djm09thu@gmail.com

EDA大作业

**数码管显示及 矩阵键盘扫描电路**

电92 丁健民 2009010942

# EDA大作业

# 数码管显示及矩阵键盘扫描电路

# 设计文档

**电92 丁健民 2009010942**

## 引言

### 设计目的

1. 学习使用硬件描述语言设计电路
2. 提高调试简单数字系统的能力

### 参考文献

1. 《数字电子技术基础（第五版）》清华大学电子学教研组编；阎石主编；高等教育出版社，2006
2. 《电子电路实验》高文焕等编著；清华大学出版社，2008
3. 《Verilog HDL综合实用教程》J.Bhasker著；清华大学出版社，2004年

## 任务概述

### 设计目标

**任务1:**

**设计一个数码管的动态扫描显示电路，在4位数码管上同时显示4个数字。**

具体内容及步骤：

设计一个分频器，建议将系统25MHz分频至250Hz。

利用状态机设计一个电路，使作业一中的两个运算数、运算结果和零同时显示在4位数码管上。

1. 设计一个分频器，建议将系统25MHz分频至250Hz。
2. 利用状态机设计一个电路，使作业一中的两个运算数、运算结果和零同时显示在4位数码管上。

**任务2：**

**设计一个4×4矩阵键盘信号检测电路识别按键数字。**

键盘中各键所代表的数字如下图所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 |  |  |  |
| 0 |  |  |  |

步骤：

1. 将输入按键的二进制编码显示在发光二极管上。
2. 设计译码电路，将输入按键数字显示在数码管上。

### 运行环境

Quartus II 9.0

## 总体设计思路

### 任务1

### 任务2

## 详细模块设计

### 分频器（divider）模块

|  |
| --- |
| module divider(clk,rst,count,clk\_did);  input clk,rst;  output clk\_did;  output[16:0] count;  reg [16:0]count;  reg clk\_did;  parameter N=100000;  always@(posedge clk or negedge rst)  begin  if (!rst)  begin  count<=0;clk\_did<=0;  end  else  if(count<N/2-1)  count<=count+1;  else begin  count<=0;clk\_did<=~clk\_did;  end  end  endmodule |

**【模块说明】**

**工程文件路径：**

2009010942\EX1\divider\divider.qdf

**功能：**

实现利用系统现有的25MHz触发信号源产生250Hz触发信号。

**设计思路：**

通过25MHz产生250Hz，为倍分频，是偶数倍分频。

偶数（N）倍分频，可以通过由待分频的时钟（clk）触发计数器计数（count），当计数器从0计数到N/2-1时，输出时钟（clk\_did）进行翻转，并给计数器一个复位信号，使得下一个时钟从零开始计数。以此循环下去，可以实现任意的偶数分频。

**输入信号：**

clk：系统固有25MHz触发信号源。

rst：选通信号，只有当rst为高电平时分频器正常进行分频。

**输出端：**

count[16..0]：计数信号。

clk\_did：分频后的250Hz触发信号

**封装为模块：**



### 状态机（fsm1）模块

|  |
| --- |
| module fsm1(input clk\_did,output reg K0,K1);  reg [1:0] state;  parameter S0=0,S1=1,S2=2,S3=3;    always @(state)  begin  case(state)  S0:  K0=0;  S1:  K0=1;  S2:  K0=0;  S3:  K0=1;  default:  K0=0;  endcase  end    always @(state)  begin  case(state)  S0:  K1=0;  S1:  K1=0;  S2:  K1=1;  S3:  K1=1;  default:  K1=0;  endcase  end  always @(posedge clk\_did )  begin  case(state)  S0:  state<=S1;  S1:  state<=S2;  S2:  state<=S3;  S3:  state<=S0;  endcase  end  endmodule |

