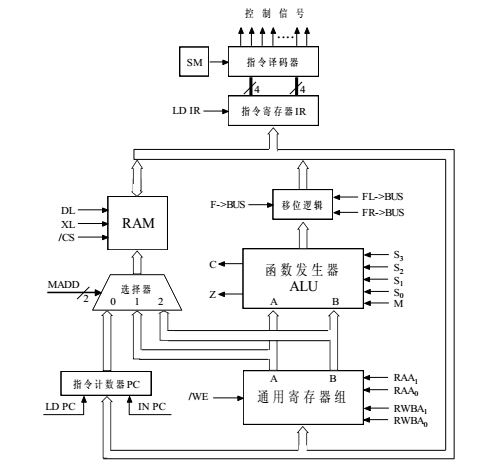
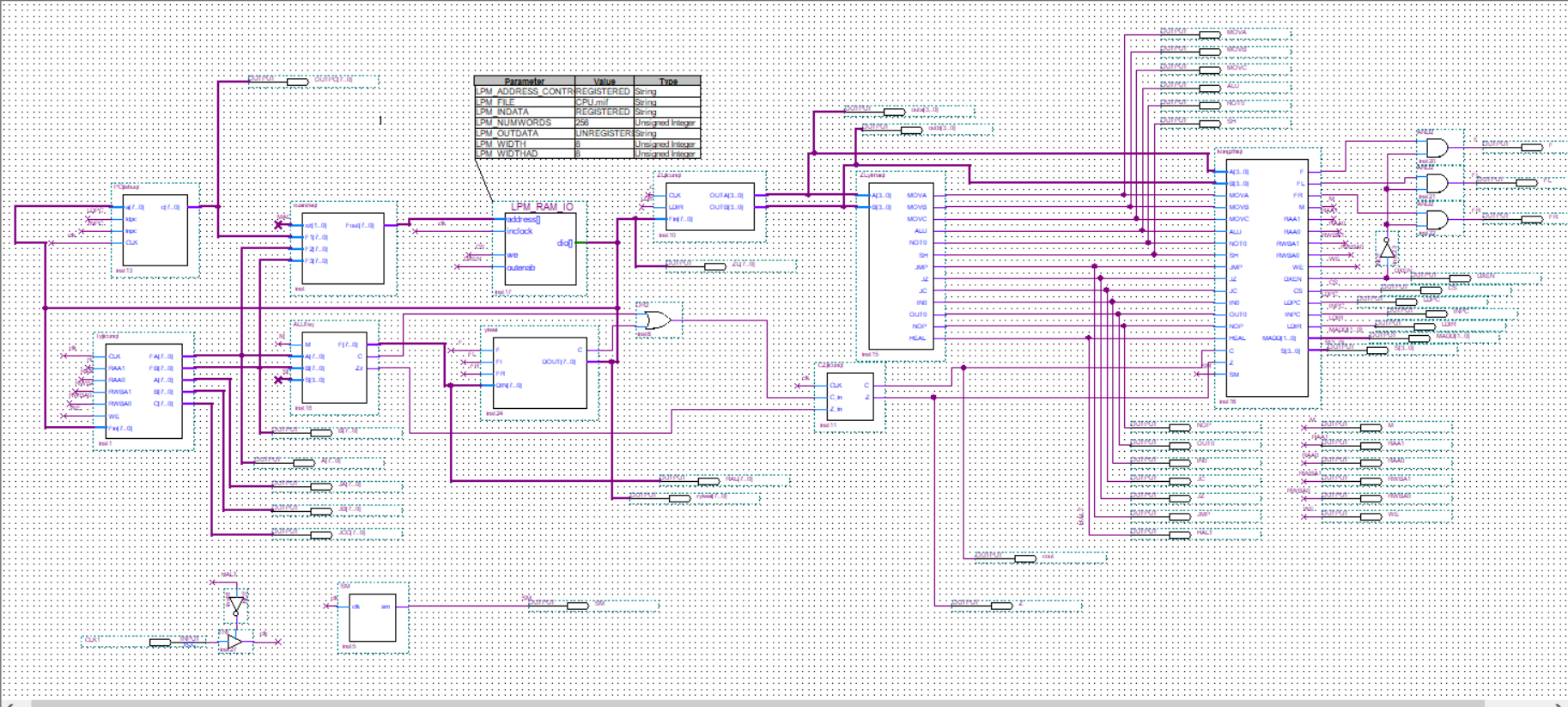
设计的整体架构

整体由8个模块构成，分别为计数器、通用寄存器组、选择器、寄存器、指令寄存器、指令译码器、控制器、函数发生器。计数器用来计数和传导。通用寄存器组包括三个寄存器A（00）、B（01）、C（10）。选择器用来选择输出与哪个端口的输入相等。寄存器存储指令，并且与指令寄存器相连。指令译码器对指令进行分解，分解为S+R1+R2。控制器根据译码器来选择指令。函数发生器用来实现逻辑运算。连接如下图所示：

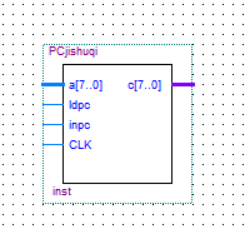




各模块的具体实现

PC计数器：

接口如图：



实现功能：

|  |  |  |  |
| --- | --- | --- | --- |
| CLK | IN PC | LD PC | 功能 |
|  | 1 | 0 | PC PC+1 |
|  | 0 | 1 | PC 输入 |

Vhdl代码如下：

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**use IEEE.STD\_LOGIC\_UNSIGNED.ALL;**

