实验四 模型机时序部件的实现

班 级 计科 1907　　姓名 杨杰　　学号 201908010705

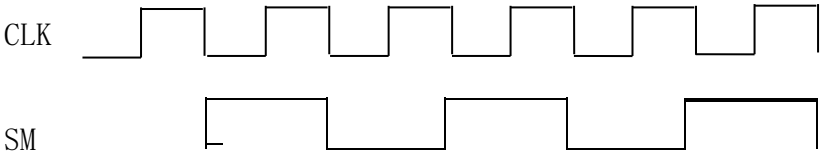
一、实验目的

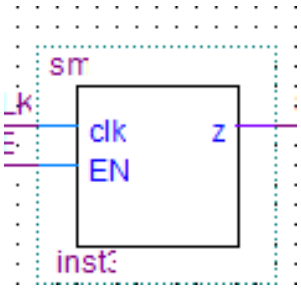
1. 熟悉计数器、寄存器和 RAM 的工作原理。
2. 了解模型机中 SM 的作用。
3. 学会使用 VHDL 语言设计时序电路。

二、实验背景

1. SM

模型机中所有指令的执行都是一个周期完成取指令，一个周期执行指令。如何区分当前周期是取指令还是执行呢？这就需要SM。当SM=0时，该周期完成取指令，当SM=1时，该周期执行指令。SM的功能及接口如下：



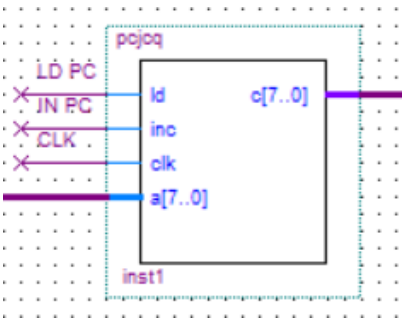


CLK	EN	功能
⌐	1	SM⇦SM取反
¬	0	SM不变

2. 指令计数器PC

指令计数器PC保存的是下一条指令在RAM中存放的地址。

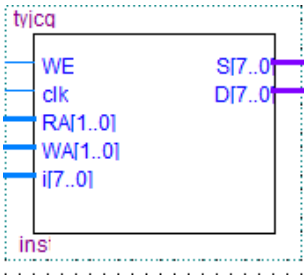
CPU执行一条指令，根据PC中存放的指令地址，将指令从RAM读出写入指令寄存器IR中，此过程称为“取指令”，与此同时，PC中的地址自动加1，指向下一条指令在RAM中的存放地址。跳转指令如JMP、JZ、JC让程序跳转至指定地址去执行，这时PC需要装载跳转地址。因此，指令计数器PC的接口及功能如下：



CLK	IN PC	LD PC	功能
⌐¯	1	0	c[7..0]中数据自加1
¯⌐	0	1	a[7..0]向入c[7..0]

3. 通用寄存器组

寄存器是CPU内的重要组成部分，模型机的通用寄存器组包含3个8位寄存器A、B、C，实现对此3个寄存器的读写操作。其接口及功能如下：



操作	CLK	WE	RAA[1..0]	RWBA[1..0]	功能
读			00或01或10	00或01或10	根据RAA[1..0]的值从A,B,C中选择一个寄存器的值由S口输出 根据RWBA[1..0]的值从A,B,C中选择一个寄存器的值由D口输出
写	⌐¯	0	XX	00或01或10	控制信号WE为0，根据RWBA[1..0]的值，在时钟下降沿将外部输入写入A,B,C三个寄存器中的某个寄存器。

4. RAM

半导体存储器的种类很多，从功能上可以分为只读存储器ROM和随机存储器RAM两大类。

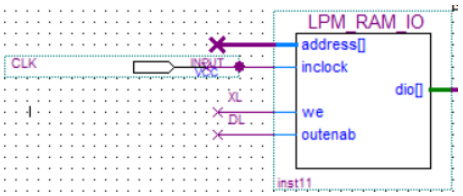
随机存储器RAM是与CPU直接交换数据的内部存储器，也叫主存（内存）。它

可以随时读写，而且速度很快，通常作为操作系统或其他正在运行中的程序的临时数据存储媒介。

存储元是构成存储器的存储介质，它可存储一个二进制位。由若干个存储元组成一个存储单元，然后再由许多存储单元组成一个存储器。一个存储器包含许多存储单元，每个存储单元可存放一个字节。每个存储单元的位置都有一个编号， 即地址，一般用十六进制表示。一个存储器中所有存储单元可存放数据的总和称为它的存储容量。比如，一个存储器的地址码由8位二进制数（即2位十六进制数） 组成，则可表示2的8次方，即256个存储单元地址，每个存储单元存放一个字节， 则该存储器的存储容量为256×8，即2Kbit。

