**东 北 大 学**

**课程设计报告**



课程名称: 数字电子技术课程设计

设计题目：基于FPGA的8位模型机的设计与实现

设计日期： 2018 年7月8日~2018 年7月14日

# 课程设计（论文）任务书

|  |
| --- |
| **课程设计（论文）题目：基于FPGA的8位模型机的设计与实现** |
| **基本内容 基本要求：**  用VHDL设计一个8位模型机系统，包括设计实现指令系统、指令格式、寻址方式、寄存器结构、数据表示方式、存储器系统，运算器、控制器等。用VHDL硬件语言进行逻辑设计与实现。  模型计算机就是以计算机实际结构为基础，将其简化，能对输入的信息进行处理运算，更便于分析设计。现代计算机主要由运算器，控制器，存储器，输入设备，输出设备五大部分组成。计算机能完成用户要求是按照提前设计好的指令进行的，指令是计算机执行具体操作的命令。一条指令就是机器语言的一个语句，用它来说明机器硬件应完成什么样的基本操作。本次所设计的8位模型机包含十个模块，分别是存储器，时钟信号源，节拍发生器，操作控制器，程序计数器，地址寄存器，累加器，算术逻辑单元，指令寄存器和指令译码器。可完成两个立即数相加，并将相加结果送入累加器，指令系统规模为3条指令：   1. LD 把立即数送入累加器A，机器码为00111110 2. ADD 将累加器A中的数与立即数相加，结果送入累加器，机器码11000110 3. HALT 运算完毕，停机，机器码为01110110 |

摘要

通过一台模型机的设计，可以帮助我们更好的学习和理解CPU的基本结构和原理。本次课程设计完成了基于VHDL的8位模型计算机的设计与实现。文中首先阐述了8位模型计算机的概念和原理，然后对其十个功能模块（算数逻辑运算单元、累加器、控制器、地址寄存器、程序计数器、数据寄存器、存储器、节拍发生器、时钟信号源、指令寄存器和指令译码器）进行了设计与分析。最后在Quartus II 18.0环境下进行了仿真，完成了8位模型计算机的整体实现。

整个8位模型机的设计与实现体现出了CPU在实现特定指令时的整个工作流程和在Quartus II软件平台上用VHDL设计数字控制系统的实用性。

**关键词：**FPGA， 8位模型机， 时钟， VHDL语言

目录

[课程设计（论文）任务书 i](#_Toc519453498)

[摘要 ii](#_Toc519453499)

[第1章 绪论 1](#_Toc519453500)

[1.1 8位模型机简介 1](#_Toc519453501)

[1.2设计主要内容 1](#_Toc519453502)

[1.2.1 时钟产生器模块（CLK\_SOURCE模块） 1](#_Toc519453503)

[1.2.2 算术逻辑单元模块（ALU模块） 2](#_Toc519453504)

[1.2.3 累加器模块（ACC模块） 3](#_Toc519453505)

[1.2.4 操作控制器模块（CTRL模块） 4](#_Toc519453506)

[1.2.5 节拍发生器模块（COUNTER模块） 5](#_Toc519453507)

[1.2.6 指令寄存器和指令译码器模块（IR模块） 6](#_Toc519453508)

[1.2.7 程序计数器模块（PC模块） 7](#_Toc519453509)

[1.2.8 地址寄存器模块（MAR模块） 8](#_Toc519453510)

[1.2.9 存储器模块（RAM模块） 9](#_Toc519453511)

[1.2.10 数据寄存器模块（DR模块） 10](#_Toc519453512)

[第2章 系统设计 11](#_Toc519453513)

[2.1系统组成 11](#_Toc519453514)

[2.2 CPU模块 12](#_Toc519453515)

[第3章 系统仿真 14](#_Toc519453516)

[3.1 CPU仿真 14](#_Toc519453517)

[3.2时钟产生器模块仿真 15](#_Toc519453518)

[3.3存储器模块仿真 15](#_Toc519453519)

[3.4程序计数器模块仿真 16](#_Toc519453520)

[3.5地址寄存器模块仿真 16](#_Toc519453521)

[3.6指令寄存器和指令译码器模块仿真 17](#_Toc519453522)

[3.7数据寄存器模块仿真 17](#_Toc519453523)

[3.8操作控制器模块仿真 18](#_Toc519453524)

[3.9节拍发生器模块仿真 18](#_Toc519453525)

[3.10算数逻辑单元模块仿真 19](#_Toc519453526)

[3.11累加器模块仿真 20](#_Toc519453527)

[第4章 系统实现 21](#_Toc519453528)

[4.1 系统下载验证 21](#_Toc519453529)

[第5章 结论 22](#_Toc519453530)

[参考文献 23](#_Toc519453531)

[附录A RTL图 24](#_Toc519453532)

[A1. 时钟信号源的RTL图 24](#_Toc519453533)

[A2. 存储器的RTL图 24](#_Toc519453534)

[A3. 程序计数器的RTL图 25](#_Toc519453535)

[A4. 地址寄存器的RTL图 25](#_Toc519453536)

[A5. 指令寄存器和指令译码器的RTL图 25](#_Toc519453537)

[A6. 数据寄存器的的RTL图 25](#_Toc519453538)

[A7. 操作控制器的RTL图 26](#_Toc519453539)

[A8. 节拍发生器的RTL图 27](#_Toc519453540)

[A9. 算术逻辑单元的RTL图 27](#_Toc519453541)

[A10. 累加器的RTL图 28](#_Toc519453542)

[A11. CPU的RTL图 28](#_Toc519453543)

[附录B VHDL程序清单 29](#_Toc519453544)

[附录C 实物测试照片 38](#_Toc519453545)

# 

# 第1章 绪论

## 1.1 8位模型机简介

所谓模型计算机就是以计算机实际结构为基础，将其简化，能对输入的信息进行处理运算，更便于分析设计。随着微电子技术的进步，现代计算机主要由运算器，控制器，存储器，输入设备，输出设备五大部分组成。计算机能完成用户要求是按照提前设计好的指令进行的，指令是计算机执行具体操作的命令。一条指令就是机器语言的一个语句，用它来说明机器硬件应完成什么样的基本操作。在本课题中把模型机划分了十个模块分别是存储器，时钟信号源，节拍发生器，操作控制器，程序计数器，地址寄存器，累加器，算术逻辑单元，指令寄存器和指令译码器。让预设指令在这些部件中按顺序执行达到预期目的。计算机执行一条指令分为三步进行：

第1步是取指令，将要执行的指令从内存取到控制器中；

第2步是分析指令，对所取的指令通过译码器进行分析判断，判断该指令要完成的操作；

第3步是执行指令，根据分析结果向各部件发出操作信息，执行该指令相应的操作功能。

## 1.2设计主要内容

### 1.2.1 时钟产生器模块（CLK\_SOURCE模块）

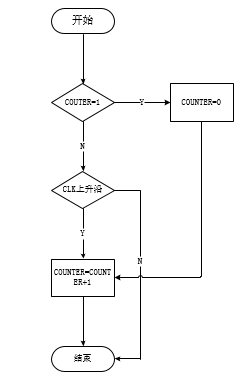
 时钟产生器模块用于产生固定频率的方波，设置COUNTER值从0到1依次按照CLK周期顺次加一得以实现，如此循环，产生满足设计要求的方波。

图1.2.1 CLK\_SOURCE模块流程图

### 1.2.2 算术逻辑单元模块（ALU模块）

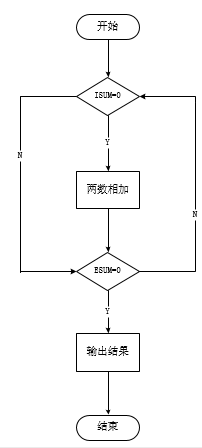
 算术逻辑单元ALU功能模块，有两个控制信号，分别是加法运算控制信号ISUM和输出控制信号ESUM，当ISUM=0时，在CLK作用下将两数相加，当输出控制命令ESUM=0时，将相加结果读入到内部数据总线上。

图1.2.2 ALU模块流程图

### 1.2.3 累加器模块（ACC模块）

累加器模块使用了一个中间信号RES\_TEMP，输入信号是DATA\_IN，输出信号时DATA\_OUT，两个控制信号分别是IA输入控制信号和EA输出控制信号。当输入控制信号IA=0时，将输入数据DATA\_IN存入到RES\_TEMP中，当输出控制信号EA=0是将以存到RES\_TEMP中的输入数据赋与输出数据DATA\_OUT输出。



图1.2.3 ACC模块流程图

### 1.2.4 操作控制器模块（CTRL模块）

该控制器模块的输入信号为三个操作指令信号LD,ADD,HALT和T0-T7八个节拍脉冲及一个时钟信号CLK，输出为九个控制信号IA,EA,ISUM,ESUM,IDR,EDR,IPC,IIR,IMAR。当HALT=1时IPC=0和ESUM=0，当HALT=0时，整个模型机正常工作，9个控制信号别根据不同的节拍发生有不同的取值来执行各自控制功能。

