**实验一Quartus 软件与状态机第二部分**

一、实验目的：

（1）掌握软件流程，掌握状态机编程；

（2）掌握时序仿真的基本方法。

（3）掌握时序电路的基本分析方法；

（4）掌握从模型到工程实际的电路变化；

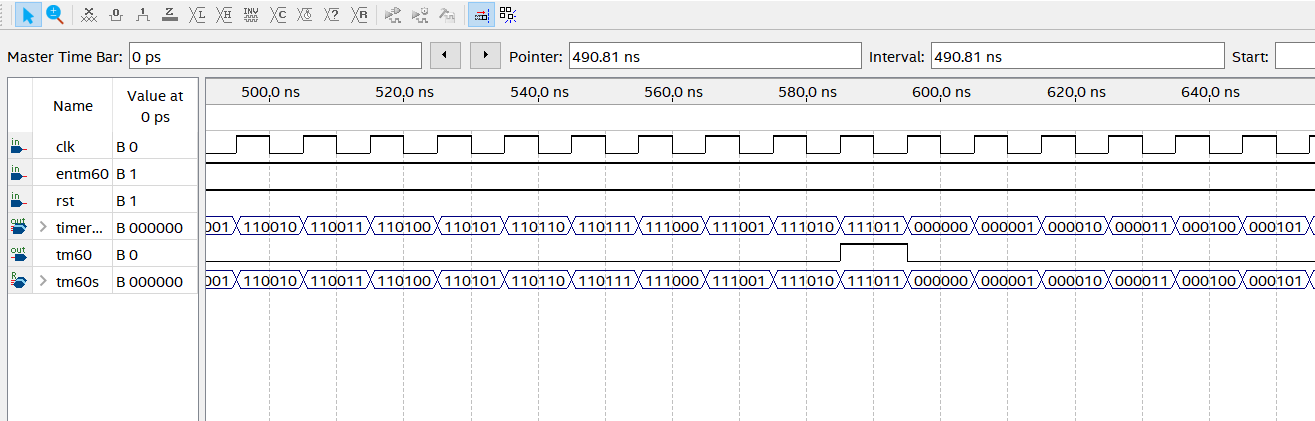
二、实验原理：

状态机设计是数字电路中使用非常广泛和方便的时序设计工具。由于硬件是并行的触发，相对软件是串行执行，那么让硬件电路按照节拍执行串行操作指令就成为一个问题，这就是状态机的主要功能。相应的，软件指令中的几十条简单顺序执行代码可能需要硬件的几十上百个触发器去实现其功能。所以，软件与硬件的设计思路有相当大的区别。

三、实验步骤

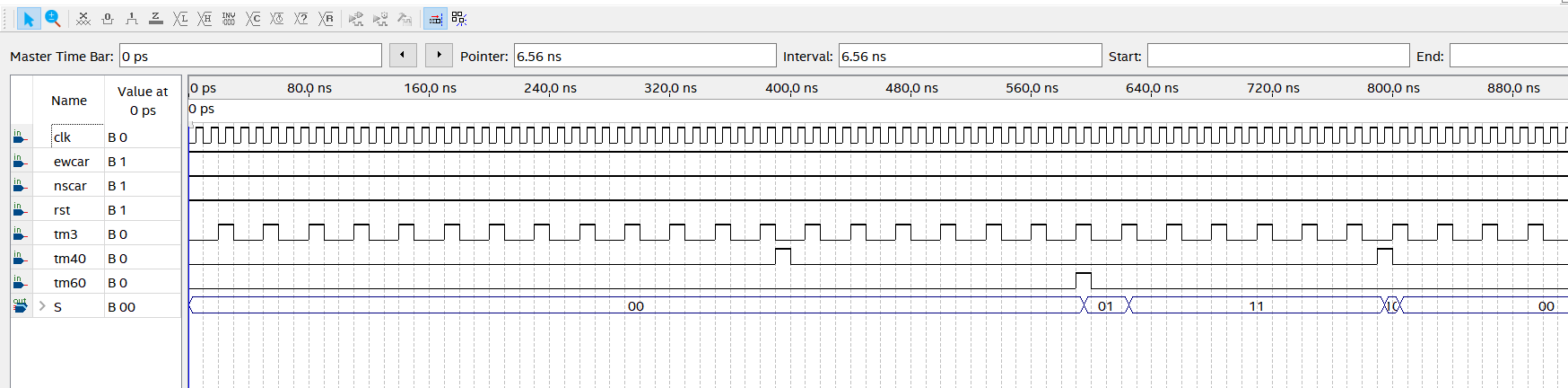
1.编译运行timer60.vhd文件，并添加波形进行仿真。

仿真得到的波形局部如下，分析得知该VHDL描述的电路为一个模60加一计数器，当第60个时钟上升沿来临时，tm60输出1，其他时刻为0.



