRWr | I | 写使能信号 |
| rst | I | 复位信号 |
| clk | I | 时钟信号 |
| [31:0] instr; | O | 指令输出 |

### (3)功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | if rst=1, instr🠄0 |
| 2 | 缓冲 | if rst=0, instr🠄im\_dout |

## 2.mux模块

### (1)基本描述

mux主要功能是多路选择器。mux.v文件包含二选一、四选一、八选一、十六选一4中多路选择器。实例化多路选择器时，可使用#(XXX)，实例化位宽为XXX的多路选择器。



### (2)模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| d0、d1、d2... | I | 供选择数据（d0、d1） |
| s | I | 片选信号 |
| y | O | 片选后的数据 |

### (3)功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 选择 | |  |  |  | | --- | --- | --- | | s | 功能 | 操作 | | 00 | 选择d0 | y 🠄 d0 | | 01 | 选择d1 | y 🠄 d1 | | 10 | 选择d2 | y 🠄 d2 | | ... | ... | ... | |

## 3.flopr模块

### (1)基本描述

DR、A、B、ALUOut由flopr模块实例化，主要功能是数据缓冲，由带复位的D触发器构成。



### (2)模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| [WIDTH-1:0] d | I | 输入数据 |
| rst | I | 复位信号 |
| clk | I | 时钟信号 |
| [WIDTH-1:0] q | O | 输出数据 |

### (3)功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 数据复位 | if rst=0, q🠄d |
| 2 | 数据缓冲 | if rst=1, q🠄0 |

## 4.RF模块

### (1)基本描述

RF主要功能是保存寄存器文件，并支持对通用寄存器的访问。



### (2)模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A1 [4:0] | I | 需要读的寄存器1的地址 |
| A2 [4:0] | I | 需要读的寄存器2的地址 |
| A3 [4:0] | I | 需要写的寄存器的地址 |
| WD [31:0] | I | 需要写的寄存器的数据 |
| RFWr | I | 寄存器写使能端  0：寄存器不写  1：寄存器写 |
| clk | I | 时钟信号 |
| RD1 [31:0] | O | 需要读的寄存器1的数据 |
| RD2 [31:0] | O | 需要读的寄存器2的数据 |

### (3)功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读取通用寄存器 | 根据输入的RS、RT域的值，输出相应通用寄存器所存储的数据。  RD1🠄RF[A1]；RD2🠄RF[A2] |
| 2 | 写回通用寄存器 | 当RF写使能有效时，将待写的数据写入给定地址的通用寄存器中。  Clk上升沿时  if (RFWr) then  RF[A3]🠄WD |

## 5.宏定义文件

### 1)global\_def.v

定义DEBUG宏，控制是否调试。

`define DEBUG 1

### 2)instruction\_def.v

定义13条指令的OPCODE/FUNCT域的。

详见附录代码

### 3)ctrl\_encode\_def.v

相关控制信号的宏。其中使用Verilog描述控制器的控制信号时，需与该宏保持一致。也可根据情况增加控制信号的宏。

详见附录代码

## 6.mips模块

### 1)基本描述

mips模块是一个CPU，只含有复位信号rst和时钟信号clk，内部由PC、NPC、DM、IM、EXT、ALU、IR、Ctrl等模块以及一些多路选择器和缓冲器组成。

### 2)模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| rst | I | 复位信号 |

### 3)功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 构建CPU数据通路 | 连接内部组成模块，构建数据通路。 |

### 4)数据通路设计

#### (1) addu rd,rs,rt

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS和RT写入A、B   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Exe(执行) | * + Op：ALU完成无符号加法计算并写入ALUOut |
| 4 | ALUWB(结果回写) | * + Op：ALUOut写入RF的RD寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch(读取指令) | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF(读取操作数) | 2个操作数存入A/B | A🠄RF[rs]; B🠄RF[rt] |  |  |
| 3 | Exe(执行) | 执行加法，结果存入ALUOut | ALUOut🠄ALU(A,B) | ALU | ALUOp🠄`ALUOp\_ADDU |
| 4 | ALUWB(结果回写) | 计算结果回写至rd寄存器 | RF[rd]🠄ALUOut | RF | RFWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **ADDU rd,rs,rt** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** |  |  |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  |  | **rd** | **ALUOut.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** | **RF.RD2** |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (2) subu rd,rs,rt

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS和RT写入A、B   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Exe(执行) | * + Op：ALU完成无符号减法计算并写入ALUOut |
| 4 | ALUWB(结果回写) | * + Op：ALUOut写入RF的RD寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch(读取指令) | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF(读取操作数) | 2个操作数存入A/B | A🠄RF[rs]; B🠄RF[rt] |  |  |
| 3 | Exe(执行) | 执行无符号减法，结果存入ALUOut | ALUOut🠄ALU(A,B) | ALU | ALUOp🠄`ALUOp\_SUBU |
| 4 | ALUWB(结果回写) | 计算结果回写至rd寄存器 | RF[rd]🠄ALUOut | RF | RFWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **subu rd,rs,rt** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** |  |  |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  |  | **rd** | **ALUOut.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** | **RF.RD2** |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (3) and rd,rs,rt

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS和RT写入A、B   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Exe(执行) | * + Op：ALU完成与运算并写入ALUOut |
| 4 | ALUWB(结果回写) | * + Op：ALUOut写入RF的RD寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch(读取指令) | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF(读取操作数) | 2个操作数存入A/B | A🠄RF[rs]; B🠄RF[rt] |  |  |
| 3 | Exe(执行) | 执行与运算，结果存入ALUOut | ALUOut🠄ALU(A,B) | ALU | ALUOp🠄`ALUOp\_AND |
| 4 | ALUWB(结果回写) | 计算结果回写至rd寄存器 | RF[rd]🠄ALUOut | RF | RFWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **and rd,rs,rt** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** |  |  |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  |  | **rd** | **ALUOut.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** | **RF.RD2** |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (4) or rd,rs,rt

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS和RT写入A、B   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Exe(执行) | * + Op：ALU完成或运算并写入ALUOut |
| 4 | ALUWB(结果回写) | * + Op：ALUOut写入RF的RD寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch(读取指令) | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF(读取操作数) | 2个操作数存入A/B | A🠄RF[rs]; B🠄RF[rt] |  |  |
| 3 | Exe(执行) | 执行或运算，结果存入ALUOut | ALUOut🠄ALU(A,B) | ALU | ALUOp🠄`ALUOp\_OR |
| 4 | ALUWB(结果回写) | 计算结果回写至rd寄存器 | RF[rd]🠄ALUOut | RF | RFWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **or rd,rs,rt** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** |  |  |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  |  | **rd** | **ALUOut.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** | **RF.RD2** |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (5) xor rd,rs,rt

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS和RT写入A、B   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Exe(执行) | * + Op：ALU完成异或运算并写入ALUOut |
| 4 | ALUWB(结果回写) | * + Op：ALUOut写入RF的RD寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch(读取指令) | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF(读取操作数) | 2个操作数存入A/B | A🠄RF[rs]; B🠄RF[rt] |  |  |
| 3 | Exe(执行) | 执行异或运算，结果存入ALUOut | ALUOut🠄ALU(A,B) | ALU | ALUOp🠄`ALUOp\_XOR |
| 4 | ALUWB(结果回写) | 计算结果回写至rd寄存器 | RF[rd]🠄ALUOut | RF | RFWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **xor rd,rs,rt** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** |  |  |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  |  | **rd** | **ALUOut.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** | **RF.RD2** |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (6) nor rd,rs,rt

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS和RT写入A、B   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Exe(执行) | * + Op：ALU完成或非运算并写入ALUOut |
| 4 | ALUWB(结果回写) | * + Op：ALUOut写入RF的RD寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch(读取指令) | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF(读取操作数) | 2个操作数存入A/B | A🠄RF[rs]; B🠄RF[rt] |  |  |
| 3 | Exe(执行) | 执行或非运算，结果存入ALUOut | ALUOut🠄ALU(A,B) | ALU | ALUOp🠄`ALUOp\_NOR |
| 4 | ALUWB(结果回写) | 计算结果回写至rd寄存器 | RF[rd]🠄ALUOut | RF | RFWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **nor rd,rs,rt** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** |  |  |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  |  | **rd** | **ALUOut.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** | **RF.RD2** |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (7) ori rt,rs,imm16

