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

本实验首先介绍寄存器的基本概念，复习寄存器的原理。然后学习常用的移位寄存器的设计，并实现在移位指令中需要用到的桶形移位器。

6.1 寄存器

D 触发器可以用于存储比特信号，给 D 触发器加上置数功能就变成了一位寄存器，如图 6-1所示。由图中可以看出，如果 load 信号为 1，则输入信号 in 被送入或门中，或门的另一个输入端为 0，此时 D=in，所以在下一个时钟里 q=in。当 load 值为 0 时，q 值被反馈到或门中，或门的另一个输入值为 0，此时 D=q，因此在下一个时钟周期里 q 值保持先前的值不变。

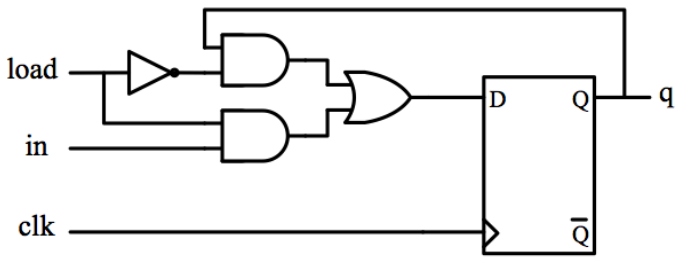


图 6-1: 1 位寄存器

用 Verilog 语言设计寄存器也很简单，如表 6-1所示。

表 6-1的程序的仿真图如图 6-2所示。

本例实现的是一个带有清 0 端和输入端的 1 位寄存器，还有的寄存器带有置位（置 1）端的，图 6-3是同时带有清 0 端、输入端和置位端的寄存器的逻辑示意图，读者可自行设计此寄存器。

表 6-1: 1 位寄存器代码

```
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
```

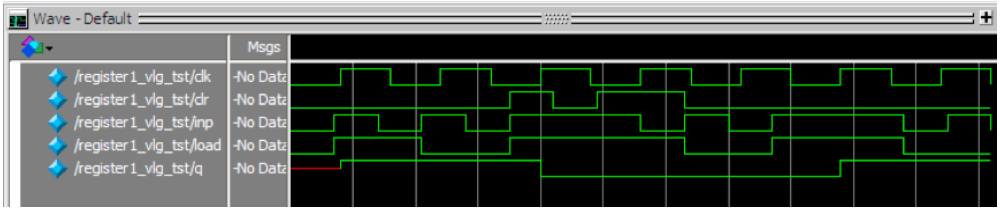


图 6-2: 1 位寄存器仿真结果

将 2 个或者 2 个以上的 1 位寄存器组合在一起，这些寄存器共用一个时钟信号，这就构成了多位寄存器，寄存器常被用在计算机中存储数据，如指令寄存器、数据寄存器等。表 6-2是利用 Verilog 语言设计寄存器的例子。