**entity PCjishuqi is**

**PORT (**

**a: in std\_logic\_vector(7 downto 0);**

**ldpc: in std\_logic;**

**inpc:IN STD\_LOGIC;**

**CLK: in std\_logic;**

**c: out std\_logic\_vector(7 downto 0)**

**);**

**end PCjishuqi;**

**architecture dataflow1 of PCjishuqi is**

**SIGNAL DATA:STD\_LOGIC\_VECTOR(7 DOWNTO 0);**

**begin**

**process (CLK,ldpc,inpc)**

**BEGIN**

**IF CLK'EVENT AND CLK='0' THEN**

**IF ldpc='1' and inpc='0' THEN**

**DATA<=a;**

**elsif inpc='1' and ldpc='0' then**

**DATA<=DATA+"00000001";**

**if data="11111111" then data<="00000000";**

**end if;**

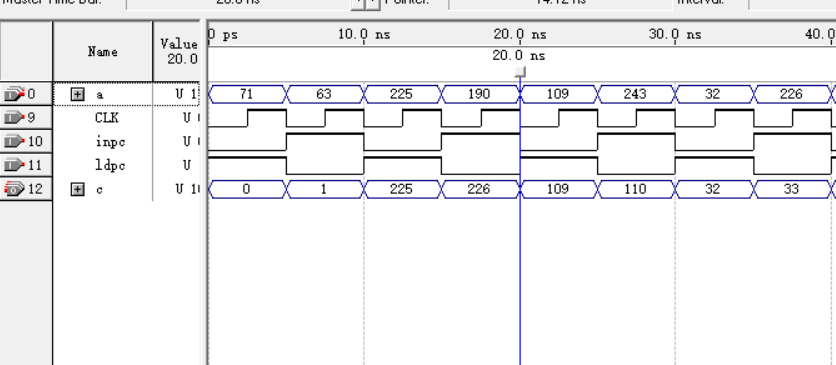
**END IF;**

**END IF;**

**END PROCESS;**

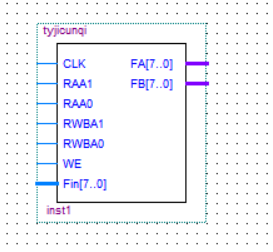
**c<=DATA;**

**end dataflow1;**



通用寄存器组：

接口如图：



实现功能：从A、B两个口读取寄存器A、B、C中的内容，在WE的控制下向A、B、C写入地址。WE高电平有效。A、B、C由R1、R2的编码控制。

代码如下：

library ieee;

use ieee.std\_logic\_1164.all;

entity tyjicunqi is

port(CLK,RAA1,RAA0:in std\_logic;

RWBA1,RWBA0:in std\_logic;

WE:in std\_logic;

Fin:in std\_logic\_vector(7 downto 0);

FA:out std\_logic\_vector(7 downto 0);

FB:out std\_logic\_vector(7 downto 0));

end tyjicunqi;

architecture data3 of tyjicunqi is

signal R1:std\_logic\_vector(7 downto 0):="10011000";-- 48<-152->76

signal R2:std\_logic\_vector(7 downto 0):="00000000";-- 7

signal C:std\_logic\_vector(7 downto 0):="00001010";--

begin

process(CLK,RAA1,RAA0,RWBA1,RWBA0,Fin,WE)

begin

if((CLK'event and CLK='0')and WE='0')then

if(RWBA1='0'and RWBA0='0')then

R1<=Fin;

elsif(RWBA1='1'and RWBA0='0')then

C<=Fin;

elsif(RWBA1='0'and RWBA0='1')then

R2<=Fin;

end if;

end if;

if(RWBA1='0'and RWBA0='0')then

FB<=R1;

elsif(RWBA1='1'and RWBA0='0')then

FB<=C;

elsif(RWBA1='0'and RWBA0='1')then

FB<=R2;

end if;

if(RAA1='0'and RAA0='1')then

FA<=R2;

elsif(RAA1='1' and RAA0='0')then

FA<=C;

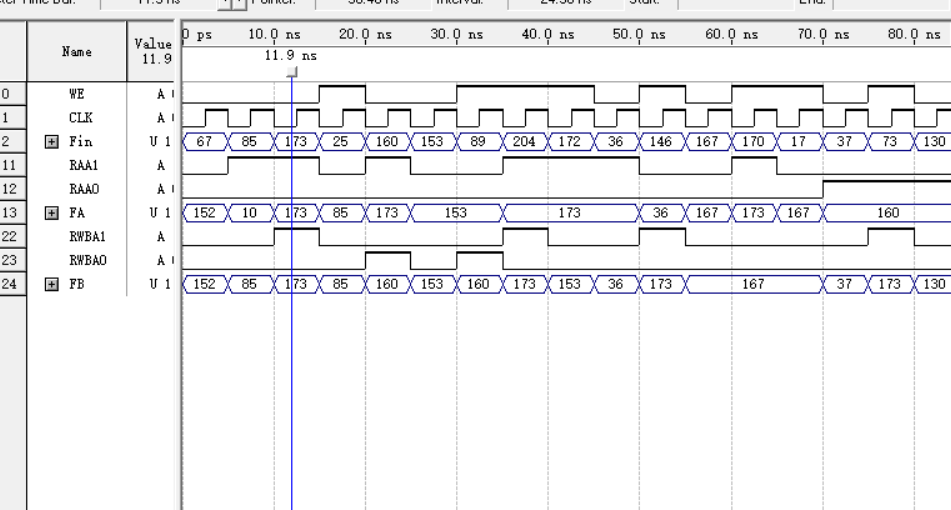
elsif(RAA1='0'and RAA0='0')then

FA<=R1;

end if;

