### ****一、设计内容****

本次课程设计的主要内容是利用设计工具，设计一个模型计算机。

CPU 的主要功能是执行指令，控制完成计算机的各项操作，包括运算操作、传送操作、输入/输出操作等。作为模型计算机设计，将重点放在寄存器级，采取较简单的组成模式，以尽量简洁的设计帮助学生掌握CPU 的基本原理。 此次设计CPU就是为了了解CPU运行的原理，从而完成从指令系统到CPU的设计，并且通过仿真对CPU设计进行正确性评定。

（1）融会贯通本课程各章节的内容，通过知识的综合运用，加深对计算机系统各功能部件的工作原理及相互联系的认识，加深计算机工作中“时间-空间”概念的理解，从而清晰地建立计算机的整机概念；

（2）学习设计和调试计算机的基本步骤和方法，提高使用Quartus等软件仿真工具和集成电路的基本技能；

（3）培养科学研究的独立工作能力，取得工程设计与组装调试的实践和经验。

### ****二、设计要求：****

设计一台多种指令的模型计算机，用VHDL 语言完成设计并调试成功。

##### **1. 数据格式与指令系统**

（1）数据格式：

   数据的长度为8位，数据采用补码格式，相对于十进制数范围是:-27<N<27-1

（2）指令格式

|  |  |  |
| --- | --- | --- |
| **指令** | **操作码** | **功能** |
| **STORE X** | 01H | ACC→[X] |
| **LOAD X** | 02H | [X]→ACC |
| **ADD X** | 03H | ACC+[X]→ACC |
| **SUB X** | 04H | ACC-[X]→ACC |
| **JMPGZ X** | 05H | IF ACC>0 THEN X→PC ELSE PC+1→PC |
| **AND X** | 06H | ACC and [X]→ACC |
| **OR X** | 07H | ACC or [X]→ACC |
| **NOT X** | 08H | Not [X]→ACC |
| **SHIFTR X** | 09H | SHIFL ACC to RIGHT 1 bit, Logic Shift |
| **SHIFTL X** | 0AH | SHIFT ACC to LEFT 1 bit, Logic Shift |
| **MPY X** | 0BH | ACC×[X]→ACC |
| **HALT** | 0CH | HALT A PROGRAM |

#### **2.设计原理与电路图**

设计代码：

ALU

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity ALU is

port( clk,reset,ACCclear:in std\_logic;

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

     BR       :in std\_logic\_vector(15 downto 0);

     PCjmp    :out std\_logic;

     ACC      :buffer std\_logic\_vector(15 downto 0));

end ALU;

architecture behave of ALU is

begin

  process(clk)

  begin

    if(clk'event and clk='0')then

     if

reset='0' then ACC<=x"0000";

     else if

ACCclear='1' then  ACC<=x"0000";

end if;

       if aluCONTR="0011" then   ACC<=BR+ACC;

end if;      --ADD

       if aluCONTR="0100" then   ACC<=ACC-BR;

end if;      --SUB

       if aluCONTR="0110" then   ACC<=ACC and BR;

end if;      --AND

       if aluCONTR="0111" then   ACC<=ACC or BR;

end if;       --OR

       if aluCONTR="1000" then   ACC<=not ACC;

end if;       --NOT

       if aluCONTR="1001" then                                   --SRR

          ACC(14 downto 0)<=ACC(15 downto 1);    ACC(15)<='0';

       end if;

       if aluCONTR="1010" then                                   --SRL

          ACC(15 downto 1)<=ACC(14 downto 0);     ACC(0)<='0';

       end if;

       if aluCONTR="1011" then   ACC<=ACC(15 downto 0)\*BR;       end if;     --MPY

      end if;

     end if;

     if ACC>0 then PCjmp<='1';

     else PCjmp<='0';

     end if;

  end process;

end behave;

BR

BR是ALU的输入，它保持ALU的其他操作数。在该设计中，BR有16位。

library ieee;

use ieee.std\_logic\_1164.all;

entity BR is

port( MBR\_BRc:in std\_logic;

     MBR\_BR:in std\_logic\_vector(15 downto 0);

     BRout:out std\_logic\_vector(15 downto 0));

end BR;

architecture behave of BR is

begin

  process(MBR\_BRc)

  begin

       if MBR\_BRc='1' then     BRout<=MBR\_BR;     end if;

  end process;

end behave;

CAR

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity CAR is

port( clk,reset :in std\_logic;

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

     CAR,OP   :in std\_logic\_vector(7 downto 0);

