**“计算机设计与实践”处理器实验设计报告**

姓名：霍峻杰

班级：11403202

学号：1140320206

哈尔滨工业大学计算机学院

2016年7月29 日

**目录**

[一：实验目的 3](#_Toc457575807)

[二：实验环境 3](#_Toc457575808)

[三：设计思想 3](#_Toc457575809)

[1、 CPU接口信号定义 3](#_Toc457575810)

[2、 CPU设计方案 3](#_Toc457575811)

[四：实验设计及测试 15](#_Toc457575812)

[1、 各模块设计及测试 15](#_Toc457575813)

[波形仿真 15](#_Toc457575814)

[2、 整体的设计及测试 19](#_Toc457575815)

[波形仿真 19](#_Toc457575816)

[五：下载 21](#_Toc457575817)

[六：遇到的问题及解决方法 25](#_Toc457575818)

[七：实验体会 25](#_Toc457575819)

## 一：实验目的

1. 掌握Xilinx ISE集成开发环境使用方法
2. 掌握VHDL语言
3. 掌握FPGA编程方法及硬件调试手段
4. 深刻理解处理器结构和计算机系统的整体工作原理

## 二：实验环境

Xilinx ISE集成开发环境，ModelSim或ISim仿真工具、COP2000实验平台或SD2100数字逻辑设计实验平台或Diligent Nexys3开发板。

## 三：设计思想

### CPU接口信号定义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 位数 | 来源/去向 | 方向 | 备注 |
| rst | 1 | 处理器板 | I | 高电平复位 |
| Clk | 1 | 处理器板 | I | 系统时钟 |
| ABUS | 16 | 主存储器 | O | 地址总线 |
| DBUS | 16 | 主存储器 | I/O | 数据总线 |
| nMERQ | 1 | 主存储器 | O | 存储器片选 |
| nRD | 1 | 主存储器 | O | 存储器读 |
| nWR | 1 | 主存储器 | O | 存储器写 |
| nBHE | 1 | 主存储器 | O | 高位字节访问允许 |
| nBLE | 1 | 主存储器 | O | 低位字节访问允许 |
| IOAD | 2 | FPGA1 | O | 地址总线 |
| IODBin | 8 | FPGA1 | I | IO输入 |
| IODBout | 8 | FPGA1 | O | IO输出 |
| nPREQ | 1 | FPGA1 | O | 外设允许访问 |
| nPRD | 1 | FPGA1 | O | 外设读 |
| nPWR | 1 | FPGA1 | O | 外设写 |

### CPU设计方案

#### 指令格式设计

1. 寄存器-寄存器指令

|  |  |  |  |
| --- | --- | --- | --- |
| OP | AD1 |  | AD2 |

3

0

2

7

8

10

11

15

1. 其他指令

0

7

8

10

11

15

|  |  |  |
| --- | --- | --- |
| OP | AD1 | AD2(X) |

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 操作码 | 指令码后半部分 | 说明 |
| Mov Ri, Rj | 00000 | Ri, 00000, Rj | Rj -> Ri |
| Mvi Ri, X | 00001 | Ri, X | X -> Ri |
| Adc Ri, Rj | 00010 | Ri, 00000, Rj | Ri + Rj + cy -> Ri |
| Adc Ri, X | 00011 | Ri, X | Ri + X + cy ->Ri |
| Sbb Ri, Rj | 00100 | Ri, 00000, Rj | Ri - Rj - cy -> Ri |
| Sbb Ri, X | 00101 | Ri, X | Ri - X - cy -> Ri |
| And Ri, Rj | 00110 | Ri, 00000, Rj | Ri and Rj -> Ri |
| And Ri, X | 00111 | Ri, X | Ri and X -> Ri |
| Or Ri, Rj | 01000 | Ri, 00000, Rj | Ri or Rj -> Ri |
| Or Ri, X | 01001 | Ri, X | Ri or X -> Ri |
| Clc | 01010 | 00000000000 | Cy = 0 |
| Stc | 01011 | 00000000000 | Cy = 1 |
| Not Ri | 01100 | Ri, 00000000 | Not Ri -> Ri |
| Sta Ri, X | 01101 | Ri, X | Ri -> R7//X |
| Lda Ri, X | 01110 | Ri, X | R7//X -> Ri |
| Jz Ri, X | 01111 | Ri, X | If Ri=0 then pc+-X->pc |
| In Ri, port | 10000 | Ri, port | [port] -> Ri |
| Out Ri, port | 10001 | Ri, port | Ri -> [port] |

#### 微操作定义

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 取址周期 | 间址周期 | 执行周期 |
| Mov Ri, Rj | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Rj->Ri |
| Mvi Ri, X | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | X(IR)->Ri |
| Adc Ri, Rj | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Ri+Rj+cy->Ri |
| Adc Ri, X | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Ri+X+cy->Ri |
| Sbb Ri, Rj | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Ri-Rj-cy->Ri |
| Sbb Ri, X | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Ri-X-cy->Ri |
| And Ri, Rj | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Ri and Rj -> Ri |
| And Ri, X | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Ri and X -> Ri |
| Or Ri, Rj | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Ri or Rj -> Ri |
| Or Ri, X | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Ri or X -> Ri |
| Clc | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | ‘0’ -> cy |
| Stc | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | ‘1’ -> cy |
| Not Ri | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Not Ri -> Ri |
| Sta Ri, X | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc | R7//X(IR)->Ad | Ad->MAR, 1->W  Ri->MDR  MDR->M(MAR) |
| Lda Ri, X | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc | R7//X(IR)->Ad | Ad->MAR, 1->R  M(MAR)->MDR  MDR->Ri |
| Jz Ri, X | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Zero(Ri)\*Ad+nzero(Ri)\*pc->pc |
| In Ri, port | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Ad->IO, 1->R  IO->Ri |
| Out Ri, port | Pc->MAR, 1->R  M(MAR)->MDR  MDR->IR  Pc+1->pc |  | Ad->IO, 1->W  Ri->IO |