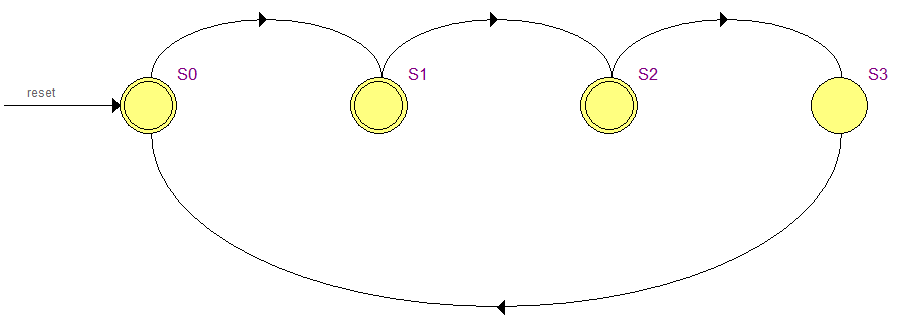
**【模块说明】**

**工程文件路径：**

2009010942\EX1\fsm1\fsm1.qdf

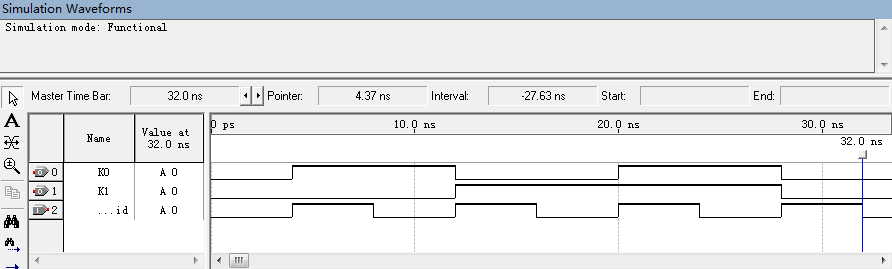
**功能：**

状态机，与《EDA讲座2》中“循环点亮LED电路”类似，实现为3位二进制运算器4位数码管扫描显示电路提供动态B0（K0）、B1（K1）控制信号。



|  |  |  |  |
| --- | --- | --- | --- |
|  | K0 | K1 |  |
| S0 | 0 | 0 | S1 |
| S1 | 1 | 0 | S2 |
| S2 | 0 | 1 | S3 |
| S3 | 1 | 1 | S0 |

**波形仿真：**



clk上升沿到达时，K0状态在0、1、0、1循环，K1状态在0、0、1、1循环，从而实现K0K1状态在00、10、01、11循环。

**设计思路：**

类似《EDA讲座2》中“循环点亮LED电路”语句实现，依次为说明部分（定义各个状态）、组合电路部分（相当于输出方程）、时序电路电路部分（相当于驱动方程和状态方程）。

