“计算机设计与实践”处理器实验设计报告

姓名：张茗帅

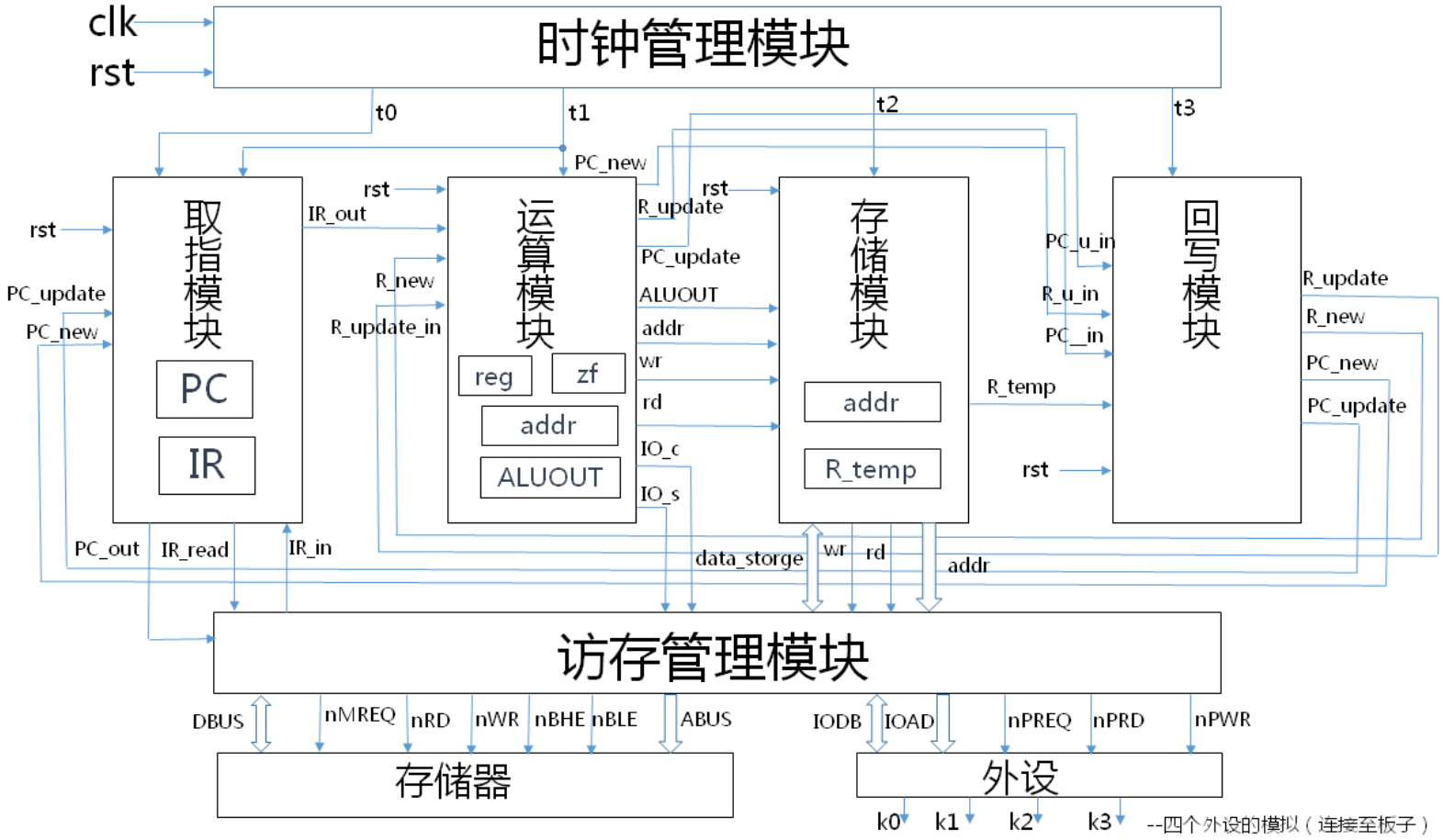
班级：1403106

学号：1140310606

哈尔滨工业大学计算机学院

2016 年 7 月

一、 详细设计整体框图



功能描述：

由于每一个指令的周期包含四个机器周期：取指周期，运算周期，访存周期，回写周期，不妨设每一个机器周期包含一个时钟节拍。本系统通过时钟管理模块控制节拍、进行调度，使各个模块依次有效。

取指模块负责指令的取指周期；运算模块负责指令的运算周期；存储模块负责指令的访存周期；回写模块负责指令的回写周期（各模块的具体功能见“二、各模块接口详细功能”）。四个机器周期依次执行，使得 CPU 可以完整的执行一条指令。

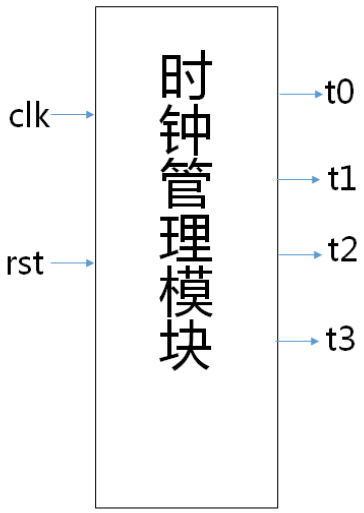
在 CPU 和存储器之间的访存管理模块主要解决各模块与主存交换信息时的冲突问题。由于取值周期和一些指令的存储周期都需要进行访存，故设此模块来控制这两个模块与存储器之间的数据传输。

同时，访存管理模块也作为 CPU 与外设连接的一个中转通道，使得 CPU 对外设的相关操作通过访存管理向外设传输相关数据信息和控制信息。

二、接口的详细说明

1 时钟管理模块

接口说明：



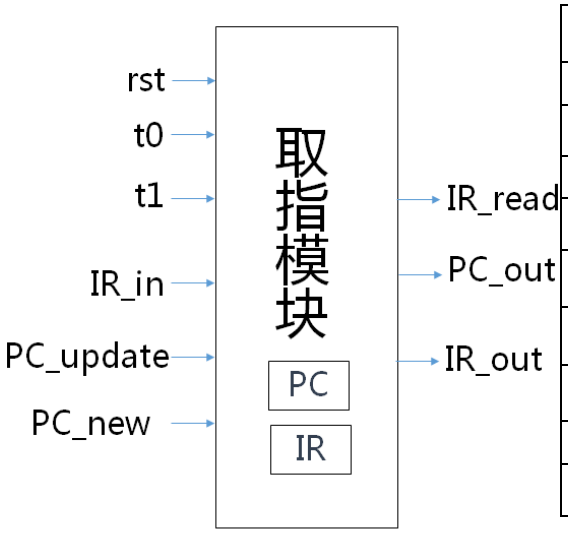
信号名	位数	方向	来源/去向	信号意义
clk	1	in	处理器板	系统时钟信号
rst	1	in	处理器板	复位信号
t0	1	out	取指模块	节拍信号（驱动取指模块）
t1	1	out	运算模块	节拍信号（驱动运算模块）
t2	1	out	存储模块	节拍信号（驱动存储模块）
t3	1	out	回写模块	节拍信号（驱动回写模块）

功能说明:

该模块根据 clk 产生，四个节拍依次按顺序有效，控制一条指令的 4 个周期按照顺序执行。

2 取指模块

接口说明：

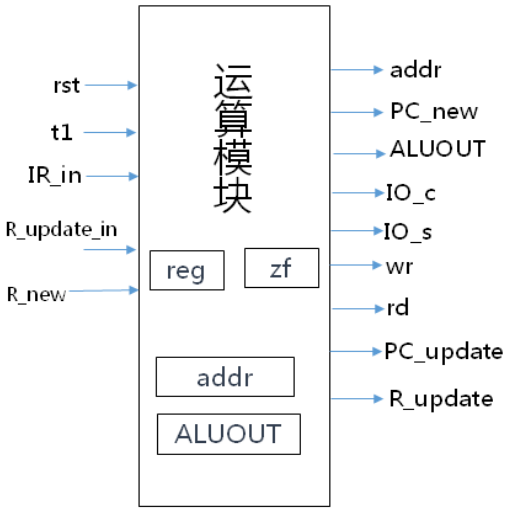


信号名	位数	方向	来源/去向	信号意义
rst	1	in	处理器板	系统时钟信号
t0	1	in	时钟模块	时钟节拍
t1	1	in	时钟模块	时钟节拍
IR_in	16	in	访存管理模块	即将执行的指令码
PC_new	16	in	回写模块	PC_update 有效时新的 PC
PC_update	1	in	回写模块	PC 更新控制信号
IR_out	16	out	运算模块	向运算模块传指令码
IR_read	1	out	访存管理模块	读指令控制信号
PC_out	16	out	访存管理模块	向存储器发送当前 PC