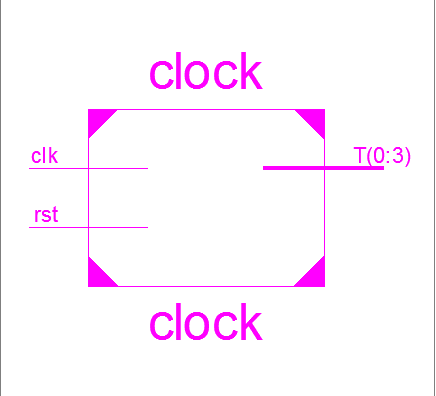
#### 节拍划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令 | T0 | T1 | T2 | T3 |
| Mov Ri, Rj | M(pc)->IR, 1->R | Rj->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| Mvi Ri, X | M(pc)->IR, 1->R | X->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| Adc Ri, Rj | M(pc)->IR, 1->R | Ri+Rj+cy->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| Adc Ri, X | M(pc)->IR, 1->R | Ri+X+cy->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| Sbb Ri, Rj | M(pc)->IR, 1->R | Ri-Rj-cy->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| Sbb Ri, X | M(pc)->IR, 1->R | Ri-X-cy->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| And Ri, Rj | M(pc)->IR, 1->R | Ri and Rj->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| And Ri, X | M(pc)->IR, 1->R | Ri and X->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| Or Ri, Rj | M(pc)->IR, 1->R | Ri or Rj->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| Or Ri, X | M(pc)->IR, 1->R | Ri or X->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| Clc | M(pc)->IR, 1->R | ‘0’->cy | Aluout->Rtmp | Pc+1->pc |
| Stc | M(pc)->IR, 1->R | ‘1’->cy | Aluout->Rtmp | Pc+1->pc |
| Not Ri | M(pc)->IR, 1->R | Not Ri->Ri | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| Sta Ri, X | M(pc)->IR, 1->R | Ri->aluout  R7//X->Ad | Aluout->Rtmp  Ad->MAR, 1->W  Rtmp->Rtmp | Pc+1->pc |
| Lda Ri, X | M(pc)->IR, 1->R | R7//X->Ad | Aluout->Rtmp  Ad->MAR, 1->W  Rtmp->M | Pc+1->pc |
| Jz Ri, X | M(pc)->IR, 1->R | Ad\*zero+pc\*nzero->aluout |  | PCaddr->pc |
| In Ri, port | M(pc)->IR, 1->R | IODBin->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |
| Out Ri, port | M(pc)->IR, 1->R | IODBin->aluout | Aluout->Rtmp | Rtmp->Ri, pc+1->pc |

#### 各模块设计

##### 时钟模块

**模块示意图：**



**模块功能：**

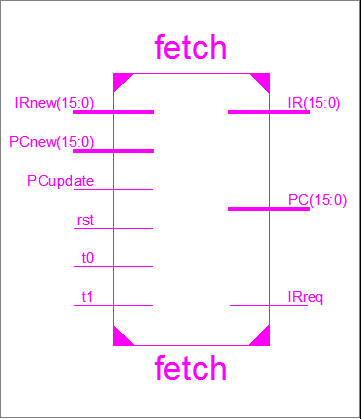
**产生4个节拍，其中，当rst=1时，节拍跳转到T3，并将pc置0.**

**接口说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **备注** |
| **Clk** | **1** | **I** | **处理器板** | **系统时钟** |
| **Rst** | **1** | **I** | **处理器板** | **高电平复位** |
| **T** | **4** | **O** | **取址，运算，存储，回写模块** | **4个节拍** |

##### 取指模块

**模块示意图：**



**模块功能：**

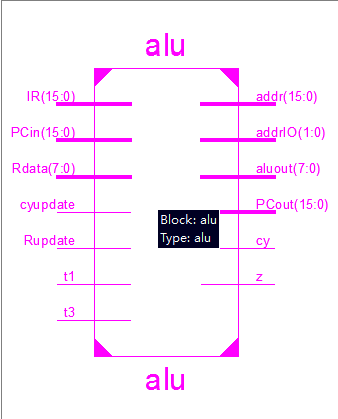
从内存取出指令。Rst=1时，pc置0.取出的IR传送给运算，存储，回写模块提供数据，当IRreq=1时，开始从内存取出指令。当pcupdate=1时，pc接收pcnew的数据，更新pc。

**接口说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **备注** |
| T0 | 1 | I | 时钟模块 | 第一节拍 |
| T1 | 1 | I | 时钟模块 | 第二节拍 |
| Rst | 1 | I | 处理器板 | 高有效复位信号 |
| PCupdate | 1 | I | 回写模块 | PC更新信号 |
| PCnew | 16 | I | 回写模块 | PC更新值 |
| IRnew | 16 | I | 存储控制模块 | IR更新值 |
| IRreq | 1 | O | 取址模块 | IR允许信号 |
| IR | 16 | O | 存储控制模块 | 导出IR到其他模块 |
| PC | 16 | O | 取址模块 | PC到运算模块 |

##### 运算模块

**模块示意图：**



**模块功能：**

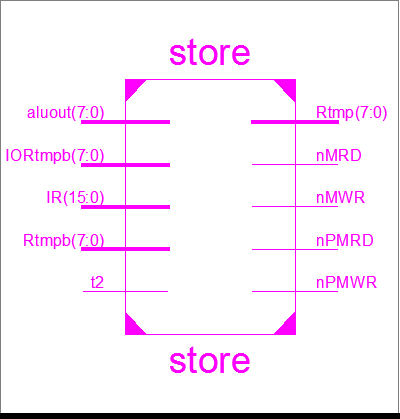
进行各种运算，并产生进位信号cy和零信号z。同时，进行pc的改变（加1或跳转），将改变后的值送给回写模块的PCnew存储等待回写。

