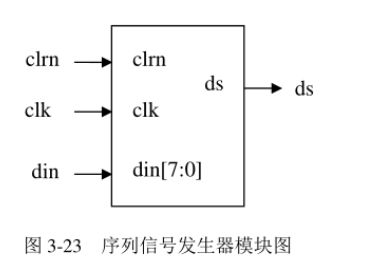
**湘潭大学实验报告**

课程名称：数字电路与逻辑电路 实验名称：序列信号发生器 实验日期：2022.5.28 班级：21级计算机科学与技术3班 学号：202105566416 姓名：熊心雨

1. **实验目的**

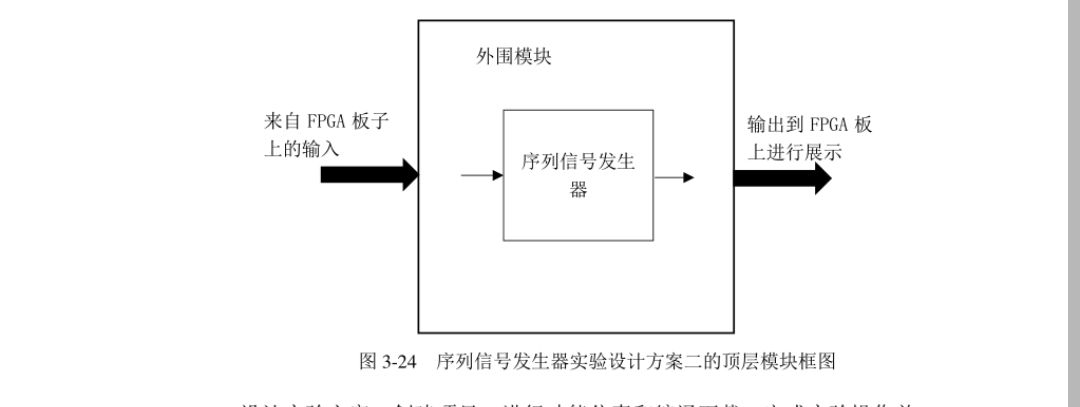
1、用状态机实现序列信号发生器的设计，了解一般状态机的设计与应用。  
2、熟悉Vivado，具备使用Verilog HDL编程、仿真并进行硬件测试的能力。

1. **实验内容及要求**
2. 序列信号发生器设计  
   通过 Verilog HDL 语言编程，实现一个8位序列信号发生器，要求有1个时钟脉冲输入、1个异步清零端、1个8位预置数据输入端和1个串行数据输出端，序列检测器输入、输出端口设计具体要求如下，序列信号发生器模块图如图3-23所示。

  
clrm：异步清零信号，低电平有效：  
clk：时钟脉冲输入，上升沿有效。  
din:8位预置数据输入端   
ds：串行数据输出端  
2、完成序列信号发生器的设计编辑和 din din[7:0]仿真测试等步骤，给出仿真波形，了解串行数据输出端的时序。 图3-23序列信号发生器模块图  
3、在硬件实现中，有不同的设计方案，  
以下例举2个。你也可以自行设计其他方案，不必拘泥于设计示例。  
(1)方案一：用拨码开关控制复位信号；按钮开关控制工作时钟；将8位预置数据由用拨码开关控制复位输入，串行数据输出端接LED灯。将具体的引脚分配填入表3-13。

  
(2)方案二：设计一个外围模块去调用该序列信号发生器模块，如图3-24

所示。外围模块中需调用封装好的LCD触摸屏模块，观察序列信号发生器的输入输出的值等。利用触摸功能输入预置的数据，实时观察输出的串行数据值的变化。



4、设计实验方案，创建项目，进行功能仿真和编译下载，完成实验操作并做好数据记录。  
如果采用实验设计方案一，建议下载后：1用复位按钮控制复位；2用8个拨码开关输入预置数据，例如“11010110”；③按单步调试按钮 8 次，产生 8个时钟上升沿，这时预置输入数据的低位到高位依次显示于LED灯。④设计数据记录表并记录实验结果。  
5、根据实验内容，写出实验报告，包括程序设计、软件编译、仿真分析、硬件测试和详细实验过程；给出程序分析报告、仿真波形图及其分析报告。