功能描述：取指周期完成指令的提取，并在 t1 节拍完成 PC+1 的操作，若 PC_update 有效时，PC_new 的值赋给 PC，为下一次取值做准备。当 rst 有效时，系统复位。

3 运算模块

接口说明：

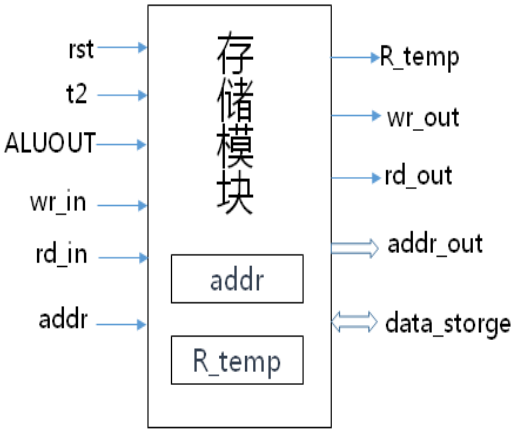


信号名	位数	方向	来源/去向	信号意义
rst	1	in	处理器板	系统时钟信号
t1	1	in	时钟模块	时钟节拍
IR_in	16	in	取指模块	即将执行的指令码
R_new	8	in	回写模块	R_pdate_in 有效时回写的值
R_pdate_in	1	in	回写模块	寄存器更新控制信号
addr	16	out	存储模块	计算得出数据的有效地址
PC_new	16	out	回写模块	计算得出的 PC 地址
ALUOUT	8	out	存储模块	计算得出的有效数据
IO_c	1	out	访存管理模块	外设启动信号
IO_s	1	out	访存管理模块	外设读写控制信号
wr	1	out	存储模块	存储器写控制信号
rd	1	out	存储模块	存储器读控制信号
PC_update	1	out	回写模块	PC 更新控制信号
R_update	1	out	回写模块	存储器更新控制信号

功能描述：在运算模块中，最重要的就是完成对指令的译码。根据不同的指令，完成相应操作数的准备和实际计算，运算包括数据加减和访存实际地址的计算。与此同时，每一条指令都产生所有相应的控制信号(如 wr,rd,IO_c,IO_s 等存储器和外设的控制信号)来控制该指令接下来的操作。

4 存储模块

接口说明



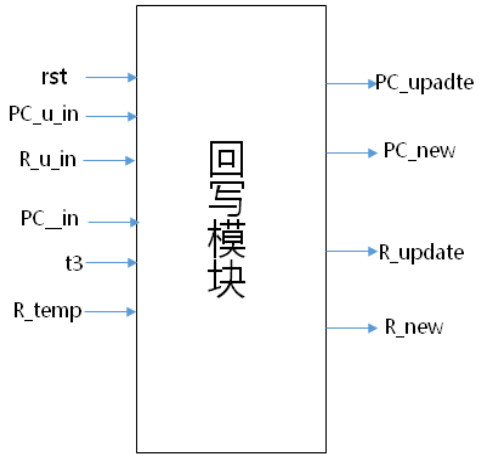
信号名	位数	方向	来源/去向	信号意义
rst	1	in	处理器板	系统时钟信号
t2	1	in	时钟模块	时钟节拍
ALUOUT	8	in	运算模块	有效运算的结果
wr_in	1	in	运算模块	存储器写控制
rd_in	1	in	运算模块	存储器读控制
addr	16	in	运算模块	运算得到的有效地址
R_temp	8	out	回写模块	存储器相关数据(写入或取出)，不访存时为 ALUOUT
wr_out	1	out	访存管理模块	存储器写控制
rd_out	1	out	访存管理模块	存储器读控制
addr_out	16	out	访存管理模块	访存的地址
data_storge	8	inout	访存管理模块	存储器的相关数据

功能描述：根据运算模块传输进来的数据进行相应的访存操作，或者不访存直接传递至回写模块。他们都是根据传递进来的存储器读写控制信号来进行控制的。此模块也将各种控制信息(wr,rd)，地址信息(addr)，数据信息(data_storge)传入访存控制模块。R_temp 为暂存器，是数据传输的中介。

5 回写模块

功能描述：回写模块将访存的结果或运算的结果回写入相应 CPU 内部寄存器中保存，同时根据转移指令是否满足(JZ,JMP 无条件)来决定是否更新 PC 寄存器中的内容。各控制信号仅在本模块启动时有效(相当于运算模块传来的控制信号在 t3 时才有效，进行回写)。

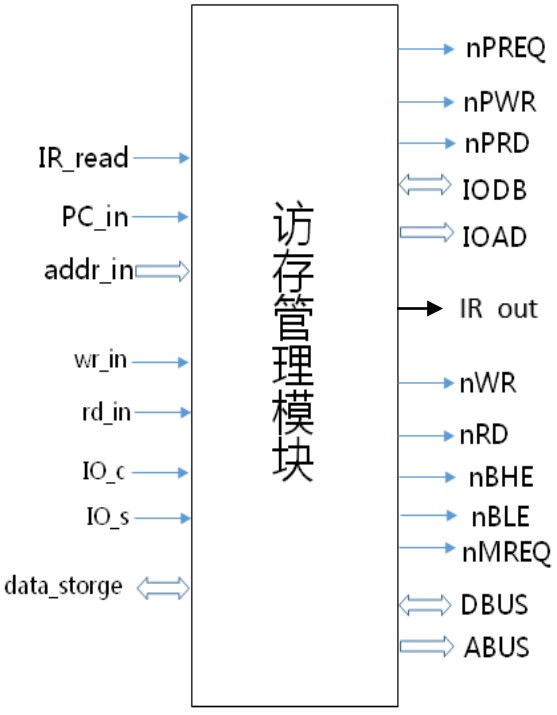
接口说明：



信号名	位数	方向	来源/去向	信号意义
rst	1	in	处理器板	系统时钟信号
t3	1	in	时钟模块	时钟节拍
PC_u_in	1	in	运算模块	决定 PC 更新控制信号
R_u_in	1	in	运算模块	决定寄存器更新控制信号
PC_in	16	in	运算模块	更新的 PC 地址
R_temp	8	in	存储模块	更新的寄存器新的数据
PC_update	1	out	取指模块	PC 更新控制信号(仅在本模块有效)
PC_new	16	out	取指模块	更新的 PC 地址
R_update	1	out	运算模块	寄存器更新控制信号(仅在本模块有效)
R_new	8	out	运算模块	更新的寄存器新的数据