**接口说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **备注** |
| T1 | 1 | I | 时钟模块 | 第二节拍 |
| T3 | 1 | I | 时钟模块 | 第四节拍 |
| Rupdate | 1 | I | 回写模块 | 寄存器回写信号 |
| Cyupdate | 1 | I | 回写模块 | Cy回写信号 |
| Rdata | 8 | I | 回写模块 | 寄存器回写数据 |
| PCin | 16 | I | 取址模块 | Pc地址 |
| IR | 16 | I | 取址模块 | 指令 |
| Addr | 16 | O | 访存控制模块 | 访问存储器的地址 |
| addrIO | 8 | O | IO控制模块 | IO地址 |
| Aluout | 8 | O | 存储模块 | 计算的数据 |
| PCout | 16 | O | 回写模块 | Pc改变之后的值 |
| Cy | 1 | O | FPGA1 | 进位信号 |
| z | 1 | O | FPGA1 | 零信号 |

##### 存储模块

**模块示意图：**



**模块功能：**

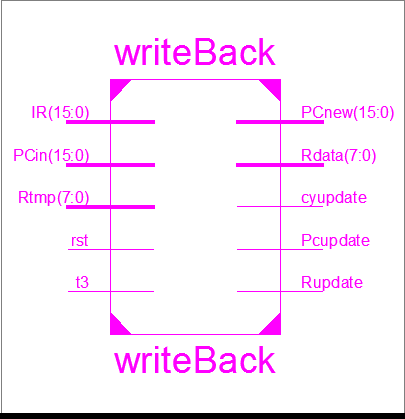
存储来自alu模块的数据到Rtmp，存数取数时进行与访存模块进行通信，取到的数存放在Rtmpb，再回写到Rtmp，统一进入回写模块等待回写。

**接口说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **备注** |
| Aluout | 8 | I | 运算模块 | 得到的运算结果 |
| IORtmpb | 8 | I | IO控制模块 | 取到的IO数据 |
| IR | 16 | I | 取址模块 | 指令 |
| Rtmpb | 8 | I | 存储模块 | 取到的主存数据 |
| T2 | 1 | I | 时钟模块 | 第三节拍 |
| Rtmp | 8 | O | 存储模块 | 存储寄存器 |
| nMRD | 1 | O | 访存控制模块 | 访存读信号 |
| nMWR | 1 | O | 访存控制模块 | 访存写信号 |
| nPMRD | 1 | O | IO控制模块 | IO读信号 |
| nPMWR | 1 | O | IO控制模块 | IO写信号 |

##### 回写模块

**模块示意图：**



**模块功能：**

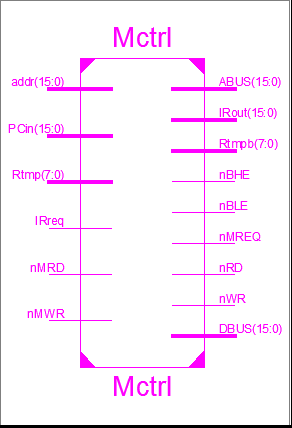
通过控制信号cyupdate，Rupdate控制cy和R的回写。

**接口说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **备注** |
| IR | 16 | I | 取址模块 | 指令 |
| PCin | 16 | I | 运算模块 | 回写的PC |
| Rtmp | 8 | I | 存储模块 | 计算的结果 |
| Rst | 1 | I | 处理器板 | 高电平复位 |
| T3 | 1 | I | 时钟模块 | 第四节拍 |
| PCnew | 16 | O | 取址模块 | 新的PC |
| Rdata | 8 | O | 运算模块 | 回写的数据 |
| Cyupdate | 1 | O | 运算模块 | Cy回写信号 |
| Pcupdate | 1 | O | 取址模块 | PC回写信号 |
| Rupdate | 1 | O | 运算模块 | R回写数据 |

##### 访存控制模块

**模块示意图：**



**模块功能：**

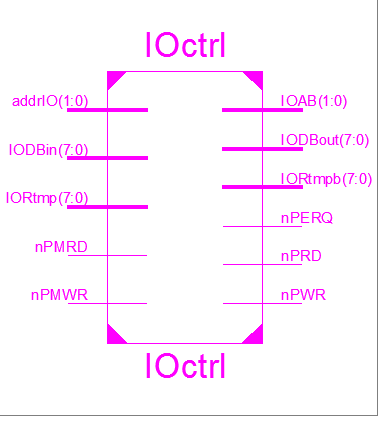
通过PC将指令从存储器中取到IR中，存数据取数据等需要访存的指令也都需要用到

**接口说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **备注** |
| **Addr** | **16** | **I** | **存储模块** | **访存地址** |
| **Pcin** | **16** | **I** | **取址模块** | **取指的地址** |
| **Rtmp** | **8** | **I** | **存储模块** | **存储数据** |
| **IRreq** | **1** | **I** | **取址模块** | **IR读取信号** |
| **nMRD** | **1** | **I** | **存储模块** | **存储器读有效** |
| **nMWR** | **1** | **I** | **存储模块** | **存储器写有效** |
| **ABUS** | **16** | **O** | **主存储器** | **地址总线** |
| **IRout** | **16** | **O** | **取址模块** | **取到的指令** |
| **Rtmpb** | **8** | **O** | **存储模块** | **回取得数据** |
| **nBHE** | **1** | **O** | **主存储器** | **高八位** |
| **nBLE** | **1** | **O** | **主存储器** | **低八位** |
| **nMREQ** | **1** | **O** | **主存储器** | **存储器读写控制信号** |
| **nRD** | **1** | **O** | **主存储器** | **存储器读** |
| **nWR** | **1** | **O** | **主存储器** | **存储器写** |
| **DBUS** | **16** | **O** | **主存储器** | **数据总线** |

