# CPU实验设计报告

**学号：1130310710**

**班号： 1303107**

**姓名： 史双玮**

## 一：实验目的

1. 掌握Xilinx ISE集成开发环境和ModelSim仿真工具的使用方法。
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

1、Xilinx ISE集成开发环境及ModelSim仿真工具。

2、SD2100数字逻辑实验设计台。

## 三：设计思想

### CPU接口信号定义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 位数 | 方向 | 来源/去向 | 意义 |
| rst | 1 | In | 处理器板 | 高电平复位 |
| clk | 1 | In | 处理器板 | 时钟信号 |
| abus | 16 | out | 主存储器 | 地址总线 |
| dbus | 16 | inout | 主存储器 | 数据总线 |
| nrd | 1 | out | 主存储器 | 存储器读信号 |
| nwr | 1 | out | 主存储器 | 存储器写信号 |
| nble | 1 | out | 主存储器 | 低字节访问信号 |
| nbhe | 1 | out | 主存储器 | 高字节访问信号 |
| cy | 1 | out | FPGA | 进位灯 |
| nmreq | 1 | out | 主存储器 | 片选信号 |
| irout | 1 | out | FPGA | 指令输出 |
| dbusout | 16 | out | FPGA | 数据总线输出 |
| tout | 4 | out | FPGA | 节拍输出 |
| abusout | 16 | out | FPGA | 地址输出 |
| nmreqout | 1 | out | FPGA | 片选信号输出 |
| nrdout | 1 | out | FPGA | 读信号输出 |
| nwrout | 1 | out | FPGA | 写信号输出 |
| nbheout | 1 | out | FPGA | 高字节访问信号输出 |
| nbleout | 1 | out | FPGA | 低字节访问信号输出 |

### CPU设计方案

#### 指令格式设计

A.加法指令 ADD Ri,Rj

|  |  |  |  |
| --- | --- | --- | --- |
| 指令码 | 寄存器地址Ri | 填充 | 寄存器地址Rj |
| 00000 | XXX | 00000 | XXX |

B.减法指令 SUB Ri,Rj

|  |  |  |  |
| --- | --- | --- | --- |
| 指令码 | 寄存器地址Ri | 填充 | 寄存器地址Rj |
| 00001 | XXX | 00000 | XXX |

C.寄存器传送指令 MOV Ri,Rj

|  |  |  |  |
| --- | --- | --- | --- |
| 指令码 | 寄存器地址Ri | 填充 | 寄存器地址Rj |
| 00010 | XXX | 00000 | XXX |

D.立即数传送指令 MVI Ri,X

|  |  |  |
| --- | --- | --- |
| 指令码 | 寄存器地址Ri | 立即数 |
| 00011 | XXX | XXXXXXXX |

E.存数指令 STA Ri,X

|  |  |  |
| --- | --- | --- |
| 指令码 | 寄存器地址Ri | 立即数 |
| 00100 | XXX | XXXXXXXX |

F.取数指令 LDA Ri,X

|  |  |  |
| --- | --- | --- |
| 指令码 | 寄存器地址Ri | 立即数 |
| 00101 | XXX | XXXXXXXX |

G.条件转移 JZ Ri,X

|  |  |  |
| --- | --- | --- |
| 指令码 | 寄存器地址Ri | 立即数 |
| 00110 | XXX | XXXXXXXX |

H.无条件转移指令 JMP X

|  |  |  |
| --- | --- | --- |
| 指令码 | 填充 | 立即数 |
| 00111 | 000 | XXXXXXXX |

I.清进位位指令 CLC

|  |  |
| --- | --- |
| 指令码 | 填充 |
| 01000 | 00000000000 |

J.置进位位指令 STC

|  |  |
| --- | --- |
| 指令码 | 填充 |
| 01001 | 00000000000 |

#### 微操作定义

**T0**

pc -> mar

1 -> apc

mdr -> ir

**T1**

1. ADD Ri,Rj (Ri)+(Rj)+cy -> rtemp , Ri -> raddr , pc+1 -> pctemp , newcy->cout
2. SUB Ri,Rj (Ri)-(Rj)-cy -> rtemp , Ri -> raddr , pc+1 -> pctemp , newcy->cout
3. MOV Ri,Rj (Rj) -> rtemp , Ri -> raddr , pc+1 -> pctemp
4. MVI Ri,X X -> rtemp , Rj -> raddr , pc+1 -> pctemp
5. STA Ri,X (Rj) -> rtemp , R7&X -> addr , pc+1 -> pctemp
6. LDA Ri,X Ri-> raddr , R7&X -> addr , pc+1 -> pctemp
7. JZ Ri,X R7+X -> pctemp
8. JMP X R7&X -> pctemp
9. CLC 0 -> cout
10. STC 1 -> cout

addr->mar

T2

1. STA Ri,X 1 -> wr , data->mdr
2. LDA Ri,X 1 -> rd , mdr->databack
3. Others 0-> wr, 0-> rd , data -> rout

T3

0/1 -> rupdate

rout -> rnew

0/1 -> pcupdate

pctemp -> pcupdate

0/1 -> cyupdate

cout->cin

#### 节拍划分

T0 取指周期

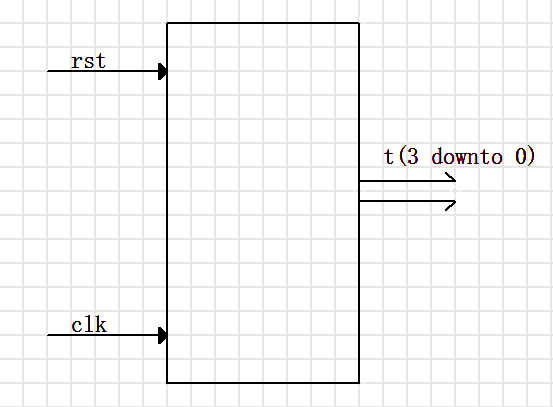
T1 运算周期

T2 访存周期

T3 回写周期

#### 各模块设计

##### 时钟模块



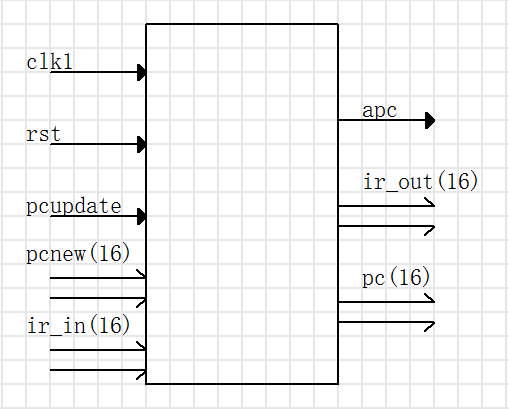
**模块功能：**

四节拍发生器，以输入时钟信号上升沿为触发条件，四个节拍循环。当rst=‘1’时节拍复位至第四节拍。

**接口说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **意义** |
| **rst** | **1** | **in** | **处理器板** | **高电平复位信号** |
| **clk** | **1** | **in** | **处理器板** | **时钟信号** |
| **t** | **4** | **out** | **各模块节拍输入** | **节拍信号** |

##### 取指模块



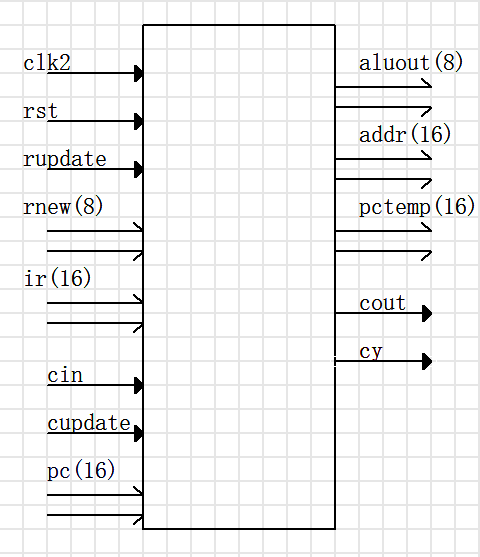
**模块功能：**

**主要负责取指操作，在第一个节拍将pc送到访存控制单元同时送访存请求信号apc=’1’,取得指令同时将指令送往运算、存储、回写模块。并将当前pc送给运算模块。**

**接口说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **意义** |
| **rst** | **1** | **in** | **处理器板** | **高电平复位信号** |
| **clk1** | **1** | **in** | **时钟模块** | **节拍信号** |
| **pcupdate** | **1** | **in** | **回写模块** | **pc更新信号** |
| **pcnew** | **16** | **in** | **回写模块** | **新的pc** |
| **ir\_in** | **16** | **in** | **访存控制** | **指令** |
| **apc** | **1** | **out** | **访存控制** | **取指允许信号** |
| **ir\_out** | **16** | **out** | **运算、储存、回写模块** | **指令** |
| **pc** | **16** | **out** | **运算模块** | **指令地址** |

##### 运算模块



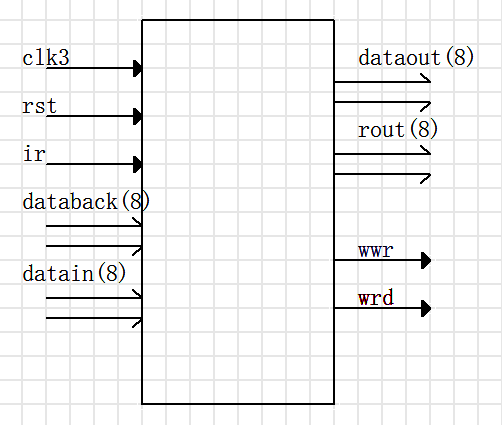
**功能说明：**

**复位信号为1 时cy清零。当回写信号为1时回写数据回写到对应地址的寄存器中。在第二节拍时完成指令译码工作，并根据译码结果对相关的addr,aluout,raddr进行置数操作，以及根据运算结果改变cy标志位。**

**接口说明**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **意义** |
| **clk2** | **1** | **in** | **时钟模块** | **节拍信号** |
| **rst** | **1** | **in** | **处理器板** | **高电平复位信号** |
| **rupdate** | **1** | **in** | **回写模块** | **寄存器更新信号** |
| **rnew** | **8** | **in** | **回写模块** | **寄存器更新值** |
| **ir** | **16** | **in** | **取指模块** | **指令** |
| **cin** | **1** | **in** | **回写模块** | **cy更新值** |
| **cupdate** | **1** | **in** | **回写模块** | **cy更新信号** |
| **pc** | **16** | **in** | **取指模块** | **指令地址** |
| **aluout** | **8** | **out** | **存储管理模块** | **运算结果** |
| **cout** | **1** | **out** | **回写模块** | **新产生的进位位** |
| **cy** | **1** | **out** | **FPGA** | **进位标志灯** |
| **addr** | **16** | **out** | **访存模块** | **主存读写地址** |
| **pctemp** | **16** | **out** | **回写模块** | **下条指令地址** |

##### 存储模块



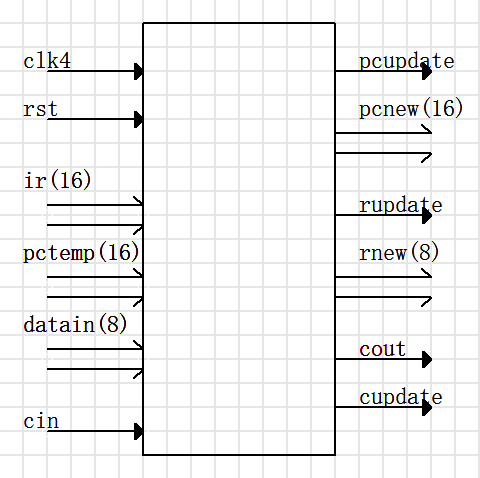
**功能说明：**

**主要完成从访存控制模块进行不同指令的存数或取数的请求。**

**接口说明:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **意义** |
| **clk3** | **1** | **in** | **时钟模块** | **节拍信号** |
| **rst** | **1** | **in** | **处理器板** | **高电平复位信号** |
| **ir** | **16** | **in** | **取指模块** | **指令** |
| **databack** | **8** | **in** | **访存模块** | **主存返回的数据** |
| **datain** | **8** | **in** | **运算模块** | **运算结果** |
| **dataout** | **8** | **out** | **访存模块** | **写入主存的数据** |
| **rout** | **8** | **out** | **回写模块** | **需要回写的数据** |
| **wwr** | **1** | **out** | **访存控制** | **写信号** |
| **wrd** | **1** | **out** | **访存控制** | **读信号** |

##### 回写模块



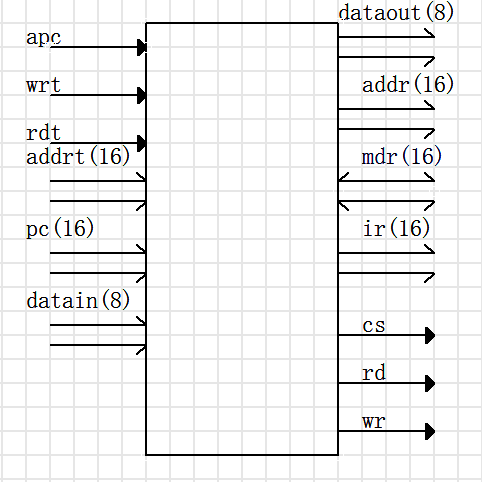
**功能说明：**

**主要用于对pc和寄存器以及cy的回写。**

**接口说明：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **意义** |
| **clk4** | **1** | **in** | **时钟模块** | **节拍信号** |
| **rst** | **1** | **in** | **处理器板** | **高电平复位信号** |
| **ir** | **16** | **in** | **取指模块** | **指令** |
| **pctemp** | **16** | **in** | **运算模块** | **下条指令的地址** |
| **datain** | **8** | **in** | **存储模块** | **需要回写数据** |
| **cin** | **1** | **in** | **运算模块** | **新进位位** |
| **pcupdate** | **1** | **out** | **取指模块** | **pc更新信号** |
| **pcnew** | **16** | **out** | **取指模块** | **pc更新值** |
| **rupdate** | **1** | **out** | **运算模块** | **寄存器更新信号** |
| **rnew** | **8** | **out** | **运算模块** | **寄存器更新值** |
| **cout** | **1** | **out** | **运算模块** | **cy更新值** |
| **cupdate** | **1** | **out** | **运算模块** | **cy更新信号** |

