



**实验报告**

课程名称\_\_EDA技术 \_\_\_\_\_\_\_\_\_\_\_

实验名称\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学生学院\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

专业班级\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学 号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学生姓名\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2023 年 月 日

# EDA课程设计

**课程设计题目一：基于Verilog的跑马灯设计**

**设计目的：**

设计一个能够有多种工作模式控制的8个灯亮灭的电路。

**设计内容：**

**程序设计要求：**

设计一个包含四个工作状态的状态机，每种工作模式重复两次，并在2分钟内完成所有工作模式。

工作模式以及内容如下：

工作模式1：按照从左到右的方向，依次点亮每一盏灯，然后依次熄灭每一盏灯；

工作模式2：分成两组灯，前四个灯为1组，后四个为2组，1组灯按从左到右依次点亮，同时2组灯按从右到左依次点亮，然后两组灯按各自点亮的顺序依次熄灭；

工作模式3：用11110000作为一组灯的序列，按照该顺序完成8盏灯亮灭：即首先灯1亮，然后灯2亮，然后灯3亮，然后灯4亮，然后灯5不亮，然后灯6不亮，然后灯7不亮，然后灯8不亮，然后八个灯同时变成亮，亮，亮，亮，不亮，不亮，不亮，不亮，并保持下去。

工作模式4：与工作模式1相同，但初始序列列为11111111。

设计源文件模块输入信号：

选择信号[1:0]S, 时钟信号clk, 复位信号rst

设计源文件模块输出信号：

跑马灯亮灭信号[7:0]Y

Verilog源文件名：

Running\_Light.v

Verilog Testbench名：

Running\_Light\_top.v

仿真软件：

Vivado Modelsim

设计要求分析：

由数电知识可以知道LED即发光二极管，可以通过共接阴阳极的方式来实现高电平/低电平控制，本次实验要输出发光的模型是共阴极LED模型，即“0000 0000”为LED全暗，“1111 1111”为LED全亮。我们设从LED左到右的对应输出Y最高位到最低位。

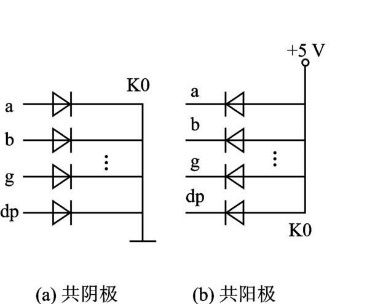
****

图1 LED两种共极连接方式的电路原理图

要实现四个工作模式的转变，首先在模块块内定义一个三位的寄存器[2:0]sel,一个作用是接收输入的S，另一个是使用最高位来判别是否需要再循环一次。如果输入的S为00，则在运行完一次工作状态1的块程序后，状态机会根据另一个寄存器(model\_repeat)的值来判定是否需要重复。若model\_repeat的值为0，则给其+1进位后状态跳转至暂存态(model\_temp1,3’b100)后再跳转回Model\_1，再一次运行完一次工作模式1的块程序后，跳转至工作模式2。至完成工作模式1-4后各两次后，跳转回工作状态1。

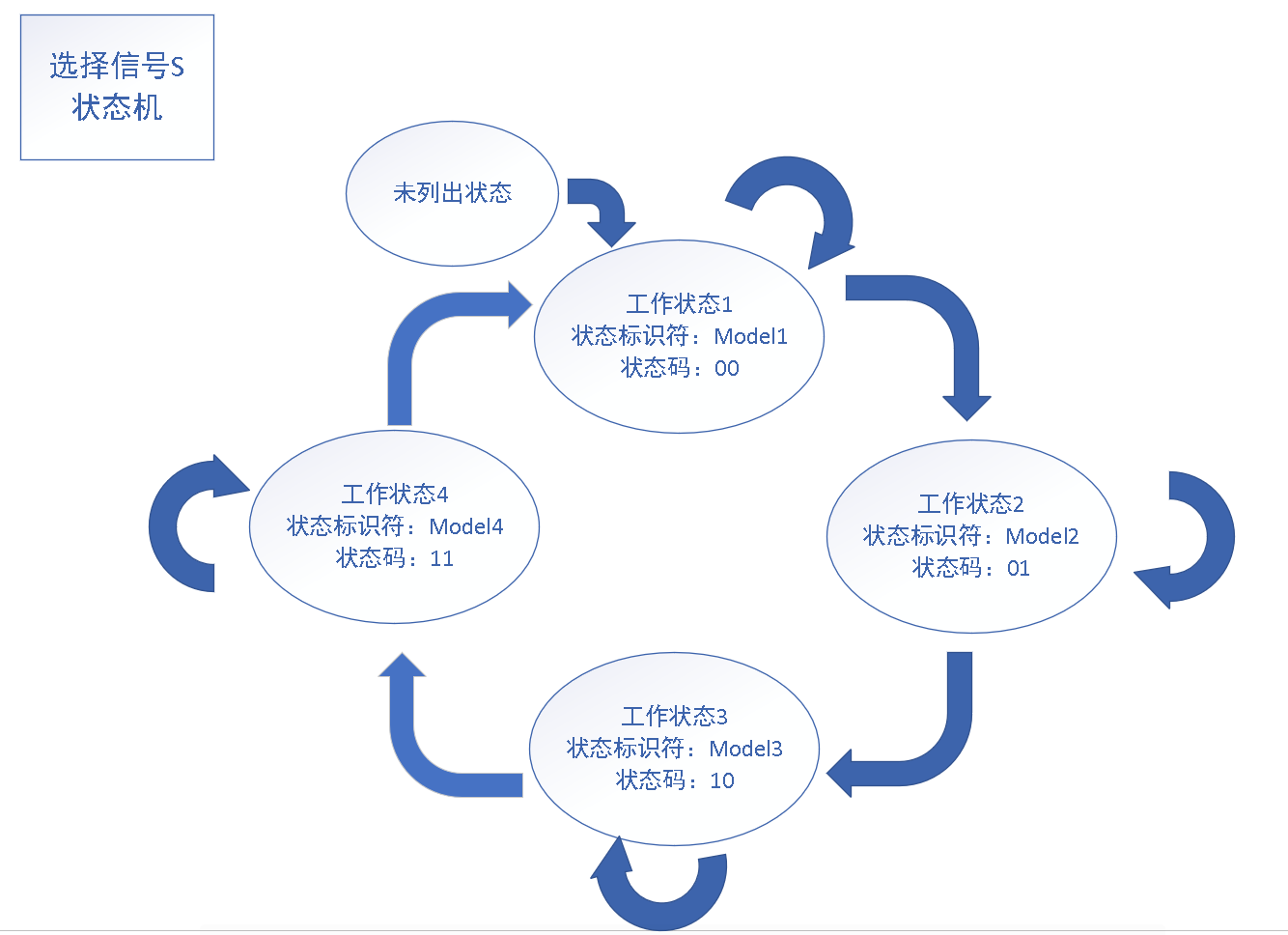
****

图2 选择信号S状态机状态转移示意图

**设计内容：**

工作状态1分析：

首先，进入工作状态1，输出初值为8’b0000 0000,可以知道要实现8个灯点灭，要进行16次操作，前8次和后8次分别均为点灯和灭灯.故可以在块外定义一个8位寄存器[7:0] state\_exist，state\_exist由计数器块决定，当state\_exist 在0-7之间时，易知使用“Y\_out[7-state\_exist]<=1;”即可完成从左往右点灯操作，当state\_exist 在8-15之间时，需要灭灯只需将上式改成“Y\_out[15-state\_exist]<=0;”即可实现从左往右灭灯。