##### IO控制模块

**模块示意图：**



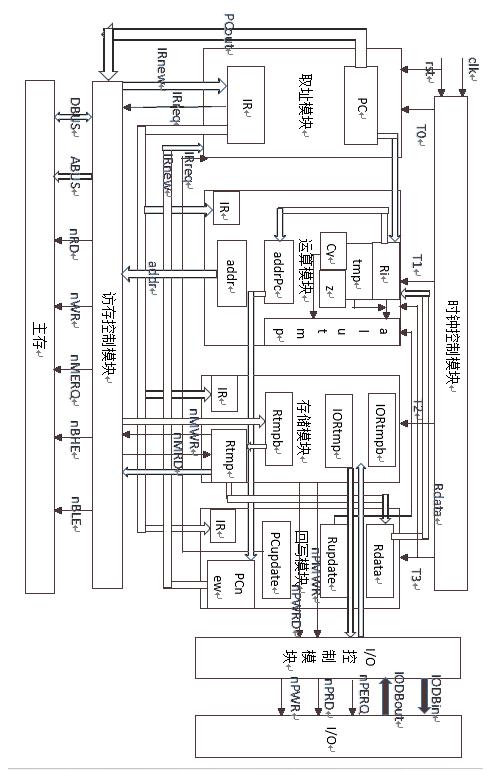
**模块功能：**

接收外设信号的输入，将寄存器中的数据输出到外设，将cpu与外设进行交互。

**接口说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **备注** |
| **AddrIO** | **2** | **I** | **存储模块** | **IO地址** |
| **IODBin** | **8** | **I** | **存储模块** | **IO数据输入** |
| **IORtmp** | **8** | **I** | **存储模块** | **IO存数据** |
| **nPMRD** | **1** | **I** | **存储模块** | **IO读控制** |
| **nPMWR** | **1** | **I** | **存储模块** | **IO写控制** |
| **IOAB** | **2** | **O** | **IO** | **IO地址总线** |
| **IODBout** | **8** | **O** | **IO** | **IO数据输出** |
| **IORtmpb** | **8** | **O** | **存储模块** | **IO取数据** |
| **nPERQ** | **1** | **O** | **IO** | **IO读写控制** |
| **nPRD** | **1** | **O** | **IO** | **IO读** |
| **nPWR** | **1** | **O** | **IO** | **IO写** |

#### CPU整体设计框图



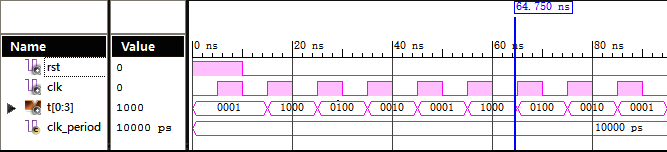
## 四：实验设计及测试

### 各模块设计及测试

波形仿真：

#### 时钟模块

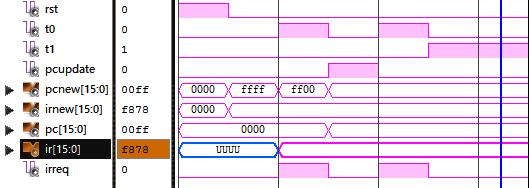
**波形演示**：



**波形解释**：当rst=1时，节拍变为第四节拍0001。当rst=0时，节拍变化按照一到四的顺序变化，即1000->0100->0010->0001->1000。

#### 取指模块

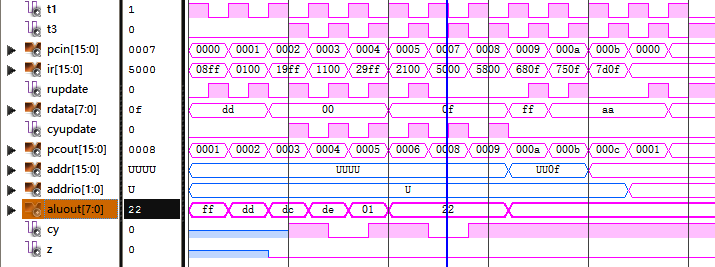
**波形演示**：



**波形解释**：rst=1时，pcnew,irnew,irreq都为0。Rst=0时，t0=1时，进行存储器读出指令，IRreq置为1，IRnew回写到IR中，当t1=1时，将IRreq置0。

#### 运算模块

**波形演示**：

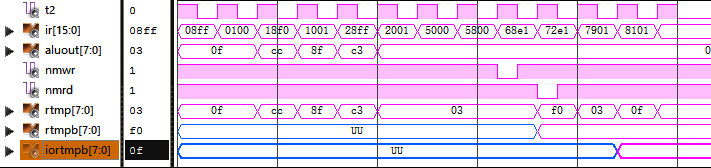


**波形解释**：

1. 将11111111写到R0寄存器中，t1=1时，将值写到aluout中，pc+1后送到PCout中，t3=1时，判断rupdate和cyupdate的值，若为1，则进行相应的回写，回写到运算模块中。
2. 将R0中的数据写入到R1中，t1=1时，将R0值写到aluout中，pc+1后送到PCout中，t3=1时，判断rupdate和cyupdate的值，若为1，则进行相应的回写，回写到运算模块的R1中。
3. R1加11111111加cy，t1=1时，将加到的结果写到aluout，得到11011100，产生了进位，所以cy置1，t3=1时，判断rupdate和cyupdate的值，若为1，则进行相应的回写，回写到运算模块中。
4. R0加R1加cy，结果送到R1，得到结果11011110，此时没有进位，cy变为0，t3回写。
5. R1减11111111减cy，t1=1时，将得到的结果写到aluout，得到00000001，产生了借位，所以cy置1，t3=1时，判断rupdate和cyupdate的值，若为1，则进行相应的回写，回写到运算模块中。
6. R0减R1减cy，结果送到R1，得到结果00100010，此时没有进位，cy变为0，t3回写。
7. Clc将cy置为0，当t3回写时，cy输出为0.
8. Stc将cy置为1，当t3回写时，cy输出为1.
9. Sta和lda都在t1=1时，计算出要访问存储器的地址11011101（测试数据），lda在t3=1时，将取到的数据回写到寄存器。
10. Jz指令判断R5是否为0，若为0，则pc加上输入的地址，若不为0，则pc正常加1，addr中可以看到此时计算出来的应加地址值为0000000000001111.
11. In指令与lda指令类似，计算出的地址为2位，放在addrIO中，t3=1时进行回写，使之进入到寄存器。
12. Out指令与sta相似，计算出的地址为2位，放在addrIO中。

