南京大学

数电实验六：

移位寄存器及桶形移位器

课程名称：数字逻辑与计算机组成实验

姓名：孙文博

学号：201830210

班级：数电一班

邮箱：201830210@smail.nju.edu.cn

实验时间：2022.4.1 - 4.14

一、 实验目的

1. 复习数字电路中寄存器的概念和原理；

2. 了解移位寄存器和桶形移位器的工作原理；

3. 用 Verilog 语言模拟实现移位寄存器和桶形移位器；

4. 设计一个移位寄存器实现的随机数发生器。

二、 实验环境

设计\编译环境：Quartus (Quartus Prime 17.1) Lite Edition

开发平台：DE10-Standard

FPGA 芯片：Cyclone II 5CSXFC6D6

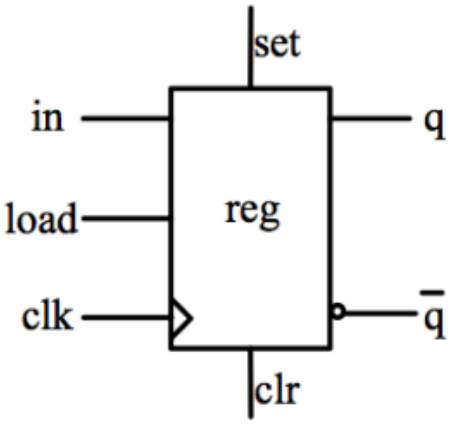
三、 实验原理

1. 寄存器概要

寄存器与存储器都是数字系统中的记忆器件，用来保存程序和数据，计算机内部除了主存以外，还有许多记录状态的通用寄存器，如程序计数器 PC 等， CPU 的状态也由这些通用寄存器组中的内容决定。FPGA 芯片通过触发器实现寄存器的功能，其逻辑如下：

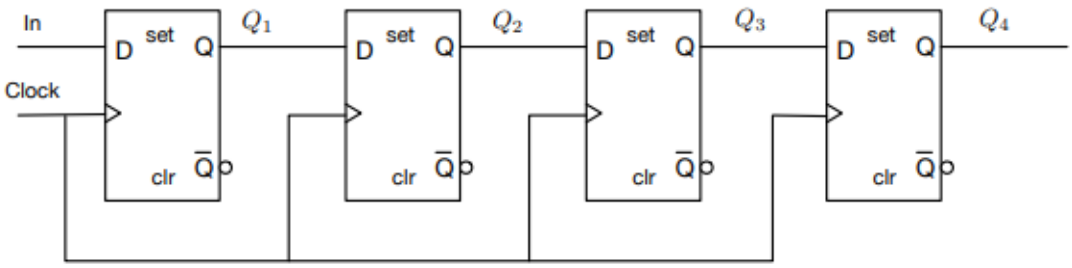
```
module register1(load,clk,clr,inp,q);
  input  load,clr,clk,inp;
  output reg q;

  always @(posedge clk)
    if (clr==1)
        q <= 0;
    else if (load == 1)
         q <= inp;
endmodule
```

上面的一位寄存器还包括清零端和写使能端，只有当写使能有效时才会写入数据，以防数据被篡改。如果将多个寄存器组合在一起就构成了寄存器组，可以用 Verilog 语言中的数组实现，代码类似于上述的一位寄存器的实现。寄存器的数字电路符号如下：



2. 移位寄存器

移位寄存器也是数字系统的常用器件。它可以在时钟的触发沿，根据其控制信号，将存储在其中的数据全体向某个方向**移动一位**。具体而言，又分为左移/右移，算术/逻辑移位，其中算术移位是指考虑到符号位的移位，要保证符号位不改变，因此算术左移同逻辑左移一样右边补 0，算术右移最左面的空位补符号位，逻辑移位则都是补 0；此外还有循环移位的方式，它是将将移出去的那一位补充到空出的最高/低位的移位方式。同时移位寄存器还具有置数功能，可以在时钟有效沿将一个 8 位的数据输入到寄存器中，即给寄存器赋一个初始值。移位寄存器的数字电路框图如下：