工作状态2分析：

从工作状态1进入工作状态2，输出初值为8’b0000 0000,可以知道要实现8个灯每次两个灯点灭，只需进行8次操作，前4次和后4次分别均为点灯和灭灯。故继续使用state\_exist。当state\_exist 在0-3之间时，易知使用“”和“Y\_out[7-state\_exist]<= 1;”即可完成从两侧往中间点灯操作.当state\_exist 在4-7之间时，需要灭灯只需改成“Y\_out[state\_exist-4]<=0和“ Y\_out[11-state\_exist]<=0;”即可完成从两侧往中间灭灯操作。

工作状态3分析：

从工作状态2进入工作状态3，输出初值为8’b1111 0000, 可以知道要实现8个灯点灭，要进行8次操作，前四次和后四次分别均为点灯和灭灯，故继续使用state\_exist当state\_exist 在0-3之间时，易知使用“Y\_out[7-state\_exist]<=1;”即可完成从左往右点灯操作，当state\_exist 在4-7之间时，需要灭灯只需将上式改成“Y\_out[15-state\_exist]<=0;”即可实现从左往右灭灯。

工作状态4分析：

从工作状态3进入工作状态4，输出初值为8’1111 1111,可以知道要实现8个灯点灭，要进行16次操作，前8次和后8次分别均为灭灯和点灯，故继续使用state\_exist当state\_exist 在0-7之间时，易知使用“Y\_out[7-state\_exist]<=0;”即可完成从左往右灭灯操作，当state\_exist 在8-15之间时，需要灭灯只需将上式改成“Y\_out[15-state\_exist]<= 1;”即可实现从左往右点灯。

计数器分析：

在主always块外，再另建一个always作为计数器。同主always块一样clk上升沿触发，rst下降沿触发。当rst高电平时，开始计数，计数Count\_time时间后给state\_exist加1再将Count\_time复位为0。当主触发块不再需要state\_exist再增加时，将count\_rst置1，即使计数器复位。