     CARout:buffer std\_logic\_vector(7 downto 0));

end CAR;

architecture behave of CAR is

begin

  process(clk)

  begin

    if(clk'event and clk='1')then

     if reset='1' then

      if CARc="1000" then         CARout<="00000000";       end if;

      if CARc="0100" then         CARout<=OP+CARout;      end if;

      if CARc="0010" then       CARout<=CAR;              end if;

      if CARc="0001" then       CARout<=CARout+1;         end if;

     else CARout<="00000000";

     end if;

    end if;

 end process;

end behave;

CONTROLR

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity CONTROLR is

port(

     control :in std\_logic\_vector(31 downto 0);

     R,W, RW, PCc1,PCinc,PCc3:out std\_logic;

     ACCclear,MBR\_MARc,PC\_MARc:out std\_logic;

     ACC\_MBRc,MBR\_OPc,MBR\_BRc:out std\_logic;

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

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

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

end CONTROLR;

architecture behave of CONTROLR is

begin

  process(control)

  begin

       CAR<=control(7 downto 0);

       PCc1<=control(8);

       PCinc<=control(9);

       PCc3<=control(10);

       ACCclear<=control(11);

       CONTRout<=control(15 downto 12);

       R<=control(16);

       W<=control(17);

       MBR\_MARc<=control(18);

       PC\_MARc<=control(19);

       ACC\_MBRc<=control(20);

       MBR\_OPc<=control(21);

       MBR\_BRc<=control(22);

       CARc<=control(26 downto 23);

       RW<=control(17);

end process;

end behave;

IR

IR包含指令的操作码部分。在该设计中，IR有8位。

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity IR is

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

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

end IR;

architecture behave of IR is

begin

    IRout<=opcode;

end behave;

MAR

MAR包含要从内存中读取或写入到内存中的字的内存位置。读操作被表示为CPU从内存中读，写操作被表示为CPU向内存中写。在该设计中，MAR有8位来访问256个内存地址中的一个。

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity MAR is

port( clk,PC\_MARc,MBR\_MARc:in std\_logic;

     PC,MBR\_MAR:in std\_logic\_vector(7 downto 0);

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

end MAR;

architecture behave of MAR is

begin

  process(clk)

  begin

    if(clk'event and clk='1')then

       if PC\_MARc='1' then      MARout<=PC;           end if;

       if MBR\_MARc='1' then    MARout<=MBR\_MAR;   end if;

    end if;

  end process;

end behave;

MBR

MBR包含要存储在内存中的值或从内存中读取的最后一个值。MBR被连接到系统总线的地址线。在该设计中，MBR有16位。

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity MBR is

port( clk, reset, MBR\_OPc, ACC\_MBRc,R,W:in std\_logic;

     ACC\_MBR :in std\_logic\_vector(15 downto 0);

     RAM\_MBR :in std\_logic\_vector(15 downto 0);

     MBR\_RAM :out std\_logic\_vector(15 downto 0);

     MBR\_BR  :out std\_logic\_vector(15 downto 0);

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

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

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

end MBR;

architecture behave of MBR is

begin

  process(clk)

  variable temp:std\_logic\_vector(15 downto 0);

  begin

    if(clk'event and clk='0')then

     if reset='1' then

       if ACC\_MBRc='1' then  temp:=ACC\_MBR;   end if;

       if R='1' then    MBR\_BR<=RAM\_MBR;     end if;

       if W='1' then      MBR\_RAM<=temp;      end if;

       MBR\_MAR<=RAM\_MBR(7 downto 0);

       MBR\_PC<=RAM\_MBR(7 downto 0);

       if MBR\_OPc='1' then   MBR\_OP<=RAM\_MBR(15 downto 8);    end if;

      else MBR\_BR<=x"0000";

           MBR\_MAR<="00000000";

           MBR\_OP<="00000000";

           MBR\_PC<="00000000";

      end if;

    end if;

  end process;

end behave;

PC

PC保持在程序中使用的指令的跟踪。在该设计中，PC有8位。

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity PC is

port( clk,PCjmp,PCc1,PCinc,PCc3,reset:in std\_logic;

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

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

     PCout      :buffer std\_logic\_vector(7 downto 0));

end PC;

architecture behave of PC is

begin

  process(clk)

  begin

    if(clk'event and clk='0')then

      if reset='1' then

         if CONTRalu="0101" then

           if PCjmp='1' then            PCout<=MBR\_PC;

            elsif PCjmp='0' then PCout<=PCout+1;

           end if;

         end if;

         if PCc1='1' then          PCout<="00000000";       end if;

         if PCinc='1' then          PCout<=PCout+1;         end if;

         if PCc3='1' then          PCout<=MBR\_PC;         end if;

      else PCout<="00000000";

      end if;