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS写入A、立即数扩展   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Exe(执行) | * + Op：ALU完成或运算并写入ALUOut |
| 4 | ALUWB(结果回写) | * + Op：ALUOut写入RF的RT寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch(读取指令) | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF(读取操作数) | 操作数存入A  无符号扩展 | A🠄RF[rs]; EXT(Imm16) | EXT | EXTOp🠄`UE |
| 3 | Exe(执行) | 执行或运算，结果存入ALUOut | ALUOut🠄ALU(A,EXT) | ALU | ALUOp🠄` ALUOp\_OR |
| 4 | ALUWB(结果回写) | 计算结果回写至rt寄存器 | RF[rt]🠄ALUOut | RF | RFWr🠄1 |

##### c)ori

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **ori rt,rs,imm16** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** |  |  |  |  | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  | **rt** |  | **ALUOut.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (8) addiu rt,rs,imm16

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS写入A、立即数扩展   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Exe(执行) | * + Op：ALU完成加法计算并写入ALUOut |
| 4 | ALUWB(结果回写) | * + Op：ALUOut写入RF的RT寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch(读取指令) | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF(读取操作数) | 操作数存入A  无符号扩展 | A🠄RF[rs]; EXT(Imm16) | EXT | EXTOp🠄`UE |
| 3 | Exe(执行) | 执行加法计算，结果存入ALUOut | ALUOut🠄ALU(A,EXT) | ALU | ALUOp🠄` ALUOp\_ADDU |
| 4 | ALUWB(结果回写) | 计算结果回写至rt寄存器 | RF[rt]🠄ALUOut | RF | RFWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **addiu rt,rs,imm16** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** |  |  |  |  | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  | **rt** |  | **ALUOut.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (9) andi rt,rs,imm16

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS写入A、立即数扩展   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Exe(执行) | * + Op：ALU完成与运算，并写入ALUOut |
| 4 | ALUWB(结果回写) | * + Op：ALUOut写入RF的RT寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch(读取指令) | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF(读取操作数) | 操作数存入A  无符号扩展 | A🠄RF[rs]; EXT(Imm16) | EXT | EXTOp🠄`UE |
| 3 | Exe(执行) | 执行与运算，结果存入ALUOut | ALUOut🠄ALU(A,EXT) | ALU | ALUOp🠄` ALUOp\_AND |
| 4 | ALUWB(结果回写) | 计算结果回写至rt寄存器 | RF[rt]🠄ALUOut | RF | RFWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **andi rt,rs,imm16** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** |  |  |  |  | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  | **rt** |  | **ALUOut.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (10) xori rt,rs,imm16

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS写入A、立即数扩展   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Exe(执行) | * + Op：ALU完成异或运算并写入ALUOut |
| 4 | ALUWB(结果回写) | * + Op：ALUOut写入RF的RT寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch(读取指令) | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF(读取操作数) | 操作数存入A  无符号扩展 | A🠄RF[rs]; EXT(Imm16) | EXT | EXTOp🠄`UE |
| 3 | Exe(执行) | 执行异或运算，结果存入ALUOut | ALUOut🠄ALU(A,EXT) | ALU | ALUOp🠄` ALUOp\_XOR |
| 4 | ALUWB(结果回写) | 计算结果回写至rt寄存器 | RF[rt]🠄ALUOut | RF | RFWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **xori rt,rs,imm16** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** |  |  |  |  | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  | **rt** |  | **ALUOut.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (11) lw rt, imm16(rs)

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS写入A、立即数扩展   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | MA计算地址 | * + 执行加法，结果存入ALUOut; |
| 4 | MR读存储器 | * + 读取DM，数据存储DR; |
| 5 | MemWB回写 | * + DR写入rt寄存器 |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch  取指令 | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1;  NPCOp🠄+4;  PCWr🠄1 |
| 2 | DCD/RF  读操作数 | 基地址存入A；  偏移符号扩展 | A🠄RF[rs];  EXT(Imm16) | EXT | EXTOp🠄`SE |
| 3 | MA  计算地址 | 执行加法，结果存入ALUOut | ALUOut🠄ALU(A,EXT) | ALU | ALUOp🠄`ALUOp\_ADD |
| 4 | MR  读存储器 | 读取DM，数据存储DR | DR🠄DM[ALUOut] |  |  |
| 5 | MemWB  回写 | DR写入rt寄存器 | RF[rt]🠄DR | RF | RFWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** | **d** | **d** | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** | **d** | **addr** | **din** | **DMWr** | **d** |
| **lw rt, imm16(rs)** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** |  |  |  |  | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.C** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **ALUOut.q** |  |  | **DM.dout** |
| **5** |  |  |  |  |  |  |  |  |  |  |  |  | **rt** | **DR.q** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** |  | **rt** | **DR.q** | **ctrl.RFWr** | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** | **ALUOut.q** |  |  | **DM.dout** |

#### (12) sw rt, imm16(rs)

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分别把RS写入A、立即数扩展   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | MA计算地址 | * + 执行加法，结果存入ALUOut; |
| 4 | MW写存储器 | * + rt寄存器写入DM; |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch  取指令 | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | IR  NPC  PC | IRWr🠄1  NPCOp🠄+4  PCWr🠄1 |
| 2 | DCD/RF  读操作数 | 基地址存入A；  偏移符号扩展 | A🠄RF[rs]  EXT(Imm16) | EXT | EXTOp🠄`SE |
| 3 | MA  计算地址 | 执行加法，结果存入ALUOut | ALUOut🠄ALU(A,EXT) | ALU | ALUOp🠄`ALUOp\_ADD |
| 4 | MW  写存储器 | rt寄存器写入DM | DM[ALUOut]🠄RF[rt] | DM | DMWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **sw rt, imm16(rs)** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** |  |  |  |  | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **ALUOut.q** | **RF.RD2** | **Ctrl.DMWr** |  |
|  | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** |  |  |  |  | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** | **ALUOut.q** | **RF.RD2** | **Ctrl.DMWr** |  |

#### (13) beq rs,rt, imm16

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分分别把RS和RT写入A、B   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Br执行 | * + 执行减法，判断Zero;如果Zero等于0.说明两个数相等，则PC🠄NPC(PC,imm16) |

##### a)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch  取指令 | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | NPC  PC  IR | IRWr🠄1  NPCOp🠄+4  PCWr🠄1 |
| 2 | DCD/RF  读操作数 | RS操作数存入A；  RT操作数存入B | A🠄RF[rs];  B🠄RF[rt]; |  |  |
| 3 | Br  执行 | 执行减法，返回判断Zero | ALUOut🠄ALU(A,B)  PC🠄NPC(PC,imm16) | ALU  NPC  PC | ALUOp🠄`ALUOp\_SUB  NPC🠄`BNPC  If(Zero==0)PCWr🠄0; |

##### b)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **beq rs,rt, imm16** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** | **Imm16** | **Ctrl.EXTOp** |  |  |  |  |  |  |  |  |
| **3** |  | **Imm16** | **Ctrl.NPCOp** |  | **Ctrl.PCWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
|  | **PC.PC** | **Imm16** | **Ctrl.NPCOp** | **NPC.NPC** | **Ctrl.PCWr** |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** | **Imm16** | **Ctrl.EXTOp** | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (14) bne rs,rt, imm16

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + Op1和Op2分分别把RS和RT写入A、B   + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | Br执行 | * + 执行减法，判断Zero;如果Zero等于1，说明两个数不等，则PC🠄NPC(PC,imm16) |

##### a)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch  取指令 | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | NPC  PC  IR | IRWr🠄1  NPCOp🠄+4  PCWr🠄1 |
| 2 | DCD/RF  读操作数 | RS操作数存入A；  RT操作数存入B | A🠄RF[rs];  B🠄RF[rt]; |  |  |
| 3 | Br  执行 | 执行减法，返回判断Zero | ALUOut🠄ALU(A,B)  PC🠄NPC(PC,imm16) | ALU  NPC  PC | ALUOp🠄`ALUOp\_SUB  NPC🠄`BNPC  If(Zero!=0)PCWr🠄0; |

##### b)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **bne rs,rt, imm16** | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** | **Imm16** | **Ctrl.EXTOp** |  |  |  |  |  |  |  |  |
| **3** |  | **Imm16** | **Ctrl.NPCOp** |  | **Ctrl.PCWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
|  | **PC.PC** | **Imm16** | **Ctrl.NPCOp** | **NPC.NPC** | **Ctrl.PCWr** |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** | **Imm16** | **Ctrl.EXTOp** | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |

#### (15) jal imm26

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | JMP执行 | * 计算并保存转移PC到RF[31]；保存PC; PC🠄NPC(PC,imm26)； |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch  取指令 | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | NPC  PC  IR | IRWr🠄1  NPCOp🠄+4  PCWr🠄1 |
| 2 | DCD/RF  读操作数 |  |  |  |  |
| 3 | JMP  执行 | 计算并保存转移PC；  保存PC | RF[31]🠄PC  PC🠄NPC(PC,imm26) | RF  NPC  PC | RFWr🠄1  NPCOp🠄`JNPC  PCWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| jal addr26 | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3** | **PC.PC** | **Imm26** | **Ctrl.NPCOp** |  | **Ctrl.PCWr** |  |  |  |  |  |  |  | **‘d31** | **PC.PC** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** | **Imm26** | **Ctrl.NPCOp** | **NPC.NPC** | **Ctrl.PCWr** |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  | **‘d31** | **PC.PC** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |

#### (16)j imm26

##### a)RTL建模分析

|  |  |  |
| --- | --- | --- |
| 周期 | 阶段 | 操作 |
| 1 | Fetch(读取指令) | * + Op1：IM中读出的数据(即指令)写入IR   + Op2：计算PC=PC+4(PC指向下条指令)     - Op2部署在同一个cycle的理由：硬件设计基本原则之一是“尽早执行”     - 即便后续操作需要再次改变PC也无所谓   + 注意1：由于寄存器时序特点，因此Op2不影响Op1   + 注意2：所有指令都必须包括该step |
| 2 | DCD/RF(读取操作数) | * + 所有指令都必须包括该step     - 该周期隐含包括另一重要步骤：指令译码(控制器内完成) |
| 3 | JMP执行 | * 计算; PC🠄NPC(PC,imm26)； |

##### b)RTL描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 周期 | 阶段 | 语义 | RTL | 需控制的功能部件 | 功能部件控制信号 |
| 1 | Fetch  取指令 | 读取指令；  计算下条指令地址 | IR🠄IM[PC]; PC🠄NPC(PC) | NPC  PC  IR | IRWr🠄1  NPCOp🠄+4  PCWr🠄1 |
| 2 | DCD/RF  读操作数 |  |  |  |  |
| 3 | JMP  执行 | 跳转 | PC🠄NPC(PC,imm26) | RF  NPC  PC | RFWr🠄0  NPCOp🠄`JNPC  PCWr🠄1 |

##### c)数据通路

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **周期** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| j addr26 | **1** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3** | **PC.PC** | **Imm26** | **Ctrl.NPCOp** |  | **Ctrl.PCWr** |  |  |  |  |  |  |  | **‘d31** | **PC.PC** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **PC.PC** | **Imm26** | **Ctrl.NPCOp** | **NPC.NPC** | **Ctrl.PCWr** |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  | **‘d31** | **PC.PC** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |

### 5)数据通路合成

综合各指令的数据通路，得表：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **目标模块** | **NPC** | | | **PC** | | | **IM** | **IR** | | | **RF** | | | | | **A** | **B** | **EXT** | | **ALU** | | | **ALUOut** | **DM** | | | **DR** |
| **PC** | **IMM** | **NPCOp** | **NPC** | **PCWr** | **rst** | **addr** | **im\_dout** | **IRWr** | **rst** | **A1** | **A2** | **A3** | **WD** | **RFWr** |  |  | **Imm16** | **EXTOp** | **A** | **B** | **ALUOp** |  | **addr** | **din** | **DMWr** | **d** |
| **addu rd,rs,rt** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** | **RF.RD2** |  |  | **A.q** | **B.q** |  |  |  |  |  |  |
| **subu rd,rs,rt** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** | **RF.RD2** |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **and rd,rs,rt** | PC.PC |  | Ctrl.NPCOp | NPC.NPC |  |  | PC.PC | IM.dout | Ctrl.IRWr |  | rs | rt | rd | aLUOut.q | ctrl.RFWr | RF.RD1 | RF.RD2 |  |  | A.q | B.q | Ctrl.ALUOp | ALUOut.d |  |  |  |  |
| **or rd,rs,rt** | PC.PC |  | Ctrl.NPCOp | NPC.NPC |  |  | PC.PC | IM.dout | Ctrl.IRWr |  | rs | rt | rd | aLUOut.q | ctrl.RFWr | RF.RD1 | RF.RD2 |  |  | A.q | B.q | Ctrl.ALUOp | ALUOut.d |  |  |  |  |
| **xor rd,rs,rt** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** | **RF.RD2** |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **nor rd,rs,rt** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** | **RF.RD2** |  |  | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **ori rt,rs,imm16** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **Addiu rt,rs,imm16** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **Andi rt,rs,imm16** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **Xori rt,rs,imm16** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **aLUOut.q** | **ctrl.RFWr** | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **lw rt, imm16(rs)** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** |  | **rt** | **DR.q** | **ctrl.RFWr** | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** | **ALUOut.q** |  |  | **DM.dout** |
| **sw rt, imm16(rs)** | **PC.PC** |  | **Ctrl.NPCOp** | **NPC.NPC** |  |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** |  |  |  |  | **RF.RD1** |  | **Imm16** | **Ctrl.EXTOp** | **A.q** | **EXT.** **Imm32** | **Ctrl.ALUOp** | **ALUOut.d** | **ALUOut.q** | **RF.RD2** | **Ctrl.DMWr** |  |
| **beq rs,rt, imm16** | **PC.PC** | **Imm16** | **Ctrl.NPCOp** | **NPC.NPC** | **Ctrl.PCWr** |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** | **Imm16** | **Ctrl.EXTOp** | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **Ben re,rt,imm16** | **PC.PC** | **Imm16** | **Ctrl.NPCOp** | **NPC.NPC** | **Ctrl.PCWr** |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** |  |  |  | **RF.RD1** | **RF.RD2** | **Imm16** | **Ctrl.EXTOp** | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** |  |  |  |  |
| **jal addr26** | **PC.PC** | **Imm26** | **Ctrl.NPCOp** | **NPC.NPC** | **Ctrl.PCWr** |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  | **‘d31** | **PC.PC** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
| **J addr 26** | **PC.PC** | **Imm26** | **Ctrl.NPCOp** | **NPC.NPC** | **Ctrl.PCWr** |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  |  |  | **‘d31** | **PC.PC** | **Ctrl.RFWr** |  |  |  |  |  |  |  |  |  |  |  |  |
| 合成 | **PC.PC** | **Imm26** | **Ctrl.NPCOp** | **NPC.NPC** | **Ctrl.PCWr** |  | **PC.PC** | **IM.dout** | **Ctrl.IRWr** |  | **rs** | **rt** | **rd** | **DR.q** | **Ctrl.RFWr** | **RF.RD1** | **RF.RD2** | **Imm16** | **Ctrl.EXTOp** | **A.q** | **B.q** | **Ctrl.ALUOp** | **ALUOut.d** | **ALUOut.q** | **RF.RD2** | **Ctrl.DMWr** | **DM.dout** |
| **rt** | **aLUOut.q** |
| **EXT. Imm32** |
| **‘d31** | **PC.PC** |
| 多路选择器 |  |  |  |  |  |  |  |  |  |  |  |  | **MUX4\_RF\_A3** | **MUX4\_RF\_WD** |  |  |  |  |  |  | **MUX2\_ALU\_B** |  |  |  |  |  |  |

## 7.mips\_tb模块

### 1)基本描述

激励文件。初始化时钟周期等信号。

# 七、负责测试样例与波形详解部分

本测试使用“code.hex”作为测试样例。汇编代码如下

1. xor $0, $0, $0
2. addiu $2, $0, 0x1234
3. andi $3, $2, 0x3456
4. addu $4, $2, $3
5. subu $5, $3, $2
6. sw $2, 0($0)
7. sw $3, 4($0)
8. sw $4, 8($0)
9. sw $5, 16($0)
10. ori $29, $0, 12
11. xori $1, $2, 0x6543
12. and $1, $1, $3
13. or $1, $1, $4
14. sw $1, 12($0)
15. lw $5, 0($0)
16. beq $2, $5, \_lb2
17. \_lb1:
18. lw $3, 4($0)
19. \_lb2:
20. lw $5, 4($0)
21. bne $3, $5, \_lb1
22. jal F\_Test\_JAL # $31 change
23. # Never return
24. sw $2, 12($0)
25. F\_Test\_JAL:
26. subu $6, $5, $2
27. sw $6, -4($29)
28. j \_loop
29. sw $2, 12($0)
30. \_loop:
31. bne $3, $4, \_loop
32. # Never return back

