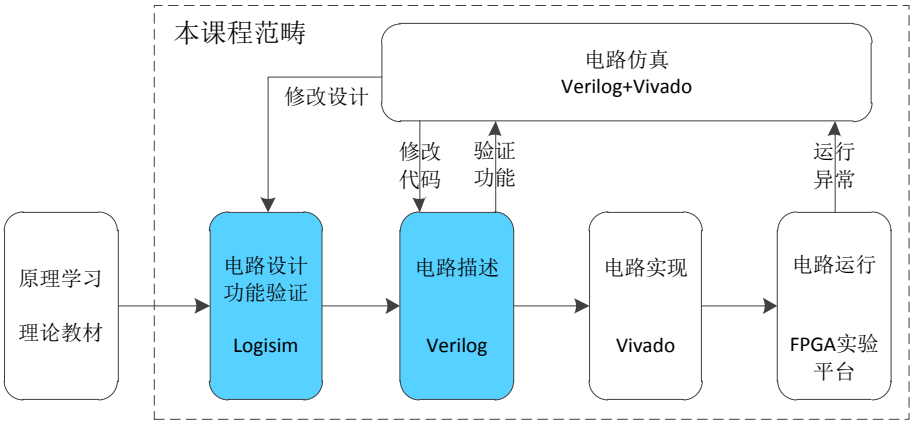
实验 03_简单时序逻辑电路

简介

下图以设计数字电路的一般流程为例，说明了电路设计过程中的关键步骤以及相关工具。



通过前面的实验，用户应该能够达到熟练使用 Logisim 工具以及阅读、编写简单组合逻辑 Verilog HDL 代码的程度。本次实验我们将使用 Logisim 设计简单时序电路，并学习用 Verilog 语言描述简单时序逻辑电路。

实验目的

掌握时序逻辑相关器件的原理及底层结构

能够用基本逻辑门搭建各类时序逻辑器件

能够使用 Verilog HDL 设计简单逻辑电路

实验环境

PC 一台

Windows 或 Linux 操作系统

Java 运行环境（jre）

Logisim 仿真工具

vlab.ustc.edu.cn（jre、Logisim 工具以及 Verilog 语法介绍都可在此网站获取）

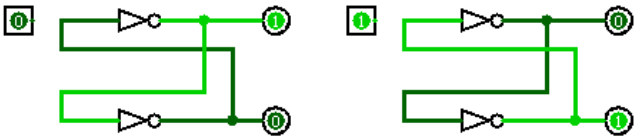
实验步骤

前面的实验中，我们在 Logisim 中使用 MOS 管搭建出了与、或、非、与非、或非、同或、异或等两输入的基本逻辑门，又使用这些基本逻辑门搭建出了加法器、选择器、译码器等各种典型的组合逻辑电路。不难发现，所有的组合逻辑电路都可以通过与、或、非三种基本门搭建出来（也可以是其它组合，如只使用与非门便可以搭出所有的组合逻辑电路），这些组合逻辑电路的共同特点就是没有记忆功能，其输出只与当前的输入信号有关，而不受电路之前状态的影响。与此相对应的便是时序逻辑电路。

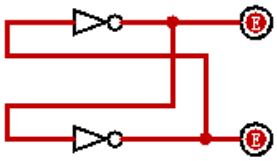
时序逻辑电路的输出受到电路当前输入和之前状态两种因素的影响。当输入信号完全一样时，时序逻辑电路的输出及状态可能不一样，因此说，时序逻辑电路具有记忆功能，能够记住电路之前的状态。下面我们将使用与或非三种基本门逐步搭建出各种时序逻辑电路的关键器件，在此过程中加深对时序逻辑器件结构和工作原理的理解。

Step1：搭建双稳态电路

双稳态电路是由两个非门交叉耦合构成，如下图所示，完全一样的电路结构，却可以具备两种完全不同的状态，这一点与组合逻辑电路存在本质的区别。双稳态电路是一种最简单的时序逻辑电路，没有输入信号，状态一旦确定之后也无法改变，没有实际使用价值，但确是所有时序逻辑电路的基础。