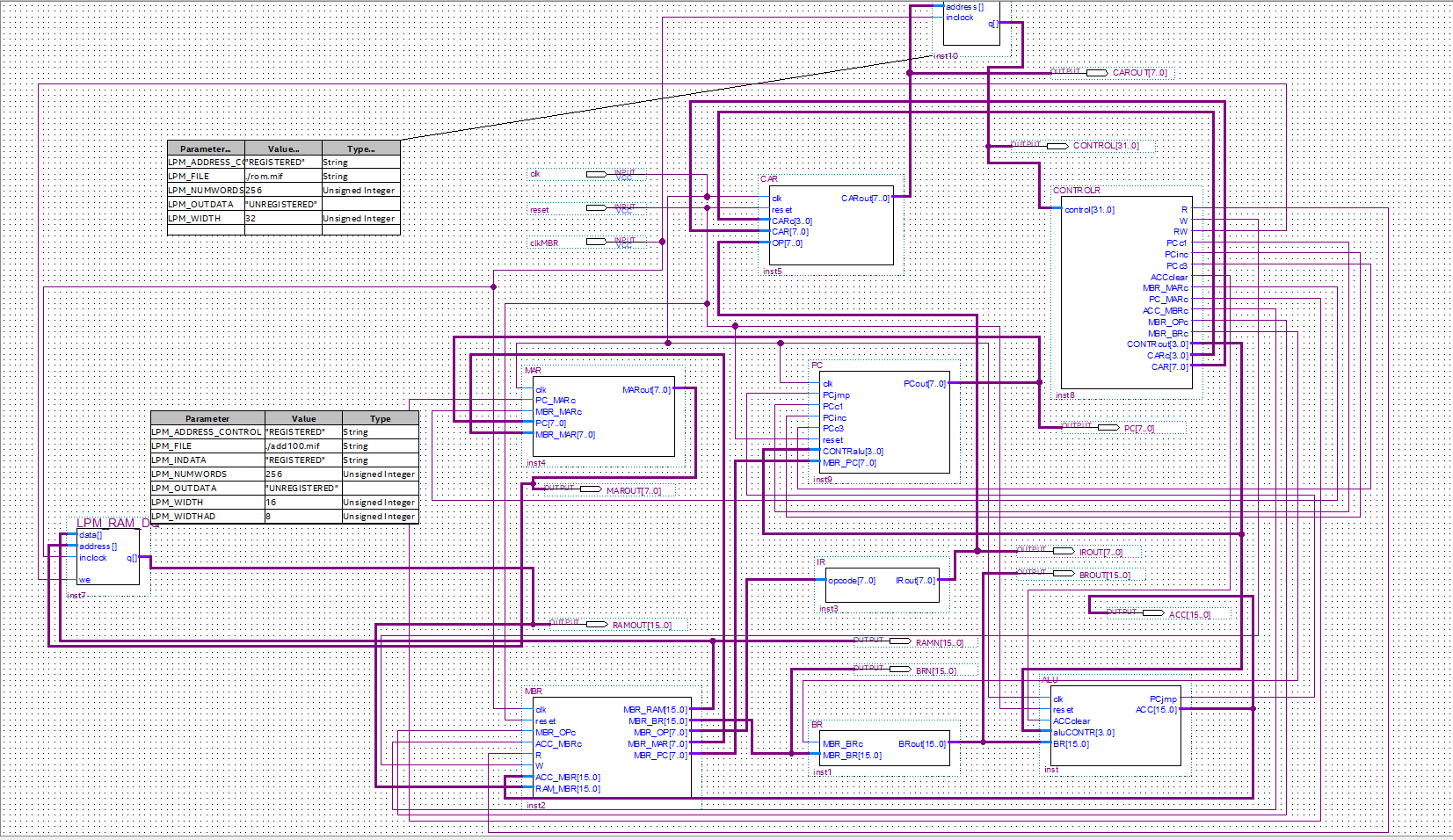
    end if;

  end process;

end behave;