汇编为机器码如下

1. 00000026
2. 24021234
3. 30433456
4. 00432021
5. 00622823
6. ac020000
7. ac030004
8. ac040008
9. ac050010
10. 341d000c
11. 38416543
12. 00230824
13. 00240825
14. ac01000c
15. 8c050000
16. 10450001
17. 8c030004
18. 8c050004
19. 1465fffd
20. 0c000c15
21. ac02000c
22. 00a23023
23. afa6fffc
24. 08000c19
25. ac02000c
26. 1464ffff

在入口程序将机器码读入指令存储器IM。

$readmemh( "code.hex" , U\_MIPS.U\_IM.imem );

设置100ns为一个时钟周期

`timescale 1ns / 1ps

always #(50) clk = ~clk;

## *1）SW指令示例:*

|  |  |
| --- | --- |
| 指令 | sw $2, 0($0) |
| 机器码 | ac020000 |

state 0阶段，IR,PC写选通

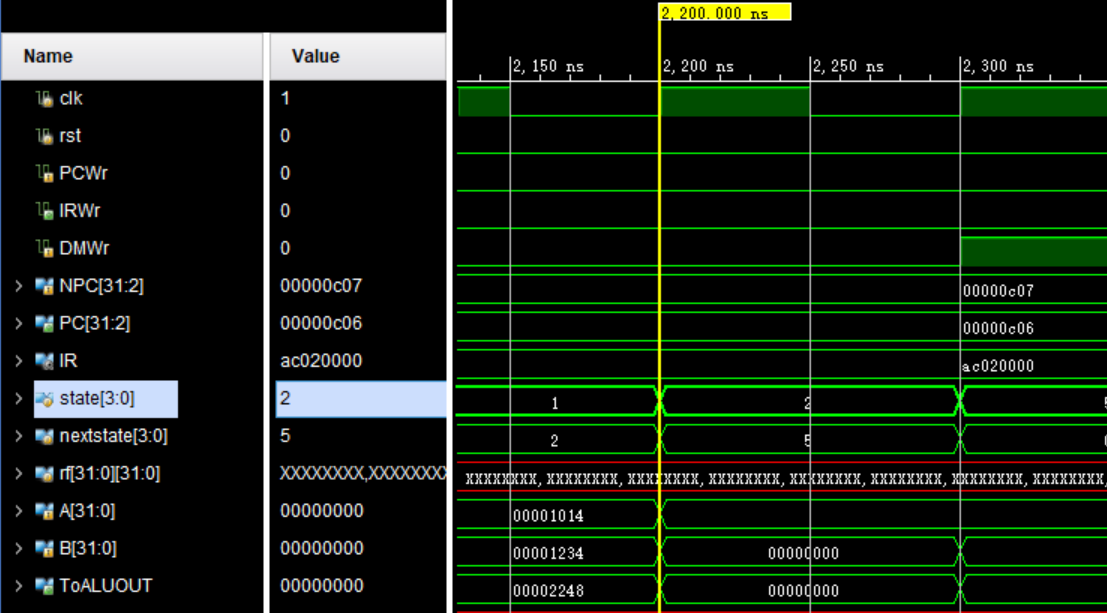


State0结束时，指令机器码从IM中取出存到IR中，NPC写入PC，State0下一阶段为state1（译码，取操作数存到AB阶段）,也是每个PC的开始阶段

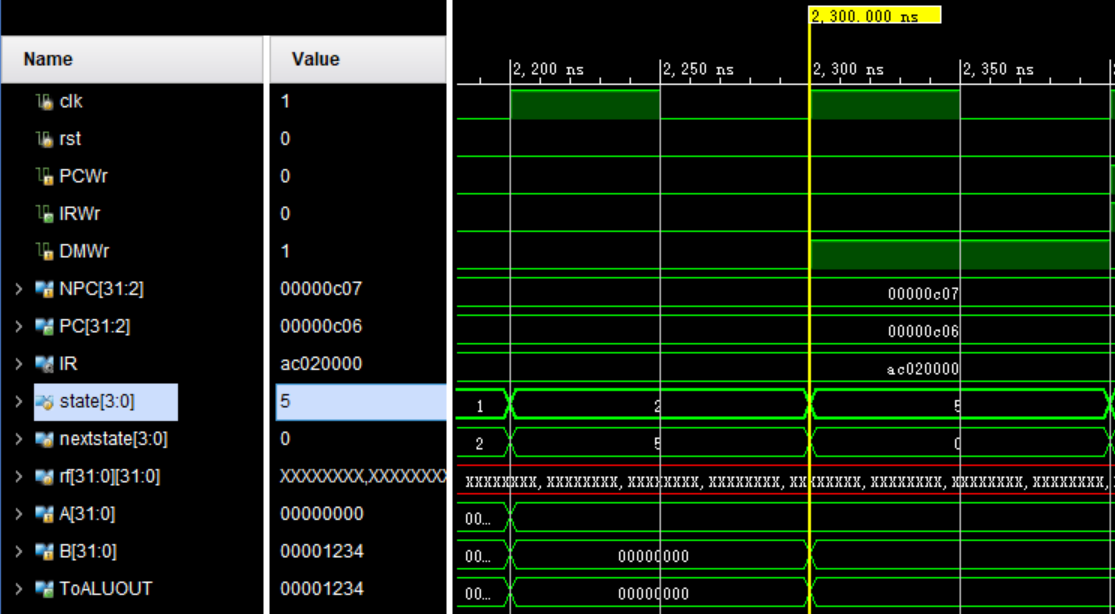


SW型指令，State1阶段将基址寄存器中数值存入A，立即数存入EXT扩展为32位数，State1阶段结束时，将扩展数存入B

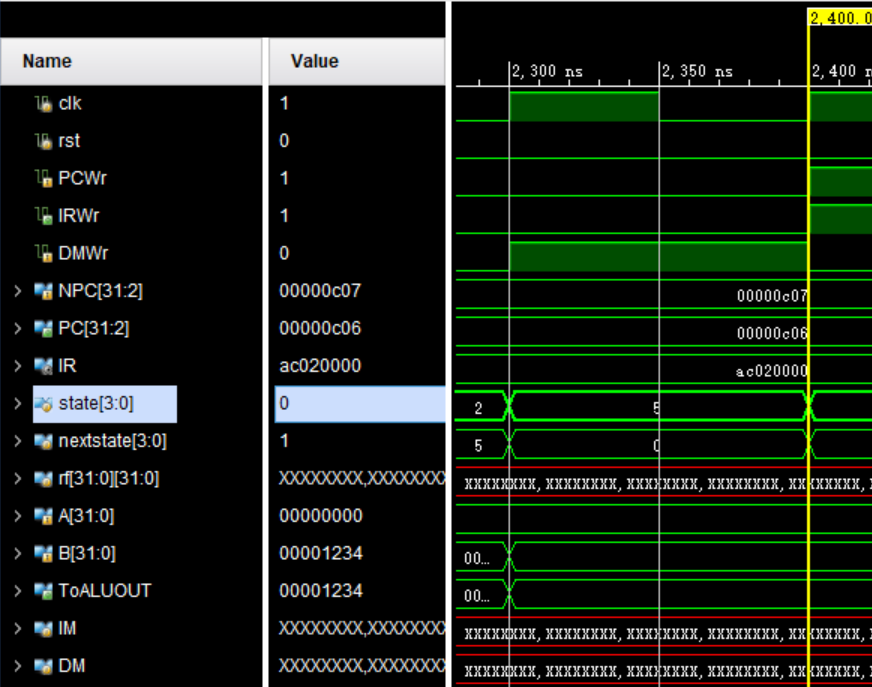
由于指令为SW/LW指令，根据OP在分支时选择state2（计算地址阶段）

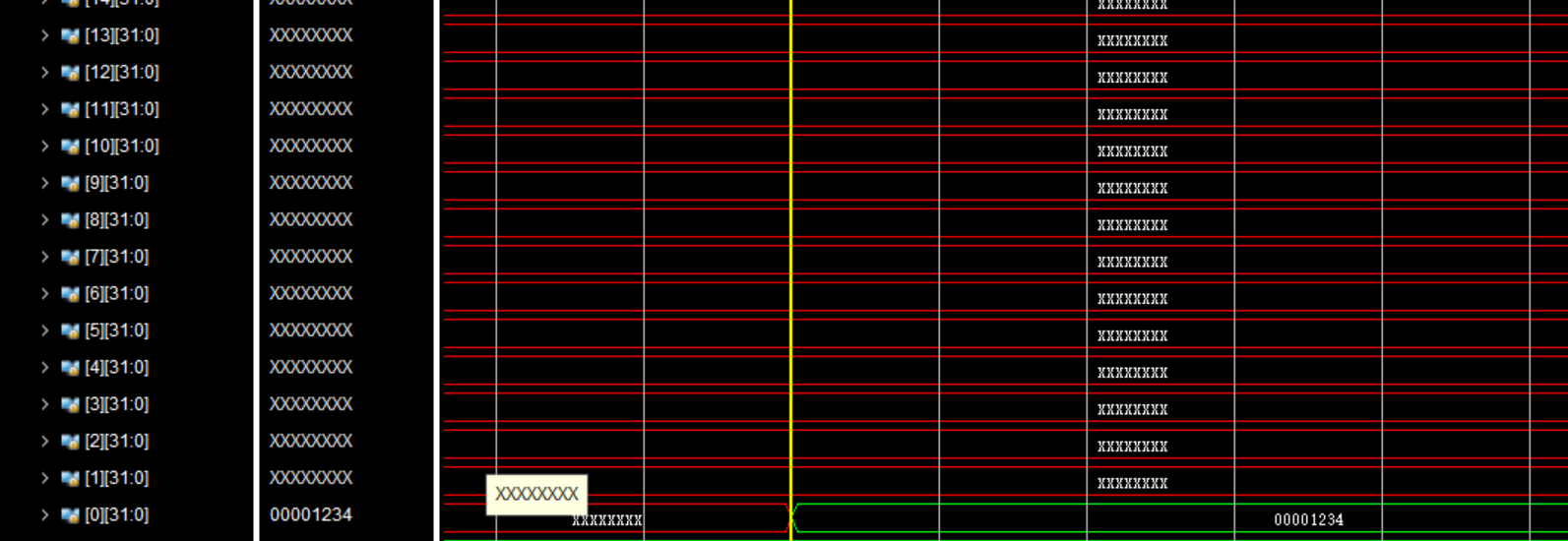


State2阶段A,B作ALUOP（从指令OP判断得出）相应操作，结果存入ALUOUT,由于是SW指令，state2下一阶段为state5(写存储器)