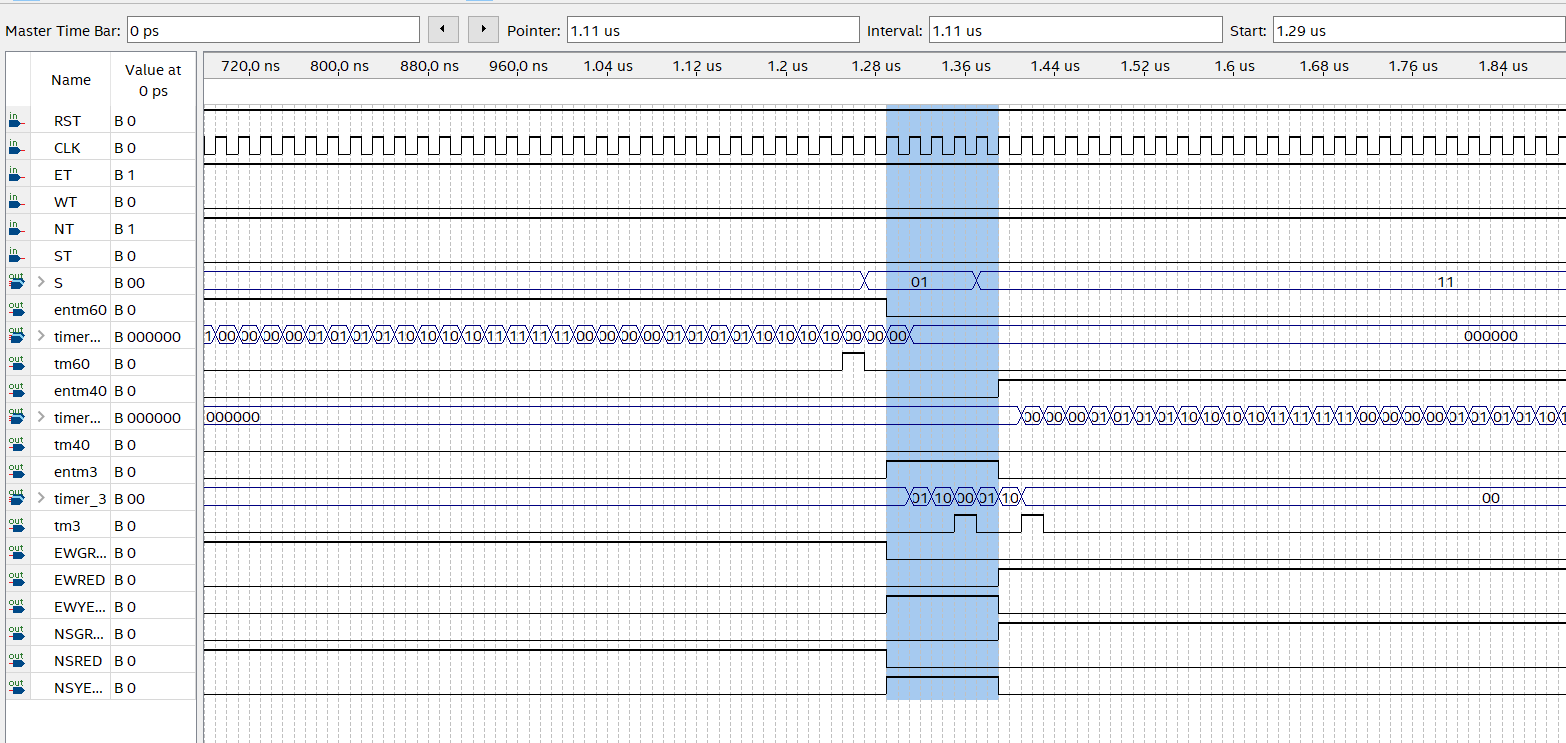
2.编译运行states.vhd文件，并添加波形进行仿真。

状态机的波形仿真如下：



状态机实际就是一组触发器，通过仿真可以看到他的输出，可以看到南北东西都来车时会s存在00（东西方向绿灯），经过01（东西绿灯向南北绿灯转换）状态，到达11（南北方向绿灯）以及相反的转化。