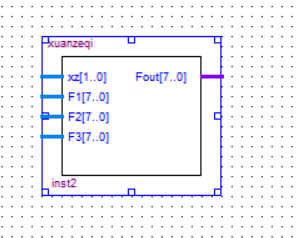
end process;

end data3;



选择器：

接口如图：



实现功能：在xz的控制下选择输出与0、还是1还是2端口的输入相等。其中端口0连接计数器，端口1连接通用寄存器组的A口，端口2连接通用寄存器组的B口。

代码如下：

library ieee;

use ieee.std\_logic\_1164.all;

entity xuanzeqi is

port(xz:in std\_logic\_vector(1 downto 0);

F1:in std\_logic\_vector(7 downto 0);

F2:in std\_logic\_vector(7 downto 0);

F3:in std\_logic\_vector(7 downto 0);

Fout:out std\_logic\_vector(7 downto 0));

end entity;

architecture data2 of xuanzeqi is

begin

process(xz,F1,F2,F3)

begin

if(xz="00")then Fout<=F1;

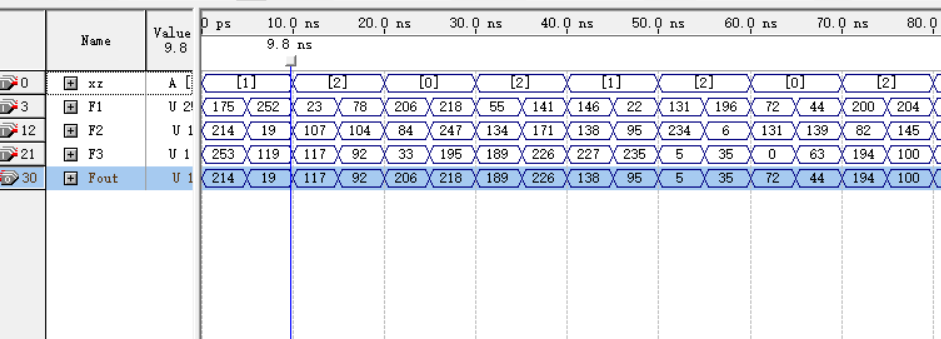
elsif(xz="01")then Fout<=F2;

elsif(xz="10")then Fout<=F3;

end if;

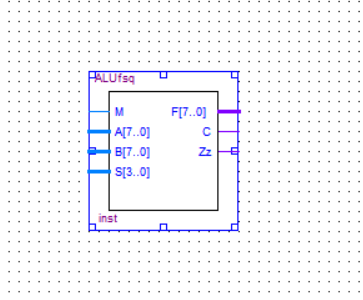
end process;

end data2;



函数发生器：

接口如图：



实现功能：在M和S的控制下，实现对R1、R2的逻辑运算和直传，包括（s=1001,m=1）加、（s=0110,m=1）减、（s=1011,m=1）或、（s=0101,m=1）非、（s=1010,m=0）直传B，（s=1111,m=0）直传A。输出C为进位输出，Zz为判断结果输出。其中R1对应A口，R2对应B口。

代码如下：

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

use ieee.std\_logic\_unsigned.all;

entity ALUfsq is

Port ( M : in STD\_LOGIC;

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

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

S : in STD\_LOGIC\_VECTOR (3 downto 0);

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

C,Zz : out STD\_LOGIC);

end ALUfsq;

architecture data4 of ALUfsq is

begin

process(S,A,B,M)

variable midc:std\_logic:='0';--linshibianliang

variable midz:std\_logic:='0';

variable res:std\_logic\_vector(8 downto 0);

begin

if(S="1001"and M='1')then

res:=('0'&A)+('0'&B);

midc:=res(8);

F<=res(7 downto 0);

if(res(7 downto 0)="00000000")then midz:='1';

else midz:='0';

end if;

elsif(S="0110"and M='1')then

res:=('0'&A)-('0'&B);

midc:=res(8);

F<=res(7 downto 0);

if(res(7 downto 0)="00000000")then midz:='1';

else midz:='0';

end if;

elsif(S="1011"and M='1')then

F<=(A or B);

elsif(S="0101"and M='1')then

F<=(not B);

elsif(S="1010"and M='0')Then

F<=B;

elsif(S="1111"and M='0')Then

F<=A;