实现代码如下：

```
    always @(posedge clk) begin
        case(func)
            0: dout=0;
            1: dout=din;
            2: //逻辑右移
            begin
                for(i=1;i<=7;i=i+1)
                    dout[i-1]=dout[i];
                dout[7]=0;
            end
            3: //逻辑左移
            begin
                for(i=7;i>=1;i=i-1)
                    dout[i]=dout[i-1];
                dout[0]=0;
            end
            4: //算术右移
            begin
                for(i=1;i<=7;i=i+1)
                    dout[i-1]=dout[i];
                dout[7]=dout[6];
            end
            5:
            begin
                for(i=1;i<=7;i=i+1)
                    dout[i-1]=dout[i];
            5:
            begin
                for(i=1;i<=7;i=i+1)
                    dout[i-1]=dout[i];
                dout[7]=lin;
            end
            6:
            begin
                x=dout[0];
                for(i=1;i<=7;i=i+1)
                    dout[i-1]=dout[i];
                dout[7]=x;
            end
            7:
            begin
                x=dout[7];
                for(i=7;i>=1;i=i-1)
                    dout[i]=dout[i-1];
                dout[0]=x;
            end
        endcase
    end
```

3. 桶形移位器

在 CPU 中，我们往往需要对数据进行多次移位操作，但是传统的移位寄存器一个周期只能移动一位，当要进行多位移位时需要多个时钟周期，效率较低。因此我们需要一种采用组合逻辑的方式来实现同时移动多位数据的桶形移位器，在效率上优势极大。桶形移位器也常常被用在 ALU 中来实现移位。它的输入端相比普通移位寄存器多了应该移动位数 shamt，修改之前的移位寄存器代码可以得到桶形移位器：

```
12      always @(*) begin
13          sign=indata[31];
14          outdata=indata;
15
16          //逻辑右移
17          if(lr==0 && al==0) begin
18
19              for(i=shamt;i<=31;i=i+1)
20                  outdata[i-shamt]=outdata[i];
21              for(i=0;i<shamt;i=i+1)
22                  outdata[31-i]=0;
23
24          end
25
26          //算术右移
27          else if(lr==0 && al==1) begin
28
29              for(i=shamt;i<=31;i=i+1)
30                  outdata[i-shamt]=outdata[i];
31              for(i=0;i<shamt;i=i+1)
32                  outdata[31-i]=sign;
33
34          end
```

```
36          //逻辑左移
37          else if(lr==1 && al==0) begin
38
39              for(i=0;i<=31-shamt;i=i+1)
40                  outdata[31-i]=outdata[31-i-shamt];
41
42              for(i=0;i<shamt;i=i+1)
43                  outdata[i]=0;
44
45          end
46
47          //算术左移和逻辑左移相同
48          else if(lr==1 && al==1) begin
49
50              for(i=0;i<=31-shamt;i=i+1)
51                  outdata[31-i]=outdata[31-i-shamt];
52              for(i=0;i<shamt;i=i+1)
53                  outdata[i]=0;
54
55          end
56      end
```

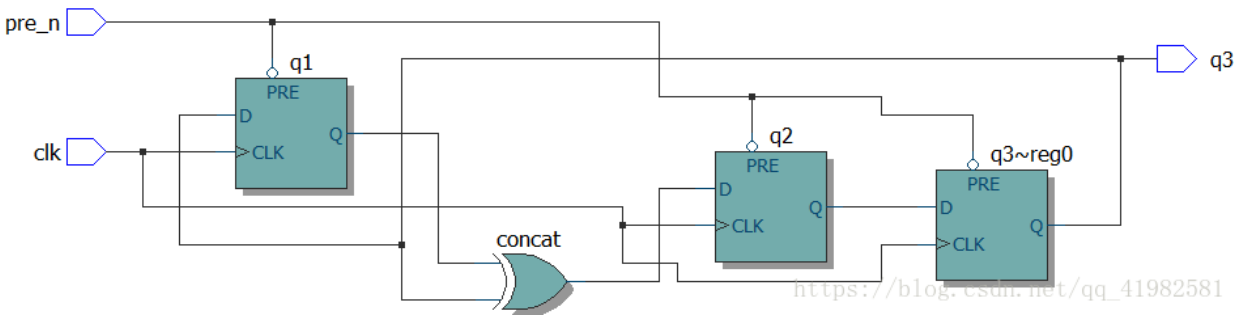
注意到这里数据分为两部分（移位的 shamt 位数据和剩下的 31-shamt 位数据），通过两个 for 循环分别实现两部分的数据赋值。此外该部分电路是纯组合逻辑的，不需要时钟信号控制。

