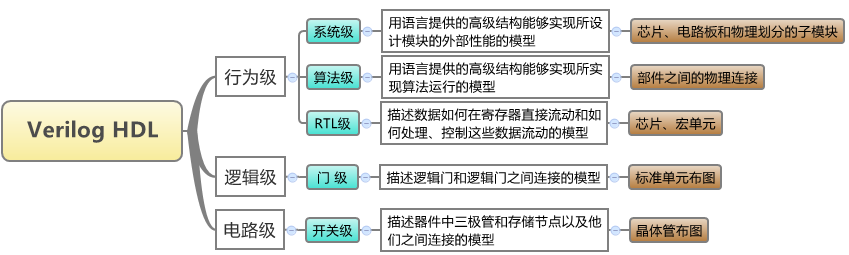
**文章来源：https://blog.csdn.net/msc1211/article/details/107011443/**

**Verilog HDL能干啥？**

Verilog HDL的特点：

* 可描述顺序执行或并行执行的程序结构
* 用延迟表达式或事件表达式来明确地控制过程的启动时间
* 通过命令的事件来触发其他过程的激活行为或停止行为
* 提供了条件/循环等逻辑控制结构
* 提供了可带参数且非零延续时间的任务程序机构
* 提供了用于建立表达式的算术运算符、逻辑运算符和位运算符
* 实现了完整的表示组合逻辑基本元件的原语
* 提供了双向通路和电阻器的描述
* 可建立MOS器件的电荷分享和衰减模型
* 可通过结构性语句精确地建立信号模型

在学习Verilog HDL之前，先明确一下FPGA的设计抽象层次：



**基本程序结构**

1. module module\_name(port\_list)
2. *//声明各种变量、信号*
3. reg  *//寄存器*
4. wire *//线网*
5. parameter *//参数*
6. input  *//输入信号*
7. output *//输出信号*
8. inout  *//输入输出信号*
9. function *//函数*
10. task     *//任务*
11. ....
13. *//程序代码*
14. initial assignment
15. always assignment
16. module assignment
17. gate   assignment
18. UDP    assignment
19. continous assignment
20. endmodule

启示：描述的是模块，其本质是数字电路：

* 组合逻辑电路模块：组合逻辑电路的特点是输入的变化直接反映了输出的变化，其输出的状态仅取决于输入的当前的状态，与输入、输出的原始状态无关。
* 时序逻辑电路模块：时序电路具有记忆功能。时序电路的特点是：输出不仅取决于当时的输入值，而且还与电路过去的状态有关。时序逻辑电路又称时序电路,主要由存储电路和组合逻辑电路两部分组成。

**数据类型及运算符**

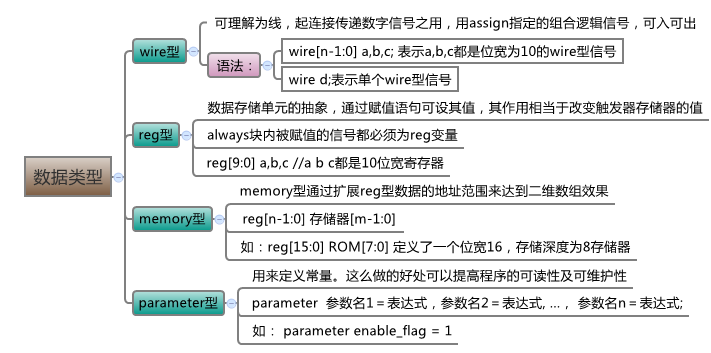
**变量名**

变量名类似C语言，以一组字母、数字、下划线和$符号的组合，且首字符须为字母或者下划线。如

input ctrl\_1;