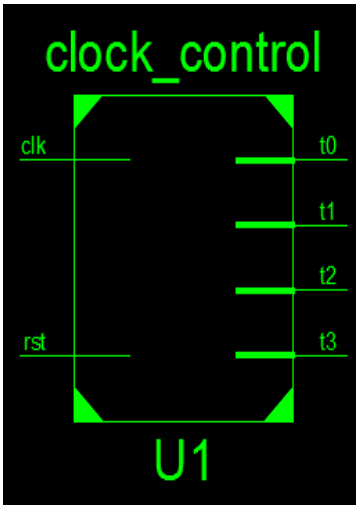
6 访存管理模块

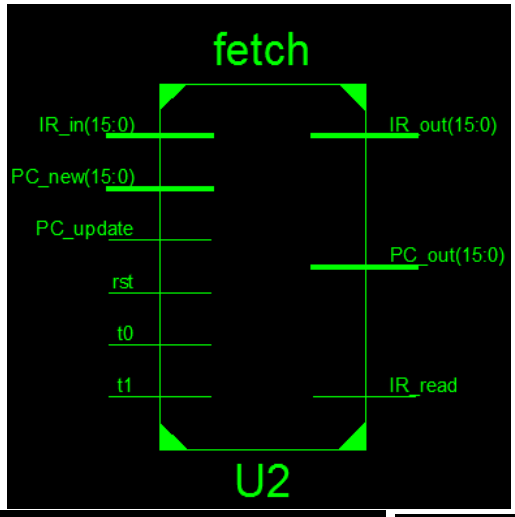
接口说明：

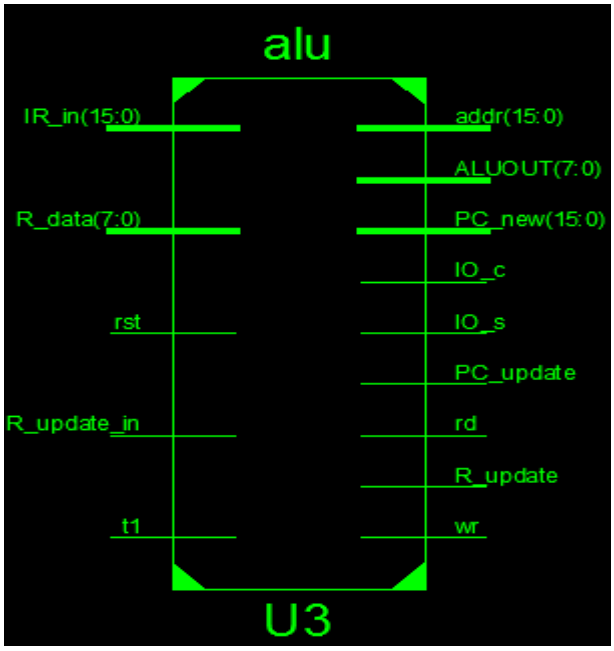


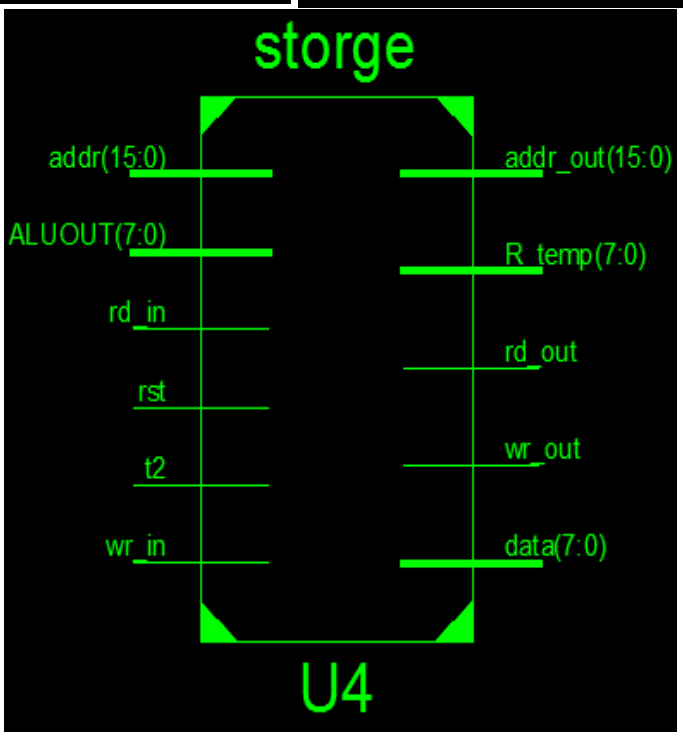
信号名	位数	方向	来源/去向	信号意义
IR_read	1	in	处理器板	读指令控制信号
PC_in	16	in	取指模块	PC 当前地址
addr_in	16	in	存储模块	地址(将用于存储器或外设)
wr_in	1	in	存储模块	存储器写控制信号
rd_in	1	in	存储模块	存储器读控制信号
IO_c	1	in	运算模块	外设启动信号
IO_s	1	in	运算模块	外设读写控制信号
IR_out	16	in	取指模块	即将执行的指令码
nPREQ	1	out	FPGA1	外设访问允许
nPWR	1	out	FPGA1	外设写
nPRD	1	out	FPGA1	外设读
IOAD	2		FPGA1	外设地址总线
nWR	1	out	主存储器	存储器写控制信号
nRD	1	out	主存储器	存储器读控制信号
nBHE	1	out	主存储器	存储器高位字节访问允许
nBLE	1	out	主存储器	存储器低位字节访问允许
nMREQ	1	out	主存储器	存储器片选
ABUS	16	out	主存储器	存储器地址总线
data_storge	8	inout	存储模块	与存储模块交换的数据
DBUS	16	inout	主存储器	存储器数据总线
IODB	8	inout	FPGA1	外设数据总线