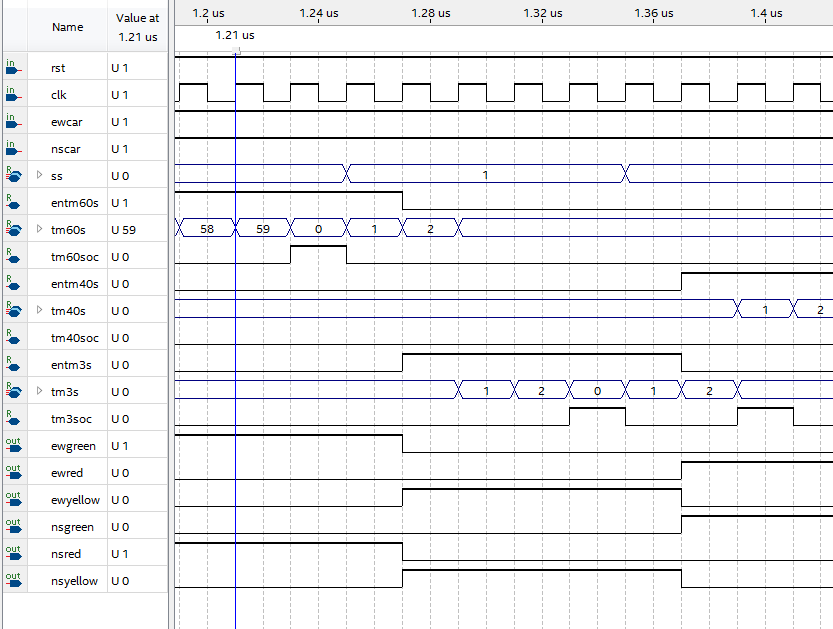
3.编译整个bdf文件并进行仿真。



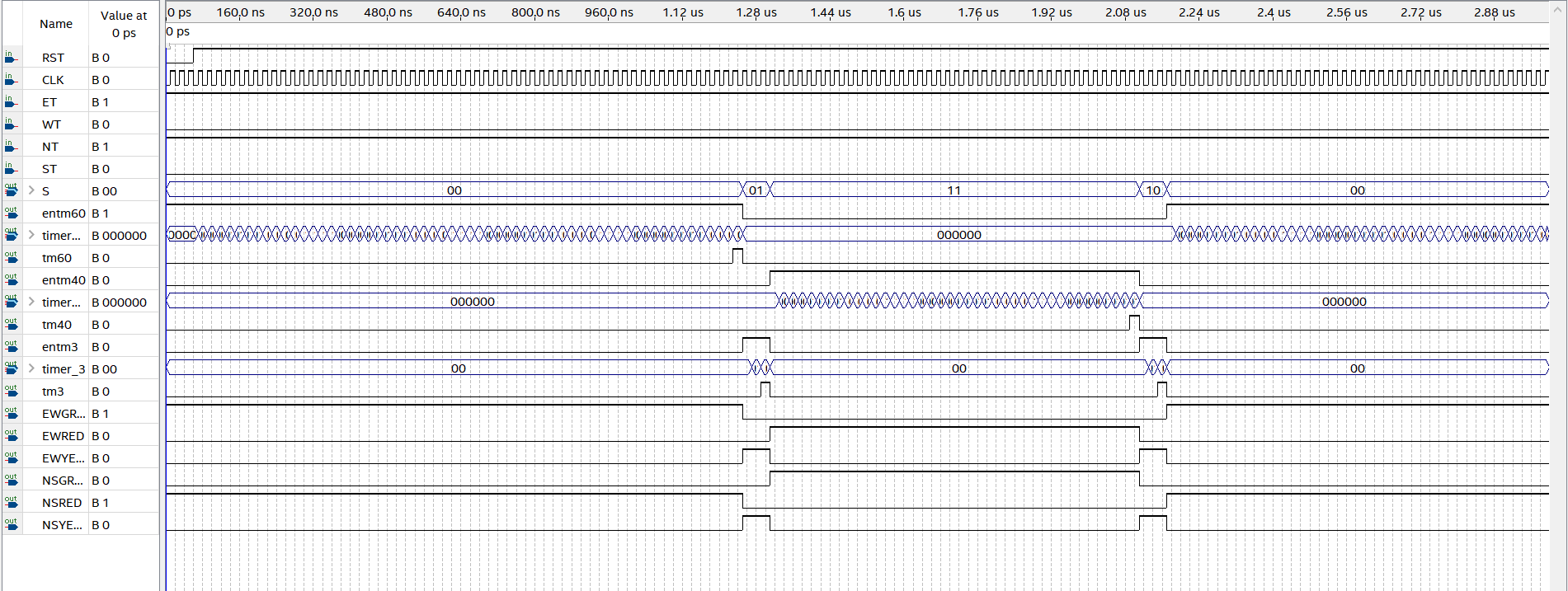
观察仿真图形可以发现，理论上黄灯所占的时钟周期应为3个，但是当东西向绿灯转黄灯之后的，所占的时钟周期为5个，出现了两个时钟周期的延时，需要找到存在错误的原因。