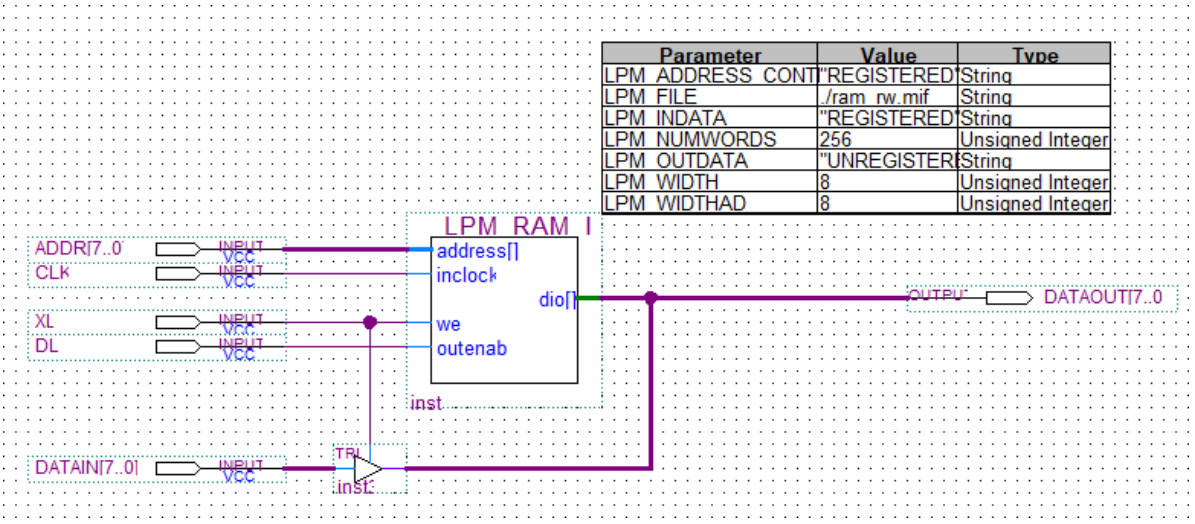
本实验可采用Quartus中已有的参数化模块来定制RAM功能，在【Symbol】元件库的【megafunctions】|【storage】中选择LPM_RAM_IO，创建RAM时加载初始化数据文件，初始化文件的创建过程是：【File】|【New】|【Memory Initialization File】。

LPM_RAM_IO的符号及功能如下：



CLK	We（XL）	outenab(DL)	功能
⎍	0	0	Dio<=高阻态Z
⎍	1	0	Dio的数据写入address所指定的存储单元
⎍	0	1	address所指定的存储单元数据从dio输出

ADDR[7..0]指定访问RAM的地址，加载时钟CLK，XL为1，将外部输入DATAIN[7..0]写入RAM的对应存储单元。不改变ADDR[7..0]的值，这时DL为1，读取RAM，查看DATAOUT[7..0]中的输出是否跟前面写入的数据是否一致，从而学习对RAM的读写操作。RAM读写操作的参考电路如图所示：



三、实验内容

1. 用 VHDL 语言设计 SM;
2. 用 VHDL 语言设计一个 8 位的指令计数器 PC;
3. 用 VHDL 语言设计 3 个 8 位寄存器组成的寄存器组，实现读写操作。
4. 用 LPM_RAM_IO 定制一个 256*8 的 RAM，实现对 RAM 的读写操作;

四、实验方法

1、

实验方法

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 Quartus II。

2、

实验步骤

1、 新建，编写源代码。

(1). 选择保存项和芯片类型：【File】-【new project wizard】-【next】-【next】（-【properties】（type=AHDL）-【next】（family=FLEX10K; name=EPF10K10TI144-4）-【next】-【finish】