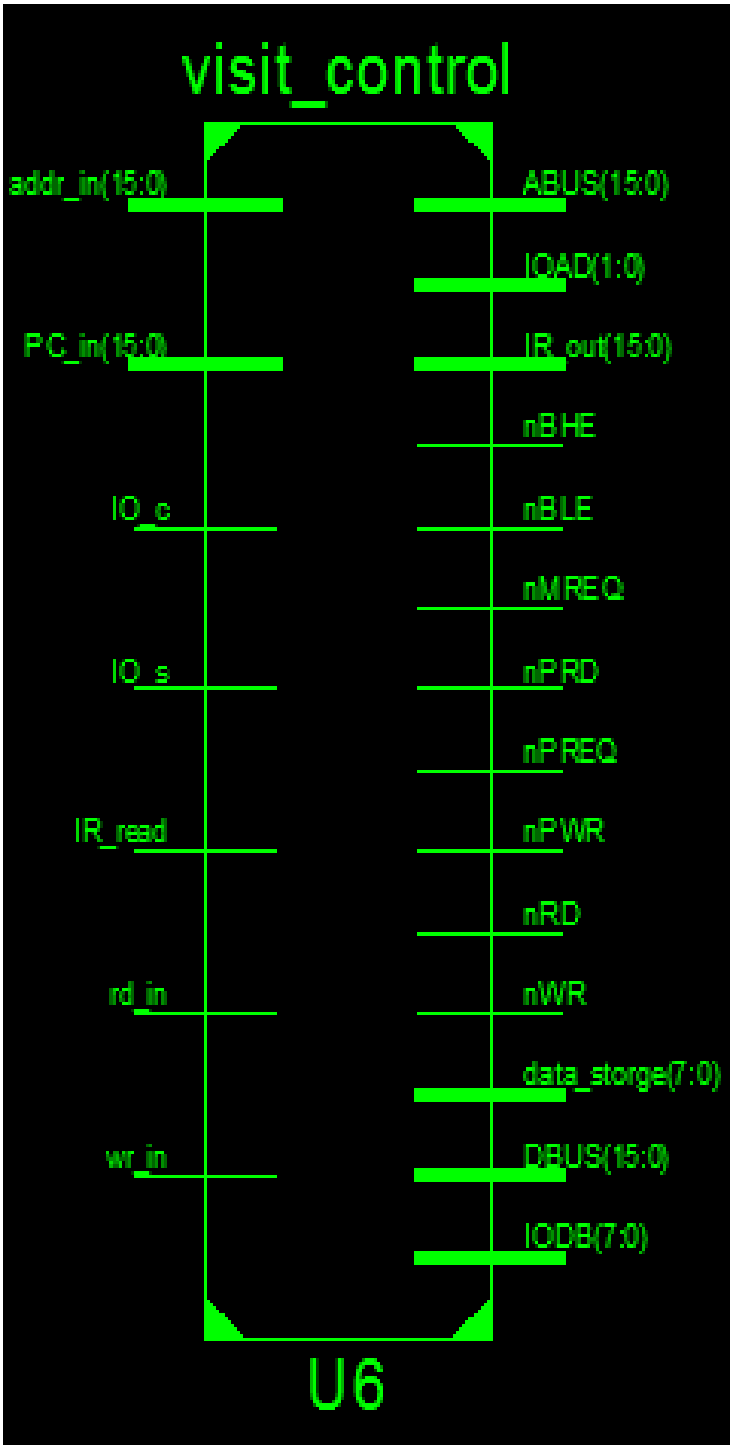
各模块在 ISE 中生成的器件图：

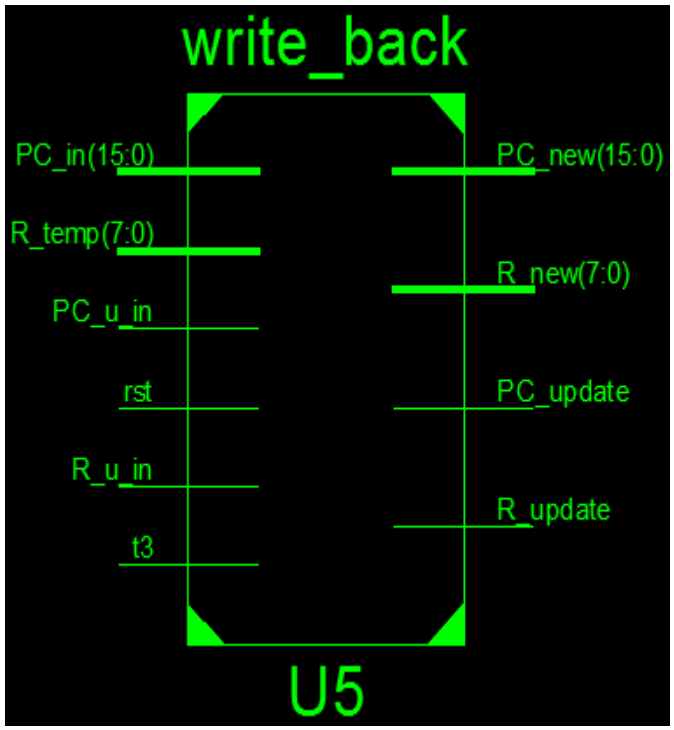










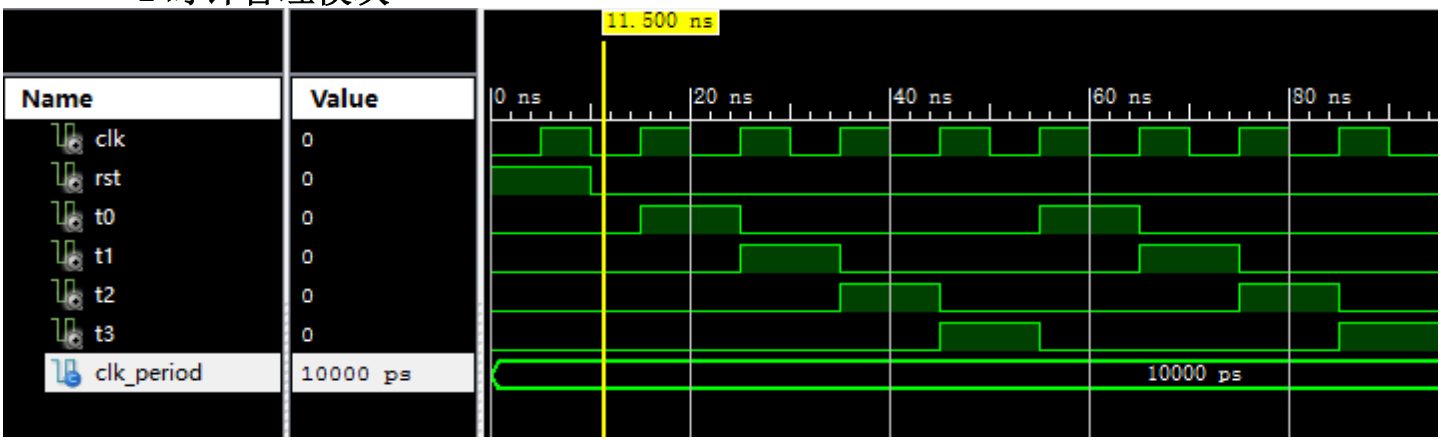


(访存管理模块)功能描述：访存管理模块主要解决各模块与主存交换信息时的冲突问题。

当取指模块的 IR_read 信号有效时，根据 PC 内容访问存储器，将 IR 取出并送回至取指模块；根据存储模块给出的地址、读写控制信号进行对存储器相应的读写操作。同时，访存管理模块也作为 CPU 与外设连接的一个中转通道，使得 CPU 对外设的相关操作通过访存管理向外设传输相关数据信息和控制信息，完成相应的外设操作(外设读或外设写)。

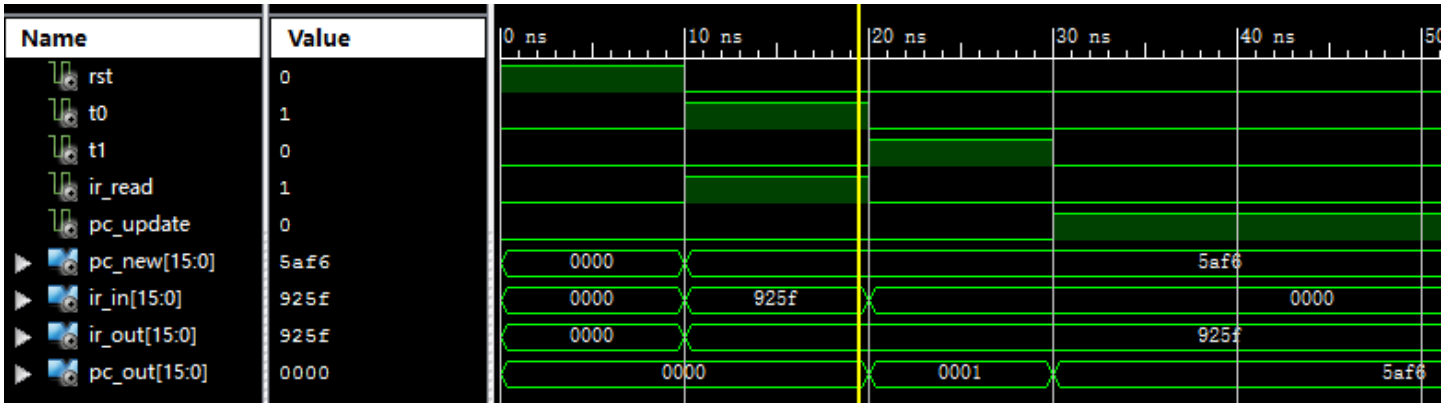
三、 系统仿真波形

1 时钟管理模块



说明：rst 无效时，四个时钟节拍按照顺序分别有效(在时钟上升沿触发)。

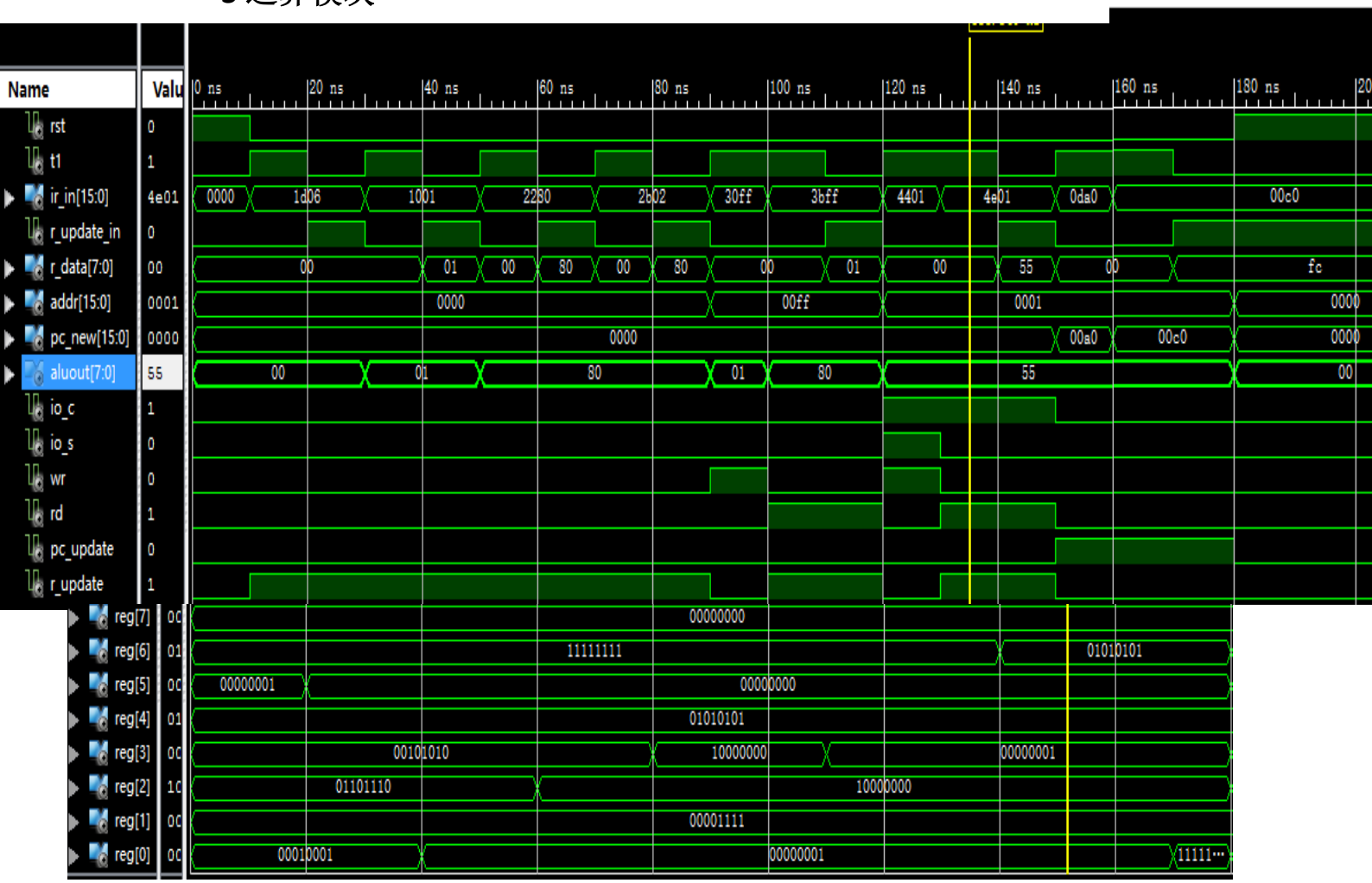
2 取指模块



说明：t0 节拍到来时，按照当前 PC 地址读指令(IR_read=1)。IR_out 值更新为 IR_in

的值；在 t1 节拍到来时，完成 PC+1 的操作；当 PC_update 有效时，将 PC_new 的值赋给 PC_out(更新 PC 寄存器的值)。

3 运算模块



说明：(在此，先说明下我设计的各个指令的格式和二进制操作码，

指令名称	助记符	二进制操作码
无条件跳转	JMP	00000
条件跳转	JZ	00001
减法操作	SUB	00010
加法操作	ADD	00011
立即数传送	MVI	00100
寄存器传送	MOV	00101
存数操作	STA	00110
取数操作	LDA	00111
输出操作	OUT	01000
输入操作	IN	01001