#### 存储模块

**波形演示**：



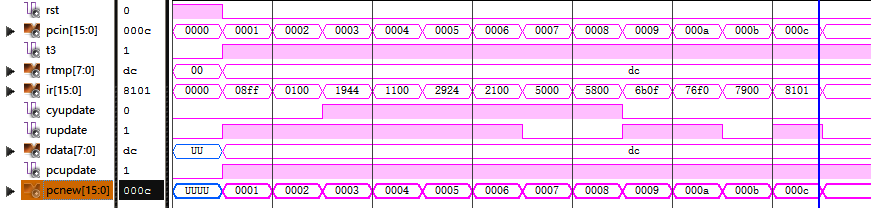
**波形解释**：t2=1时，计算得到的结果通过aluout传入，传到rtmp中，如图所示。

Sta指令时，nmwr置0，低有效，开始进行存储器写，从寄存器取出的数据经aluout传入rtmp。

Lda指令时，nmrd置0，低有效，开始进行存储器读，从存储器中取出的数据存到rtmpb中，传到回写模块等待回写到寄存器。

#### 回写模块

**波形演示**：

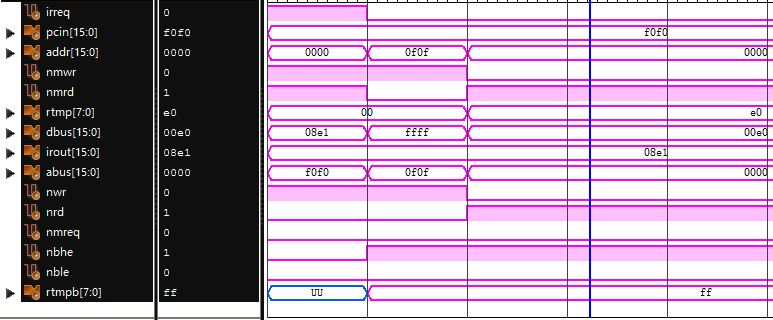


**波形解释**：

1. Clc指令和stc指令只将cyupdate置1，而Rupdate置0。
2. Adc，sbb指令将cyupdate和Rupdate置1.
3. Lda将Rupdate置1，而cyupdate置0.
4. Jz和out指令将cyupdate和Rupdate置0.
5. 其他指令将cyupdate置0，Rupdate置1.

#### 访存控制模块

**波形演示**：

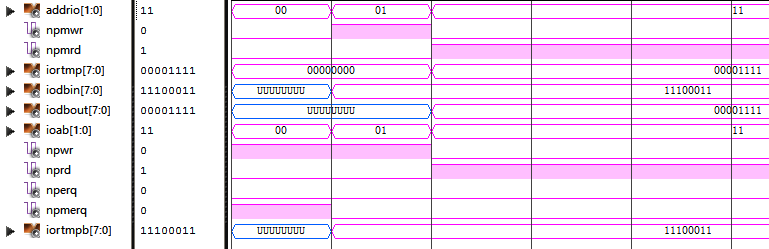


**波形解释**：

1. 当IRreq=1时，从主存储器中读出指令数据，nrd, nmerq置0，低有效。因为读出的是指令，为16位，所以低有效的nbhe, nble也为0.
2. 当IRreq=0时，当nmrd=1时，nrd, nmerq置0，低有效。因为读出的是数据，为8位，所以低有效的nble也为0，而nbhe置1.
3. 当IRreq=0时，当nmrd=1时，nwr, nmerq置0，低有效。因为读出的是数据，为8位，所以低有效的nble也为0，而nbhe置1.