**数据类型**

将四种基本数据类型整理成一张导图：



其中须注意的是，对于memory型存储单元进行读写，须指定地址，如：

1. reg[15:0] addr; *//定义addr为16位位宽的存储器变量*
2. addr = 1;       *//ok*
4. reg addr[15:0]; *//定义addr为1位位宽的16个存储器变量*
5. addr    = 1;    *//错误*
6. addr[0] = 1;    *//正确*
8. *//又如：*
9. reg[15:0] addr[3:1]; *//定义3个位宽为16位存储器*
10. addr[1] = 16'h0      //16'指定位宽，h 表示16进制，0
11. addr[2] = 16'b011    //b表示二进制

对于parameter变量的实用价值可读性比较好理解，那么可维护性怎么体现呢？

熟悉C语言编程的，联想一下宏，如果宏变了，有宏的地方全替换，这里parameter变量作用类似，如：

1. module Decode(A,F);
2. parameter  Width=1, Polarity=1;
3. ……………
4. endmodule
5. module  Top;
6. wire[3:0] A4;
7. wire[4:0] A5;
8. wire[15:0] F16;
9. wire[31:0] F32;
10. Decode  #(4,0)  D1(A4,F16);
11. Decode  #(5)    D2(A5,F32);
12. Endmodule

**常量**

parameter定义常量，那么对于常数，整型常量即整常数有以下四种进制表示形式:

1. 二进制整数(b或B)
2. 十进制整数(d或D)
3. 十六进制整数(h或H)
4. 八进制整数(o或O)

数字表达方式有以下三种:

1. <位宽><进制><数字>这是一种全面的描述方式。
2. <进制><数字>在这种描述方式中,数字的位宽采用缺省位宽(这由具体的机器系统决定,但至少32位)。
3. <数字>在这种描述方式中,采用缺省进制十进制。

**x和z值**

在数字电路中,x代表不定值,z代表高阻值。不确定是啥？高阻又是啥？记住verilog描述的数字电路，那么对于一个模块的I/O就有可能是高阻，或者状态不确定。

**负数:**

一个数字可以被定义为负数,只需在位宽表达式前加一个减号,减号必须写在数字定义表达式的最前面。注意减号不可以放在位宽和进制之间也不可以放在进制和具体的数之间。

1. -8'd7  //-号直接放在最前面
2. 8'd-7  *//这样则不正确*

**实数**

实数可用十进制方式表述或者科学计数法描述，如：

1. *//十进制表示*
2. 1.0
3. 20.234
4. *//科学计数法表示*
5. 6e-4

**模块端口**

* input:模块从外界读取数据的接口，在模块内**可读不可写**
* output:模块向外部输出数据的接口，模块内部**可写不可读**
* inout：可读写数据，数据双向流动。

