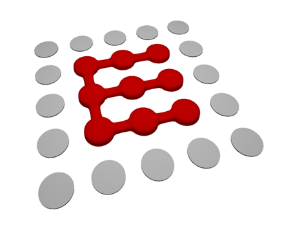
 

**本科实验报告**

**实验名称： VerilogHDL数字电路设计实验**

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称： | 数字系统设计与实验（软件部分） | 实验时间： | 2018/6/9、6/10 下午 |
| 任课教师： | 赵宏图 | 实验地点： | 理学楼B-404 |
| 实验教师： | 南方 | 实验类型： | ■ 原理验证  □ 综合设计  □ 自主创新 |
| 学生姓名： | 施念 |
| 学号/班级： | 1120161302/05011609 | 组 号： |  |
| 学 院： | 信息与电子学院 | 同组搭档： |  |
| 专 业： | 电子信息工程 | 成 绩： |  |





目录

[实验一 QuartusII9.1软件的使用 3](#_Toc516696651)

[一、实验目的： 3](#_Toc516696652)

[二、实验步骤： 3](#_Toc516696653)

[1. 范例运行 3](#_Toc516696654)

[2. VHDL编写3-8译码器 9](#_Toc516696655)

[三、实验心得 12](#_Toc516696656)

[实验二 模十状态机与7段译码器显示 13](#_Toc516696657)

[一、实验目的： 13](#_Toc516696658)

[二、实验流程： 13](#_Toc516696659)

[三、实验步骤 13](#_Toc516696660)

[1. 设计思路 13](#_Toc516696661)

[2. 实验代码： 13](#_Toc516696662)

[3. 实验结果： 16](#_Toc516696663)

[四 实验心得 18](#_Toc516696664)

[实验三 数字钟的设计与仿真 19](#_Toc516696665)

[一、实验目的： 19](#_Toc516696666)

[二、实验流程： 19](#_Toc516696667)

[三、实验步骤: 19](#_Toc516696668)

[1. 设计思路 19](#_Toc516696669)

[2. 实验代码 19](#_Toc516696670)

[3. 实验结果 21](#_Toc516696671)

[四、实验心得: 22](#_Toc516696672)

# 实验一 QuartusII9.1软件的使用

## 一、实验目的：

1. 通过实现简单组合逻辑电路，掌握QUARTUSII 9.1软件的使用；

2. 编程实现3-8译码电路以掌握VHDL组合逻辑的设计以及QUARTUSII 9.1软件的使用。

## 二、实验步骤：

### 1. 范例运行

（1）实验程序:

module count10

(

input clk, load, en,

input [3:0] qin,

output reg [7:0] seg

);

reg [3:0] qout;

always @ (posedge clk or posedge load) begin

if (load)

qout <= qin;

else

if ( en )

if (qout == 4'b1001)

qout <= 4'b0000;

else

qout <= qout +1 ;

else

qout <= qout ;

end

always @ (qout) begin

case (qout)

0:

seg <= 7'b1000000;

1:

seg <= 7'b1111001;

2:

seg <= 7'b0100100;

3:

seg <= 7'b0110000;

4:

seg <= 7'b0011001;

5:

seg <= 7'b0010010;

6:

seg <= 7'b0000010;

7:

seg <= 7'b1111000;

8:

seg <= 7'b0000000;

9:

seg <= 7'b0010000;

default:

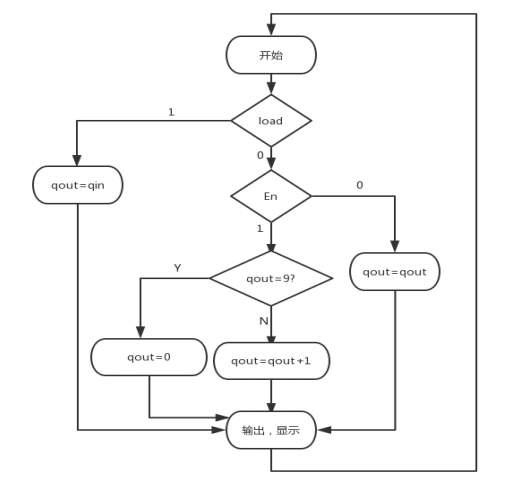
seg <= 7'b0001000;

