用 Verilog HDL 语言设计数字逻辑电路

本文以数字逻辑电路实验教材的第 28 页的电路为例，说明如何利用 Verilog HDL 语言设计简单的数字逻辑电路。读者看完本文即可完成实验九的课程内容。若需要进一步深入了解 Verilog HDL 语言，请参考另外的讲义。

待设计的电路如图 1 所示，包括一组反馈移位寄存器、形成触发器时钟信号的施密特电路和一个防抖动的闩锁。在设计电路时，应先分析电路的功能，确定输入输出管脚的数目。

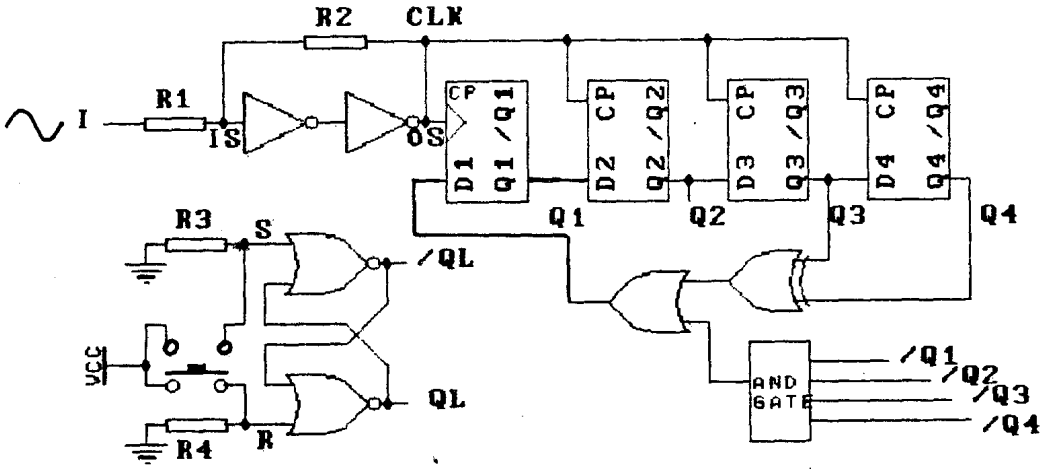

图 1 设计电路举例

按照图 1，该电路需要 4 个 D 触发器组成移位寄存器，并需要一个时钟（CLK）输入端；组成闩锁需要一个组合信号输出端（Q_L）和两个输入端（S、R）；组成施密特电路需要一个组合信号输出端 OS 和一个输入端 IS。

电路的方程是设计文件的一个重要的组成部分，它以引脚信号作为输入信号和输出信号，描述设计者对器件内部电路的要求。在器件内，施密特电路只是同门，它的方程为

$$OS = IS \qquad (1)$$

对应的 Verilog HDL 语句是： assign OS = IS;

参照图 1 锁存器的方程为

$$Q_L = \overline{R + \overline{Q_L}} = \overline{R + \overline{S + Q_L}} \qquad (2)$$

对应的 Verilog HDL 语句：assign QL = ~(R|(~(S|QL)));

在（1）式和（2）式中，等号左边是输出的新状态，等号右边是输入和反馈的原状态。在设计文件中，用同一符号表示原状态和新状态。这种写法实际上是视器件为无延时理想器件，等号两边的信号在任一时刻都是相等的，或原状态和新状态在任一时刻都是相同的。

在 Verilog HDL 语言中，常用“assign”持续赋值语句产生组合逻辑电路，如（1）式和（2）式。相应的位运算符有：

~ //按位取反

| //按位或

^ 按位异或

& //按位与

^~ //按位同或（异或非）

触发器则不同，它的 D 端的输入信号仅在时钟上升沿或下降沿有效时才呈现为 Q 端的输出信号。在设计文件中用符号“<=”表示这种关系。

按照图 1，移位寄存器的方程为

$$Q_2 <= Q_1$$

$$Q_3 <= Q_2$$

$$Q_4 <= Q_3$$

$$Q_1 <= (Q_3 \oplus Q_4) + \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3} \cdot \overline{Q_4} \quad (3)$$

对应的 Verilog HDL 语句如下：

```
always@(posedge CLK)
begin
  Q4<=Q3;
  Q3<=Q2;
  Q2<=Q1;
  Q1<=(Q3^Q4)|((~Q1)&(~Q2)&(~Q3)&(~Q4));
end
```

在 Verilog HDL 语言中，常用“always”过程块赋值语句产生时序逻辑电路。在 always 语句中，推荐用” <=” 符号产生 D 触发器。

always 语句是不断重复执行的。posedge 表示上升沿，negedge 表示下降沿。posedge CLK 表示 CLK 信号的上升沿有效，即每当 CLK 信号的上升沿出现时则更新寄存器 Q1、Q2、Q3 和 Q4。

图 1 的完整电路描述文件如下：

```
module Demo(IS,OS,CLK,Q1,Q2,Q3,Q4,R,S,QL);
   input IS,CLK,R,S;                  //输入管脚定义
   output OS,Q1,Q2,Q3,Q4,QL;   //输出管脚定义

   reg Q1,Q2,Q3,Q4;                   //寄存器定义

   always@(posedge CLK)         //反馈移位寄存器
   begin
     Q4 <= Q3;
     Q3 <= Q2;
     Q2 <= Q1;
     Q1 <= (Q3^Q4)|((~Q1)&(~Q2)&(~Q3)&(~Q4));
   end

   assign  QL = ~(R|(~(S|QL)));   //闩锁
   assign  OS = IS;                   //时钟整形

endmodule
```

说明：在 Verilog HDL 模块中过程块（如：always 块）、连续赋值语句（如：assign 语句）都是并行的；在同一个模块中它们的先后顺序没有关系。

Verilog HDL 程序的基本设计单元是“模块”。一个 module 就是一个模块。每个模块的内容都是位于 module 和 endmodule 两个语句之间。除了 endmodule 语句外，每个语句和数据定义的最后必须有分号。可以用/*……*/或//……对程序作注释。

每个模块要进行端口定义，并说明是输入口（input）还是输出口（output），然后对模块的功能进行描述。

端口信号类型缺省为 wire 型。wire 型数据用来表示以 assign 关键字指定的组合逻辑信号。wire 型信号可以用做任何方程式的输入，也可以用做“assign”语句的输出。reg 是寄存器数据类型的关键字，常代表触发器。在“always”块内被赋值的每一个信号都必须定义成 reg 型。若端口信号将在 always 块中被赋值，例如，本电路的输出端口 Q1，在 always 块中被赋值了，因此，必须重新定义为 reg 型，即 reg Q1。

以上完成了图 1 所示电路的 Verilog HDL 语言设计。接下来运行 Lattice 公司的 ispLEVER 软件进行编译和仿真。仿真通过后，最后一步是生成 JEDEC 文件，然后就可以下载到 GAL20V8 芯片了。关于 ispLEVER 软件的使用，参考相关文档和教学视频。