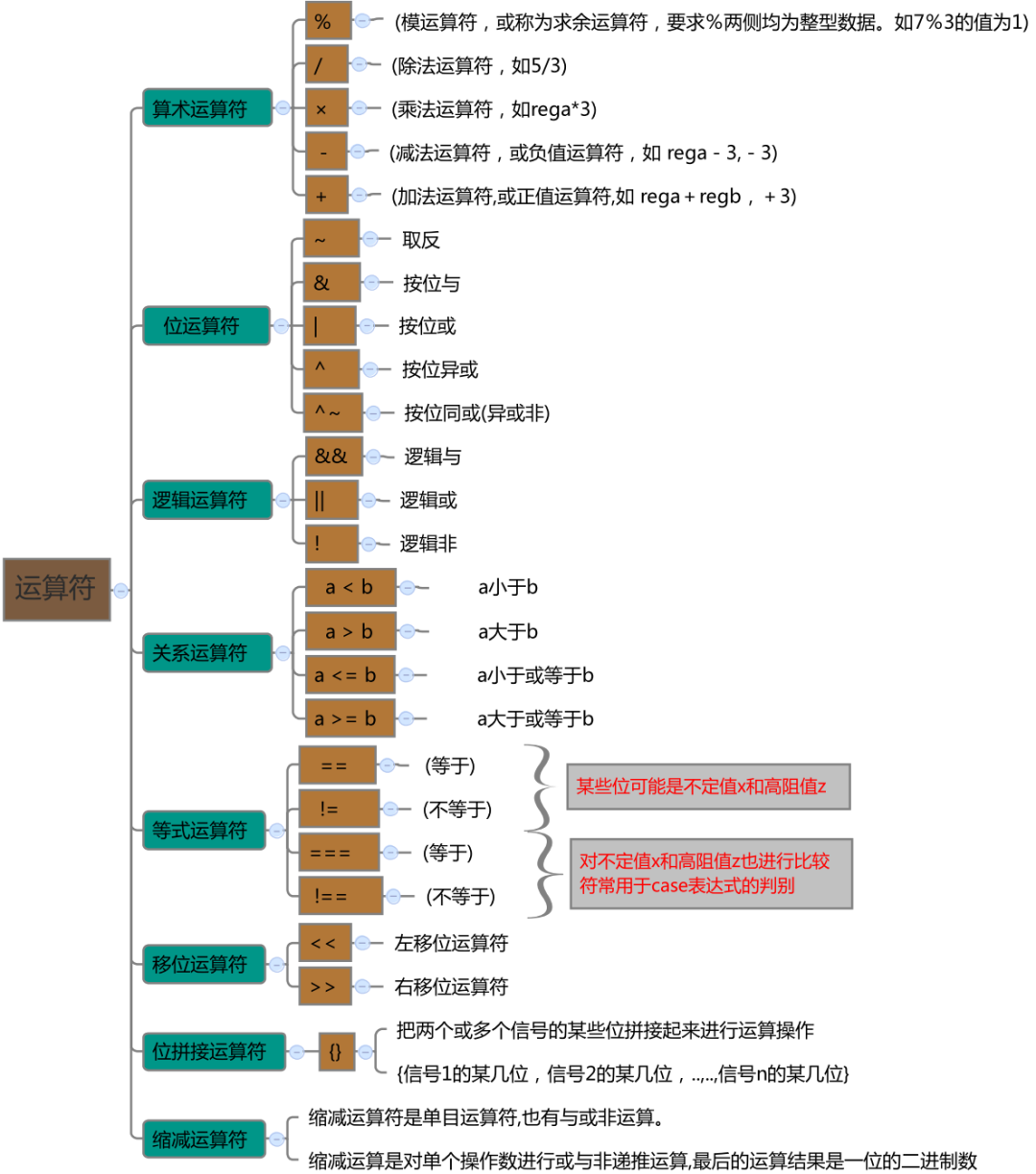
学习硬件描述语言，一定要时刻记住，这是描述的是电路，风格类C，但不是C！

**表达式及运算符**

和C语言类似，运算符也有三种：

* 单目运算符(unary operator):可以带一个操作数,操作数放在运算符的右边。
* 二目运算符(binary operator):可以带二个操作数,操作数放在运算符的两边。
* 三目运算符(ternary operator):可以带三个操作,这三个操作数用三目运算符分隔开。