endcase

end

endmodule

（2）功能图



（3）操作步骤：

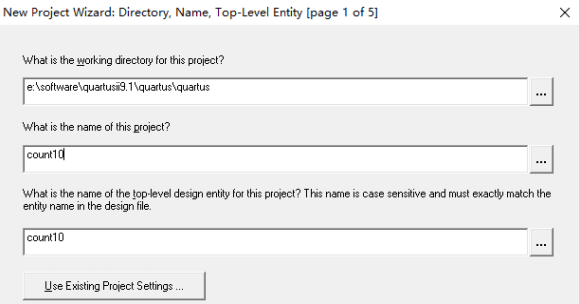
（一）建立 Verilog HDL 文件

1、先建立一个工作目录文件

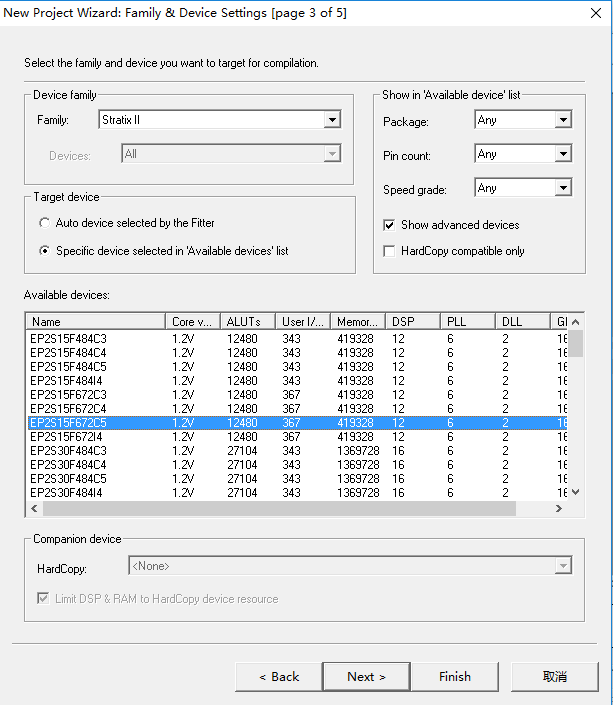
2、创建一个新项目：



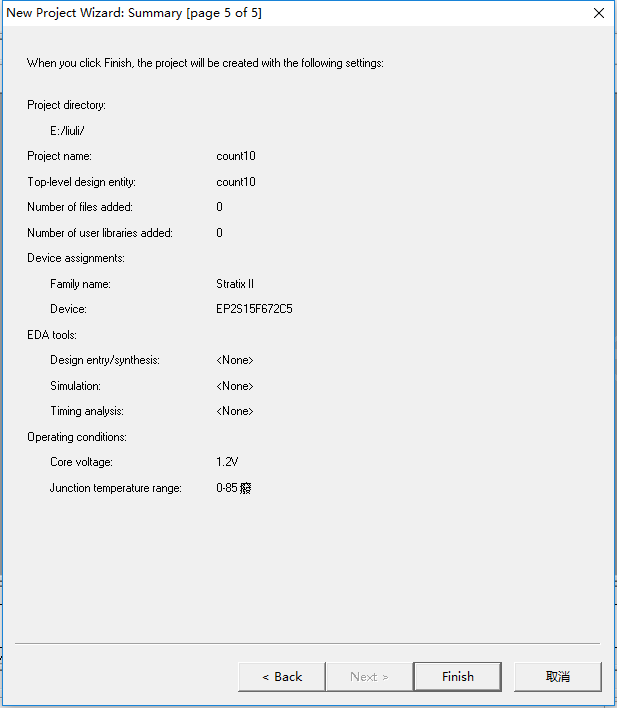
对项目进行命名：



参数如下：



确定无误后，点击 Finish



3、新建文件

点击 File—>New，淡出对话框后选择 Verilog HDL File，然后进行编写代码。

（二）全编译与功能仿真

1、对 Verilog HDL File 编译： 从菜单栏中选择 Processing－Start－Start Analysis&Synthesis，选取菜单中 Processing －Start Compilation 进行全编译。

2、指定功能仿真模式 选择菜单

选择菜单中 Assignments－settings 或快捷按钮

3、通过建立波形文件进行仿真

在 Quartus II 主界面菜单栏中选择 File－New，在 Other Files 页选中 Vector Waveform File 项，如图所示。点击 OK 按钮打开空白波形编辑窗口，其默认文件名为 “Waveforml.vwf”。

选择菜单栏中 Edit－Insert Node or Bus，选择其中的 node finder

再点击弹出窗口中的 List 按钮在左侧 Nodes Found 窗口中选取 clk、qout、data\_in、 en、load、seg，然后点击按钮将选中信号选取至右侧 Selected Nodes 窗口中，最后点 击 OK 回到插入节点窗口

运行菜单 Processing－Generate Functional Simulation Net list 命令产生用于功能仿真 的网表文件。

选取 Processing－Start Simulation 或快捷按钮执行模拟仿真

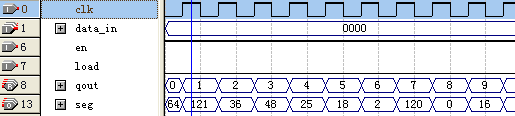
（三）时序仿真

选择菜单中 Assignments－settings 或快捷按钮，在左侧 Category 栏中选中 Simulator Settings，然后在右侧 Simulation mode 的下拉菜栏中选中 Timing，确定。然后点击 Processing－Start Simulation 或快捷按钮执行模拟仿真

功能仿真设置：



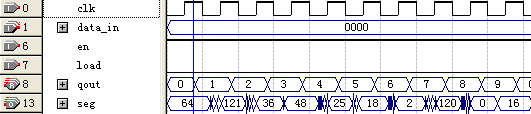
功能仿真结果：



时序仿真设置：



时序仿真结果：



### 2. VHDL编写3-8译码器

（1）设计思路

定义实体comp，当输入D\_IN从“000”到“111”时，输出Q\_OUT依次从“00000001”到“100000000”，如果输入错误，则输出“11111111”：

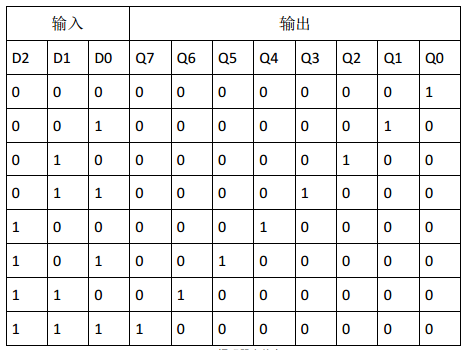


Figure 1 3-8 译码器真值表

（2）实验代码（VHDL）

-------------------------main.vhd-------------------------------

library ieee;

use ieee.std\_logic\_1164.all;

----------------------------------------------------------------

entity comp is

port (

D\_IN : in std\_logic\_vector(2 downto 0);

Q\_OUT : out std\_logic\_vector(7 downto 0)

);

end entity comp ;

----------------------------------------------------------------

architecture beha of comp is

--选择输出38译码

begin

----------------------------------------------------------------

CHOOSE : process( D\_IN )

begin

case (D\_IN) is

when "000" => Q\_OUT<="00000001";

when "001" => Q\_OUT<="00000010";

when "010" => Q\_OUT<="00000100";

when "011" => Q\_OUT<="00001000";

when "100" => Q\_OUT<="00010000";

when "101" => Q\_OUT<="00100000";

when "110" => Q\_OUT<="01000000";

when "111" => Q\_OUT<="10000000";

when others => Q\_OUT<="11111111";

end case;

end process ; -- CHOOSE

---------------------------------------------------------------

end architecture beha;