四、 实验过程

1. LFSR

LFSR（Linear-feedback shift register）是一种特殊的移位寄存器，他的输入取决于其先前状态，可以使用 n 位移位寄存器生成长度为 $2^n - 1$ 的二进制循环序列。这类序列的片段在表观上是随机的，所以被广泛用于通信中的随机序列生成。LFSR 的使用异常广泛，涉及到方方面面，最常见的应用包括我们计算机网络中正在学习的 CDMA 编码技术。

一个简单的 LFSR 计数器的数字电路图如下：



2. 利用移位寄存器实现随机数发生器

本实验中我们基于 LFSR 的思想，用一个八位右移移位寄存器实现一个随机数发生器，假设从左到右的比特以 x7 x6 x5 x4 x3 x2 x1 x0 表示，每个时钟周期右移一位，x0 将被移出，最左边移入的 x7’按照以下公式（上个周期的值）重新计算：

$$x_8 = x_4 \oplus x_3 \oplus x_2 \oplus x_0$$

实现的 Verilog 代码如下：

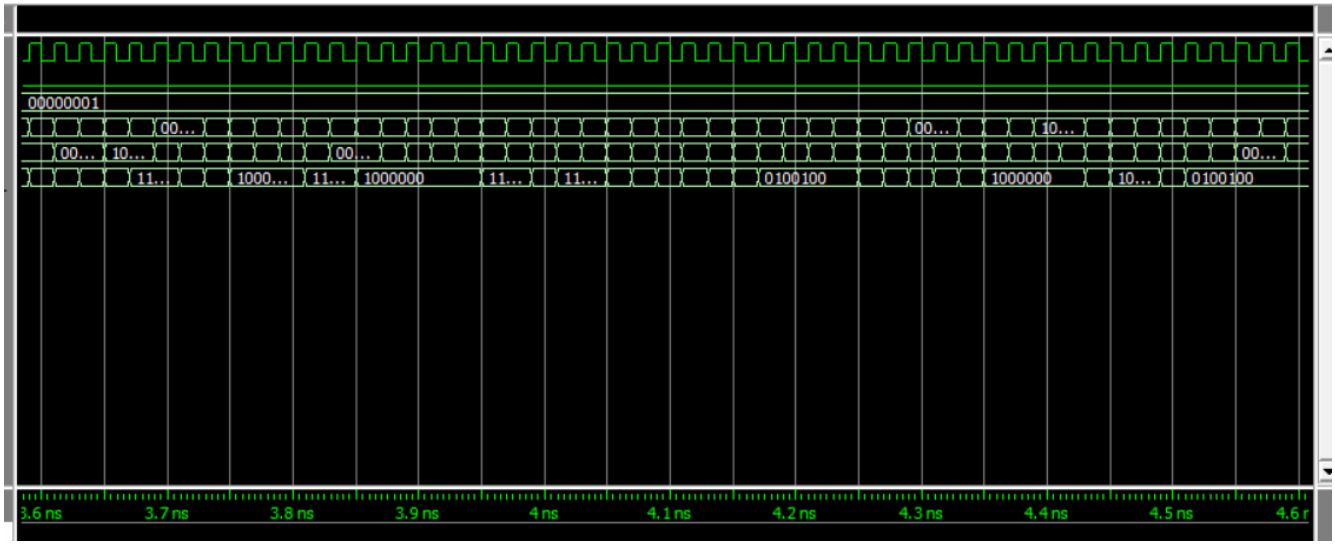
```
module exp_6(
    input  [7:0]  seed,
    input  clk,
    input  load,
    output [6:0] LED_0,
    output [6:0] LED_1,
    output [6:0] LED_2
    );

    reg x;
    reg [7:0] out;
    reg [6:0] led0;
    reg [6:0] led1;
    reg [6:0] led2;
    integer i=0;

    always @(posedge clk) begin

        if(load)
            out = seed;

        else begin
            x=out[0] ^ out[2] ^ out[3] ^ out[4];
            for(i=1;i<=7;i=i+1)
                out[i-1]=out[i];
            out[7]=x;
        end
```

其中输入端为时钟信号（由开发板上的按钮手动控制），八位二进制初始值 seed，置数使能 load（当 load 为 1 时将当前移位寄存器初值设为 seed），输出端 out 为移位寄存器中的值（0-255），通过三个七段数码管以十进制方式显示。