**输入信号：**

clk\_did：触发源，由分频器产生的250Hz触发信号源提供。

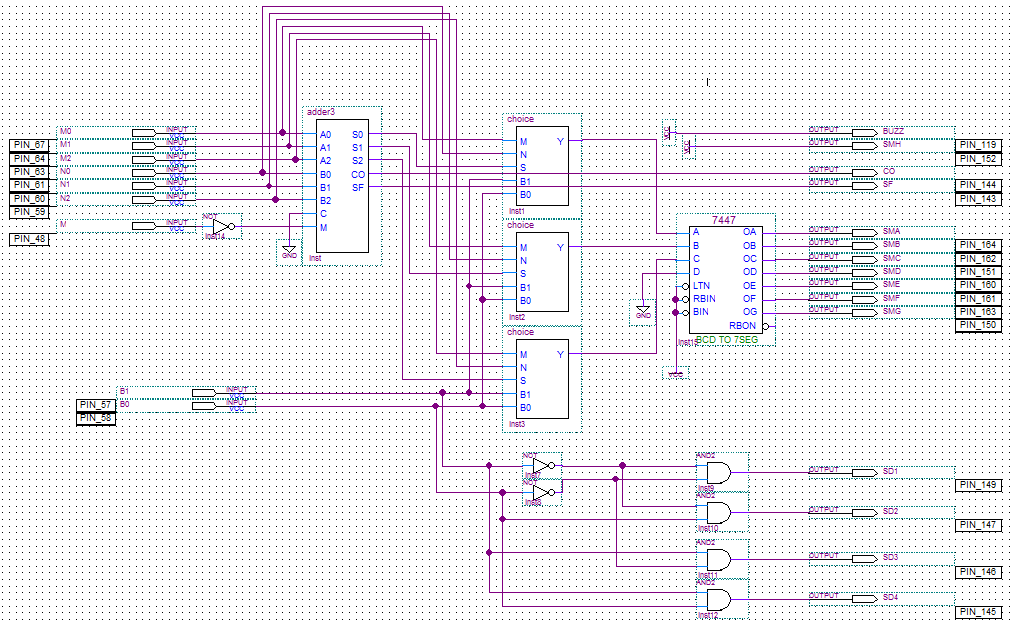
**输出端：**

K0、K1：“3位二进制运算器4位数码管扫描显示电路”所需的控制信号，对应B0、B1。

**封装为模块：**



### 3位二进制运算器4位数码管扫描显示模块



**【模块说明】**

**顶层原理图文件路径：**

2009010942\EX1\EX1\EX2.bdf

**功能：**

将3位二进制运算器的两个运算数和运算结果根据B1B0的状态在4位数码管上轮流显示，标志位（加法时进位标志和减法时借位标志）在发光二极管上显示。

**设计思路：**

EDA作业1中已经阐述，此处不再赘述。

**输入端：**

M：控制进行加法还是减法运算（低电平进行加法运算，高电平进行减法运算）

M0、M1、M2：加数（或被减数）最低位、第二位、第三位

N0、N1、N2：被加数（或减数）最低位、第二位、第三位

B0、B1：数据选择端，其状态决定显示运算数还是运算结果还是0。

**输出端：**

CO：（进行加法运算时）向高位的进位

SF：（进行减法运算时）借位标志

SMA~SMG：BCD—七段显示译码器7447的输出，控制数码管显示结果。

SMH：数码管小数点控制端，接高电平表示默认不显示小数点。

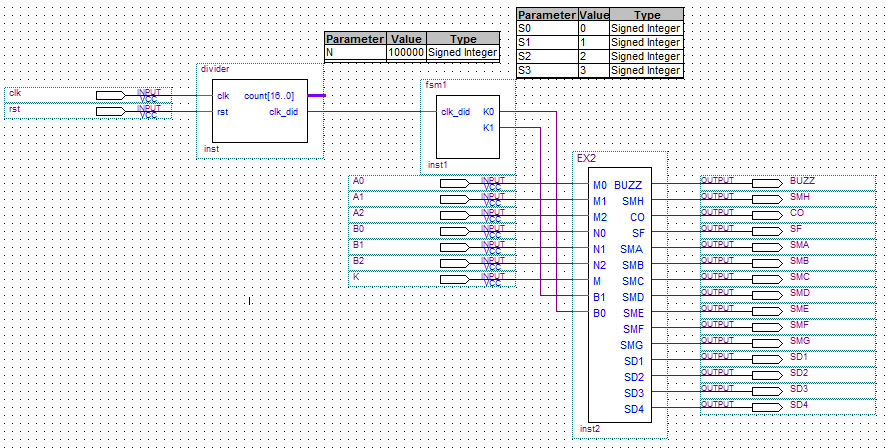
BUZZ：FPGA开发板蜂鸣器控制端，接高电平表示默认蜂鸣器不发声。

SD1~SD4：数码管亮灭控制端，接高电平表示对应数码管亮。

**封装为模块：**



### 任务1：数码管动态扫描显示电路



**【模块说明】**

**工程文件路径：**

2009010942\EX1\EX1\EX1.qdf

**功能：**

将EDA作业一“3位二进制运算器4位数码管扫描显示电路”中的两个运算数、运算结果和零同时显示在4位数码管上。

**设计思路：**

利用分频器和状态机产生动态扫描的“3位二进制运算器4位数码管扫描显示电路”所需的控制信号B0、B1，使其状态在00、10、01、11间依次跳变。此时“3位二进制运算器4位数码管扫描显示电路”依次在1~4号数码管上显示加数（或被减数）、被加数（或减数）、运算结果、0，由于B0、B1状态频率为250Hz，即周期0.004s，低于人眼睛的视觉残留（0.1s）时间，因此人眼看到的就是加数（或被减数）、被加数（或减数）、运算结果、0同时显示在4位数码管上。

**管脚锁定：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 输入端  /输出端 | 原理图上  信号名 | EP2C5Q208C8N  引脚号 | 输入端  /输出端 | 原理图上  信号名 | EP2C5Q208C8N  引脚号 |
| A0 | SB0 | 67 | SMA | SMA | 164 |
| A1 | SB1 | 64 | SMB | SMB | 162 |
| A2 | SB2 | 63 | SMC | SMC | 151 |
| B0 | SB3 | 61 | SMD | SMD | 160 |
| B1 | SB4 | 60 | SME | SME | 161 |
| B2 | SB5 | 59 | SMF | SMF | 163 |
| K | RST | 48 | SMG | SMG | 150 |
| rst | SB6 | 58 | SMH | SMH | 152 |
| clk | CLK | 132 | SD1 | SD1 | 149 |
| CO | LD1 | 144 | SD2 | SD2 | 147 |
| SF | LD2 | 143 | SD3 | SD3 | 146 |
| BUZZ | BUZZ | 119 | SD4 | SD4 | 145 |