---------------------------------------------------------------

（3）测试代码

------------------------main\_tb.vhd-------------------------

library ieee;

use ieee.std\_logic\_1164.all;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

------------------------------------------------------------

entity comp\_tb is

end entity comp\_tb;

------------------------------------------------------------

architecture comp\_tb of comp\_tb is

component comp is

port (

D\_IN : in std\_logic\_vector(2 downto 0);

Q\_OUT : out std\_logic\_vector(7 downto 0)

);

end component comp;

signal D\_IN:std\_logic\_vector(2 downto 0);

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

constant clk\_period :time :=1 us; --时钟周期的定义

begin

dut : comp port map(D\_IN=>D\_IN,Q\_OUT=>Q\_OUT);

-------------------------------------------------------------

D\_IN\_GEN : process

begin

D\_IN <= "000";

while true loop

--if(CLK\_IN'event and CLK\_IN ='1') then

wait for clk\_period/2;

D\_IN <= D\_IN + 1;

--end if;

end loop ;

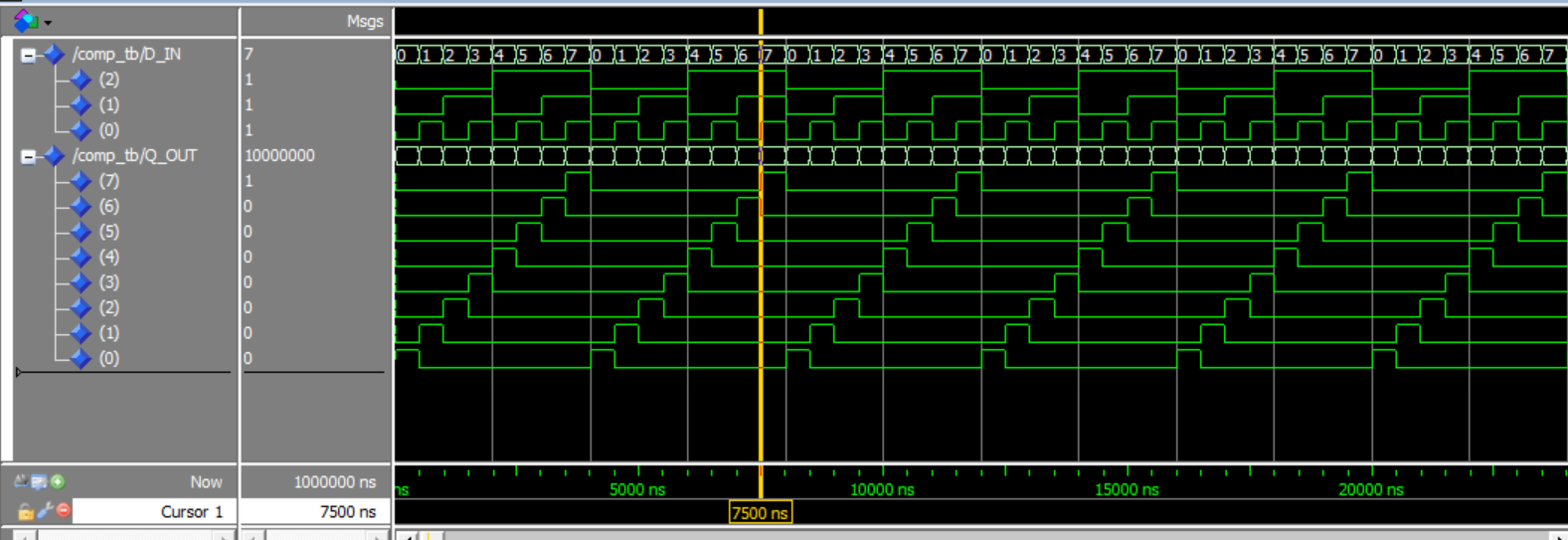
end process ; -- D\_IN\_GEN

---------------------------------------------------------------

end architecture comp\_tb;

---------------------------------------------------------------

（4）实验结果



## 三、实验心得

通过实验一，掌握了quartusⅡ9.1进行编程的基本方法，了解到quartus高版本就是从quartus里面调用modelsim进行仿真，所以思考后决定用modelsim进行3-8译码器的编写。这次试验，相当于是既复习了书本上的知识，又掌握了软件的应用，收获很大。

# 实验二 模十状态机与7段译码器显示

## 一、实验目的：

通过设计频率可选的模十状态机以及7段译码电路以进一步掌握VerilogHDL硬件描述语言。

## 二、实验流程：

本设计有分频器、多路选择器、状态机和译码器。

时钟输入作为分频器的输入，输出时钟分别为2分频、4分频、8分频和16分频；

四个频率的时钟信号由4选1的多路选择器选择其中之一作为状态机的时钟输入；

使用选中的时钟频率作为输入驱动状态机按照以下的次序输出：0->2->5->6->1->9->4->8->7->3->0的顺序输出；

使用此输出作为驱动输入到7段译码器的显示逻辑。

## 三、实验步骤

### 1. 设计思路

（1）、时钟信号 clk 作为分频器的输入，分频器的设计思路为设计一个模十六计 数器，cp0（Q0）输出即为二分频信号，cp1（Q1）输出即为四分频信号，cp2（Q2） 输出即为八分频信号，cp3（Q3）输出即为十六分频信号。分频器的输出由 4 选 1 多 路选择器的选择输入端 select 选择 2 分频、4 分频、8 分频和 16 分频其中之一作为状 态机的时钟输入，当 select 为 0 时，输出为二分频信号；为 1 时，输出为四分频信 号；为 2 时，输出为八分频信号；为 3 时，输出为十六分频信号。