##### 访存模块



**功能说明**：

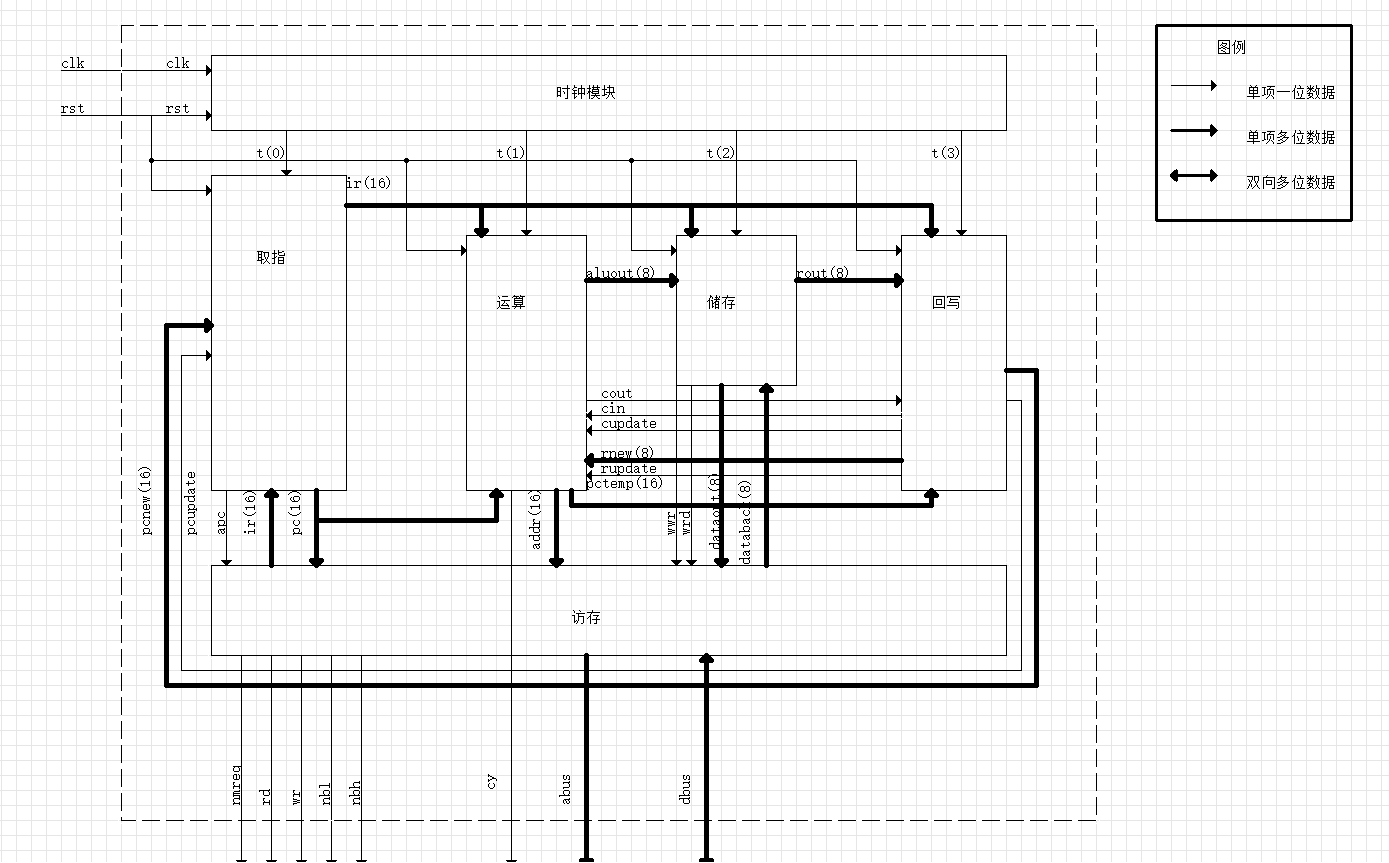
当取指模块发出取指请求时，根据pc取出指令送给取指模块。根据存储管理模块给的读写请求对存储器进行读写操作。

**接口说明:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名** | **位数** | **方向** | **来源/去向** | **意义** |
| **apc** | **1** | **in** | **取指模块** | **读指令信号** |
| **wrt** | **1** | **in** | **存储管理** | **写信号** |
| **rdt** | **1** | **in** | **存储管理** | **读信号** |
| **addrt** | **16** | **in** | **运算管理** | **数据地址** |
| **pc** | **16** | **in** | **取指模块** | **指令地址** |
| **datain** | **8** | **in** | **存储管理** | **需要写的数据** |
| **dataout** | **8** | **out** | **存储管理** | **需要读的数据** |
| **addr** | **16** | **out** | **存储器** | **地址总线** |
| **mdr** | **16** | **inout** | **存储器** | **数据总线** |
| **ir** | **16** | **out** | **取指模块** | **指令** |
| **cs** | **1** | **out** | **存储器** | **片选信号** |
| **rd** | **1** | **out** | **存储器** | **读信号** |
| **wr** | **1** | **out** | **存储器** | **写信号** |

#### CPU整体设计框图

见下页

1. 

## 四：实验设计及测试

### 各模块设计及测试（测试波形统一在后面标注）

#### 时钟模块

代码：

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity clock is

Port ( clk : in STD\_LOGIC;

rst : in STD\_LOGIC;

t : out STD\_LOGIC\_VECTOR (3 downto 0));

end clock;

architecture Behavioral of clock is

signal temp: STD\_LOGIC\_VECTOR(3 DOWNTO 0):="1000";

begin

process (clk,rst)

begin

if rst='1' then

temp<="0001";

elsif clk'event and clk='1' then

temp(3 downto 1)<=temp(2 downto 0);

temp(0)<=temp(3);

end if;

end process;

t<=temp;

end Behavioral;

波形图见4.1

#### 取指模块

代码：

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity pcm is

Port ( clk1 : in STD\_LOGIC;

rst : in STD\_LOGIC;

apc : out STD\_LOGIC;

pcupdate : in STD\_LOGIC;

pcnew : in STD\_LOGIC\_VECTOR (15 downto 0);

ir\_in : in STD\_LOGIC\_VECTOR (15 downto 0);

ir\_out : out STD\_LOGIC\_VECTOR (15 downto 0);

pc : out STD\_LOGIC\_VECTOR (15 downto 0));

end pcm;

architecture Behavioral of pcm is

signal apctemp:std\_logic;

begin

process(rst,pcupdate,pcnew)

begin

if rst='1' then

pc<="0000000000000000";

elsif pcupdate='1' then

pc<=pcnew;

end if;

end process;

process(clk1,rst,pcupdate,pcnew)

begin

if rst='1' then

apctemp<='0';

elsif clk1='1' then

apctemp<='1';

elsif clk1='0' then

apctemp<='0';

end if;

end process;

apc<=apctemp;

ir\_out<=ir\_in;

end Behavioral;

波形图见4.2

#### 运算模块

代码：

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity ALU is

Port ( clk2 : in STD\_LOGIC;

rupdate : in STD\_LOGIC;

rnew : in STD\_LOGIC\_VECTOR (7 downto 0);

ir : in STD\_LOGIC\_VECTOR (15 downto 0);

pc : in STD\_LOGIC\_VECTOR (15 downto 0);

aluout : out STD\_LOGIC\_VECTOR (7 downto 0);

pctemp : out STD\_LOGIC\_VECTOR (15 downto 0);

rst : in STD\_LOGIC;

cin : in STD\_LOGIC:='0';

cy : out STD\_LOGIC;

cupdate : in STD\_LOGIC;

cout: out STD\_LOGIC;

addr : out STD\_LOGIC\_VECTOR (15 downto 0));

end ALU;

architecture Behavioral of ALU is

type data8 IS array(0 to 7) of std\_logic\_vector(7 downto 0);

signal reg:data8; --定义寄存器

begin

process(rupdate,rnew,rst)

begin

if(rupdate='1')then

reg(conv\_integer( ir( 10 downto 8 )))<=rnew;

end if;

end process;

process(ir,clk2,rst,rnew,rupdate,reg,pc,cupdate,cin)

--variable cyy:STD\_LOGIC;

variable ax:STD\_LOGIC\_VECTOR (8 downto 0):="000000000";

variable bx:STD\_LOGIC\_VECTOR (8 downto 0):="000000000";

variable r7:STD\_LOGIC\_VECTOR (7 downto 0);

variable t:STD\_LOGIC\_VECTOR (15 downto 0);

variable k:STD\_LOGIC\_VECTOR (8 downto 0);

variable aluouttemp:STD\_LOGIC\_VECTOR (8 downto 0):="000000000";

begin

ax(7 downto 0):= reg(conv\_integer( ir( 10 downto 8 )));

bx(7 downto 0):= reg(conv\_integer( ir( 2 downto 0 )));

r7:= reg(conv\_integer("111"));

ax(8):='0';

bx(8):='0';

if(rst = '1')then

cy<='0';

cout<='0';

elsif (clk2='1')then

case ir(15 downto 11) is

when"00000" =>

aluouttemp:=ax+bx+cin;

pctemp<=pc+1;

aluout<=aluouttemp(7 downto 0);

cout<=aluouttemp(8);

when"00001" =>

aluouttemp:=ax-bx-cin;

pctemp<=pc+1;

aluout<=aluouttemp(7 downto 0);

cout<=aluouttemp(8);

when"00010"=>

aluouttemp:=bx;

aluout<=aluouttemp(7 downto 0);

pctemp<=pc+1;

when"00011"=>

aluouttemp:='0'&ir(7 downto 0);

pctemp<=pc+1;

aluout<=aluouttemp(7 downto 0);

when"00100"=>

aluouttemp:=ax;

addr<=r7&ir(7 downto 0);

pctemp<=pc+1;

aluout<=aluouttemp(7 downto 0);

when"00101"=>

t:=r7&ir(7 downto 0);

addr<=t;

pctemp<=pc+1;

when"00110"=>

if(ax="000000000" and ir(7) ='0')then

aluouttemp:=ax;

aluout<=aluouttemp(7 downto 0);

t:="00000000"&ir(7 downto 0);

pctemp <= pc+t+1;

elsif (ax="000000000" and ir(7) ='1')then

aluouttemp:=ax;

aluout<=aluouttemp(7 downto 0);

t:="11111111"&ir(7 downto 0);

pctemp <= pc+t+1;

else

aluouttemp:=ax;

aluout<=aluouttemp(7 downto 0);

pctemp <= pc+1;

end if;

when"00111"=>

pctemp<=r7&ir(7 downto 0);

when"01000"=>

cout<='0';

pctemp<=pc+1;

when"01001"=>

cout<='1';

pctemp<=pc+1;

when others=> NULL;

end case;

elsif (rst='0' and cupdate='1')then

cy<=cin;

end if;

end process;

end Behavioral；

波形见图4.3 4.4

#### 存储模块

代码：

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mcm is

Port ( clk3 : in STD\_LOGIC;

rst : in STD\_LOGIC;

ir : in STD\_LOGIC\_VECTOR (15 downto 0);

datain : in STD\_LOGIC\_VECTOR (7 downto 0);

dataout : out STD\_LOGIC\_VECTOR (7 downto 0);

databack : in STD\_LOGIC\_VECTOR (7 downto 0);

rout : out STD\_LOGIC\_VECTOR (7 downto 0);

wrd : out STD\_LOGIC;

wwr : out STD\_LOGIC);

end mcm;

architecture Behavioral of mcm is

begin

process(rst,clk3,ir,databack)

begin

if (rst='1')then

wrd<='0';

wwr<='0';

elsif(clk3='1') then

if(ir(15 downto 11)="00000")then

wrd<='0';

wwr<='0';

rout<=datain;

elsif(ir(15 downto 11)="00001")then

wrd<='0';

wwr<='0';

rout<=datain;

elsif(ir(15 downto 11)="00010")then

wrd<='0';

wwr<='0';

rout<=datain;

elsif(ir(15 downto 11)="00011")then

wrd<='0';

wwr<='0';

rout<=datain;

elsif(ir(15 downto 11)="00100")then

wrd<='0';

wwr<='1';

dataout<=datain;

elsif(ir(15 downto 11)="00101")then

wrd<='1';

wwr<='0';

rout <= databack;

elsif(ir(15 downto 11)="00110" )then

wrd<='0';

wwr<='0';

rout<=datain;

elsif(ir(15 downto 11)="00111")then

wrd<='0';

wwr<='0';

rout<=datain;

elsif(ir(15 downto 11)="01000")then

wrd<='0';

wwr<='0';

rout<=datain;

elsif(ir(15 downto 11)="01001")then

wrd<='0';

wwr<='0';

rout<=datain;

else

wrd<='0';

wwr<='0';

rout<=datain;

end if;

else

wrd<='0';

wwr<='0';

end if;

end process;

end Behavioral;

波形见图4.8

#### 回写模块

代码：

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity rewrite is

Port ( datain : in STD\_LOGIC\_VECTOR (7 downto 0);

clk4 : in STD\_LOGIC;

ir : in STD\_LOGIC\_VECTOR (15 downto 0);

pcupdate : out STD\_LOGIC;

pcnew : out STD\_LOGIC\_VECTOR (15 downto 0);

pctemp : in STD\_LOGIC\_VECTOR (15 downto 0);

rnew : out STD\_LOGIC\_VECTOR (7 downto 0);

rupdate : out STD\_LOGIC;

cin : in STD\_LOGIC;

cout : out STD\_LOGIC;

cupdate : out STD\_LOGIC;

rst : in STD\_LOGIC);

end rewrite;

architecture Behavioral of rewrite is

begin

process(rst,clk4)

begin

if(rst='1')then

pcupdate<='1';

rupdate<='0';

pcnew<="0000000000000000";

rnew<="00000000";

elsif (clk4='1')then

if(ir(15 downto 11)="00000" or ir(15 downto 11)="00001" or ir(15 downto 11)="00010" or ir(15 downto 11)="00011" or ir(15 downto 11)="00101")then

rnew<=datain;

rupdate<='1';

pcnew<=pctemp;

pcupdate<='1';

elsif (ir(15 downto 11)="00100" or ir(15 downto 11)="01000" or ir(15 downto 11)="01001" or ir(15 downto 11)="00110" or ir(15 downto 11)="00111") then

rupdate<='0';

pcupdate<='1';

pcnew<=pctemp;

end if;

else

pcupdate<='0';

rupdate<='0';

end if;

end process;

process(cin,clk4)

begin

if (clk4='1' and(ir(15 downto 11)="00000" or ir(15 downto 11)="00001" or ir(15 downto 11)="01000" or ir(15 downto 11)="01001" ) )then

cout<=cin;

cupdate<='1';

else

cupdate<='0';

end if;

end process;

end Behavioral;

波形见图4.5 4.6

#### 访存模块

代码：

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity mcenter is

Port ( pc : in STD\_LOGIC\_VECTOR (15 downto 0);

apc : in STD\_LOGIC;

wrt : in STD\_LOGIC;

rdt : in STD\_LOGIC;

datain : in STD\_LOGIC\_VECTOR (7 downto 0);

addrt : in STD\_LOGIC\_VECTOR (15 downto 0);

addr : out STD\_LOGIC\_VECTOR (15 downto 0);

ir : out STD\_LOGIC\_VECTOR (15 downto 0);

cs : out STD\_LOGIC;

dataout : out STD\_LOGIC\_VECTOR (7 downto 0);

mdr : inout STD\_LOGIC\_VECTOR (15 downto 0);

rd : out STD\_LOGIC;

wr : out STD\_LOGIC);

end mcenter;

architecture Behavioral of mcenter is

signal mdrtemp:STD\_LOGIC\_VECTOR (15 downto 0);

signal irtemp:STD\_LOGIC\_VECTOR (15 downto 0);

begin

process(wrt,rdt,apc)

begin

if wrt='1' and apc='0' and rdt='0' then

wr<='0';

cs<='0';

rd<='1';

elsif wrt='0' and apc='1' and rdt='0' then

wr<='1';

cs<='0';

rd<='0';

elsif wrt='0' and apc='0' and rdt='1' then

wr<='1';

cs<='0';

rd<='0';

else

wr<='1';

cs<='1';

rd<='1';

end if;

end process;

process(wrt,rdt,apc,mdrtemp,irtemp,mdr)

begin

if wrt='0' and apc='1' and rdt='0' then

irtemp<=mdrtemp;

mdr<="ZZZZZZZZZZZZZZZZ";

elsif wrt='0' and apc='0' and rdt='1' then

dataout<=mdrtemp(7 downto 0);

mdr<="ZZZZZZZZZZZZZZZZ";

elsif wrt='1' and apc='0' and rdt='0' then

mdr(7 downto 0)<=datain;

mdr(15 downto 8)<=datain;

else

mdr<="ZZZZZZZZZZZZZZZZ";

end if;

end process;

process(wrt,rdt,apc,addrt,pc,irtemp,mdr)

begin

if (rdt='1' or wrt='1') and apc='0' then

addr<=addrt;

elsif apc='1' then

addr<=pc;

else

addr<="ZZZZZZZZZZZZZZZZ";

end if;

end process;

process(wrt,rdt,apc,mdr,irtemp)

begin

if rdt='1' or apc='1' then

mdrtemp<=mdr;

end if;

end process;

ir<=irtemp;

end Behavioral;