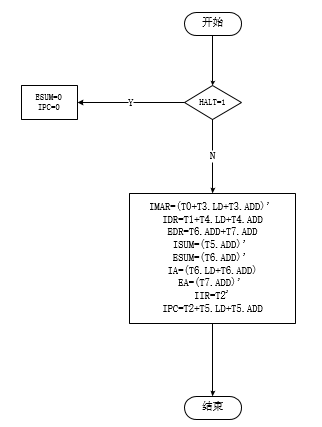


图1.2.4 CTRL模块流程图

### 1.2.5 节拍发生器模块（COUNTER模块）

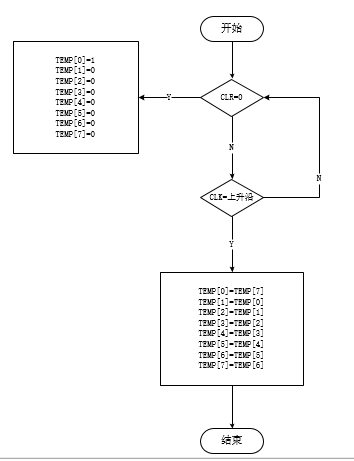
 该节拍发生器模块用于产生T0-T7八个节拍的脉冲信号，使得模型机按照此节拍有序地工作。输入信号为CLR和CLK，当CLR=0时给八个节拍赋初值，否则在CLK上升沿到来时实现循环右移，以实现特定的功能。

图1.2.5 COUNTER模块流程图

### 1.2.6 指令寄存器和指令译码器模块（IR模块）

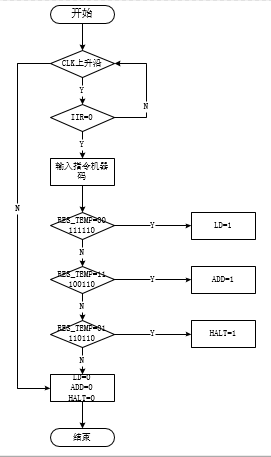
该指令寄存器和指令译码器模块的输入为CLK，IIR和DATA\_IN[7..0]，输出为LD，ADD，HALT。当CLK的上升沿到来时，如果IIR=0则将DATA\_IN送入RES\_TEMP，再对指令进行译码。

图1.2.5 IR模块流程图

### 1.2.7 程序计数器模块（PC模块）

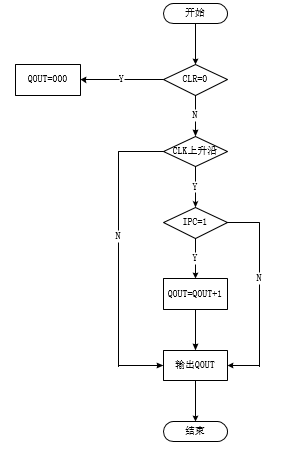
该程序计数器模块的作用是确定下一条指令的地址。由于模型计算机只有5个字节的机器码，所以程序计数器PC的输出只使用3位，当IPC=0时，计数器保持原状态；IPC=1时，计数器处于计数状态，当时钟信号CLK上升沿到来时，做加1的运算。

图1.2.7 PC模块流程图

### 1.2.8 地址寄存器模块（MAR模块）

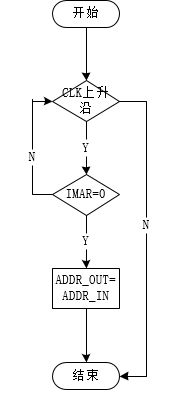
该地址寄存器模块用来保存当前CPU所访问的主存储器单元的地址，由于主存储器与CPU之间存在操作速度上的差别，所以必须使用地址寄存器来保存地址信息，直到主存储器的读/写操作完成为止。当IMAR=0且时钟信号CLK上升沿到来时，将地址存入，并直接输出到存储器的地址线上。

图1.2.8 MAR模块流程图

### 1.2.9 存储器模块（RAM模块）

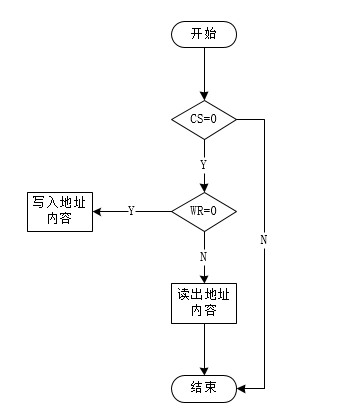
 该存储器模块用来存储相关的机器码，共有四个输入：CS片选端，WR读写控制端，ADDR输入的地址码，DIN输入的机器码。输出为DOUT[7..0]。当CS=0时RAM工作，WR=0时将DIN写入RAM中ADDR指定的内存单元，WR=1时将指定的ADDR内存单元的内容送入DOUT并输出。

图1.2.9 RAM模块流程图

### 1.2.10 数据寄存器模块（DR模块）

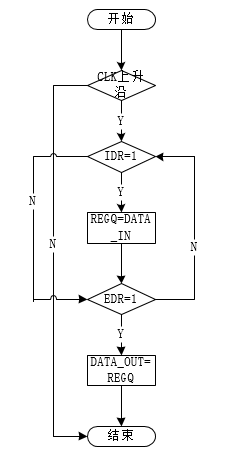
该数据寄存器模块用来暂时存放由主存储器读出的一条指令或数据。当IDR=1且时钟信号CLK上升沿到来时，将被选中的存储单元中的数据存入DR。当EDR=1时DR输出成高阻态；当EDR=0时，DR将所存储数据送到数据总线。

图1.2.10 DR模块流程图

# 

# 第2章 系统设计

## 2.1系统组成

图2.1 8位模型机系统顶层设计图

本次设计的8位模型机包含十个模块，分别是存储器，时钟信号源，节拍发生器，操作控制器，程序计数器，地址寄存器，累加器，算术逻辑单元，指令寄存器和指令译码器。可实现取数，两个立即数相加和停机的功能。

## 2.2 CPU模块

图2.2 CPU整体结构图

表2.1简易微处理器的指令系统

|  |  |  |
| --- | --- | --- |
| 汇编语句 | 操作符 | 功能 |
| LD A, #data | 00111110 | Data 🡪A |
| ADD A, #data | 11000110 | (Data)+A 🡪A |
| HALT | 01110110 | 停机 |

模型机CPU整个工作流程如下：

1. 上电后，CPU复位（CLR=0），程序计数器PC清零；节拍发生器控制T0=1，在T0节拍内，IMAR=0，地址寄存器将地址码送入地址线，控制RAM可将LD的指令码00111110送入DR的输入端。
2. 下一个CLK上升沿到来时，节拍发生器控制T1=1，此时IDR=1，EDR=0；可以将LD的 指令码00111110送入DR并将其送入内部数据总线。
3. 下一个CLK上升沿到来时，节拍发生器控制T2=1，IIR=0，IPC=1，PC执行PC+1操作，PC内容变成001。IIR=0内部数据总线将指令码存入指令寄存器IR并进行译码，译码输出为LD=1。
4. 下一个CLK上升沿到来时，节拍发生器控制T3=1，因为LD=1，所以IMAR=0；地址寄存器MAR将001存入，由存储器读出操作数为00000001并将其送入外部数据总线中。
5. 下一个CLK上升沿到来时，节拍发生器控制T4=1，IDR=1，EDR=0；数据寄存器DR将指令码送入内部数据总线中。
6. CLK的第6个上升沿到来时，节拍发生器控制T5=1，IPC=1，PC执行PC+1操作，即PC内容变成010。
7. CLK的第七个上升沿到来时，节拍发生器控制T6=1，IA=0，内部数据总线将00000001送入累加器ACC。
8. 下一个CLK上升沿到来时，节拍发生器控制T7=1，各个模块保持原来的状态，第一条指令”LD 1”执行完毕。
9. 下一个CLK上升沿到来时，开始新的节拍周期，第2条指令的T0-T2节拍期间由PC的内容为010，从RAM中取出指令码11000110，经译码后ADD=1，即执行相加操作；T2期间PC执行PC+1操作，PC内容变为011。
10. 在下一个CLK上升沿到来时，T3=1，IMAR=0；PC将011存入地址寄存器MAR，由存储器RAM读出00000110，该数在外部数据总线中。
11. 下一个CLK上升沿到来时，T4=1，IDR=1，EDR=0，外部数据总线将00000110送入DR并送入内部数据总线。
12. 下一个CLK上升沿到来时，T5=1，ISUM=0，算数逻辑单元ALU将累加器ACC中00000001和内部数据总线中00000110两数相加并将结果存在ALU中。
13. 下一个CLK上升沿到来时，T6=1，ESUM=0，ALU将两数相加的结果输出。IA=0，EDR=1，数据寄存器DR的输出呈现高阻态，与内部数据总线切断。
14. T7=1节拍期间，EA=0，EDR=1，将累加器ACC的结果送入内部数据总线，第二条指令”ADD 6”执行完毕。
15. 在第3条指令的T0-T2期间，PC内容为100，从存储器RAM中取出指令码为01110110，经译码后为HALT=1，停机。