RTL图：

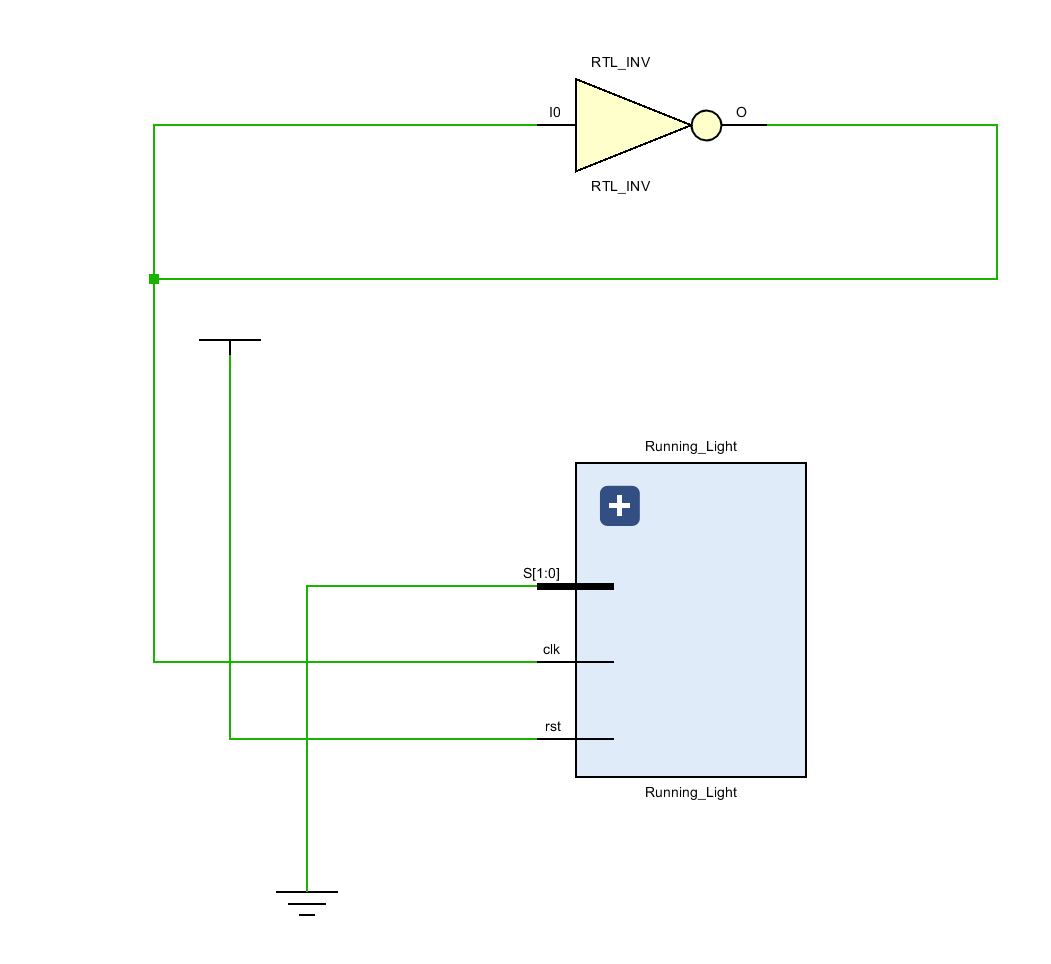


图3 顶层模块RTL图

可以看到顶层模块的RTL图显示，Running\_Light输入为三个端口，即clk，S[1:0],rst。时钟信号是由外界激励不断反向产生的结果。

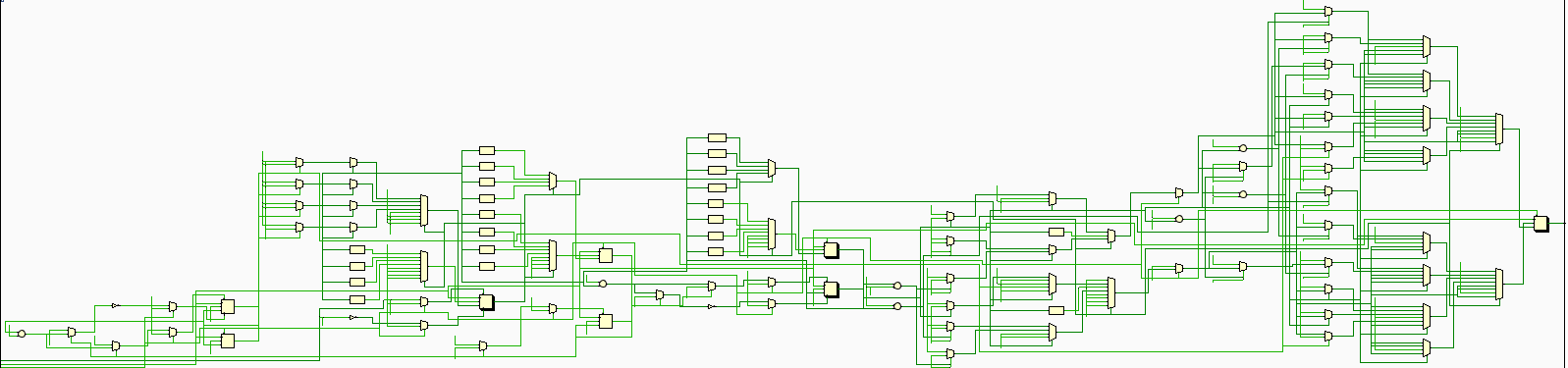


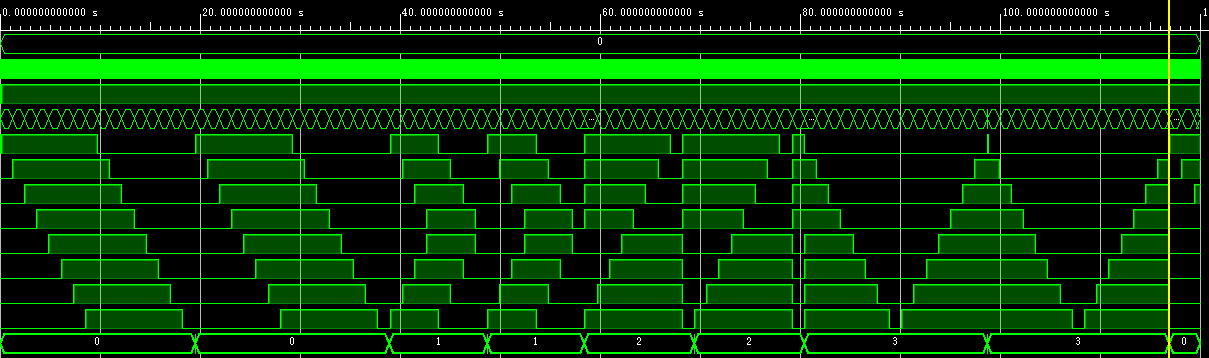
图4 Running\_Light模块RTL图

可以看到Running\_Light模块的RTL图，显示模块使用了较多的寄存器和多路选择器和触发器，可能存在较高的延迟。

**设计结果分析**

**仿真整体结果分析：**

Vivado**仿真：**

****

**图5** Vivado**仿真 四个工作模式波形图**

可以看到Vivado仿真出的四个工作模式的波形图，不难看到完成四个工作模式所需时间约为116 s，四个工作模式均重复两次。

**Modelsim仿真：**

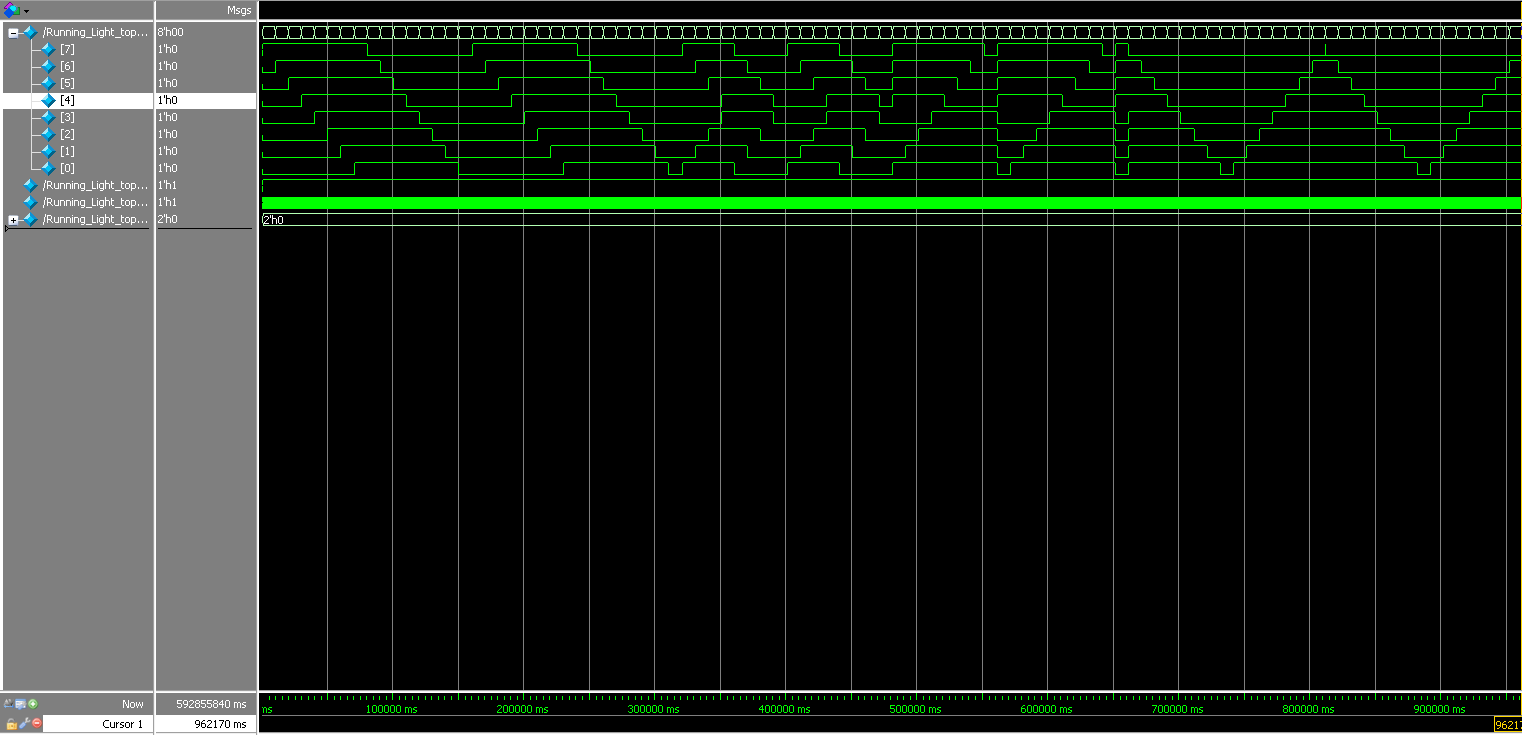
****

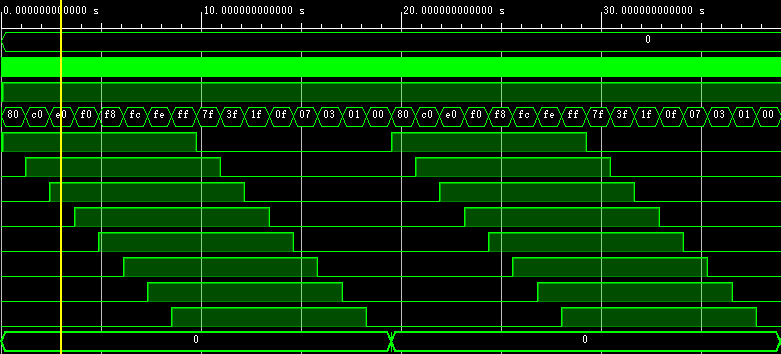
图6 ModelSim仿真 四个工作状态波形图

可以看到ModelSim仿真出的四个工作模式的波形图，不难看到完成四个工作模式所需时间约为96.2 s。相较于Vivado仿真的116s，推测可能是Vivado仿真使用的是虚拟FPGA资源，ModelSim直接使用计算机资源。