关于指令的其他部分与老师课堂所讲设计相同，例如寄存器 A 为指令第 8 位-10 位，寄存器 B 为指令第 0 位-2 位，地址 X 为第 0 位-7 位，真正地址为 reg(7)||X。

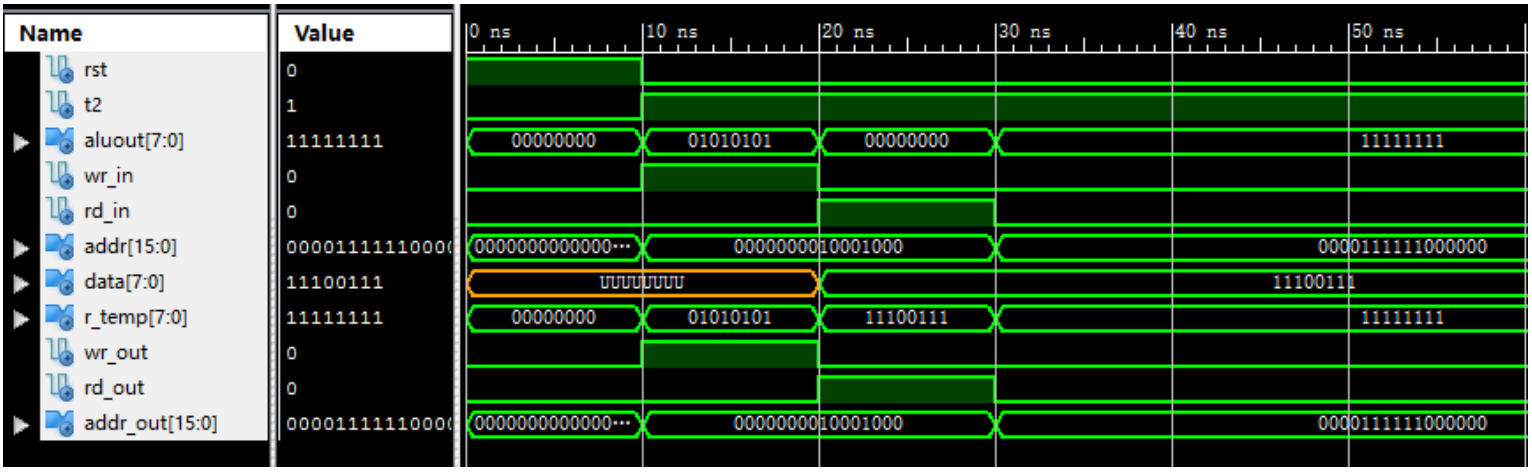
本波形说明：寄存器的初始赋值在 rst=1 时完成，他们的初始值为;

reg(0)<="00010001";
reg(1)<="00001111";
reg(2)<="01101110";
reg(3)<="00101010";
reg(4)<="01010101";
reg(5)<="00000001";
reg(6)<="11111111";
reg(7)<="00000000";
(在乐学网中的代码可以看到)

当 rst 无效时，我给定的波形测试指令码依次为(所有数据为 16 进制数):

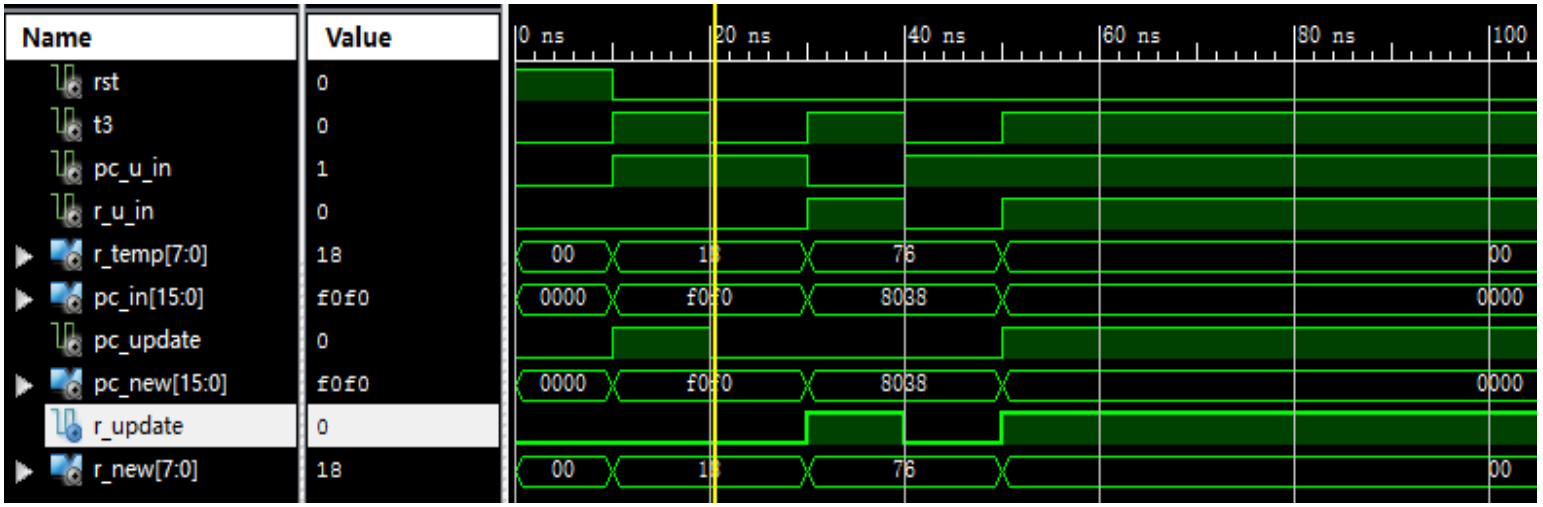
ADDreg(5),reg(6)	结果保存在 reg(5)里
SUBreg(0),reg(1)	结果保存在 reg(0)里
MVIreg(2),80	将立即数存至 reg(2)里
MOVreg(3),reg(2)	将 reg(2)的值赋给 reg(3)
STA reg(0),FF	将 reg(0)的值存至存储器中
LDA reg3),FF	将存储器 00FF 地址中的数存至 reg(0)中
OUTreg(4),01	将 reg(4)的值放入地址号为 01 的外设展示
INreg(6),01	将设备 01 外设的值输入至 reg(6)
JZreg(5),A0	若 reg(5)为 0，则 PC 值更新为 00A0
JMP C0	无条件跳转至 00C0(PC 值)