#### I/O控制模块

**波形演示**：

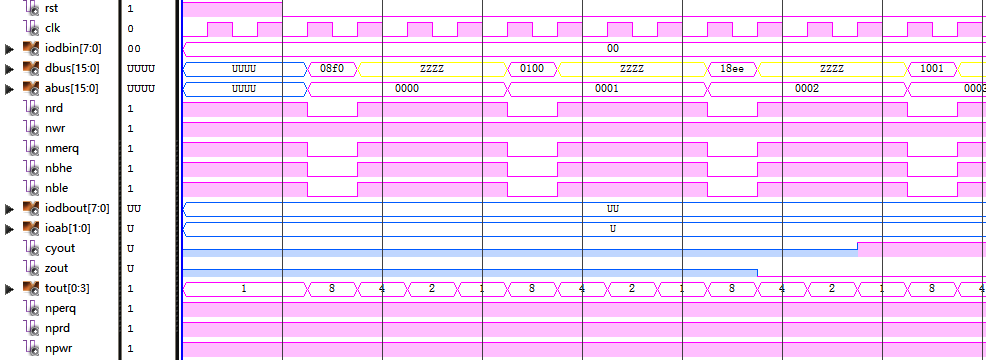


**波形解释**：

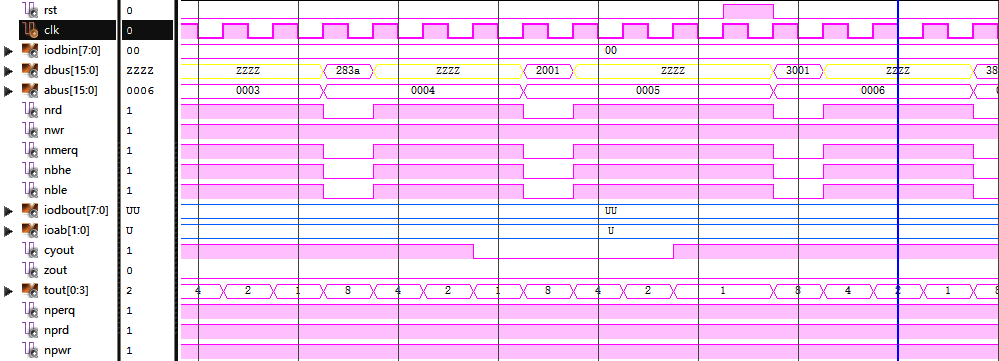
1. 当npmwr=1并且npmrd=1时，或当npmerq=1时，IO读写处于空闲状态，所有信号置1（无效）。
2. 当npmerq=0并且mpmrd=0时，nperq， nprd置0，npwr置1，从IODBin读入数据到IOrtmpb，等待回写到寄存器。
3. 当npmerq=0并且mpmwr=0时，nperq， npwr置0，nprd置1，数据从IOrtmp将数据写出到IODBout。

### 整体的设计及测试

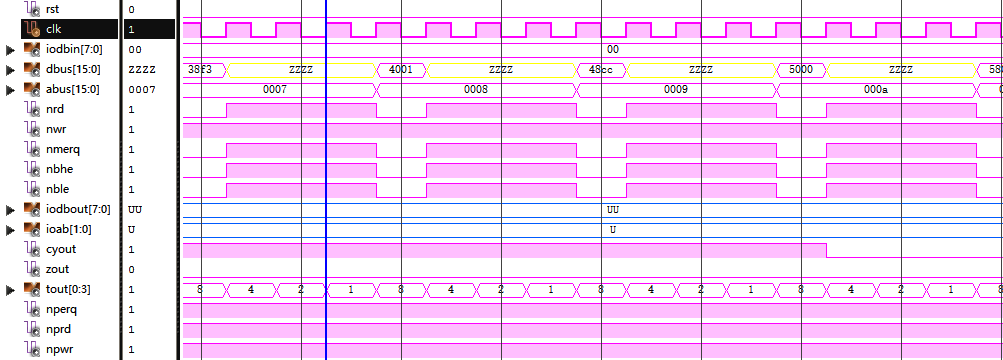
波形仿真：



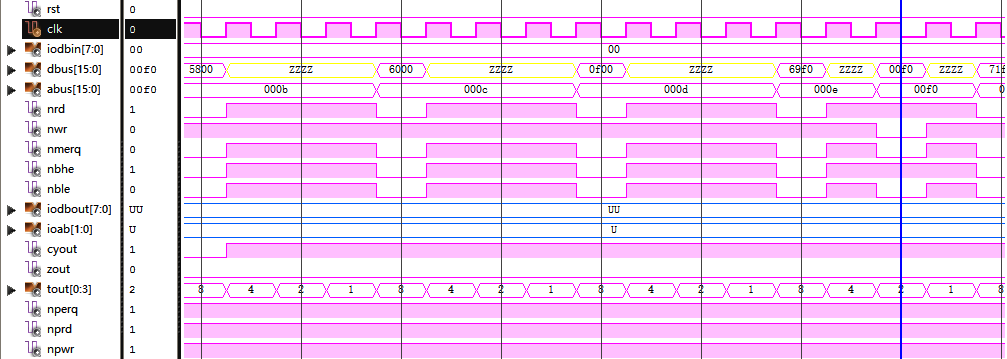
1. Rst=1时，节拍跳到第四节拍，pc置0.
2. Rst=0时，第一条指令mvi将11110000写入到R0寄存器，取址后，数据总线为当前IR指令，ABUS为当前pc值，读指令时，nrd，nmerq，nbhe，nble置0，nperq，nprd，npwr都置1.
3. 第二条指令为mov将R0的值写入到R1中，取址后，数据总线为当前IR指令，ABUS为当前pc值。
4. 第三条指令为adc，将R0的值加上一个立即数11101110和cy，此时加法已经溢出，在第三节拍的回写模块将cy置为1.
5. 第四条指令为adc，将R1的值加上R0和cy送到R0，加法保持溢出，cy为1.



1. 第五条指令为sbb，R0减去立即数00111010和cy，送到R0中，此时没有溢出，cy置为0。
2. 第六条指令为sbb，R0减去R1的值和cy，送到R0中，cy又发生溢出，cy置为1.
3. 第七条指令为and，R0和R1的值相与，送到R0中。
4. 第八条指令为and，R0和立即数11110011相与，结果送到R0中。



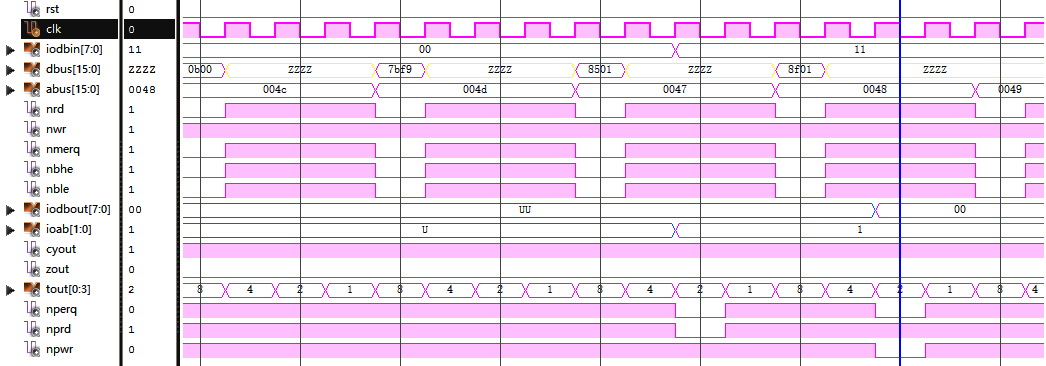
1. 第九条指令为or，R0和R1的值相或，送到R0中。
2. 第十条指令为or，R0和立即数11001100相或，送到R0中。
3. 第十一条指令为clc，将cy清零。
4. 第十二条指令为stc，将cy置1.



