**湘 潭 大 学 实 验 报 告**

**课程名称** 数字逻辑与电路实验 **实验名称** 序列信号发生器  **实验日期** 2021年5月29日 **班 级** 20级计算机科学与技术2班 **学 号** 201905190813 **姓 名**  洪雨芸

**同组者** 李玲玉、马欣悦

1. **实验目的**

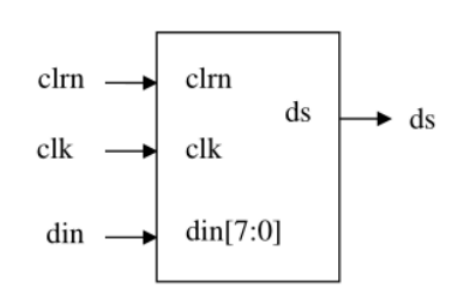
1、设计一个8位序列信号发生器，理解序列信号发生器的工作原理，掌握并入串出端口控制的描述方法。

2、熟悉Vivado，掌握层次化设计的方法，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**

1、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、一个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下图所示。



clrn: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。

3、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。

4、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。

**三、实验设备与软件环境**

设备：Cpu FPGA教学实验系统 V2.0 LS-CPU-EXB-001

环境：Verilog HDL

**四、实验方案或原理**

1. 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。

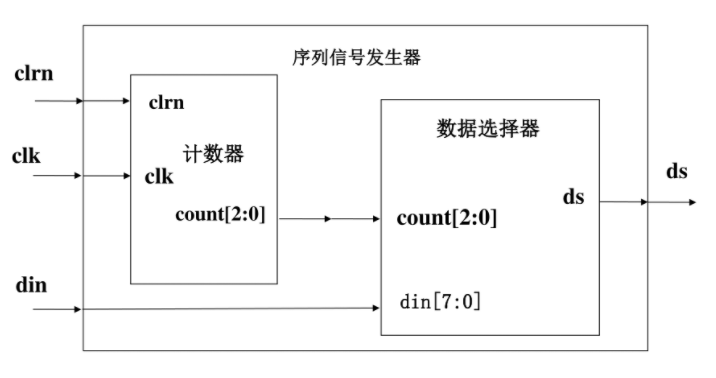
序列信号发生器的设计方法有多种。比如可用计数器和数据选择器组成序列信号发生器，如需产生一个8位的序列信号11010110 (时间顺序为自左向右),则可用一个8进制计数器和一个 8选1数据选择器组成。

1. 设计方案：

方案一：

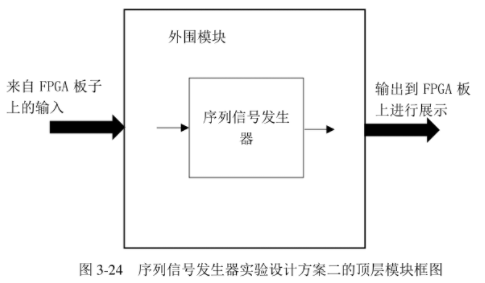
用拨码开关控制复位信号;按钮开关控制工作时钟;将8位预置数据由用拨码开关控制复位输入，串行数据输出端接LED灯。具体的引脚分配如下表。

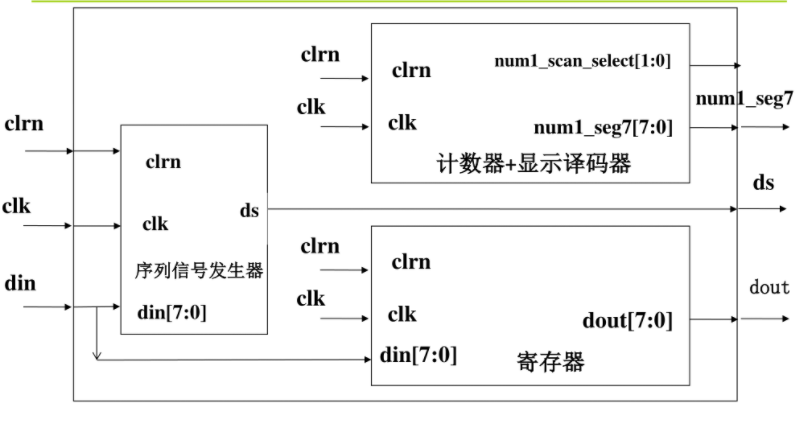
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | clrn | clk | din | ds |
| 按键名 | RSTn\_in | STEP1 | SW0、SW1、SW2、SW3、SW4、SW5、SW6、SW7 | LED9 |
| 引脚号 | Y3 | V6 | AC21、AD24、AC22、AC23、AB6、W6、AA7、Y6 | J8 |
| 功能 | 时钟与复位 | 单步调试 | 8个拨码开关，预置din[7:0] | LED灯 |