**工作模式分析**

各工作模式形图从上到下分别为时钟信号clk，复位信号rst，输出信号Y[7:0](从上到下分别是7-0)和选择信号[2:0]sel。

**工作模式1工作结果分析：**



**图7工作状态1波形图**

工作模式1最初输出为8’d0000\_0000，然后是变成8’d1000\_0000，实现了最高位灯亮，即最左边灯亮。再然后是变成8’d1100\_0000，即次最左灯亮。以此类推，到最右侧灯亮，即输出变为8’d1111\_1111时，全部灯都被点亮。完成点灯后是灭灯操作，首先是Y\_out[7]转变成0，即最左侧灯灭。也可以以此类推，到最右侧灯灭，输出变为8’d0000\_0000，即全部灯灭。在完成一次工作模式1后，选择信号跳变到3’b100(即暂存态Model\_temp1)后经过一个时钟周期，在变成工作模式1，再进行一次工作模式1的内容。

**工作模式2工作结果分析：**

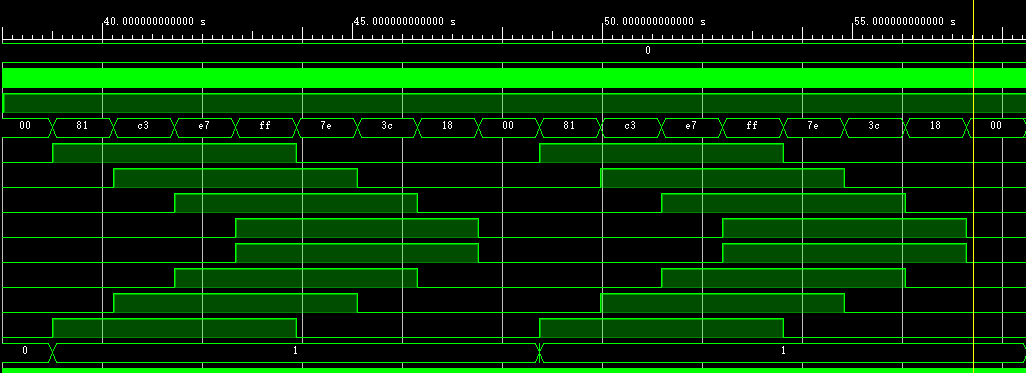


图8 工作状态2波形图

工作模式2最初输出也为8’d0000\_0000，因为只需进行8次点灭，所以与工作模式1有所不同。首先第一次点灯是变成8’d1000\_0001，实现了最高位和最低位灯亮，即最左边和最右边灯亮。再然后是变成8’d1100\_0011，即次最左灯和次最右灯亮。以此类推，到中间两灯亮，即输出变为8’d1111\_1111时，全部灯都被点亮。完成点灯后是灭灯操作，首先是Y\_out[7]和Y\_out[0]转变成0，即最左边和最右边灭。也可以以此类推，到最中间两灯灭，输出变为8’d0000\_0000，即全部灯灭。在完成一次工作模式2后，选择信号跳变到3’b101(即暂存态Model\_temp2)后经过一个时钟周期，再变成工作模式2，再进行一次工作模式2的内容。

**工作模式3工作结果分析：**

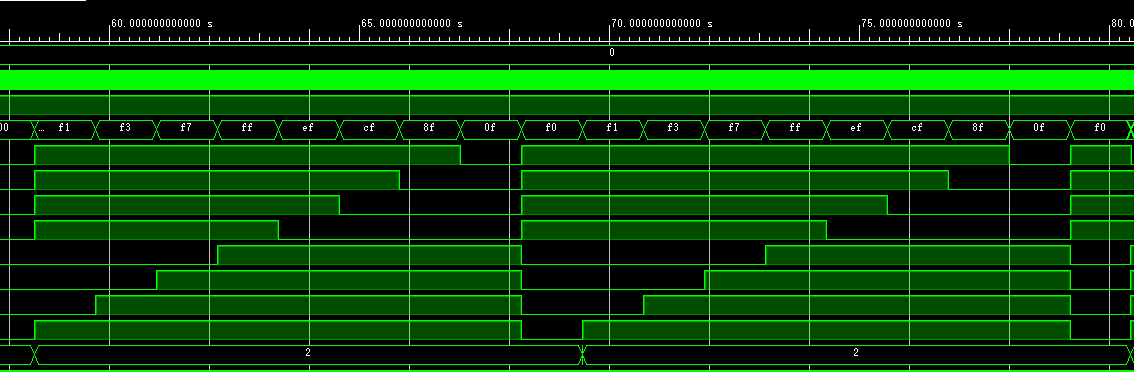
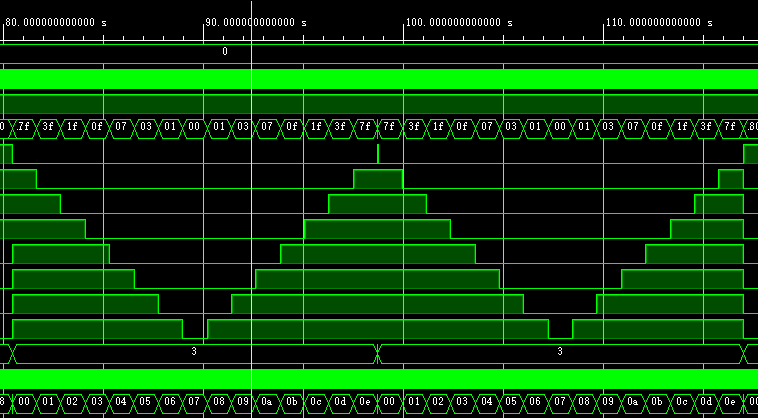


图9 工作状态3波形图

工作模式1最初输出为8’d1111\_0000，即左边四个灯全亮，右侧四个灯全灭，然后是变成8’d1111\_0001，实现了最低位灯亮，即最右边灯亮。再然后是变成8’d1111\_0011，即次最右灯也点亮。以此类推，到中心靠右侧灯亮，即输出变为8’d1111\_1111时，全部灯都被点亮。完成点灯后是灭灯操作，首先是Y\_out[4]转变成0，即8’d1110\_1111，也就是中心靠左侧灯灭。也可以以此类推，到最左侧灯灭，输出变为8’d0000\_1111，即左边四个灯全灭，右侧四个灯全亮，然后瞬间变回8’d1111\_0000，即左边四个灯全亮，右侧四个灯全灭。在完成一次工作模式3后，选择信号跳变到3’b110(即暂存态Model\_temp3)后经过一个时钟周期，再变成工作模式3，再进行一次工作模式3的内容。

**工作模式4工作结果分析：**

****

**图10 工作状态4波形图**

工作模式4最初输出为8’d1111\_1111，然后是变成8’d0111\_1111，实现了最高位灯灭，即最左边灯灭。再然后是变成8’d0011\_1111，即次最左灯灭。以此类推，到最右侧灯灭，即输出变为8’d0000\_0000时，全部灯都被熄灭。完成灭灯后是点灯操作，首先是Y\_out[7]转变成1，即最左侧灯亮。也可以以此类推，到最右侧灯亮，输出变为8’d1111\_1111，即全部灯亮。在完成一次工作模式4后，选择信号跳变到3’b111(即暂存态Model\_temp4)后经过一个时钟周期，在变成工作模式4，再进行一次工作模式4的内容。