State5阶段，将源操作寄存器内容写至ALUOUT指向的目标地址存储器。



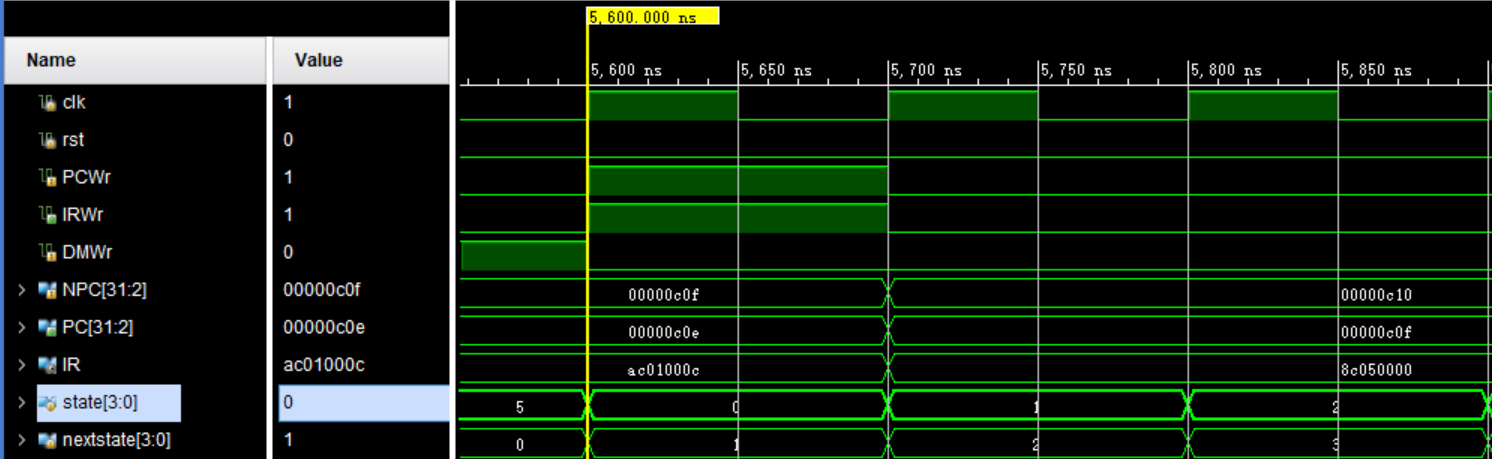


最后回到state0阶段取下一条指令。

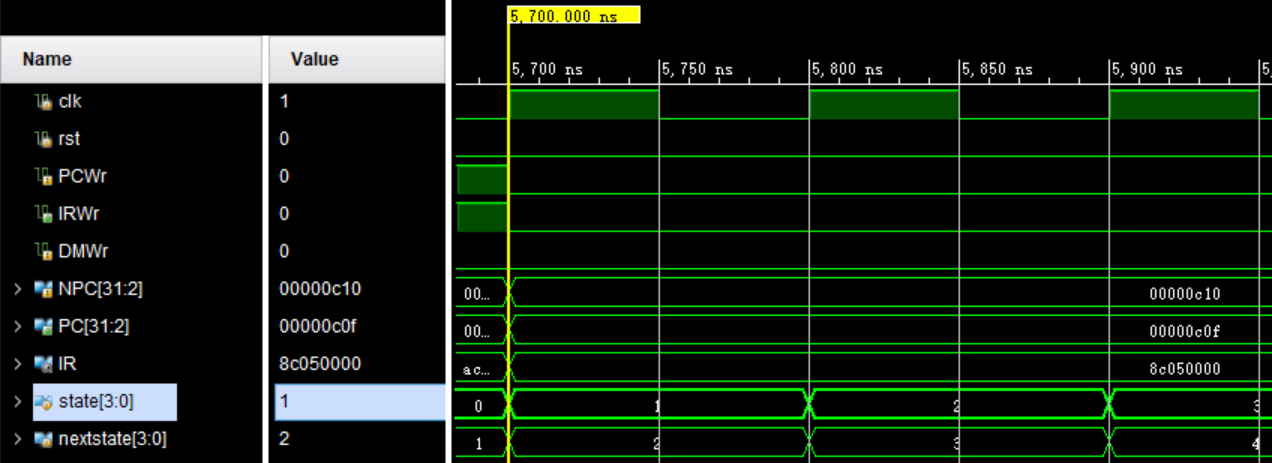
## *2)LW指令示例:*

|  |  |
| --- | --- |
| 指令 | lw $5, 0($0) |
| 机器码 | 8c050000 |

state 0阶段，IR,PC写选通

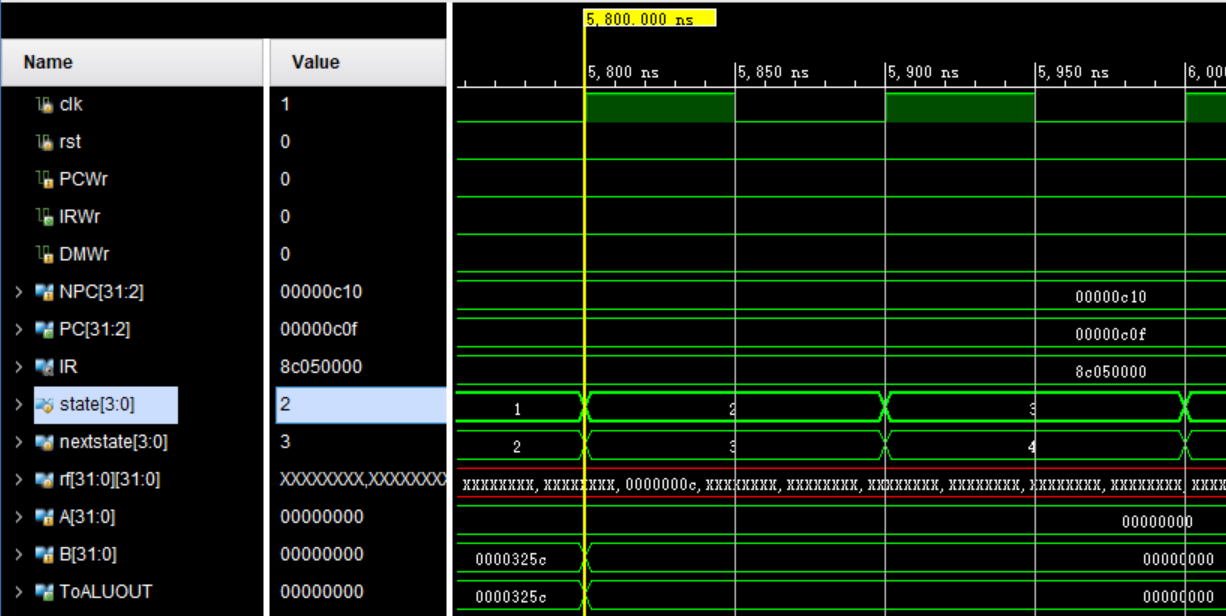


State0结束时，指令机器码从IM中取出存到IR中，NPC写入PC，State0下一阶段为state1（译码，取操作数存到AB阶段）,也是每个PC的开始阶段

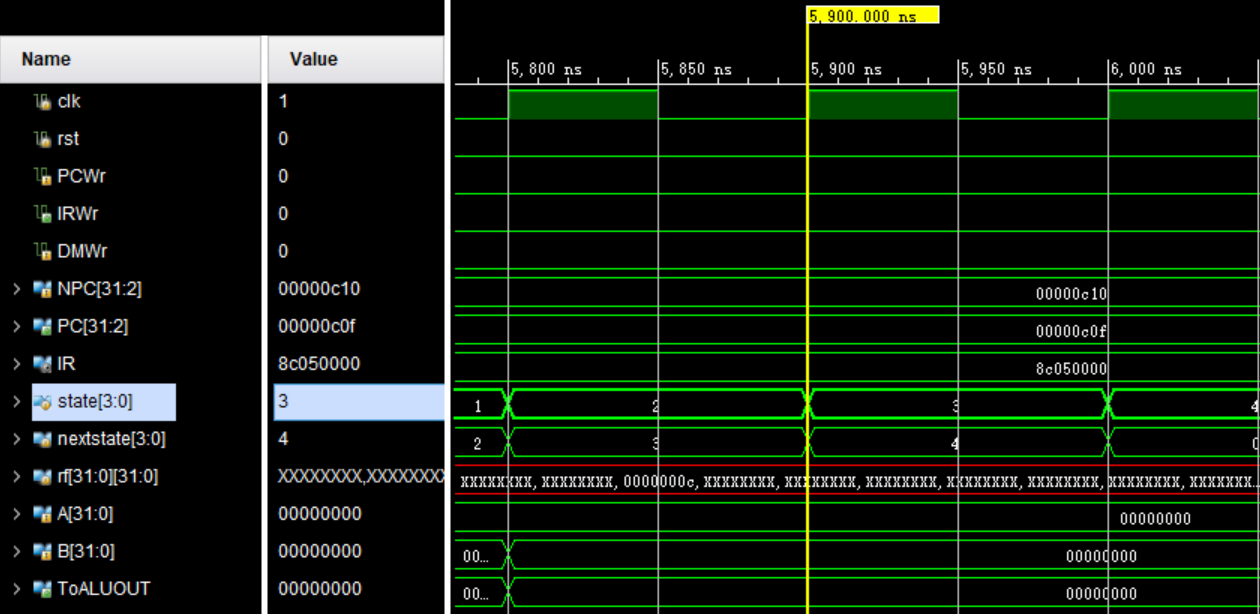


LW型指令，State1阶段将基址寄存器中数值存入A，立即数存入EXT扩展为32位数，State1阶段结束时，将扩展数存入B

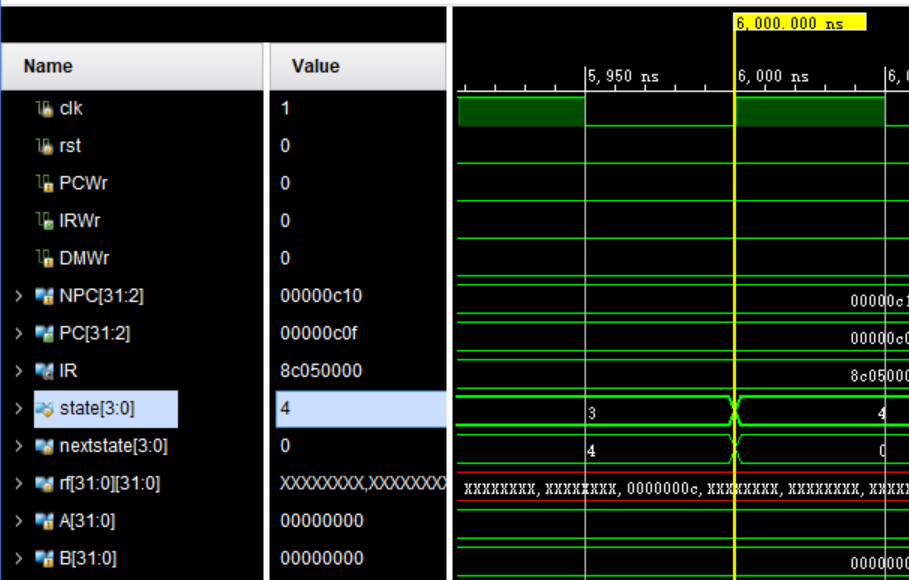
由于指令为SW/LW指令，根据OP在分支时选择state2（计算地址阶段）

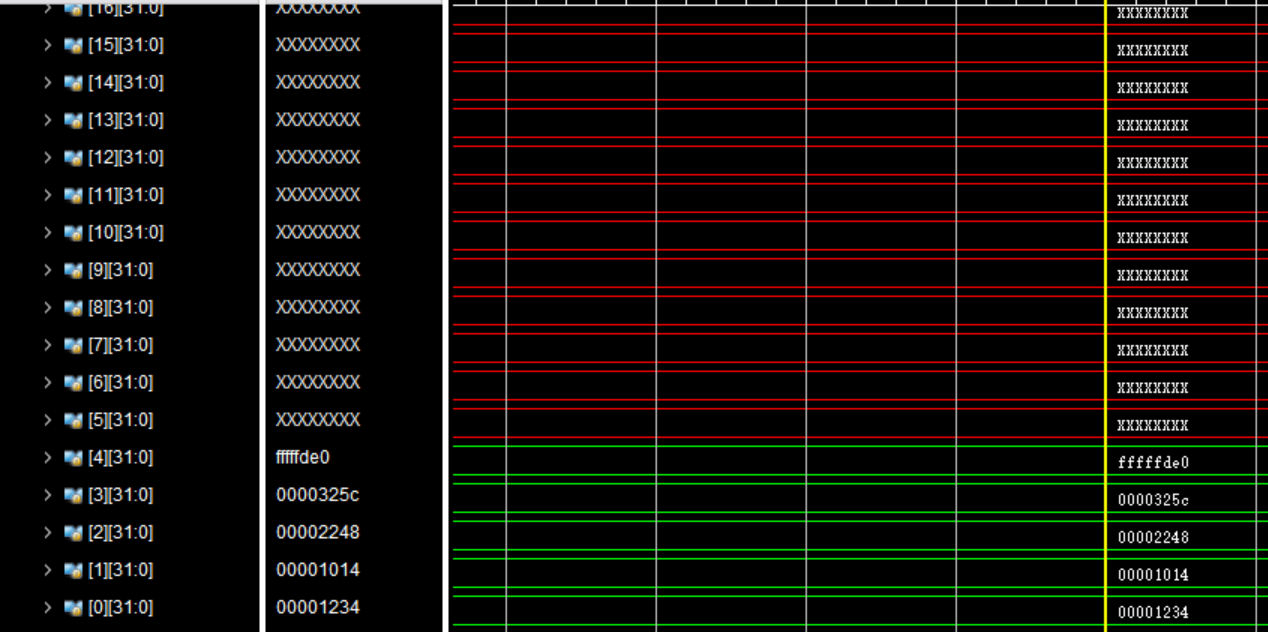


State2阶段A,B作ALUOP（从指令OP判断得出）相应操作，结果存入ALUOUT,由于是LW指令，state2下一阶段为state3(读存储器)