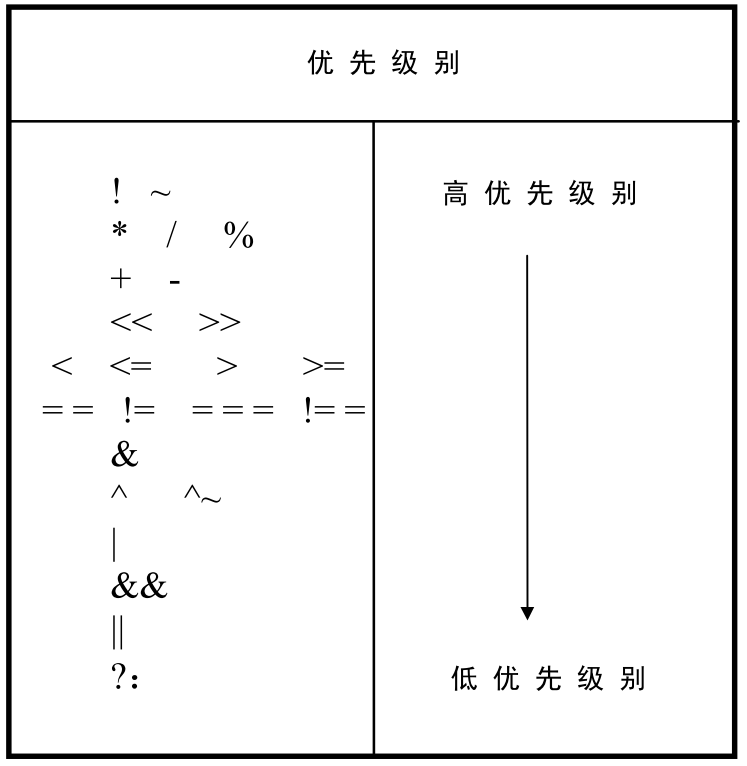
对于运算符，整理了一张导图：



大部分与C语言类似，除了等式运算符、位拼接运算符、缩减运算符，这里放点例子方便理解：

1. *//缩减运算符*
2. reg [3:0] B;
3. reg C;
4. C = &B;
5. *//相当于：*
6. C =( (B[0]&B[1]) & B[2] ) & B[3];
7. *//位拼接运算符*
8. {a,b[3:0],w,3’b101}
9. *//相当于：*
10. {a,b[3],b[2],b[1],b[0],w,1’b1,1’b0,1’b1}

**运算符优先级**：



**赋值语句**

* 非阻塞(Non\_Blocking)赋值方式， 如 b <= a; 加粗是非阻塞的含义
  + **块结束后才完成赋值操作。**
  + **b的值并不是立刻就改变的。**
  + 这是一种比较常用的赋值方法。
* 阻塞(Blocking)赋值方式，如 b = a;
  + **赋值语句执行完后,块才结束。**
  + **b的值在赋值语句执行完后立刻就改变的。**
  + 可能会产生意想不到的结果。

**块语句**

块语句有两种，一种是begin\_end语句，通常用来标识顺序执行的语句，用它来标识的块称为顺序块。一种是 fork\_join语句，通常用来标识并行执行的语句，用它来标识的块称为并行块。

**顺序块**

* 块内的语句是按顺序执行的，即只有上面一条语句执行完后下面的语句才能执行。
* 每条语句的延迟时间是相对于前一条语句的仿真时间而言的。
* 直到最后一条语句执行完，程序流程控制才跳出该语句块。

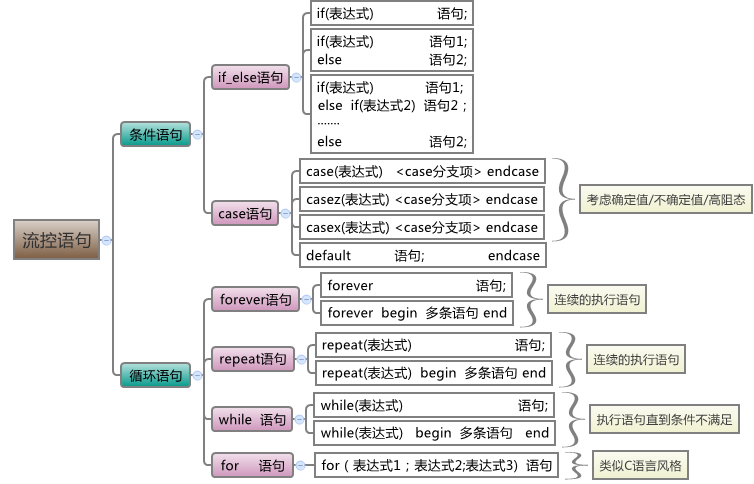
1. begin
2. 语句1;
3. 语句2;
4. ......
5. 语句n;
6. end