**课程设计题目二：基于Verilog的十字路口交通灯控制电路设计**

**设计目的：**

设计并实现一个简单的十字路口交通灯控制电路。

**设计内容：**

**程序设计要求：**

以4个红色指示灯、4个绿色指示灯和4个黄色指示灯模拟路口东西南北4个方向的红绿黄交通灯。控制这些灯，使它们安下列规律亮灭。

工作模式以及内容如下：

1、东西方向绿灯亮，南北方向红灯亮。东西方向通车，时间30秒；

2、东西方向黄灯闪烁，南北方向红灯亮，时间2秒。

3、东西方向红灯亮，南北方向绿灯亮。南北方向通车，时间30秒；

4、东西方向红灯亮，南北方向黄灯闪烁，时间2秒。

5、返回1，继续运行。

设计源文件模块输入信号：

选择信号[1:0]S, 时钟信号clk, 复位信号rst

设计源文件模块输出信号：

跑马灯亮灭信号[7:0]Y

Verilog源文件名：

Traffic\_lights.v

Verilog Testbench名：

Traffic\_lights\_top.v

仿真软件：

Vivado ModelSim

设计要求分析：



图1 交通灯的物理模型图

由日常知识可以知道，交通灯是由三个LED模块组合而成的，我们可以这样认为每一个方向的交通灯都可以用一个三位寄存器表示，每一个颜色的灯都可以认为是寄存器的其中的一位的高电平表示。可以做出如下定义，红灯对应的是3'b001,黄灯对应的是3'b010,绿灯对应的是3'b100,不工作状态对应的是Idle=3'b000。要限制四个方向灯的亮灭，我们需要另外定义对应四个方向的四个三位寄存器，分别是面北的[2:0]North\_Lights，面南的[2:0]South\_Lights,面西的[2:0]West\_Lights,面东的[2:0]East\_Lights，以及通过计数器分频控制状态持续时间。

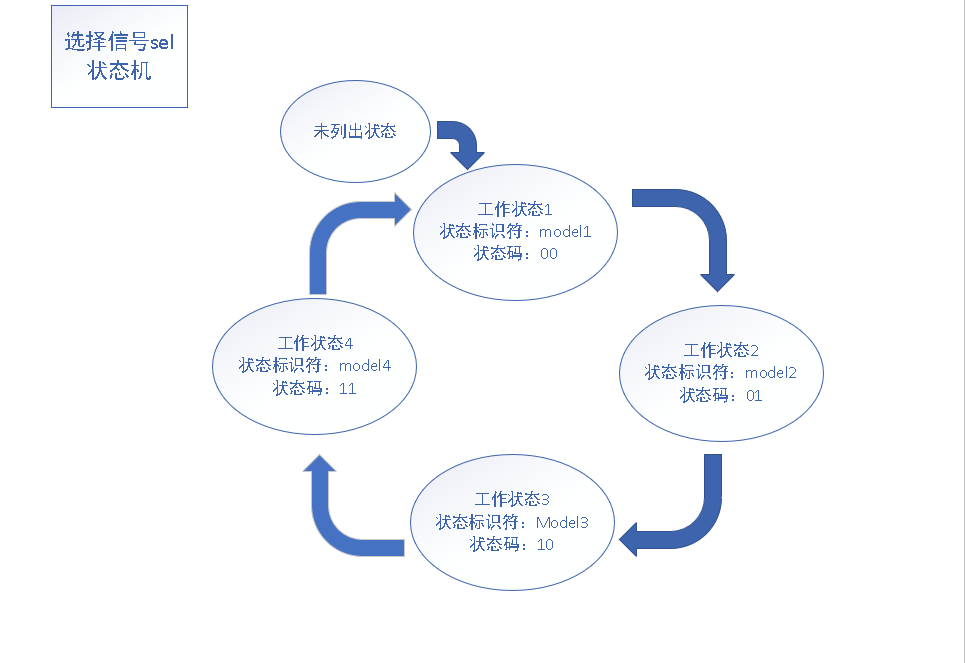
****

图2 选择信号sel状态机状态转移示意图

要实现五个工作内容的转变，只需实现一个四个状态的状态机。即，首先在模块块内定义一个两位的寄存器[2:0]sel,作用是接收输入的S和更变状态。如果输入的S为00，则在运行完一次工作状态1的块程序后，跳转至工作模式2，执行工作模式2后，跳转到至工作模式3，执行工作模式4后，跳转至工作模式4，执行工作模式4后跳转回工作模式1。

**设计内容：**

工作状态1分析：

首先，进入工作状态1，由不工作状态变成为面东的灯变绿（East\_Lights<=green），面西的灯变绿（West\_Lights<=green），面北的灯变红（North\_Lights<=red），面南的灯变红（South\_Lights<=red）。此过程持续30s，完成后，跳转至工作状态2，面东，面西灯进入不工作状态。

工作状态2分析：

进入工作状态2，由工作状态1变成为面东的灯变黄且闪烁East\_Lights[1]<=East\_Lights[1]^1；），面西的灯也变黄且闪烁即有（West\_Lights[1]<=West\_Lights[1]^1;），面北的灯一直为保持为红灯（North\_Lights<=red），面南的灯也一直为保持为红灯（South\_Lights<=red）。此过程持续2s，完成后，跳转至工作状态3。

工作状态3分析：

进入工作状态3，由工作状态2变成为面东的灯变红（East\_Lights<= red），面西的灯变红（West\_Lights<=red），面北的灯变绿（North\_Lights<= green），面南的灯变绿（South\_Lights<= green）。此过程持续30s，完成后，跳转至工作状态4，面东，面西灯进入不工作状态。

工作状态4分析：

进入工作状态4，由工作状态3变成为面东的灯一直保持为红灯（East\_Lights<= red），面西的灯也一直保持为红灯即有（West\_Lights<=red;），面北的灯变黄且闪烁（North\_Lights[1]<=North\_Lights[1]^1），面南的灯也变黄且闪烁（South\_Lights[1]<=South\_Lights[1]^1）。此过程持续2s，完成后，跳转回工作状态1。

计数器分析：

在主always块外，再另建一个always作为计数器。同主always块一样clk上升沿触发，rst下降沿触发。当rst高电平时，开始计数，计数Count\_time时间后给state\_exist加1再将Count\_time复位为0。当主触发块不再需要state\_exist再增加时，将count\_rst置1，即使计数器复位。