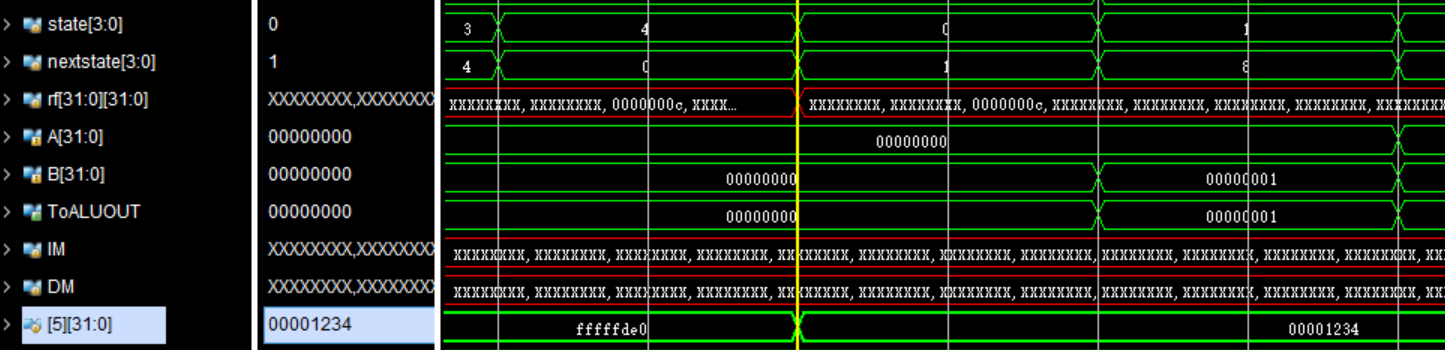


State3阶段，将ALUOUT对应存储器内容读出，state3下一阶段为state4(写回读取内容到目标寄存器)。





state4阶段写回目标寄存器5.

