第八周

笔记本： 电子电路设计训练

更新时间： 2020/4/19 星期日 3:38

作者： Hantao Li

第三讲、Verilog HDL高级语法

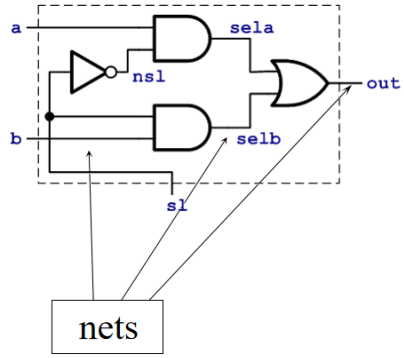
一、主要“数据类型”

Verilog 共定义19种数据类型

三种主要的数据类型：

1. **Net** 连接 —— 其中最常见线网wire，表示器件之间的物理连接，称为（线网）连接类型
2. **Register** 寄存器 —— 表示抽象的储存单元，称为寄存器/变量类型
3. **Parameter** —— 表示运行时的常数，称为参数类型

连接（Net）类型：在为不同工艺的基本元件建立库模型的时候，常常需要用不同的连接类型来与之对应，使其行为与实际器件一致。如果不明确地说明连接是何种类型，应该是指wire类型。



由模块或门驱动的连线；驱动端信号的改变会立刻传递到输出的连线上；上图中，selb信号的改变，会自动地立刻影响或门的输出。
wire型变量：

1. 最常用的net型变量，常用来表示以assign关键字指定的组合逻辑信号
2. 模块中的输入/输出信号类型缺省为wire型
3. 可用做任何方程式的输入，也可以用做“assign”语句或实例元件的输出

寄存器（Register）类型：寄存器型是数据存储单元的抽象，但不能表示真正的硬件，相当于高级语言中的变量。
reg型变量：在过程块中被赋值的信号，往往代表触发器，但不一定就是触发器（也可以是组合逻辑信号）
register型变量与net型变量的根本区别：

1. register型变量需要被明确地赋值，并且在被重新赋值前一直保持原值
2. register型变量必须通过过程赋值语句赋值！不能通过assign语句赋值
3. 在过程块内被赋值的每个信号必须定义成reg型

参数（parameter）类型：用parameter来定义一个标识符，代表一个常量——称为符号常量。

1. 参数型常量经常用于定义延迟时间和变量位宽

2. 在模块或实例引用时，可通过参数传递改变在被引用模块或实例中已定义的参数
3. 可用字符串表示的任何地方，都可以用定义的参数来代替。 参数是本地的，其定义只在本模块内有效

标量与矢量：位宽是1bit的线网和寄存器类型数据被定义为标量，而位宽大于1bit的线网和寄存器类型数据被定义为矢量。

二、赋值语句

赋值语句分为两类：

1. 连续赋值语句——assign语句，用于对wire型变量赋值，是描述组合逻辑最常用的方法之一。 assign c=a&b; //a、b、c均为wire型变量
2. 过程赋值语句——用于对reg型变量赋值，有两种方式： 非阻塞（non-blocking)赋值方式： 赋值符号为<=，如 b <= a ； 阻塞（blocking)赋值方式： 赋值符号为=，如 b = a ；

非阻塞赋值与阻塞赋值方式的主要区别：
非阻塞（non-blocking）赋值方式 (b<= a)：

1. b的值被赋成新值a的操作, 并不是立刻完成的，而是在块结束时才完成；
2. 块内的多条赋值语句在块结束时同时赋值；
3. 可参考对应的同步数字电路。

阻塞（blocking）赋值方式 (b = a)：

1. b的值立刻被赋成新值a；
2. 完成该赋值语句后才能执行下一句的操作；
3. 可能会由于疏忽，使综合结果未知。（可用但慎用）

三、运算符/块语句/条件语句/循环语句/always块

第四讲、简单数字电路设计

一、门级电路

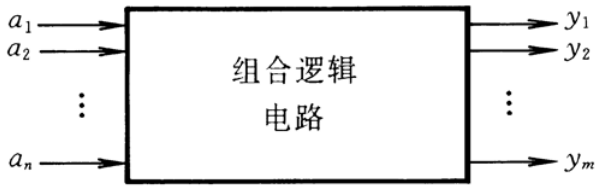
逻辑电路通常由许多逻辑门和开关所组成，因此用逻辑门的模型来描述逻辑电路结构是最直观的。Verilog的基本元件模型共有26种，其中14种为基本门级元件，12种为开关级元件。