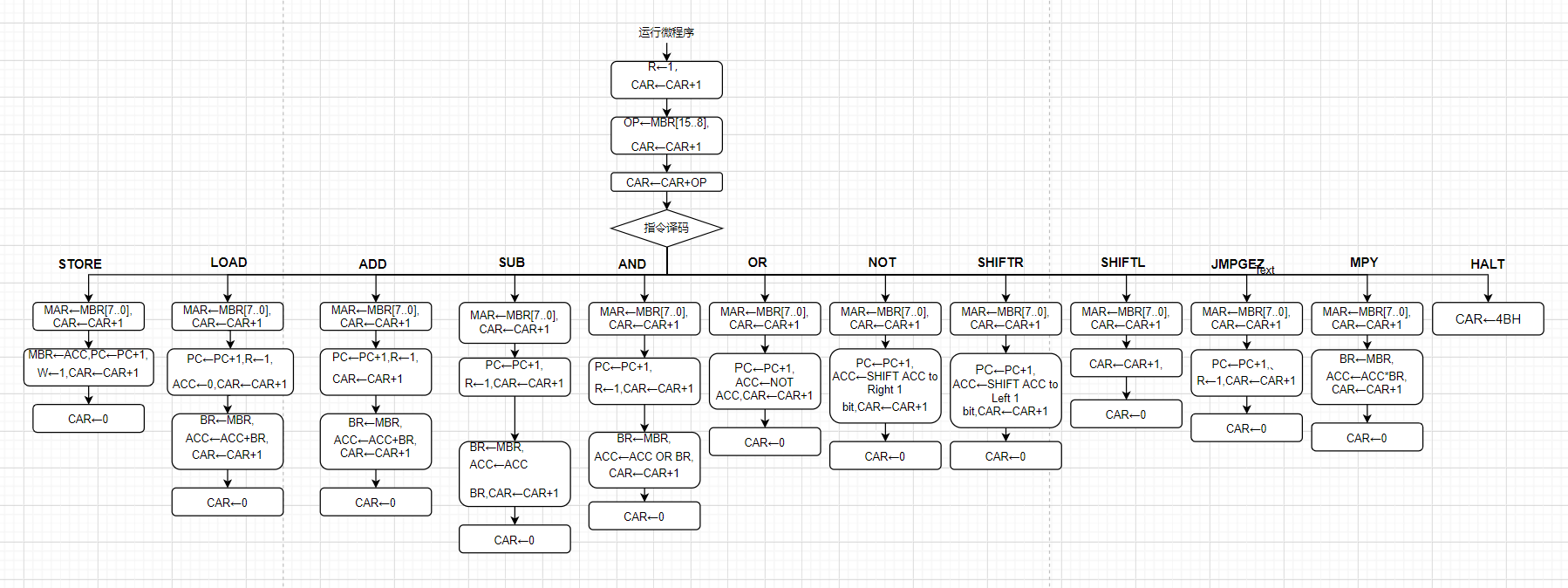
Rom

连线图：



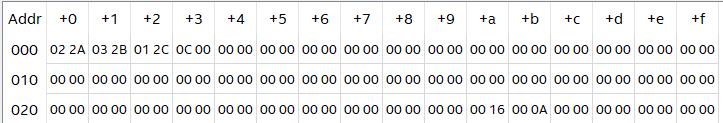
#### **3.微程序流程图、代码表**

（1） 微程序流程图

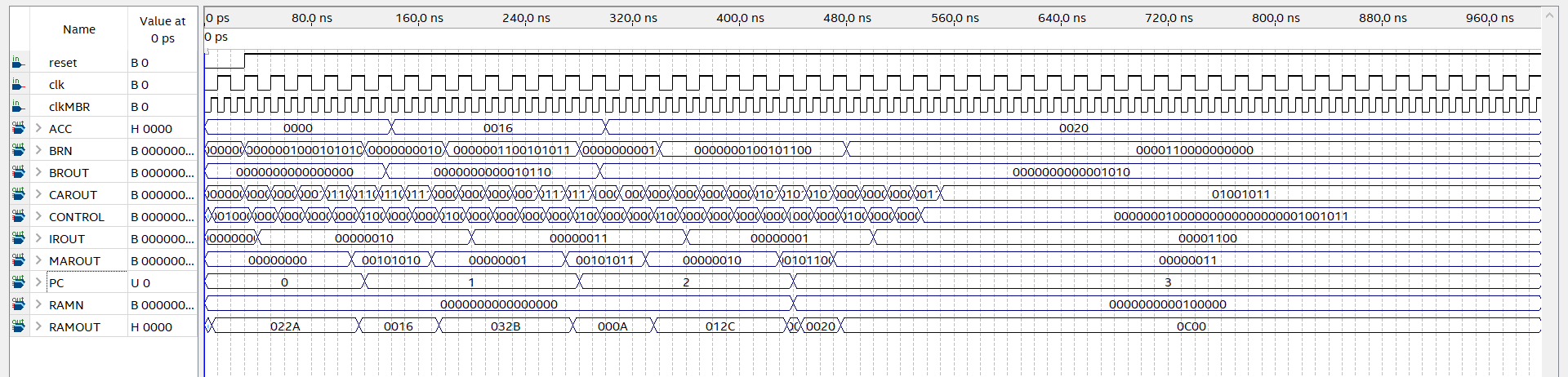


#### **4.系统调试情况**

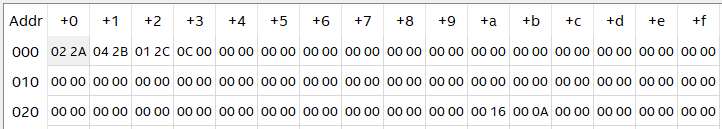
（1）ram\_add