**输入端：**

clk：系统的25MHz触发信号源

rst：选通端，当为高电平时电路正常工作。

K：控制进行加法还是减法运算（低电平进行加法运算，高电平进行减法运算）

A0、A1、A2：加数（或被减数）最低位、第二位、第三位

B0、B1、B2：被加数（或减数）最低位、第二位、第三位

**输出端：**

CO：（进行加法运算时）向高位的进位

SF：（进行减法运算时）借位标志

SMA~SMG：BCD—七段显示译码器7447的输出，控制数码管显示结果。

SMH：数码管小数点控制端，接高电平表示默认不显示小数点。

BUZZ：FPGA开发板蜂鸣器控制端，接高电平表示默认蜂鸣器不发声。

SD1~SD4：数码管亮灭控制端，接高电平表示对应数码管亮。

### 状态机（fsm2）模块

|  |
| --- |
| module fsm2(clk,in,rst,out);  input clk,in,rst;  reg [1:0]state;  output reg[3:0]out;  parameter S0=0,S1=1,S2=2,S3=3;    always@(state)  begin  case(state)  S0: out=4'b0001;  S1: out=4'b0010;  S2: out=4'b0100;  S3: out=4'b1000;  default: out=4'b0001;  endcase  end    always@(posedge clk or negedge rst)  begin  if(!rst)  state<=S0;  else  case(state)  S0:  if(in)  state<=S1;  else  state<=S0;  S1:  if(in)  state<=S2;  else  state<=S1;  S2:  if(in)  state<=S3;  else  state<=S2;  S3:  if(in)  state<=S0;  else  state<=S3;  endcase  end  endmodule |

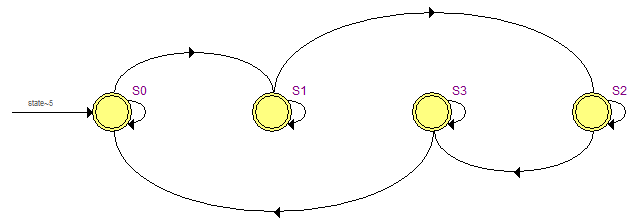
**【模块说明】**

**工程文件路径：**

2009010942\EX2\fsm2\fsm2.qdf

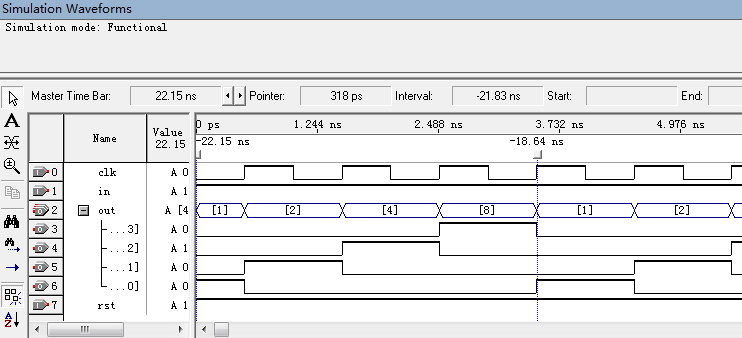
**功能：**

状态机，与《EDA讲座2》中“循环点亮LED电路”类似，实现为“行列线电平状态确定键值” 电路提供动态的列控制信号。



|  |  |  |  |
| --- | --- | --- | --- |
|  | out[3..0] | in=0时 | In=1时 |
| S0 | 0001 | S0 | S1 |
| S1 | 0010 | S1 | S2 |
| S2 | 0100 | S2 | S3 |
| S3 | 1000 | S3 | S0 |

**波形仿真：**



当rst和in都为高电平，clk上升沿到达时，out[3..0]状态在0001、0010、0100、1000间跳变。

**设计思路：**

类似《EDA讲座2》中“循环点亮LED电路”语句实现，依次为说明部分（定义各个状态）、组合电路部分（相当于输出方程）、时序电路电路部分（相当于驱动方程和状态方程）。