**并行块**

* 块内语句是同时执行的，即程序流程控制一进入到该并行块，块内语句则开始同时并行地执行。
* 块内每条语句的延迟时间是相对于程序流程控制进入到块内时的仿真时间的。
* 延迟时间是用来给赋值语句提供执行时序的。
* 当按时间时序排序在最后的语句执行完后或一个disable语句执行时，程序流程控制跳出该程序块。

1. fork
2. 语句1;
3. 语句2;
4. .......
5. 语句n;
6. join

**流控语句**



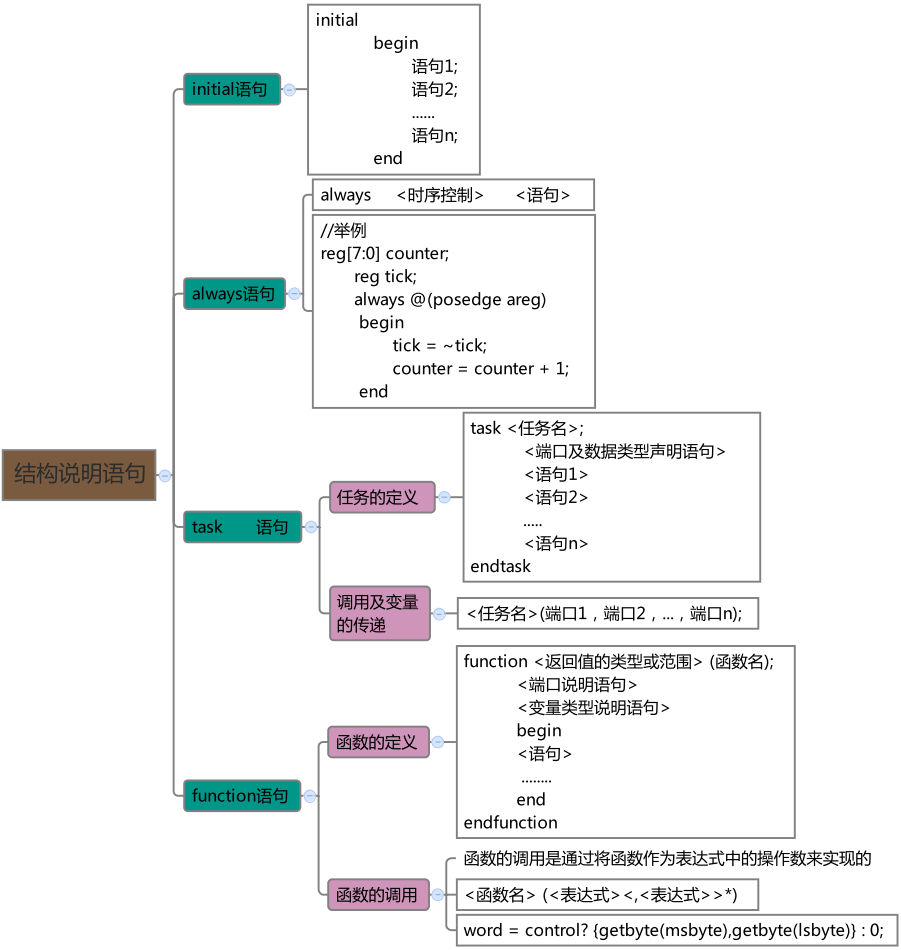
流控语句风格与C语言类似，仅仅需要注意的有下面几点：

* if 语句别忘了考虑else的情况，如忘了处置则最终硬件会最终产生意想不到的后果
* 多条语句在条件内部需要用begin/end对包起来。
* case语句与C语言也有default分支，实际使用注意处置default分支