表 6-2的程序的仿真图如图 6-4所示。

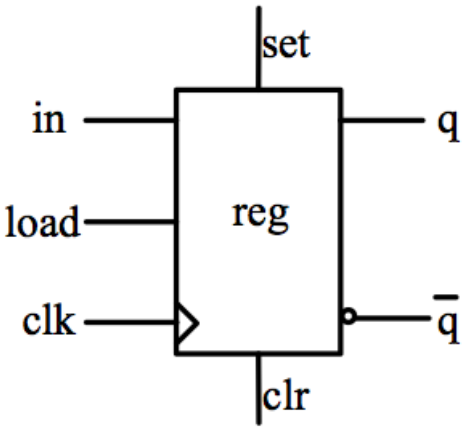


图 6-3: 1 位寄存器框图

表 6-2: 4 位寄存器代码

```
module register4(load,clk,clr,d,q);
  input  load,clr,clk;
  input  [3:0] d;
  output reg [3:0] q;

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

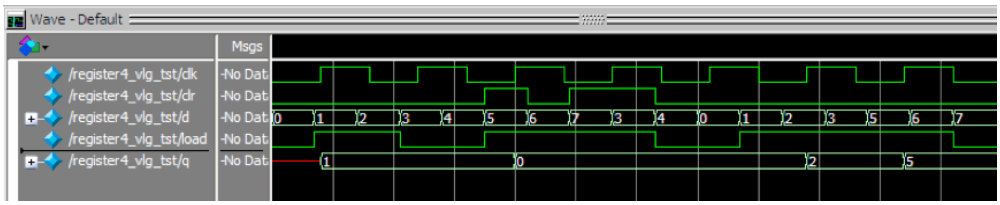


图 6-4: 4 位寄存器仿真结果

6.2 移位寄存器

移位寄存器在时钟的触发沿，根据其控制信号，将存储在其中的数据向某个方向移动一位。移位寄存器也是数字系统的常用器件。

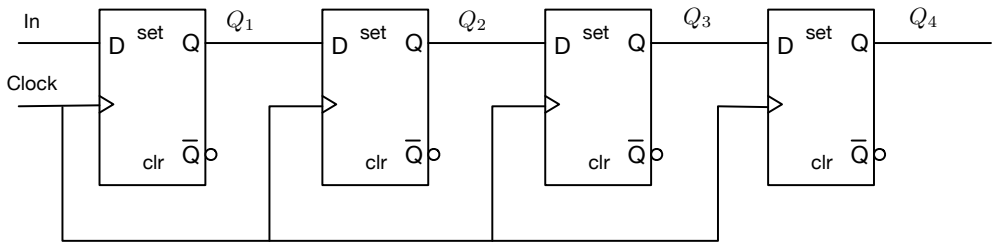
图 6-5(a)中是一个由 4 个 D 触发器构成的简单向右移位寄存器，数据从移位寄存器的左端输入，每个触发器的内容在时钟的正跳变沿（上升沿）将数据传到下一个触发器。图 6-5(b)是一个此移位寄存器的序列传递实例。

6.2.1 算术移位和逻辑移位寄存器

这里的算术移位是指考虑到符号位的移位，算术移位要保证符号位不改变，算术左移同逻辑左移一样，算术右移最左面的空位补符号位。逻辑移位不管是向左移位还是向右移位都是空缺处补 0。循环是将移出去的那一位补充到空出的最高/低位的移位方式。置数是将一个 8 位的数据输入到寄存器中，即给寄存器赋一个初始值。

用 Verilog HDL 语言很容易描述出移位寄存器，如：