# 第3章 系统仿真

## 3.1 CPU仿真

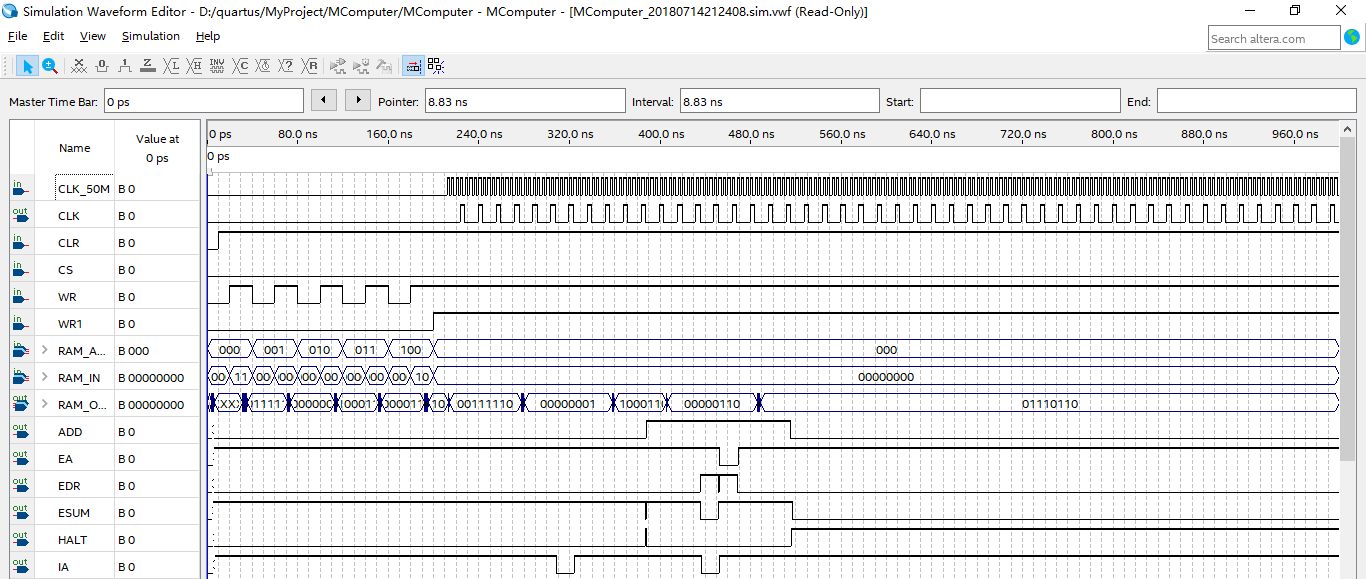
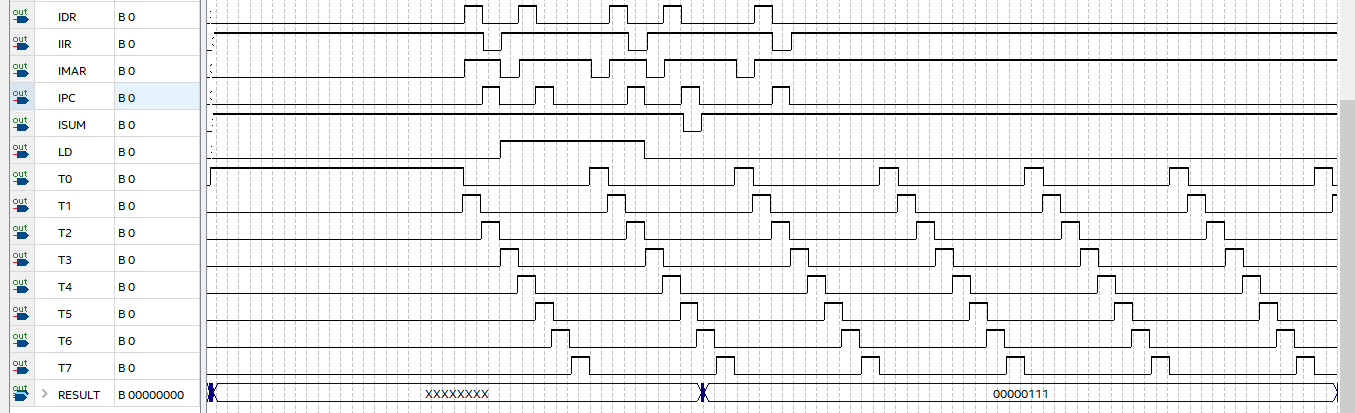
8位模型机CPU仿真如图3.1所示

图3.1 8位模型机CPU仿真波形图

波形分析：刚开始CLR=0，实现上电复位。WR=0时向存储器中写入数据，依次将机器码00111110、00000001、11100110、00000110、01110110送入存储器，分别表示LD 1，ADD 6，HALT。随后WR=1使存储器为只读状态，给出时钟信号源，模型机开始逐条分析和执行指令，最终结果显示为00000111，即1+6=7，结果正确，停机。