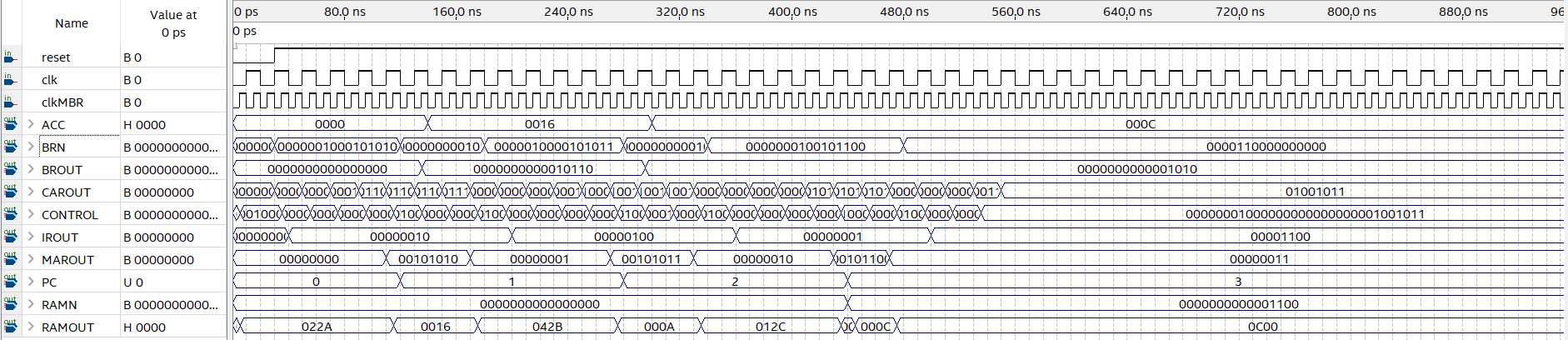
仿真波形图：



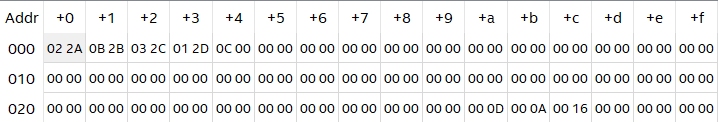
（2）rom\_sub



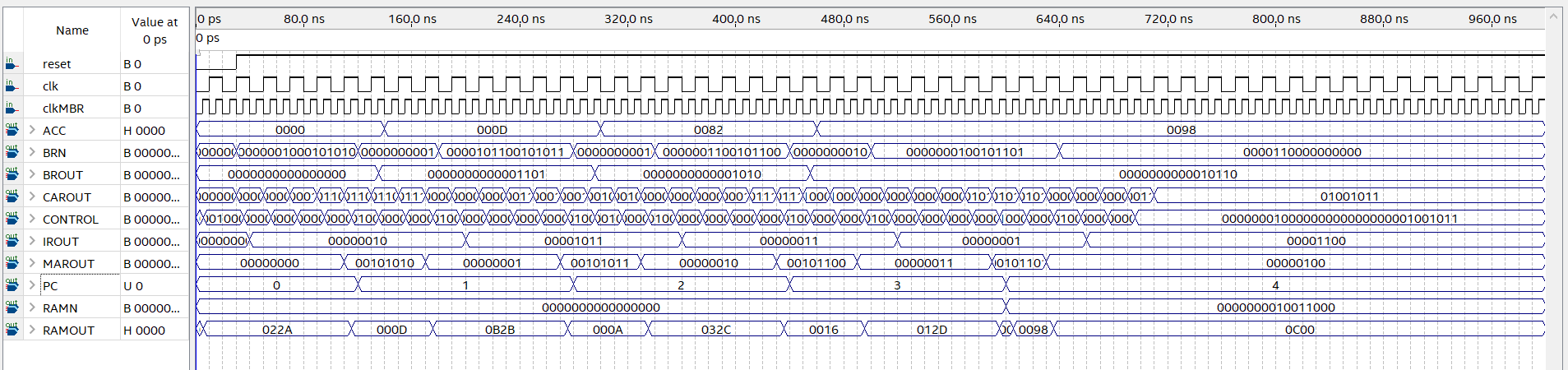
仿真波形图：



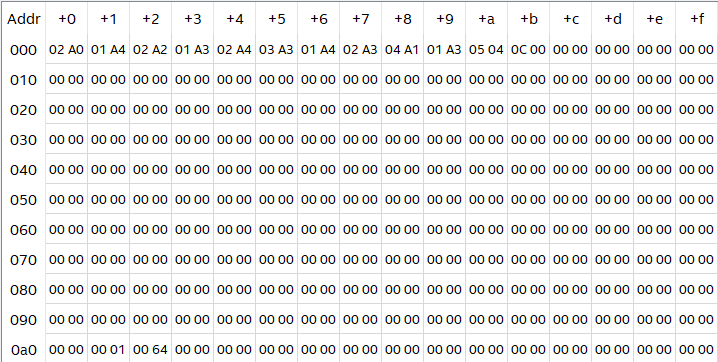
（3）ram\_mpy



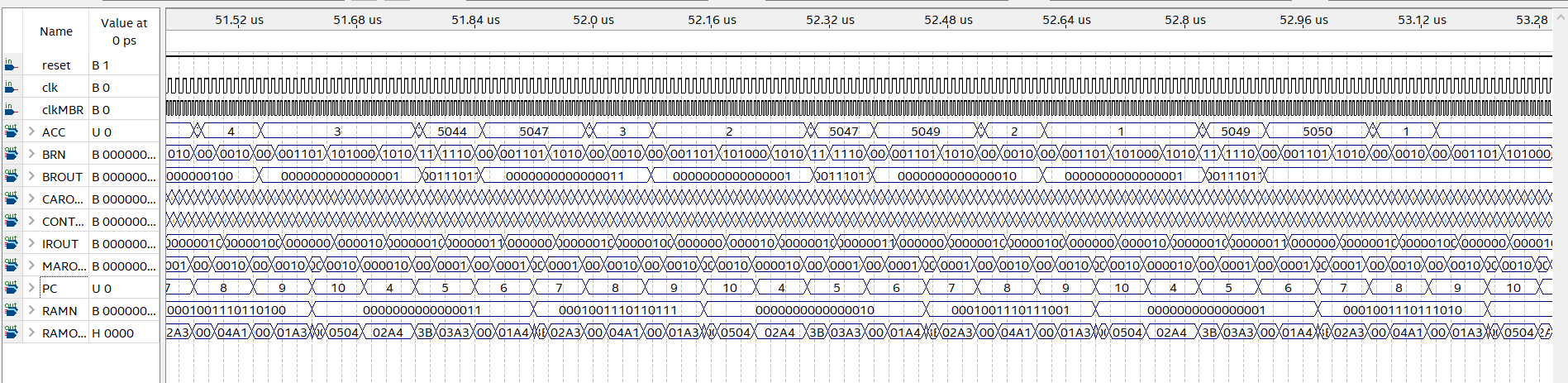
仿真波形图：