**输入信号：**

clk：触发源，由分频器产生的250Hz触发信号源提供。

rst：选通端1，为高电平时状态机正常工作。

in：选通端2，当为低电平时保持现有状态，否则状态机正常工作。

**输出端：**

out[3..0]：“行列线电平状态确定键值” 电路所需列信号。

**封装为模块：**



### 行列线电平状态确定键值（Martrix）模块

|  |
| --- |
| module Martrix(clk,rst,line,row,out);  input clk,rst;  input [3:0]line;  input [3:0]row;  reg [7:0]data;  output reg[3:0]out;  always@(posedge clk or negedge rst)  begin  if (!rst)  data<=0;  else  if (row!=0)  data<={line,row};  else  data<=data;  case (data)  8'b1000\_1000: out<=1;  8'b0100\_1000: out<=2;  8'b0010\_1000: out<=3;  8'b0001\_1000: out<=4;  8'b1000\_0100: out<=5;  8'b0100\_0100: out<=6;  8'b0010\_0100: out<=7;  8'b0001\_0100: out<=8;  8'b1000\_0010: out<=9;  8'b1000\_0001: out<=0;  default:out<=0;  endcase  end  endmodule |

**【模块说明】**

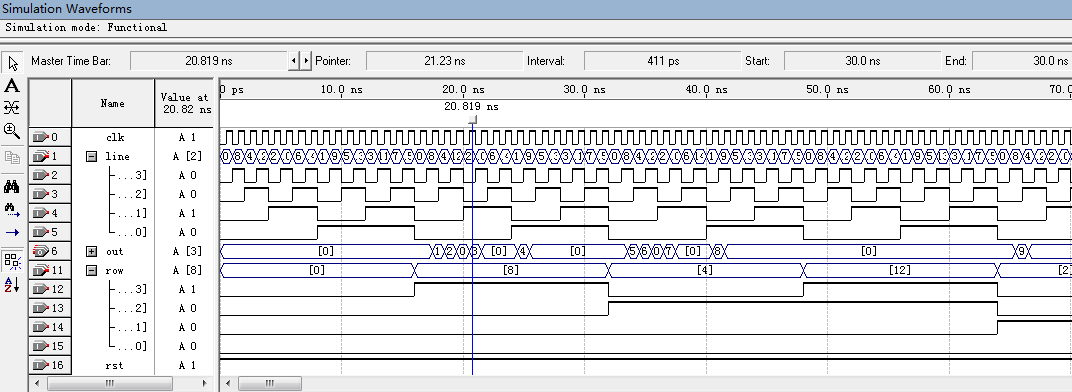
**工程文件路径：**

2009010942\EX2\Martrix\Martrix.qdf

**功能：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | line（3） | line（2） | line（1） | line（0） |
| row（3） | 1 | 2 | 3 | 4 |
| row（2） | 5 | 6 | 7 | 8 |
| row（1） | 9 |  |  |  |
| row（0） | 0 |  |  |  |

根据输入行和列信号的电平状态确定键值。当某行和某列信号同时为高电平时，键值即为相应行列对应的数字。例：row（3）=1，line（1）=1，则键值即为3。

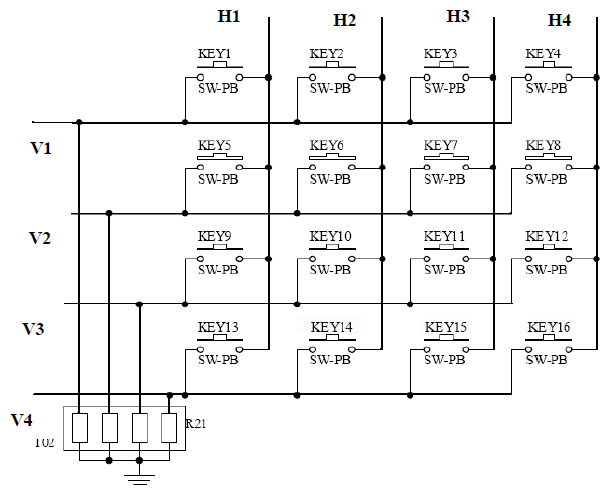


上图的波形仿真中，line信号中line（1）=1，而row信号中row（3）=1，因此键值（out[3..0]]）为3。

**设计思路：**

采用了常用的4\*4矩阵键盘的设计思路。

矩阵键盘又称行列键盘，下图为4\*4矩阵键盘的原理图。四条列线（H1,H2,H3,H4）为输入，四条行线（V1,V2,V3,V4）为输出，行线和列线均与FPGA芯片引脚相连。在行线和列线的每个交叉点上设置一个按键，当某一个按键被按下时，相应的行线和列线就会接通。