## 3.2时钟产生器模块仿真

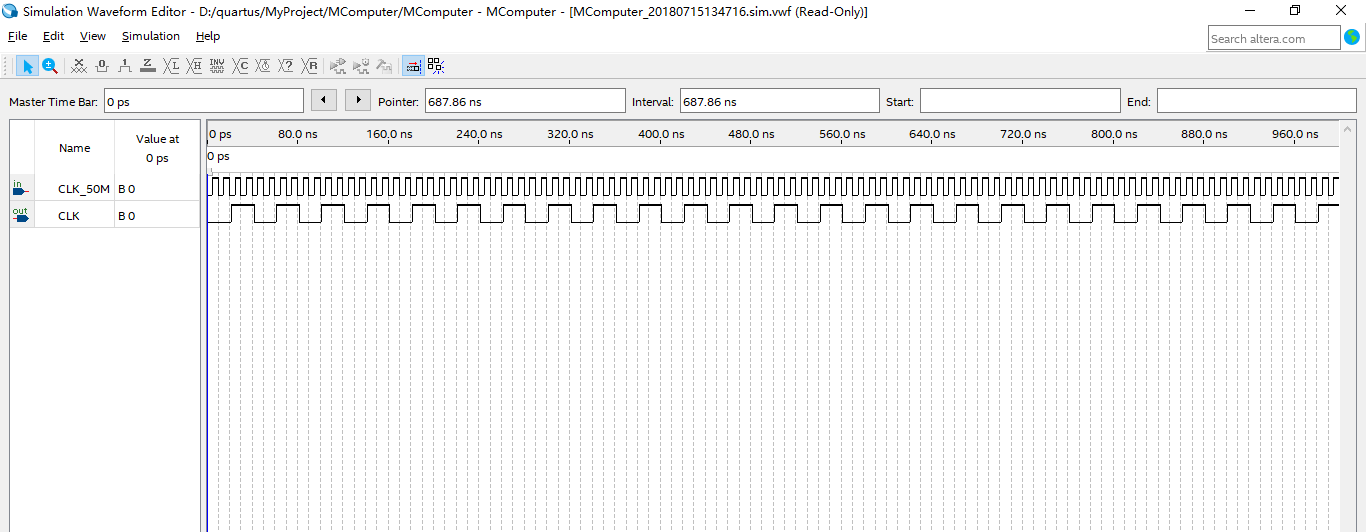
模型机时钟产生器模块仿真结果如图3.2所示

图3.2 时钟产生器模块仿真结果

波形分析：输入信号CLK\_50M周期为5.0ns，在每个时钟上升沿到来时COUNT自动加1，到1后再清零，如此循环产生固定周期的输出时钟信号。

## 3.3存储器模块仿真

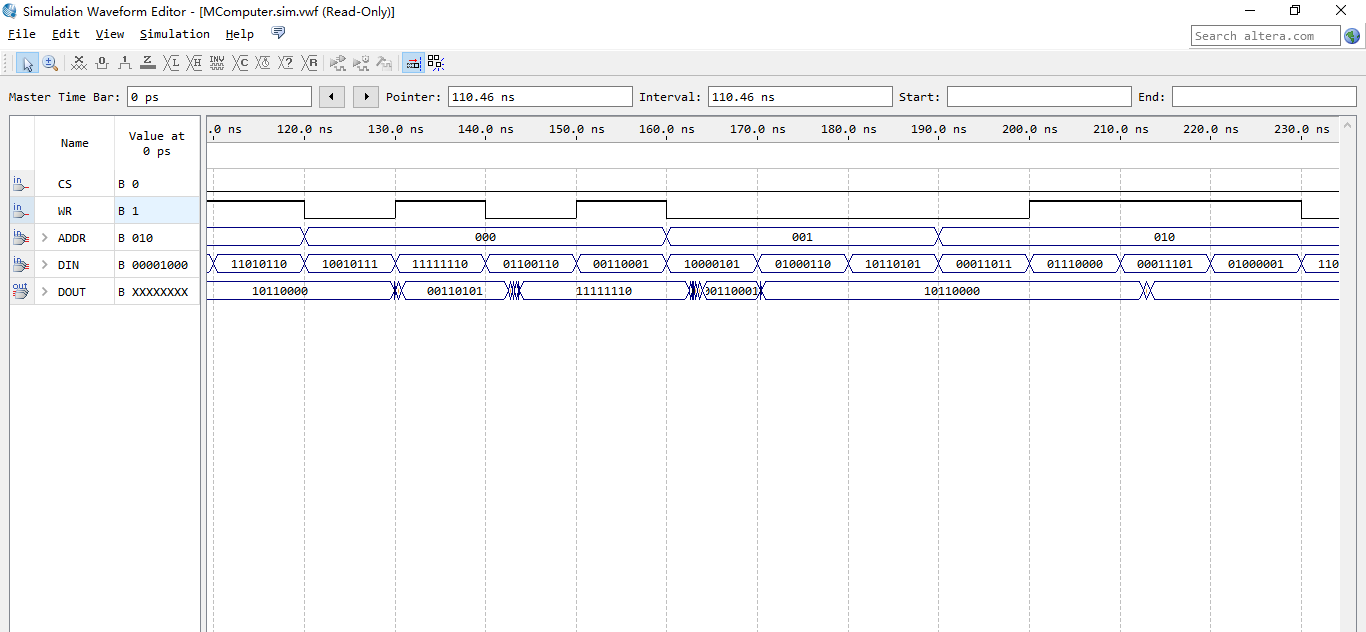
模型机存储器模块仿真结果如图3.3所示

图3.3 存储器模块仿真结果

波形分析：CS为片选端，当CS=0时存储器正常工作。当WR=0时存储器进行写入操作，由波形可知可以成功将下一周期DIN给出的机器码写入ADDR所指出的内存单元中。当WR=1时存储器进行读出操作，DOUT显示ADDR所指出内存单元的内容。