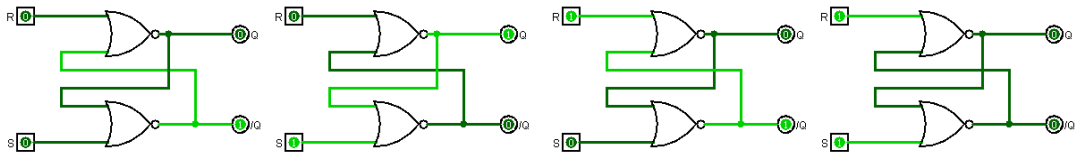


注意，在 Logisim 中搭建此电路时，应先将两条交叉耦合线断开一条，等输入信号将其状态初始到确定状态后再将耦合线连上。否则电路将处于一种不确定状态。



Step2：搭建 SR 锁存器

双稳态电路没有输入信号，所以无法进行操作，我们对其进行修改，将两个非门用或非门代替。两个输入信号分别命名为 S 和 R，输出信号命名为 Q 和/Q，其中/Q 是 Q 取反的意思，S 信号负责对 Q 置位（Set），R 信号负责对 Q 信号置位（Reset）。当 SR 信号都无效（为 0）时，电路将保持之前的状态，即处于锁存状态，因此这种电路称为 SR 锁存器。SR 信号都有效（为 1）时，Q 和/Q 信号都为零，虽然也是一种确定状态，但不符合/Q 为 Q 取反的定义，因此我们将其看成是一种未定义状态，在实际使用过程中应避免这种状态的出现。

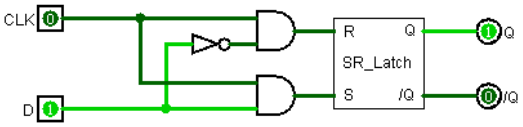


用户可在 Logisim 中尝试改变 S、R 端的数据输入，观察电路状态的改变，以加深对 SR 锁存器工作原理和行为特性的理解。

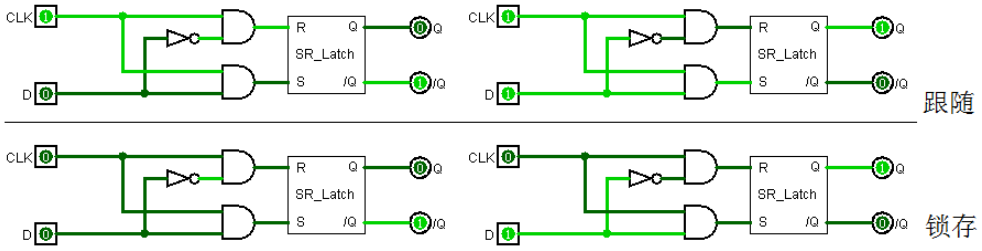
Step3：搭建 D 锁存器

前面提到，SR 锁存器两个输入都为 1 是一种未定义状态，我们不

希望这种状态出现，为此我们在 SR 锁存器前面添加两个与门和一个非门，如下图所示，便构成了 D 锁存器。

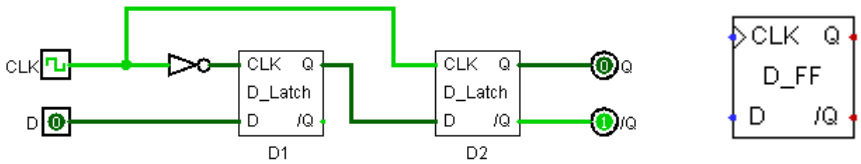


分析 D 锁存器电路可以发现，当 CLK 信号为高电平时，Q 信号将随着 D 端输入信号的变化而变化，称之为“跟随”状态。当 CLK 信号为低电平时，Q 信号将保持之前的值，不会收到 D 信号变化的影响，称之为“锁存”状态。D 锁存器是一种电平敏感的时序逻辑器件。