（2）、reset 为高有效，则若 reset 信号为 1 时，qout 置为 0，则 now\_state 为 0。 若 reset 信号为 0 时，qout 自加，并作为状态机的输入驱动，让状态机按照 0->2->5->6->1->9->4->8->7->3->0 的顺序输出。

（3）、状态机按照 0->2->5->6->1->9->4->8->7->3->0 的顺序输出，并使用此输出 作为驱动输入到 7 段译码器的显示逻辑。

### 2. 实验代码：

module compile7

(

input [1:0]sel,

input clk,reset,

output reg[3:0]cp,

output reg[6:0]seg,

output reg fp,

output reg [3:0]now\_state,

output reg [3:0]next\_state

);

always @ (posedge clk or posedge reset)begin

if(reset)

cp<=4'b0000;

else

cp<=cp+4'b0001;

end

always @ (sel) begin

case(sel)

2'b00:

fp=cp[0];

2'b01:

fp=cp[1];

2'b10:

fp=cp[2];

2'b11:

fp=cp[3];

default:fp<=0;

endcase

end

always @ (posedge fp or posedge reset)begin

if(reset)

now\_state<=4'd0;

else

now\_state<=next\_state;

end

always @ (now\_state)begin

case(now\_state)

4'b0000:

next\_state=4'b0010;

4'b0010:

next\_state=4'b0101;

4'b0101:

next\_state=4'b0110;

4'b0110:

next\_state=4'b0001;

4'b0001:

next\_state=4'b1001;

4'b1001:

next\_state=4'b0100;

4'b0100:

next\_state=4'b1000;

4'b1000:

next\_state=4'b0111;

4'b0111:

next\_state=4'b0011;

4'b0011:

next\_state=4'b0000;

default:

next\_state=4'b1111;

endcase

end

always @ (now\_state) begin

case (now\_state)

0:

seg <= 7'b1000000;

1:

seg <= 7'b1111001;

2:

seg <= 7'b0100100;

3:

seg <= 7'b0110000;

4:

seg <= 7'b0011001;

5:

seg <= 7'b0010010;

6:

seg <= 7'b0000010;

7:

seg <= 7'b1111000;

8:

seg <= 7'b0000000;

9:

seg <= 7'b0010000;

default:

seg <= 7'b0001000;