## 3.4程序计数器模块仿真

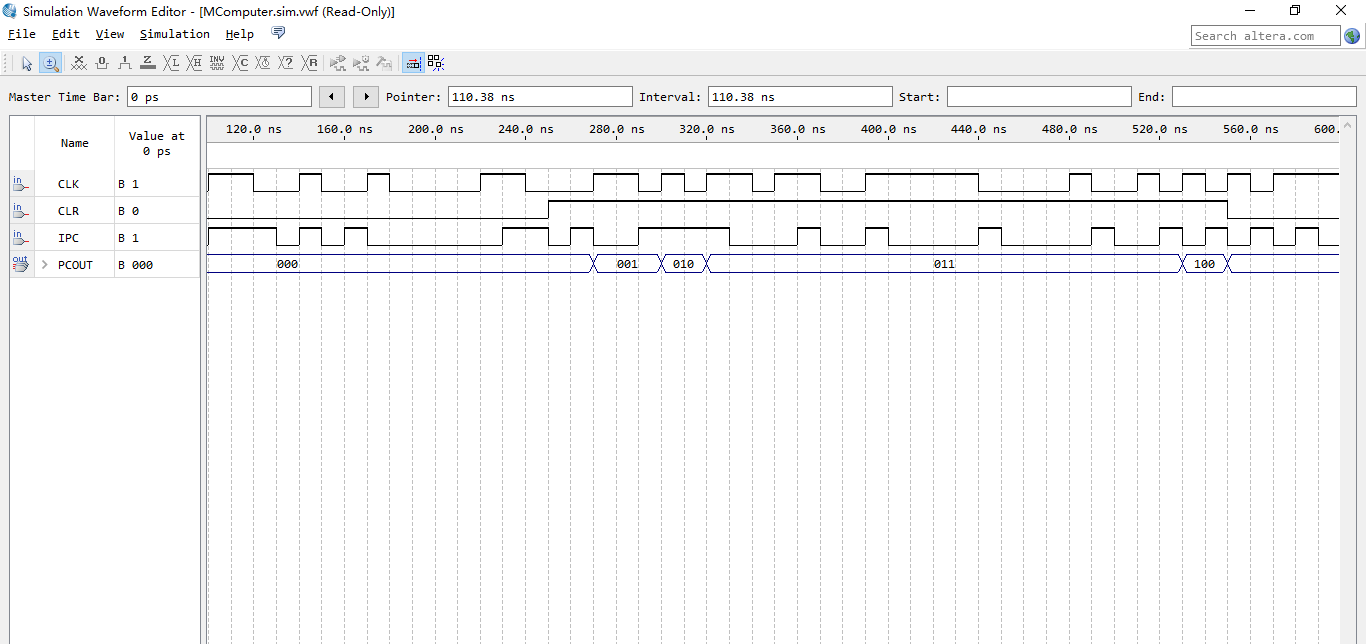
模型机程序计数器模块仿真结果如图3.4所示

图3.4 程序计数器模块波形仿真结果

波形分析：当复位端CLR=0时，地址码清零，PCOUT结果为000，当CLK上升沿到来时，IPC=1时模块执行PC+1的操作，PCOUT+1，结果正确。

## 3.5地址寄存器模块仿真

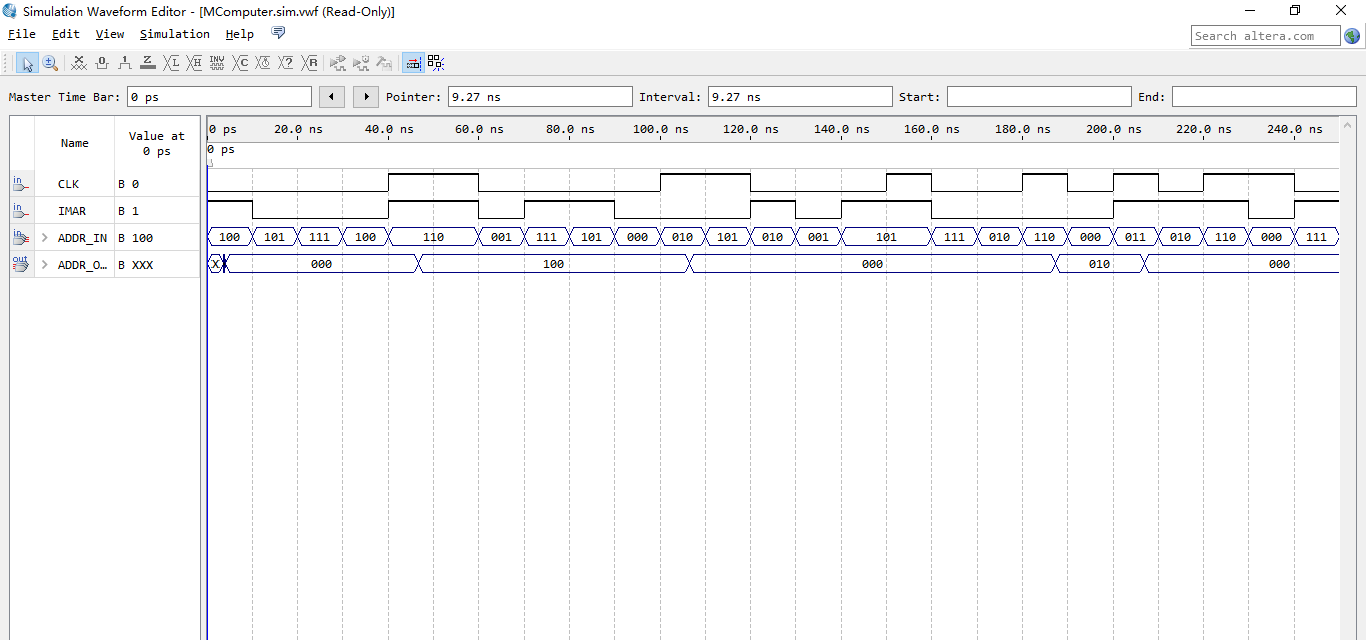
模型机地址寄存器模块仿真结果如图3.5所示

图3.5 地址寄存器模块波形仿真结果

波形分析：在CLK上升沿到来时，模块允许输出信号IMAR=0时MAR将ADDR\_IN的地址码输出，结果显示正确。

## 3.6指令寄存器和指令译码器模块仿真

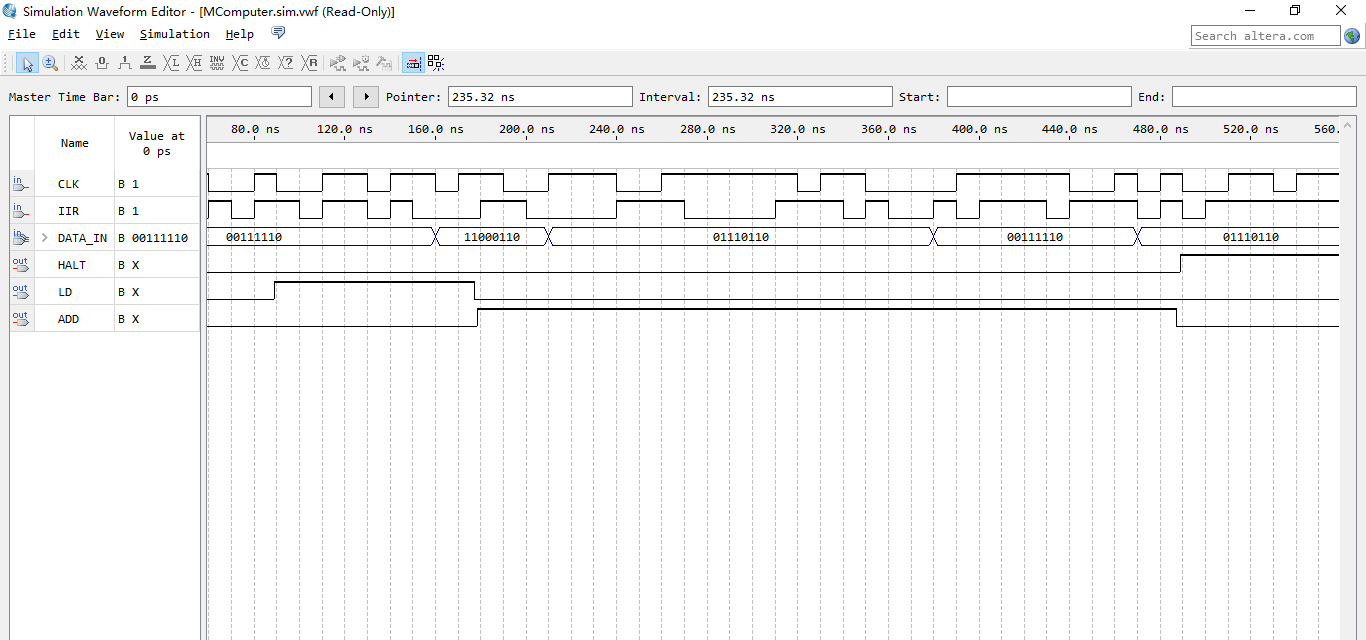
模型机指令寄存器和指令译码器模块的仿真结果如图3.6所示

图3.6 指令寄存器和指令译码器模块波形仿真结果

波形分析：当CLK上升沿到来时，IR允许输入信号IIR=0时IR将输入的指令码DATA\_IN送入IR，IR对送入的指令码进行译码。由波形可知输出结果正确。

## 3.7数据寄存器模块仿真

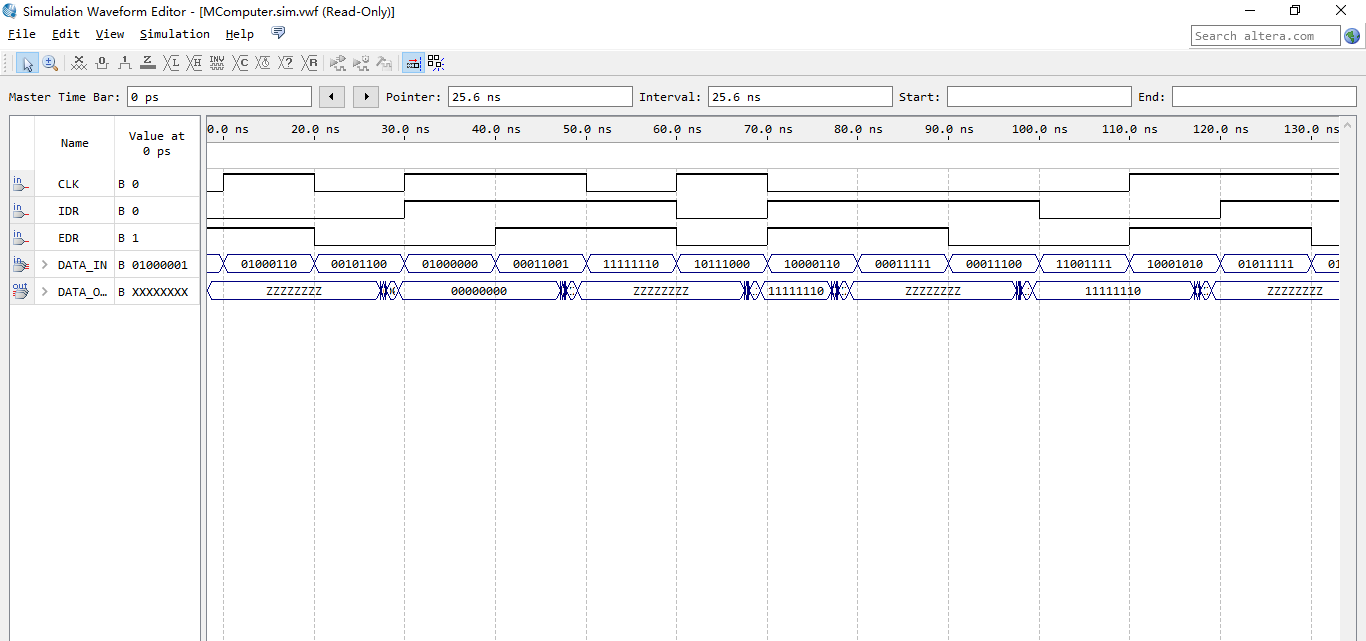
模型机数据寄存器模块仿真结果如图3.7所示

图3.7 数据寄存器模块波形仿真结果

波形分析：当CLK上升沿到来时，DR允许写入信号IDR=1时DR将DATA\_IN写入DR中，当允许输出信号EDR=0时将DR中的指令码输出，由波形可知输出结果正确。

## 3.8操作控制器模块仿真

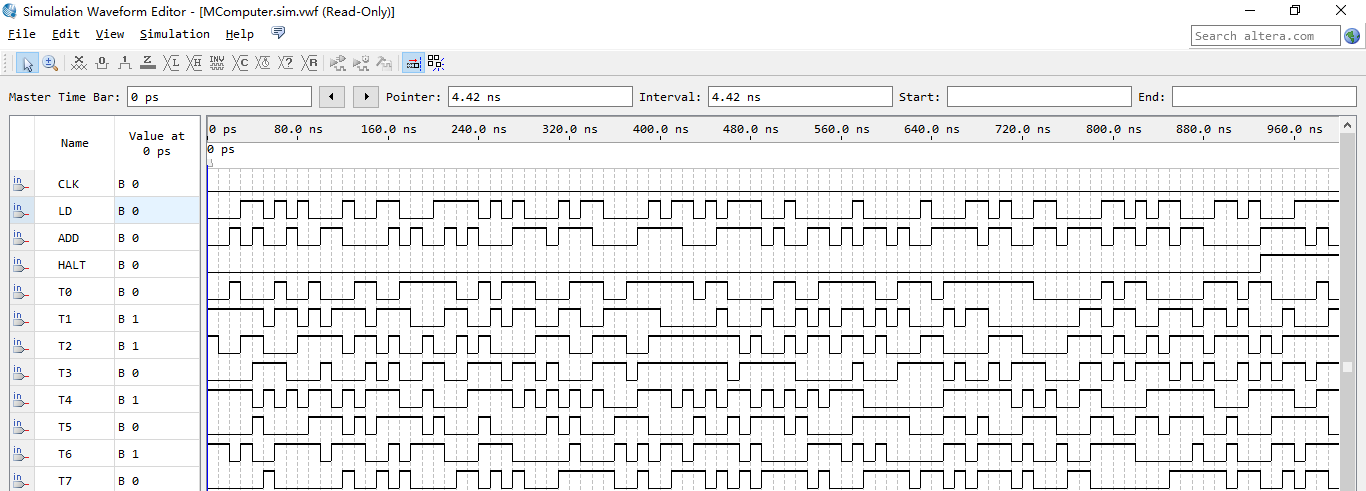
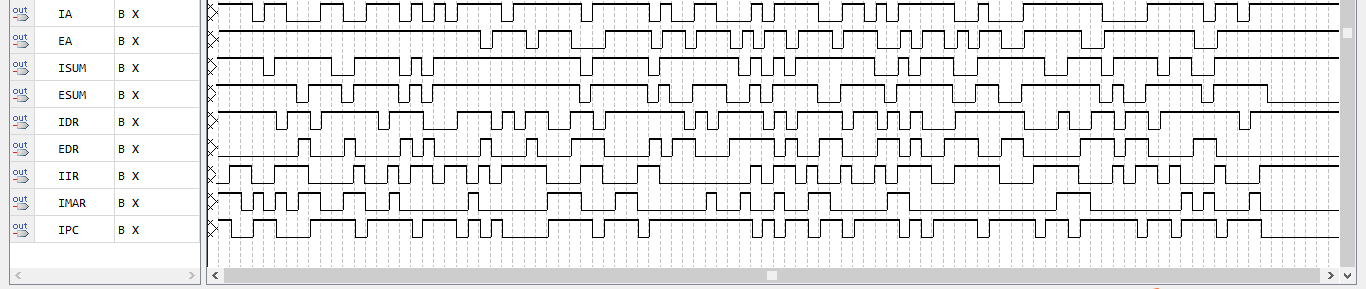
模型机操作控制器模块的仿真结果如图3.8所示