(2). 新建：【file】-【new】（第二个 AHDL File）-【OK】

2、写好源代码，保存文件

3、编译与调试。确定源代码文件为当前工程文件，点击【processing】-【start compilation】进行文件编译，编译成功。

4、波形仿真及验证。新建一个 vector waveform file。

5、时序仿真或功能仿真。

6、查看 RTL Viewer:【Tools】-【netlist viewer】-【RTL viewer】。

五、实验过程

SM:

1、 编译过程

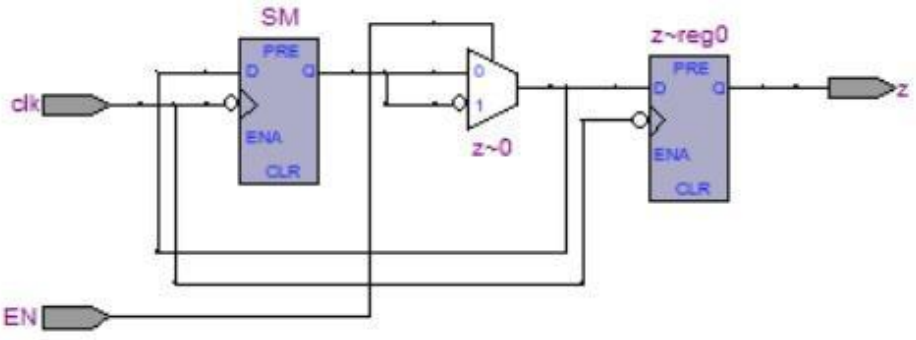
a）源代码如图（VHDL 设计）

```
library ieee;
use ieee.std_logic_1164.all;
entity SM is
  port(clk,EN:in std_logic;
       z:out std_logic);
end SM;
architecture SM_arc of SM is
signal SM:std_logic:='0';
  begin
  process(EN,clk)
    begin
      if(clk' event and clk='0') then
        if(EN='1') then
          z<=not SM;
          SM<=not SM;
        else
          z<=SM;
        end if;
      end if;
  end process;
end SM_arc;
```

b)编译、调试过程

编译成功

c) RTL 视图

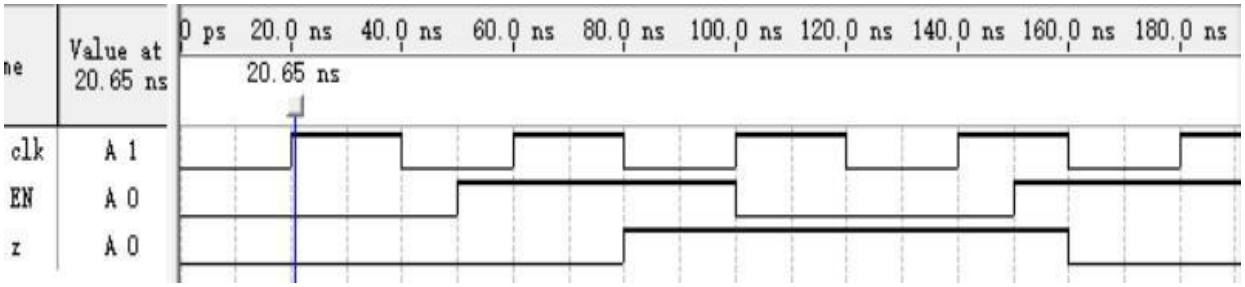


2、 功能仿真

a) 功能仿真过程

做好上述步骤后，在【processing】中选择【simulator tool】-【generate functional simulation netlist】-【start simulation】