**结构说明语句**

Verilog语言中的任何过程模块都从属于以下四种结构的说明语句：

* initial说明语句：只执行一次
* always说明语句 ：是不断地重复执行
* task说明语句
* function说明语句



对于task/function的不同点，使用时需要注意：

* **函数只能与主模块共用同一个仿真时间单位，而任务可以定义自己的仿真时间单位。函数的定义不能包含有任何的时间控制语句，即任何用＃、@、或wait来标识的语句。**
* **函数不能启动任务，而任务能启动其它任务和函数。**
* **函数至少要有一个输入变量，而任务可以没有或有多个任何类型的变量。**
* **函数返回一个值，而任务则不返回值。**
* **函数的目的是通过返回一个值来响应输入信号的值。任务却能支持多种目的，能计算多个结果值，这些结果值只能通过被调用的任务的输出或总线端口送出**
* **在函数的定义中必须有一条赋值语句给函数中的一个内部变量赋以函数的结果值，该内部变量具有和函数名相同的名字。**

**系统函数和任务**

在Verilog HDL语言中每个系统函数和任务前面都用一个标识符$来加以确认,有这些系统函数和任务。

rtoi, setup, skew, setuphold, strobe, time, timefoemat, width, write, $recovery,

按字面意思理解，需要用到时查询手册即可。

**编译预处理**

**宏定义 `define**

用法：

`define 标识符(宏名) 字符串(宏内容)

如：

1. *//类似C宏替换*
2. `define signal hello

与C语言宏类似，除了关键字不一样，也支持嵌套。组成宏内容的字符串不能够被以下的语句记号分隔开的，下面几点需要注意：

* 注释行
* 数字
* 字符串
* 确认符
* 关键词
* 双目和三目字符运算符

**“文件包含”处理`include**

用法：`include “文件名”

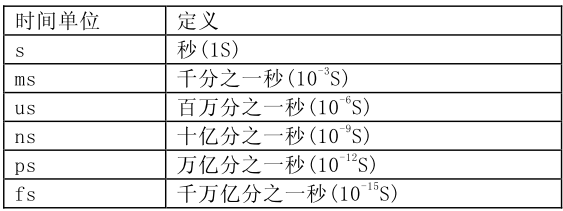
四点说明：

* 一个`include命令只能指定一个被包含的文件，如果要包含n个文件，要用n个`include命令。注意下面的写法是非法的`include"aaa.v""bbb.v"
* `include命令可以出现在Verilog HDL源程序的任何地方，被包含文件名可以是相对路径名，也可以是绝对路径名。例如：'include"parts/count.v"
* 可以将多个`include命令写在一行，在`include命令行，只可以出空格和注释行。
* 如果文件1包含文件2，而文件2要用到文件3的内容，则可以在文件1用两个`include命令分别包含文件2和文件3，而且文件3应出现在文件2之前

**时间尺度 `timescale**

`timescale命令用来说明跟在该命令后的模块的时间单位和时间精度。使用`timescale命令可以在同一个设计里包含采用了不同的时间单位的模块。用法：

`timescale<时间单位>/<时间精度>



1. *//模块中所有的时间值都表示是1ns的整数倍*
2. *//1ns/ps：1纳秒/脉冲*
3. `timescale  1ns/1ps

注意：如果在同一个设计里，多个模块中用到的时间单位不同，需要用到以下的时间结构：

* 用`timescale命令来声明本模块中所用到的时间单位和时间精度。
* 用系统任务$printtimescale来输出显示一个模块的时间单位和时间精度。
* 用系统函数 realtime及%t格式声明来输出显示EDA工具记录的时间信息。

**条件编译命令**

`ifdef、`else、`endif

这与C语言用法类似，这里就不赘述了。

**总结一下**

Verilog HDL的语法与C语言的语法类似，但是一定要意识到Verilog HDL描述的是电路，光有代码还不够，器件可能运行的结果并不是代码想要的效果。另外要注意理解并行的概念，这里的并行是硬件在时钟驱动真的同时按照所设计的逻辑运行。一些重要的概念：

* 阻塞〔Blocking〕和非阻塞〔Non-Blocking〕赋值的不同
* 顺序块和并行块的不同
* 块与块之间的并行执行的概念；
* task和function的概念。

那么最好的学习办法是什么呢？写代码、仿真、综合、优化布局布线，挖坑、踩坑、填坑，在错误中总结，渐进明晰、不断实践总结