4 存储模块



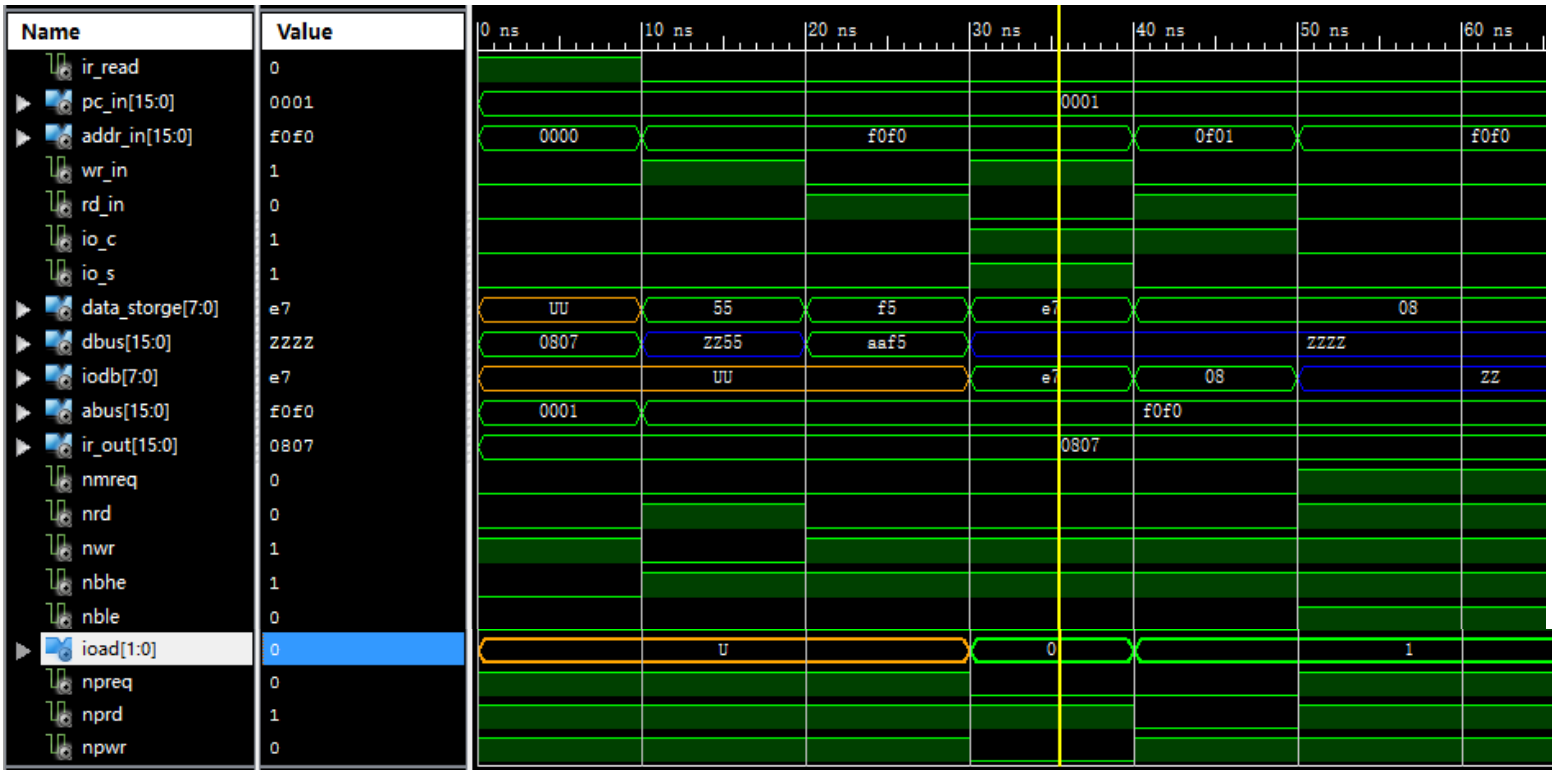
波形说明：rst 无效时，wr_out 有效时，R_temp 为即将写入存储器的值(ALUOUT),当 rd_out 有效时 R_temp 值为从存储器中取出的数(data),既不读也不写的时候，wr_out=rd_out=0,R_temp 的值为 ALUOUT 的值，传入下一模块用于回写。

5 回写模块



波形说明：在 rst 无效时，首先 PC_new 和 R_new 的值为传入的 PC_in 和 R_in 的值。但只有当 t3 有效时，相应控制信号 PC_update 和 R_update 的值才会为传入的 PC_u_in 和 R_u_in 的值，其他时刻均为 0，即只有在回写周期(t3)才会对 PC 或寄存器进行回写。

6 访存管理模块



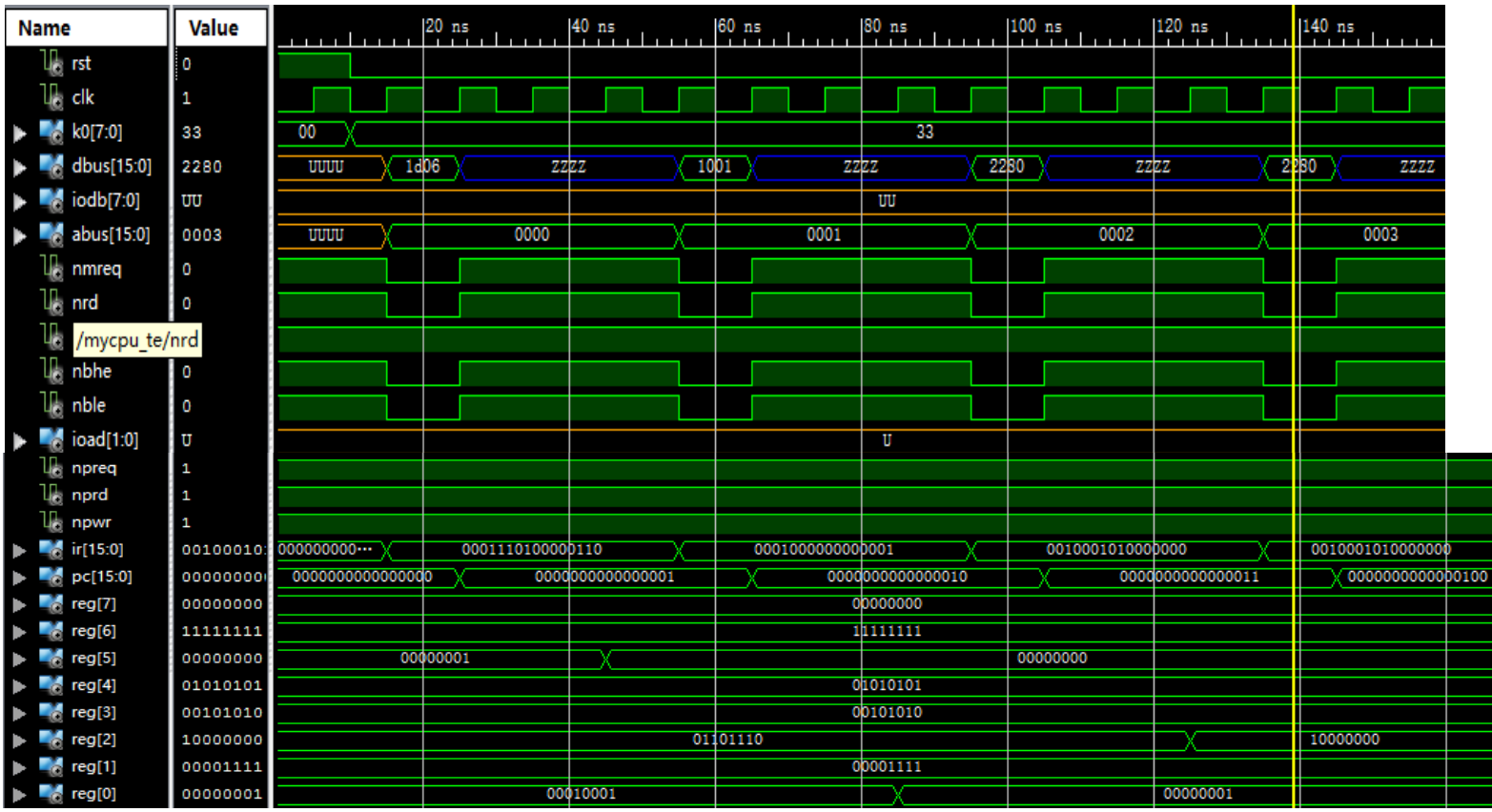
波形说明：访存控制模块根据传入的各种控制信号(wr,rd,io_s,io_c……)以及相应地址和数据，得出存储器和外设的控制信号及地址和数据。