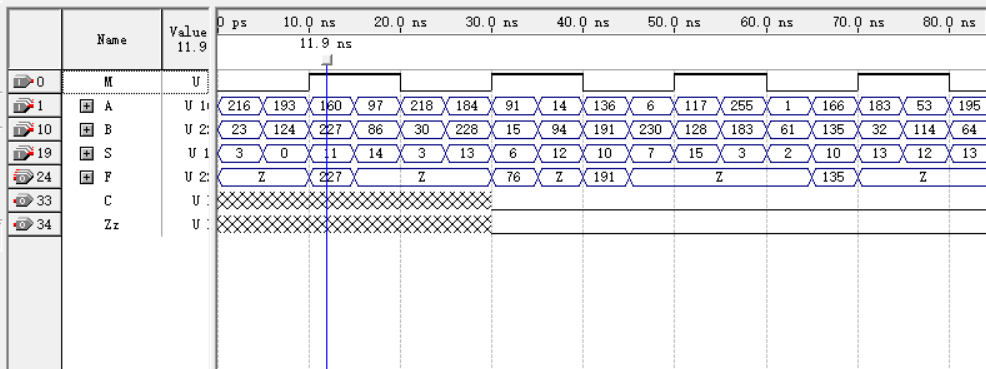
else F<="ZZZZZZZZ";

end if;

C<=midc;

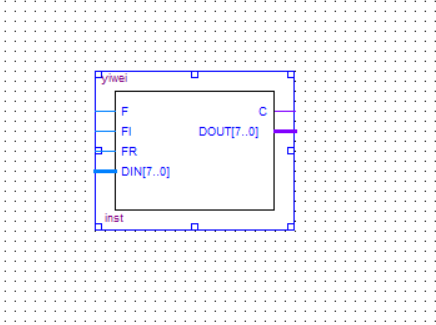
Zz<=midz;

end process;

end data4;

移位寄存器：

接口如图：



实现功能：

在F,FL,FR的控制下对DIN进行循环移位，C为被移动的那一位，DOUT为循环移位后的结果

代码如下：  
library ieee;

use ieee.std\_logic\_1164.all;

entity yiwei is

port(F,Fl,FR:in std\_logic;

DIN:in std\_logic\_vector(7 downto 0);

C:out std\_logic;

DOUT:out std\_logic\_vector(7 downto 0));

end yiwei;

architecture data of yiwei is

signal Dtmp:std\_logic\_vector(7 downto 0);

signal Ctmp:std\_logic;

begin

process(F,FL,FR,DIN)

begin

if F='1' and FL='0'and FR='0' then

Dtmp<=DIN;

Ctmp<='0';

elsif FL='1' and F='0' and FR='0' then

Dtmp<=DIN(6 downto 0)&DIN(7);

Ctmp<=DIN(7);

elsif FR='1' and F='0' and FL='0' then

Dtmp<=DIN(0)&DIN(7 downto 1);

Ctmp<=DIN(0);

elsif F='0' then

Dtmp<="ZZZZZZZZ";

Ctmp<='0';

end if;

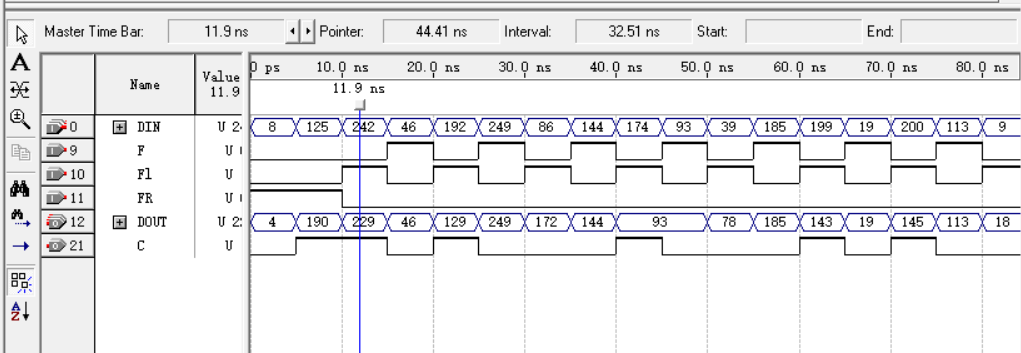
end process;

DOUT<=Dtmp;

C<=Ctmp;

end data;

波形仿真：



CZ触发器：

接口如图：

功能：

C和Z的值分别是跳转指令JC和JZ是否有效的一个判断条件：当C=1时，JC有效；Z=1时，JZ有效。C是逻辑运算的进位输出，包括加法的进位、减法的借位以及移位时被移出去的那一位(注意是移出去的那一位，不是最高位)。Z的值由加法和减法控制，当运算结果为0时，Z=1。

既然只有当执行以上提及操作时C和Z的值才发生改变，那么当不执行这些操作时，C和Z的值应该保持不变才对，所以我们需要设计一个触发器来判断c，z是否发生变化。

Vhdl代码如下：

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity CZjicunqi is

port ( CLK: in std\_logic;

C\_in, Z\_in: in std\_logic;

C, Z: out std\_logic);

end CZjicunqi;

architecture aCZIR of CZjicunqi is

signal cout : std\_logic:='0';

signal zout : std\_logic:='0';

begin

process(CLK)

begin

if(CLK'event and (CLK='0')) then

cout <= c\_in;

zout <= z\_in;

end if;

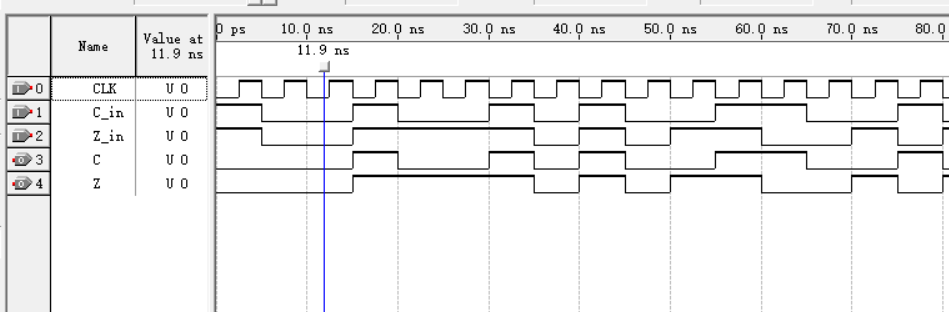
end process;

c <= cout;

z <= zout;