b)功能仿真图



b) 结果分析及结论

适中下降沿时：

当 EN=0 时，SM=0，不变；

当 EN=1 时，SM=1，取反；

仿真结果正确。

PC：

3、 编译过程

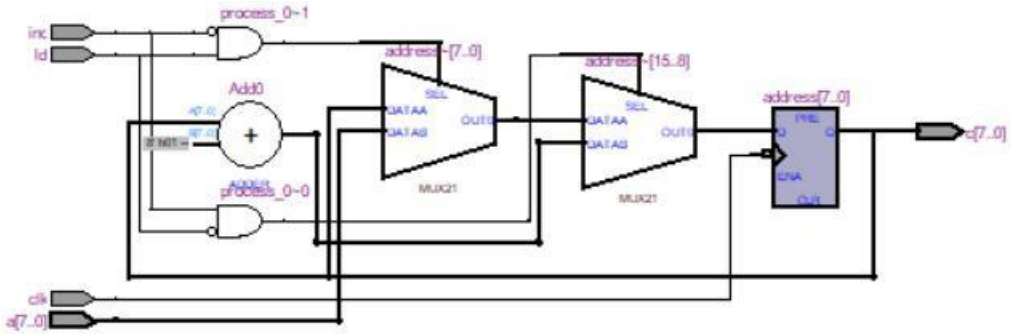
a）源代码如图（VHDL 设计）

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity PC is
  port(ld,inc,clk:in std_logic;
       a:in std_logic_vector(7 downto 0);
       c:out std_logic_vector(7 downto 0)
       );
end PC;
architecture bhv of PC is
signal address:std_logic_vector(7 downto 0):="00000000";
begin
  process(clk)
    begin
      if(clk' event and clk='0') then
        if(inc='1' and ld='0') then
          address<=address + "00000001";
        elsif(inc='0' and ld='1') then
          address<=a;
        end if;
      end if;
  end process;
  c<=address;
end architecture;
```

b)编译、调试过程

编译成功

c) RTL 视图

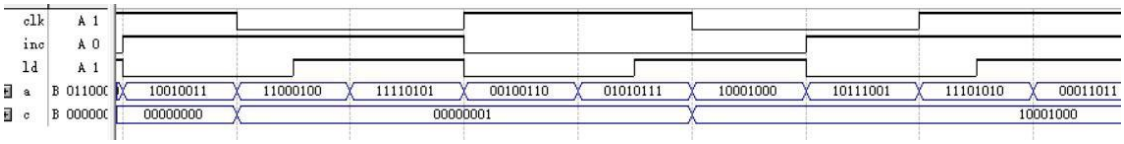


4、 功能仿真

c) 功能仿真过程

做好上述步骤后，在【processing】中选择【simulator tool】-【generate functional simulation netlist】-【start simulation】

b)功能仿真图



d) 结果分析及结论

当时钟下降沿时：

当 Inc=0，ld=1，a=11000100 时，c 自动加一，C=00000001;

当 Inc=1，ld=0，a=10001000 时，a 的值写向 c,C=10001000;

其余情况 c 保持上一次操作的值不变。

仿真结果正确。

寄存器组：

5、 编译过程

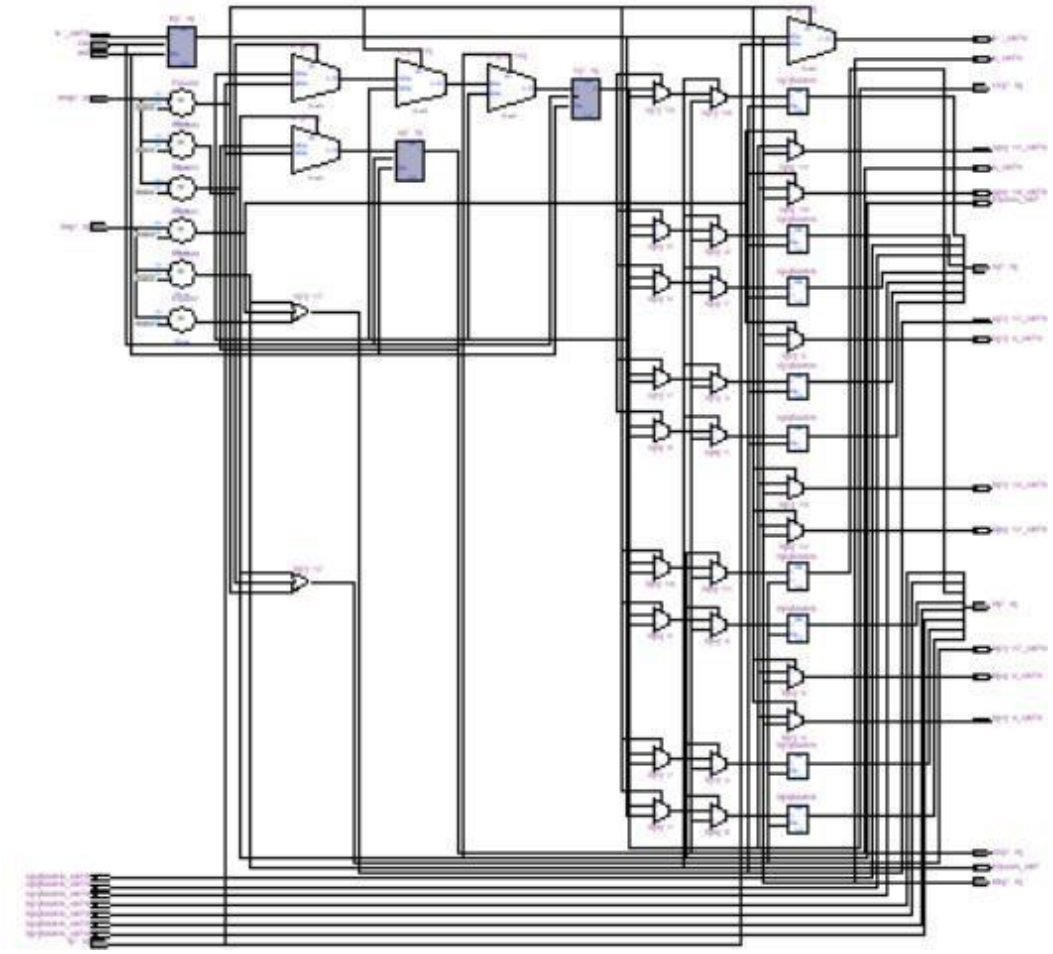
a）源代码如图（VHDL 设计）