波形见图4.7

波形仿真：

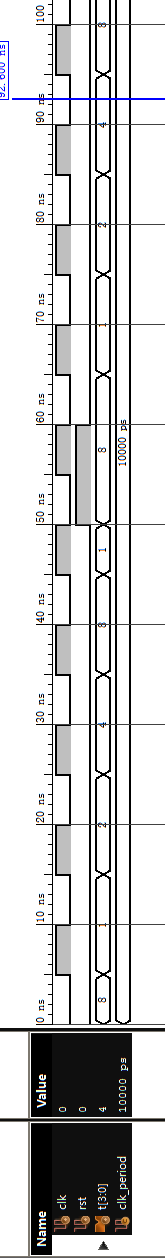
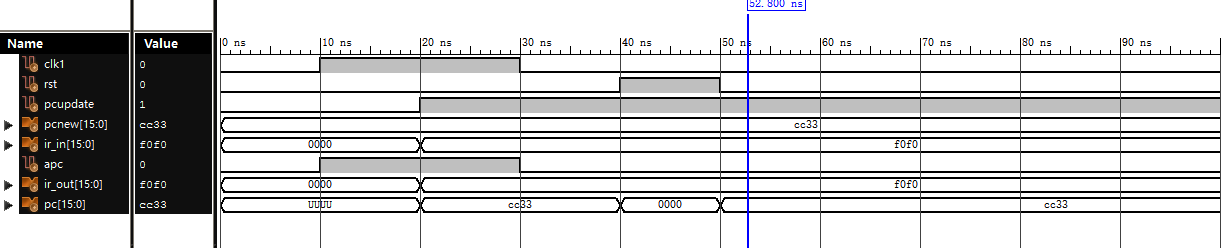


图4.2（取指模块）

图4.1（时钟模块）



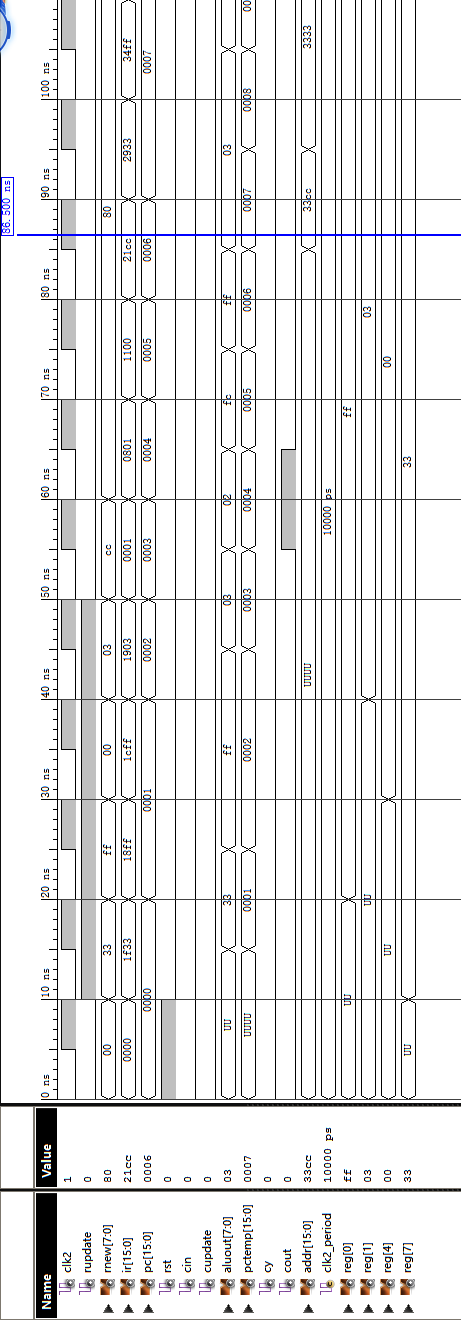
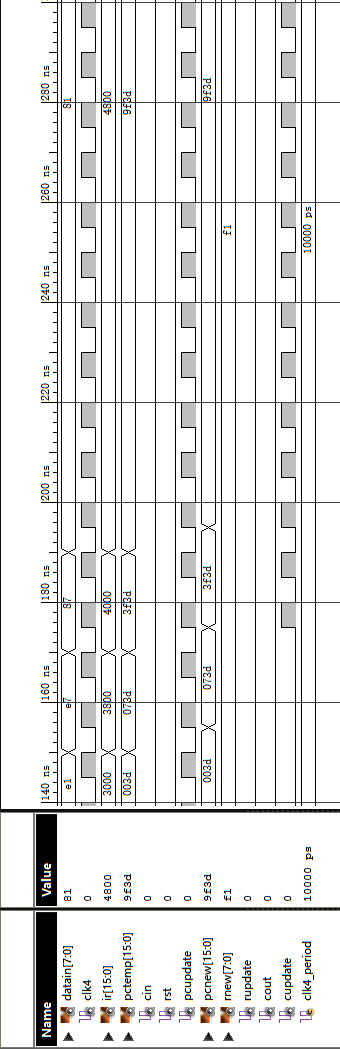


图4.4（运算模块）

图4.3（运算模块）



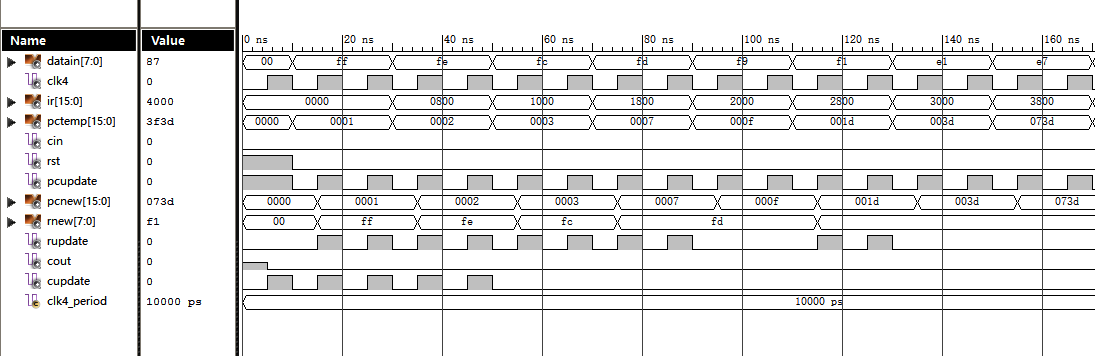


图4.6（回写模块）

图4.5（回写模块）

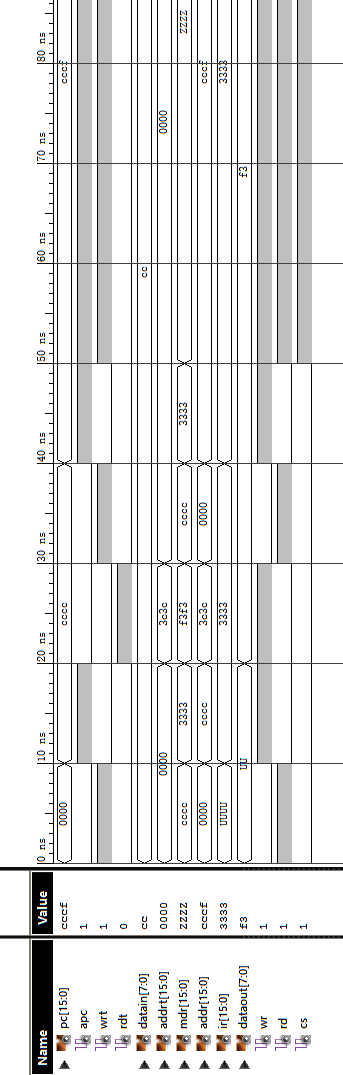
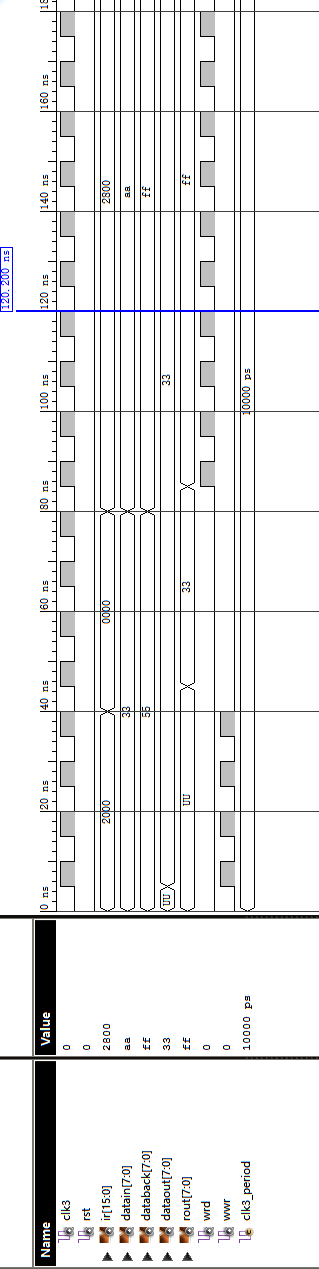


图4.7（访存控制）

图4.8（存储管理）

### 整体的设计及测试

代码：

**entity CPUM is**

**Port ( rst : in STD\_LOGIC;**

**clk : in STD\_LOGIC;**

**abus : out STD\_LOGIC\_VECTOR (15 downto 0);**

**dbus : inout STD\_LOGIC\_VECTOR (15 downto 0);**

**nrd : out STD\_LOGIC;**

**nwr : out STD\_LOGIC;**

**nble : out STD\_LOGIC;**

**nbhe : out STD\_LOGIC;**

**cy: out STD\_LOGIC;**

**z: out STD\_LOGIC;**

**abusout : out STD\_LOGIC\_VECTOR (15 downto 0);**

**irout : out STD\_LOGIC\_VECTOR (15 downto 0);**

**dbusout : out STD\_LOGIC\_VECTOR (15 downto 0);**

**tout : out STD\_LOGIC\_VECTOR (3 downto 0);**

**nmreq : out STD\_LOGIC;**

**nmreqout : out STD\_LOGIC;**

**nrdout : out STD\_LOGIC;**

**nwrout : out STD\_LOGIC;**

**nbheout : out STD\_LOGIC;**

**nbleout : out STD\_LOGIC);**

**end CPUM;**

**architecture Behavioral of CPUM is**

**component ALU is**

**Port ( clk2 : in STD\_LOGIC;**

**rupdate : in STD\_LOGIC;**

**rnew : in STD\_LOGIC\_VECTOR (7 downto 0);**

**ir : in STD\_LOGIC\_VECTOR (15 downto 0);**

**pc : in STD\_LOGIC\_VECTOR (15 downto 0);**

**aluout : out STD\_LOGIC\_VECTOR (7 downto 0);**

**pctemp : out STD\_LOGIC\_VECTOR (15 downto 0);**

**rst : in STD\_LOGIC;**

**cin : in STD\_LOGIC;**

**cy : out STD\_LOGIC;**

**cupdate : in STD\_LOGIC;**

**cout: out STD\_LOGIC;**

**addr : out STD\_LOGIC\_VECTOR (15 downto 0));**

**end component;**

**component clock is**

**Port ( clk : in STD\_LOGIC;**

**rst : in STD\_LOGIC;**

**t : out STD\_LOGIC\_VECTOR (3 downto 0));**

**end component;**

**component mcenter is**

**Port ( pc : in STD\_LOGIC\_VECTOR (15 downto 0);**

**apc : in STD\_LOGIC;**

**wrt : in STD\_LOGIC;**

**rdt : in STD\_LOGIC;**

**datain : in STD\_LOGIC\_VECTOR (7 downto 0);**

**addrt : in STD\_LOGIC\_VECTOR (15 downto 0);**

**addr : out STD\_LOGIC\_VECTOR (15 downto 0);**

**ir : out STD\_LOGIC\_VECTOR (15 downto 0);**

**dataout : out STD\_LOGIC\_VECTOR (7 downto 0);**

**mdr : inout STD\_LOGIC\_VECTOR (15 downto 0);**

**rd : out STD\_LOGIC;**

**cs : out STD\_LOGIC;**

**wr : out STD\_LOGIC);**

**end component;**

**component mcm is**

**Port ( clk3 : in STD\_LOGIC;**

**rst : in STD\_LOGIC;**

**ir : in STD\_LOGIC\_VECTOR (15 downto 0);**

**datain : in STD\_LOGIC\_VECTOR (7 downto 0);**

**databack : in STD\_LOGIC\_VECTOR (7 downto 0);**

**dataout : out STD\_LOGIC\_VECTOR (7 downto 0);**

**rout : out STD\_LOGIC\_VECTOR (7 downto 0);**

**wrd : out STD\_LOGIC;**

**wwr : out STD\_LOGIC);**

**end component;**

**component pcm is**

**Port ( clk1 : in STD\_LOGIC;**

**rst : in STD\_LOGIC;**

**apc : out STD\_LOGIC;**

**pcupdate : in STD\_LOGIC;**

**pcnew : in STD\_LOGIC\_VECTOR (15 downto 0);**

**ir\_in : in STD\_LOGIC\_VECTOR (15 downto 0);**

**ir\_out : out STD\_LOGIC\_VECTOR (15 downto 0);**

**pc : out STD\_LOGIC\_VECTOR (15 downto 0));**

**end component;**

**component rewrite is**

**Port ( datain : in STD\_LOGIC\_VECTOR (7 downto 0);**

**clk4 : in STD\_LOGIC;**

**ir : in STD\_LOGIC\_VECTOR (15 downto 0);**

**pcupdate : out STD\_LOGIC;**

**cin : in STD\_LOGIC;**

**cout : out STD\_LOGIC;**

**cupdate : out STD\_LOGIC;**

**pcnew : out STD\_LOGIC\_VECTOR (15 downto 0);**

**pctemp : in STD\_LOGIC\_VECTOR (15 downto 0);**

**rnew : out STD\_LOGIC\_VECTOR (7 downto 0);**

**rupdate : out STD\_LOGIC;**

**rst : in STD\_LOGIC);**

**end component;**

**signal tclk,tcy,tz,trst,tpcupdate,tapc,trd,twr,trupdate,tnrd,tnwr,tnmreq,twwr,twrd,tcin,tcout,tcupdate:std\_logic;**

**signal tt : std\_logic\_vector(3 downto 0);**

**signal tpctemp,tabus,tir,tpc,tpcin,tirin,taddr:std\_logic\_vector(15 downto 0);**

**signal taluout,tdataout,trout,tdataback,trnew:std\_logic\_vector(7 downto 0);**

**begin**

**u1: clock port map(clk=>tclk,rst=>trst,t=>tt);**

**u2: pcm port map(clk1=>tt(0),rst=>trst,apc=>tapc,pcupdate=>tpcupdate,pcnew=>tpcin,ir\_in=>tirin,ir\_out=>tir,pc=>tpc);**