图3.8 操作控制器模块的波形仿真结果

波形分析：LD,ADD,T0-T7设置为0,1交替，HALT先为低电平再为高电平，由各个控制信号的输出结果可知控制器设计成功。

## 3.9节拍发生器模块仿真

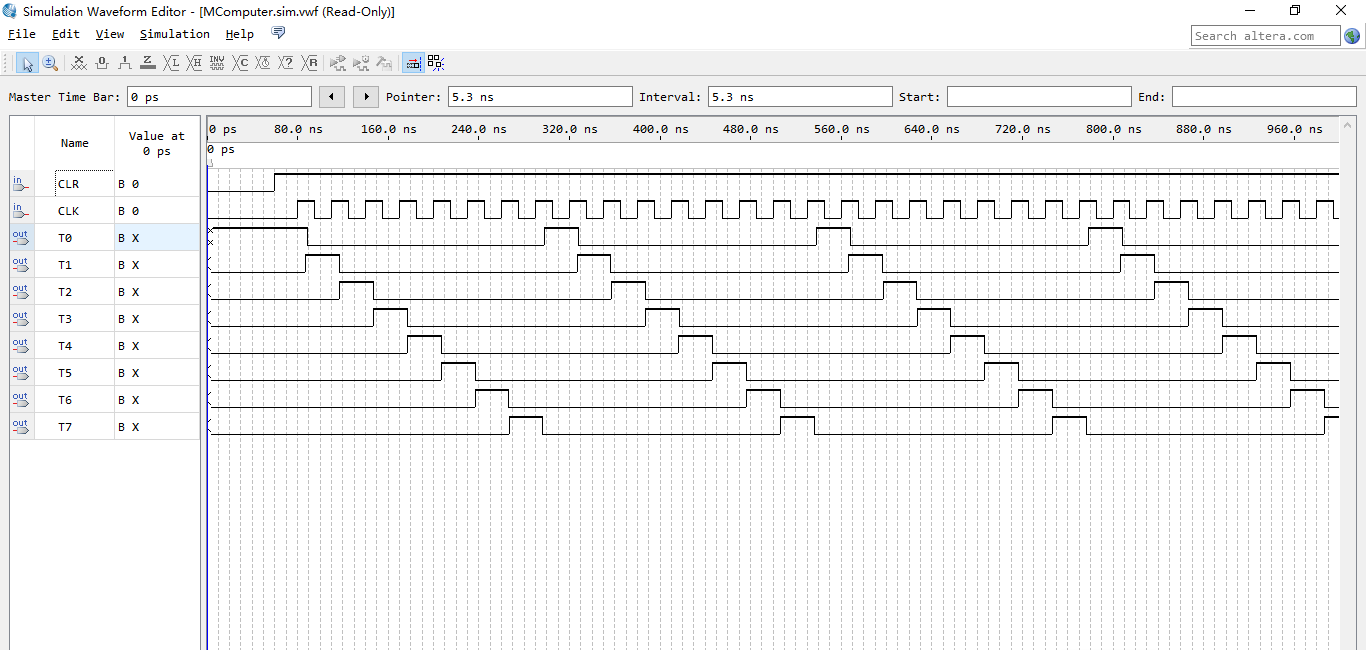
模型机节拍发生器模块的仿真结果如图3.9所示

图3.9 节拍发生器模块的波形仿真结果

波形分析：根据输入的时钟信号以8个时钟周期为一个大周期发生一个节拍脉冲，从T0到T7依次循环发生，波形结果显示节拍发生器模块功能设计成功。

## 3.10算数逻辑单元模块仿真

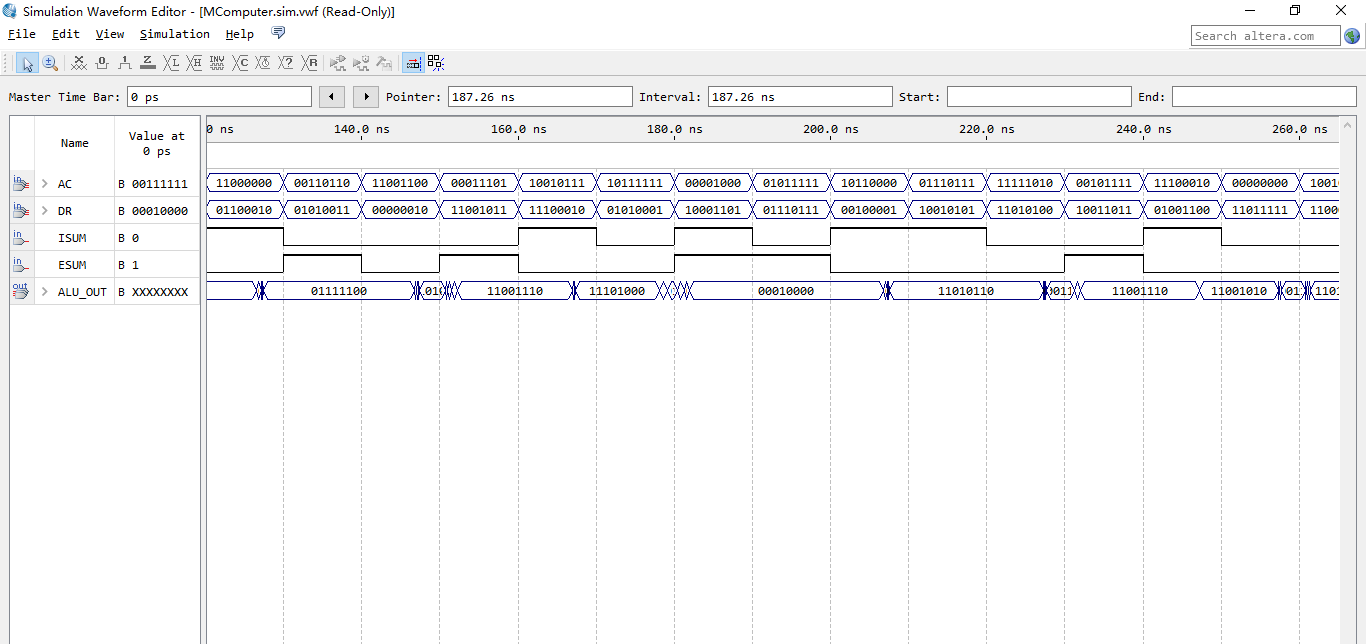
模型机算数逻辑单元的仿真结果如图3.10所示

图3.10 算数逻辑单元的波形仿真结果

波形分析：该模块有两个输入信号AC，DR，一个输出信号ALU\_OUT和两个控制命令信号ISUM,ESUM。在仿真设置中将AC与DR的值设为随机数，两个控制命令信号全部设为0即即时相加即时输出，运行后如图所示，输出结果也显示功能仿真结果正确。