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，实现观察输出的串行数据值的变化。





**五、代码设计及实验步骤（含列表说明符号的意义、位数、信号有效）**

代码设计：

1. 源文件代码counter.v

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

1. 源文件代码mux81.v

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];

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

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

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

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

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

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

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

default:ds<=1'b0;

endcase

endmodule

源文件代码sequencer\_module.v

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 mux81\_1(count,din,ds);

endmodule

源文件register.v

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

源文件sequencer.v

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

input clk,clrn;

input [7:0]din;

output [7:0]dout;

output ds;

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

sequencer\_module seq\_1(clk,clrn,din,ds);

endmodule

1. 仿真测试文件代码sequencer\_tb.v

`include"counter.v"

`include"mux81.v"

`include"sequencer\_module.v"

`include"sequencer.v"

`include"register.v"

`timescale 1ns/1ns

module sequencer\_tb;

parameter DATA\_WIDTH=8;

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

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

wire ds;

reg clk,clrn;

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 seq\_1(.clk(clk),.clrn(clrn),.din(din),.ds(ds),.dout(dout));

endmodule

1. 约束文件代码sequencer.xdc

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]}]

实验步骤：

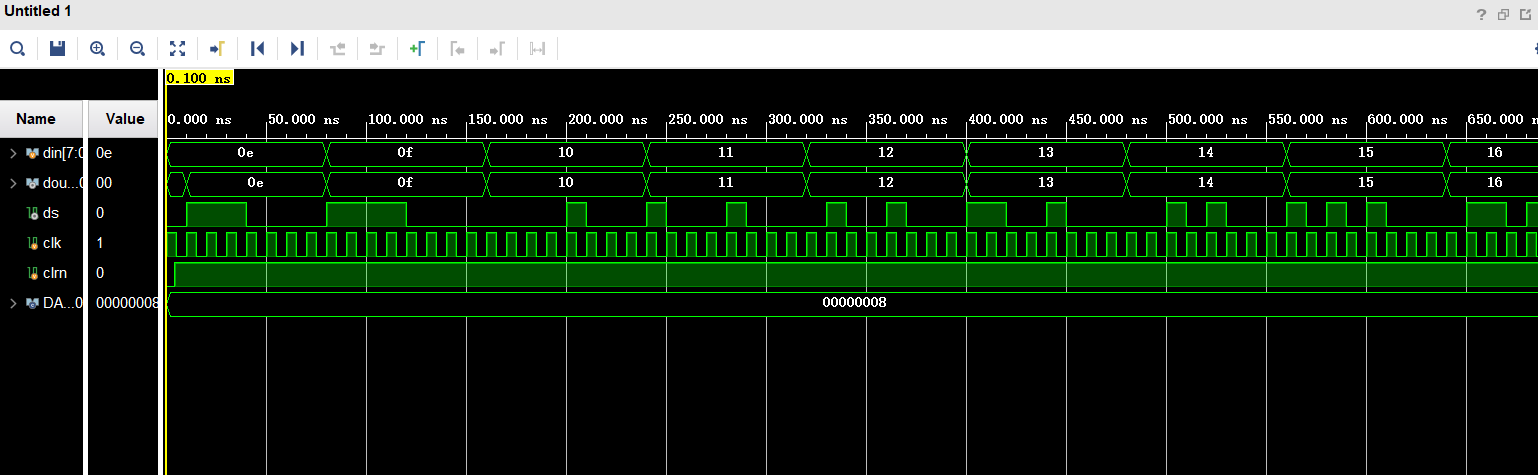
1. 创建工程

新建一个文件夹SEQUENCER，然后在该文件夹中用vivado软件创建新工程sequencer，并指定FPGA器件。

1. 模块设计

设计源代码并添加、编译、改错

1. 功能仿真

设计仿真测试代码sequencer\_tb.v并添加，进行波形仿真，仿真波形图如下图

1. 引脚绑定

根据实验方案设计约束文件代码sequencer.xdc并添加，

1. 综合、布局布线并产生可烧写文件、文件下载
2. 实验操作与数据记录

SW0~SW7对应din[7:0]的八位数，实验箱左边8个LED灯对应dout[7:0],第九个灯代表ds。

开关上拨代表0，下拨代表1，LED灯亮代表0，灭代表1。

**六、实验结果验证及分析**

如下图实验操作图片

测试结果符合实验预期。

**七、实验过程中出现的问题及处理情况（如排故障的方法等）或可研究与探索的问题与方法**

实验现象：文件综合下载失败。

原因分析：下载线与电脑连接后忘记打开电源

排除故障的方法：拔掉下载线后重新连接并打开电源，再次进行综合下载烧写成功

**八、实验操作评分照片**