**u3: ALU port map(clk2=>tt(1),cy=>tcy,cin=>tcin,cout=>tcout,cupdate=>tcupdate,rupdate=>trupdate,rnew=>trnew,ir=>tir,pc=>tpc,aluout=>taluout,pctemp=>tpctemp,rst=>trst,addr=>taddr);**

**u4: mcm port map(clk3=>tt(2),rst=>trst,ir=>tir,datain=>taluout,databack=>tdataback,dataout=>tdataout,rout=>trout,wwr=>twwr,wrd=>twrd);**

**u5: rewrite port map(datain=>trout,clk4=>tt(3),cin=>tcout,cout=>tcin,cupdate=>tcupdate,ir=>tir,pcupdate=>tpcupdate,pcnew=>tpcin,pctemp=>tpctemp,rnew=>trnew,rupdate=>trupdate,rst=>trst);**

**u6: mcenter port map(pc=>tpc,apc=>tapc,wrt=>twwr,rdt=>twrd,datain=>tdataout,addrt=>taddr,addr=>tabus,ir=>tirin,dataout=>tdataback,mdr=>dbus,rd=>tnrd,wr=>tnwr,cs=>tnmreq);**

**cy<=tcy;**

**z<=tz;**

**tclk<=clk;**

**trst<=rst;**

**abus<=tabus;**

**abusout<=tabus;**

**irout<=tir;**

**dbusout<=dbus;**

**tout<=tt;**

**nmreq<=tnmreq;**

**nmreqout<=tnmreq;**

**nrd<=tnrd;**

**nrdout<=tnrd;**

**nwr<=tnwr;**

**nwrout<=tnwr;**

**nbhe<='0';**

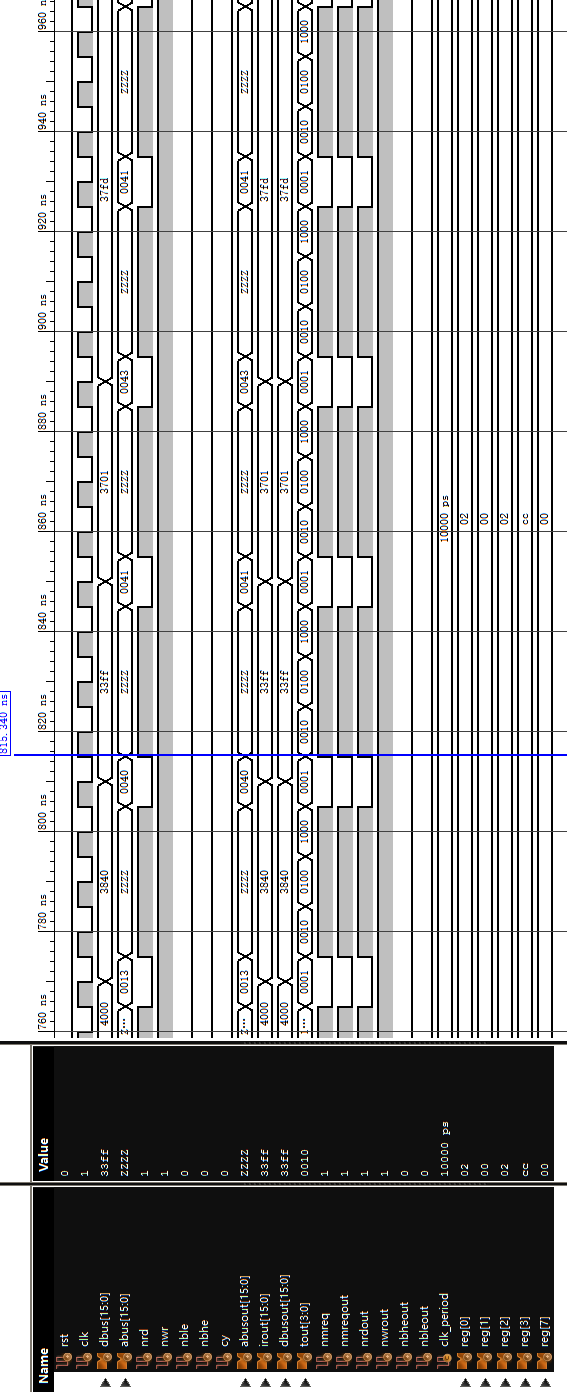
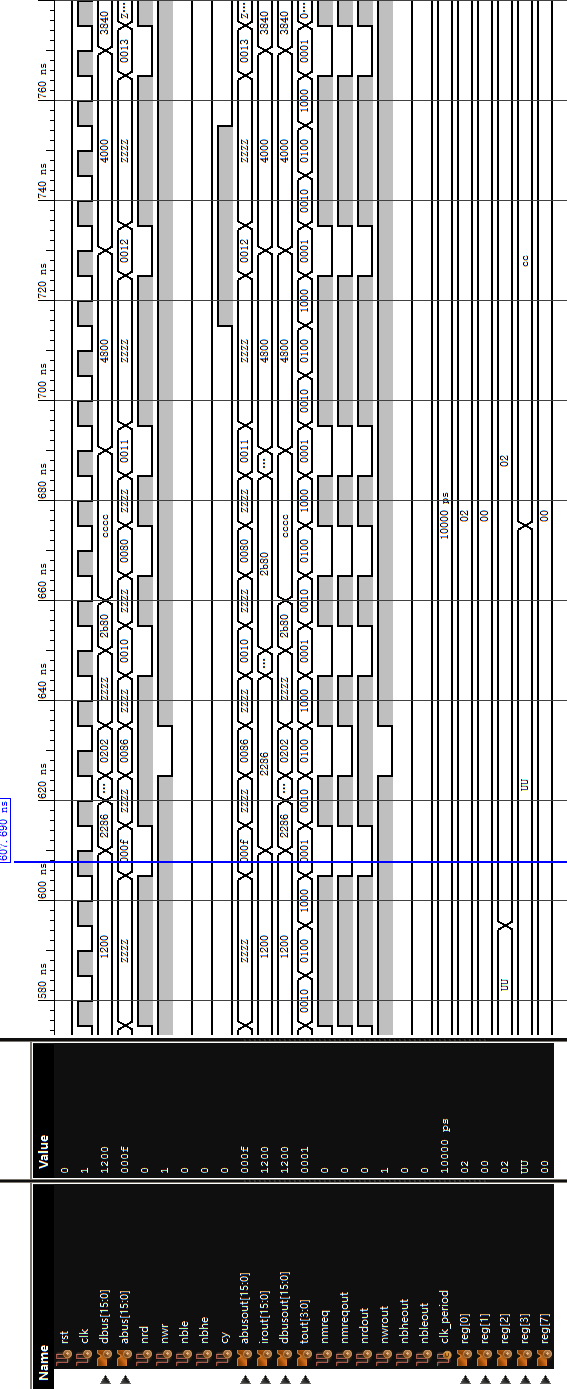
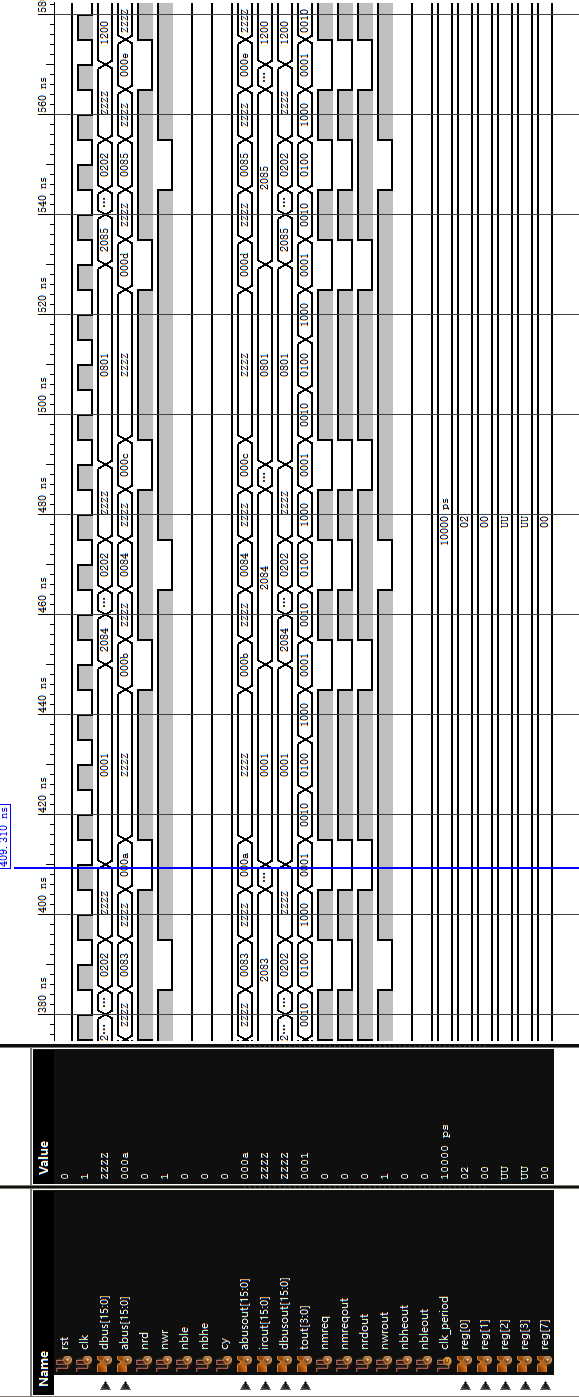
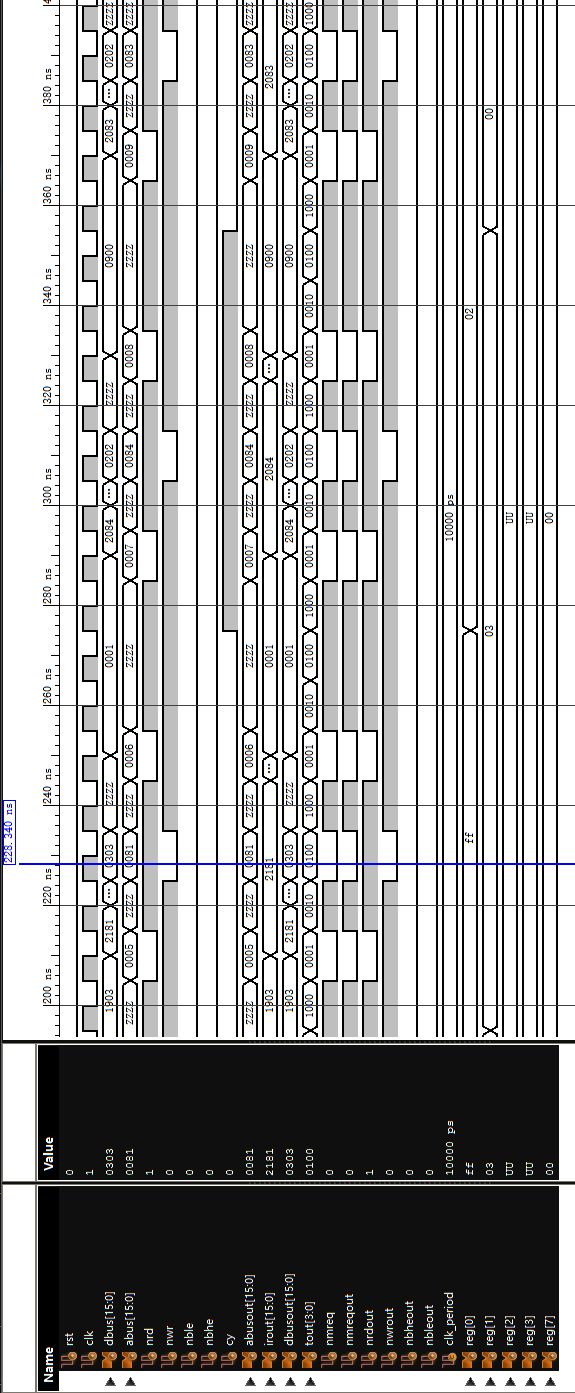
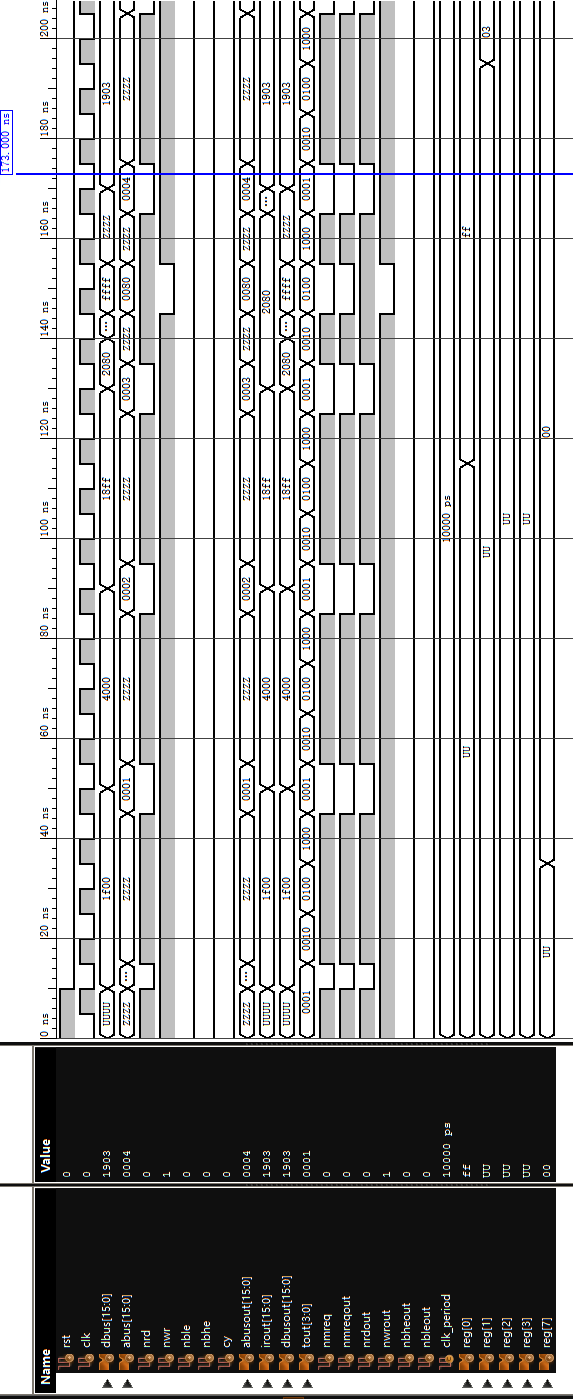
**nbheout<='0';**