## 3.11累加器模块仿真

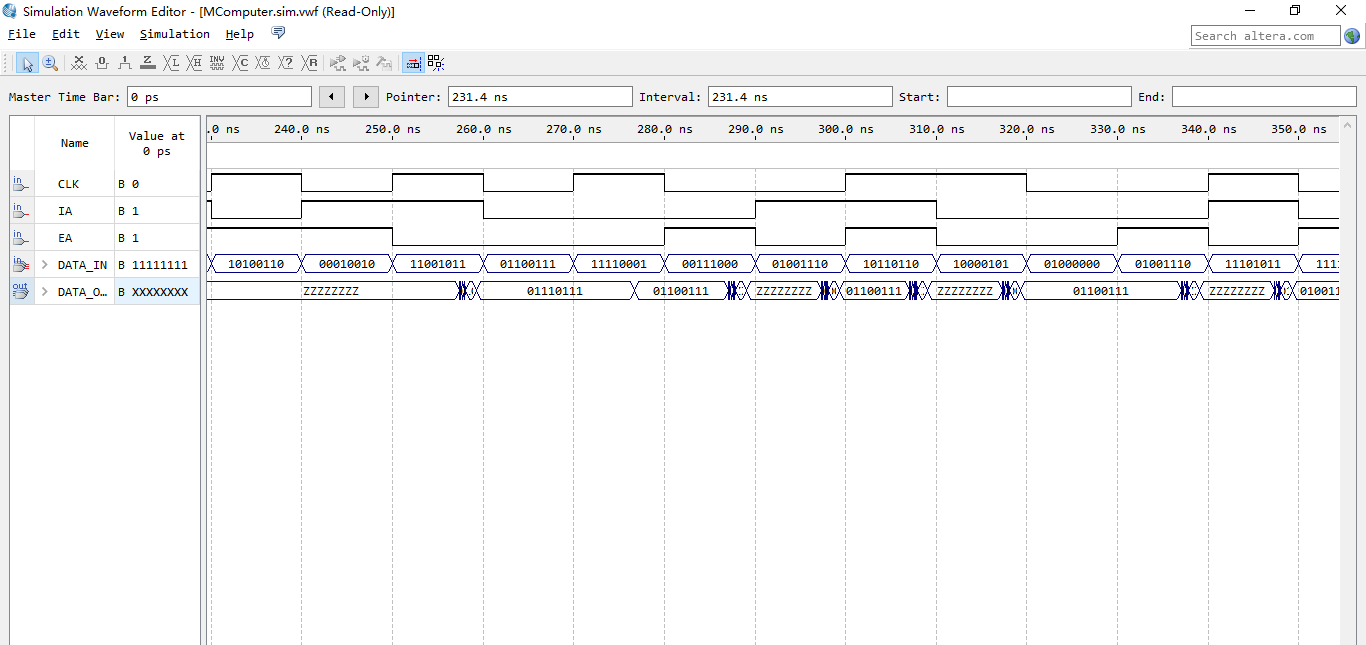
模型机累加器模块的仿真结果如图3.11所示

图3.11 累加器模块波形仿真结果

波形分析：该累加器模块设置一个中间信号RES\_TEMP，输入信号是DATA\_IN，输出信号时DATA\_OUT，两个控制信号分别是IA输入控制信号和EA输出控制信号。当时钟信号上升沿到来时，如果输入控制信号IA=0，将输入数据DATA\_IN存入到RES\_TEMP中，输出控制信号EA=0时将以存到RES\_TEMP中的输入数据赋与输出数据DATA\_OUT输出。

# 第4章 系统实现

## 4.1 系统下载验证

图4.1.1 八位模型机管脚分配



图4.1.2 系统实现实物图

# 第5章 结论

本次数字电路技术课程设计我所选的课题是“基于FPGA的8位模型机的设计与实现”，其基本要求是用VHDL设计一个8位模型机系统，包括设计实现指令系统、指令格式、寻址方式、寄存器结构、数据表示方式、存储器系统，运算器、控制器等。用VHDL硬件语言进行逻辑设计与实现。课程设计需要在Quartus II软件中用VHDL语言实现各个模块的功能与波形仿真，再将程序下载到到电路板中。在一周的紧张学习中，我主要完成了以下工作：

1. 对8位模型机所能实现的功能进行分析和设计，再针对总体功能将8位模型机划分为十个相对独立的模块。
2. 对每个模块进行单独的设计与分析，再用VHDL语言实现，在调式完毕后再运用波形仿真检验所设计的模块是否能正确实现相应的功能。
3. 每个独立模块波形仿真通过后，用原理图的方式将各个模块连接起来，进行总体调试与波形仿真，使其能实现8位模型机的功能。
4. 将程序下载到FPGA的板子中，反复调试，使其最终实现了8位模型机的功能。

通过本次课程设计，极大地提高了我的自学能力，我学会了使用Quartus II软件开发数字电路课程相关的逻辑器件并最终在硬件中实现。同时也加深了我对计算机CPU的基本组成结构和工作原理。本次所设计的8位模型机也存在着功能简单、实现的指令码单一的不足之处，希望能在日后的学习中不断地改进和完善8位模型机。

# 参考文献

1. 王诚等. Alteration FPGA/CPLD设计（基础篇）(M) .北京：人民邮电出版社，2005.
2. 刘欲晓,方强.EDA技术与VHDL电路开发应用实践[M].北京: 电子工业出版社,2009.
3. 吕炎军,王泽勇. 基于FPGA的I2C总线接口设计[J].自动化技术与应用, 2006,(04):65-69.

# 附录A RTL图

## A1. 时钟信号源的RTL图

图A1 时钟信号源的RTL图

## A2. 存储器的RTL图

图A2 存储器的RTL图

## A3. 程序计数器的RTL图

图A3 程序计数器的RTL图

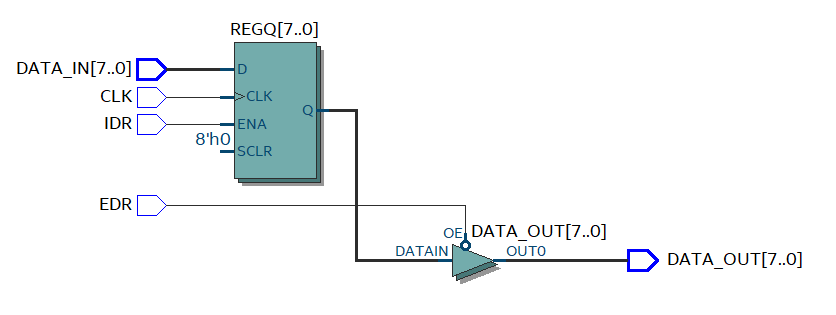
## A4. 地址寄存器的RTL图

图A4 地址寄存器的RTL图

## A5. 指令寄存器和指令译码器的RTL图

图A5 指令寄存器和指令译码器的RTL图

## A6. 数据寄存器的的RTL图



图A6 数据寄存器的RTL图

## A7. 操作控制器的RTL图

图A7 操作控制器的RTL图

## A8. 节拍发生器的RTL图

图A8 节拍发生器的RTL图

## A9. 算术逻辑单元的RTL图

图A9 算数逻辑单元的RTL图

## A10. 累加器的RTL图

图A10 累加器的RTL图

## A11. CPU的RTL图

图A11 CPU的RTL图

# 附录B VHDL程序清单

1. CLK\_SOURCE.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

--时钟产生器模块

--用于产生固定频率的方波

ENTITY CLK\_SOURCE IS

PORT(

CLK\_50M:IN STD\_LOGIC;

CLK:OUT STD\_LOGIC

);

END ENTITY;

ARCHITECTURE A OF CLK\_SOURCE IS

SIGNAL COUNT:INTEGER RANGE 0 TO 1;

SIGNAL CLK\_TEMP:STD\_LOGIC:='0';

BEGIN

PROCESS(CLK\_50M)

BEGIN

IF RISING\_EDGE(CLK\_50M) THEN

IF(COUNT >= 1)THEN --COUNT=1时清零

COUNT <= 0;

CLK\_TEMP <= NOT CLK\_TEMP;

ELSE

COUNT <= COUNT +1; --否则，COUNT执行COUNT+1操作

END IF;

END IF;

END PROCESS;

CLK <= CLK\_TEMP; --将CLK\_TEMP输出

END A;

1. RAM.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

--存储器模块

--可以实现读写功能

ENTITY RAM IS

PORT(

WR,CS: IN STD\_LOGIC; --CS为片选端，WR为读写端

DIN: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); --输入指令码

DOUT: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0); --输出的指令码

ADDR: IN STD\_LOGIC\_VECTOR(2 DOWNTO 0) --输入的地址码

);

END;

ARCHITECTURE A OF RAM IS

TYPE MEMORY IS ARRAY(0 TO 4) OF STD\_LOGIC\_VECTOR(7 DOWNTO 0); --MEMORY为大小为5的数组

BEGIN

PROCESS(CS,WR)

VARIABLE MEM:MEMORY;

BEGIN

IF(CS='0') THEN

IF(WR='0') THEN --WR=0进行写入操作

MEM(CONV\_INTEGER(ADDR(2 DOWNTO 0))):=DIN; --强制类型转换为整数

ELSIF(WR='1') THEN --WR=1进行读出操作

DOUT <= MEM(CONV\_INTEGER(ADDR(2 DOWNTO 0)));

END IF;

END IF;

END PROCESS;

END A;

1. PC.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

--程序计数器模块

--用以产生正确的地址

ENTITY PC IS

PORT(

IPC,CLK,CLR: IN STD\_LOGIC; --CLK为时钟周期，CLR为复位端，IPC为允许计数指令

PCOUT: OUT STD\_LOGIC\_VECTOR(2 DOWNTO 0) --输出产生的地址码

);

END;

ARCHITECTURE A OF PC IS

SIGNAL QOUT: STD\_LOGIC\_VECTOR(2 DOWNTO 0);

BEGIN

PROCESS(CLK,CLR,IPC)