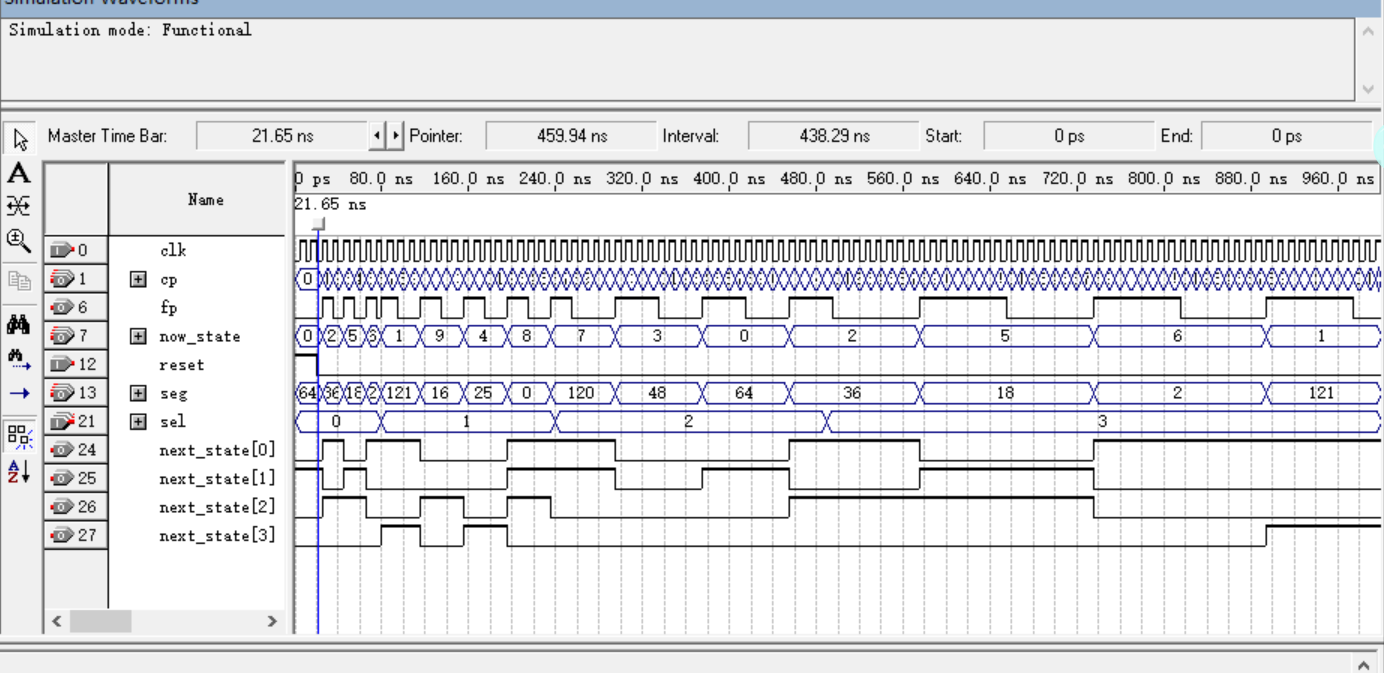
endcase

end

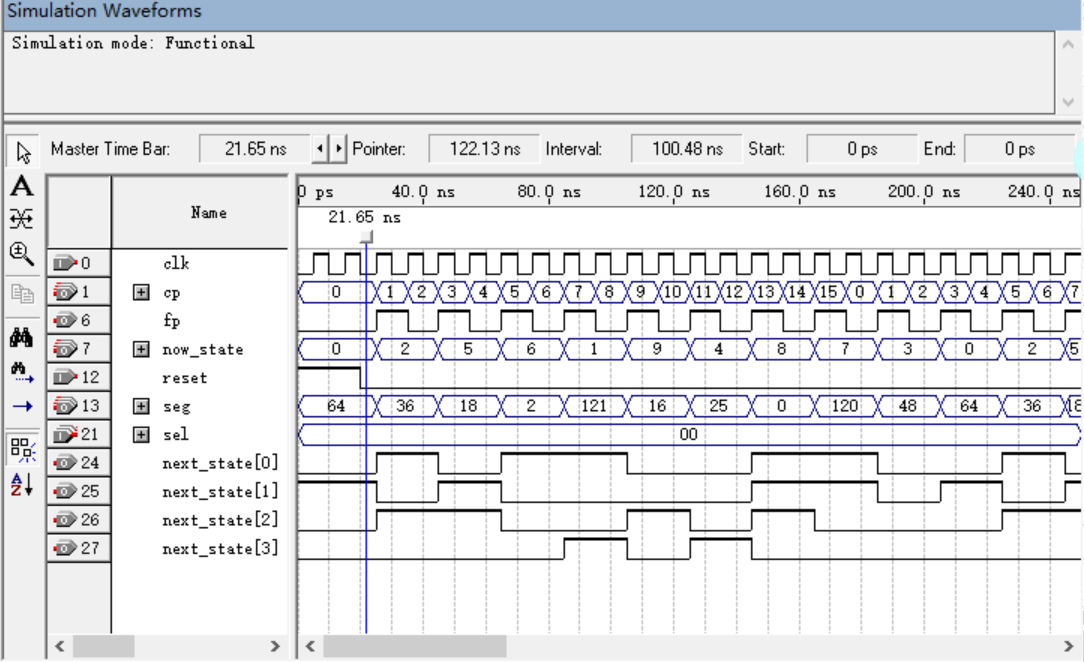
endmodule

### 3. 实验结果：

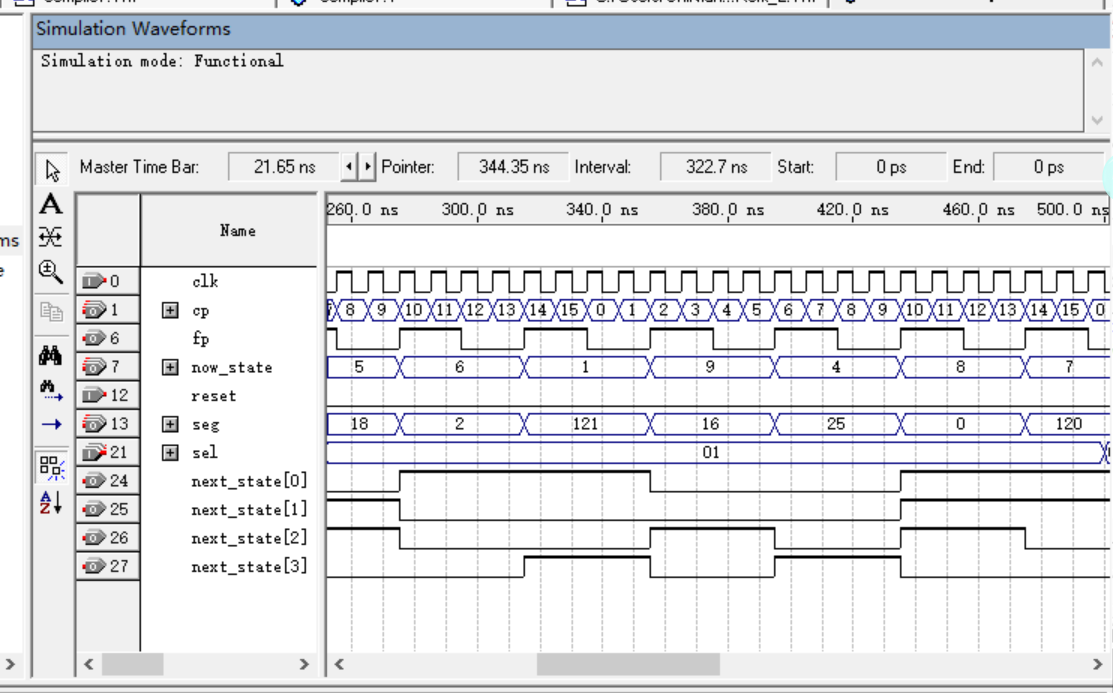
（1）总体结果



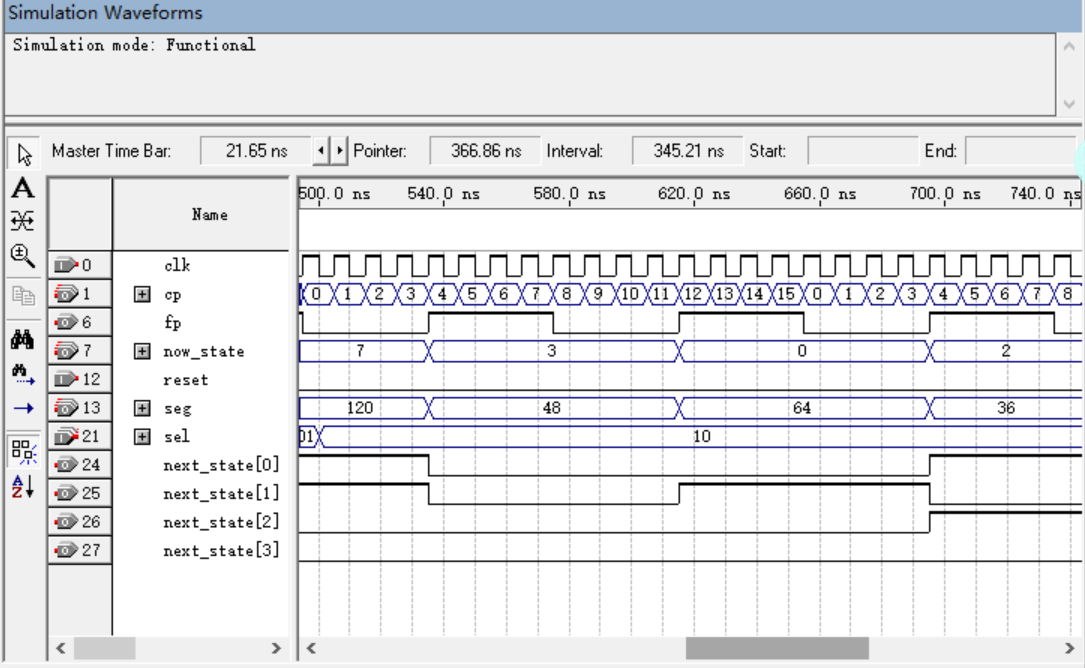
（2）二分频和复位



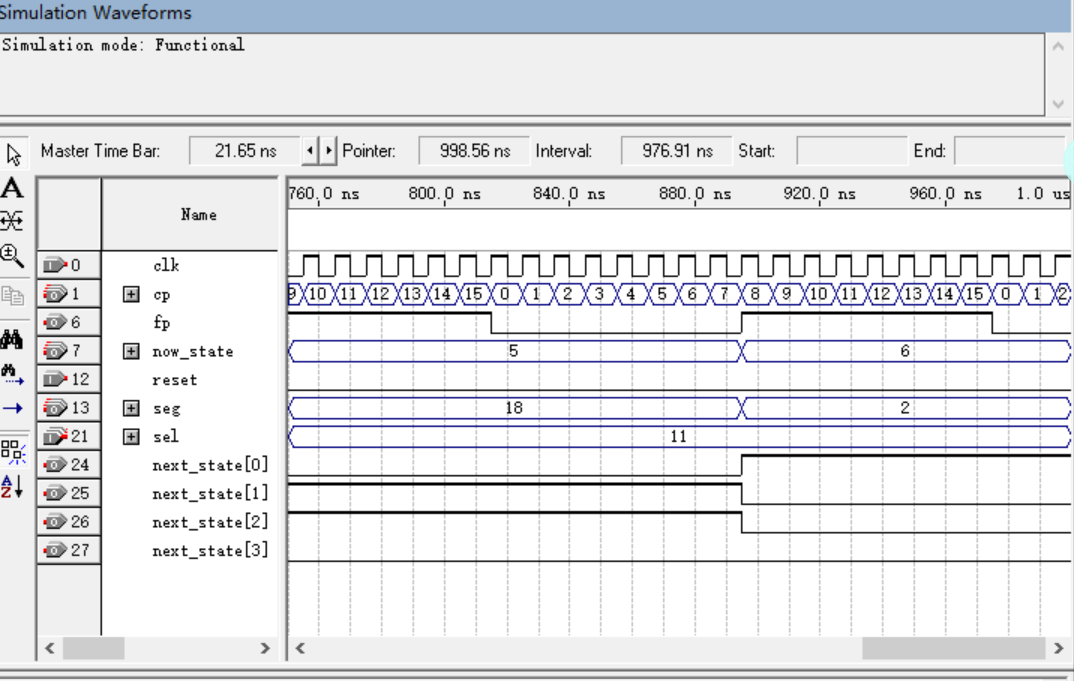
（3）四分频



（4）八分频



（5）十六分频



## 四 实验心得

通过本次设计频率可选的模十状态机以及 7 段译码电路的实验，学会了以代码形式实现频率多路选择器的功能，巩固了模十状态机的实现方式，进一步掌握 VerilogHDL 硬件描述语言。加深了数字电路理论的理解。

# 实验三 数字钟的设计与仿真

## 一、实验目的：

通过设计实现的数字钟的设计与仿真，以熟悉和掌握VerilogHDL语言编程。

## 二、实验流程：

系统整体计数器模块组成。

输入引脚有3+16根，其中三位分别为时钟（提供整个系统的时钟信号）、复位（系统复位信号）和置位信号（用于将时间设置到需要观察的位置）。十六位分别为分钟个位和十位、秒个位和十位。

输出引脚有16根，分别位分钟个位和十位、秒个位和十位。