**nble<='0';**

**nbleout<='0';**

**end Behavioral;**

**仿真波形：（见下页）**



## 五：下载

### 指令序列

1. MVI R7,00 1F00
2. MVI R0,FF 18FF
3. STA R0,80 2080
4. MVI R1,03 1903
5. STA R1,81 2181
6. STC 4800
7. CLC 4000
8. ADD R0,R1 0001
9. STA R0,82 2082
10. SUB R1,R0 0900
11. STA R1,83 2183
12. ADD R0,R1 0001
13. STA R0,84 2084
14. SUB R0,R1 0801
15. STA R0,85 2085
16. MOV R2,R0 1200
17. STA R2,86 2286
18. LDA R3,80 2B80
19. JMP 40 3840
20. JZ R3,FF 33FF
21. JZ R7,01 3701
22. JZ R7,FD 37FD

### Ucf文件

Net "rst" CLOCK\_DEDICATED\_ROUTE = TRUE;

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 "abusout<0>" LOC = "P31" ;

NET "abusout<1>" LOC = "P33" ;

NET "abusout<2>" LOC = "P34" ;

NET "abusout<3>" LOC = "P35" ;

NET "abusout<4>" LOC = "P36" ;

NET "abusout<5>" LOC = "P39" ;

NET "abusout<6>" LOC = "P40" ;

NET "abusout<7>" LOC = "P41" ;

NET "abusout<8>" LOC = "P126" ;

NET "abusout<9>" LOC = "P127" ;

NET "abusout<10>" LOC = "P129" ;

NET "abusout<11>" LOC = "P202" ;

NET "abusout<12>" LOC = "P203" ;

NET "abusout<13>" LOC = "P205" ;

NET "abusout<14>" LOC = "P206" ;

NET "abusout<15>" LOC = "P103" ;

NET "clk" LOC = "P75" ;

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 "dbusout<0>" LOC = "P42" ;

NET "dbusout<1>" LOC = "P45" ;

NET "dbusout<2>" LOC = "P47" ;

NET "dbusout<3>" LOC = "P48" ;

NET "dbusout<4>" LOC = "P49" ;

NET "dbusout<5>" LOC = "P50" ;

NET "dbusout<6>" LOC = "P55" ;

NET "dbusout<7>" LOC = "P56" ;

NET "dbusout<8>" LOC = "P60" ;

NET "dbusout<9>" LOC = "P61" ;

NET "dbusout<10>" LOC = "P62" ;

NET "dbusout<11>" LOC = "P63" ;

NET "dbusout<12>" LOC = "P2" ;

NET "dbusout<13>" LOC = "P108" ;

NET "dbusout<14>" LOC = "P109" ;

NET "dbusout<15>" LOC = "P112" ;

NET "irout<0>" LOC = "P4" ;

NET "irout<1>" LOC = "P5" ;

NET "irout<2>" LOC = "P8" ;

NET "irout<3>" LOC = "P9" ;

NET "irout<4>" LOC = "P11" ;

NET "irout<5>" LOC = "P12" ;

NET "irout<6>" LOC = "P15" ;

NET "irout<7>" LOC = "P16" ;

NET "irout<8>" LOC = "P18" ;

NET "irout<9>" LOC = "P19" ;

NET "irout<10>" LOC = "P22" ;

NET "irout<11>" LOC = "P23" ;

NET "irout<12>" LOC = "P24" ;

NET "irout<13>" LOC = "P25" ;

NET "irout<14>" LOC = "P28" ;

NET "irout<15>" LOC = "P29" ;

NET "nbhe" LOC = "P138" ;

NET "nbheout" LOC = "P82" ;

NET "nble" LOC = "P137" ;

NET "nbleout" LOC = "P83" ;

NET "nmreq" LOC = "P168" ;

NET "nmreqout" LOC = "P185" ;

NET "nrd" LOC = "P139" ;

NET "nrdout" LOC = "P77" ;

NET "nwr" LOC = "P152" ;

NET "nwrout" LOC = "P98" ;

NET "rst" LOC = "P51" ;

NET "cy" LOC = "P186" ;

NET "tout<0>" LOC = "P196" ;

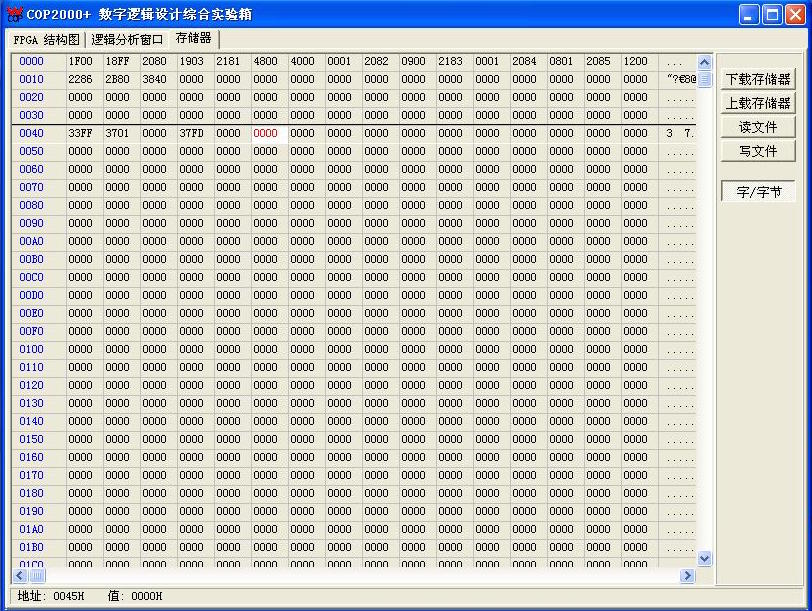
NET "tout<1>" LOC = "P193" ;

NET "tout<2>" LOC = "P192" ;

NET "tout<3>" LOC = "P190" ;

### 下载概览

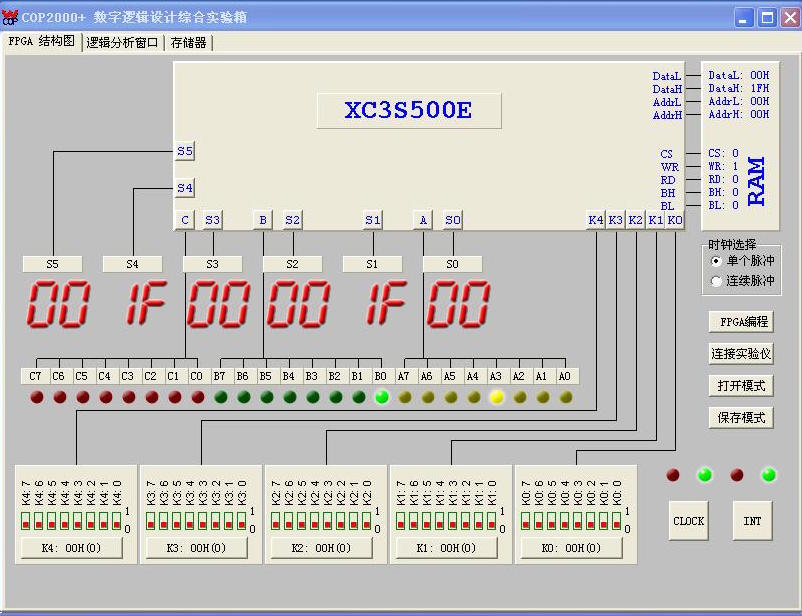
1.首先将指令写入存储器



2.第一条指令(MVI R7,00)取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前pc=0000H，在该存储单元中的指令为1F00。

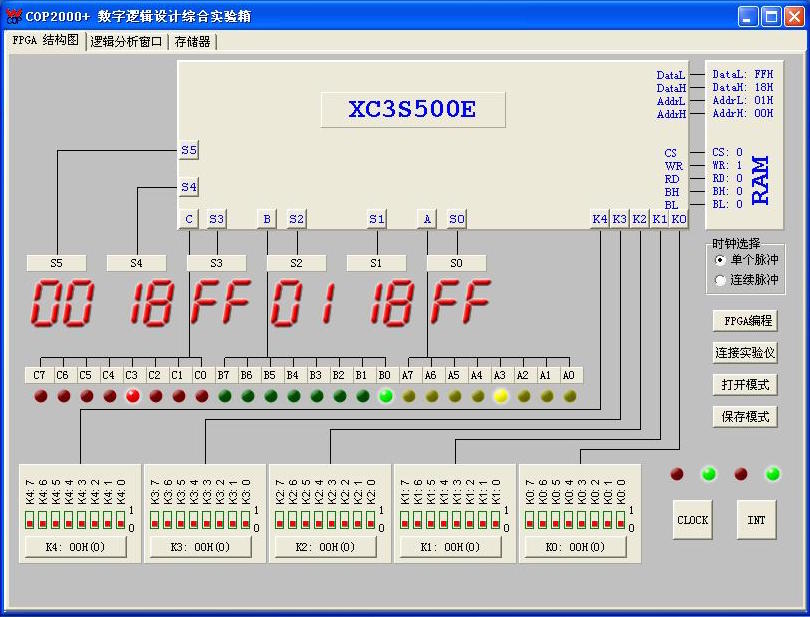
指令含义为将00送入R7 寄存器中。



3. (MVI R0,FF)取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前pc=0001H，在该存储单元中的指令为18FF。

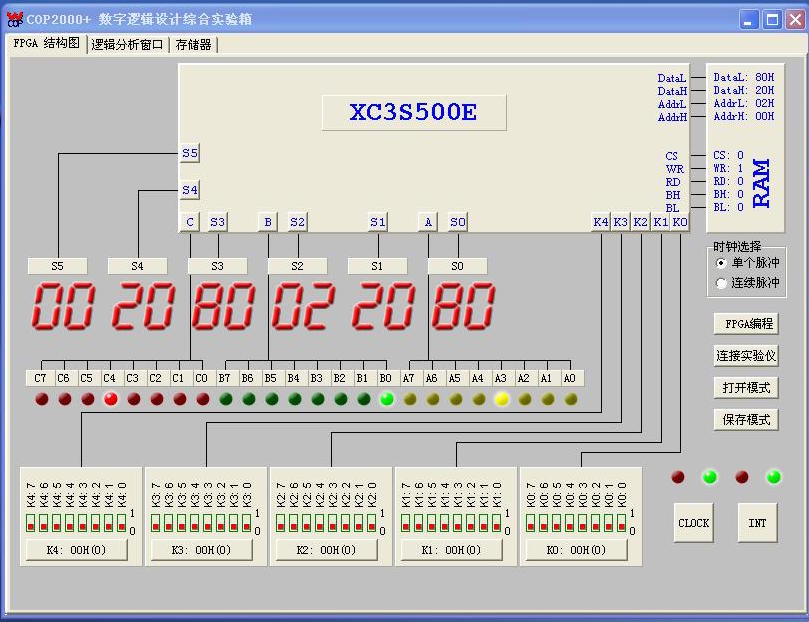
指令含义为将FF送入R0 寄存器中。



4.(STA R0,80) 取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前pc=0002H，在该存储单元中的指令为2080。

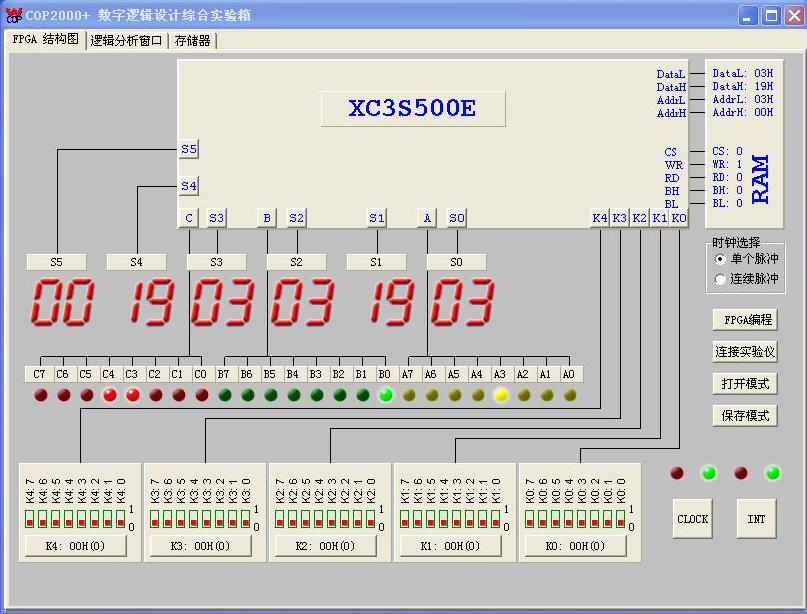
指令含义为将R0寄存器中的数值送入地址为R7&80的存储单元中。



5. (MVI R1,03)取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前pc=0003H，在该存储单元中的指令为2080。

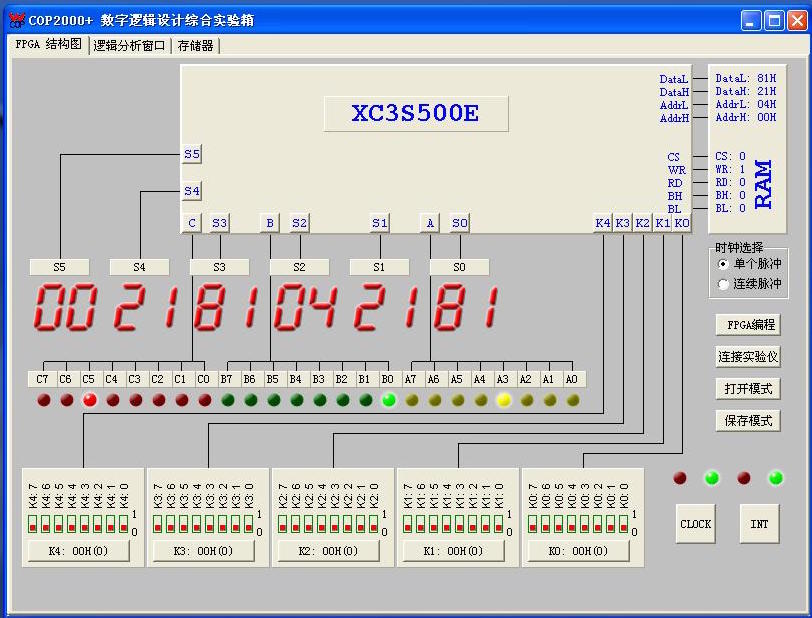
指令含义为将03送入R1 寄存器中。



6.(STA R1,81) 取指周期,存储器读信号、片选、高地位信号均有效。

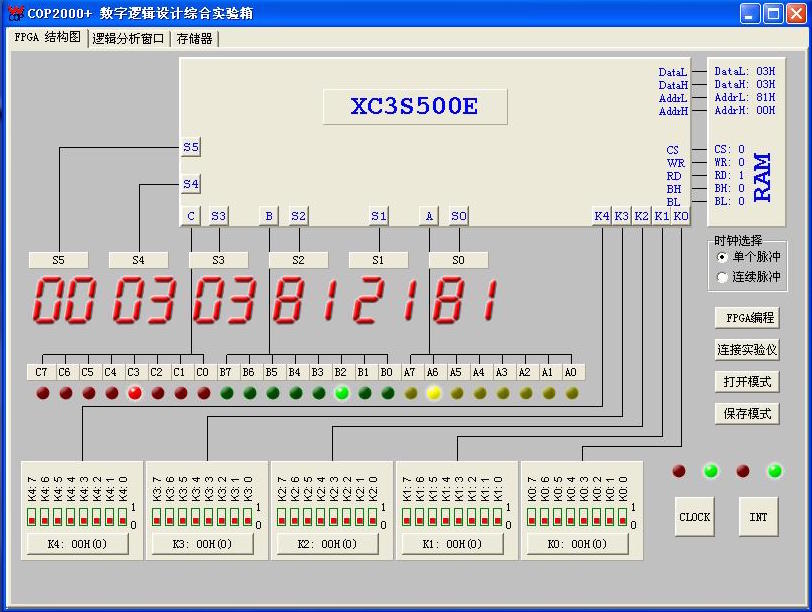
S4S3显示数据总线，S2S5显示地址总线。当前pc=0004H，在该存储单元中的指令为2181。

