数字逻辑与计算机组成实验

LAB 06：移位寄存器及桶形移位器

郑凯琳 205220025
205220025@smail.nju.edu.cn

（一）实验目的

利用移位寄存器实现随机数发生器

用一个 8 位右移移位寄存器，从左到右的比特以 $x_7\,x_6\,x_5\,x_4\,x_3\,x_2\,x_1\,x_0$ 表示。每个时钟周期右移一位，x_0 被移出，最左边移入的位按照上一周期的值计算：

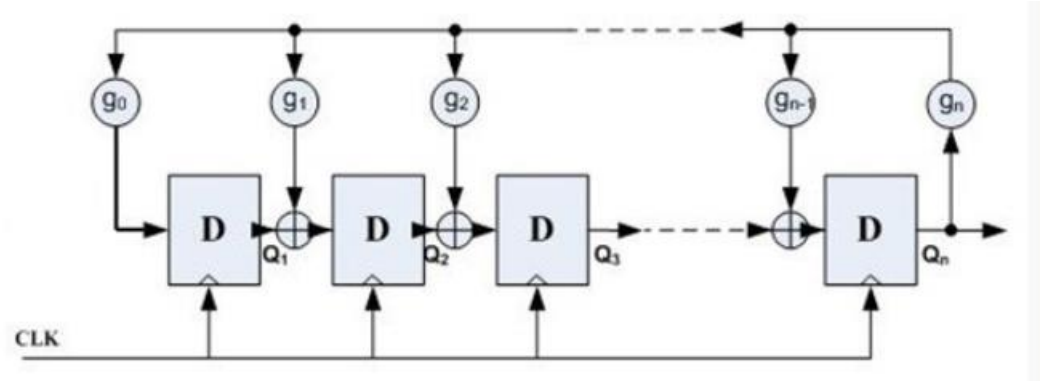
$$x_8 = x_4 \oplus x_3 \oplus x_2 \oplus x_0$$

例如，初始二进制值为 00000001 时，移位寄存器的状态将按 00000001 → 10000000 → 01000000 → 001000000 → 00010000 → 10001000 ... 变化。该序列的周期为 255。当然，当初始值为全零时，系统将一直停留在全零状态，所以需要对全零状态进行特殊处理。

实现一个 8 位的周期为 255 的伪随机序列，以按钮为时钟信号，并请将 8 位二进制数以十六进制显示在数码管上，在 Nexys A7-100T 开发板上观察生成的随机数序列。

（二）实验原理

经典的 LFSR（线性反馈移位寄存器，Linear-feedback shift register）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。例如，在 CDMA 通信中的长码的长度就是 $2^{42} - 1$ 的伪随机序列。



LSFR 由 n 个 D 触发器和若干个异或门组成。其中，g_n 为反馈系数，取值只能为 0 或 1，取为 0 时表明不存在该反馈之路，取为 1 时表明存在该反馈之路。

n 个 D 触发器最多可以提供 2^n-1 个状态（不包括全 0 的状态），为了保证这些状态没有重复，g_n 的选择必须满足一定的条件。

同时可以发现，D 触发器的个数越多，产生的状态就越多，也就越“随机”。

（三）实验环境/器材等

硬件器材：Nexys A7-100T 开发板

软件平台：Vivado 开发平台

（四）实验过程

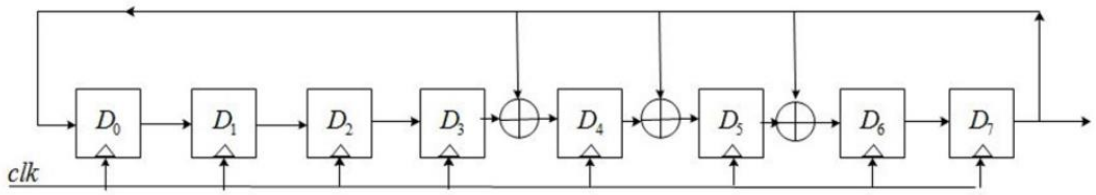
数字抽象

1. 输入：
   - clk —— 时钟信号，与分频器的输出时钟信号连接
   - rst —— 清零信号，同步清零
   - en —— 置数信号，同步置数，为 1 时读入 seed
   - seed [7:0] —— 置数的数据输入
2. 输出：
   - LED [7:0] —— 随机数输出
   - AN [7:0] —— 七段 LED 数码管
   - HEX [6:0] —— 数码管上的 LED

设计思路 & 设计代码

1. 线性反馈方程：$x_8 = x_4 \oplus x_3 \oplus x_2 \oplus x_0$，算出新的一位后向左移。
2. 设计一个时间周期为 1 秒钟的时钟，令随机数的显示每秒切换一次。
3. 随机数为 0 时，对其进行单独处理，使其能够自启动。

LSFR 整体结构：




```
always @ (posedge clk_1s)
begin
    if(rst) // 清零
        LED <= 0;
    else if(en) // 输入
        LED <= seed;
    else if(LED == 0)
        LED <= 8'b10001000;
    else
        LED <= {(LED[4] ^ LED[3] ^ LED[2]^ LED[0]), LED[7:1]};
end
```

