# 湘 潭 大 学 实 验 报 告

课程名称:数字逻辑与数字电路 实验名称: 移位寄存器实验 实验时间: 2022.5.21

同组人员：无 班级:计算机科学与技术3班 学号:202105566416 姓名:熊心雨

## 一、实验目的

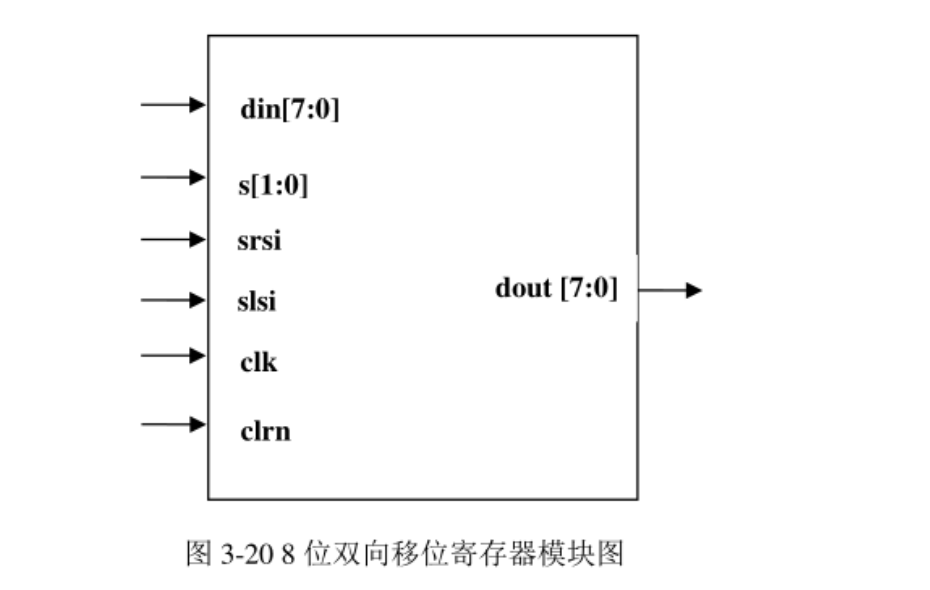
1）设计一个8位双向移位寄存器，理解移位寄存器的工作原理，掌握串入/并出端口控制的描述方法。

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。

## 二、实验内容和要求

1） 8位双向移位寄存器设计：

通过Verilog HDL编程，实现双向移位寄存器。



其输入、输出端口设计具体要求如下。

clrn:异步清零信号,低电平有效,为 clr=0时,8位输出端清 0。

clk:移位寄存器时钟脉冲输入,上升沿有效。

srsi:串行右移输入端。

slsi:串行左移输入端。

din:8位的并行数据输入端。

dout:8位数据并行输出端。

s:2位控制端,为00时,8 位输出端保持原来的状态不变;为 01 时,允许串行左移输入1位数据;为10时,允许串行右移输入 1位数据;为11时,允许 8 位数据从并行端口输入。

1. 仿真测试 ：

完成移位寄存器的设计编程和仿真测试等步骤，给出仿真波形。

1. 设计实验方案 ：

根据以上实验内容设计实验方案，进行项目设计、功能仿真和编译下载，完成实验操作并做好数据记录。

1. 撰写实验报告 ：

撰写实验报告，包括：程序设计、仿真分析、硬件测试和详细实验过程，程序分析报告、仿真波形图及其分析报告等。

## 三、实验设备与软件环境

1、装有Xilinx Vivado 的教学计算机一台。

2、LS-CPU-EXB-002 教学系统实验箱一套。

## 四、实验方案或原理

移位寄存器不仅具有存储代码的功能,在移位脉冲作用下,还有左移、右移等功能。设计一个8位二进制双向移位寄存器,能实现数据保持、右移、左移、并行置入和开行输出等功能，要求有 3 种输入方式:8 位并行输入、1 位左移串行输入、1位右移串行输入,有一种输出方式:8位并行输出。