```
library ieee;
use ieee.std_logic_1164.all;
entity SRG_Group is
  port(WE,clk:in std_logic;
       RA,WA:in std_logic_vector(1 downto 0);
       i:in std_logic_vector(7 downto 0);
       S,D:out std_logic_vector(7 downto 0);
       aa,bb,cc:out std_logic_vector(7 downto 0):="00000000"
      );
end SRG_Group;
architecture bhv of SRG_Group is
signal a,b,c:std_logic_vector(7 downto 0):="00000000";
begin
  process(clk,WE,RA,WA)
    begin
      if(WE='0' and falling_edge(clk)) then
        if WA="00" then a<=i;
        elsif WA="01" then b<=i;
        elsif WA="10" then c<=i;
        end if;
      end if;

      if RA="00" then S<=a;
      elsif RA="01" then S<=b;
      elsif RA="10" then S<=c;
      end if;

      if WA="00" then D<=a;
      elsif WA="01" then D<=b;
      elsif WA="10" then D<=c;
      end if;
  end process;
  aa<=a;
  bb<=b;
  cc<=c;
end bhv;
```

b) 编译、调试过程

编译成功

c) RTL 视图

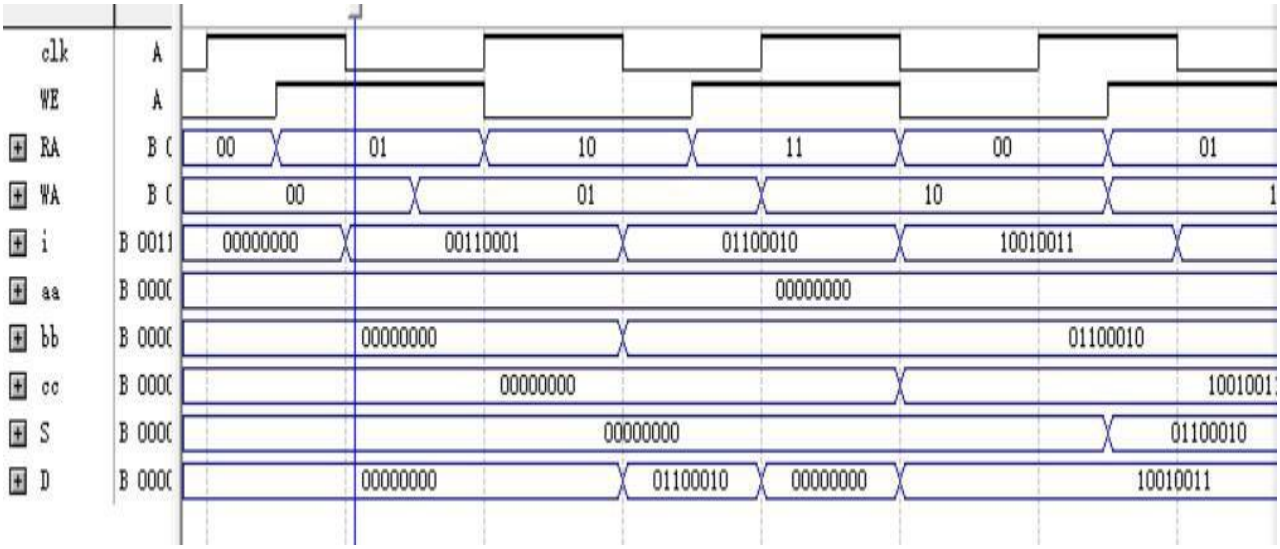


6、 功能仿真