RTL图：

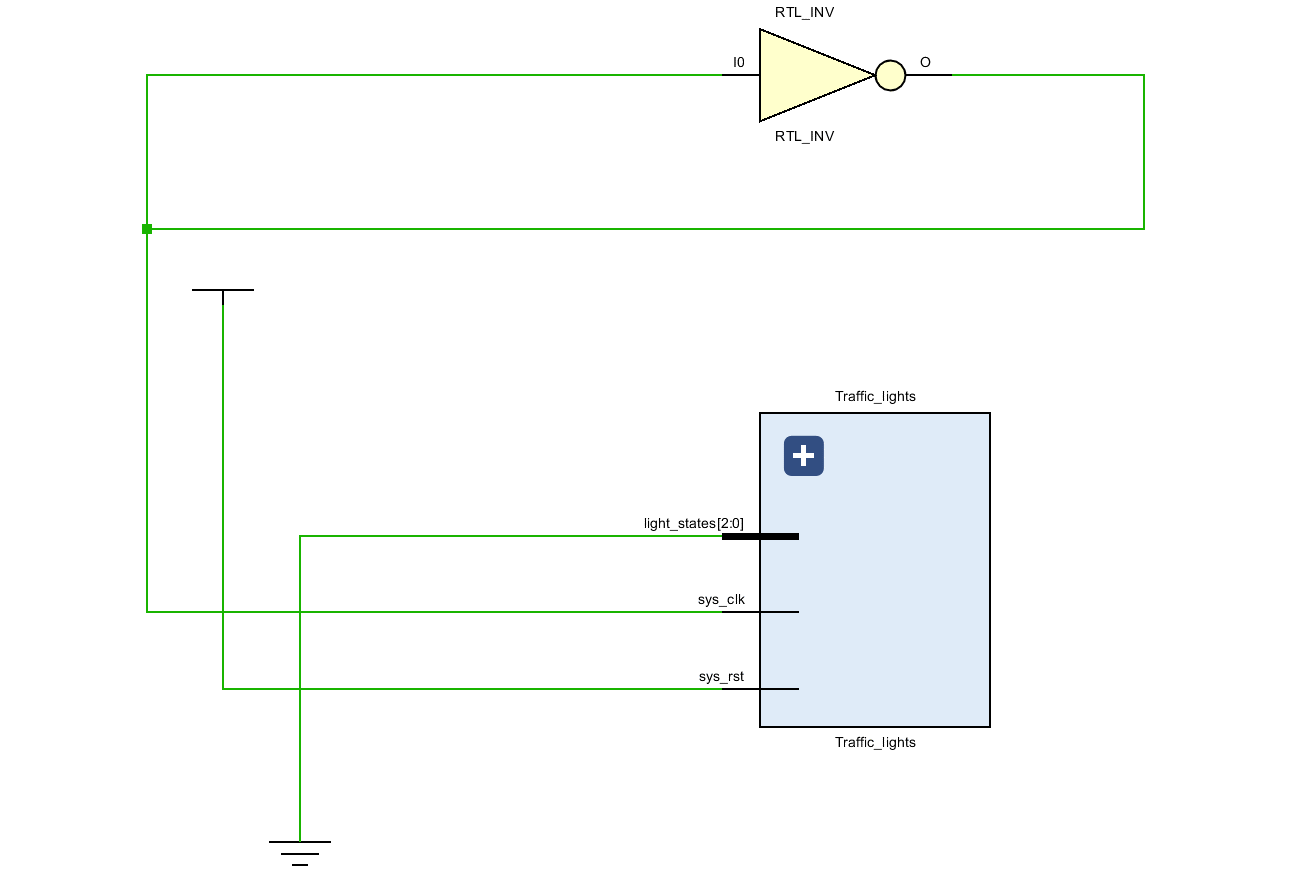


图3 顶层模块RTL图

可以看到顶层模块的RTL图显示，Traffic\_lights一共有三个端口，即sys\_clk，light\_states[2:0],sys\_rst。时钟信号是由外界激励不断反向产生的结果。

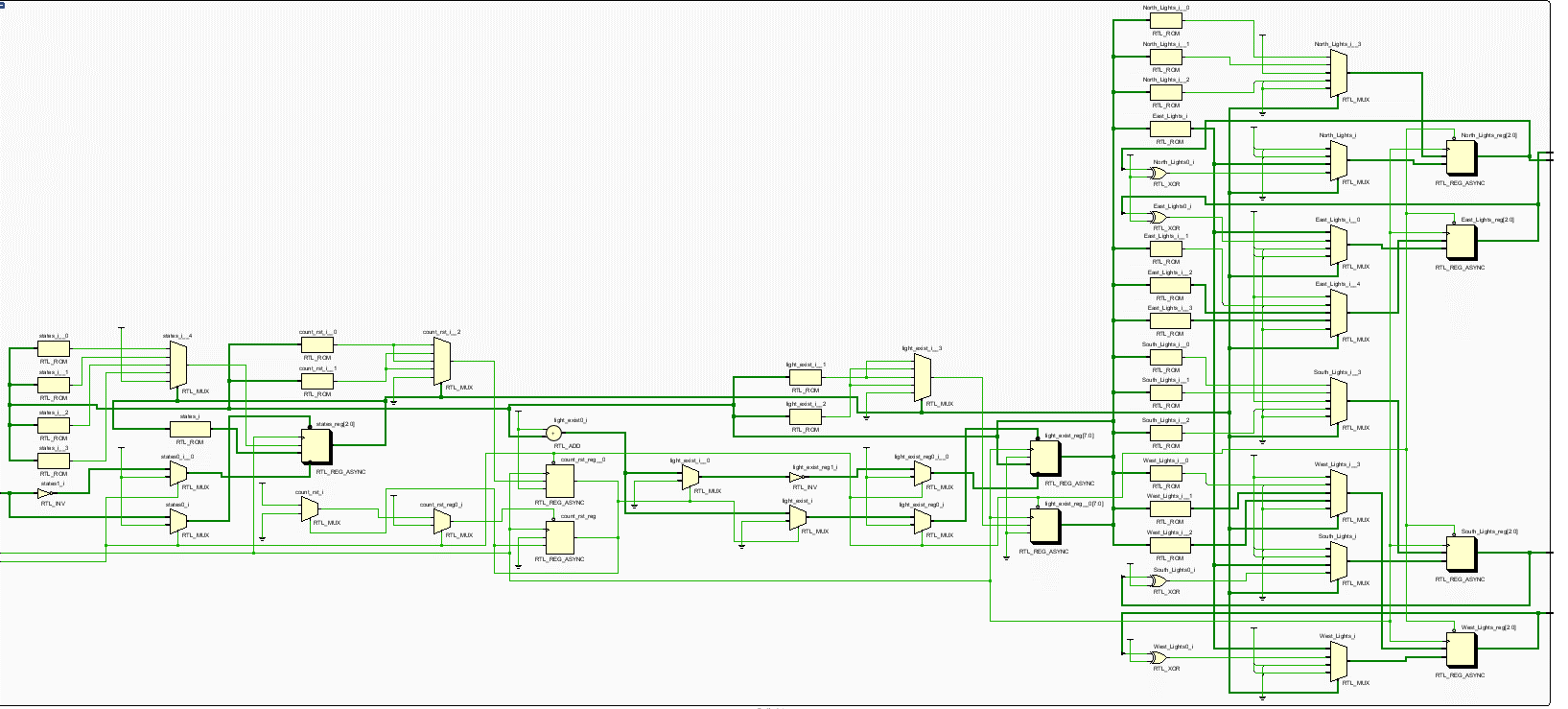


图4 Running\_Light模块RTL图

可以看到Traffic\_lights模块的RTL图，显示模块使用了较多的寄存器和多路选择器和触发器，可能存在较高的延迟,但相较于题目一器件使用的数量更少。

**设计结果分析**

**仿真整体结果分析：**

**Vivado仿真：**

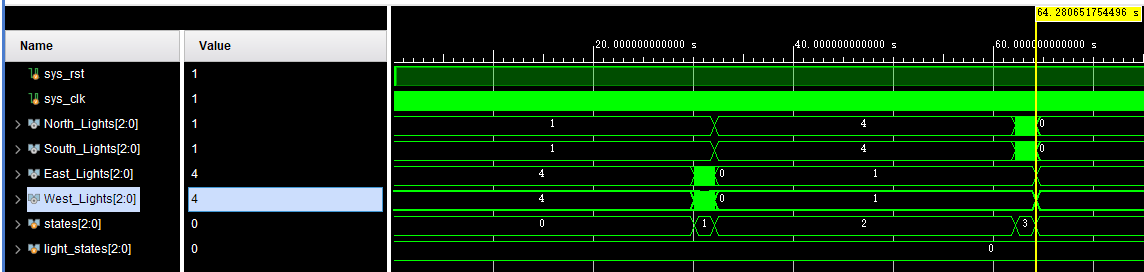
****

图5 Vivado仿真 四个工作模式波形图

可以看到Vivado仿真出的四个工作模式的波形图，不难看到完成四个工作模式所需时间约为64 s，符合题目要求的(30s+2s+30s+2s=64s)红绿灯灯工作时间。