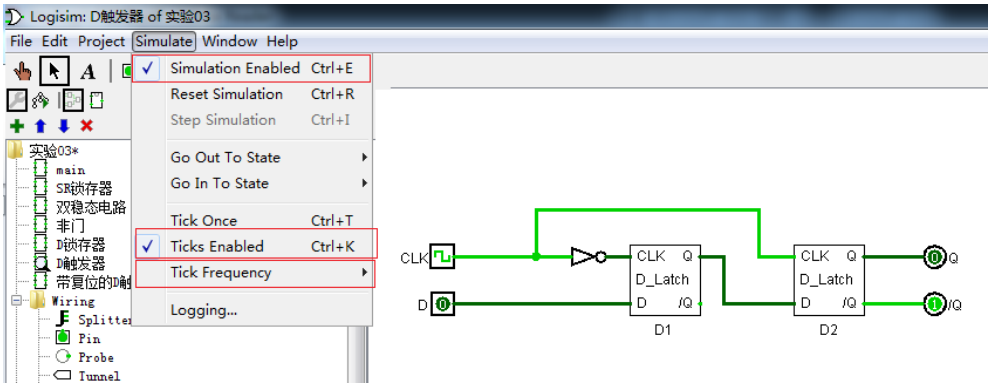


Step4：搭建 D 触发器

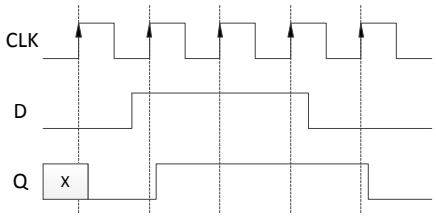
通过对 D 锁存器的行为特性分析，我们可以发现，D 锁存器在信号的传输过程中起到了类似于开关的作用，当开关（CLK 信号）打开的时候，信号能够传输过去，当开关（CLK 信号）关闭时信号无法通过。如果我们将两个 D 锁存器串起来，其控制信号有效值始终相反，会是什么样的情况呢？实际上这就构成了 D 触发器，如下图所示，CLK 信号为低电平时，D 信号通过了 D1，当 CLK 信号由低电平变为高电平时，D1 关闭，D2 打开，信号到达 Q 端。



用户可在 Logisim 菜单栏中点击“simulation”选项，首先将“Tick Frequency”设置为“1Hz”，然后使能仿真和触发功能，在“CLK”信号以 1Hz 频率跳变过程中，改变 D 信号的输入值，观察 Q 信号的输出。



通过分析我们可以发现，只有在 CLK 信号由低电平变为高电平的瞬间，D 信号才会传播到 Q 端，其余时刻 Q 端的值都保持不变。将 D 触发器作为一个整体观察，请行为特性如下波形图所示。



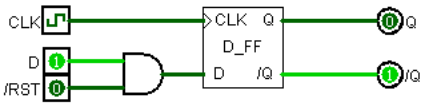
其 Verilog 代码如下所示：

```
module d_ff(
input clk,d,
output reg q);
always@(posedge clk)
    q <= d;
endmodule
```

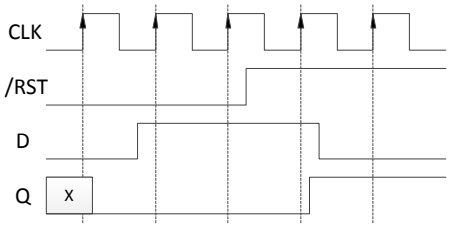
说明：reg、always 和 posedge 是 Verilog 中的关键字，其中 always 表示其后是个过程语句块。reg 与前面学习到的 wire 关键字类似，是一种数据类型，称为寄存器类型。对于初学者，可以简单的理解为凡是在 always 语句块内被赋值的信号，都应定义为 reg 类型。posedge 为事件控制关键字，例如代码中的“posedge clk”表示“clk 信号的上升沿”这一事件。

我们还可以为触发器添加复位信号，如下图所示，可以看出，当

复位信号有效（低电平有效）时，输出信号 Q 始终为零。



其波形图如下图所示。



这种触发器的复位信号只有在时钟信号的上升沿才起作用，在非上升沿时刻，复位信号不起作用。这种复位方式称为同步复位。其 Verilog 代码如下所示：

```
module d_ff_r(
input clk,rst_n,d,
output reg q);
always@(posedge clk)
begin
    if(rst_n==0)
        q <= 1'b0;
    else
        q <= d;
end
endmodule
```

说明：这段代码中又新出现了 begin、end、if、else 四个关键字，其中 begin/end 必须成对出现，用于表征语句块的作用区间，如上述例子中，begin/end 之间的代码都属于同一 always 块。if、else 用于条件判断，在很多其它语言中都有出现，其含义也都一样，此处不再赘述。“1'b0”是一种数据表示方式，一般格式为“数据位宽'进制数值”，本例中表示这是一个 1bit 的数据，用二进制表示，其值为 0。

与此同步复位相对应的，还有一种异步复位方式，即不论时钟和 D 信号如何，一旦复位信号有效，输出端 Q 立即变为确定的复位值（一般为低电平），读者可考虑一下这种触发器电路结构。其 Verilog 代码为：

```
module d_ff_r(
input clk,rst_n,d,
output reg q);
always@(posedge clk or negedge rst_n)
begin
    if(rst_n==0)
        q <= 1'b0;
    else
        q <= d;
end
endmodule
```

说明：negedge 是与 posedge 同类型的一个关键字，只不过它表示信号的下降沿事件。关键字“or”表示“或”操作

可以看出，异步复位与同步复位最大的区别在于，复位信号与时钟信号同时出现在了 always 语句的敏感变量列表中，在没有时钟上升沿的情况下，复位信号也能够起作用。因为复位操作不再完全与时钟信号的上升沿同步，因此称为异步复位。