e) 功能仿真过程

做好上述步骤后，在【processing】中选择【simulator tool】-【generate functional simulation netlist】-【start simulation】

a) 功能仿真图



b) 结果分析及结论

当时钟下降沿且 WE=0 时：

WA=00，i=00000000 时，aa=00000000，bb=00000000，cc=00000000;

WA=01，i=00110001 时，aa=00000000，bb=01100010，cc=00000000;

WA=10，i=10010011 时，aa=00000000，bb=01100010，cc=10010011;

不论时钟信号以及 WE 为何值：

当 RA=00 时，S=aa=00000000;

当 RA=01 时，S=bb=00000000;

当 RA=10 时，S=cc=00000000;

当 WA=00 时 ，D=aa=00000000;

当 WA=01 时，D=bb=01100010;

当 WA=10 时，D=cc=10010011;

对于上图中第二个下降沿：WE=0，RA=10，WA=01，i=01100010 时，

将外部输入 i=01100010 写入 b 寄存器中，从 S 口输出 b 寄存器储存的值

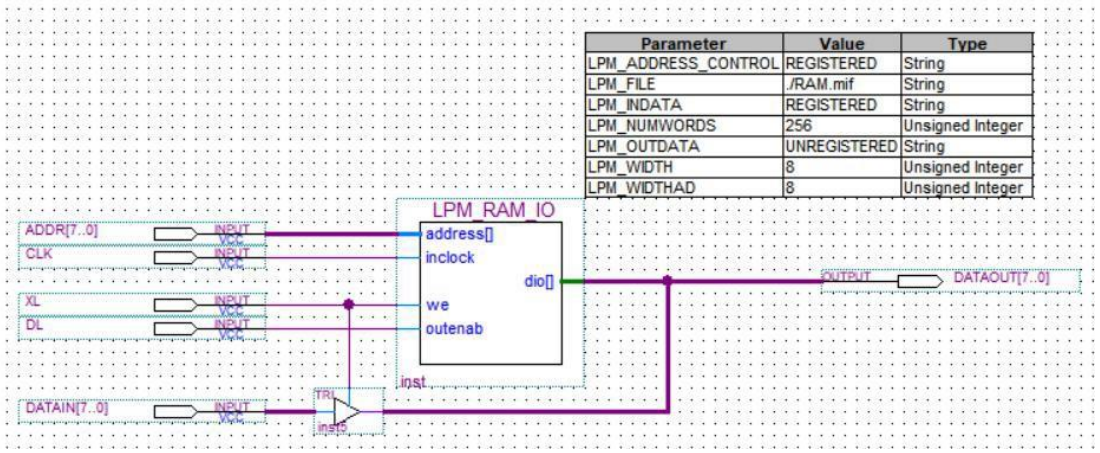
从 D 口输出 c 寄存器储存的值;

仿真结果正确。

RAM：

7、 编译过程

a）原理图



b)编译、调试过程

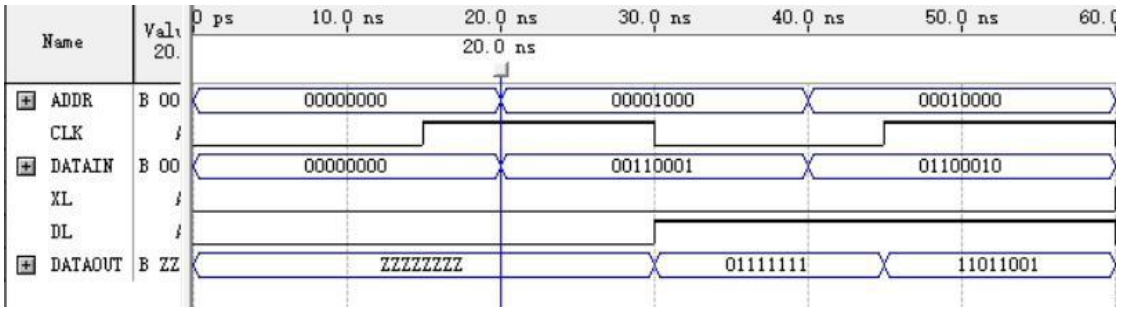
编译成功

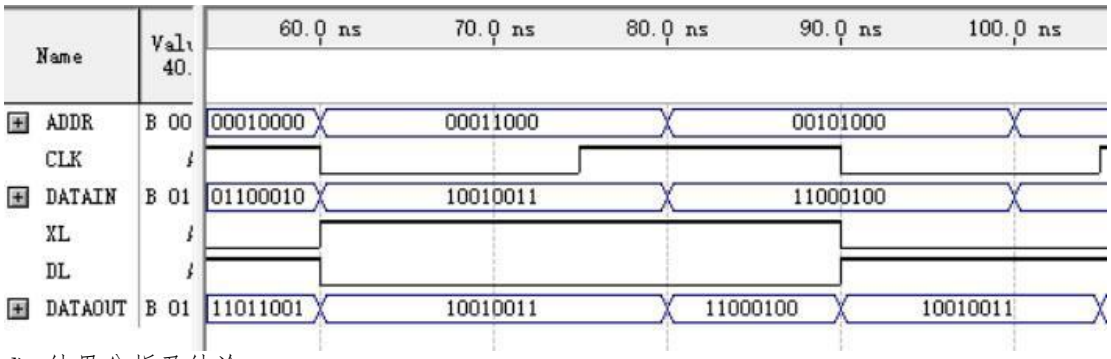
8、 功能仿真

f) 功能仿真过程

做好上述步骤后，在【processing】中选择【simulator tool】-【generate functional simulation netlist】-【start simulation】

c) 功能仿真图





d) 结果分析及结论

RAM 为时钟上升沿时触发：