**三、实验设备与软件环境**

实验电脑设备规格：

设备名称：xd60510

处理器：Intel(R) Core(TM) i5-9500 CPU @ 3.00GHz 3.00GHz

系统类型：64位操作系统 基于x64的处理器

实验电脑Windows规格：

版本：windows 10专业版

操作系统版本：18362.329

实验箱设备：

设备名称：Cpu FPGA 教学实验系统V2.0

设备型号：LS-CPU-EXB-001

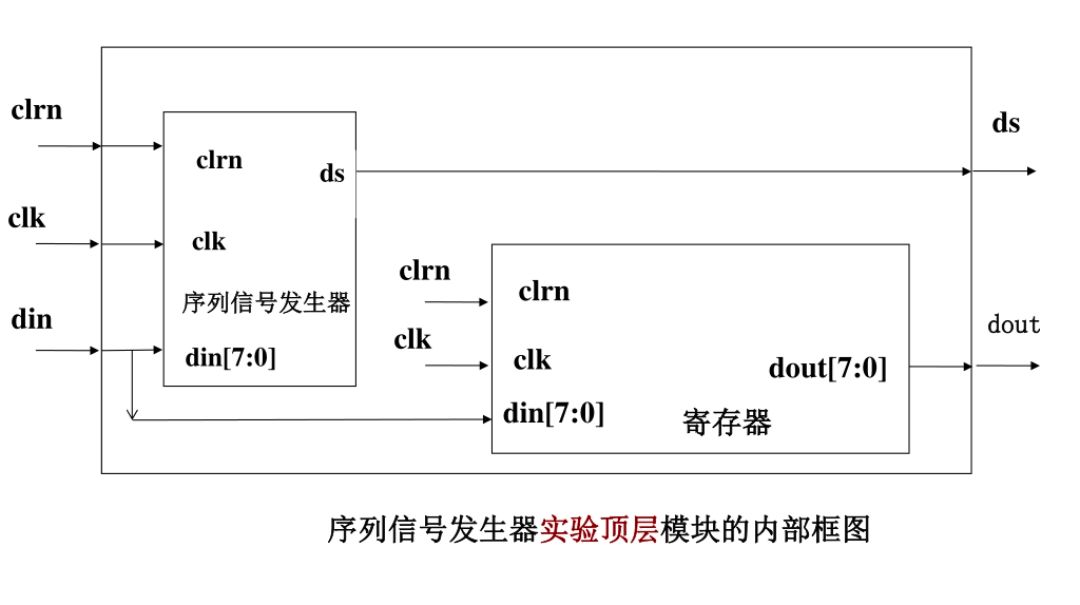
软件环境：

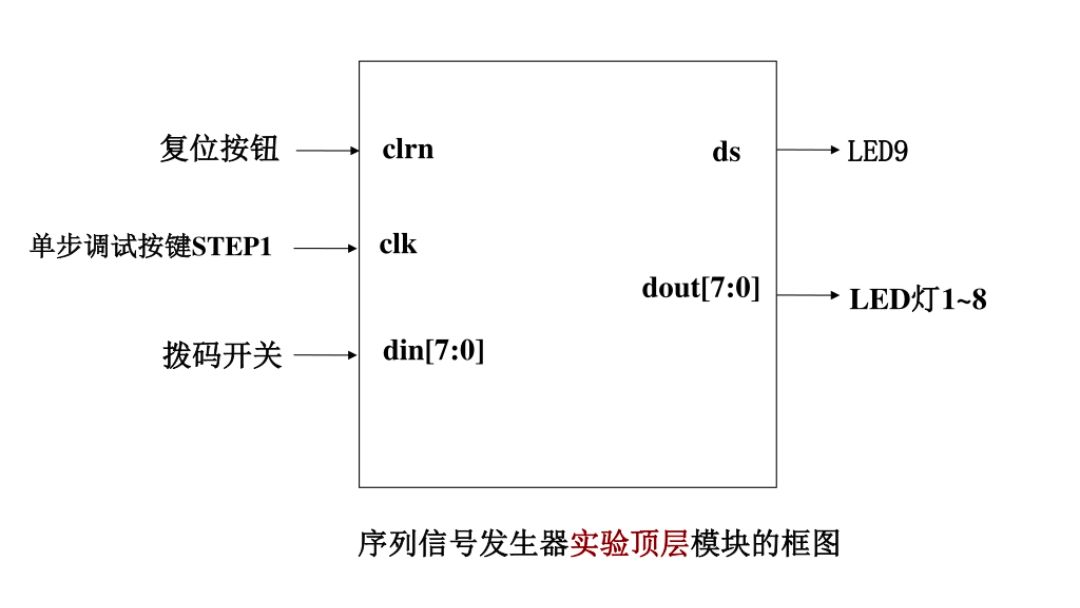
Vivado v2019.2 (64-bit)