4.改正该工程代码，实现正确的交通灯模型；

可以看到，timer计时器、states均由代码实现。但是其输出却经过了D触发器这一时序逻辑器件，根据D触发器的内部结构，其输入与输出之间必然存在三级与非门延迟，因此按照时钟周期的来临分析如下仿真图形，可以得知正是由于D触发器内部的三级与非门延迟导致了绿灯亮起多了两个时钟周期。



找到原因之后我们发现，只要将造成延时的逻辑器件D触发器删除后，重新进行仿真，就消除了三级与非门延迟带来的影响，电路仿真结果和理论分析值一致。东西向绿灯转黄灯之后的，所占的时钟周期减少为3个。



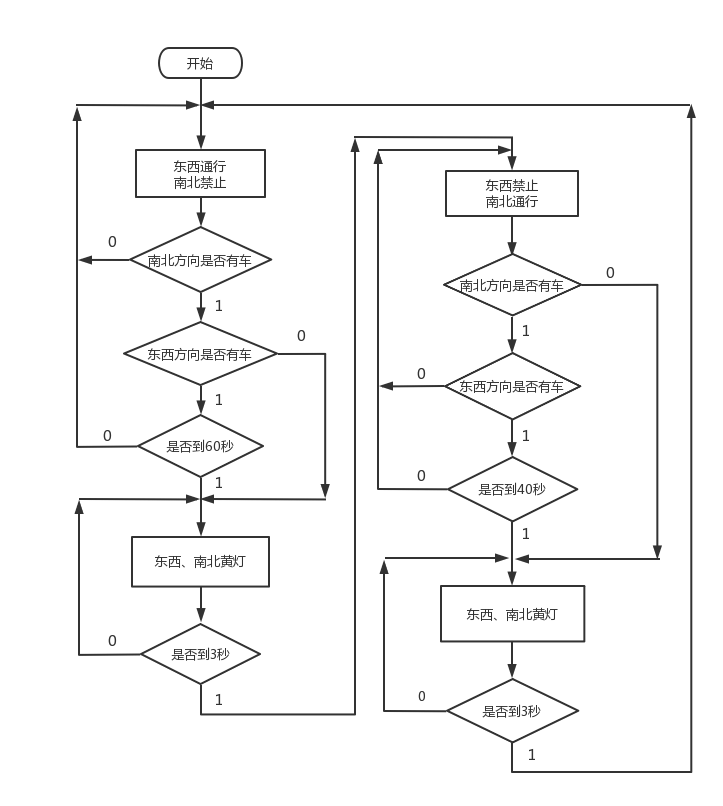
5.回答实验课件中提出的问题

仿真应当按照timer计数器、states状态机、tlight28\_sim进行。

状态转化时间的问题和多出的时钟周期的原因均为D触发器存在三级与非门延时，当时钟周期的上升沿来临时D触发器的输出存在演示并不能立刻送达下一级，因此造成错误。解决方法理论上只要删去D触发器即可。

电路中存在多个计数器，可以合并为一个可变模的计数器。

设计红绿灯框图。



四、思考题

回答可变模计数器的“条件甲”

完整的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 timer is

port (

rst : in std\_logic;

clk : in std\_logic;

entm60: in std\_logic;

tm60: out std\_logic;

entm40: in std\_logic;

tm40: out std\_logic;

entm3: in std\_logic;

tm3: out std\_logic

);

end timer;

architecture timer\_behav of timer is

signal tms :std\_logic\_vector(5 downto 0);

begin

process(rst,clk)

begin

if rst = '0' then

tms <= B"000000";

elsif clk'event and clk = '1' then

if entm60 = '1' or entm40 = '1' or entm3 = '1' then

if tms = 59 then

tms <= B"000000";

else

tms <= tms + 1;

end if;

else

tms <= B"000000";

end if;

end if;

end process;

tm60 <= '1' when tms = 59 else '0';

tm40 <= '1' when tms = 39 else '0';

tm3 <= '1' when tms = 2 else '0';

end timer\_behav;

只有当三个entm均为1，才进行下一次完整计数，仿真结果如下：