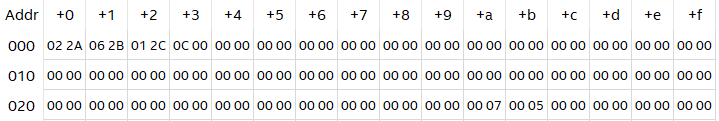
（4）ram\_sum from 1 to 100



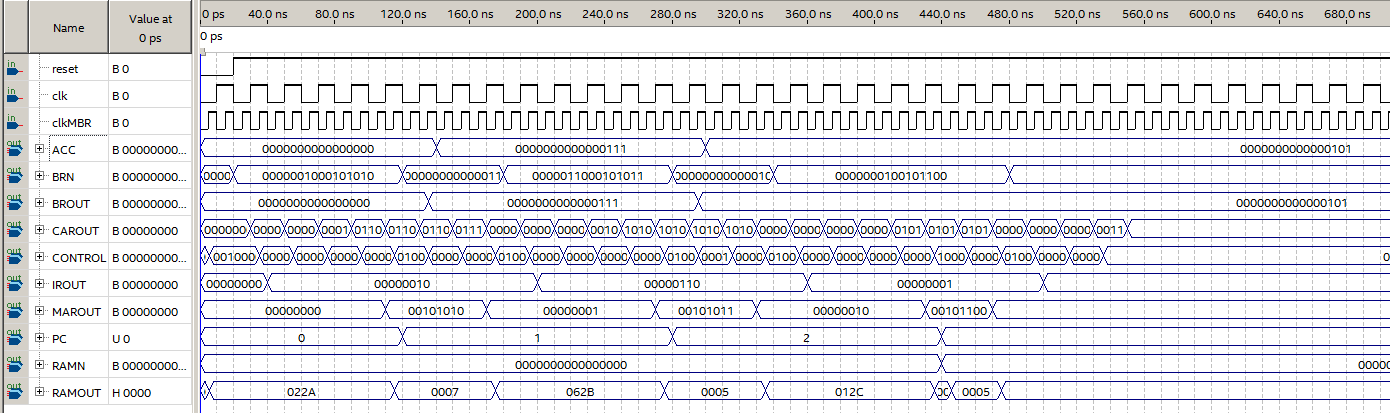
仿真波形图：



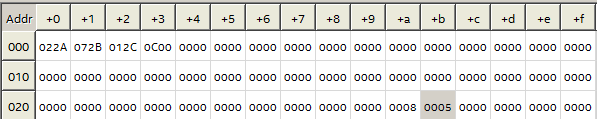
(5)ram\_and



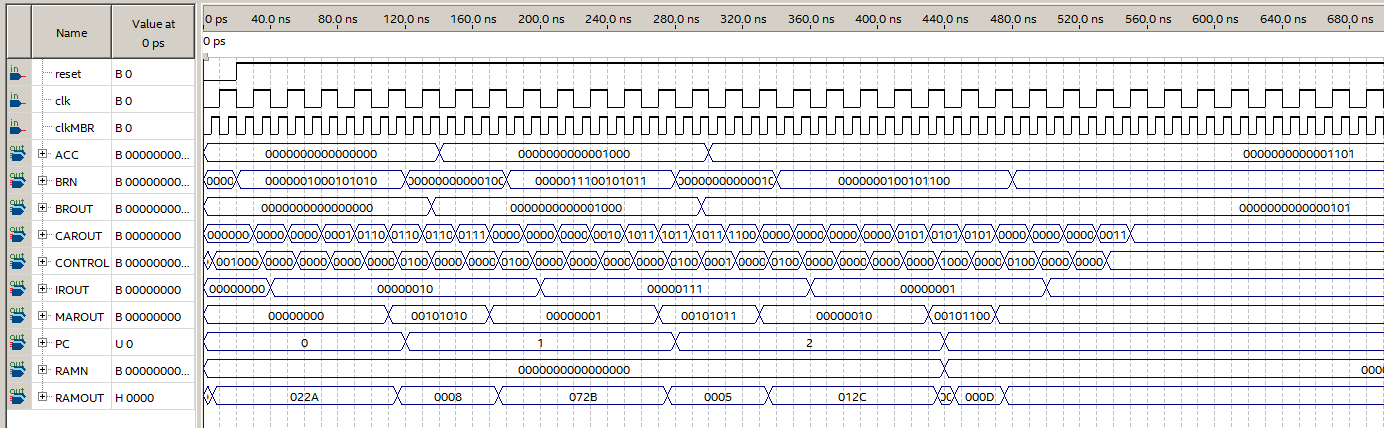
仿真波形图：



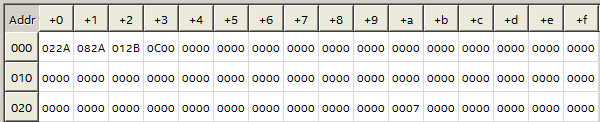
(6)ram\_or