将输入列线H1～H4全部置为高电平，然后检测行线的状态。若所有行线均为低电平，则没有按键按下；若有一根行线为高电平，则表示键盘中有按键按下，并且这个按键位于这根行线的4个按键中。因此，要判断出按下按键的具体位置可以运用逐行扫描法。具体操作是：依次在列线H1～H4中的某一条上加高电平（其余为低电平），再逐行检测各行线的电平状态。若某条行线为高电平，则表示这条行线与置为高电平的列线相交叉处的按键被按下。而没有按键按下的行线则会输出低电平。根据此时的行列线的电平状态就可以确定唯一被按下的键值。例如，某时刻输入H1、H2、H3、H4为1000，若检测到输出V1、V2、V3、V4为0000，表示没有按键按下；若输出为0100，则表示行线V2为高电平，即KEY5被按下。

在我的电路实现中，列线“H1～H4全部置为高电平”通过状态机提供动态扫描的列信号实现，在clk作用下，line（3）（对应H1）~line（0）（对应H4）依次被置高电平（其他则为低电平），由于clk为250Hz触发信号源，因此line信号的跳变频率远远高于人手按键的频率，相当于将列线“H1～H4全部置为高电平”。

**输入信号：**

clk：触发源，由分频器产生的250Hz触发信号源提供。

rst：选通端，为高电平时正常工作。

line[3..0]：列信号，由状态机（fsm2）提供。

row[3..0]：行信号，由操作者的按键提供。

**输出端：**

out[3..0]：根据行列信号电平状态确定的键值。

**封装为模块：**



### 译码器（i7447）模块

|  |
| --- |
| module i7447(in,rst,out);  input rst;  input [3:0]in;  output reg [6:0]out;  always@(in)  begin  if(!rst)  out<=7'b1111111;  else  case(in)  4'b0000: out<=7'b1000000;  4'b0001: out<=7'b1111001;  4'b0010: out<=7'b0100100;  4'b0011: out<=7'b0110000;  4'b0100: out<=7'b0011001;  4'b0101: out<=7'b0010010;  4'b0110: out<=7'b0000010;  4'b0111: out<=7'b1111000;  4'b1000: out<=7'b0000000;  4'b1001: out<=7'b0010000;  default:out<=7'b1111111;  endcase  end  endmodule |