## 三、实验步骤:

### 1. 设计思路

本实验设计数字钟的实质为设计 60\*60 的计数器。数字钟为上升沿触发，三个 输入管脚为提供整个系统的时钟信号 clk，系统复位信号 clr（高有效）和置位信号 load （高有效），复位信号 clr 用于将输出的分钟个位 min\_l、十位 min\_h 和秒个位 sec\_l、 十位 sec\_h 清 0，置位信号 load 将输出的分钟个位 min\_l、十位 min\_h 和秒个位 sec\_l、 十位 sec\_h 设置到需要观察的位置。十六位分别为分钟个位 min\_l0、十位 min\_h0 和 秒个位 sec\_l0、十位 sec\_h0，为 load 信号变为高电平时的预置信号，此时输出结果 为预置信号。在 load=0，clr=0 时，随着 clk 的上升沿到来，秒个位 sec\_l 进行自加， 加到 9 时下一个 clk 上升沿到来时秒个位 sec\_l 变为 0，sec\_h 加 1。当时钟为 59 秒 是，下一个 clk 上升沿到来时，秒清 0，分钟低位 min\_l 加 1。当时钟为 9 分 59 秒时， 下一个 clk 上升沿到来时，分钟低位 min\_l，秒高位 sec\_h,秒低位 sec\_l 清 0，分钟高 位 min\_h 加 1。当时钟为 59 分 59 秒是，下一个 clk 上升沿到来时，分钟个位 min\_l、 十位 min\_h 和秒个位 sec\_l、十位 sec\_h 均清 0。