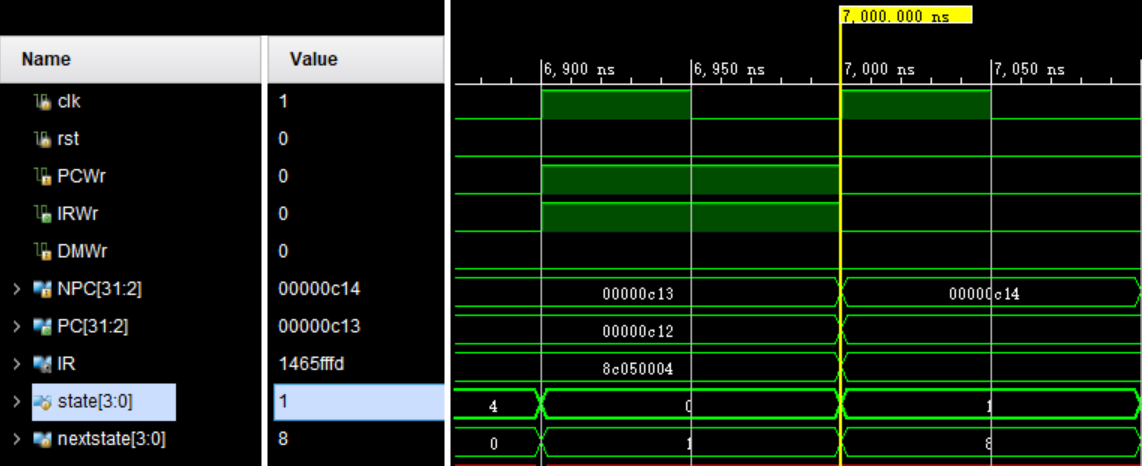


最后回到state0阶段取下一条指令。

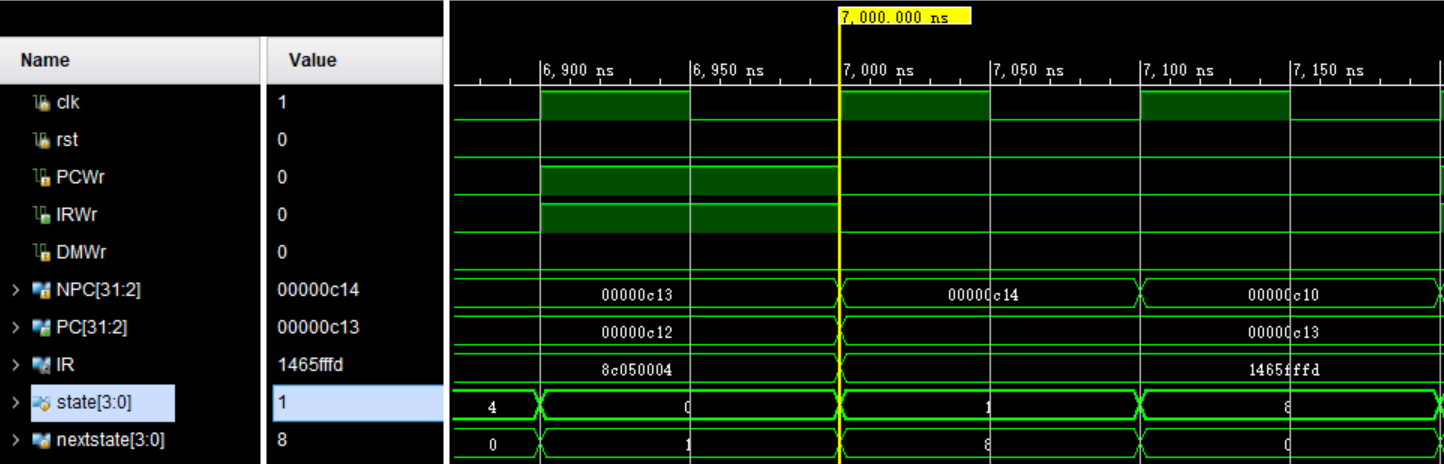
## *3)B型指令示例:*

|  |  |
| --- | --- |
| 指令 | bne $3, $5, \_lb1 |
| 机器码 | 1465fffd |

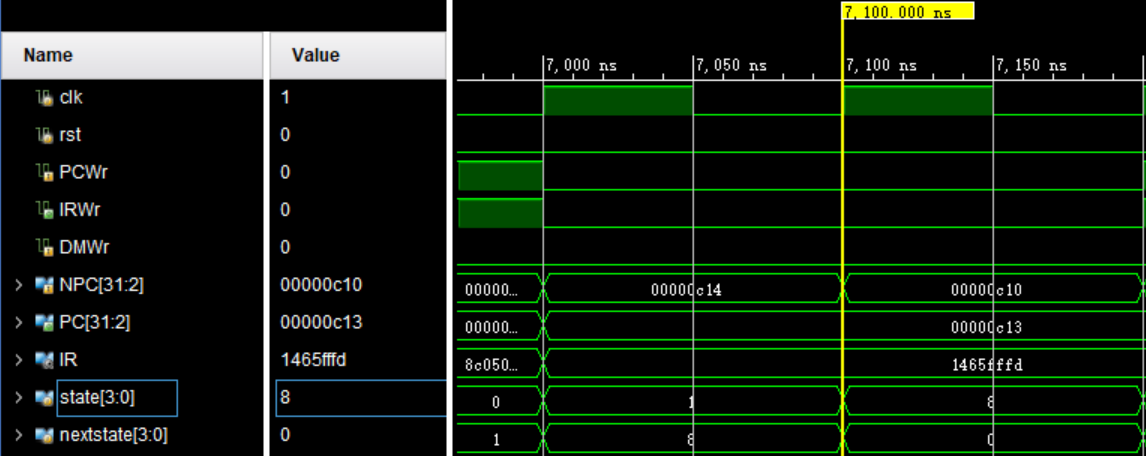
state 0阶段，IR,PC写选通

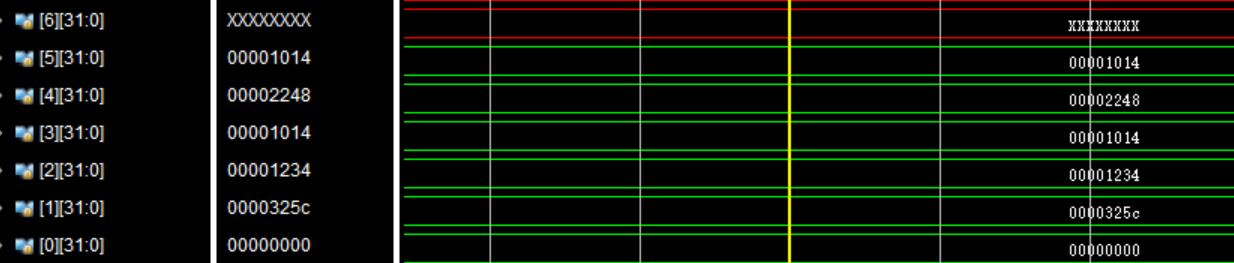


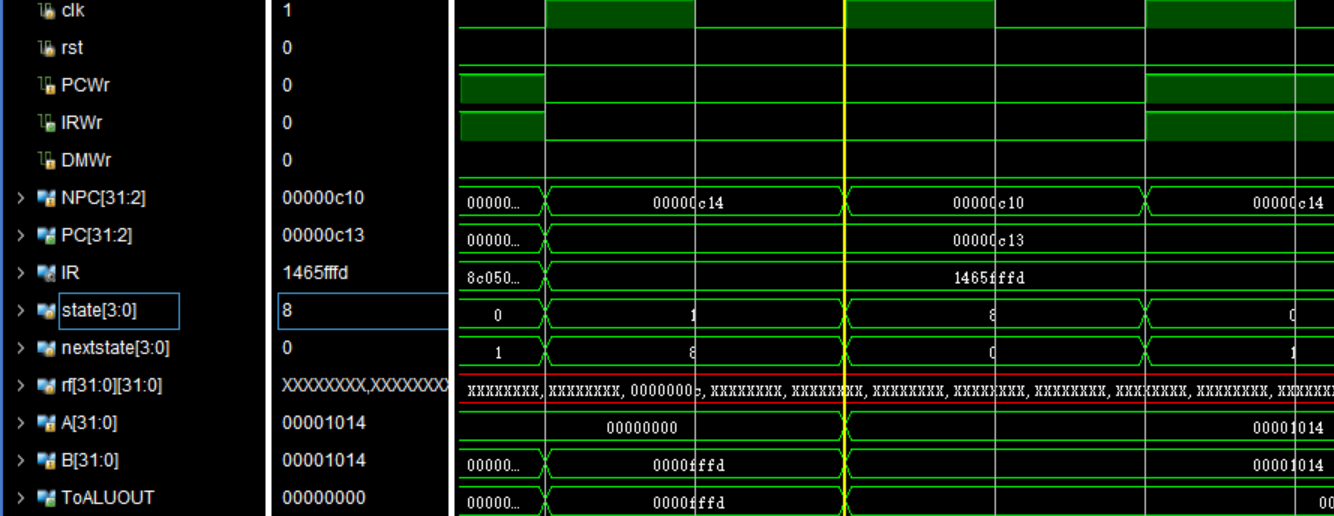
State0结束时，指令机器码从IM中取出存到IR中，NPC写入PC，State0下一阶段为state1（译码，取操作数存到AB阶段）,也是每个PC的开始阶段



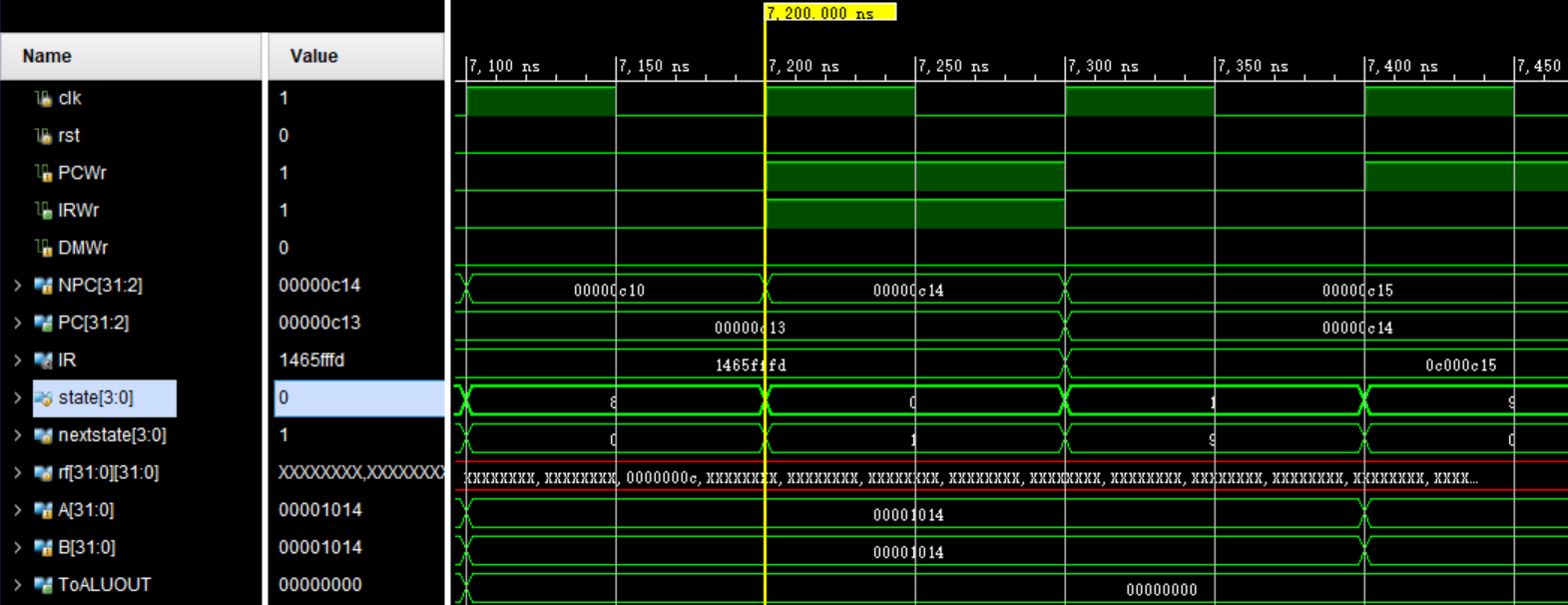
B型指令，State1阶段将操作数RS,RT分别存于A,B，state1下一阶段为state8（跳转阶段，执行A,B减法影响Zero标志位，Zero等于1，则PC🠄NPC(PC,imm16)）







State8阶段A,B作ALUOP（从指令OP判断得出）相应操作，结果存入ALUOUT，并影响ZERO标志位，由于相等，所以此次不跳转，PC写选通，正常进入下一指令。



最后回到state0阶段取下一条指令。