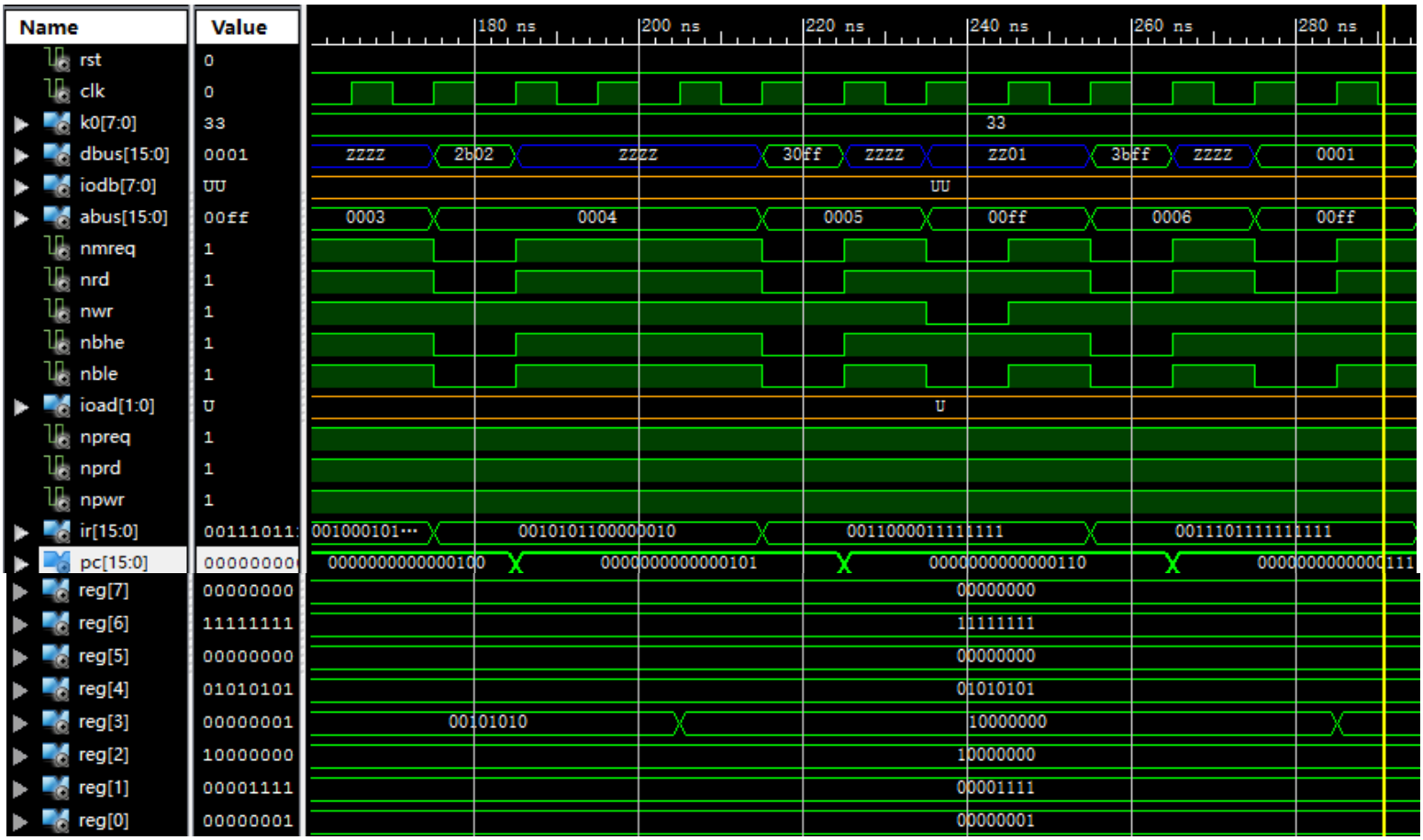
7 系统仿真波形(下一页)

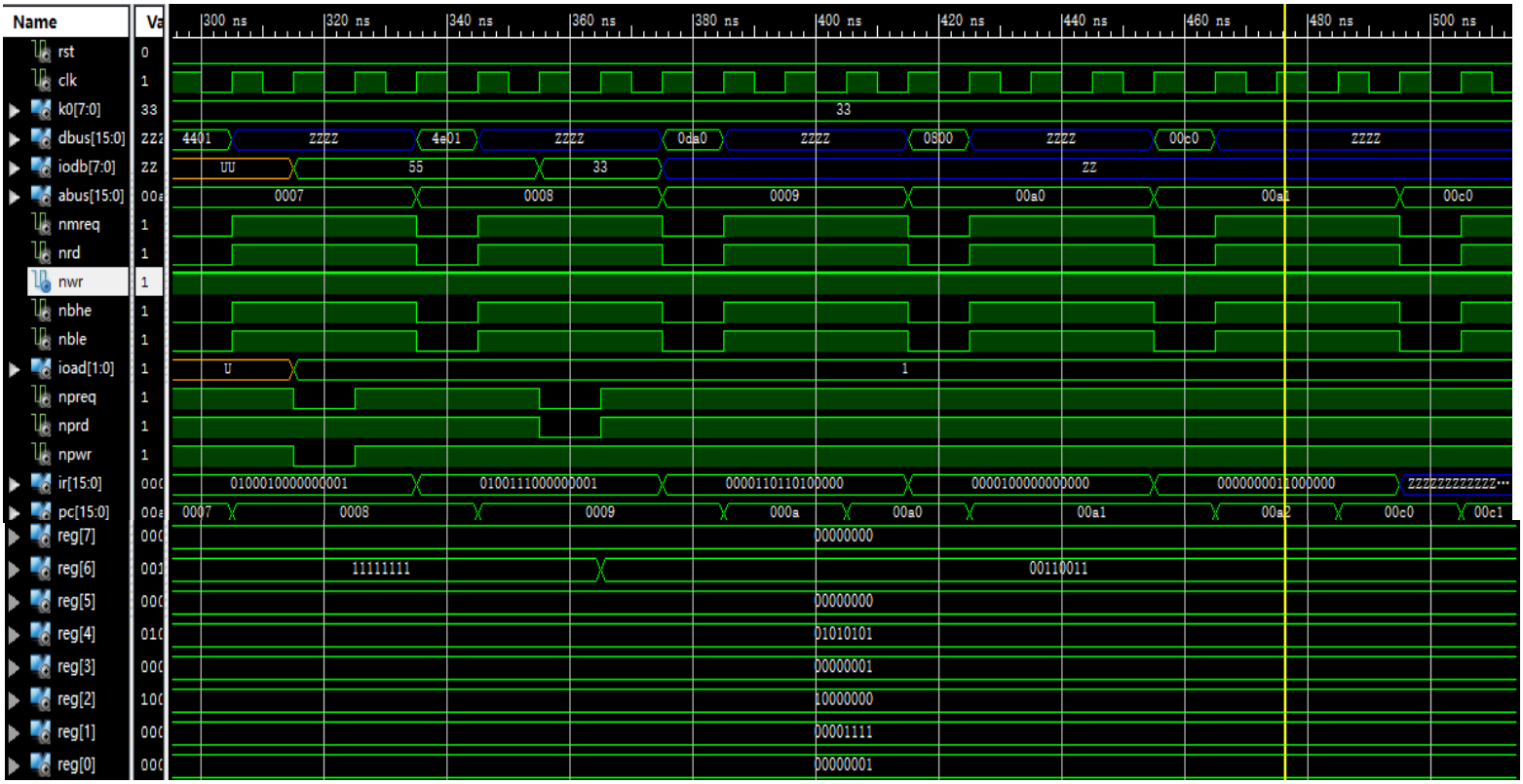
波形说明：寄存器的初值仍然是已经设计好的，在不同的指令周期，给出相应存储器或外设的不同的 DBUS 值(指令的值或数据)。本波形测试，我按照我的指令格式设计的指令为：ADDreg(5),reg(6) SUBreg(0),reg(1) MVIreg(2),80

MOVreg(3),reg(2) STA reg(0),FF LDA reg3),FF OUTreg(4),01

INreg(6),01 JZreg(5),A0 JZreg(0),00 JMP C0


Name
Value
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
rst
0
clk
1
k0[7:0]
33
dbus[15:0]
2280
iodb[7:0]
UU
abus[15:0]
0003
nmreq
0
nrd
0
/mycpu_te/nrd
nbhe
0
nble
0
ioad[1:0]
U
npreq
1
nprd
1
npwr
1
ir[15:0]
pc[15:0]
reg[7]
00000000
reg[6]
11111111
reg[5]
00000000
reg[4]
01010101
reg[3]
00101010
reg[2]
10000000
reg[1]
00001111
reg[0]
00000001


Name
Value
180 ns
200 ns
220 ns
240 ns
260 ns
280 ns
rst
0
clk
0
k0[7:0]
33
dbus[15:0]
0001
iodb[7:0]
UU
abus[15:0]
00ff
nmreq
1
nrd
1
nwr
1
nbhe
1
nble
1
ioad[1:0]
U
npreq
1
nprd
1
npwr
1
ir[15:0]
pc[15:0]
reg[7]
00000000
reg[6]
11111111
reg[5]
00000000
reg[4]
01010101
reg[3]
00000001
reg[2]
10000000
reg[1]
00001111
reg[0]
00000001