测试代码

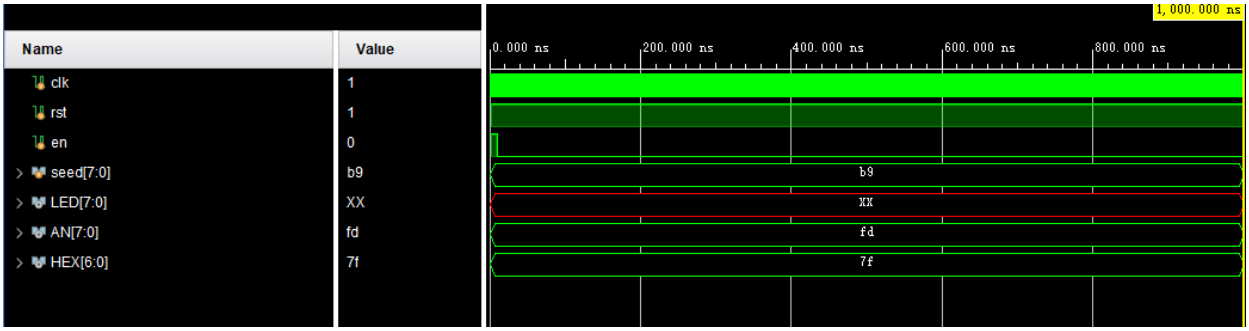
```
initial
begin
    clk = 0; seed = 8'b10111001; rst = 1; en = 1; #10;
    en = 0; #2000;
    $stop;
$display("RUnning testbench");
end

always
begin
   clk = ~clk; #1;
end
```

硬件实现（引脚分配）

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
#create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { seed[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { seed[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { seed[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { seed[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { seed[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { seed[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { seed[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { seed[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
#set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
#set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { SW[9] }]; #IO_25_34 Sch=sw[9]
#set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
#set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { SW[11] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
#set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { SW[12] }]; #IO_L24P_T3_35 Sch=sw[12]
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { SW[13] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
#set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { SW[14] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { en }]; #IO_L21P_T3_DQS_14 Sch=sw[15]
## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { LED[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { LED[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { LED[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { LED[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { LED[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { LED[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { LED[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { LED[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
#set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { LED[8] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
#set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports { LED[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
#set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports { LED[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
#set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { LED[11] }]; #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]
#set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { LED[12] }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]
#set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports { LED[13] }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
#set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports { LED[14] }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
#set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 } [get_ports { LED[15] }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]

##7 segment display
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { HEX[0] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { HEX[1] }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { HEX[2] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { HEX[3] }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { HEX[4] }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { HEX[5] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { HEX[6] }]; #IO_L4P_T0_D04_14 Sch=cg
#set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { DP }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { AN[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { AN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { AN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { AN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { AN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { AN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { AN[6] }]; #IO_L23P_T3_35 Sch=an[6]
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { AN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]

##CPU Reset Button
#set_property -dict { PACKAGE_PIN C12   IOSTANDARD LVCMOS33 } [get_ports { CPU_RESETN }]; #IO_L3P_T0_DQS_AD1P_15 Sch=cpu_resetn

##Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { rst }]; #IO_L9P_T1_DQS_14 Sch=btnc
#set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports { BTNU }]; #IO_L4N_T0_D05_14 Sch=btnu
#set_property -dict { PACKAGE_PIN P17   IOSTANDARD LVCMOS33 } [get_ports { BTNL }]; #IO_L12P_T1_MRCC_14 Sch=btnl
#set_property -dict { PACKAGE_PIN M17   IOSTANDARD LVCMOS33 } [get_ports { BTNR }]; #IO_L10N_T1_D15_14 Sch=btnr
```

（五）实验结果

仿真结果:



（六）实验中遇到的问题及解决方法

无

（七）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?

答：两种方法

方法 1：根据 m 序列的相关原理构造伪随机数序列

m 序列相关原理:

递推方程:

$$a_k = \sum_{i=1}^{n} c_i \, a_{k-i}$$

特征方程:

$$\mathrm{f(x)} = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n = \sum_{i=0}^{n} c_i x^i$$

若一个 n 次多项式 f(x)满足下列条件:

1）f(x)为既约的;

2）f(x)可整除($x^m + 1$), $m = 2^n - 1$;

3）f(x)除不尽($x^q + 1$), $q < m$;

则称 f(x)为本原多项式。

8 位二进制数的本原多项式为 $x^8 + x^4 + x^3 + x^2 + 1$。由此可以得到新的随机数生成方式如下:

```
Q <= {(Q[4]^Q[5]^Q[6]^Q[0]), Q[7:1]};
```

该随机数生成方式需要在开始前对 Q 进行初始化为一个非全 0 数。

方法二：通过外接设备收集环境噪音再加以一定的计算实现随机数序列。随机性更强。