1. 第十三条指令为not，将R0的值取反，送到R0.
2. 第十四条指令为mvi，将R7置全0.
3. 第十五条指令为sta，地址为R7//11110000，将R1的值送到这个地址中，在访存时，nmerq，nwr，nble置0.
4. 第十六条指令为lda，将地址中的值取出送到R1中。



1. 第十七条指令为mvi，将R6传入000000011。
2. 第十八条指令为jz，判断R6不为0，pc+1->pc。
3. 第十九条指令为jz，判断R7为0，pc直接跳转到01001100（pc+00111001+1）。
4. 第二十条指令为mvi，将R3置全0.



1. 第二十一条指令为jz，判断传入的地址为负数（向前跳转），跳转到01000111.
2. 第二十二条指令为in，将输入的IODBin写到R5中，在数据总线上显示。
3. 第二十三条指令为out，将R5写到指定输出IODBout中。

## 五：下载

#### 指令序列

|  |  |  |  |
| --- | --- | --- | --- |
| 地址 | 指令 | 二进制指令 | 十六进制指令 |
| 0000 | Mvi R7, 00H | 0000111100000000 | 0F00 |
| 0001 | Sta R7, 2FH | 0110111100101111 | 6F2F |
| 0002 | Mvi R0, 12H | 0000100000010010 | 0812 |
| 0003 | Mov R1, R0 | 0000000100000000 | 0100 |
| 0004 | Sta R1, 2EH | 0110100100101110 | 692E |
| 0005 | Stc | 0101100000000000 | 5800 |
| 0006 | Clc | 0101000000000000 | 5000 |
| 0007 | Adc R0, FFH | 0001100011111111 | 18FF |
| 0008 | Sta R0, FFH | 0110100011111111 | 68FF |
| 0009 | Adc R0, R1 | 0001000000000001 | 1001 |
| 000A | Sta R0, FEH | 0110100011111110 | 68FE |
| 000B | Sbb R0, 13H | 0010100000010011 | 2813 |
| 000C | Sta R0, FDH | 0110100011111101 | 68FD |
| 000D | Sta R0, FCH | 0110100011111100 | 68FC |
| 000E | Lda R1, FCH | 0111000111111100 | 71FC |
| 000F | Sta R1, 2DH | 0110100100101100 | 692D |
| 0010 | Mvi R6, 12H | 0000111000010010 | 0E12 |
| 0011 | Jz R7, 31H | 0111111100110001 | 7F31 |
| 0043 | Jz R7, F2H | 0111111111110010 | 7FF2 |
| 0036 | Jz R6, 31H | 0111111000110001 | 7E31 |
| 0037 | In R5, 00 | 1000010100000000 | 8500 |
| 0038 | Sta R5, 2CH | 0110110100101100 | 6D2C |
| 0039 | Out R5,00 | 1000110100000000 | 8D00 |
| 0040 | Sbb R0, R0 | 0010000000000000 | 2000 |
| 0041 | Sta R0, 2BH | 0110100000101011 | 682B |
| 0042 | And R1, R6 | 0011000100000110 | 3106 |
| 0043 | Sta R1, 1FH | 0110100100011111 | 691F |

#### Ucf文件

NET"rst" LOC="P33";

NET"clk" LOC="P92";

NET"ABUS(15)" LOC="P187";

NET"ABUS(14)" LOC="P205";

NET"ABUS(13)" LOC="P206";

NET"ABUS(12)" LOC="P207";

NET"ABUS(11)" LOC="P208";

NET"ABUS(10)" LOC="P185";

NET"ABUS(9)" LOC="P186";

NET"ABUS(8)" LOC="P176";

NET"ABUS(7)" LOC="P175";

NET"ABUS(6)" LOC="P174";

NET"ABUS(5)" LOC="P173";

NET"ABUS(4)" LOC="P157";

NET"ABUS(3)" LOC="P156";

NET"ABUS(2)" LOC="P155";

NET"ABUS(1)" LOC="P154";

NET"ABUS(0)" LOC="P153";

NET"DBUS(15)" LOC="P192";

NET"DBUS(14)" LOC="P193";

NET"DBUS(13)" LOC="P194";

NET"DBUS(12)" LOC="P195";

NET"DBUS(11)" LOC="P199";

NET"DBUS(10)" LOC="P200";

NET"DBUS(9)" LOC="P201";

NET"DBUS(8)" LOC="P202";

NET"DBUS(7)" LOC="P170";

NET"DBUS(6)" LOC="P169";

NET"DBUS(5)" LOC="P168";

NET"DBUS(4)" LOC="P167";

NET"DBUS(3)" LOC="P163";

NET"DBUS(2)" LOC="P162";

NET"DBUS(1)" LOC="P161";

NET"DBUS(0)" LOC="P160";

NET "IRout<0>" LOC = "p223" ;

NET "IRout<10>" LOC = "p232" ;

NET "IRout<11>" LOC = "p231" ;

NET "IRout<12>" LOC = "p230" ;

NET "IRout<13>" LOC = "p229" ;

NET "IRout<14>" LOC = "p228" ;

NET "IRout<15>" LOC = "p224" ;

NET "IRout<1>" LOC = "p222" ;

NET "IRout<2>" LOC = "p221" ;

NET "IRout<3>" LOC = "p220" ;

NET "IRout<4>" LOC = "p218" ;