四、系统管脚定义的 UCF 文件

```
NET "DBUS(0)" LOC="p167";
NET "DBUS(1)" LOC="p165";
NET "DBUS(2)" LOC="p164";
NET "DBUS(3)" LOC="p163";
NET "DBUS(4)" LOC="p162";
NET "DBUS(5)" LOC="p161";
NET "DBUS(6)" LOC="p160";
NET "DBUS(7)" LOC="p153";

NET "DBUS(8)" LOC="p120";
NET "DBUS(9)" LOC="p122";
NET "DBUS(10)" LOC="p123'
NET "DBUS(11)" LOC="p128'
NET "DBUS(12)" LOC="p132'
NET "DBUS(13)" LOC="p133'
NET "DBUS(14)" LOC="p134'
NET "DBUS(15)" LOC="p135'

NET "ABUS(0)" LOC="p179";
NET "ABUS(1)" LOC="p178";
NET "ABUS(2)" LOC="p177";
NET "ABUS(3)" LOC="p172";
NET "ABUS(4)" LOC="p171";
NET "ABUS(5)" LOC="p151";
NET "ABUS(6)" LOC="p150";
NET "ABUS(7)" LOC="p147";

NET "ABUS(8)" LOC="p146";
NET "ABUS(9)" LOC="p113";
NET "ABUS(10)" LOC="p115";
NET "ABUS(11)" LOC="p116";
NET "ABUS(12)" LOC="p119";
NET "ABUS(13)" LOC="p140";
NET "ABUS(14)" LOC="p144";
NET "ABUS(15)" LOC="p145";

NET "nMREQ" LOC="p168";
NET "nWR" LOC="p152";
NET "nRD" LOC="p139";
NET "nBHE" LOC="p138";
NET "nBLR" LOC="p137";
NET "clk" LOC="p75";
net "rst" loc="p51";
net "k0<0>" loc="p154";
net "k0<1>" loc="p148";
net "k0<2>" loc="p142";
net "k0<3>" loc="p136";
net "k0<4>" loc="p130";
net "k0<5>" loc="p124";
net "k0<6>" loc="p118";
net "k0<7>" loc="p110";

net "k1<0>" loc="p174";
net "k1<1>" loc="p204";
net "k1<2>" loc="p194";
net "k1<3>" loc="p175";
net "k1<4>" loc="p169";
net "k1<5>" loc="p101";
net "k1<6>" loc="p97";
net "k1<7>" loc="p96";

net "k2<0>" loc="p94";
net "k2<1>" loc="p93";
net "k2<2>" loc="p91";
```

```
net "k2<3>" loc="p90";    net "s0<7>" loc="p16";      net "DBUS_out<2>" loc="p47";   net "ABUS_out<6>" loc="p206";
net "k2<4>" loc="p89";                                net "DBUS_out<3>" loc="p48";   net "ABUS_out<7>" loc="p103";
net "k2<5>" loc="p87";    net "s1<0>" loc="p18";      net "DBUS_out<4>" loc="p49";
net "k2<6>" loc="p80";    net "s1<1>" loc="p19";      net "DBUS_out<5>" loc="p50";
net "k2<7>" loc="p78";    net "s1<2>" loc="p22";      net "DBUS_out<6>" loc="p55";   net "nBLE_out" loc="p102";
                          net "s1<3>" loc="p23";      net "DBUS_out<7>" loc="p56";   net "nBHE_out" loc="p100";
net "k3<0>" loc="p72";    net "s1<4>" loc="p24";                                     net "nRD_out" loc="p99";
net "k3<1>" loc="p71";    net "s1<5>" loc="p25";      net "DBUS_out<0>" loc="p60";   net "nWR_out" loc="p98";
net "k3<2>" loc="p69";    net "s1<6>" loc="p28";      net "DBUS_out<1>" loc="p61";   net "nMREQ_out" loc="p83";
net "k3<3>" loc="p68";    net "s1<7>" loc="p29";      net "DBUS_out<2>" loc="p62";
net "k3<4>" loc="p65";                                net "DBUS_out<3>" loc="p63";   net "t0" loc="p196";
net "k3<5>" loc="p64";    net "ABUS_out<0>" loc="p31"; net "DBUS_out<4>" loc="p2";   net "t1" loc="p193";
net "k3<6>" loc="p58";    net "ABUS_out<1>" loc="p33"; net "DBUS_out<5>" loc="p108"; net "t2" loc="p192";
net "k3<7>" loc="p57";    net "ABUS_out<2>" loc="p34"; net "DBUS_out<6>" loc="p109"; net "t3" loc="p190";
                          net "ABUS_out<3>" loc="p35"; net "DBUS_out<7>" loc="p112";
net "s0<0>" loc="p4";     net "ABUS_out<4>" loc="p36";                               net "nPREQ" loc="p200";
net "s0<1>" loc="p5";     net "ABUS_out<5>" loc="p39"; net "ABUS_out<0>" loc="p126"; net "nPWR" loc="p199";
net "s0<2>" loc="p8";     net "ABUS_out<6>" loc="p40"; net "ABUS_out<1>" loc="p127"; net "nPRD" loc="p197";
net "s0<3>" loc="p9";     net "ABUS_out<7>" loc="p41"; net "ABUS_out<2>" loc="p129";
net "s0<4>" loc="p11";                                net "ABUS_out<3>" loc="p202";  NET "rst" CLOCK_DEDICATED_ROUTE = FALSE;
net "s0<5>" loc="p12";    net "DBUS_out<0>" loc="p42"; net "ABUS_out<4>" loc="p203";
net "s0<6>" loc="p15";    net "DBUS_out<1>" loc="p45"; net "ABUS_out<5>" loc="p205";
```

五、处理器功能测试程序(包括助记符和二进制代码)

助记符	十六进制代码	二进制代码
mvi R7, 0x0	4700	0100 0111 0000 0000
mvi R0, 55	4055	0100 0000 0101 0101
mov R3, R0	5300	0101 0011 0000 0000
out R3	8300	1000 0011 0000 0000
sta R3, 60	6360	0110 0011 0110 0000
jz R0, 00	1000	0001 0000 0000 0000
lda R5, 0060	7560	0111 0101 0110 0000
out R5	8500	1000 0101 0000 0000
add R5, R0	3500	0011 0101 0000 0000
out R5	8500	1000 0101 0000 0000
sub R3, R3	2303	0010 0011 0000 0011
out R3	8500	1000 0101 0000 0000
in R6 11	9603	1001 0110 0000 0011
out R6	8600	1000 0110 0000 0000
jz R3, 0010	1310	0001 0011 0001 0000
jmp 0000	0000	0000 0000 0000 0000

六、设计、调试、波形、下载过程中遇到的问题及解决方法

1 在设计取指模块的时候，IR_out 和 PC_out 一直无法更新，而且我已经把这个个赋值语句写在了 process 的后面，不存在信号延迟赋值的问题。最后我将 process 语句里面的每一个 if 条件针对每一个敏感信号的值都做了判断，才解决了这个问题。

2 在设计存储模块时，对于 inout 型信号，波形仿真时，它一直为高阻，不发生任何变化，随后自己检查自己的程序，发现在敏感信号表中没有添加该信号，同时，当 inout 型信号作为输入时，其输出应该设为高阻。

3 整个系统连到一起之后，无法运行，错误信息为：“process will terminate”，经过上网查询，得知需要重新建立工程文件，再引入.vhd 文件。按照其说法操作之后，我成功解决了这个问题。

4 系统波形调试的时候，整个波形都是错的，究其本源之后发现，每一条指令根本没有正确从取指模块发到运算模块，发出的指令在 t1 周期都会变为“0000000000000000”，经检查，发现是取指模块中的 IR 寄存器的值只在 t0 周期有效，因为它的赋值语句为 IR<= IR_in，而访存管理模块中的 IR_in <= DBUS，分析知，当 DBUS 赋值为高阻时，IR_in 就会改变，从而 IR 也变，便不会正确的传如运算模块。我的解决办法是在访存管理模块增设了一个 IR 暂存器，解决了这个问题。

5 编写 ucf 文件的时候，无法生成.bit 文件，通过上网查询知，原因是 CLOCK_DEDICATED_ROUTE 的约束问题。添加了 NET "rst" CLOCK_DEDICATED_ROUTE = FALSE;这句话之后，问题得到解决。

6 在扳子上操作时，显示有错误，经检查是 ucf 文件管脚定义写窜行了。

七、实验体会

通过本次实验，首先，掌握了 Xilinx ISE 集成开发环境和 ModelSim 仿真工具的使用方法；掌握了 FPGA 编程方法及硬件调试手段，深刻理解了处理器结构和计算机系统的整体工作原理。然后我对 VHDL 语言的使用能力和认识有了难以想象的提升。同时对自顶向下的硬件逻辑设计方法有了更加深入的理解。对 CPU 也有了初步的认识（指令在 CPU 中如何执行）。在编写代码和扳子调试的过程中，错误层出不穷，Dbug 的过程十分痛苦，但在网络各种资料和老师的帮助下，成功解决了各种问题。当自己的 CPU 代码完成后，看到自己的波形图和成功的扳子显示，感觉非常开心。

本次实验也使我对 CPU 的构成和工作方式产生了浓厚的兴趣，提升了我的学习热情，使这个小学期变得有趣而丰富。