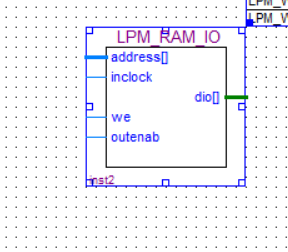
end aCZIR;

波形仿真：



Ram寄存器：

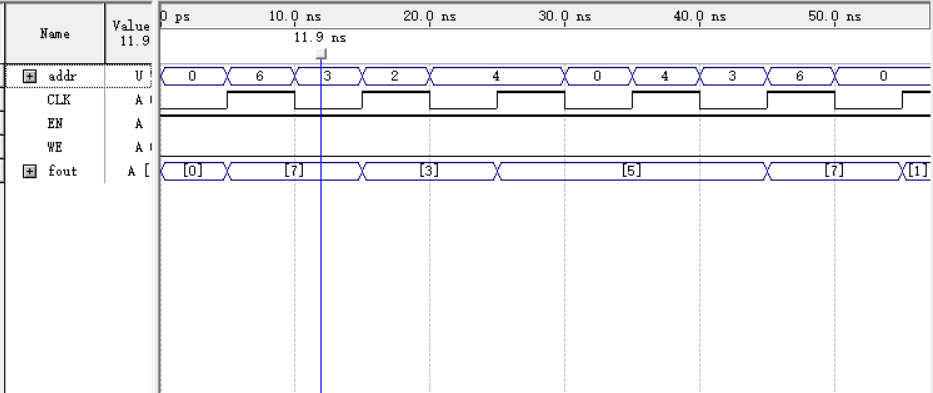
接口如图：



实现功能：在we和outenab的控制下，在时钟上升沿读取或者写入mif中的指令，we为1时写入，outenab为1时读取。

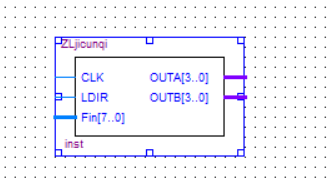
使用的是quartus自带的寄存器。

波形仿真：



指令寄存器：

接口如图：



实现功能：在LDIR的控制下载时钟下降沿把指令分解为OUTA（高四位）,OUTB（低四位）两个部分。

代码如下：

library ieee;

use ieee.std\_logic\_1164.all;

entity ZLjicunqi is

port(CLK:in std\_logic;

LDIR:in std\_logic;

Fin:in std\_logic\_vector(7 downto 0);

OUTA,OUTB:out std\_logic\_vector(3 downto 0));

end entity ZLjicunqi;

architecture data6 of ZLjicunqi is

signal ZL:std\_logic\_vector(7 downto 0);

begin

process(CLK,LDIR)

begin

if((CLK'event and CLK='0') and LDIR='1')then

ZL<=Fin;

end if;

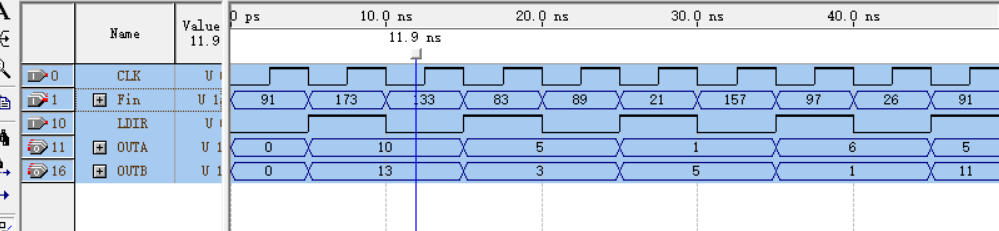
OUTA<=ZL(7 downto 4);

OUTB<=ZL(3 downto 0);

end process;

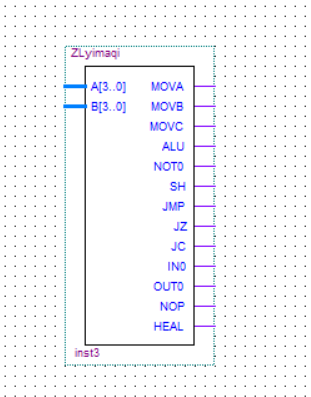
end data6;