门级电路列表：<实例元件名>(<数据输出>, <数据输入>, <控制输入>)
门级电路调用（实例化）：<门的类型> [<驱动能力><延时>]<门实例1>[，<门实例2>，<门实例3>……];

二、RTL电路

RTL级电路例子：对于数据选择器，可以采用逻辑门电路的方法进行描述，但更加有效的是采用数据流建模的方法。

三、常用组合逻辑电路



$$y_1 = f_1(a_1, a_2, \ldots, a_n)$$
$$y_2 = f_2(a_1, a_2, \ldots, a_n)$$
$$\ldots$$
$$y_m = f_m(a_1, a_2, \ldots, a_n)$$

组合逻辑电路：功能上无记忆，结构上无反馈。

1. 电路任一时刻的输出状态只取决于该时刻各输入状态的组合，而与电路的原状态无关。
2. 对于任何一个多输入多输出组合逻辑电路，可以用框图或者一组逻辑函数来表示。

加法器、乘法器、多路器、编码器、译码器

四、常用时序逻辑电路

输出不只是当前输入的逻辑电平的函数，还与目前电路所处的状态有关。时序逻辑通常是由多个触发器和多个组合逻辑块组成的网络，时序逻辑电路是设计复杂数字电路的核心。

RS触发器、同步RS触发器、D触发器

五、任务与函数

task 和 function 说明语句分别用来定义任务和函数，利用任务和函数可以把一个很大的程序模块分解成许多较小的任务和函数，便于理解和调试。学会使用task和function语句可以简化程序的结构，使程序明白易懂，是编写较大型模块的基本功。

第五讲、复杂数字电路设计

一、计数器设计

D触发器构成的计数器、计数器verilog HDL实现、计数器verilog实现

二、寄存器与数据流动

寄存器变量：

1. 在“同步块”中使用 reg 类型变量
2. 在“组合块”中使用 reg 类型变量

同步寄存器、组合逻辑寄存器、锁存器、寄存器、开关逻辑

三、流水线设计

1. 把规模较大、层次较多的组合逻辑电路分成几个级，在每一级插入寄存器并暂存中间数据。
2. K级的流水线就是从组合逻辑的输入到输出恰好有K个寄存器组，上一级的输出是下一级的输入而又无反馈的电路。
3. 好处：提高数据处理的吞吐量。

4. 资源：性能的提高是以消耗更多的寄存器资源为代价的。

首次延迟：采用流水线技术获取第一个稳定计算结果需要的时间总量。
吞吐延迟：执行一次重复操作需要的时间总量。

四、阻塞与非阻塞

---

1. 阻塞赋值“=”：赋值时先计算等号右手方向的值，赋值语句不允许任何别的Verilog HDL语句干扰，直到现行赋值操作完成；排序不当的阻塞式赋值操作会出现竞争情况。
2. 非阻塞赋值“<=”：赋值时先计算等号右手方向的值，但过程块退出才一次性地更新左手值，在计算右手值和更新左手值期间，允许其它Verilog HDL语句同时进行表达式计算。非阻塞赋值操作只能用于对寄存器类型变量进行赋值 。

第六讲、有限状态机（Finite State Machine)

一、基本概念

有限状态机（Finite State Machine）是表示实现有限个离散状态及其状态之间的转移等行为动作的数学模型，又称为有限状态自动机或简称状态机。状态机主要有状态和转移两方面功能。

1. 是由寄存器组和组合逻辑构成的时序电路，公共时钟信号。
2. 状态的改变只可能发生在时钟的跳变沿时。
3. 状态是否改变以及如何改变取决于当前状态与输入信号。
4. 状态机可用于产生在时钟跳变沿开关的复杂的控制逻辑，是同步数字逻辑的控制核心。

二、简单的有限状态机设计

---

1. 状态转移图表示
2. RTL级可综合的Verilog HDL模块表示

有限状态机的Verilog HDL描述：

1. 定义模块名和输入输出端口
2. 定义输入、输出变量或reg变量
3. 定义时钟和复位信号
4. 定义状态变量和状态寄存器
5. 用时钟边沿触发的always块表示状态转移过程
6. 在复位信号有效时给状态寄存器赋初始值
7. 描述状态的转换过程
8. 验证状态转移的正确性，必须完整和全面

讨论题：阻塞赋值与非阻塞复制区别

非阻塞：b的值被赋成新值a的操作, 并不是立刻完成的，而是在块结束时才完成； 块内的多条赋值语句在块结束时同时赋值；可参考对应的同步数字电路。
阻塞：b的值立刻被赋成新值a；完成该赋值语句后才能执行下一句的操作；可能会由于疏忽，使综合结果未知。