# 八、FPGA程序烧录测试

# 九、心得体会

硬件综合设计是目前接触到的比较庞大的的工程，尤其是在参考资料不如C语言等丰富的情况下，只能从起点入门书籍一步步着手，疑难解答网页很少，大多要实际探讨。在开发过程中，发现计算机组成原理，计算机体系结构，汇编语言，嵌入式等中的很多知识都融会贯通。CPU要实现指令的功能，就要从指令处理入手，多周期的指令处理到执行就要考虑指令阶段分级，取址，译码，执行，访存，写回。再进一步就要从模块设计入手，功能模块分别实现对应功能，控制模块统合功能实现，最后搭建数据通路，对信号的定义也要谨慎细心。综合过程中还出现了很多细节性问题，在同学老师帮助下逐步排除问题，终于成功解决。

在设计CPU过程中，感觉自己对计算机微处理器的工作原理更加了解了，尤其是汇编指令与机器语言以及机器语言的执行过程。

遗憾的是此次只完成了多周期，为流水打下基础，没有真正实现流水，希望在今后学习中有机会去实现。

# 十、程序附件



# 十一、参考文献

[1]自己动手写CPU[M]. 电子工业出版社 , 雷思磊, 2014

[2]汤勇明，等.搭建你的数字积木:数字电路与逻辑设计 [M]. Verilog HDL&Vivado版.北京:清华大学出版社，2017

[3]杨朝辉. 基于FPGA的开放式CPU模型研究[D].西安石油大学,2018.

[4]周宁宁,刘胜.基于FPGA技术的CPU模型机的设计与实现[J].南京邮电学院学报,2003(01):77-80.

[5]王力生,余智铭,张冬冬.“计算机组成原理”实验CPU设计方法研究[J].实验技术与管理,2018,35(05):1-5+9.

[6]康磊,杨朝辉.基于FPGA的开放式CPU实验模型设计[J].电脑知识与技术,2017,13(30):235-237.

[7]唐朔飞.计算机组成原理[M].2版.北京:高等教育出版社，2008

[8]张晨曦，王志英，等.计算机系统结构[M].2版.北京:高等教育出版社，2014