指令含义为将R1寄存器中的数值送入地址为R7&81的存储单元中。

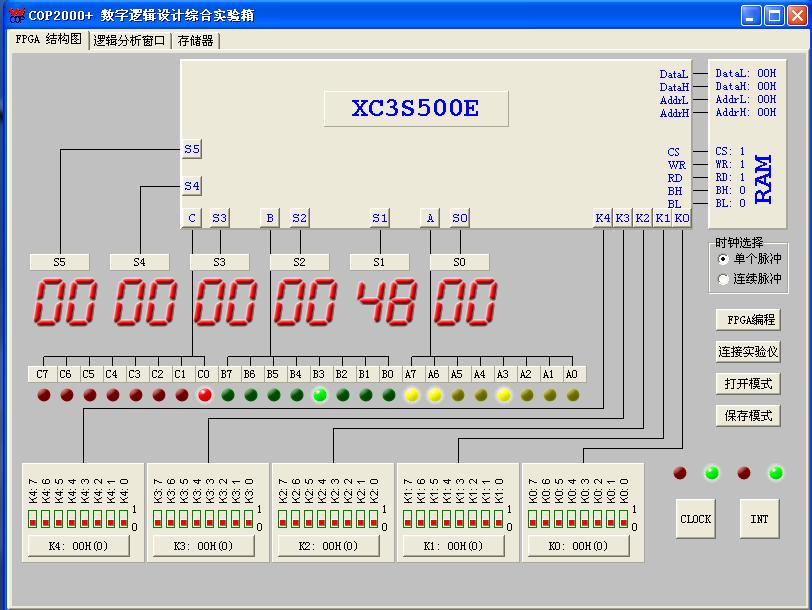


7. (STA R1,81) 存取周期,存储器写信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前地址总线上为0081H，数据总线上的数据位刚刚存入R1中的值03H。



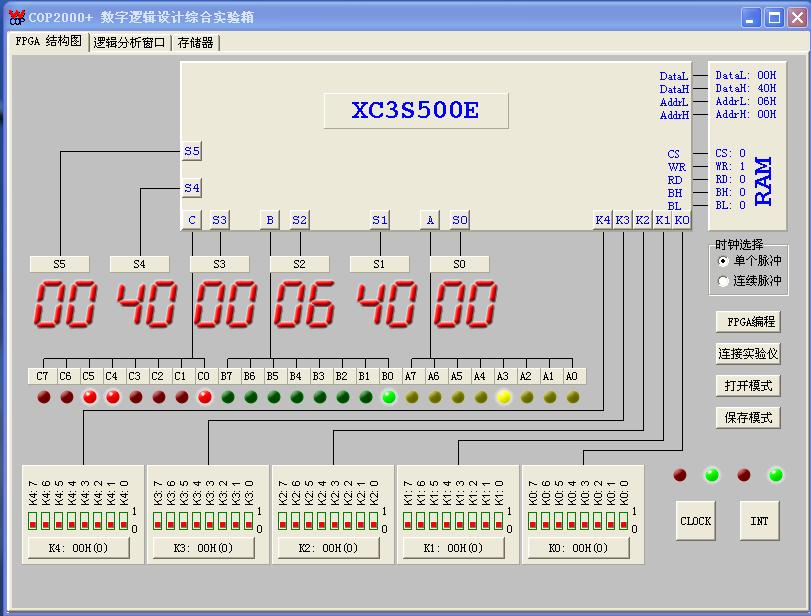
8. (STC) 回写周期，C0灯显示进位位，此时进位位已经被置为1。



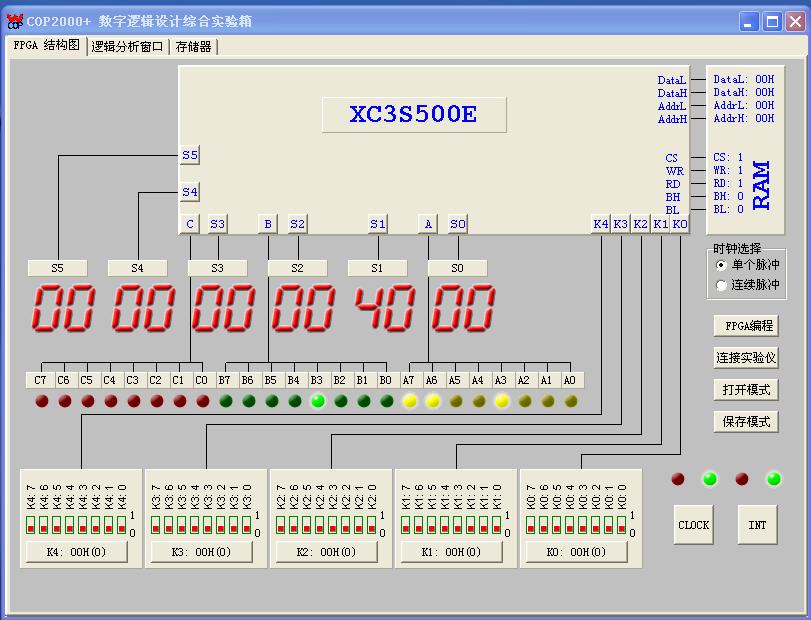
9 (CLC) 取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前pc=0006H，在该存储单元中的指令为4000。

指令含义为将进位位置为0。



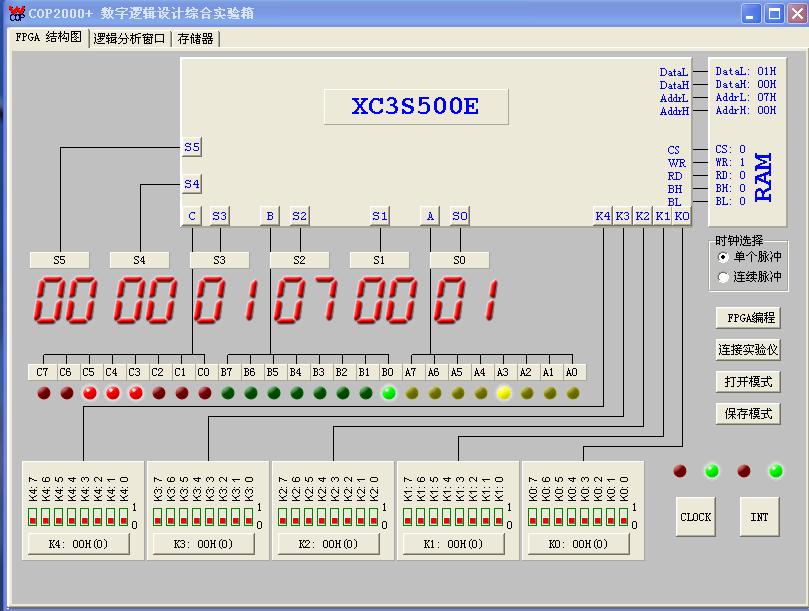
10. (CLC) 回写周期，C0灯显示进位位，此时进位位已经被置为0。



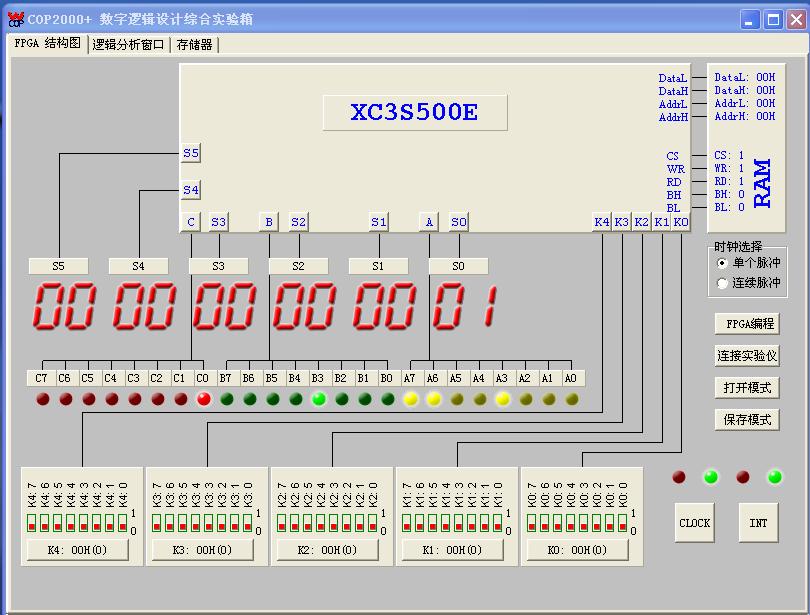
11.(ADD R0,R1) 取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前pc=0007H，在该存储单元中的指令为0001。

指令含义为将R0,R1,CY相加结果送给R0。



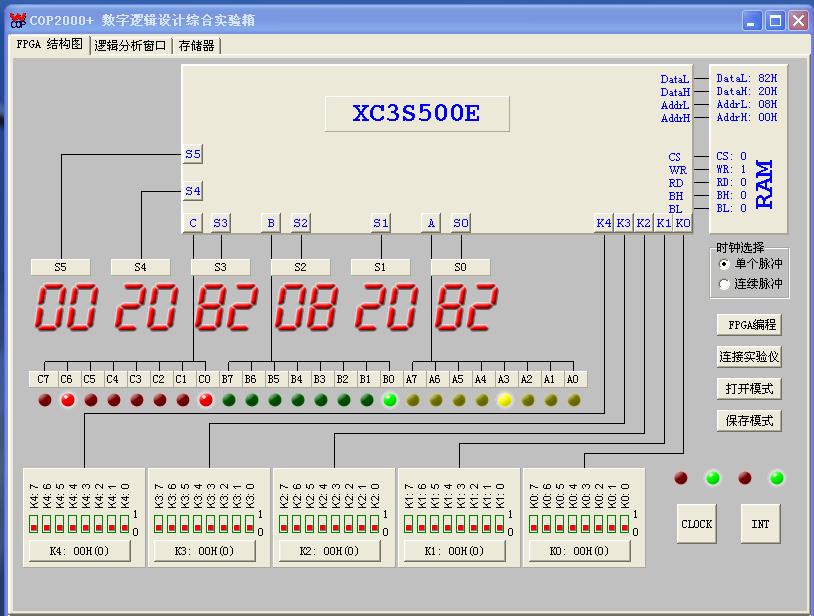
12. (ADD R0,R1) 回写周期，C0灯显示进位位。R0为FFH，R1为03H,CY为0。运算结果为102H，产生进位位。进位位灯亮。



13. (STA R0,82) 取指周期,存储器读信号、片选、高地位信号均有效。

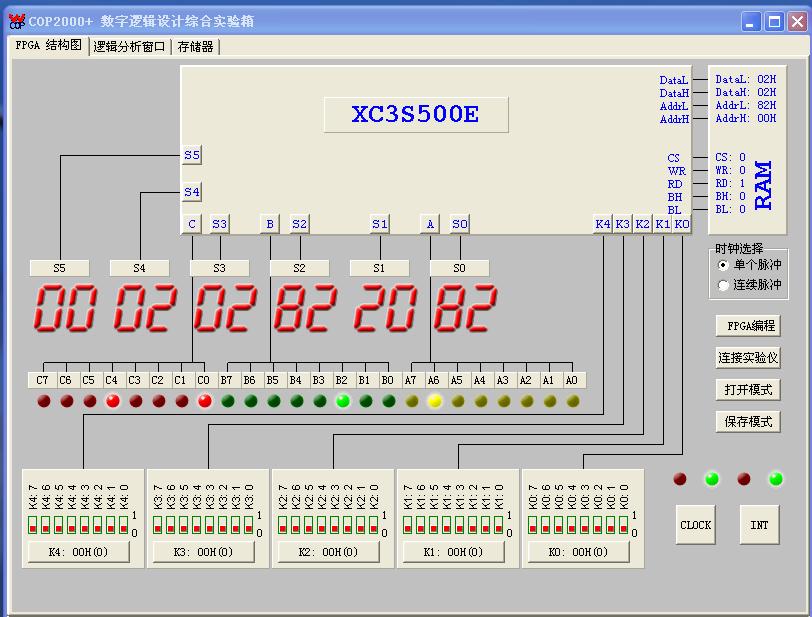
S4S3显示数据总线，S2S5显示地址总线。当前pc=0008H，在该存储单元中的指令为2082。

指令含义为将R0寄存器中的数值送入地址为R7&82的存储单元中。



14. (STA R0,82) 存取周期,存储器写信号、片选、高地位信号均有效。

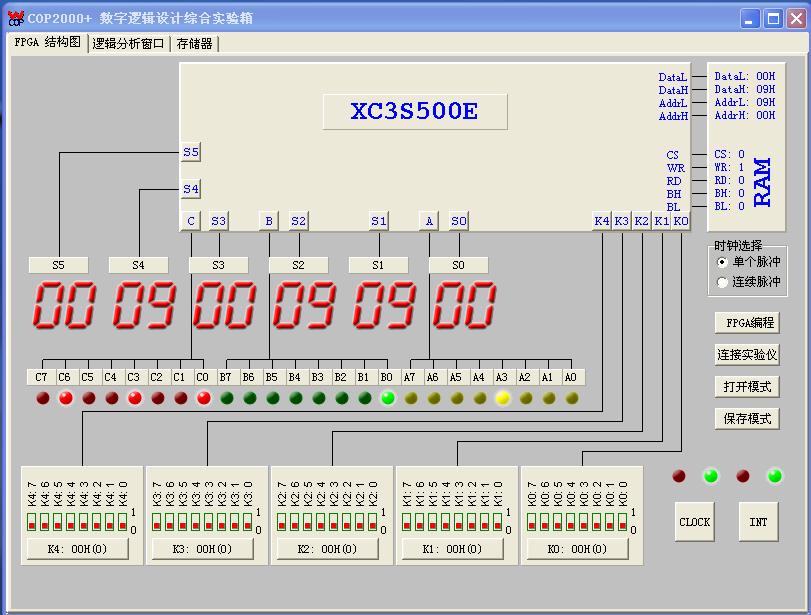
S4S3显示数据总线，S2S5显示地址总线。当前地址总线上为0082H，数据总线上的数据位刚刚存入R0中的值02H。