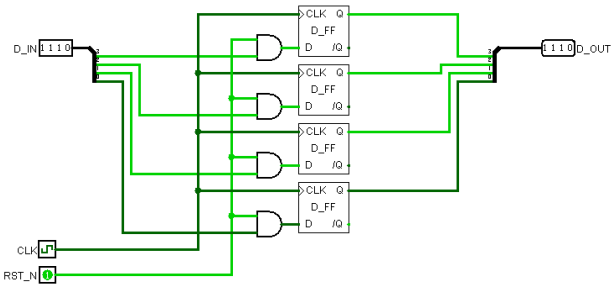
D 触发器与 D 锁存器的最大不同在于它是边沿敏感的器件，电路输出状态只在时钟信号的边沿（一般来说是上升沿）发生（异步复位除外），时钟成为整个电路的同步信号，因此由 D 触发器为核心构成的电路一般称为同步时序逻辑电路，而锁存器构成的一般都是异步时序逻辑电路。

同步时序逻辑电路在电路设计中非常重要，绝大部分的电路都是同步时序逻辑电路，而 D 触发器又是同步时序逻辑电路的核心器件，D 触发器的重要性不言而喻。

Step5：搭建寄存器

寄存器本质上来说就是 D 触发器，如下图所示，我们用 4 个 D 触发器构成了一个能够存储 4bit 数据的寄存器，带有低电平有效的同步复位信号。请在 Logisim 中使用仿真功能对其进行仿真，观察行为

特性。



其 Verilog 代码为：

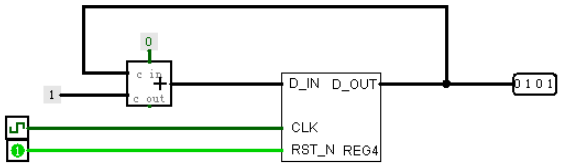
```
module REG4(
input CLK,RST_N,
input [3:0] D_IN,
output reg [3:0] q);
always@(posedge CLK)
begin
    if(RST_N==0)
        D_OUT <= 4'b0;
    else
        D_OUT <= D_IN;
end
endmodule
```

说明：对于多 bit 位宽的信号，在 Verilog 中使用“[x:y]”这种方式声明，例如上述代码中，D_OUT 就是一个 4bit 的信号，它包含了 D_OUT[0]、D_OUT[1]、D_OUT[2]、D_OUT[3]四个单 bit 信号。

细心的读者也许已经发现了一个问题，上述电路在复位时其输出信号只能是 0，这是不符合实际需求的，有些情况下信号的复位值需要为 1，而且 Verilog 语法也支持信号复位值不为 0。如上述代码复位部分可以改成“if(RST_N==0) D_OUT <= 4'b0011;”其电路结构应如何实现呢？请读者思考一下。

Step6：搭建简单时序逻辑电路

我们利用 4bit 寄存器，搭建一个 4bit 的计数器，该计数器在 0~15 之间循环计数，复位时输出值为 0，电路图如下所示



其 Verilog 代码为：

```
module REG4(
input CLK,RST_N,
output reg [3:0] CNT);
always@(posedge CLK)
begin
    if(RST_N==0)
        CNT <= 4'b0;
    else
        CNT <= CNT + 4'b1;
end
endmodule
```

实验练习

题目 1. 在 Logisim 中用与非门搭建 SR 锁存器，画出电路图，并分析其行为特性，列出电路在不同输入时的状态。

题目 2. 在 Logisim 中搭建一个支持同步置位功能的 D 触发器，画出其电路图，并编写对应的 Verilog 代码。

题目 3. 在 Logisim 中搭建一个带有异步复位功能的 D 触发器，画出其完整电路图，并进一步调用该触发器设计一个从 0~15 循环计数的 4bit 计数器（可使用 Logisim 中的加法器模块，也可自行设计计数器），写出计数器的 Verilog 代码。

题目 4. 在 Logisim 中搭建一个 9~0 循环递减的计数器，复位值为 9，每个周期减一（可使用 Logisim 中的减法器模块，也可自行设计计数器），画出电路图，进行正确性测试，并写出其对应的 Verilog 代码。

题目 5.前面所有电路的复位信号都是低电平有效，如要使复位信号高电平有效，应如何实现？试用 Logisim 画出一个示例电路，并编写 Verilog 代码。

总结与思考

1.请总结本次实验的收获

2.请评价本次实验的难易程度

3.请评价本次实验的任务量

4.请为本次实验提供改进建议