波形仿真：

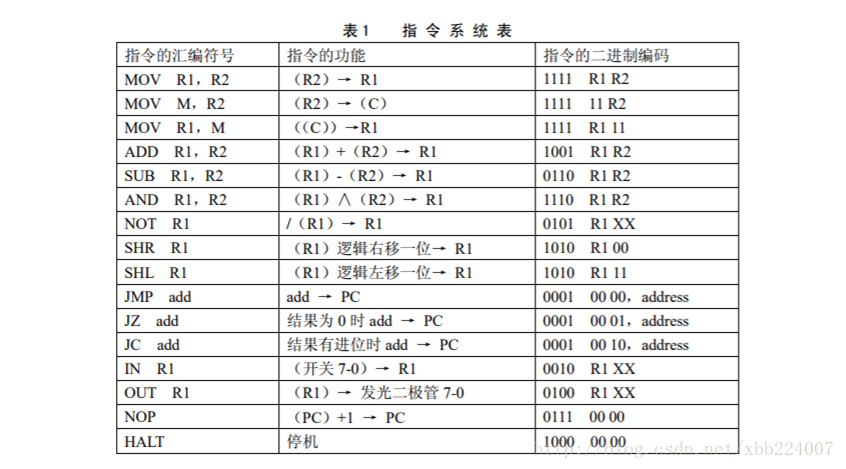


指令译码器：

接口如图：



实现功能：根据指令选择对应的要实现功能。初始化为0，对应的功能输出为‘1’。



指令的二进制编码前四位对应A[3..0]后四位对应B[3.0]。

代码如下：

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.std\_logic\_arith.all;

entity ZLyimaqi is

port(A :in std\_logic\_vector(3 downto 0);

B :in std\_logic\_vector(3 downto 0);

MOVA,MOVB,MOVC,ALU,NOT0,SH,JMP,

JZ,JC,IN0,OUT0,NOP,HEAL:

out std\_logic);

end entity ZLyimaqi;

architecture data7 of ZLyimaqi is

signal output:std\_logic\_vector(12 downto 0):="0000000000000";

begin

process (A,B)

begin

if A="1111" then

if(B(3 downto 2)="11") then output<="0000000000010";

elsif(B(1 downto 0)="11")then output<="0000000000100";

else output<="0000000000001";

end if;

elsif(A="1001"or A="0110"or A="1011")then output<="0000000001000";

elsif(A="0101")then output<="0000000010000";

elsif(A="1010")then output<="0000000100000";

elsif(A="0001")then

if(B(1 downto 0)="00")then output<="0000001000000";

elsif(B(1 downto 0)="01")then output<="0000010000000";

elsif(B(1 downto 0)="01")then output<="0000100000000";

end if;

elsif(A="0010")then output<="0001000000000";

elsif(A="0100")then output<="0010000000000";

elsif(A="0111")then output<="0100000000000";

elsif(A="1000")then output<="1000000000000";

else output<="0000000000000";

end if;

end process;

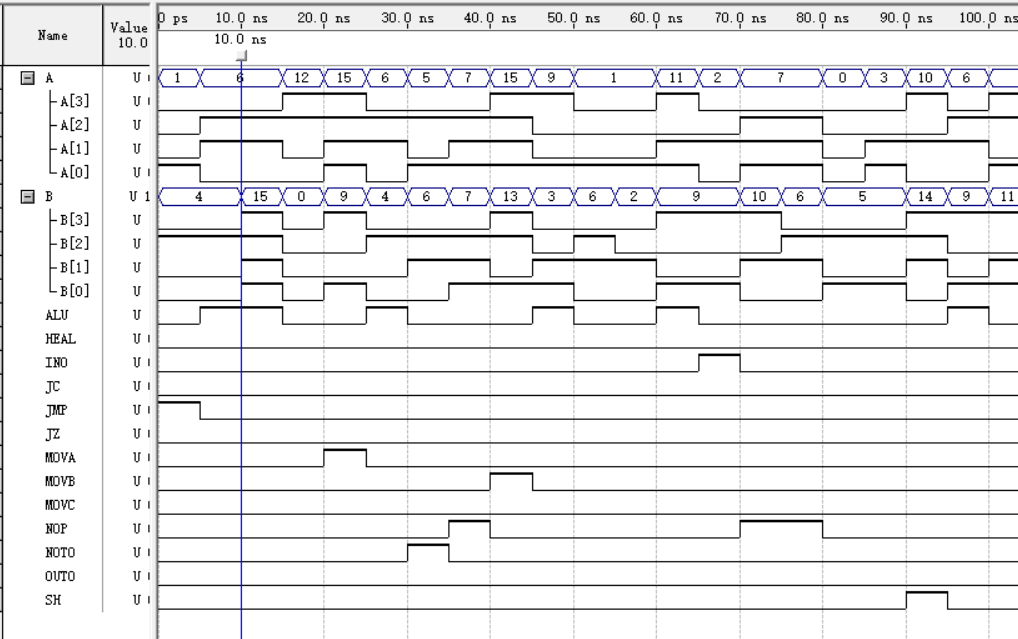
MOVA<=output(0);MOVB<=output(1);MOVC<=output(2);ALU<=output(3);

NOT0<=output(4);SH<=output(5);JMP<=output(6);JZ<=output(7);

JC<=output(8);IN0<=output(9);OUT0<=output(10);NOP<=output(11);HEAL<=output(12);

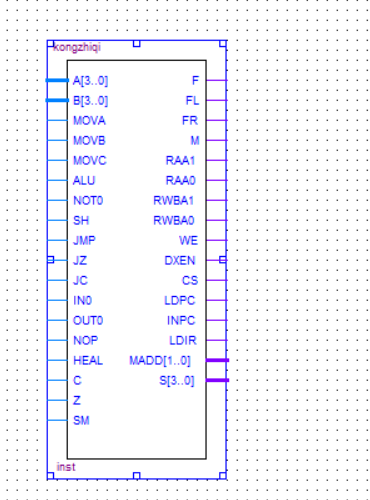
end architecture data7;