双向移位寄存器的工作过程如下。

① 当1位数据从左移串行输入端输入时,先进入内部寄存器最高位,并在并行输出口最高位输出,后由同步时钟的上升沿触发向左移位。

② 当1位数据从右移串行输入端输入时,先进入内部寄存器最低位,并在并行输出口的最低位输出,后由同步时钟的上升沿触发向右移位。

## 五、代码设计及实验步骤

1）代码设计

**1.** **shifter.v**

module shifter(din,s,srsi,slsi,clk,clrn,dout);

input clrn,clk,srsi,slsi;

input [7:0]din;

input [1:0]s;

output [7:0]dout;

reg [7:0]dout;

always @ (negedge clrn or posedge clk)

begin

if(clrn==1'b0) dout <= 8'b00000000;

else

case(s)

2'b00:dout<=dout;

2'b01: begin dout<=(dout<<1); dout[0]<=slsi; end

2'b10: begin dout<=(dout>>1); dout[7]<=srsi; end

2'b11:dout<=din;

default:dout<=dout;

endcase

end

endmodule

**2.** **shifter\_display.v**

module shifter\_display(

input clk,

input resetn,

input [1:0]input\_sel,

input clock,

output [7:0]dout,

output lcd\_rst,

output lcd\_cs,

output lcd\_rs,

output lcd\_wr,

output lcd\_rd,

inout [15:0]lcd\_data\_io,

output lcd\_bl\_ctr,

inout ct\_int,

inout ct\_sda,

output ct\_scl,

output ct\_rstn);

reg [7:0]din;

reg [1:0]s;

reg slsi,srsi;

shifter shifter\_module(.din(din),.s(s),.srsi(srsi),.slsi(slsi),.clk(clock),.clrn(resetn),.dout(dout));

reg display\_valid;

reg [39:0]display\_name;

reg [31:0]display\_value;

wire [5:0]display\_number;

wire input\_valid;

wire [31:0]input\_value;

lcd\_module lcd\_module(

.clk (clk),

.resetn (resetn),

.display\_valid (display\_valid),

.display\_name (display\_name),

.display\_value (display\_value),

.display\_number (display\_number),

.input\_valid (input\_valid),

.input\_value (input\_value),

.lcd\_rst (lcd\_rst),

.lcd\_cs (lcd\_cs),

.lcd\_rs (lcd\_rs),

.lcd\_wr (lcd\_wr),

.lcd\_rd (lcd\_rd),

.lcd\_data\_io (lcd\_data\_io),

.lcd\_bl\_ctr (lcd\_bl\_ctr),

.ct\_int (ct\_int),

.ct\_sda (ct\_sda),

.ct\_scl (ct\_scl),

.ct\_rstn (ct\_rstn));

always@(posedge clk)

begin

if(!resetn)

begin

din <= 4'd0;

end

else if(input\_valid&&input\_sel==2'b00)

begin

din <= input\_value[7:0];

end

end

always @(posedge clk)

begin

if(!resetn)

begin

s <= 32'd0;

end

else if(input\_valid&&input\_sel==2'b11)

begin

s <= input\_value[1:0];

end

end

always @(posedge clk)

begin

if(!resetn)

begin

slsi <= 32'd0;

end

else if(input\_valid&&input\_sel==2'b10)

begin

slsi <= input\_value[0];

end

end

always @(posedge clk)

begin

if(!resetn)

begin

srsi <= 32'd0;

end

else if(input\_valid&&input\_sel==2'b01)

begin

srsi <= input\_value[0];

end

end

always @(posedge clk)

begin

case(display\_number)

6'd1:

begin

display\_valid <= 1'b1;

display\_name <="CLRN";

display\_value <=resetn;

end

6'd2:

begin

display\_valid <= 1'b1;

display\_name <="DIN";

display\_value <={24'b0,din};

end

6'd3:

begin

display\_valid <= 1'b1;

display\_name <="S";

display\_value <={30'b0,s};

end

6'd4:

begin

display\_valid <= 1'b1;

display\_name <="SRSI";

display\_value <=srsi;

end

6'd5:

begin

display\_valid <= 1'b1;

display\_name <="SLSI";

display\_value <=slsi;

end

6'd6:

begin

display\_valid <= 1'b1;

display\_name <="DOUT";

display\_value <={24'b0,dout};

end

default:

begin

display\_valid <= 1'b0;

display\_name <= 40'd0;

display\_value <= 32'd0;

end

endcase

end

endmodule

**3.** **shifter\_tb.v**

`timescale 1ns/1ns

`include"shifter.v"

module shifter\_tb;

parameter bit\_width=8;

reg srsi,slsi,clk,clrn;

reg [1:0]s;

reg [bit\_width-1:0]din;

wire [bit\_width-1:0]dout;

initial begin

clk<=0; clrn<=1; din=8'b00001111; srsi=1; slsi=0; s<=3;