BEGIN

IF(CLR='0') THEN --CLR=0时复位，将000赋值给QOUT

QOUT <= "000";

ELSIF(CLK'EVENT AND CLK='1') THEN --CLK上升沿

IF(IPC='1') THEN --IPC=1时程序计数

QOUT <= QOUT+1; --QOUT执行QOUT+1操作

END IF;

END IF;

END PROCESS;

PCOUT<=QOUT;

END A;

1. MAR.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

--地址寄存器模块

--临时存放地址码

ENTITY MAR IS

PORT(

ADDR\_IN: IN STD\_LOGIC\_VECTOR(2 DOWNTO 0); --从PC接收的地址码

IMAR: IN STD\_LOGIC; --MAR允许输出信号

CLK: IN STD\_LOGIC; --时钟信号

ADDR\_OUT: OUT STD\_LOGIC\_VECTOR(2 DOWNTO 0) --MAR输出地址码端口

);

END;

ARCHITECTURE A OF MAR IS

BEGIN

PROCESS(CLK,IMAR)

BEGIN

IF(CLK'EVENT AND CLK='1') THEN --CLK上升沿到来时

IF (IMAR='0') THEN --IMAR=0时允许输出

ADDR\_OUT <= ADDR\_IN;

END IF;

END IF;

END PROCESS;

END A;

1. IR.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

--指令寄存器和指令译码器

--暂存指令和对指令码译码

ENTITY IR IS

PORT(

DATA\_IN: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); --输入的指令码

IIR: IN STD\_LOGIC; --IR允许输入信号

CLK: IN STD\_LOGIC; --时钟信号

LD,ADD,HALT: OUT STD\_LOGIC --输出的译码结果

);

END IR;

ARCHITECTURE A OF IR IS

SIGNAL RES\_TEMP: STD\_LOGIC\_VECTOR(7 DOWNTO 0);

BEGIN

PROCESS(CLK,IIR)

BEGIN

IF(CLK'EVENT AND CLK='1')THEN --CLK上升沿到来时

IF(IIR='0')THEN

RES\_TEMP <= DATA\_IN; --IIR=0时将DATA\_IN赋值给RES\_TEMP

END IF;

END IF;

END PROCESS;

PROCESS(CLK,RES\_TEMP)

BEGIN

CASE RES\_TEMP IS --对RES\_TEMP进行译码

WHEN"00111110"=> LD <='1'; ADD <='0'; HALT<='0';

WHEN"11000110"=> LD <='0'; ADD <='1'; HALT<='0';

WHEN"01110110"=> LD <='0'; ADD <='0'; HALT<='1';

WHEN OTHERS=>LD <='0'; ADD <='0'; HALT<='0';

END CASE;

END PROCESS;

END A;

1. DR.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

--数据寄存器

--暂时存放指令码

ENTITY DR IS

PORT(

DATA\_IN: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); --输入的指令码

IDR: IN STD\_LOGIC; --DR允许写入信号

EDR: IN STD\_LOGIC; --DR允许输出信号

CLK: IN STD\_LOGIC; --时钟信号

DATA\_OUT: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) --DR的指令码输出信号

);

END;

ARCHITECTURE A OF DR IS

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

BEGIN

PROCESS(CLK,IDR,EDR)

BEGIN

IF(CLK'EVENT AND CLK='1') THEN --CLK上升沿到来时

IF(IDR='1') THEN --IDR=1时允许写入

REGQ <= DATA\_IN;

END IF;

END IF;

END PROCESS;

DATA\_OUT <= REGQ WHEN EDR='0' ELSE "ZZZZZZZZ"; --EDR=0时允许输出，否则结果为高阻态

END A;

1. CTRL.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

--操作控制器模块

--控制模型机正常运行

ENTITY CTRL IS

PORT(

LD,ADD,HALT: IN STD\_LOGIC; --输入的译码结果

CLK: IN STD\_LOGIC; --时钟信号

T0,T1,T2,T3,T4,T5,T6,T7:IN STD\_LOGIC; --输入的节拍信号

IPC,IMAR,IDR,EDR,IA,EA,ISUM,ESUM,IIR: OUT STD\_LOGIC --输出的指令信号

);

END ENTITY;

ARCHITECTURE A OF CTRL IS

BEGIN

PROCESS(LD,ADD,HALT)

BEGIN

IF (HALT='1') THEN --HALT=1时模型机停止工作

IPC <= '0';

ESUM <= '0';

ELSE --由译码输入和节拍输入控制各个指令信号

IMAR <= not(t0 or (t3 and ld)or (t3 and add));

IIR <= not t2;

IA <= not((t6 and ld)or(t6 and add ));

IDR <= t1 or(t4 and ld) or (t4 and add);

IPC <= t2 or (t5 and ld) or (t5 and add);

ISUM <= not (t5 and add);

ESUM <= not(t6 and add);

EA <= not (t7 and add);

EDR <= (t6 and add) or (t7 and add);

END IF;

END PROCESS;

END A;

1. COUNTER.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

--节拍发生器模块

--产生T0-T7 八个节拍信号

ENTITY COUNTER IS

PORT(

CLK,CLR:IN STD\_LOGIC; --输入的时钟信号和CLR复位信号

T0,T1,T2,T3,T4,T5,T6,T7:OUT STD\_LOGIC --输出的T0-T7节拍信号

);

END COUNTER;

ARCHITECTURE A OF COUNTER IS

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

BEGIN --T0-T7赋值

T0<=TEMP(0);

T1<=TEMP(1);

T2<=TEMP(2);

T3<=TEMP(3);

T4<=TEMP(4);

T5<=TEMP(5);

T6<=TEMP(6);

T7<=TEMP(7);

PROCESS(CLK,CLR)

BEGIN

IF(CLR='0')THEN --CLR=0时复位,T0=1

TEMP(0)<='1';

TEMP(1)<='0';

TEMP(2)<='0';

TEMP(3)<='0';

TEMP(4)<='0';

TEMP(5)<='0';

TEMP(6)<='0';

TEMP(7)<='0';

ELSIF(CLK'EVENT AND CLK='1')THEN --时钟信号上升沿到来时

TEMP(0)<=TEMP(7); --T0-T7信号循环右移

TEMP(1)<=TEMP(0);

TEMP(2)<=TEMP(1);

TEMP(3)<=TEMP(2);

TEMP(4)<=TEMP(3);

TEMP(5)<=TEMP(4);

TEMP(6)<=TEMP(5);

TEMP(7)<=TEMP(6);

END IF;

END PROCESS;

END A;

1. ALU.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

--算数逻辑单元

--可以实现两个立即数相加并将结果输出

ENTITY ALU IS

PORT(

AC,DR:IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); --输入的两个立即数

ISUM: IN STD\_LOGIC; --ALU加法操作信号

ESUM: IN STD\_LOGIC; --ALU允许输出信号

ALU\_OUT: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) --输出结果信号

);

END ALU;

ARCHITECTURE A OF ALU IS

SIGNAL ALU\_TEMP: STD\_LOGIC\_VECTOR(7 DOWNTO 0);

BEGIN

ALU\_TEMP <= AC+DR WHEN ISUM='0'; --ISUM=0时实现两个立即数相加

ALU\_OUT <= ALU\_TEMP WHEN ESUM='0'; --ESUM=0输出。

END A;

1. ACC.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

--累加器模块

ENTITY ACC IS

PORT(

DATA\_IN: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); --输入指令码

IA: IN STD\_LOGIC; --ACC允许输入信号

EA: IN STD\_LOGIC; --ACC允许输出信号

CLK: IN STD\_LOGIC; --时钟信号

DATA\_OUT: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) --输出指令码

);

END ACC;

ARCHITECTURE A OF ACC IS

SIGNAL RES\_TEMP: STD\_LOGIC\_VECTOR(7 DOWNTO 0);

BEGIN

PROCESS(IA,EA,CLK)

BEGIN

IF(CLK'EVENT AND CLK='1') THEN --CLK时钟上升沿到来时

IF(IA='0') THEN -- IA=0时输入

RES\_TEMP<=DATA\_IN;

END IF;

END IF;

END PROCESS;

DATA\_OUT<= RES\_TEMP WHEN EA='0' ELSE "ZZZZZZZZ"; --EA=0时允许输出，否则输出为高阻态

END A;

1. RAM\_MUX.vhd 文件

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

--数据选择器

--可以实现手动向存储器制定地址单元写入数据

ENTITY RAM\_MUX IS

PORT(

SEL : IN STD\_LOGIC;

DATA0X : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

DATA1X : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

RESULT : OUT STD\_LOGIC\_VECTOR(2 DOWNTO 0)

);

END;

ARCHITECTURE A OF RAM\_MUX IS

BEGIN

RESULT <= DATA0X WHEN SEL='0' ELSE DATA1X;

END A;

# 附录C 实物测试照片