NET "IRout<5>" LOC = "p217" ;

NET "IRout<6>" LOC = "p216" ;

NET "IRout<7>" LOC = "p215" ;

NET "IRout<8>" LOC = "p235" ;

NET "IRout<9>" LOC = "p234" ;

NET"nMERQ" LOC="P159";

NET "nMERQ1" LOC = "p147" ;

NET"nRD" LOC="P188";

NET "nRD1" LOC = "p152" ;

NET"nWR" LOC="P171";

NET "nWR1" LOC = "p178" ;

NET"nBHE" LOC="P189";

NET"nBLE" LOC="P191";

NET"IODBout(7)" LOC="P78";#B7

NET"IODBout(6)" LOC="P93";#B6

NET"IODBout(5)" LOC="P99";#B5

NET"IODBout(4)" LOC="P107";#B4

NET"tout(0)" LOC="P108"; #B3

NET"tout(1)" LOC="P109"; #B2

NET"tout(2)" LOC="P124";#B1

NET"tout(3)" LOC="P125";#B0

NET"IODBin(7)" LOC="P94";

NET"IODBin(6)" LOC="P95";

NET"IODBin(5)" LOC="P96";

NET"IODBin(4)" LOC="P97";

NET"IODBin(3)" LOC="P100";

NET"IODBin(2)" LOC="P101";

NET"IODBin(1)" LOC="P102";

NET"IODBin(0)" LOC="P103";

NET "nPRD" LOC = "p110" ;

NET "nPERQ" LOC = "p203" ;

NET "nPWR" LOC = "p111" ;

NET"cyout" LOC="P184"; #A4

NET"zout" LOC="P177"; #A3

NET "ABUSout<0>" LOC = "p74" ;

NET "ABUSout<1>" LOC = "p68" ;

NET "ABUSout<2>" LOC = "p67" ;

NET "ABUSout<3>" LOC = "p57" ;

NET "ABUSout<4>" LOC = "p52" ;

NET "ABUSout<5>" LOC = "p46" ;

NET "ABUSout<6>" LOC = "p42" ;

NET "ABUSout<7>" LOC = "p31" ;

NET "ABUSout<8>" LOC = "p7" ;

NET "ABUSout<9>" LOC = "p6" ;

NET "ABUSout<10>" LOC = "p5" ;

NET "ABUSout<11>" LOC = "p4" ;

NET "ABUSout<12>" LOC = "p3" ;

NET "ABUSout<13>" LOC = "p238" ;

NET "ABUSout<14>" LOC = "p237" ;

NET "ABUSout<15>" LOC = "p236" ;

NET "DBUSout<0>" LOC = "p19" ;

NET "DBUSout<1>" LOC = "p18" ;

NET "DBUSout<2>" LOC = "p17" ;

NET "DBUSout<3>" LOC = "p13" ;

NET "DBUSout<4>" LOC = "p12" ;

NET "DBUSout<5>" LOC = "p11" ;

NET "DBUSout<6>" LOC = "p10" ;

NET "DBUSout<7>" LOC = "p9" ;

NET "DBUSout<8>" LOC = "p28" ;

NET "DBUSout<9>" LOC = "p27" ;

NET "DBUSout<10>" LOC = "p26" ;

NET "DBUSout<11>" LOC = "p25" ;

NET "DBUSout<12>" LOC = "p24" ;

NET "DBUSout<13>" LOC = "p23" ;

NET "DBUSout<14>" LOC = "p21" ;

NET "DBUSout<15>" LOC = "p20" ;

## 六：遇到的问题及解决方法

1. 一开始，存储模块的Rtmp始终无法和时钟节拍的跳变同步变化，后来在学长的帮助下，在敏感信号表里加入了Rtmp，就可以了。
2. 在写jz回跳时，遇到了一些问题。老师告诉我一个好方法，判断传入地址的最高位，然后补上全0或全1，就好使了。
3. 下载后，发现out指令不能将数据输出到LED灯上。发现代码中有一句多余的，导致传输的时候产生冲突，注释后，就可以正常输出了。
4. 有时候板子不好使。重新启动实验台。
5. 帮同学解决问题时，发现他的全局变量定义和端口信号定义一致，这样会产生冲突。改变其中一个，使其不一样。
6. 一个信号只能对应绑定一个管脚，这也是帮其他人解决的问题。

## 七：实验体会

这次cpu的 实验虽然在做的过程中非常痛苦，但是回顾一下，是苦中带甜的。我们在其中学到了许多从理论上得不到的东西。更加深刻地理解了cpu的各部分组成及其指令执行过程。除此之外，还更加深刻的理解了VHDL语言，元件例化等简单地VHDL知识现在基本已经没问题了。在实验的过程中，我们耐下心来，从简单的设计，到写出每个模块的VHDL代码，再到将每个模块连接成整个的CPU，然后进行波形仿真，最后下载到板子上，每一步都用到前面实验的基础。其实在刚开始时，进行到每一步都感觉非常的困难，但是，一努力，一咬牙就坚持下来，最后下载通过后，感到无比的欣喜和自豪。其实，最后成功之后，往前看看，觉得前面的那些真的不是很难，但是没入门就会觉得很困难，庆幸的是，我通过毅力坚持了下来，并最终通过自己的努力，自主的设计并作出了一个属于自己的CPU。在实验过程中，自己也遇到了许多问题，通过自己的钻研和学长老师的指导，我解决了这些问题，并且感觉学到了更多的东西。在完成实验后，我并没有急着去做其他事情，我得到了许多帮助，应该多帮助别人。我第二天去帮助了同学，尽自己的微薄力量，让一些同学解决了问题，大多数是我之前遇到过的。

总体来说，这次CPU实验应该会令我印象十分深刻，也是目前为止，硬件方面使我提升最大的一门课。虽然过程并不甜美，但结果很好，我很感谢这门课以及徐老师，助教，同学。