* 1. **实验方案或原理**

序列信号发生器可用于产生一组特定的由二进制码组成的脉冲序列信号。例如在数字信号的传输和数字系统的测试中，需要用到一组特定的串行数字信号‘11010110，就可以利用序列信号发生器产生该序列信号。  
序列信号发生器的设计方法有多种。比如可用计数器和数据选择器组成序列信号发生器，如需产生一个8位的序列信号11010110(时间顺序为自左向右)，则可用一个8进制计数器和一个8选1数据选择器组成。

**五．代码设计及实验步骤**  
 1.代码设计





八选一数据选择器

module counter(clk,clrn,count);

input clk,clrn;

output [2:0] count;

reg [2:0] count;

always@(posedge clk or negedge clrn)

begin

if(!clrn)

begin

count<=3'd0;

end

else

begin

count<= count +1;

end

end

endmodule

module mux81(count,din,ds);

input [7:0] din;

input [2:0] count;

output ds;

reg ds;

always @(\*)

case( count )

3'd0:ds <= din[0];//0

3'd1:ds <= din[1];//1

3'd2:ds <= din[2];//2

3'd3:ds <= din[3];//3

3'd4:ds<= din[4];//4

3'd5:ds <= din[5];//5

3'd6:ds<= din [6];//6

3'd7:ds<=din[7];//7

default:ds<=1'b0;

endcase

endmodule

module register(clk,clrn,din,dout);

input clk,clrn;

input [7:0] din;

output [7:0] dout;

reg [7:0] dout;

always @ (posedge clk or negedge clrn)

if (clrn==0)

begin

dout<=8'd0;

end

else begin

dout<=din;

end

endmodule

module sequencer(clk,clrn,din,ds,dout);

input clk,clrn;

input [7:0] din;

output [7:0] dout;

output ds;

//8位寄存器

register rgs(clk,clrn,din,dout);

//8位序列发生器

sequencer\_module m(clk,clrn,din,ds);

endmodule

module sequencer\_module(clk,clrn,din,ds);

input [7:0] din;

input clk ,clrn;

output ds;

wire [2:0] count;

counter cnt(clk,clrn,count);

mux81 M(count ,din, ds );

endmodule

`include "counter.v"

`include"mux81.v"

`include "sequencer\_module.v"

`include "register.v"

module sequencer\_tb;

parameter DATA\_WIDTH=8;

//--------------Input Ports-----//

reg [DATA\_WIDTH-1:0] din;

reg clk,clrn;

//--------------output Ports---------//

wire ds;

wire [DATA\_WIDTH-1:0] dout;

initial begin

clk=1;

clrn=0;

din=8'b00001110;

#0 $display("time\tclk\tclrn\tdin\tds\tdout");

#4 clrn=1;

end

always #5 clk=~clk;

always #80 din=din+1;

sequencer m(clk,clrn,din,ds,dout);

endmodule

set\_property PACKAGE\_PIN V6 [get\_ports clk]

set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets clk\_IBUF]

set\_property PACKAGE\_PIN Y3 [get\_ports clrn]

set\_property PACKAGE\_PIN AC21 [get\_ports {din[7]}]

set\_property PACKAGE\_PIN AD24 [get\_ports {din[6]}]

set\_property PACKAGE\_PIN AC22 [get\_ports {din[5]}]

set\_property PACKAGE\_PIN AC23 [get\_ports {din[4]}]

set\_property PACKAGE\_PIN AB6 [get\_ports {din[3]}]

set\_property PACKAGE\_PIN W6 [get\_ports {din[2]}]

set\_property PACKAGE\_PIN AA7 [get\_ports {din[1]}]

set\_property PACKAGE\_PIN Y6 [get\_ports {din[0]}]

set\_property PACKAGE\_PIN J8 [get\_ports ds]

set\_property PACKAGE\_PIN H7 [get\_ports {dout[7]}]

set\_property PACKAGE\_PIN D5 [get\_ports {dout[6]}]

set\_property PACKAGE\_PIN A3 [get\_ports {dout[5]}]