```
Q <= {Q[0],Q[7: 1]}; //循环右移
Q <= {Q[7],Q[7: 1]}; //算术右移
```



(a) 移位寄存器框图

	In	Q1	Q2	Q3	Q4=Out
t0	1	0	0	0	0
t1	0	1	0	0	0
t2	1	0	1	0	0
t3	1	1	0	1	0
t4	1	1	1	0	1
t5	0	1	1	1	0
t6	0	0	1	1	1
t7	0	0	0	1	1

(b) 移位实例

图 6-5: 移位寄存器

表6-3描述了常见的移位寄存器工作方式。其中左端串行输入 1 位数值，并行输出 8 位数值是指每个时钟到来时右移一位，并且移入的最左位由外部开关决定是 1 还是 0，输出同其他情况一样为同时输出 8 位。这个功能在进行串行转换为并行时比较有用，可以将时间上顺序输入的 8 个 bit 存入移位寄存器，在 8 个周期后形成一个 8bit 数一起输出。后续键盘串行输入可以利用这个功能。请自行思考这些功能如何用 Verilog 语言实现。

6.3 桶形移位器

在 CPU 中，我们往往需要对数据进行移位操作。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。桶形移位器采用组合逻辑的方式来实现同时移动多位，在效率上优势极大。因此，桶形移位器常常被用在 ALU 中来实现移位。图6-6为 8 位桶形移位器的输

表 6-3: 移位寄存器的工作方式

控制位	工作方式
0 0 0	清 0
0 0 1	置数
0 1 0	逻辑右移
0 1 1	逻辑左移
1 0 0	算术右移
1 0 1	左端串行输入 1 位值，并行输出 8 位值
1 1 0	循环右移
1 1 1	循环左移

入输出引脚图。其中输入数据 din 和输出数据 dout 均为 8 位，移位位数 shamt 为 3 位。选择端 L/R 表示左移和右移，置为 1 为左移，置为 0 为右移。选择端 A/L 为算术逻辑选择，置为 1 为算术移位，置为 0 为逻辑移位。

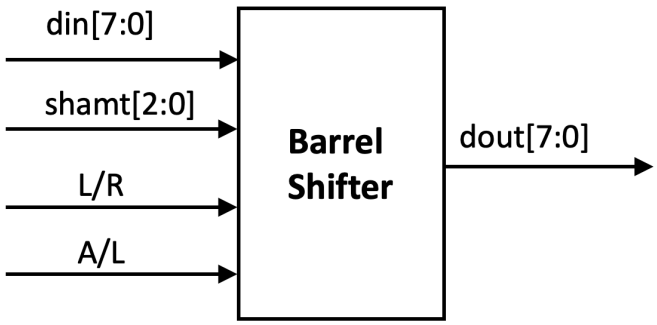


图 6-6: 桶形移位器器件引脚图

图 6-7显示了桶形移位器的具体实现。该实现使用了大量的四选一选择器来分三阶段实现 0 至 7 位的任意左移或右移。第一级利用 shamt[0] 来控制是否需要移动一位，第二级在第一级的移动结果上用 shamt[1] 来控制是否要移动两位，第三级在第二级的基础上再对应判断是否要移动四位。每个四选一选择器有两位控制端，控制端低位 S0 为当前级是否需要移动，为 0 时，对应选择输入的 0 或 2，均不会做任何移动。四选一的高位 S1 由 L/R 输入控制，当需要移动

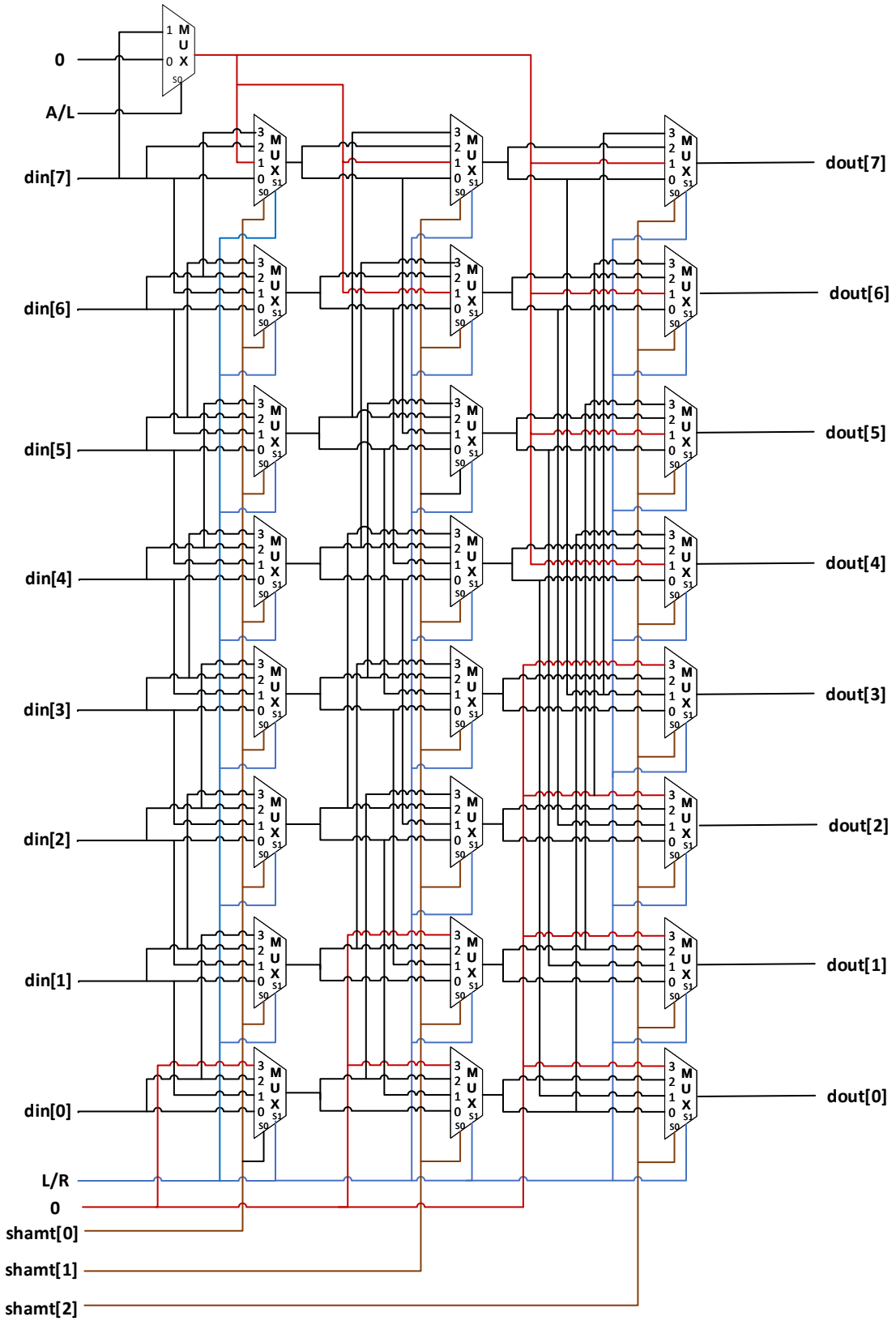


图 6-7: 桶形移位器电路结构图

S0=1 时，左移时选中 11=3 号输入端，右移时选中 01=1 号输入端。这两个输入端分别连接了数据低位或高位的上一级输出。对于算术和逻辑右移的选择，是通过控制高位移入的是 0 还是 din[7] 来决定的。注意，这个电路是纯组合逻辑，所以可以在输入改变时无需时钟信号就直接改变输出的移位结果。

思考：在 RV32I 指令集中需要实现 32 位数据的移位，应该如何用 Verilog 语言实现?

6.4 实验验收内容

6.4.1 上板验证

利用移位寄存器实现随机数发生器 我们可以利用 8 位移位寄存器来实现一个简单的随机数发生器。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。

具体实现时，可以用一个 8 位右移移位寄存器，从左到右的比特以 $x_7x_6x_5x_4x_3x_2x_1x_0$ 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[①]：

$$x_8 = x_4 \oplus x_3 \oplus x_2 \oplus x_0 \tag{6-1}$$

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。

请实现一个 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 DE10-Standard 开发板上观察生成的随机数序列。

☞ **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?

[①]此类循环周期为 $2^n - 1$ 的随机序列的生成公式可以在教科书和网络上找到。

6.4.2 在线测试

必做 移位寄存器实现

必做 桶形移位器

必做 线性反馈移位寄存器