**Modelsim仿真：**

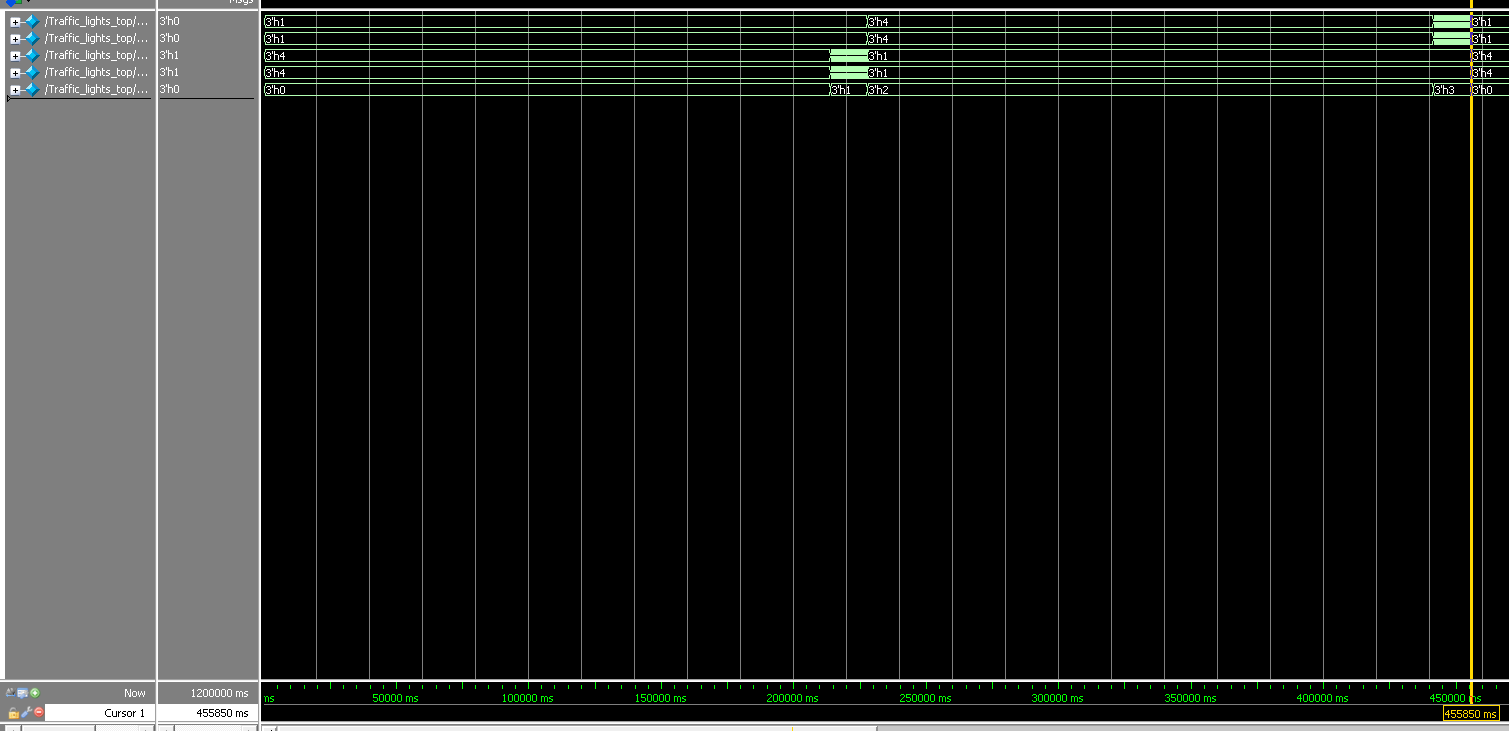
****

图6 ModelSim仿真 四个工作状态波形图

可以看到ModelSim仿真出的四个工作模式的波形图，不难看到完成四个工作模式所需时间约为45.5 s。相较于Vivado仿真的64 s，推测也应该是Vivado仿真使用的是虚拟FPGA资源，ModelSim直接使用计算机资源。

**工作模式分析**

各工作模式形图从上到下分别为复位信号sys\_rst，时钟信号sys\_clk，面北面灯[2:0]North\_lights,面南面灯[2:0]South\_lights,面东面灯[2:0]East\_lights,面西面灯[2:0]West\_lights和状态信号[2:0]states。

**工作模式1工作结果分析：**

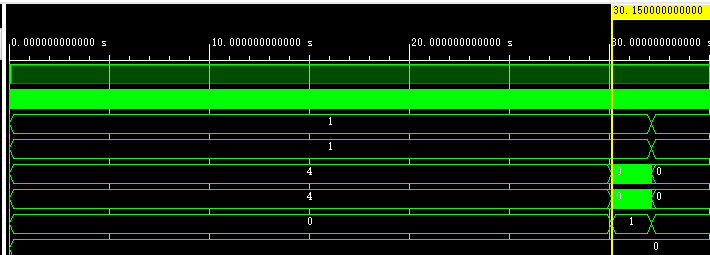
****

图7工作状态1波形图

在该工作状态，下面北面灯和面南面灯始终为3’b001（即红灯），面东面灯和面西面灯始终为3’b100（即绿灯）,且从竖线可以看到工作状态1结束的时间为30.15s符合题目要求的持续时间为30s。

**工作模式2工作结果分析：**

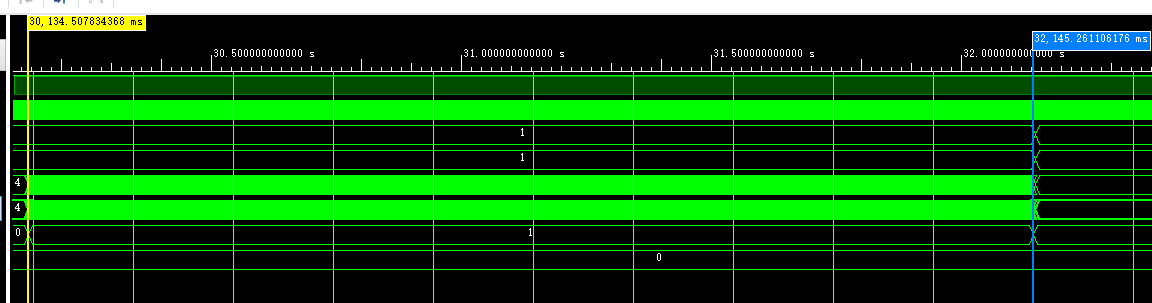


图8 工作状态2波形图

在该工作状态下，面北面灯和面南面灯始终保持为3’b001（即红灯），面东面灯和面西面灯突变为3’b010（即黄灯）且开始不断闪烁,且从竖线可以看到工作状态2结束的时间约为2.011s符合题目要求的持续时间为2s。