在时钟为 44.935ns（ 第二个时钟上升沿） 时， XL=0， DL=1， 从 mif 文件中读入 DATAOUT=11011001；

在时钟为 75.042ns 时，XL=1，DL=0，将外部输入写入 DATAOUT=DATAIN=10010011；

仿真结果正确。

六、实验结论

1、思考题

1）时钟周期的上升沿实现对 RAM 的读写操作，为何 PC，SM 以及寄存器组的操作是下降沿完成的？

RAM 进行存储操作时，在时钟的上升沿到来时实现，因为此时数据跟地址都已经是个稳定的状态，这样才能保证数据的正确存储。

RAM 进行读数据时，也在时钟的上升沿到来时实现，因为此时读地址已经处于稳定的状态，这样才能保证读到的数是相应地址内的数据，数据在读时钟的上升沿到来后输出。

而对于 PC、SM 及寄存器组则在下降沿时进行操作稳定性更强。2）

总结 VHDL 语言描述时序部件的方法和常用语句。

时序逻辑电路具有“记忆”功能，因此触发器是时序电路最常用的记忆元件， 任何时序逻辑电路都是以时钟信号为驱动信号，在时钟信号边沿到来时发生状态变化。

常用语句：

EVENT:

上升沿检测表达式和信号属性函数，关键字 EVENT 是信号属性函数，用来获得信号行为信号的函数称为信号属性函数。

例如：

CLK'EVENT AND CLK='1'为检测时钟的上升沿的语句，其作用为如果检到 CLK 的上升沿，表达式值为 TURE。

If 语句：

在描述时序部件时，首先考查时钟信号 CLK 的情况，即满足 IF 语句条件的情况。

Process 语句：

当 CLK 发生变化时， PROCESS 语句被启动， IF 语句将测定条件表达式 CLK'EVENT AND CLK='1'是否满足条件，如果 CLK 的确出现了上升沿，则满足条件表达式是对上升沿的检测，于是执行后面的操作。

2、实验总结与实验心得

本次实验，SM，PC 以及寄存器组相对简单，在自习阅读实现指导书自己可以独立完成。RAM 对于我来说你比较难，阅读完实验指导书，不能够很好地理解功能以及实现方式。在网络查找资料后，才完成。通过本次实现，我学习了如何通过 quartus 已有的参数模块实现来定制 RAM，对于 symbol 元件库有了一定的了解。