**【模块说明】**

**工程文件路径：**

2009010942\EX2\i7447\i7447.qdf

**功能：**

自行设计的与7447基本功能相同的译码器。

**设计思路：**

列写真值表，对于in信号（0000~1001即0~9）给出相应的out信号，out信号参照BCD—七段显示译码器真值表。

**输入信号：**

in[3..0]：“行列线电平状态确定键值” 模块的输出（0000~1001即1~9）。

rst：选通信号，只有当rst为高电平时正常工作。

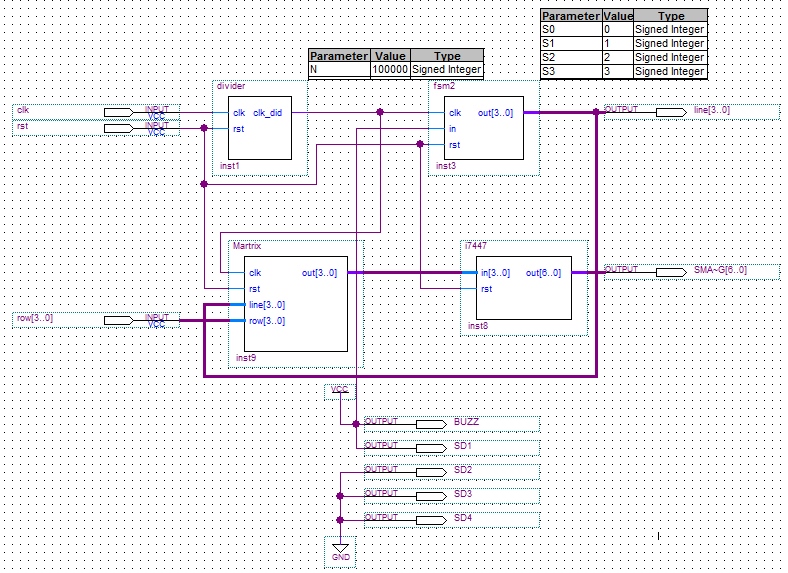
**输出端：**

out[6..0]：译码输出。

**封装为模块：**



### 任务2：矩阵键盘



**【模块说明】**

**工程文件路径：**

2009010942\EX2\keyboard\keyboard.qdf

**功能：**

4×4矩阵键盘信号检测电路，识别按键数字。

**设计思路：**

利用分频器产生250Hz触发信号，由状态机产生动态扫描的列信号，行信号由操作者按键提供，由“行列线电平状态确定键值模块”根据行列信号确定键值提供给译码器模块在数码管1上显示相应的键值。

**管脚锁定：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 输入端  /输出端 | 原理图上  信号名 | EP2C5Q208C8N  引脚号 | 输入端  /输出端 | 原理图上  信号名 | EP2C5Q208C8N  引脚号 |
| clk | CLK | 132 | SMA | SMA | 164 |
| rst | RST | 48 | SMB | SMB | 162 |
| line(0) | H4 | 118 | SMC | SMC | 151 |
| line(1) | H3 | 117 | SMD | SMD | 160 |
| line(2) | H2 | 116 | SME | SME | 161 |
| line(3) | H1 | 115 | SMF | SMF | 163 |
| row(0) | V4 | 110 | SMG | SMG | 150 |
| row(1) | V3 | 112 | SD1 | SD1 | 149 |
| row(2) | V2 | 113 | SD2 | SD2 | 147 |
| row(3) | V1 | 114 | SD3 | SD3 | 146 |
| BUZZ | BUZZ | 119 | SD4 | SD4 | 145 |

**输入端：**

clk：系统的25MHz触发信号源

rst：选通端，当为高电平时电路正常工作。

row[3..0]：行信号，由操作者按键提供

**输出端：**

line[3..0]：列信号，提供给FPGA开发板矩阵键盘部分的H1~H4。

SMA~SMG：BCD—七段显示译码器7447的输出，控制数码管显示结果。

BUZZ：FPGA开发板蜂鸣器控制端，接高电平表示默认蜂鸣器不发声。

SD1~SD4：数码管亮灭控制端，接高电平表示对应数码管亮。

## 收获与体会

1. “磨刀不误砍柴工”，在完成此次EDA大作业之前，我对老师提供的《VHDL和QuartusII使用介绍》以及从图书馆借阅的EDA相关书籍（如《Verilog HDL综合实用教程》）进行了仔细研读，学习了设计电路的另一种输入方式——硬件描述语言Verilog，并对任务1和2的设计进行了认真的构想。从而，在开始着手使用Quartus II9.0进行HDL设计及仿真中进展相对较顺利，在最后的下载及功能演示环节也是几经尝试后就成功。收获成功的喜悦的同时，也让我意识到，“预则立，不预则废”，特别是本次EDA作业布置给我们留下相对比较充裕的完成时间，我们更应该做好相关的软件技术和系统设计的相关准备，这样就避免了盲目上手和一些不必要的磕磕绊绊。
2. 进一步掌握了EDA软件Quartus II进行设计的主要环节和HDL输入方式。Quartus II提供了强大的FPGA开发设计集成环境，在进行设计时主要有设计输入、分析与综合、仿真、布局布线、时序分析和引脚锁定及下载等工作。功能强大的同时，其实也是一款很有意思的软件，在本次EDA作业中我们主要面对使用硬件描述语言（主要是Verilog）实现数字系统设计，相对原理图输入方式关心电路的结构和连接关系，其更关心电路功能的实现，因此具体的语法实现只需要阅读相关书籍和在C语言学习的基础上就可以解决，重头戏仍在电路功能的设计实现。
3. 一次EDA作业，酸甜苦辣只有自己清楚，非常感谢王红老师和助教们，这次作业对我个人的能力和信心的提升作用都是异乎寻常的。试想当把设计的电路下载到FPGA上后成功实现了预想的功能，那是怎样的一份惊叹和喜悦！