波形仿真：



控制器：

接口如图：



实现功能：

根据要实现的功能，选择对应的操作流程。具体体现在对一些元件的使能输入端口赋值使之能实现功能。

代码如下：

library ieee;

use ieee.std\_logic\_1164.all;

entity kongzhiqi is

port(A,B:in std\_logic\_vector(3 downto 0);

MOVA,MOVB,MOVC,ALU,NOT0,SH,JMP,JZ,

JC,IN0,OUT0,NOP,HEAL,C,Z,SM:in std\_logic;

F,FL,FR,M,RAA1,RAA0,RWBA1,RWBA0,WE:out std\_logic;

DXEN,CS,LDPC,INPC,LDIR:out std\_logic;

MADD:out std\_logic\_vector(1 downto 0);

S:out std\_logic\_vector(3 downto 0));

end kongzhiqi;

architecture data8 of kongzhiqi is

begin

process(A,B,MOVA,MOVB,MOVC,ALU,NOT0,SH,JMP,JZ,JC,IN0,OUT0,NOP,HEAL,C,Z,SM)

begin

F <= SM and (MOVA or MOVB or NOT0 or ALU) ;

if(SH='1')then

if(B(1 downto 0)="00")then FR<='1';FL<='0';

else FR<='0';FL<='1';

end if;

else FR<='0';FL<='0';

end if;

M<=ALU or NOT0;

RAA1<=B(1);RAA0<=B(0);RWBA1<=B(3);RWBA0<=B(2);

WE <= not (SM and (MOVA or MOVC or ALU or NOT0 or SH or IN0));

S<=A;

DXEN <= (SM and (MOVC or JMP or (JC and C) or (JZ and Z))) or (not SM);

CS<=SM and MOVB;

LDPC<=SM and (JMP or (JC and C) or (JZ and Z));

INPC <= (SM and (NOP or (JC and (not C)) or (JZ and (not Z)))) or (not SM);

LDIR<=not SM;

if(SM='0')then MADD<="00";

elsif(MOVB='1')then MADD<="10";

elsif(MOVC='1')then MADD<="01";

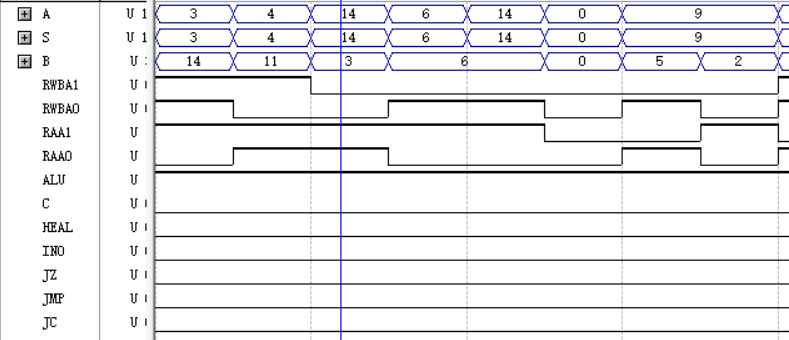
else MADD<="00";

end if;

end process;

end data8;