#0 $display("time\tdin\ts\tsrsi\tslsi\tclk\tclrn\tdout");

#1 clrn<=0; din=8'b00001101;

#20 clrn<=1; s<=3;

#20 s<=0;

#20 s<=1;din=8'b10001101;

#20 s<=2;din=8'b10011101;

#20 s<=3;din=8'b11001101;

end

always #1 clk=~clk;

shifter sft(din,s,srsi,slsi,clk,clrn,dout);

endmodule

**4.** **shifter.xdc**

set\_property PACKAGE\_PIN AC19 [get\_ports clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

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

set\_property IOSTANDARD LVCMOS33 [get\_ports resetn]

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

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

set\_property IOSTANDARD LVCMOS33 [get\_ports clock]

set\_property PACKAGE\_PIN AC21 [get\_ports {input\_sel[1]}]

set\_property PACKAGE\_PIN AD24 [get\_ports {input\_sel[0]}]

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

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

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 {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]}]

set\_property PACKAGE\_PIN J25 [get\_ports lcd\_rst]

set\_property PACKAGE\_PIN H18 [get\_ports lcd\_cs]

set\_property PACKAGE\_PIN K16 [get\_ports lcd\_rs]

set\_property PACKAGE\_PIN L8 [get\_ports lcd\_wr]

set\_property PACKAGE\_PIN K8 [get\_ports lcd\_rd]

set\_property PACKAGE\_PIN J15 [get\_ports lcd\_bl\_ctr]

set\_property PACKAGE\_PIN H9 [get\_ports {lcd\_data\_io[0]}]

set\_property PACKAGE\_PIN K17 [get\_ports {lcd\_data\_io[1]}]

set\_property PACKAGE\_PIN J20 [get\_ports {lcd\_data\_io[2]}]

set\_property PACKAGE\_PIN M17 [get\_ports {lcd\_data\_io[3]}]

set\_property PACKAGE\_PIN L17 [get\_ports {lcd\_data\_io[4]}]

set\_property PACKAGE\_PIN L18 [get\_ports {lcd\_data\_io[5]}]

set\_property PACKAGE\_PIN L15 [get\_ports {lcd\_data\_io[6]}]

set\_property PACKAGE\_PIN M15 [get\_ports {lcd\_data\_io[7]}]

set\_property PACKAGE\_PIN M16 [get\_ports {lcd\_data\_io[8]}]

set\_property PACKAGE\_PIN L14 [get\_ports {lcd\_data\_io[9]}]

set\_property PACKAGE\_PIN M14 [get\_ports {lcd\_data\_io[10]}]

set\_property PACKAGE\_PIN F22 [get\_ports {lcd\_data\_io[11]}]

set\_property PACKAGE\_PIN G22 [get\_ports {lcd\_data\_io[12]}]

set\_property PACKAGE\_PIN G21 [get\_ports {lcd\_data\_io[13]}]

set\_property PACKAGE\_PIN H24 [get\_ports {lcd\_data\_io[14]}]

set\_property PACKAGE\_PIN J16 [get\_ports {lcd\_data\_io[15]}]

set\_property PACKAGE\_PIN L19 [get\_ports ct\_int]

set\_property PACKAGE\_PIN J24 [get\_ports ct\_sda]

set\_property PACKAGE\_PIN H21 [get\_ports ct\_scl]

set\_property PACKAGE\_PIN G24 [get\_ports ct\_rstn]

set\_property IOSTANDARD LVCMOS33 [get\_ports lcd\_rst]

set\_property IOSTANDARD LVCMOS33 [get\_ports lcd\_cs]