**工作模式3工作结果分析：**

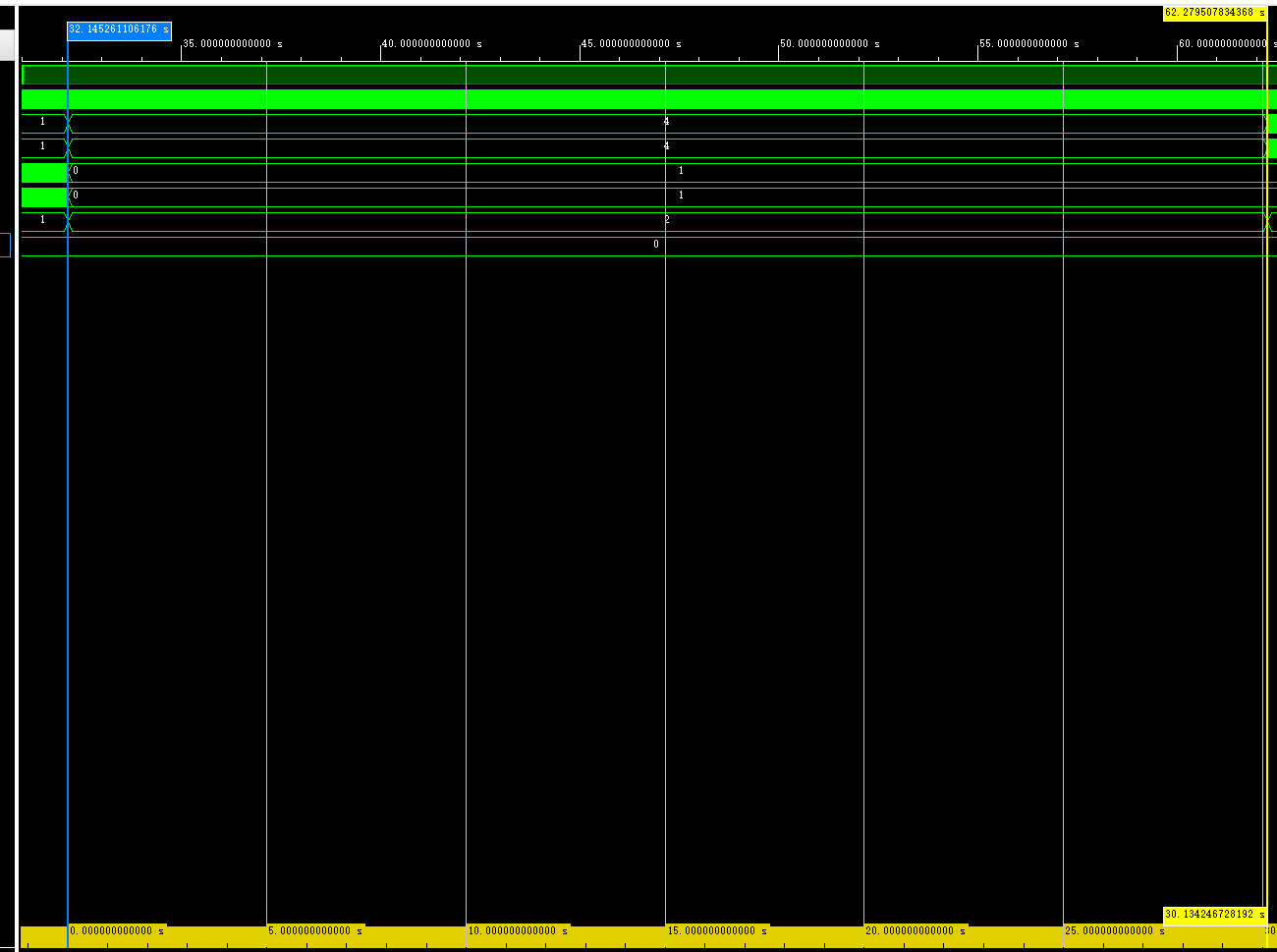
****

图9 工作状态3波形图

在该工作状态下，面北面灯和面南面灯由3’b001（即红灯）变为3’b100（即绿灯），面东面灯和面西面灯由闪烁的黄灯变为3’b001（即红灯）,且从竖线可以看到工作状态1结束的时间为30.134s符合题目要求的持续时间为30s。

**工作模式4工作结果分析：**

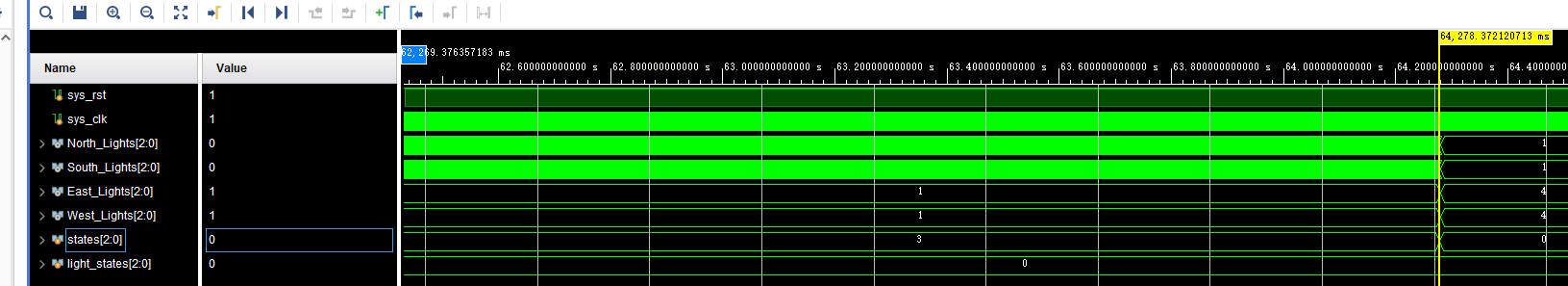
****

图10 工作状态4波形图

在该工作状态下，面北面灯和面南面灯突变为3’b010（即黄灯）且开始不断闪烁，面东面灯和面西面灯始终保持为3’b001（即红灯）,且从竖线可以看到工作状态2结束的时间约为2.008s符合题目要求的持续时间为2s。

**设计心得体会：**

本次课程设计，使我对Verilog这门语言有了更加深入的了解。在本次设计中，我遇到的最大问题有两个，一个是题目一的重复操作和两个题目的分频操作。问题一的重复操作主要是要考虑到非阻塞赋值的特点，即等号右边的值都先计算，然后再到下一个时钟的上升沿处，再统一赋值给等号左边。我的问题是在状态转换时，会出现一些参数还没复位但状态已经转过去，然后直接略过转过去的状态跳到下下个状态，解决方法主要是再另写一个去处理复位而不去对LED 操作的状态。问题二，主要还是在题目一怎么分频上卡住了，一开始的想法是写自建任务，但是无论怎么样都无法分频第二次，发现是要写automatic(动态)去修饰多次调用的任务，来给数据划分多个空间，再然后发现分频的计数参数只会变化一次，参考C语言写了个嵌套，但一开始仿真就会卡在13ns，把电路综合出来发现是出现了多个锁存器。于是我放弃了task的写法，把任务删了，然后写第二个always语句用来分频。在仿真阶段，我发现写always语句要比task更好处理本题的变化，于是题目一和题目二都改用为always。究其原因，还是把HDL语言当作C语言来写，写Verilog程序应该是带着会综合成什么硬件的视角来看待这个问题。

**参考文献**

[1] 夏宇闻.Verilog数字系统设计教程.北京：北京航空航天大学出版社.2017

[2] 刘福奇.Verilog HDL设计与实战.北京：北京航空航天大学出版社.2012