### 2. 实验代码

module clock

(

input clk,clr,load,

input [3:0]min\_h0,

input [3:0]min\_l0,

input [3:0]sec\_h0,

input [3:0]sec\_l0,

output reg [3:0]min\_h,

output reg [3:0]min\_l,

output reg [3:0]sec\_h,

output reg [3:0]sec\_l

);

always @(posedge clr or posedge clk) begin

if (clr) begin

min\_h<=0;

min\_l<=0;

sec\_h<=0;

sec\_l<=0;

end

else if(load) begin

min\_h<=min\_h0;

min\_l<=min\_l0;

sec\_h<=sec\_h0;

sec\_l<=sec\_l0;

end

else begin

if (sec\_l==9)begin

sec\_l<=0;

if (sec\_h==5)begin

sec\_h<=0;

if (min\_l==9)begin

min\_l<=0;

if (min\_h==5)begin

min\_h<=0;

end

else begin

min\_h<=min\_h+1;

end

end

else begin

min\_l<=min\_l+1;

end

end

else begin

sec\_h<=sec\_h+1;

end

end

else begin

sec\_l<=sec\_l+1;

end

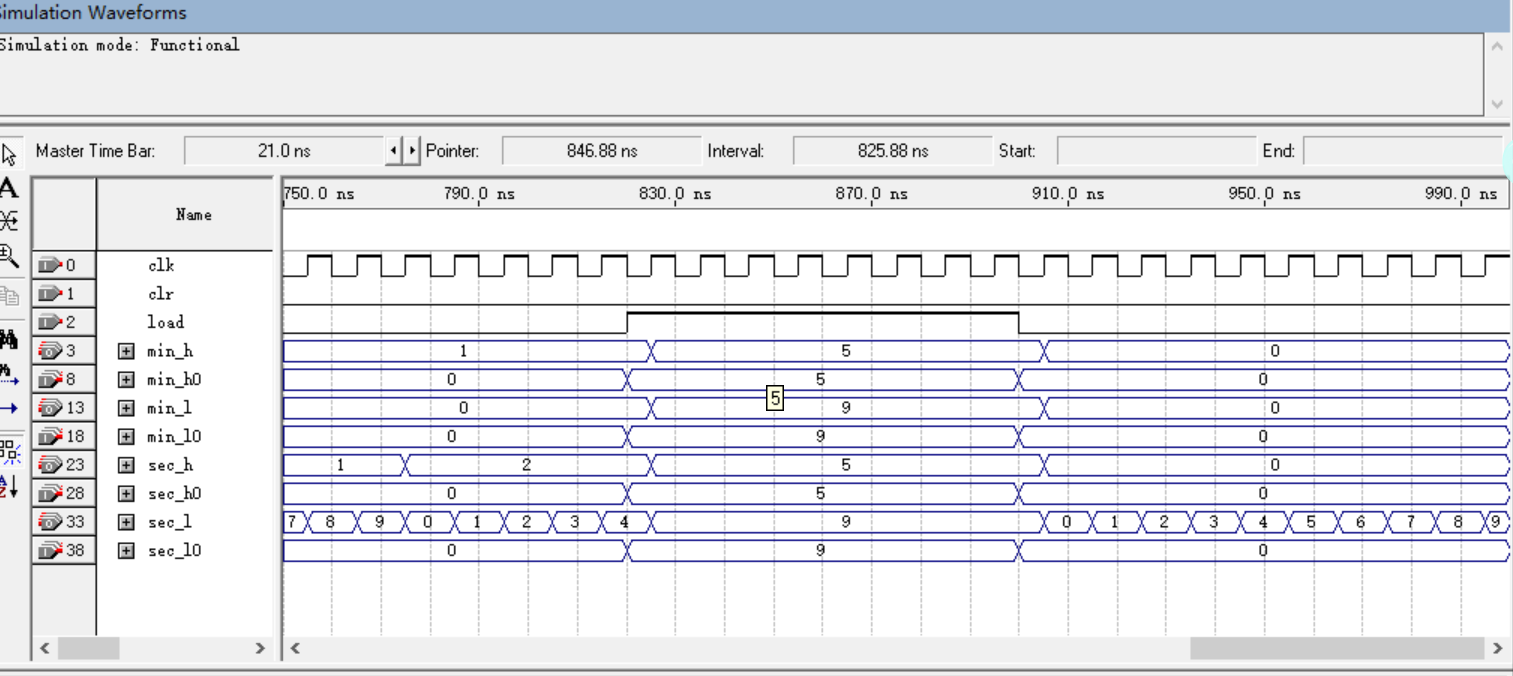
end

end

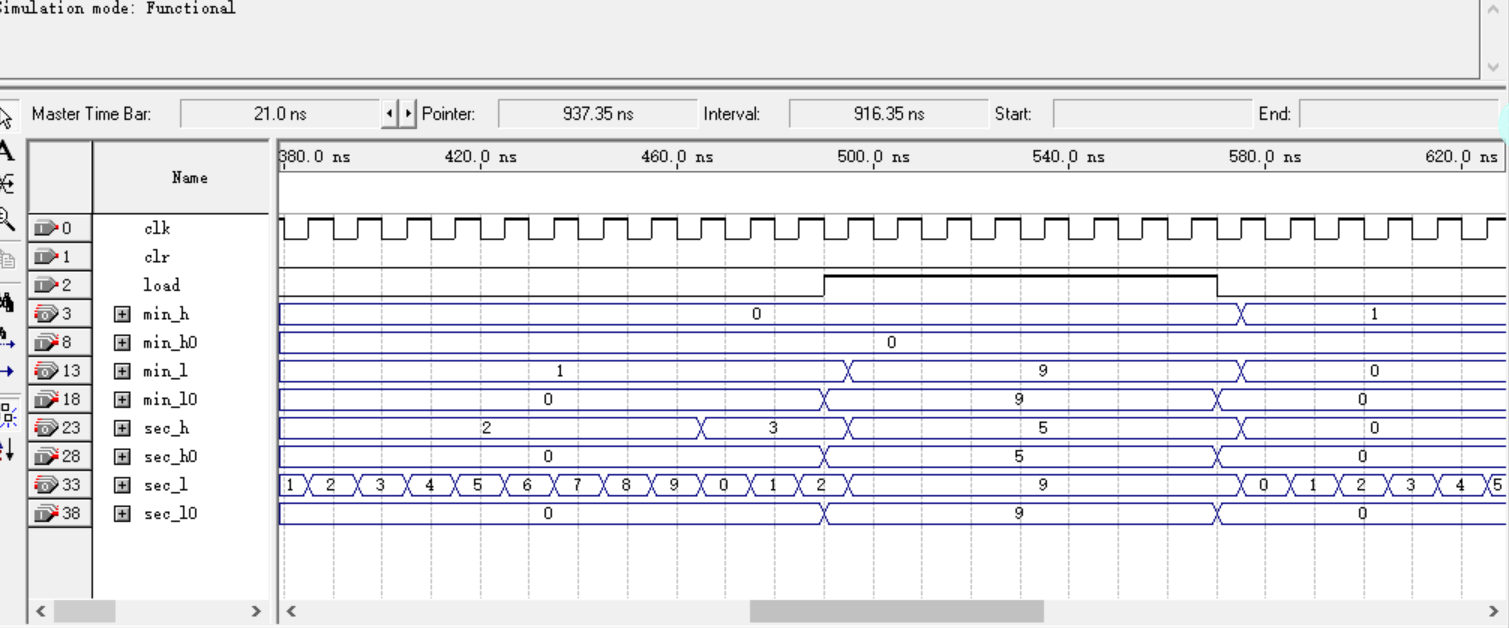
endmodule

### 3. 实验结果

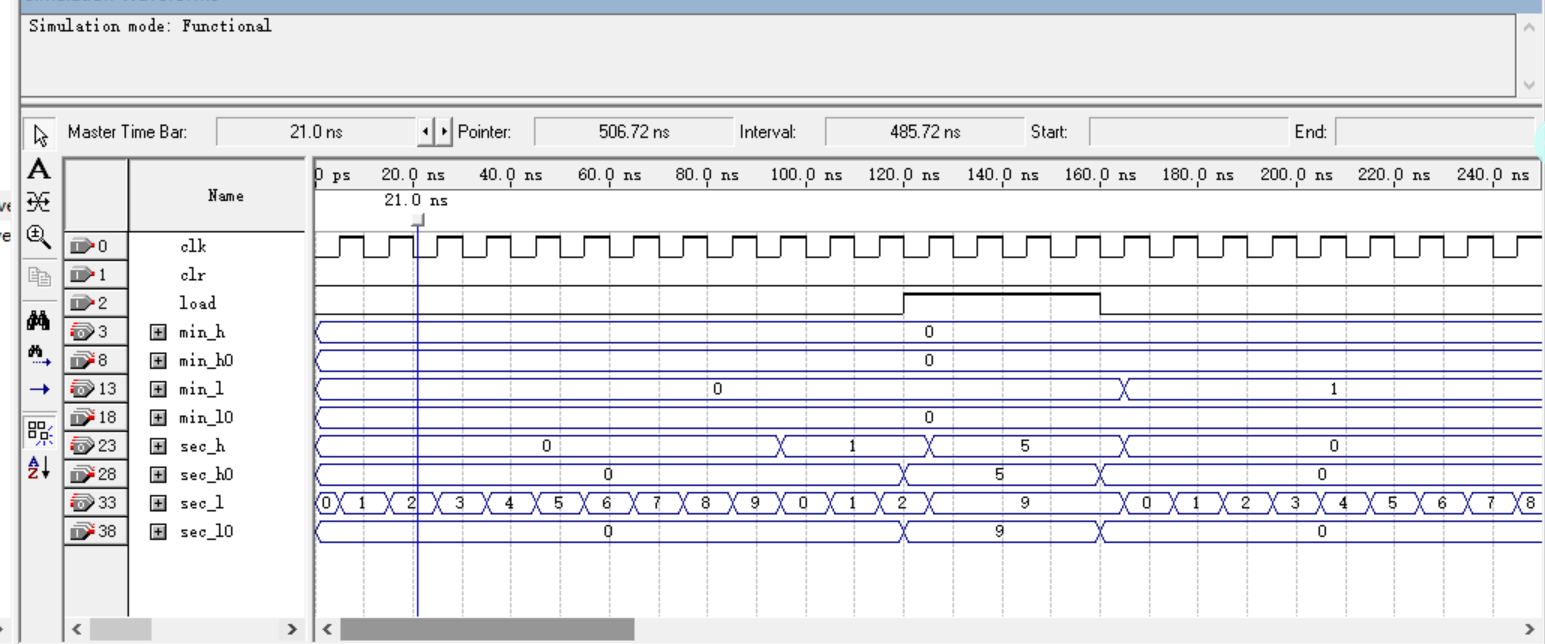
59min59s



9min59s



59s



复位：



分析：

**Load是同步的，复位是异步的。遇到进位时的操作：每当一个时钟有效沿到来时，都要进行是否进位的判断，通过if-else语句，从最低位到最高位层层判断，若不需进位，则相应位数值加一，若需要进位，则该位清零，并进位，用相同操作判断下一位的情况。Clr有效时可以让时钟清零，load有效时能够将输入的时间值置位给时钟，从而也利于仿真观察需要进位时的时钟状态。**

## 四、实验心得:

本次实验中，以代码的形式设计实现的数字钟的功能仿真，通过本次实验对 C语言和 VerilogHDL 语言的相同之处和不同之处有了更深刻的了解和认识，对 always 和 if 语句的使用更加熟练，从而更加熟悉 VerilogHDL 语言编程。同时，对同步和异步有了更直观的见识。