set\_property IOSTANDARD LVCMOS33 [get\_ports lcd\_rs]

set\_property IOSTANDARD LVCMOS33 [get\_ports lcd\_wr]

set\_property IOSTANDARD LVCMOS33 [get\_ports lcd\_rd]

set\_property IOSTANDARD LVCMOS33 [get\_ports lcd\_bl\_ctr]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[8]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[9]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[10]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[11]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[12]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[13]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[14]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {lcd\_data\_io[15]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports ct\_int]

set\_property IOSTANDARD LVCMOS33 [get\_ports ct\_sda]

set\_property IOSTANDARD LVCMOS33 [get\_ports ct\_scl]

set\_property IOSTANDARD LVCMOS33 [get\_ports ct\_rstn]

2）创建工程 ：

在E盘新建一个文件夹SHIFTER\_DISPLAY并且将设计源代码shifter\_display.v、shifter.v、lcd\_module.dcp以及仿真测试代码shifter\_tb.v都放在该文件夹下，后在该文件夹下创建工程。

启动Vivado软件,在Quick Start中点击 Create Project，出现新建工程向导，选择“Next”，输入工程名称shifter，选择工程的文件位置位E:/SHIFTER\_DISPLAY。

3）添加设计模块：

在工程中添加设计文件和仿真测试文件,并指定FPGA器件的型号为“Arix”的“xc7a200fbg676-2”

4）功能仿真：

在左侧的导航栏中的“Run simulation“中选择”Run Behaviorl simulation“进行仿真。

5）实验方案设计 ：

通过 LCD 触摸屏输入并行数据、控制信号和左右串入,并显示控制信号和并行输出数据。前面设计的外围展示模块shifter\_display.v 的功能是调用 LCD 触摸屏,完成下载进行验证。