波形仿真：



1. 系统测试

4.1 测试环境

采用基于FPGA进行数字逻辑电路设计的方法。

采用的软件工具是QuartusII软件仿真平台，采用的硬件平台是Altera EP2C5T144C8的FPGA试验箱。

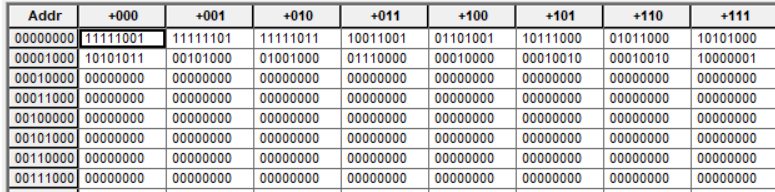
4.2 测试代码



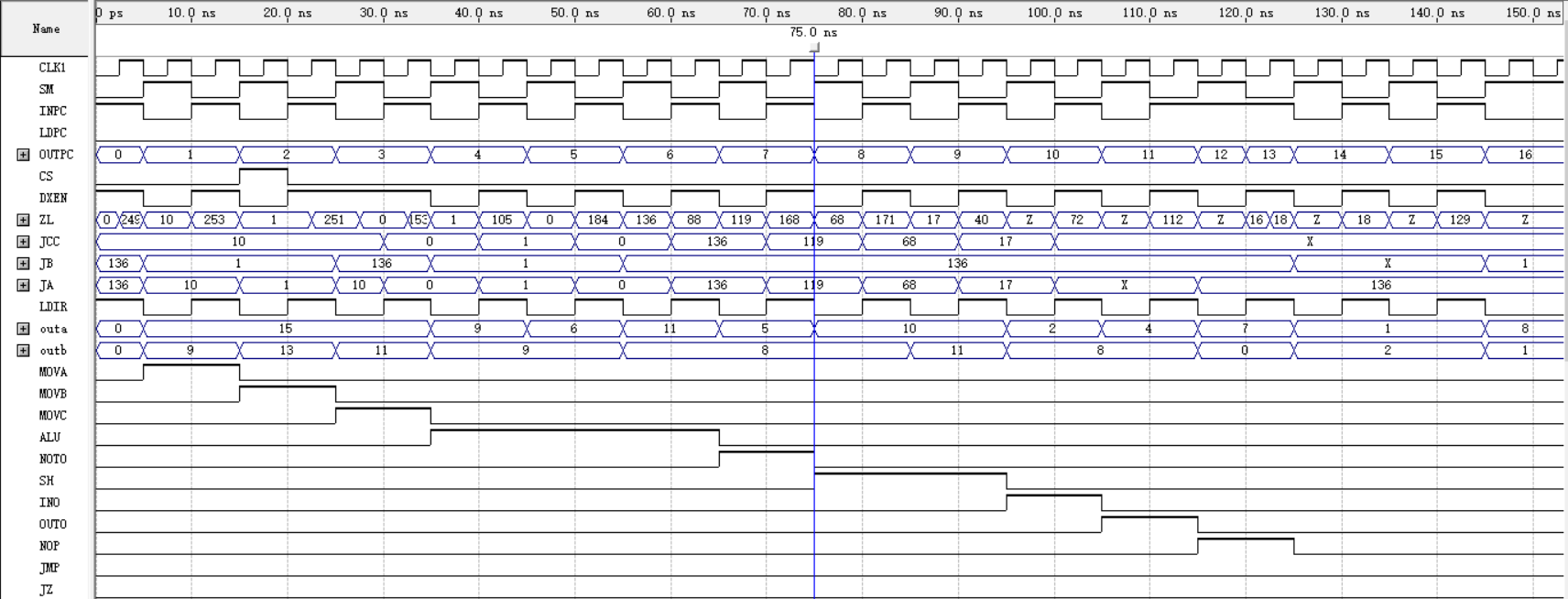


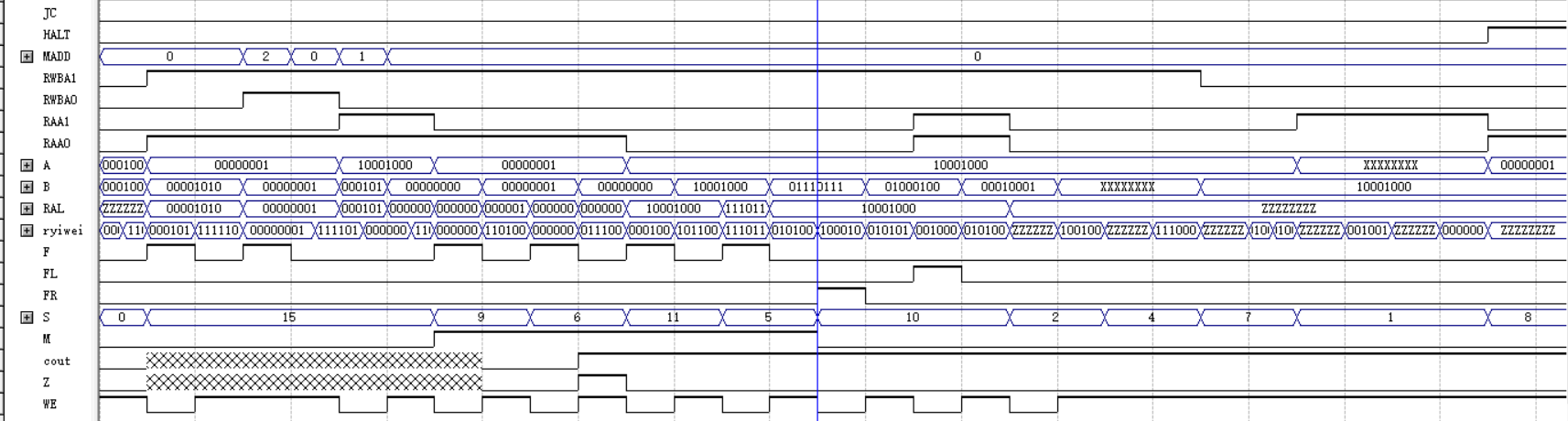
* 1. 测试结果

Mif文件：



波形文件：





其中jmp jc jz 指令未实现成功。其余指令实现成功。ALU中包含了加减或三种运算，具体区别由输出端口S判断，结果由RAL判断。SH中包含了左循环移位和右循环移位。具体区别由FL和FR体现，结构由cout和Z体现。JA,JB,JCC为寄存器r1，寄存器r2和寄存器c。

1. 总结

CPU的难点在于如何把各个模块互相连接起来和如何传递信息以及如何在做完整个CPU后检验CPU是否有错误。每一个元器件要实现的功能其实并不复杂。分别实现每个元件并不困难，但是一旦把8个元件组合在一块，要考虑到每个元器件之间数据的传输和连接方式，还要考虑是否受时钟影响以及如何在总线上面不冲突。不是写好一部分的元件最后一连就可以，而是对每个元件的草稿打完后，在连接的时刻考虑到数据传输问题再修改每块元件的内容。因此整个CPU的耗时非常长，做出来的东西也非常的繁杂。这种时候只能慢慢理清楚过程，一定要有耐心。千万不可急功近利。心得体会就是没有什么事情是没法完成的，只要能静下心能用心去钻研，任何事情都是能够解决的。通过这个实验，让我对计算机系统的组成的工作原理有了一定的了解，做实验的过程阅读了很多资料，也请教了很多同学，让我对VHDL这门语言和QuartusII这个软件有了深入的了解。更对课本中的知识和CPU有了更深一层的理解。