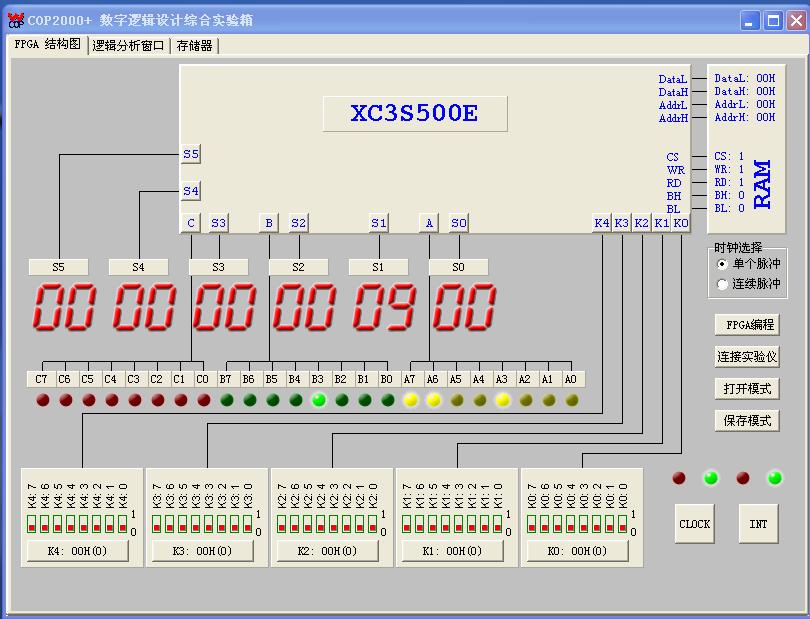
15.( SUB R1,R0 )取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前pc=0009H，在该存储单元中的指令为0900。

指令含义为将R1,R0,CY相减结果送给R1。



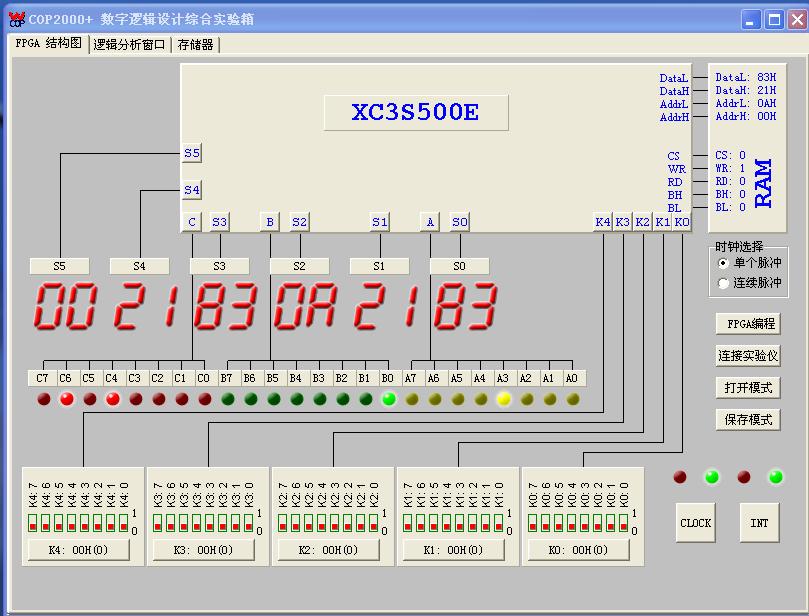
16. (SUB R1,R0) 回写周期，C0灯显示进位位。R0为02H，R1为03H,CY为1。运算结果为00H，无进位位。进位位灯灭。



17. (STA R1,83) 取指周期,存储器读信号、片选、高地位信号均有效。

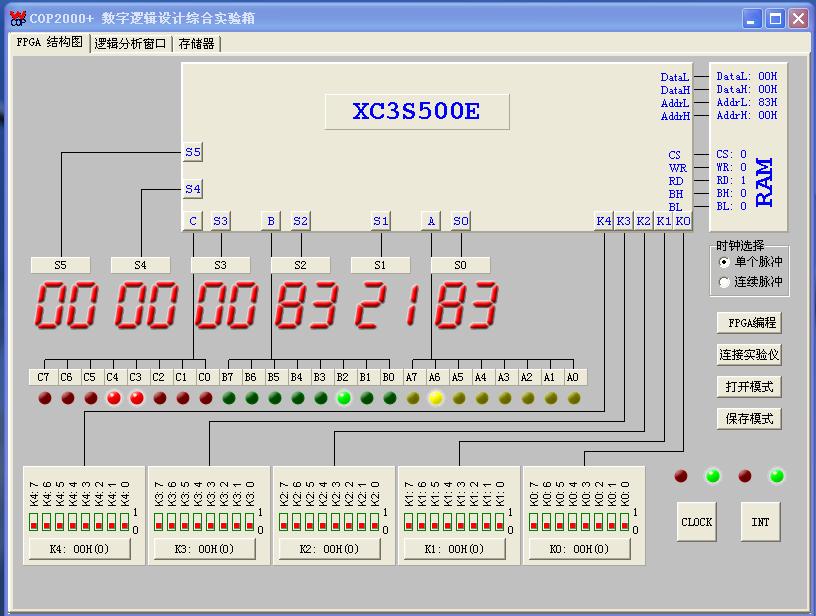
S4S3显示数据总线，S2S5显示地址总线。当前pc=000AH，在该存储单元中的指令为2183。

指令含义为将R1寄存器中的数值送入地址为R7&83的存储单元中。



18. (STA R1,83) 存取周期,存储器写信号、片选、高地位信号均有效。

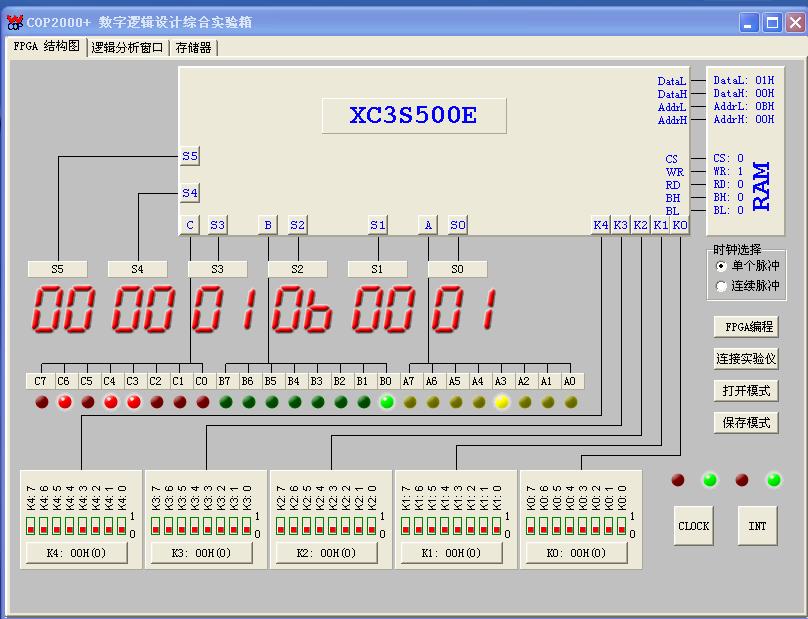
S4S3显示数据总线，S2S5显示地址总线。当前地址总线上为0083H，数据总线上的数据位刚刚存入R1中的值00H。



19. (ADD R0,R1) 取指周期,存储器读信号、片选、高地位信号均有效。

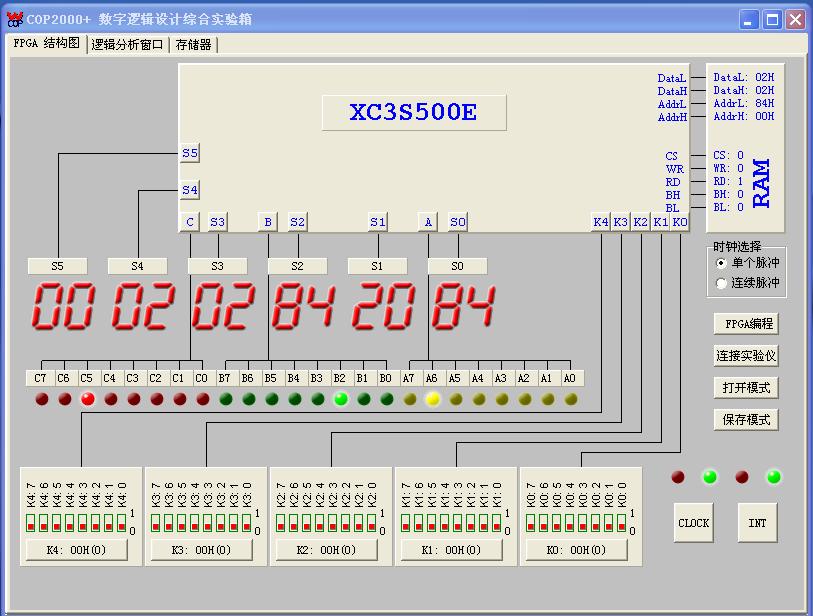
S4S3显示数据总线，S2S5显示地址总线。当前pc=000BH，在该存储单元中的指令为0001。

指令含义为将R0,R1,CY相加结果送给R0。



20. (STA R0,84) 存取周期,存储器写信号、片选、高地位信号均有效。

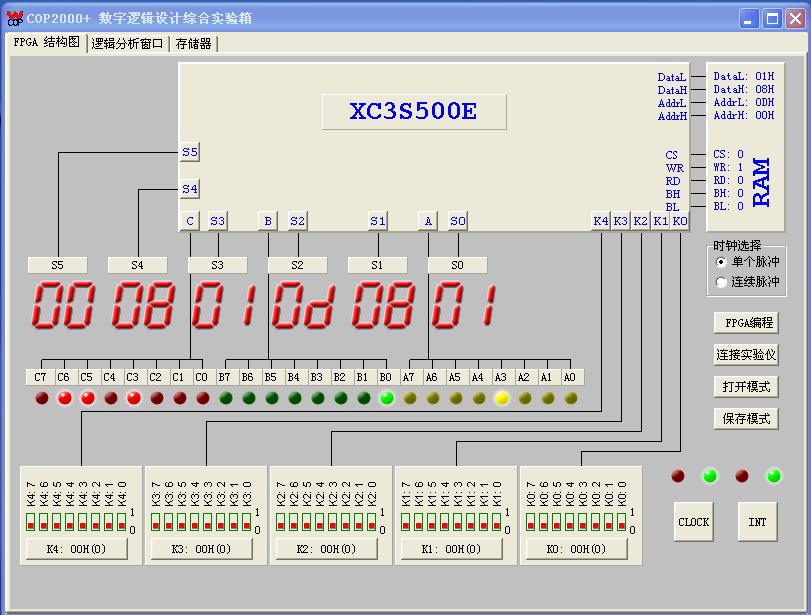
S4S3显示数据总线，S2S5显示地址总线。当前地址总线上为0084H，数据总线上的数据位刚刚存入R0中的值02H。



21. (SUB R0,R1) 取指周期,存储器读信号、片选、高地位信号均有效。

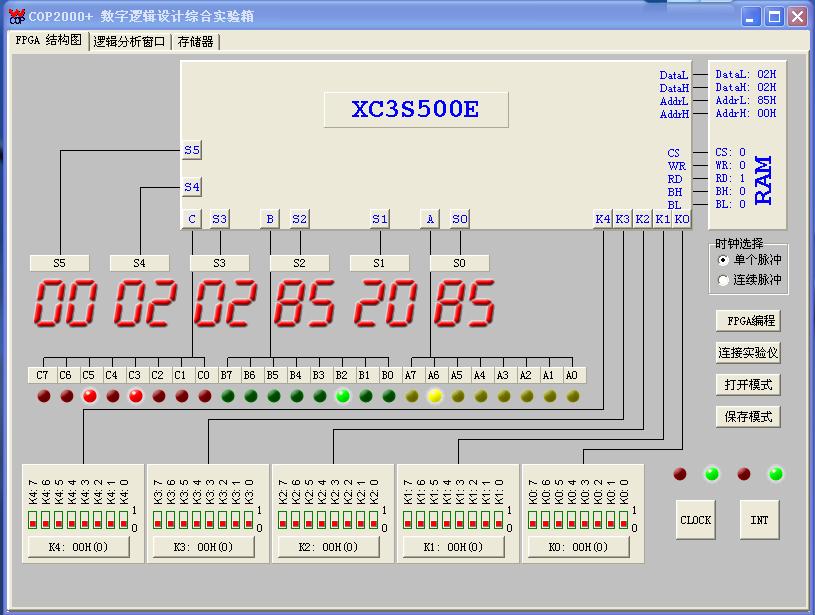
S4S3显示数据总线，S2S5显示地址总线。当前pc=000DH，在该存储单元中的指令为0801。

指令含义为将R0,R1,CY相减结果送给R0。



22. (STA R0,85) 存取周期,存储器写信号、片选、高地位信号均有效。

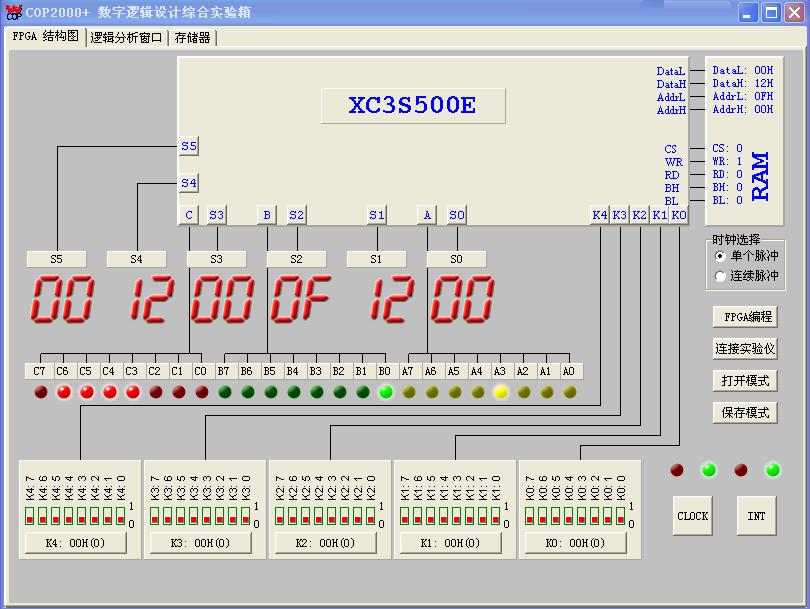
S4S3显示数据总线，S2S5显示地址总线。当前地址总线上为0085H，数据总线上的数据为刚刚存入R0中的值02H。