set\_property PACKAGE\_PIN A5 [get\_ports {dout[4]}]

set\_property PACKAGE\_PIN A4 [get\_ports {dout[3]}]

set\_property PACKAGE\_PIN F7 [get\_ports {dout[2]}]

set\_property PACKAGE\_PIN G8 [get\_ports {dout[1]}]

set\_property PACKAGE\_PIN H8 [get\_ports {dout[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports clrn]

set\_property IOSTANDARD LVCMOS33 [get\_ports {din[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {din[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {din[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {din[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {din[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {din[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {din[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {din[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports ds]

set\_property IOSTANDARD LVCMOS33 [get\_ports {dout[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {dout[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {dout[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {dout[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {dout[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {dout[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {dout[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {dout[0]}]

2.实验步骤

（1）创建工程

新建文件夹JG3，在该文件创建工程文件，并制定FPGA器件

1. 启动Vivado软件
2. 选择“RTL Project”，勾选“Do not specify sources at this time”
3. 指定FPGA器件
4. 出现总结界面时点击finnish，完成创建。

（2）模块设计

添加源文件。在其他文件编辑器里写好代码，添加到新建的工程中。

通过 Verilog HDL 语言编程，实现一个8位序列信号发生器，要求有1个时钟脉冲输入、1个异步清零端、1个8位预置数据输入端和1个串行数据输出端，序列检测器输入、输出端口设计具体要求如下，

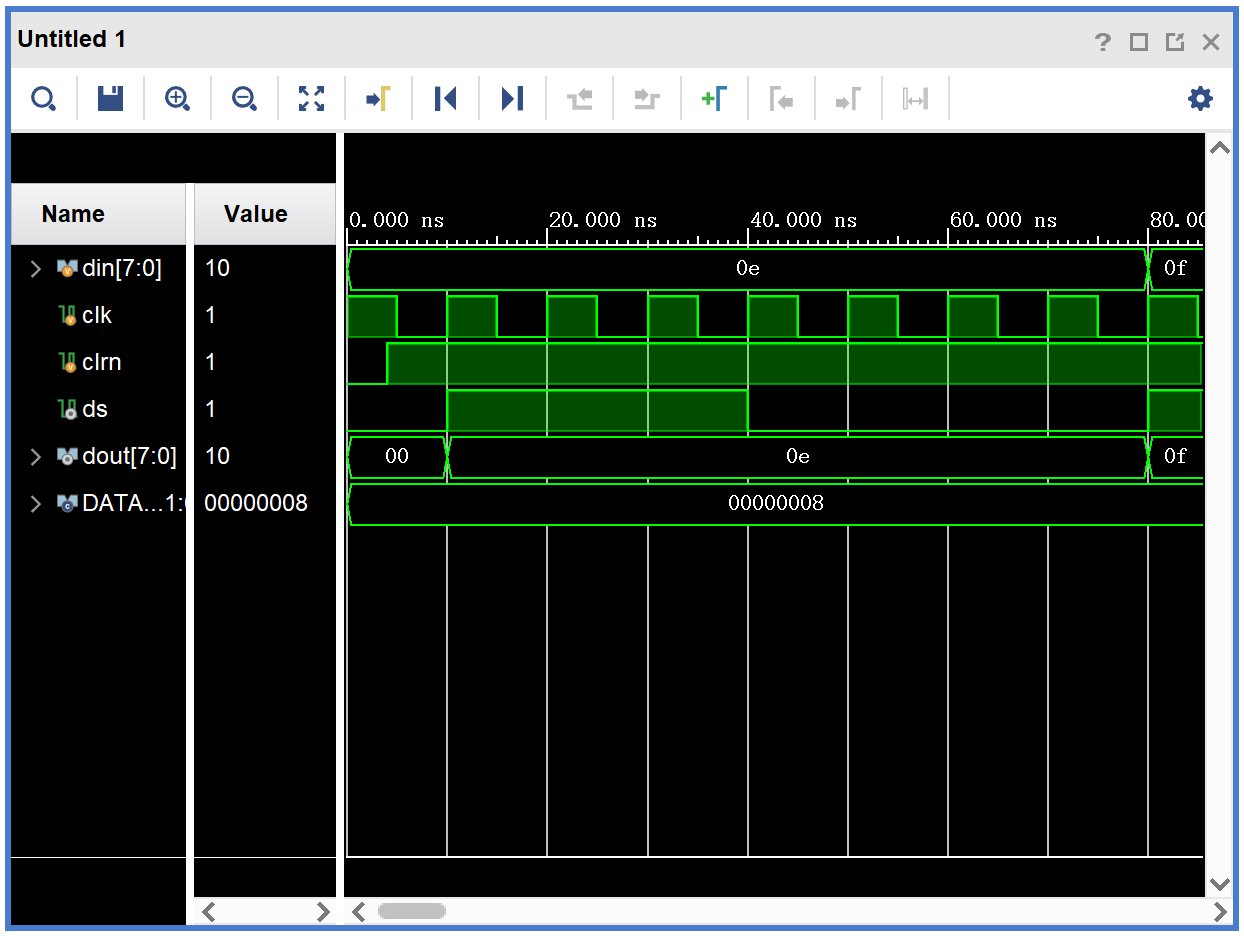
clrm：异步清零信号，低电平有效：  
clk：时钟脉冲输入，上升沿有效。  
din:8位预置数据输入端   
ds：串行数据输出端