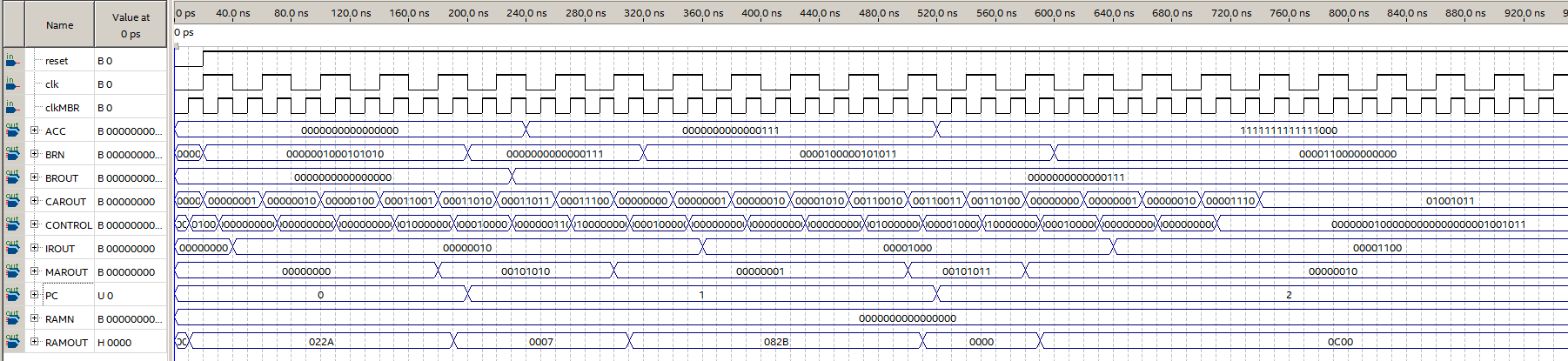


仿真波形图：

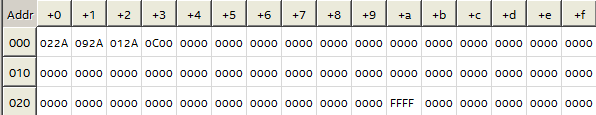


（7）ram\_not

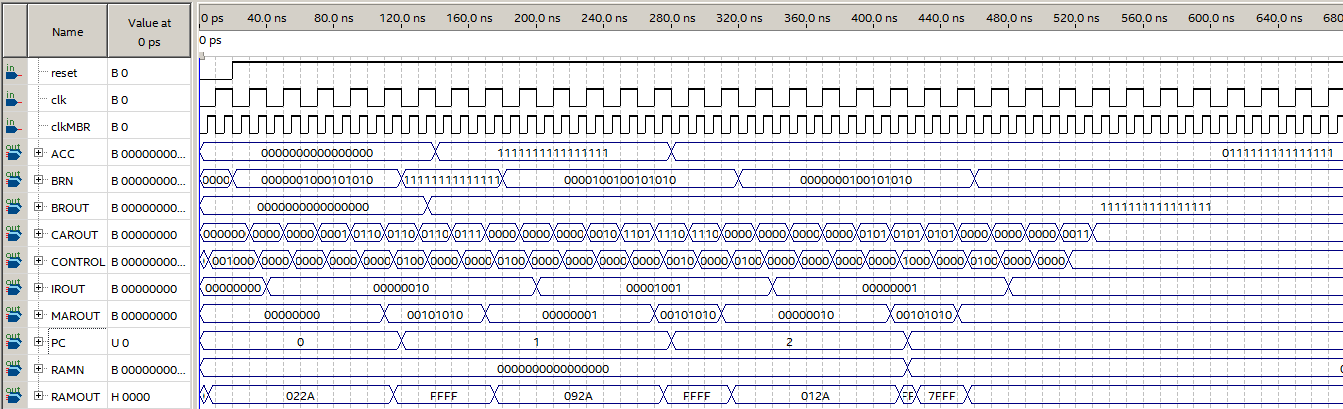


仿真波形图

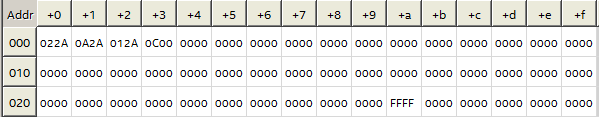
（8）ram\_shiftr



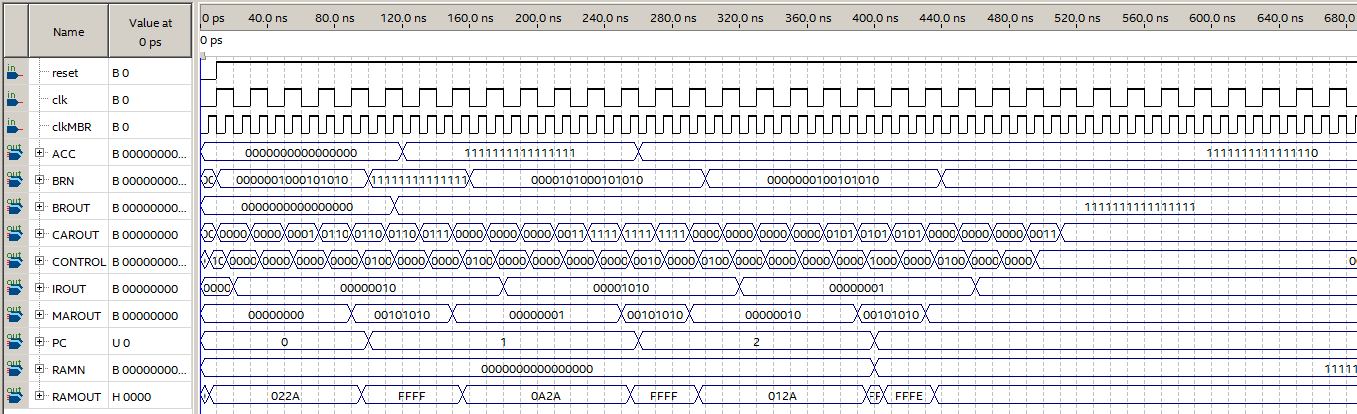
仿真波形图：



（9）ram\_shiftl



仿真波形图：



#### **五、课程设计的收获及体会**

#### **六、参考资料**

[1] 潘松,潘明. 现代计算机组成原理[M]. 北京：科学出版社,2007.

[2] 陈华光. 计算机组成原理[M].北京：机械工业出版社，2004：5-10.

[3] 侯伯亨等. VHDL硬件描述语言与数字逻辑电路设计（修订版）[M]，西安：西安电子科技大学出版社，1999：75-78.

[4] 杨东旭等.计算机组成原理实用教程[M]. 北京：清华大学出版社, 2001：120-122.