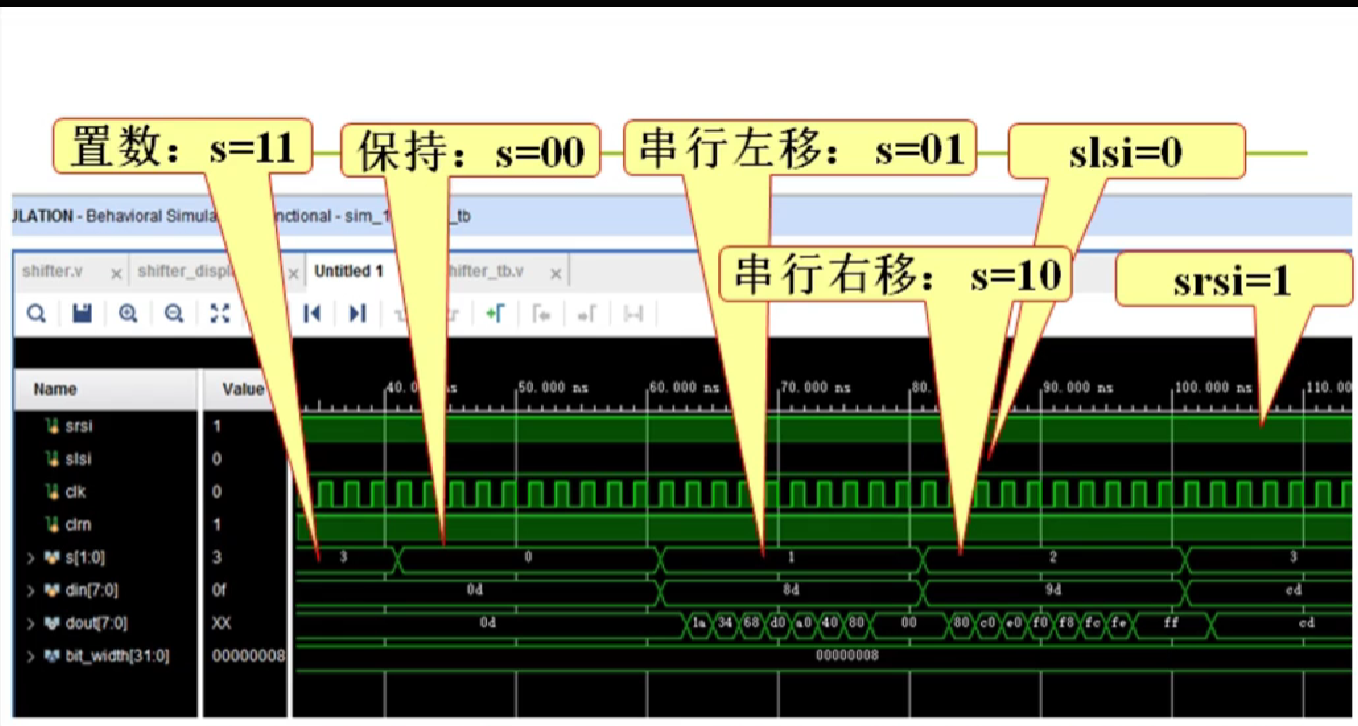
通过拨码开关 SW0和 SW1 选择输入数,拨上为0,拨下为1。

6）创建约束文件

7）生产下载文件

8）下载

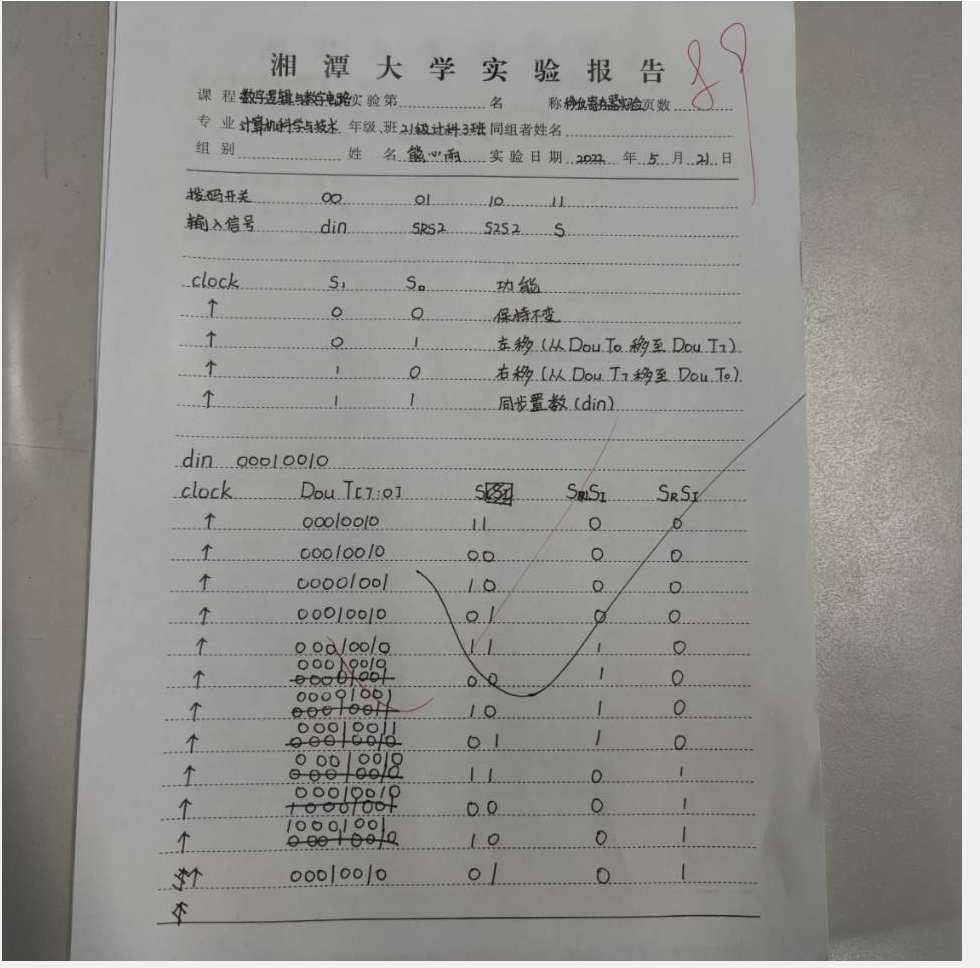
## 六、实验结果验证及分析（包括实验现象及原因分析）



## 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。

无

## 八、实验操作评分照片



右移