（3）功能仿真

a.建立一个测试模块，一个比较完备的测试模块能够产生输入激励信号，送入到要测试的功能模块中，然后读出该功能模块的执行结果，并与预期结果进行比较，以此来验证功能模块的正确性。

b.波形仿真

点击“Run Simulation”,选择“Run Behavioral Simulation”没有语法错误会弹出仿真波形界面。



**六．实验结果验证及分析**

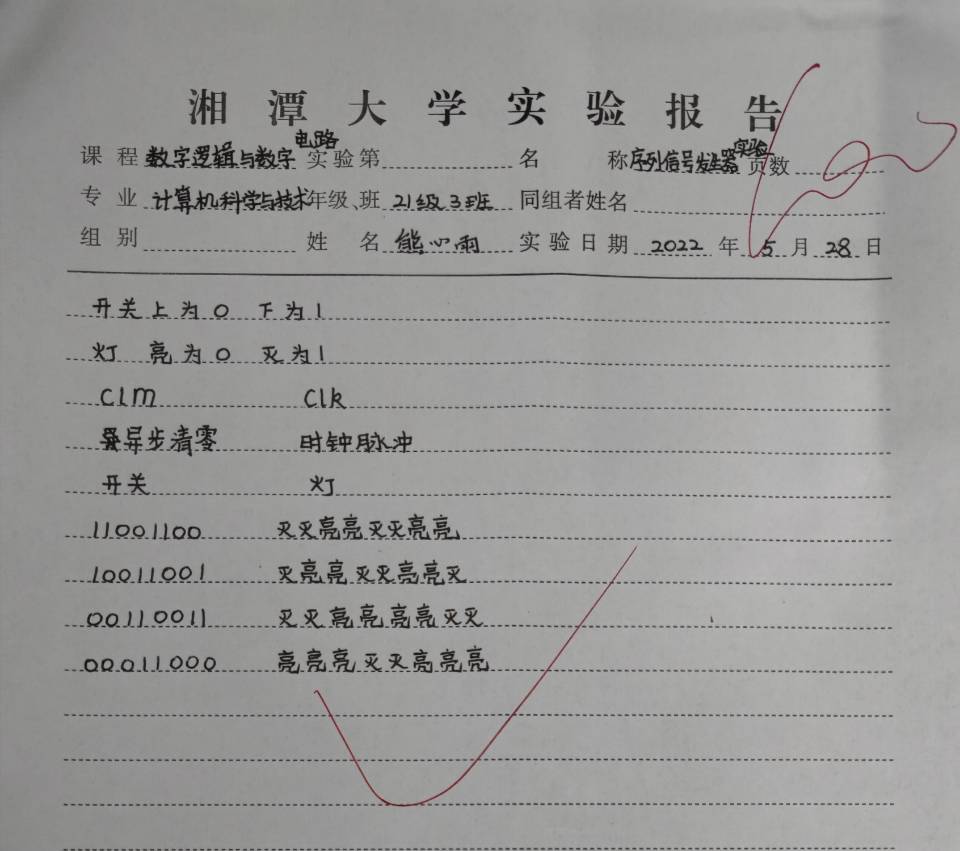
实验结果与预测结果一致

**七.实验过程中出现的问题及处理情况**

问题： 数码管无法显示正确的数据

处理办法： 对原代码进行修改

**八.实验操作评分照片**

****