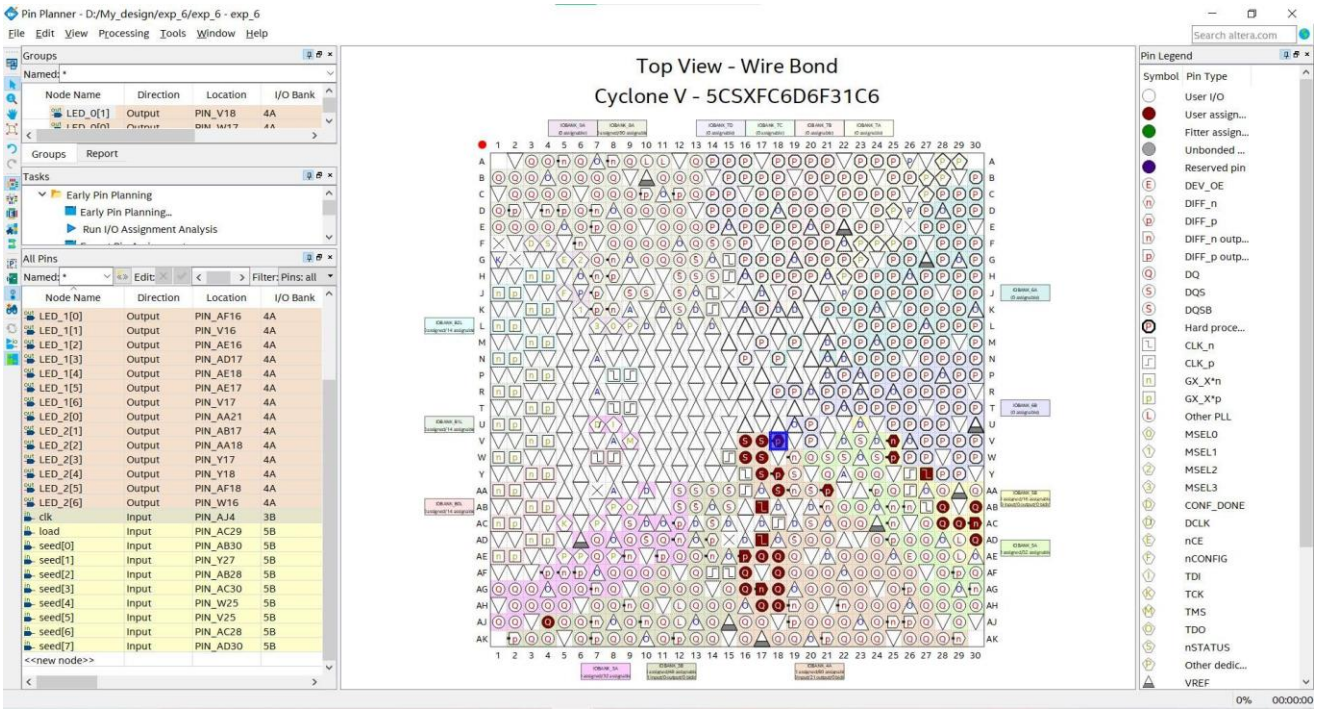
设计 testbench 文件进行仿真检测：

```
initial
begin
// code that executes only once
// insert code here --> begin
    load = 1;
    seed = 1;
    clk=1;
    #10;
    clk=0;
    #10;
    load = 0;
// --> end
end
always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
        #10;
        clk = ~clk;
// --> end
end
endmodule
```

注意，这里我们在 always 语块中为 clk 变量赋值，这样即可自动模拟时钟上升下降的变化，检测结果如下，可以看到输出端的值在一个周期内（255 个时钟周期）是个随机序列：