23.（MOV R2,R0）取指周期,存储器读信号、片选、高地位信号均有效。

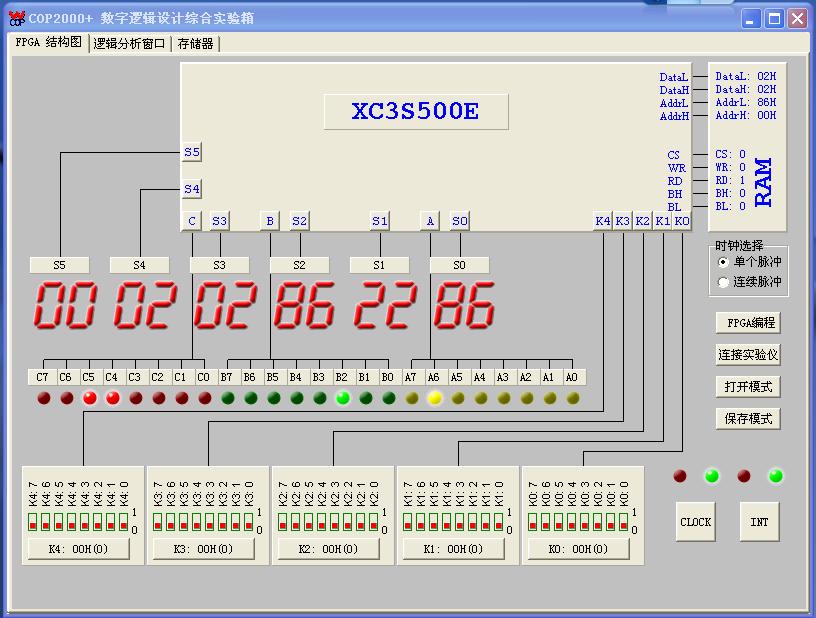
S4S3显示数据总线，S2S5显示地址总线。当前pc=000FH，在该存储单元中的指令为1200。

指令含义为将R0数据送给R2。



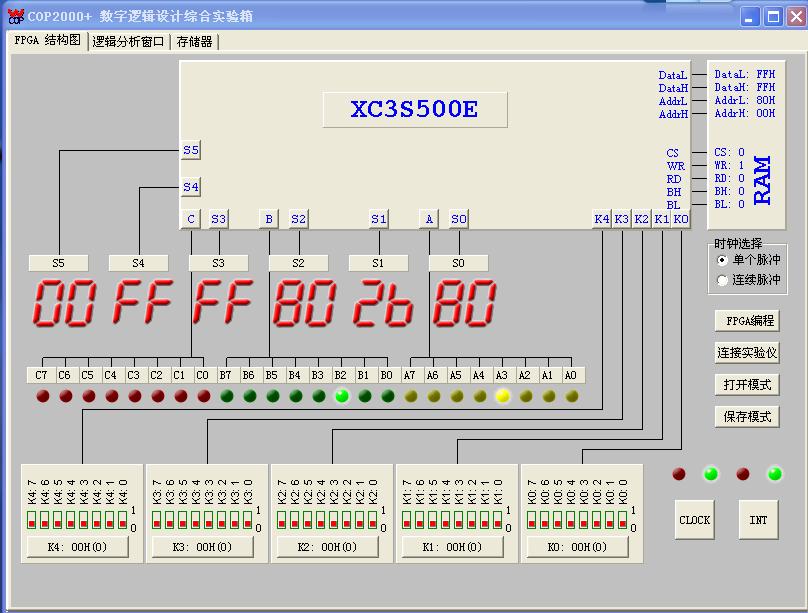
24.（STA R2,86）存取周期,存储器写信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前地址总线上为0086H，数据总线上的数据为刚刚存入R2中的值02H。



25.（LDA R3,80）存取周期,存储器读信号、片选、高地位信号均有效。

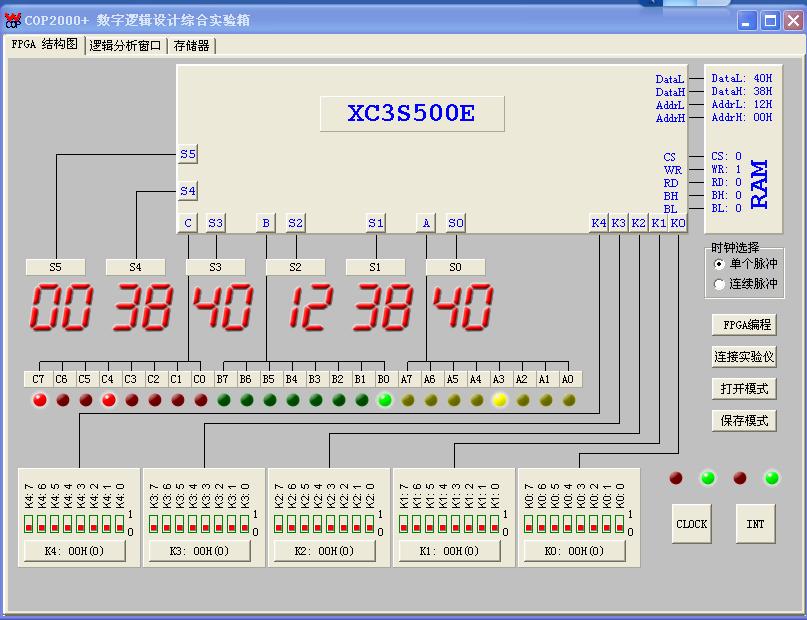
S4S3显示数据总线，S2S5显示地址总线。当前地址总线上为0080H，数据总线上的数据为准备送入R3中的值FFH。



26.(JMP 40) 取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前pc=0012H，在该存储单元中的指令为3840。

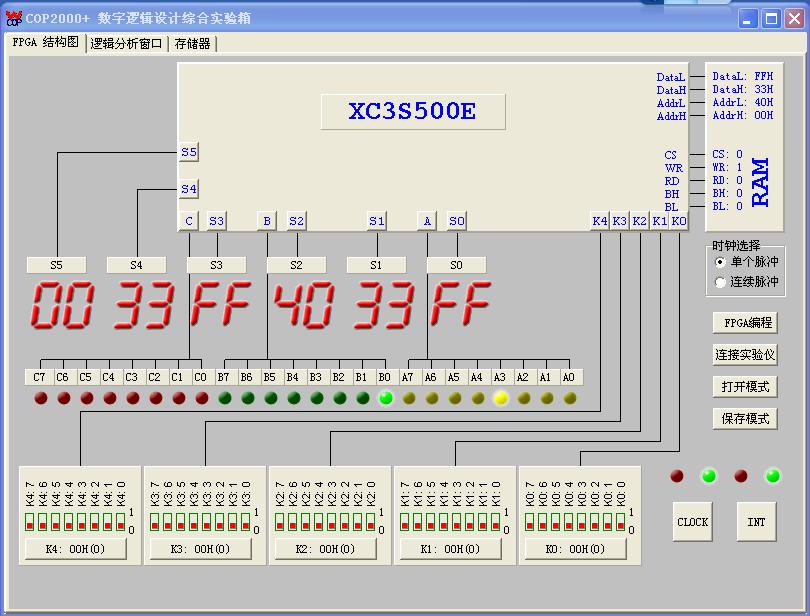
指令含义为下一条指令无条件跳转至0040H。



27.(JZ R3,FF) 取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前已经跳转至0040H，在该存储单元中的指令为33FF。

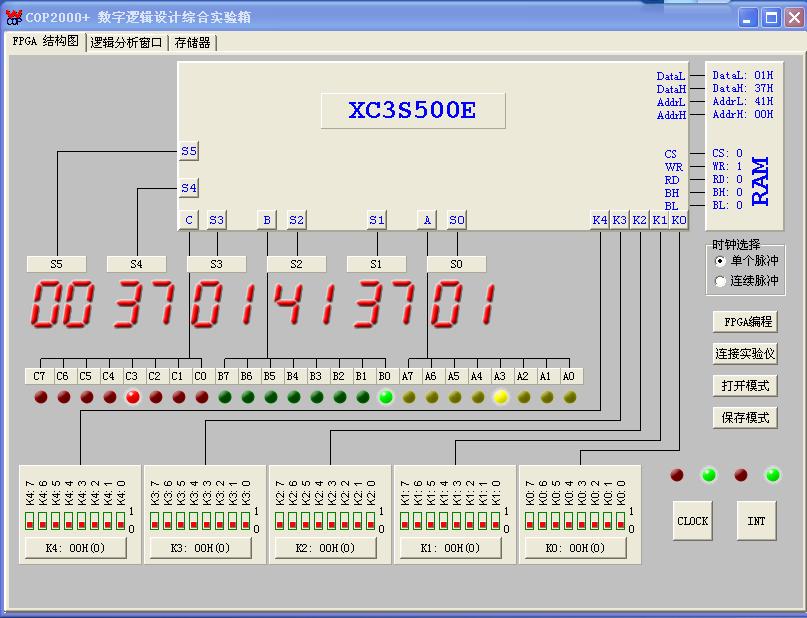
指令含义为若R3为0下一条指令向后跳转一个单位。但此时R3不为1，则正常pc+1。



28．（JZ R7,01 ）取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前pc=0041H，在该存储单元中的指令为3701。

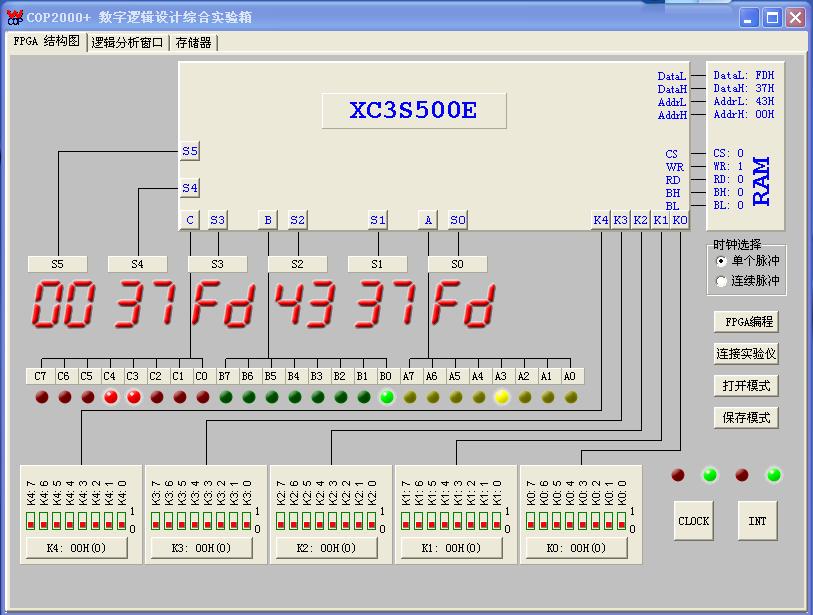
指令含义为若R7为0下一条指令向前多跳转一个单位。



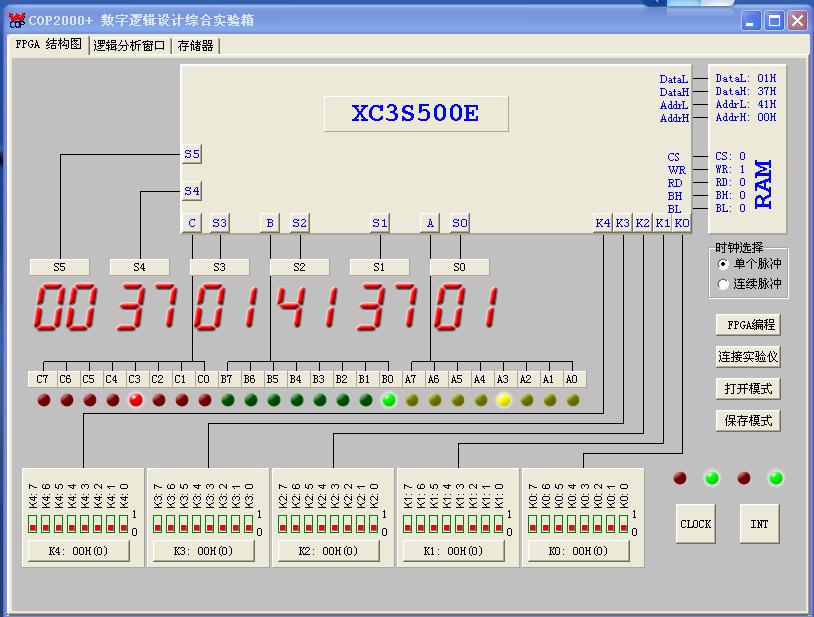
29. （JZ R7,FD ）取指周期,存储器读信号、片选、高地位信号均有效。

S4S3显示数据总线，S2S5显示地址总线。当前指令地址已经跳转至0043H，在该存储单元中的指令为37FD。

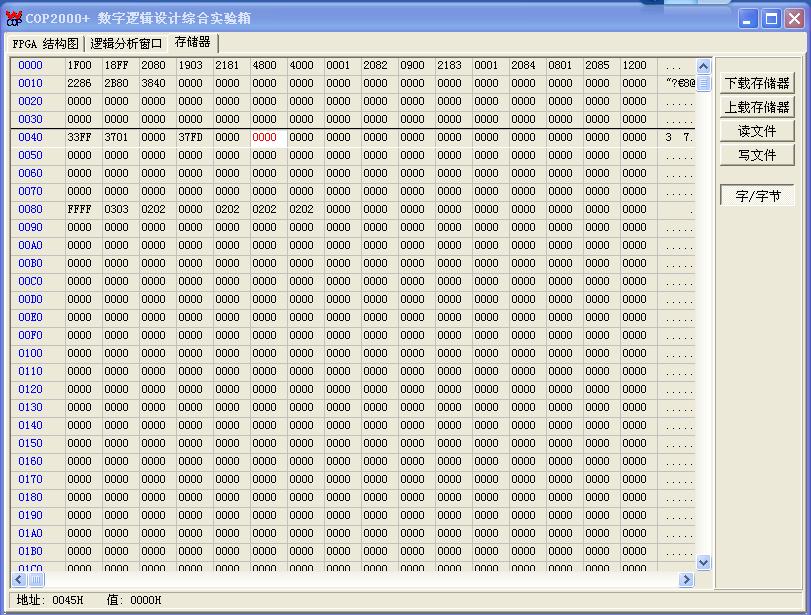
指令含义为若R7为0下一条指令向后多跳转三个单位。



30.此时pc跳转至0041H.



31.指令结束后存储器中的数据。与运算结果相符合。



## 六、遇到的问题及解决方法

1. 问题一：在最初，储存器的读写信号是由运算单元分析后送至存储管理单元在送到访存控制单元。下载到板子上后，存取数指令在第三节拍时并不能正常输出控制读取的命令。

分析：为了控制读写允许信号的时间，在第二节拍后读写信号已经被置为无效了。等信号到第三节拍时，被传给访存控制的读写信号已经无效了，不能正确读取。

解决方法：在存储模块加入ir译码模块，对不同的指令由存储控制模块直接分析后输出控制命令。

1. 问题二：在FPGA上，当加法运算产生进位位时，出现进位位被重复加一次的情况。

分析：原来产生进位位的方式是在第二节拍的下降沿进行置位，猜测可能是在下降沿的时候硬件又进行了把新产生的进位位带回去运算。

解决方法：使用rupdate信号和ir指令的判断，在第四节拍对加减法的cy 进行更新

1. 问题三：在采用了上面的方法后，不会出现产生的进位位又被加一次的情况了，但上个运算产生的进位位并不能参与运算

分析：这个原因找了好久也没有找到问题所在

解决方法：将产生的进位位送到回写单元，在第四节拍时在更新控制信号控制下回写至运算单元中.