接下来将我们的 verilog 文件编译运行无误后，分配引脚，烧录到开发板上，等待验收。



五、 实验结果

1. 思考题

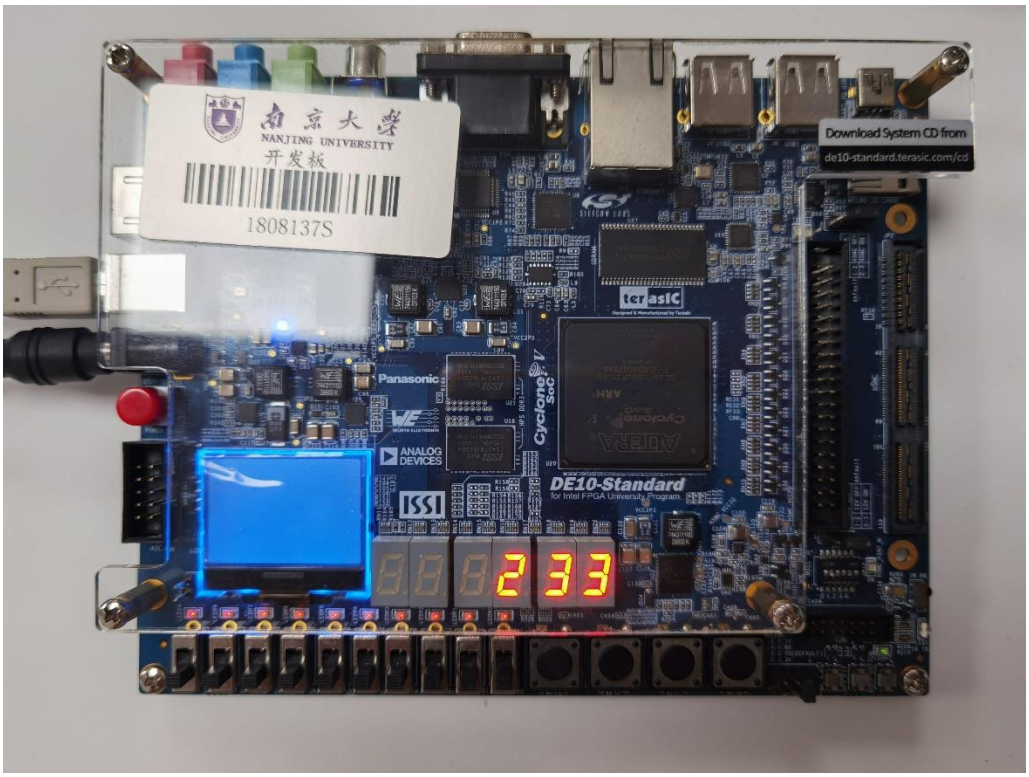
☞ **思考题：**

生成的伪随机数序列仍然有一定的规律，如何能够生成更加复杂的伪随机数序列?

有两种方法可以实现，一种是通过增加位数，因为周期为 $2^n - 1$，当 n = 64 时，周期数高达 18,446,744,073,709,551,616，即使是 3GHz 的时钟每秒移位 10^9 次，也需要运行 585 年，因此近乎是没有规律的；另一种方法是赋随机的初始值，即我们可以随机改变初始值的一个比特位，也会使整个伪随机数序列发生巨大变化，从而变得不可预测。

2. 上板验收（随机数发生器）

我们实现的 255 周期伪随机数发生器如下：



其中前八个开关控制初始值 seed，第九个开关 sw8 控制 load 使能，时钟通过一个按钮 key0 控制。验收中每按一下都会显示一个新的随机数（三位十进制）。

完整的验收视频已在附件中上传到 cms 网站。

六、 总结与反思

本次实验是目前为止进行最快得实验，不包括实验报告大概只花了半小时左右，主要源于之前对 Quartus 软件的熟练使用以及上次实验中对寄存器的理解。此外如果加以思考，可以发现实验手册开头部分引用的小说节选并非毫无关系，反而是与我们的实验息息相关。这里利用形象化的手法，将移位寄存器抽象理解为爱丽丝、三月兔、睡鼠和疯帽子之间的位子挪动，于是冷冰冰的数字电路便突然活泼了起来，不得不佩服老师和助教编写手册时的别出心裁，也希望我们可以带着这份兴趣继续接下来的实验！

实验六　　移位寄存器及桶形移位器

2022 年春季学期

“我想要只干净的茶杯，”帽匠插嘴说，“咱们全部挪动一下位子吧！”

说着，他就挪了一个地方，睡鼠紧随其后，三月兔就挪到了睡鼠的位子上，爱丽丝也只好很不情愿地坐到了三月兔的位子上。这次挪动唯一得到好位子的是帽匠，爱丽丝的位子比以前差了，因为刚才三月兔把牛奶打翻在位子上了。

–《爱丽丝漫游奇境记》刘易斯·卡罗尔